CS103
Winter 2026

Lecture 16:

Finite Automata

Part 3 of 3

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

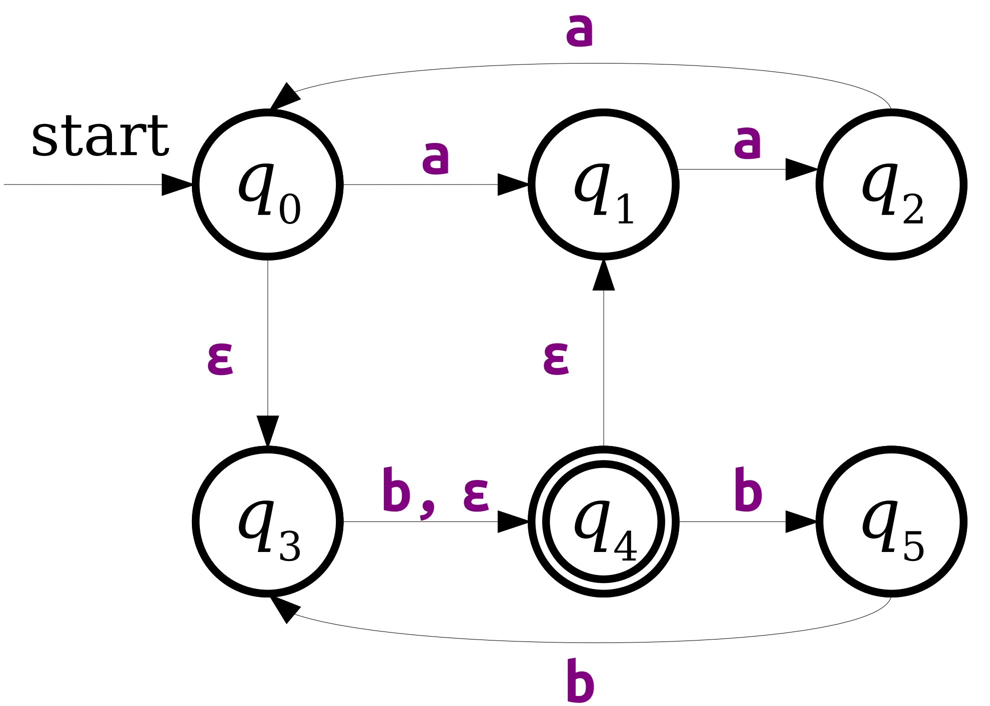
Finite Automata

Part 3

1. **Recap from Last Time**
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

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 ε-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.

Finite Automata

Part 3

1. **Recap from Last Time**
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What's Next?

Finite Automata

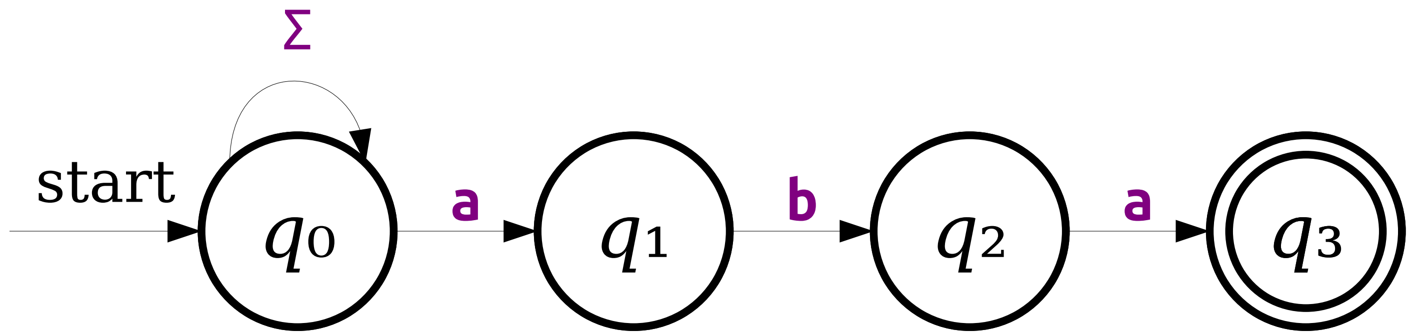
Part 3

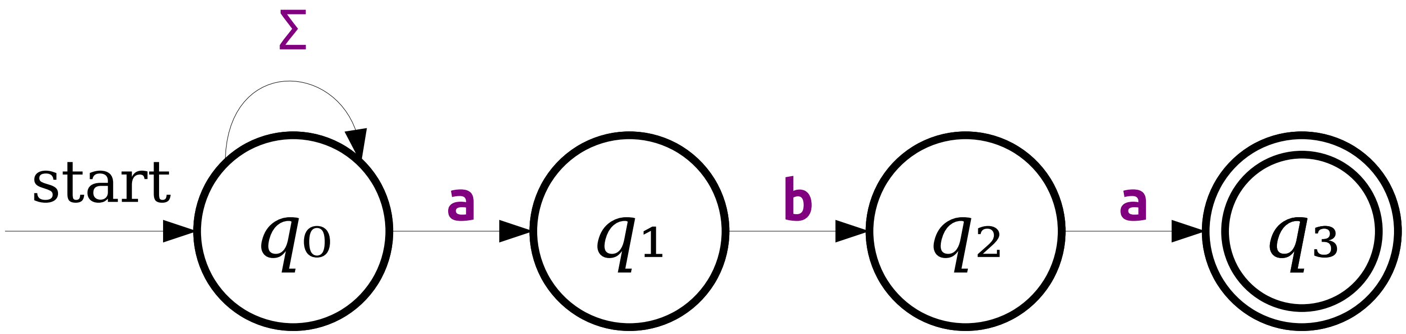
NFAs and DFAs

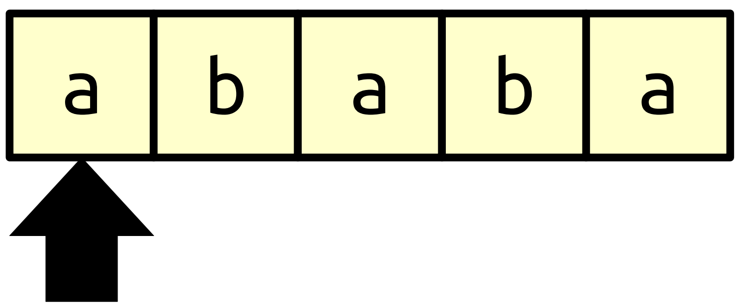
- Any language that can be accepted by a DFA can be accepted by an NFA.
- Why?
  - Every DFA essentially already *is* an NFA!
- ***Question:*** Can any language accepted by an NFA also be accepted by a DFA?
- Surprisingly, the answer is ***yes***!

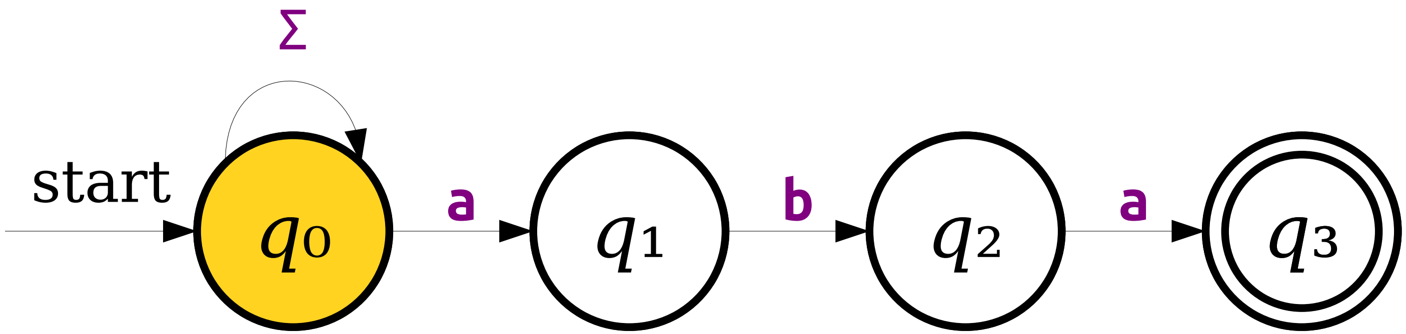
Thought Experiment:

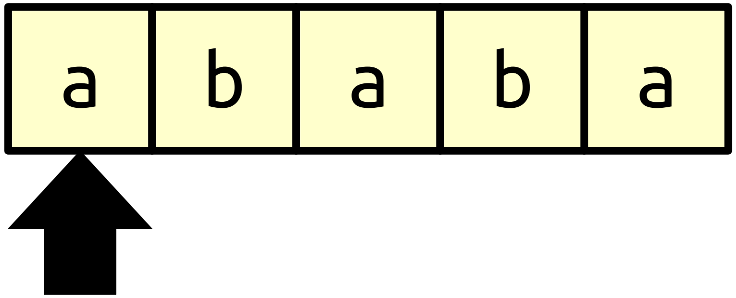
How would you simulate an NFA in software?

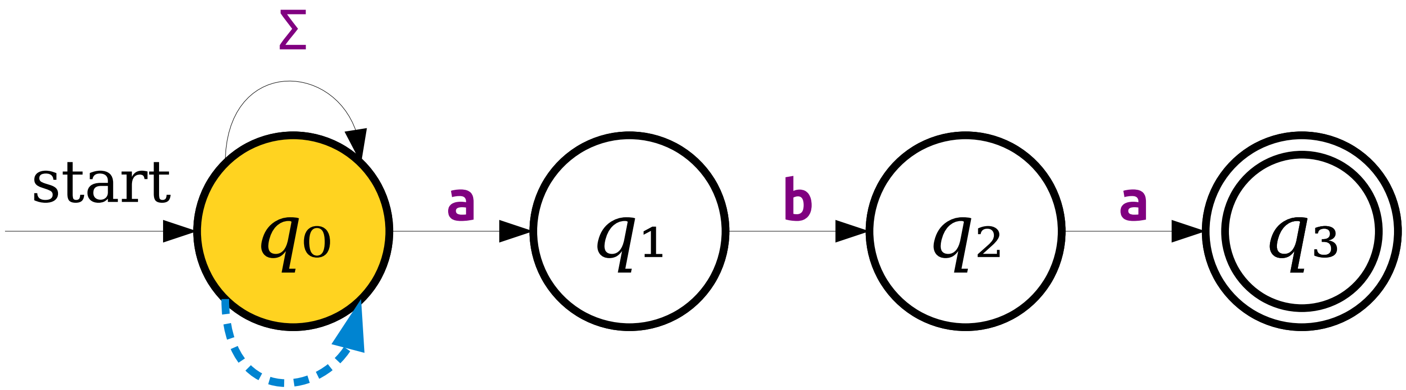

Σ
start
q0
a
q1
b
q2
a
q3

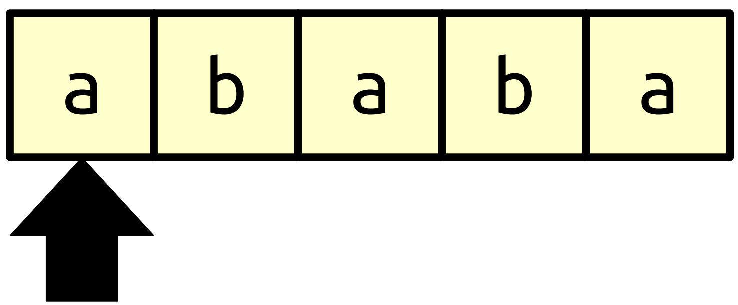

Σ
start
q0
a
q1
b
q2
a
q3

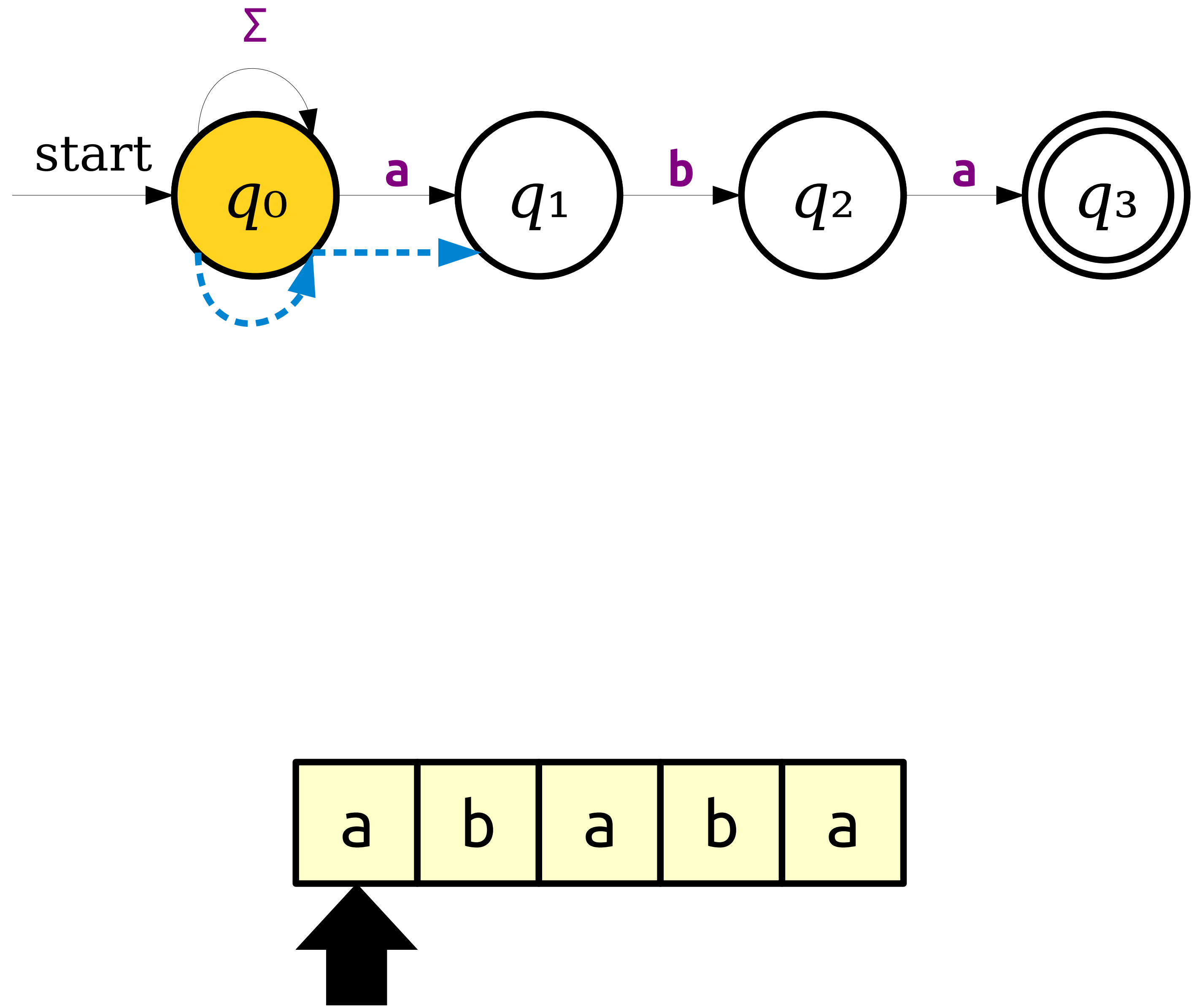

a b a b a

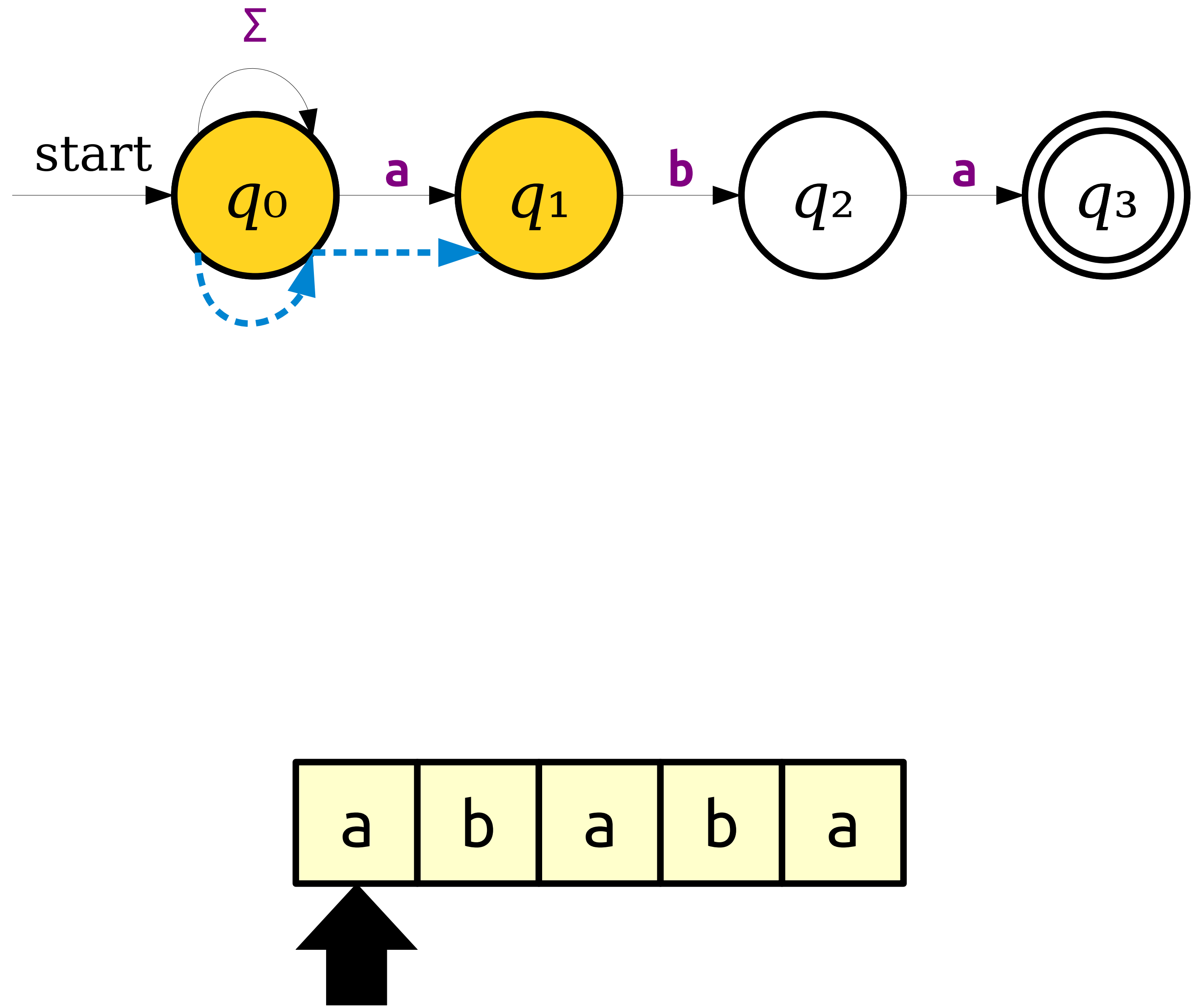

Σ
start
q0
a
q1
b
q2
a
q3


a b a b a

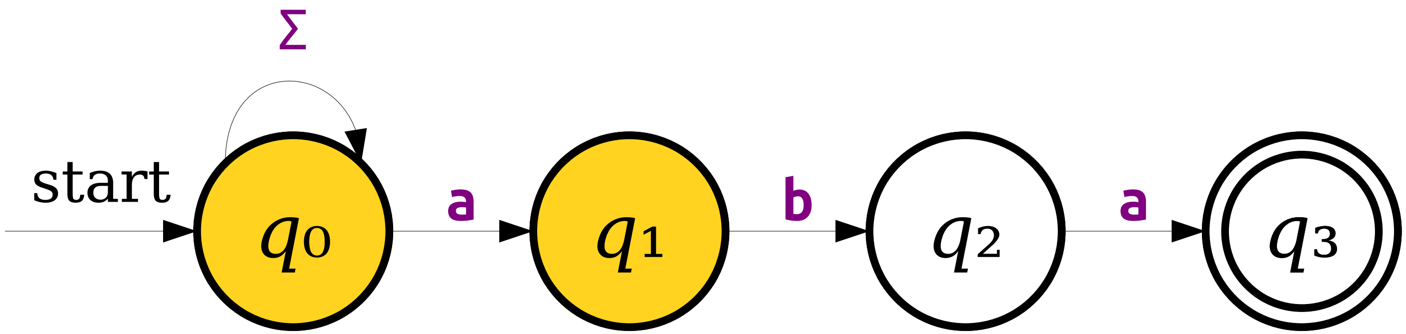

Σ
start
q_0
a
q_1
b
q_2
a
q_3

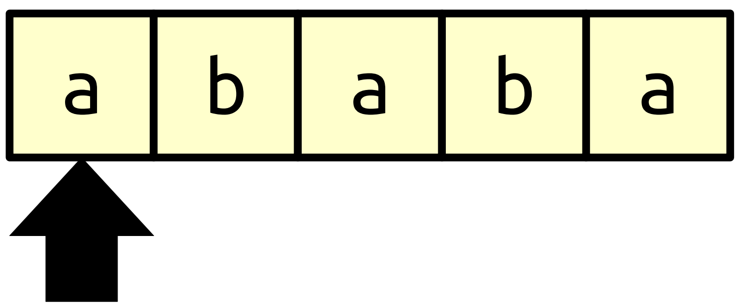

a
b
a
b
a

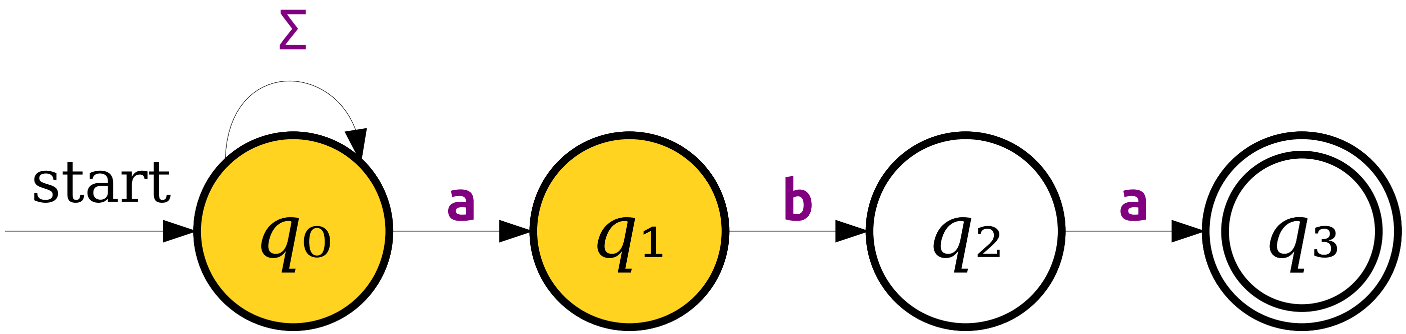

Σ
start
q_0
a
q_1
b
q_2
a
q_3
a
b
a
b
a

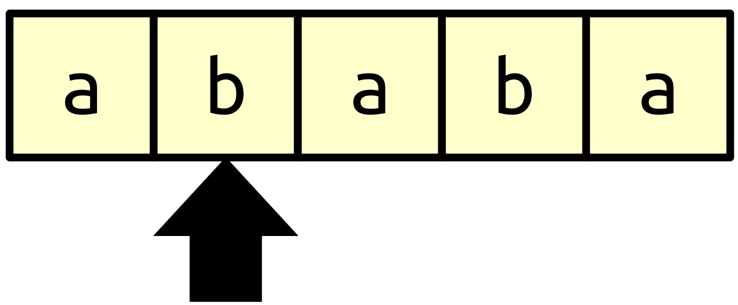


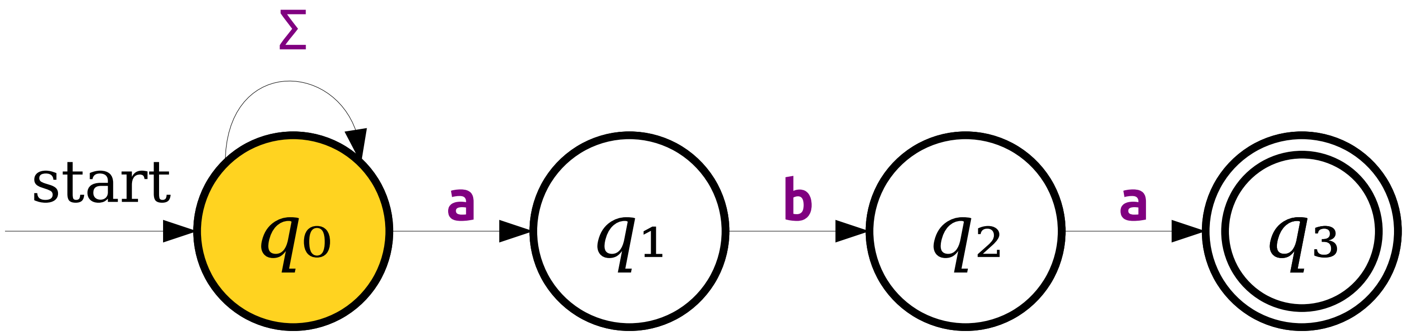
Σ
start
q0
a
q1
b
q2
a
q3
a
b
a
b
a

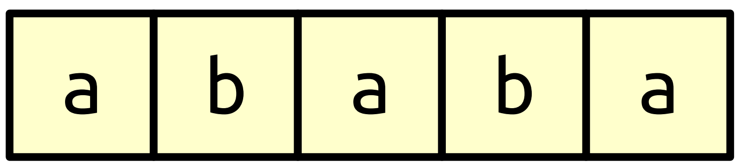

Σ
start
q0
a
q1
b
q2
a
q3

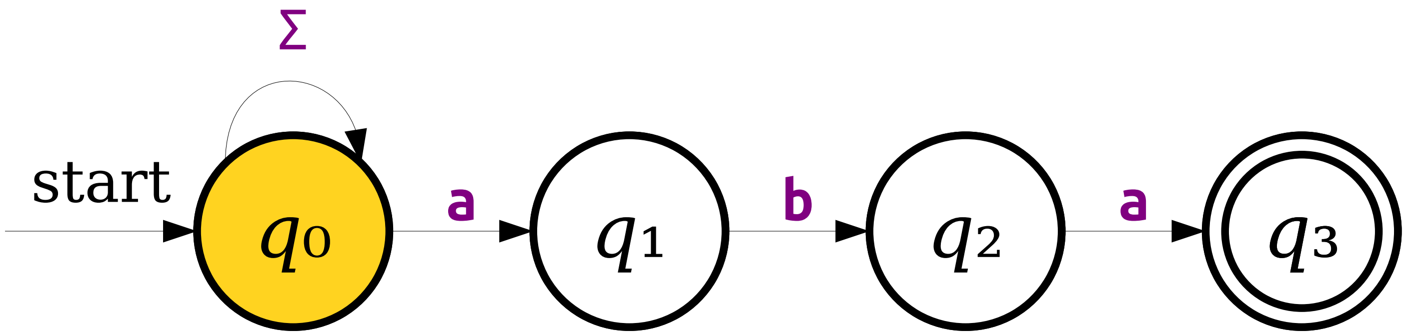

a
b
a
b
a

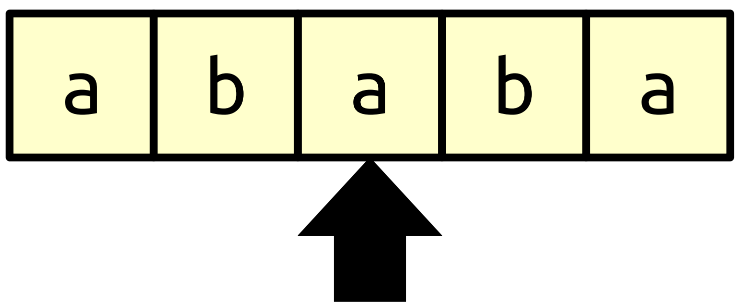

Σ
start
q_0
a
q_1
b
q_2
a
q_3

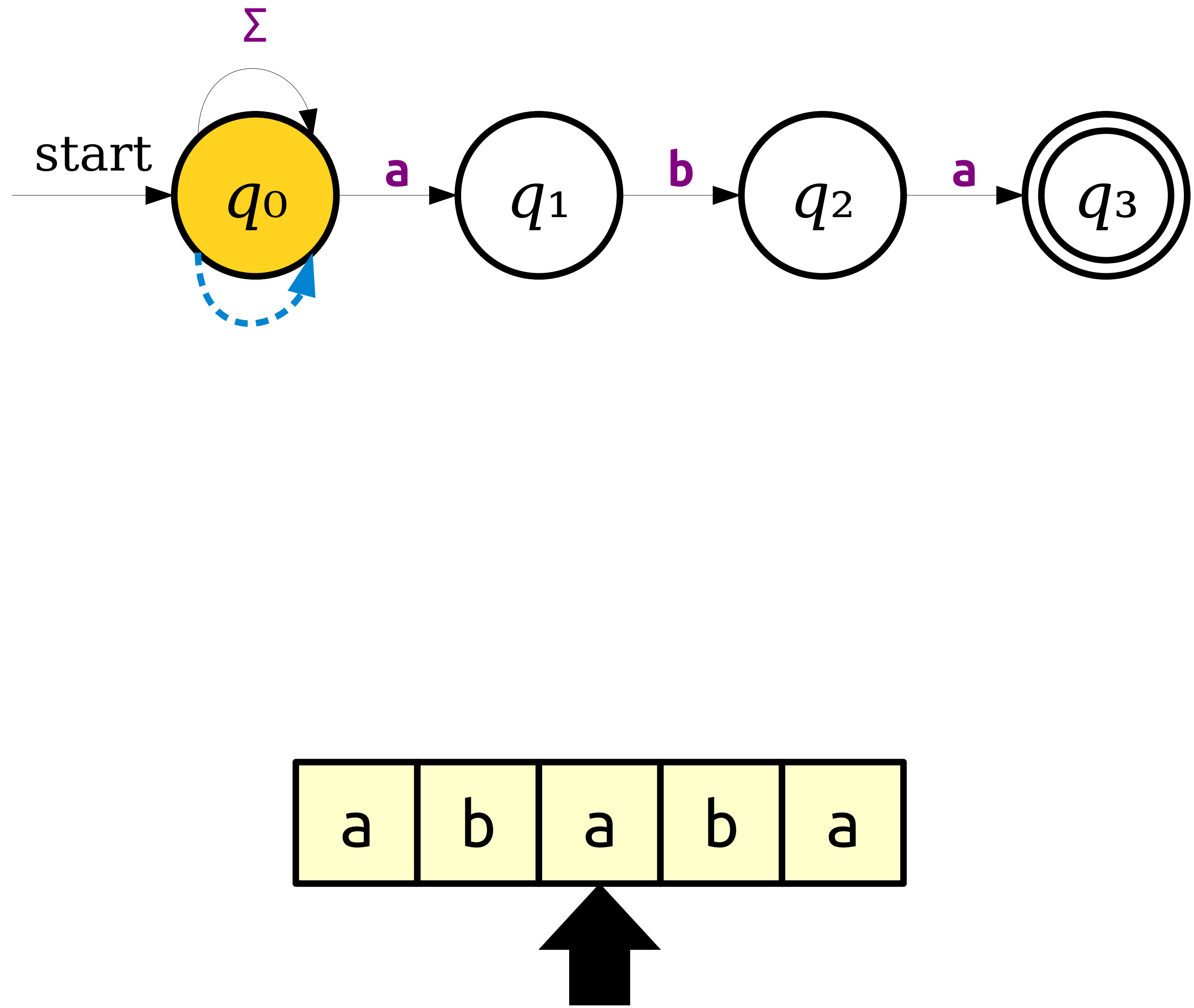

a
b
a
b
a

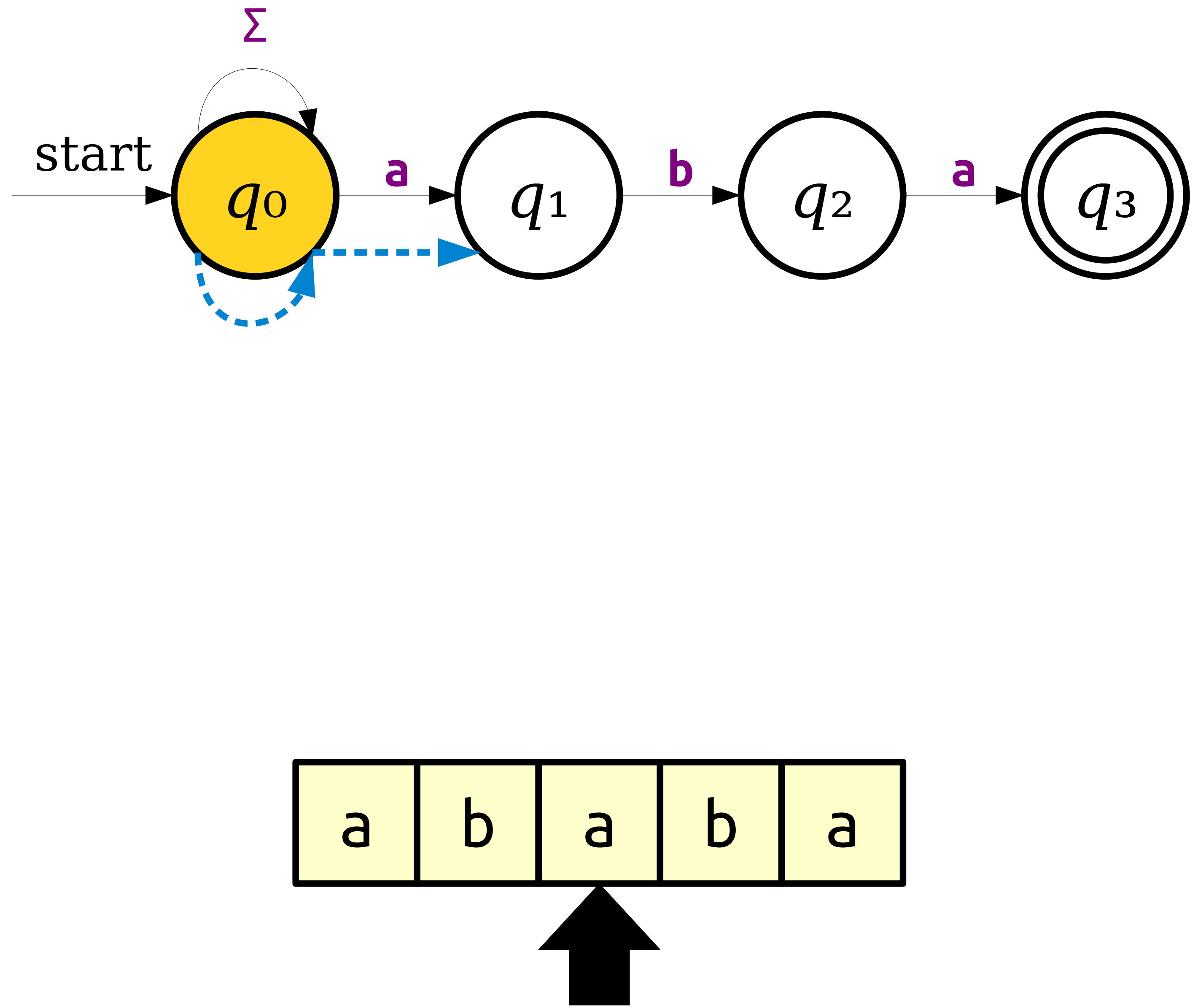

Σ
start
q0
a
q1
b
q2
a
q3


a
b
a
b
a

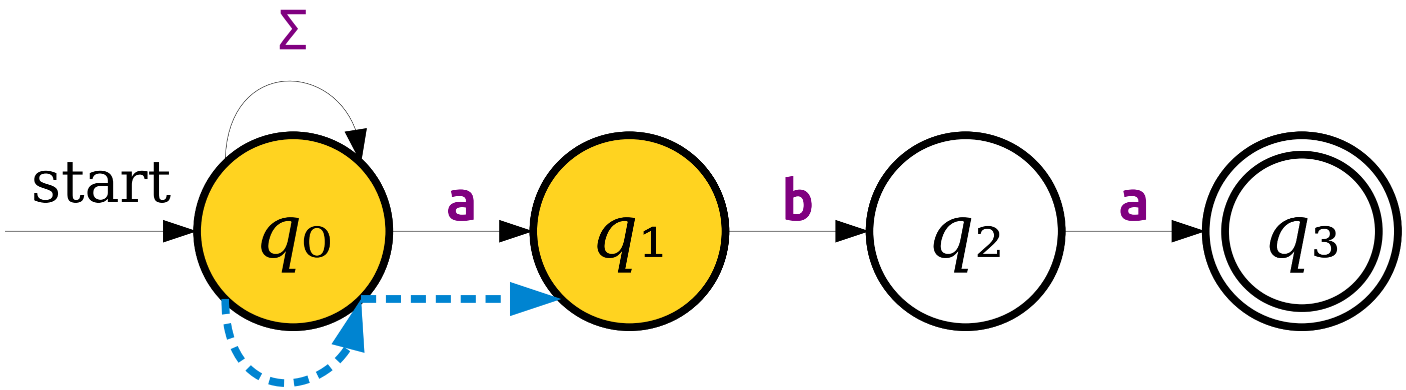

Σ
start
q0
a
q1
b
q2
a
q3

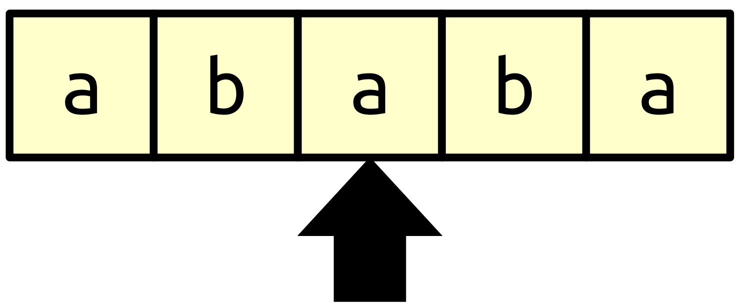

a
b
a
b
a

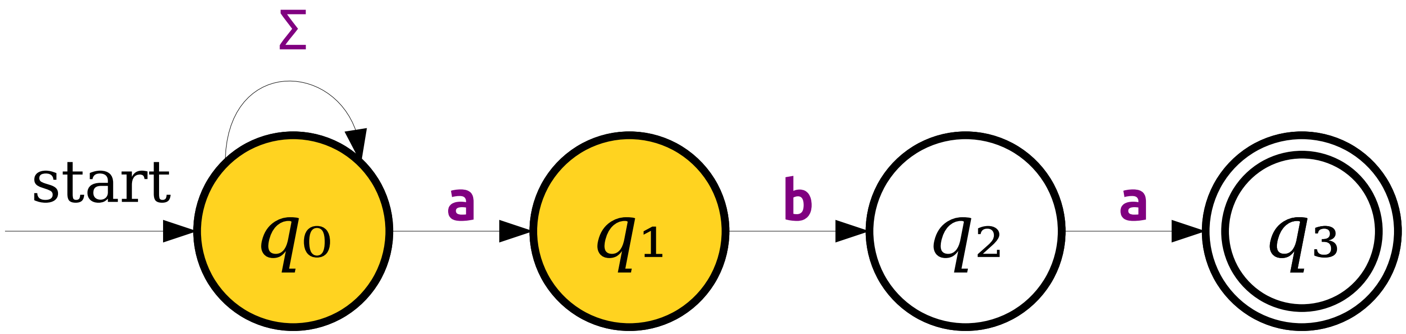

Σ
start
q0
a
q1
b
q2
a
q3
a
b
a
b
a

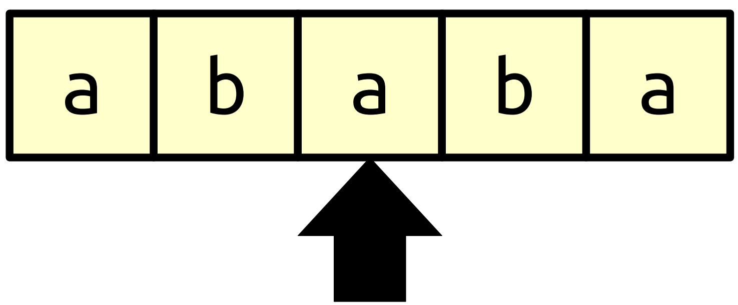


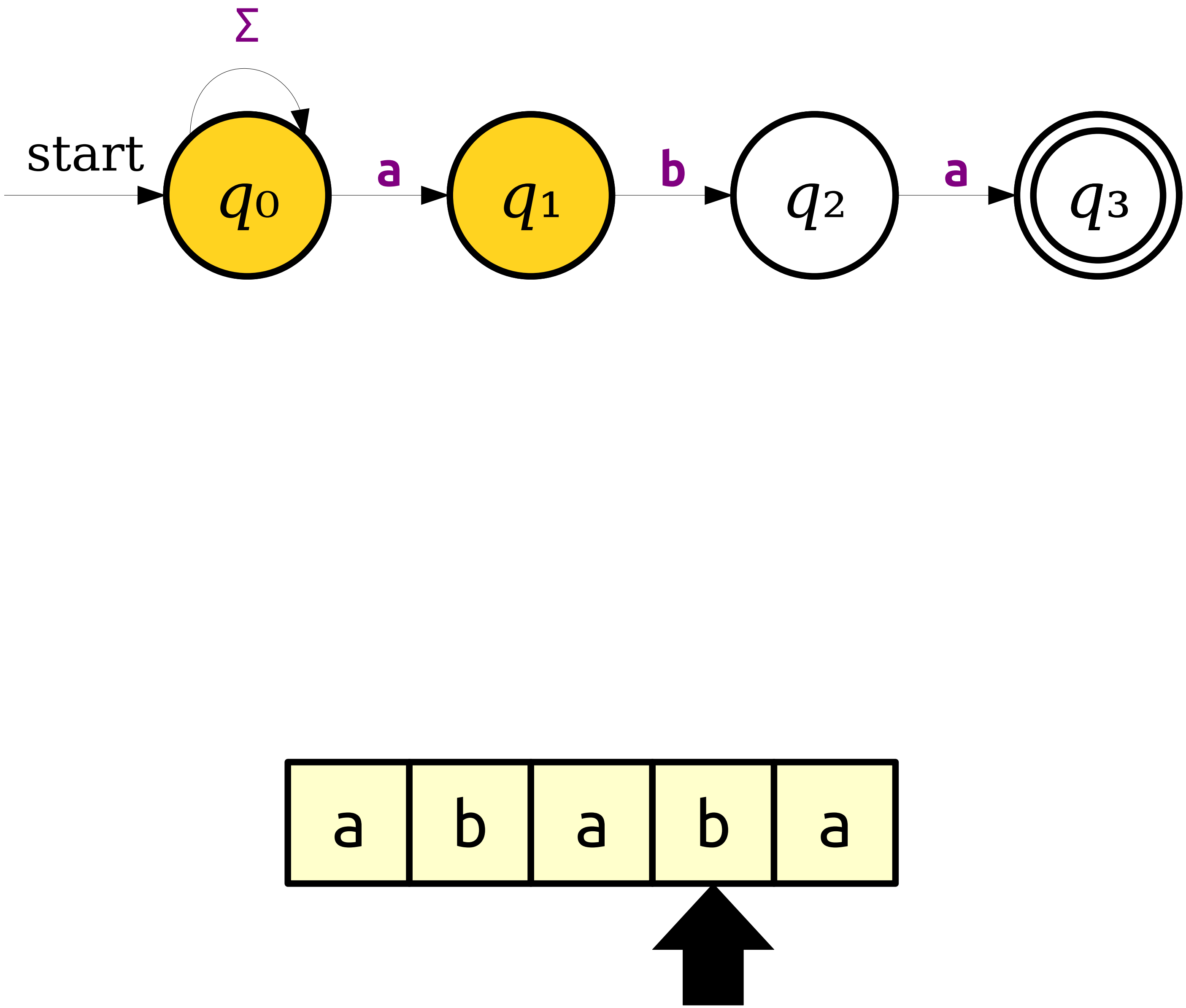
Σ
start
q_0
a
q_1
b
q_2
a
q_3
a
b
a
b
a

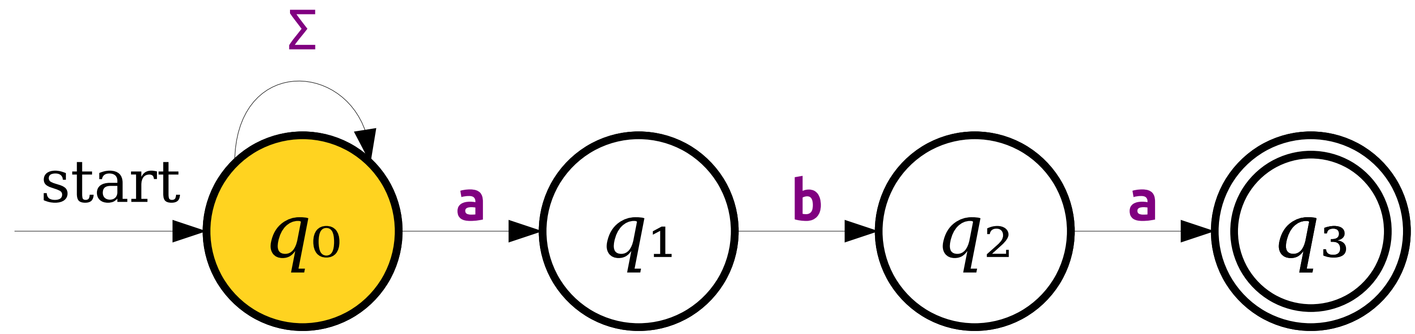

Σ
start
q0
a
q1
b
q2
a
q3

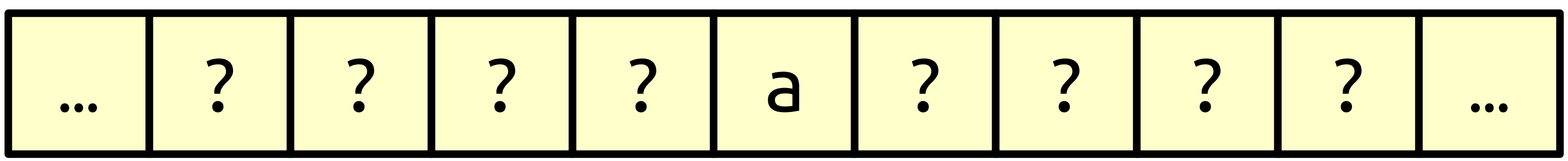

a
b
a
b
a

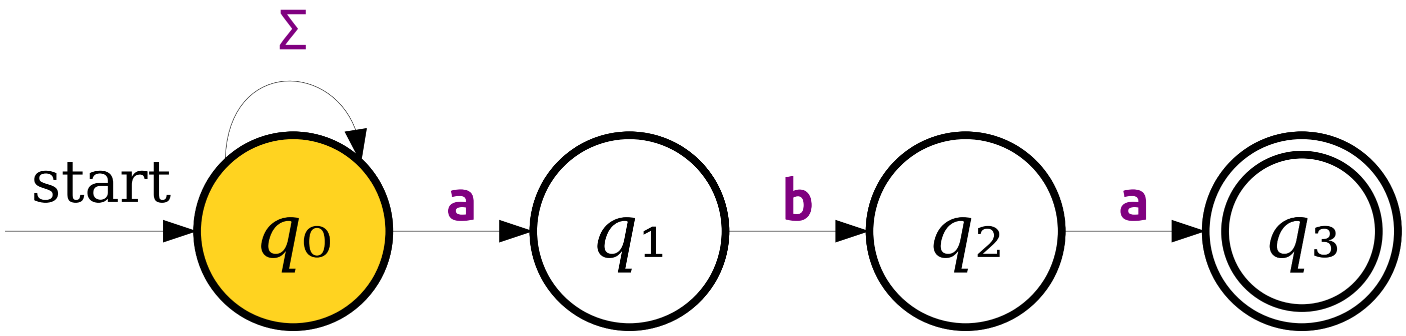

Σ
start
a
b
a
q_0
q_1
q_2
q_3

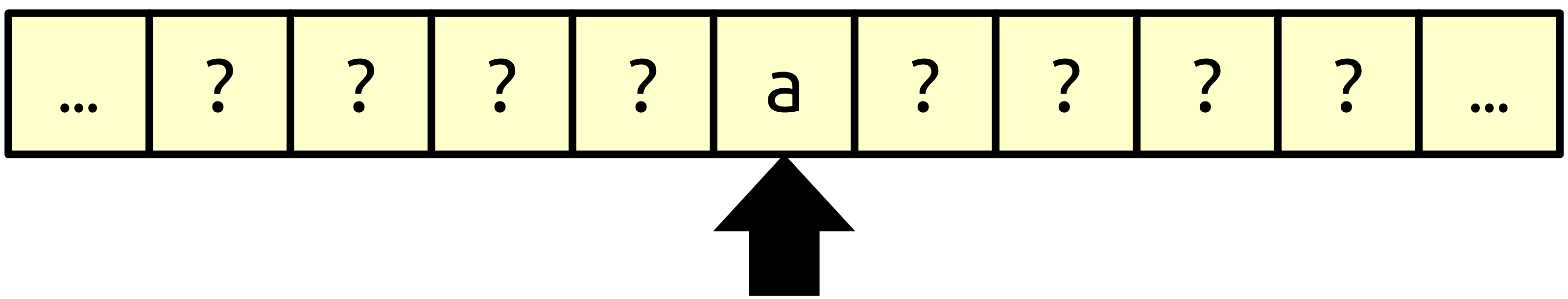

a
b
a
b
a

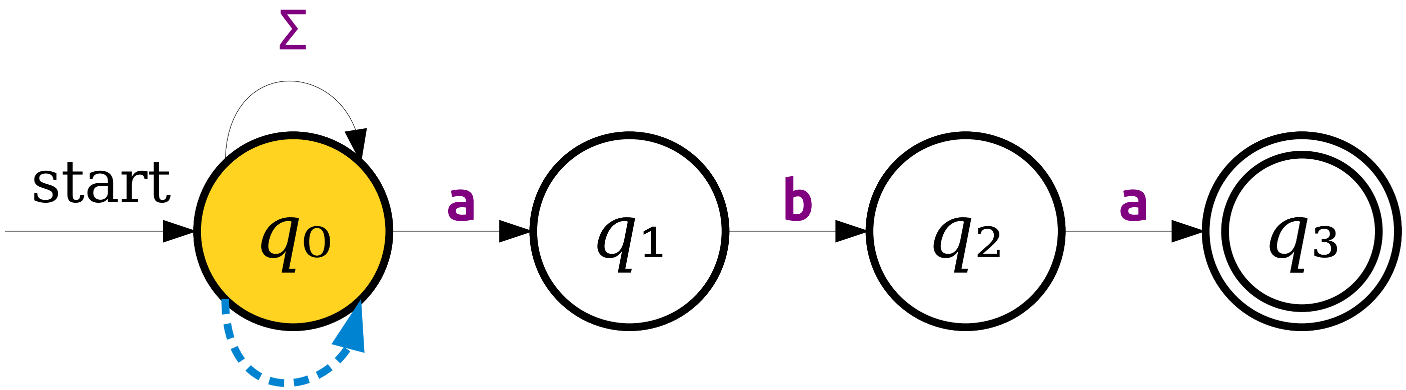

Σ
start
q_0
a
q_1
b
q_2
a
q_3
a
b
a
b
a

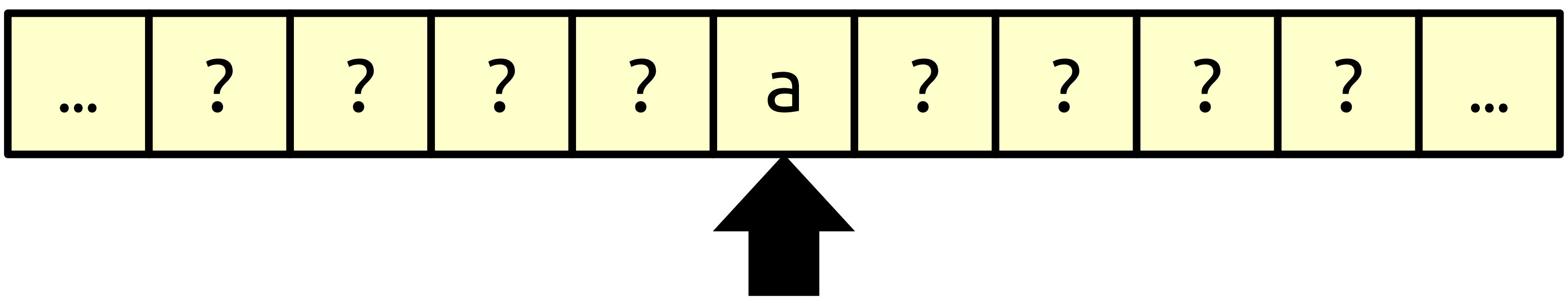

Σ
start
q_0
a
q_1
b
q_2
a
q_3

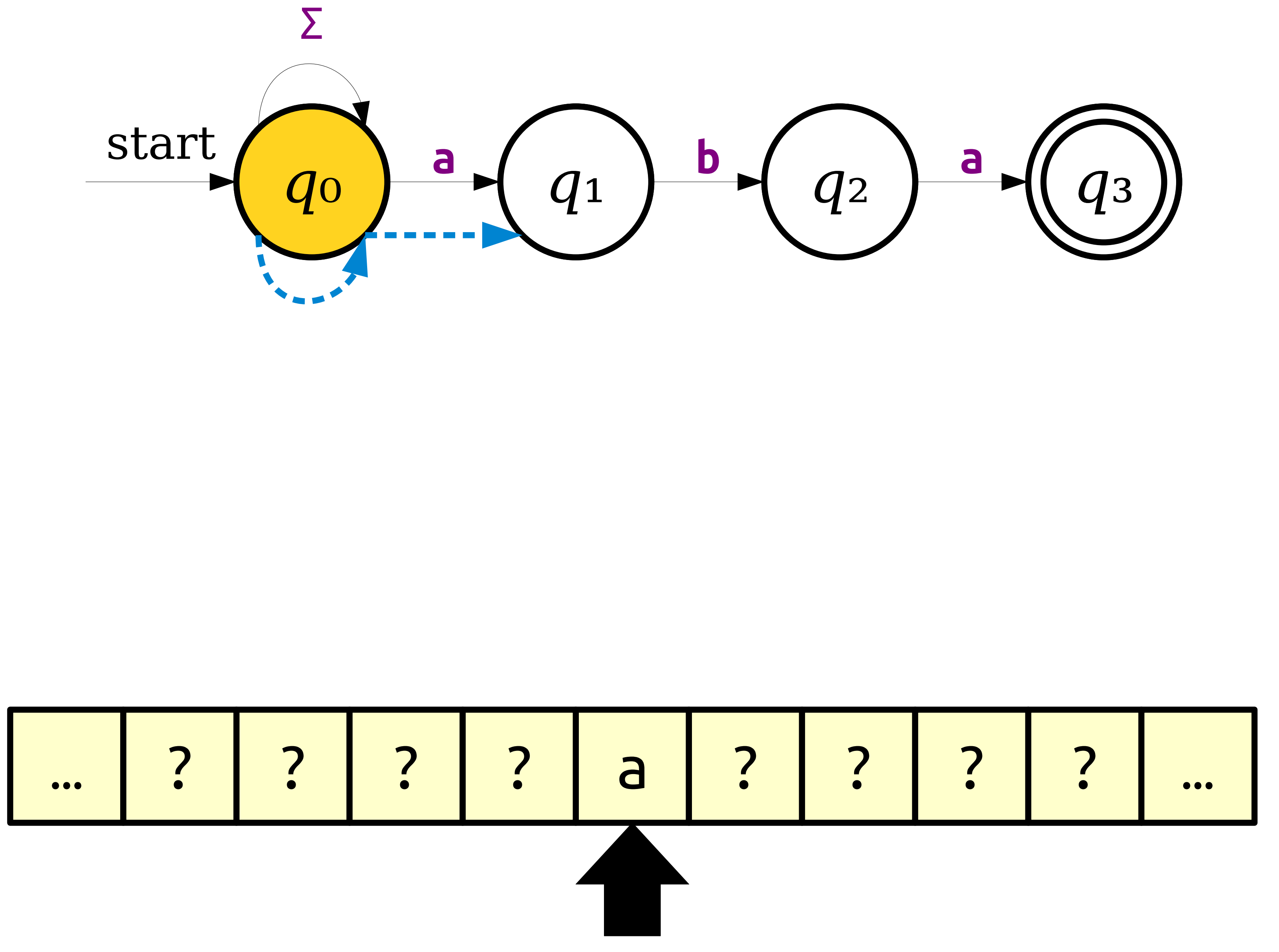

...
?
?
?
?
a
?
?
?
?
...


Σ
start
q_0
a
q_1
b
q_2
a
q_3

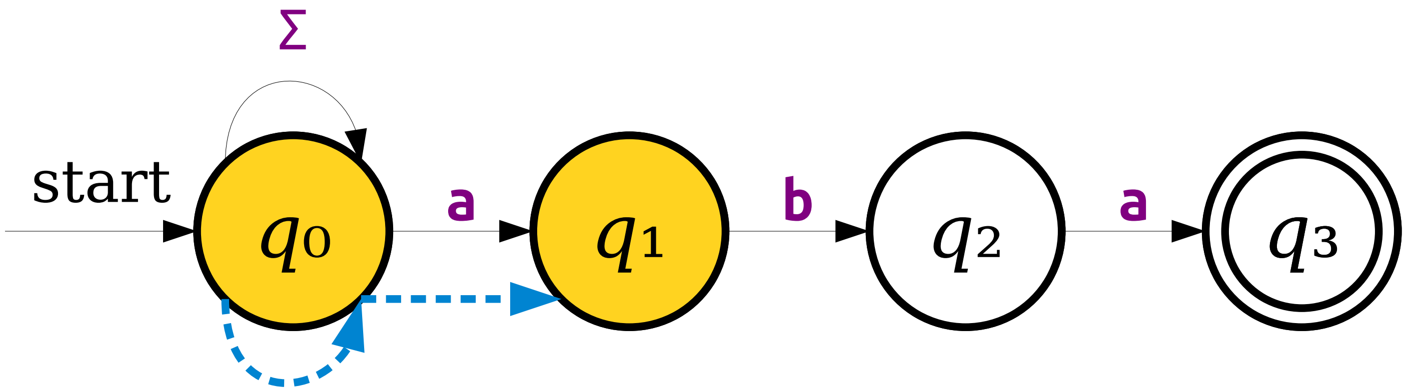

… ? ? ? ? a ? ? ? ? …

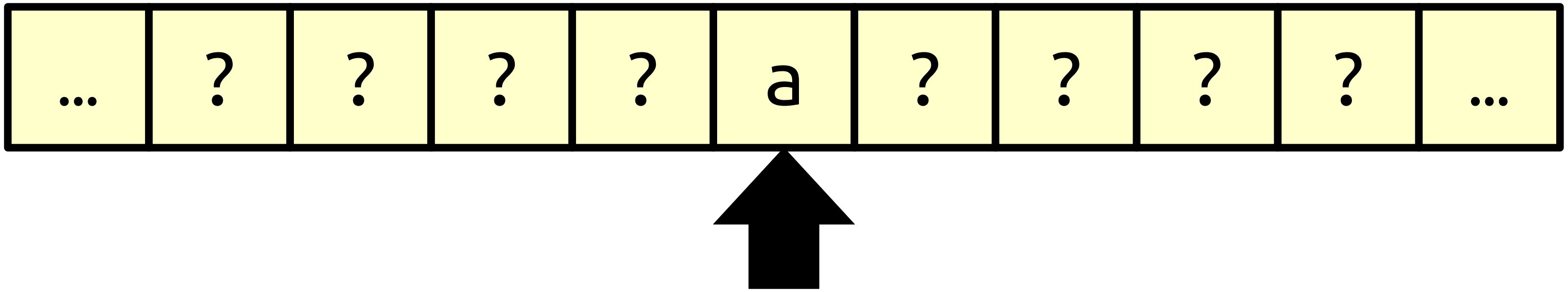

Σ
start
q_0
a
q_1
b
q_2
a
q_3

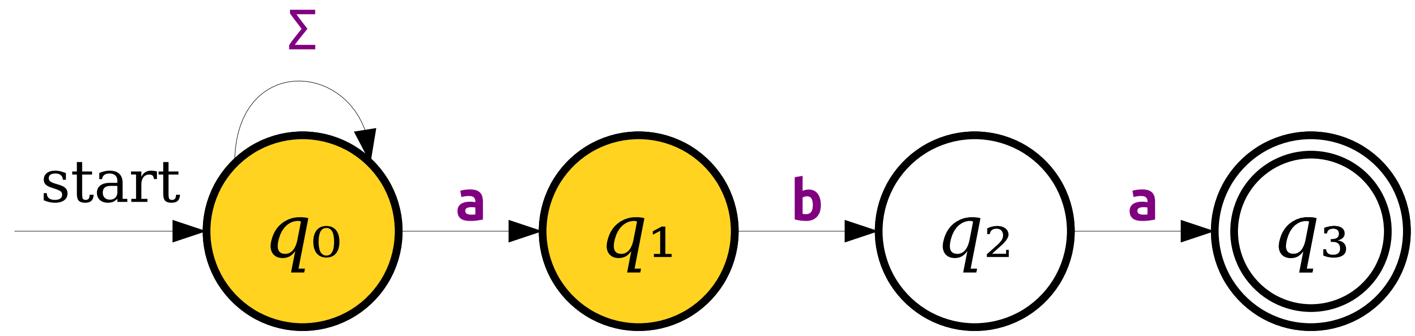

...
?
?
?
?
a
?
?
?
?
...

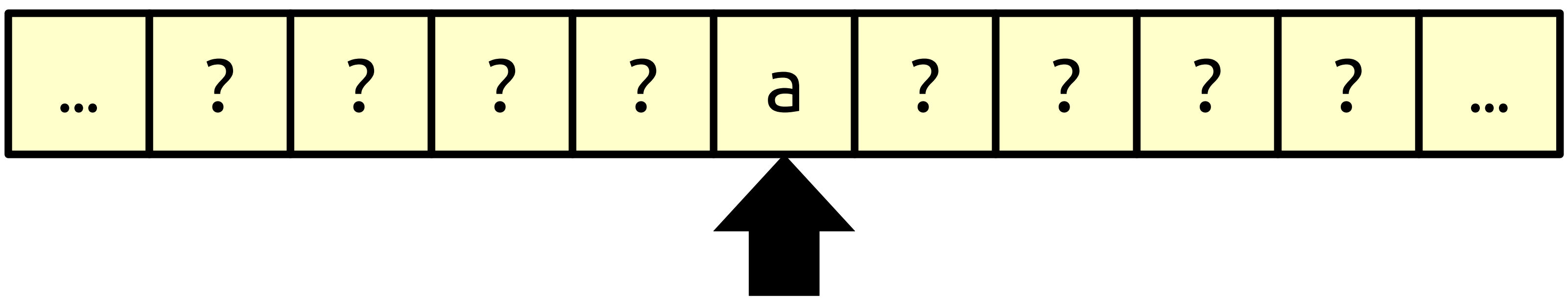


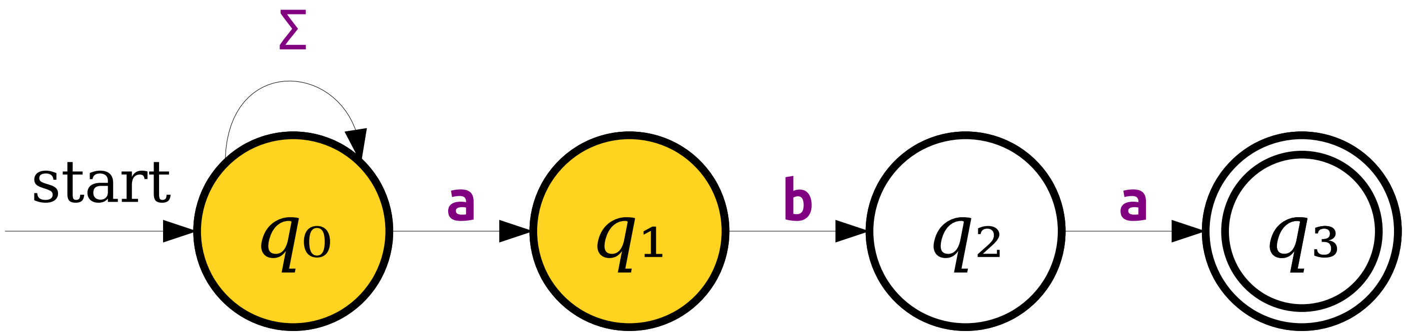
Σ
start
q0
a
q1
b
q2
a
q3
...
?
?
?
?
a
?
?
?
?
...

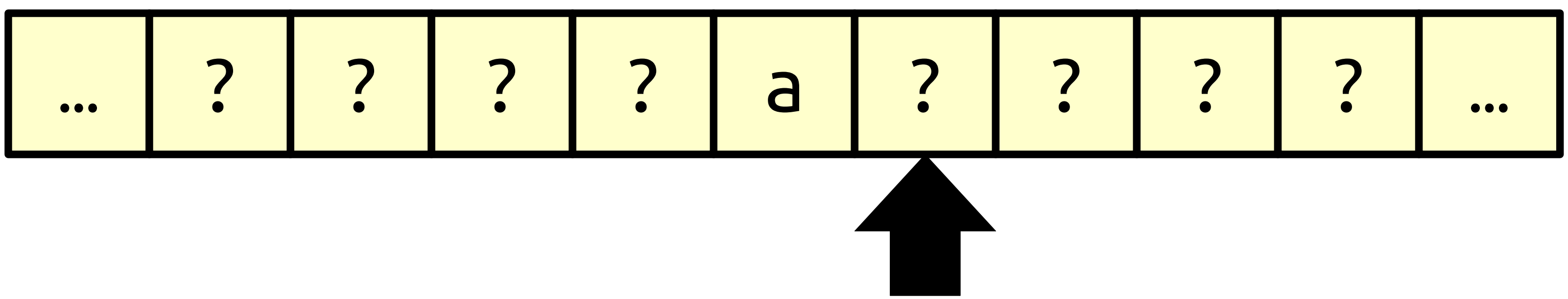

Σ
start
q0
a
q1
b
q2
a
q3


…
?
?
?
?
a
?
?
?
?
…


Σ
start
q_0
a
q_1
b
q_2
a
q_3


...
?
?
?
?
a
?
?
?
?
...


Σ
start
q_0
a
q_1
b
q_2
a
q_3


...
?
?
?
?
a
?
?
?
?
...

Finite Automata

Part 3

1. Recap from Last Time
2. **How Powerful Are NFAs?**
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

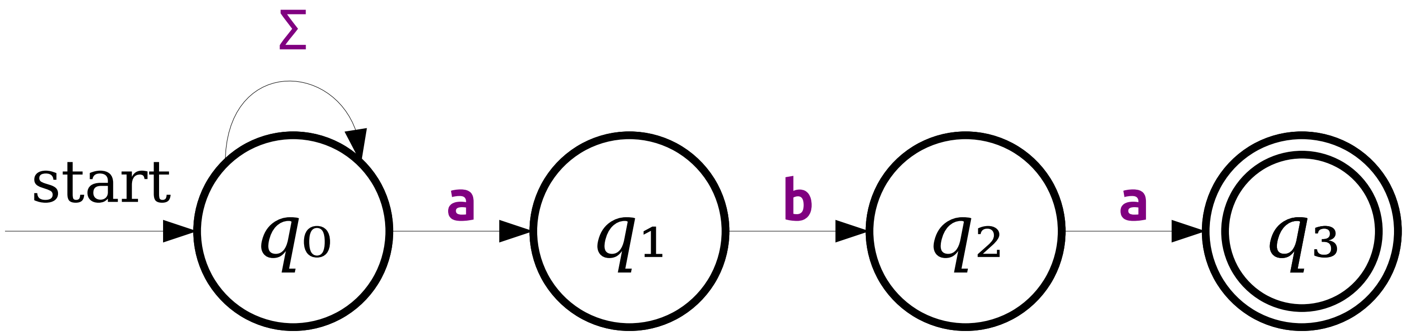
Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

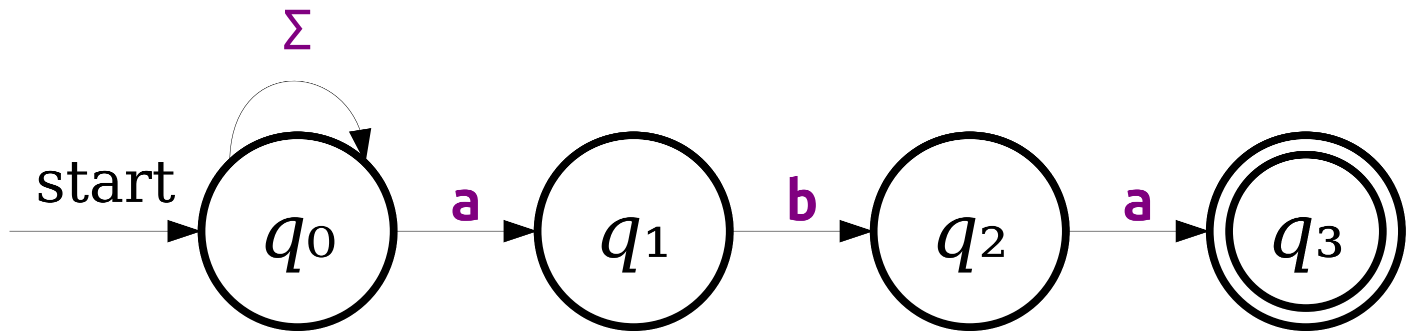
Finite Automata

Part 3

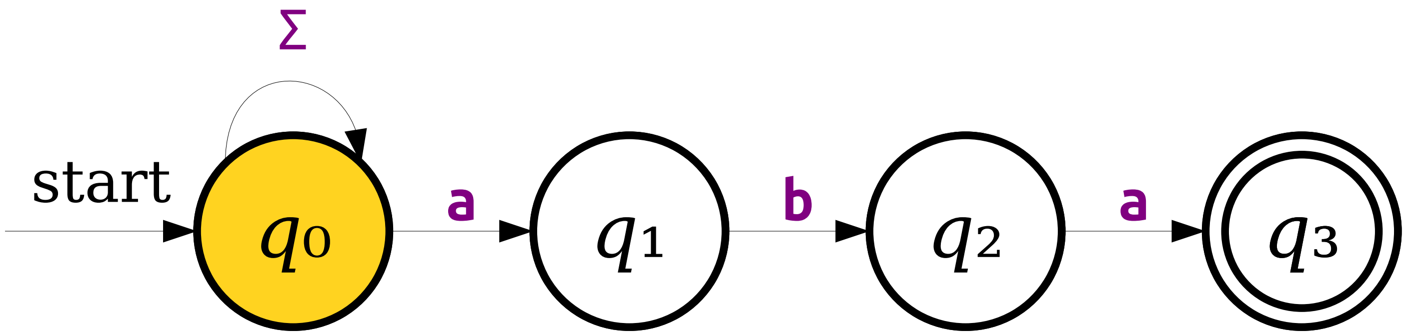
1. Recap from Last Time
2. How Powerful Are NFAs?
3. **The Subset Construction**
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?



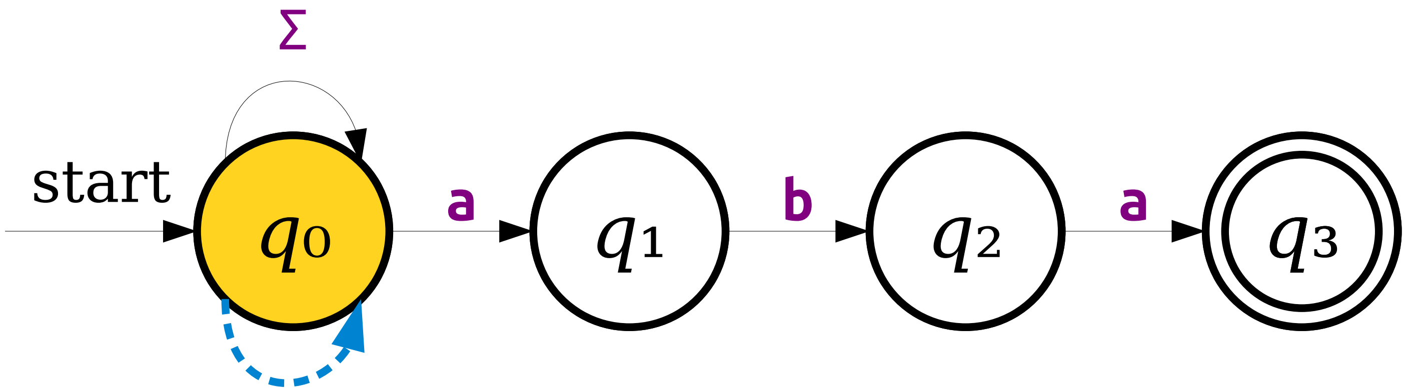
	a
$\{q_0\}$	$\{q_0, q_1\}$



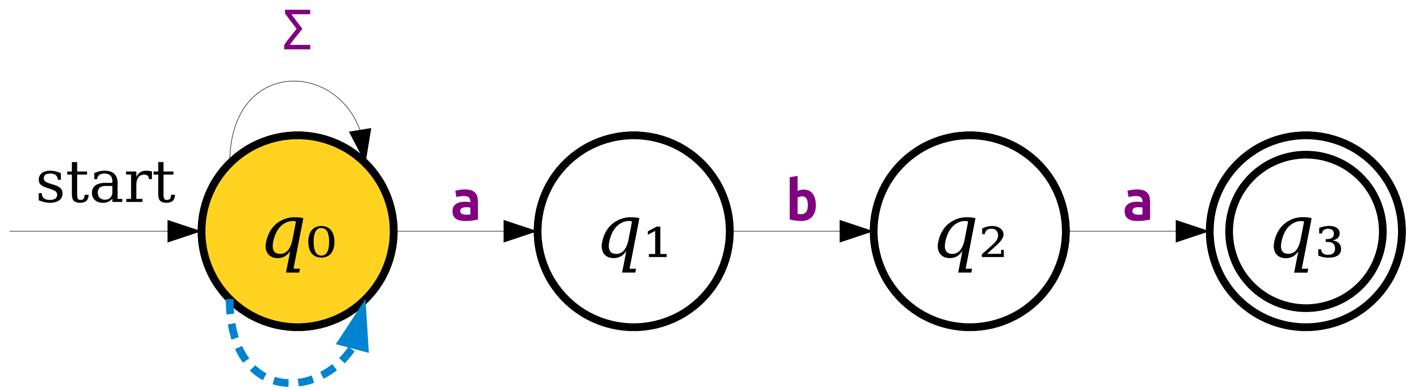
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	



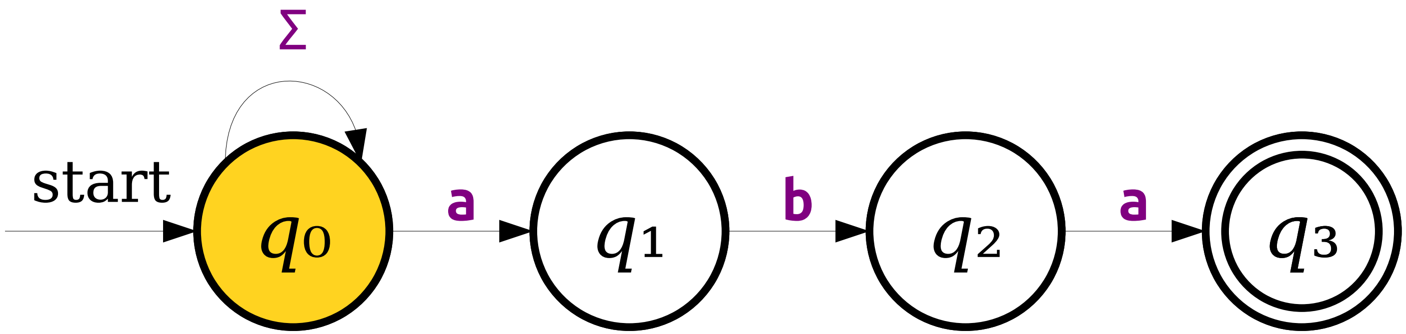
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	



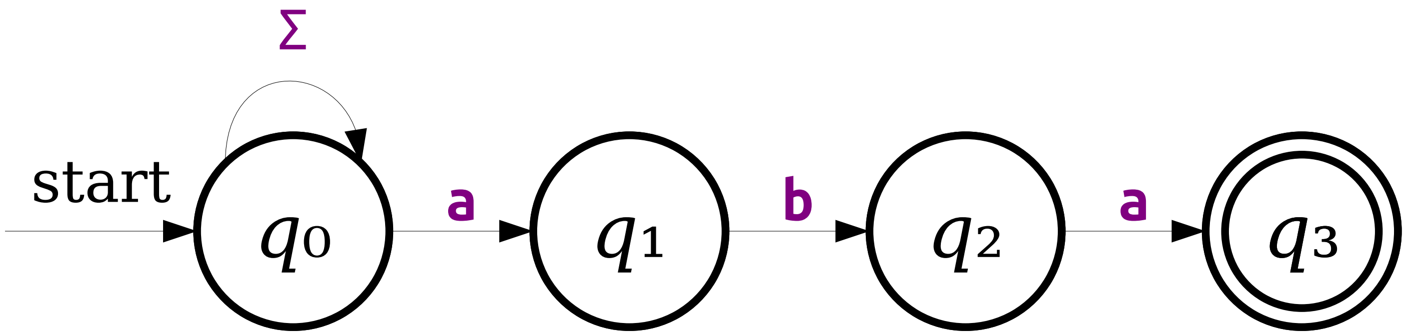
	a	*b*
$\{q_0\}$	$\{q_0, q_1\}$	



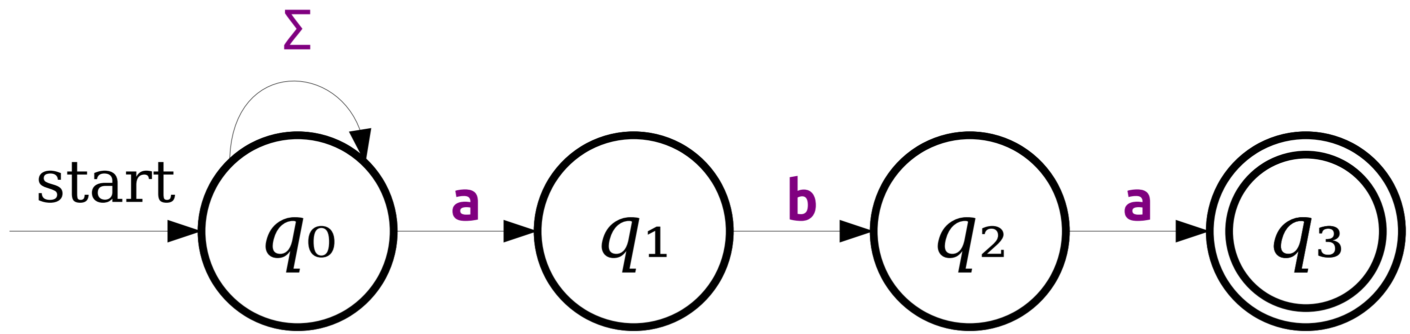
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



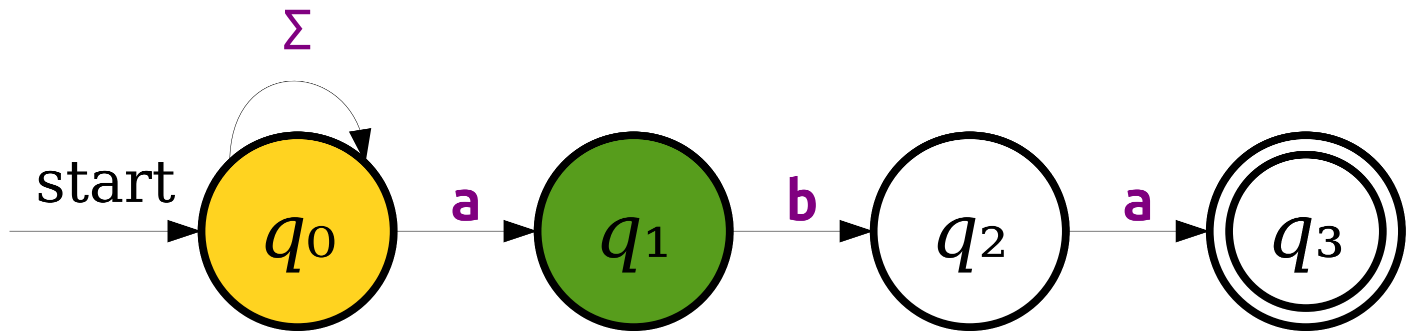
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



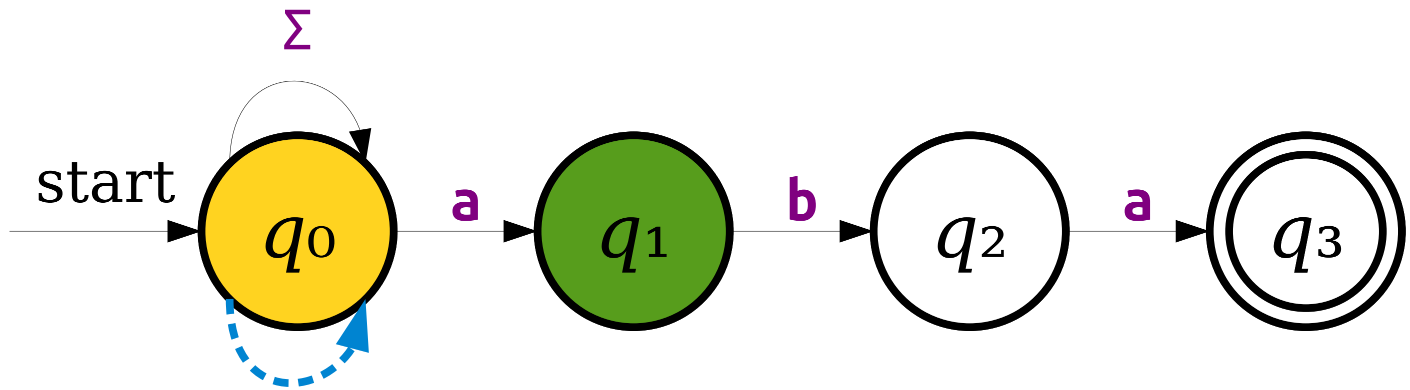
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$



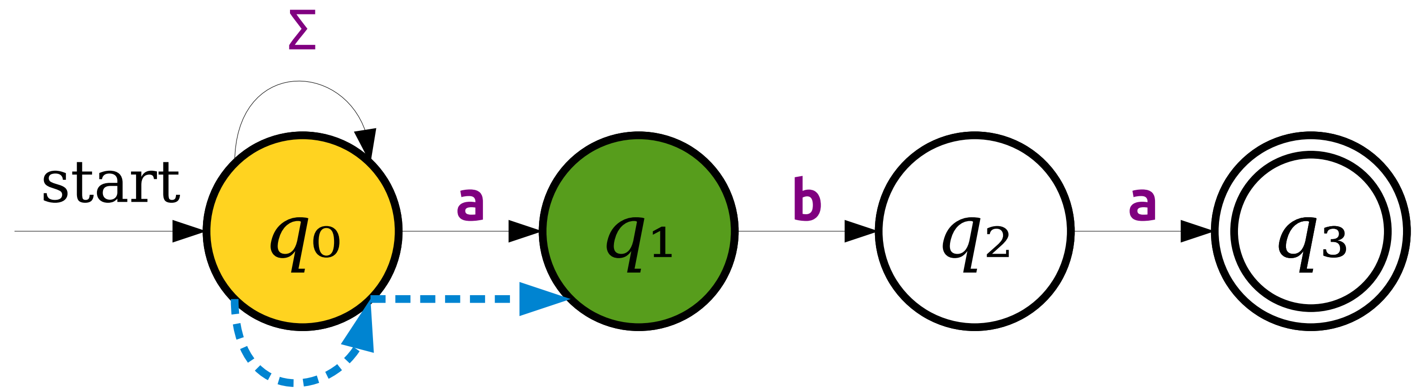
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



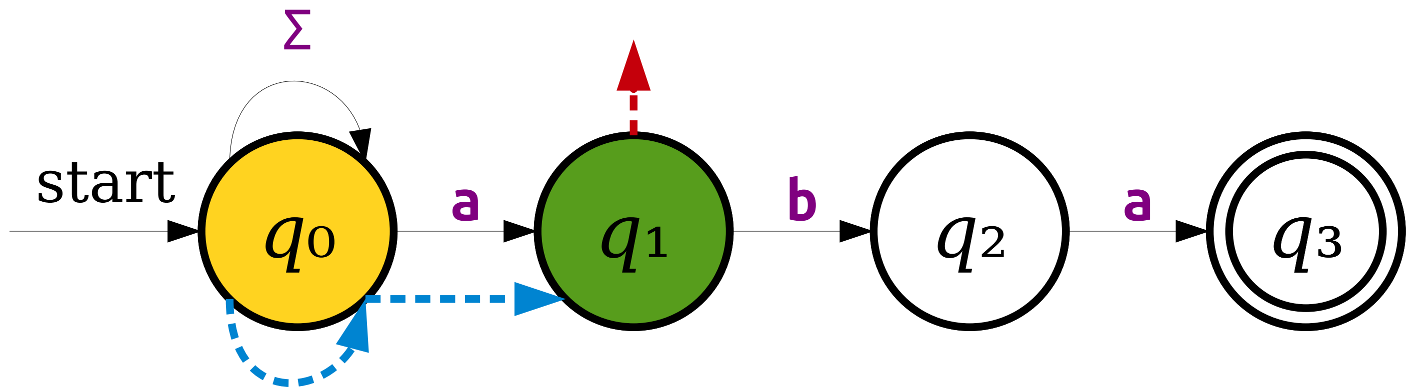
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



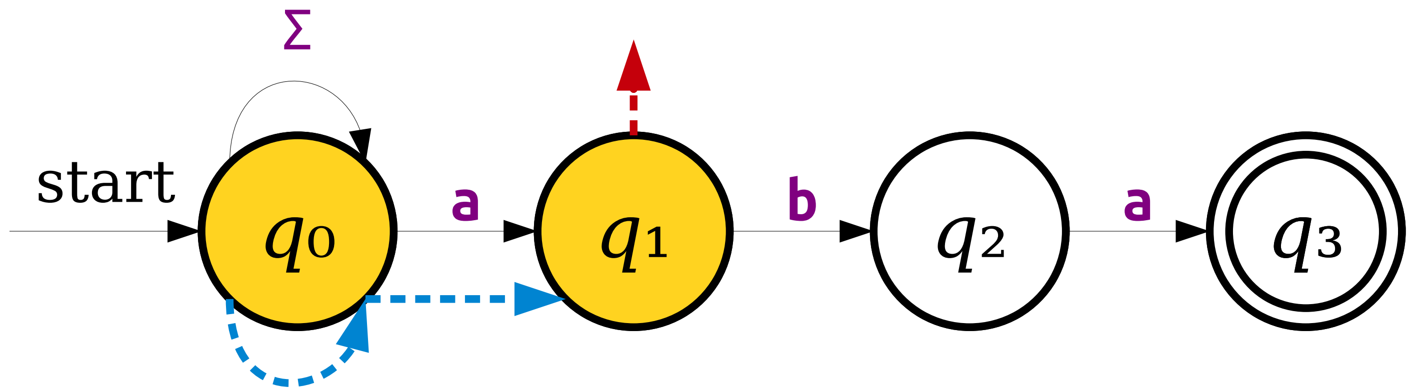
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



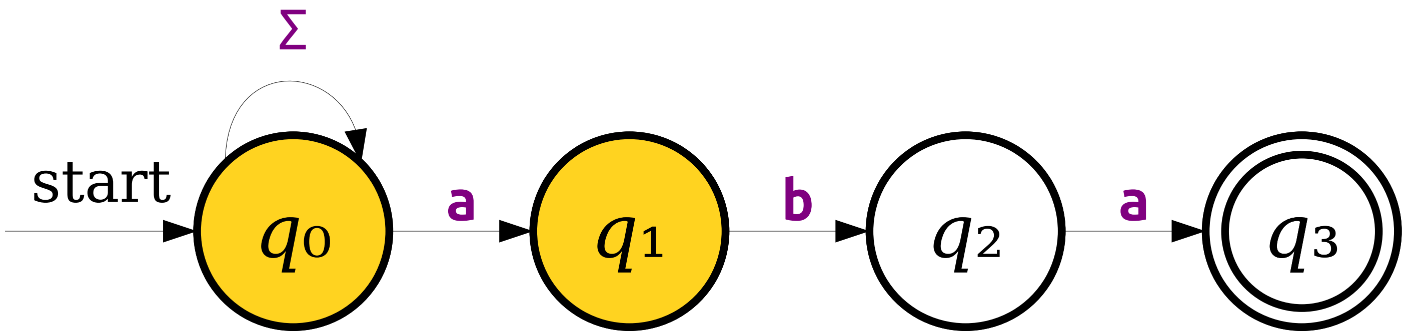
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



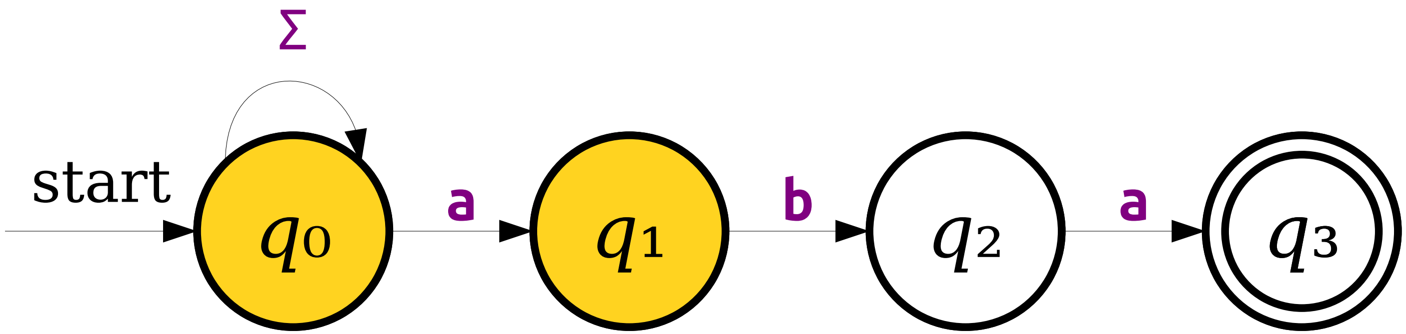
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



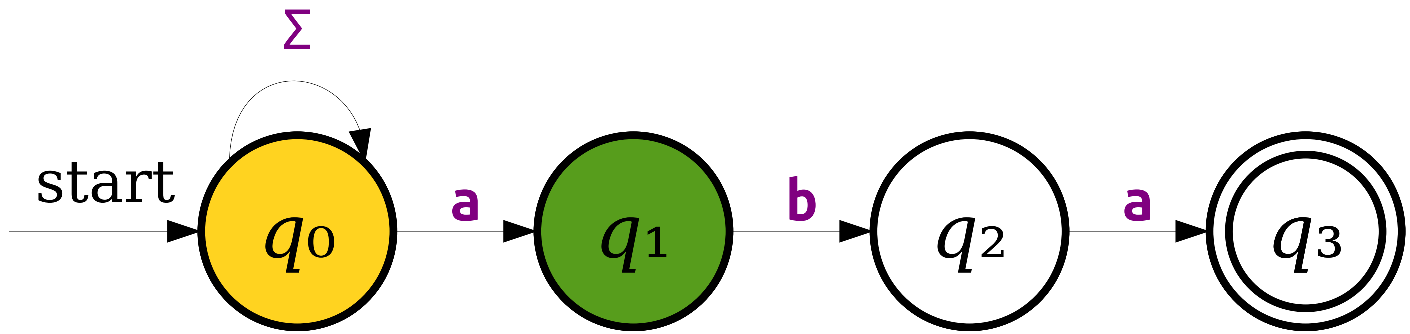
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



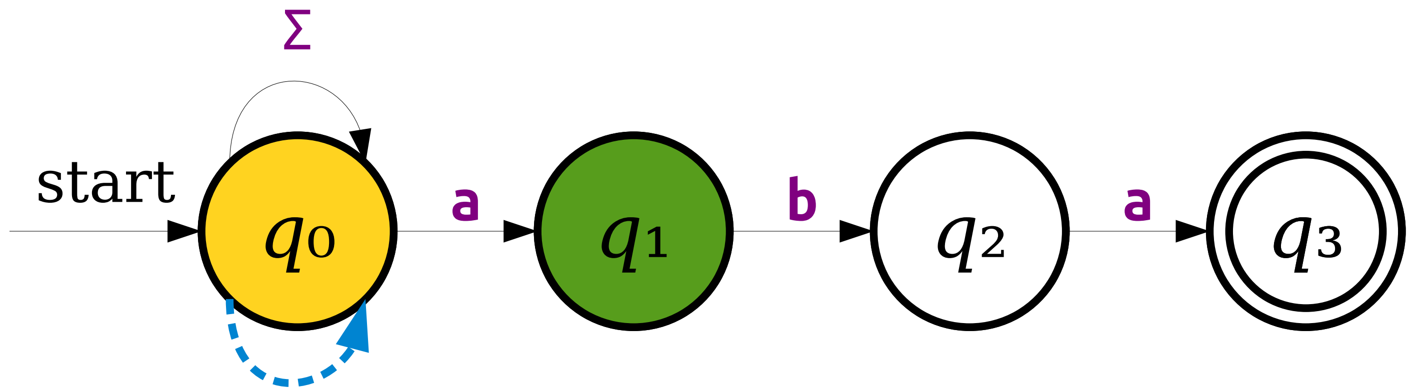
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$		



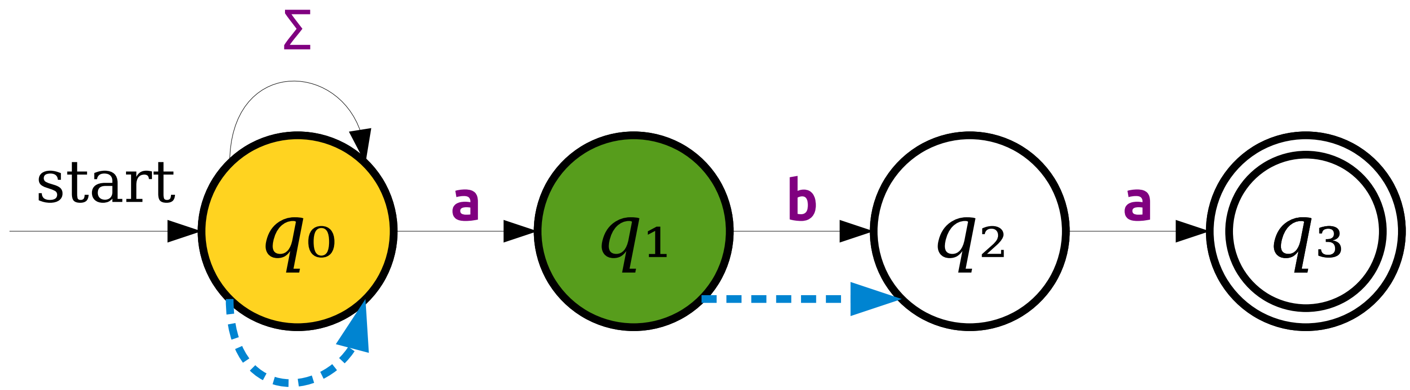
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



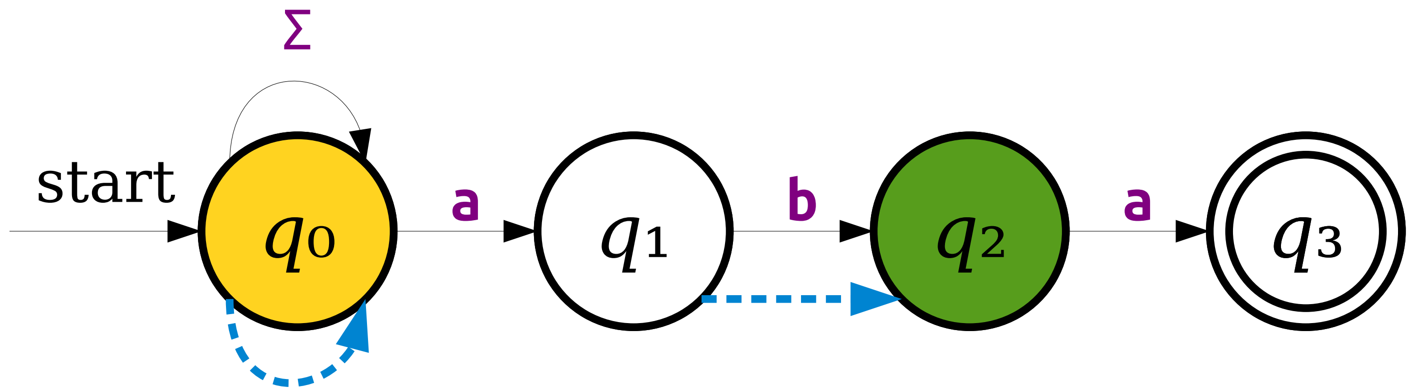
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



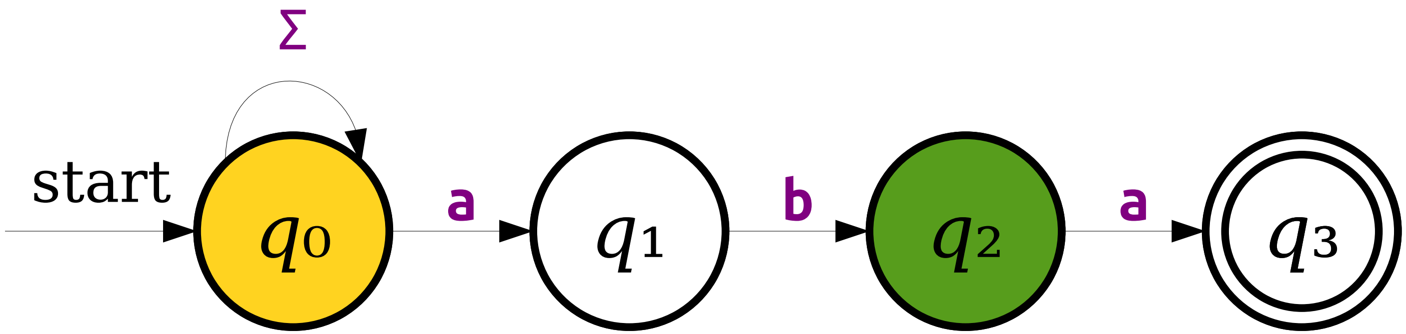
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



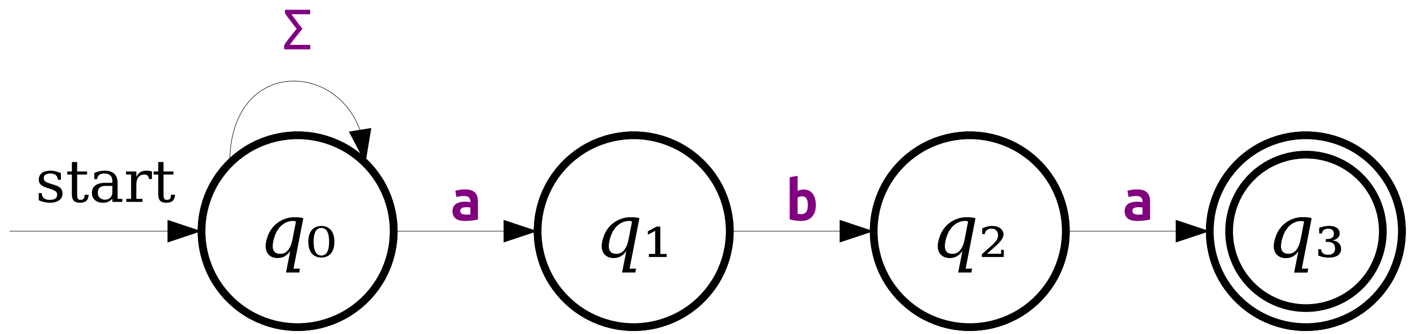
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



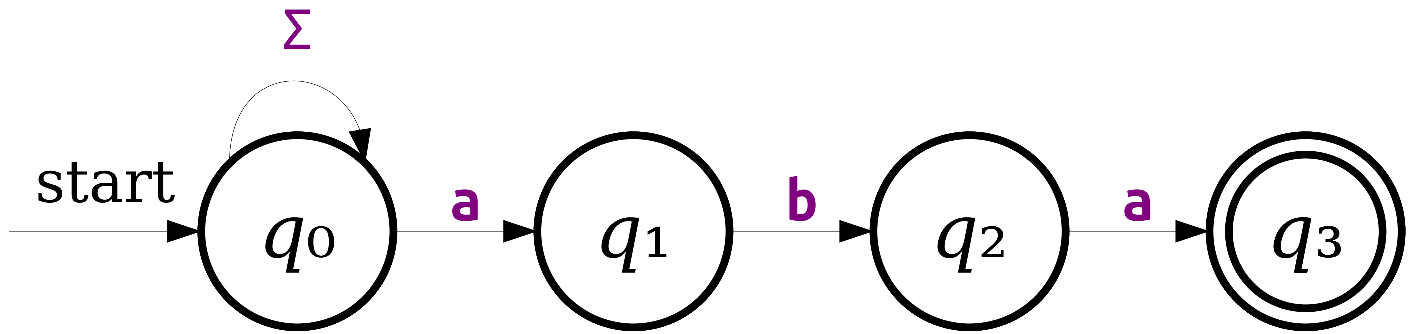
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	



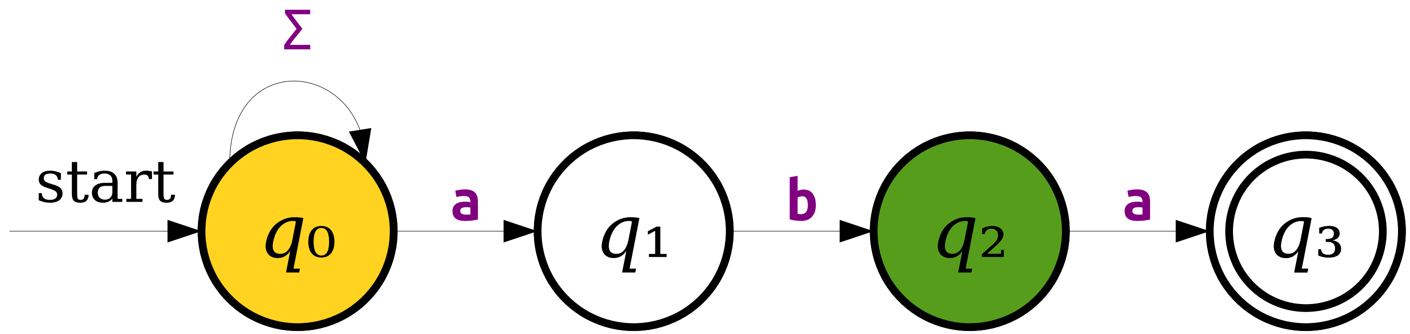
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



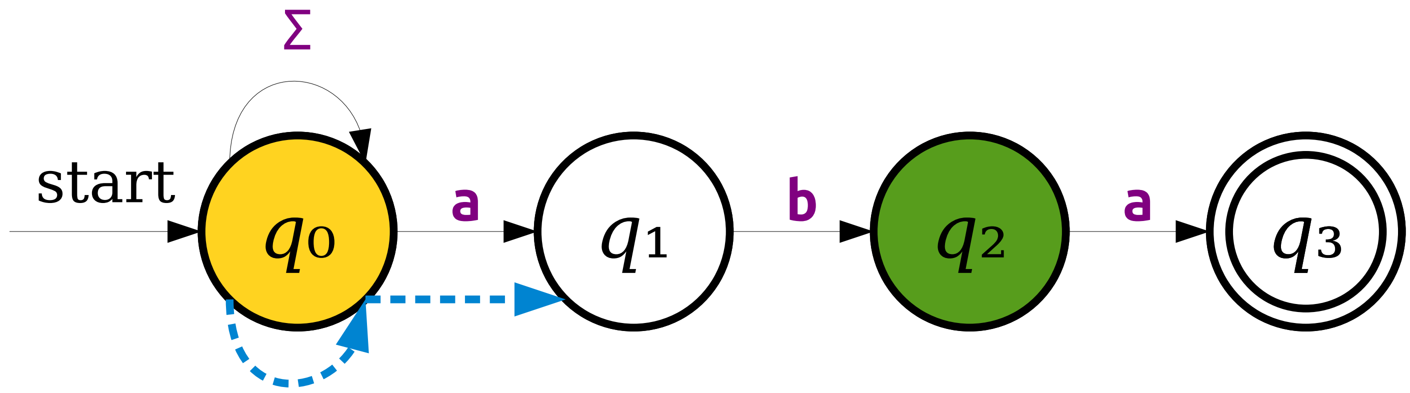
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



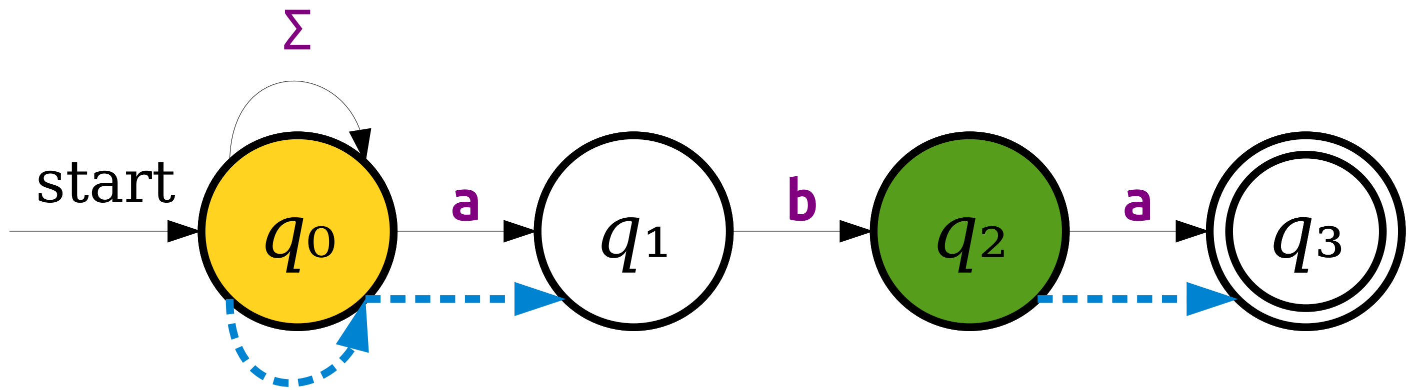
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



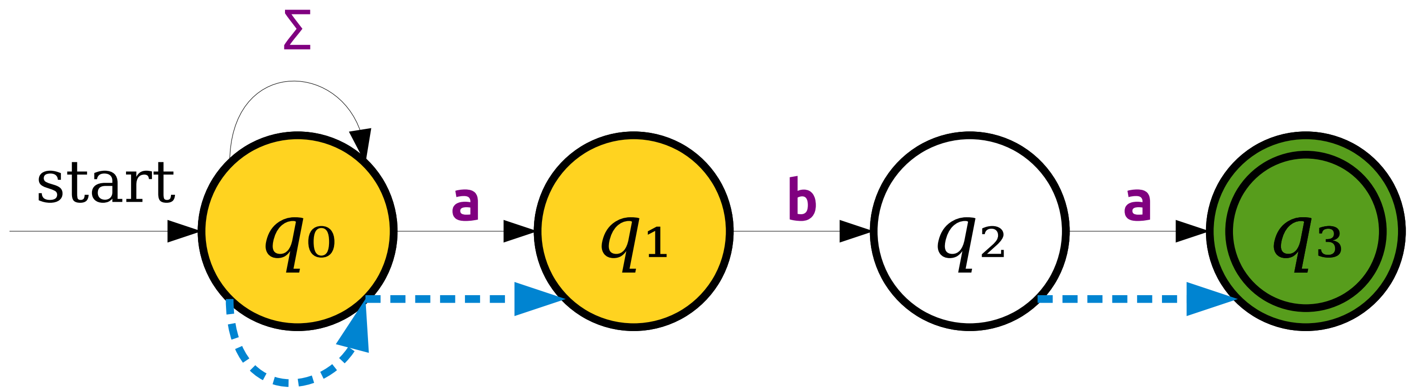
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



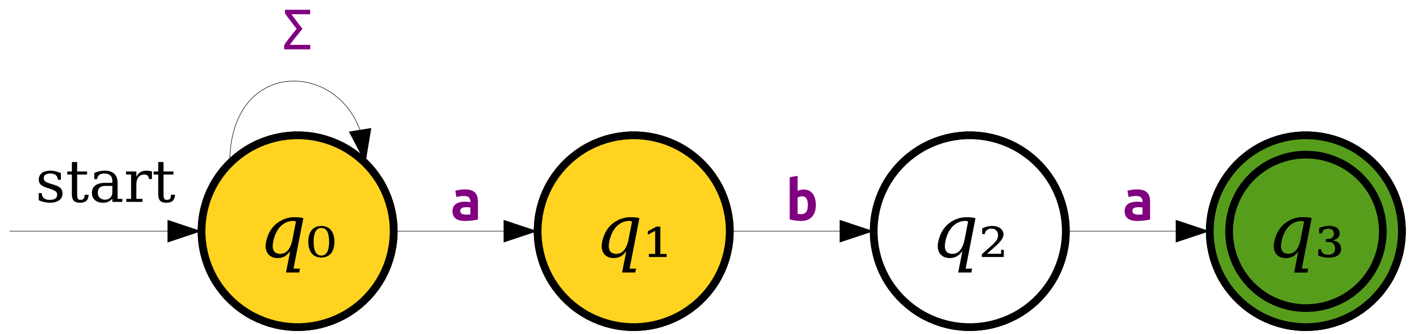
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



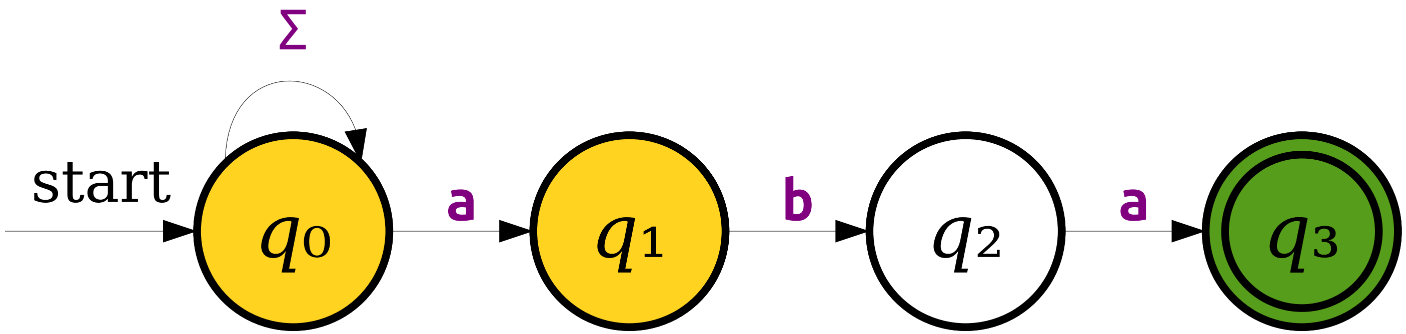
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



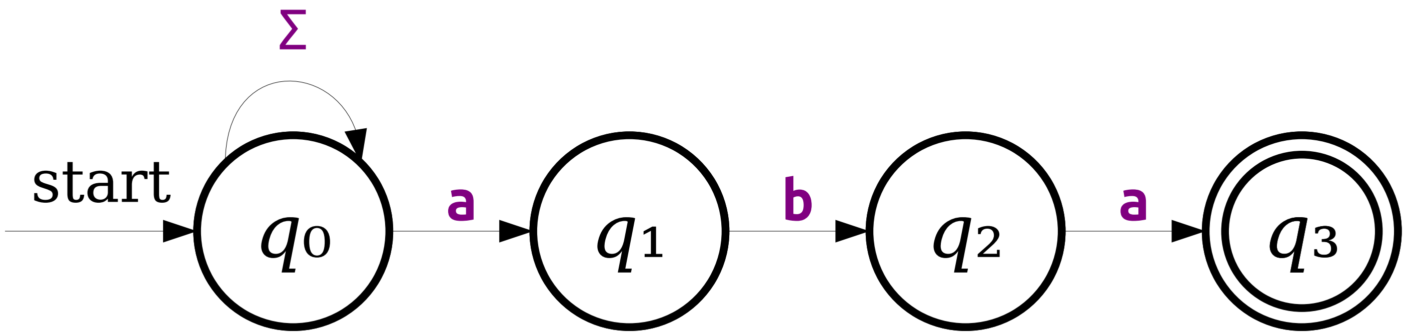
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



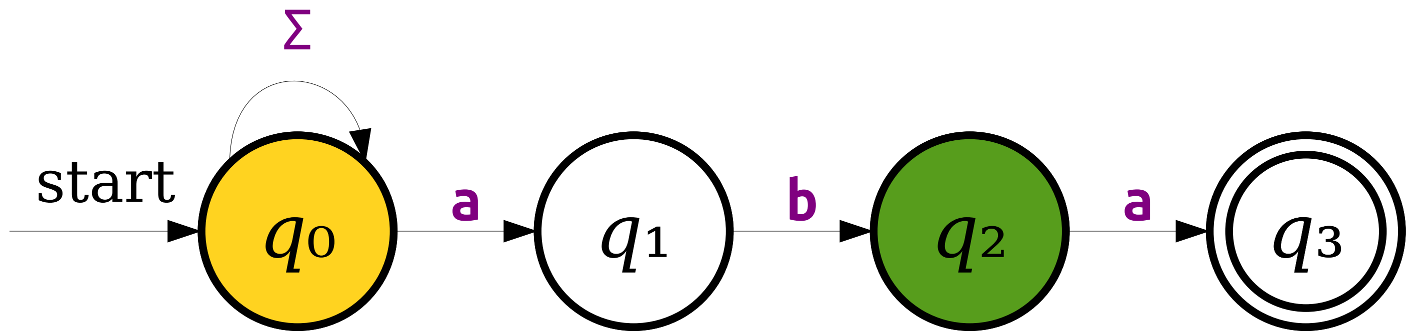
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$		



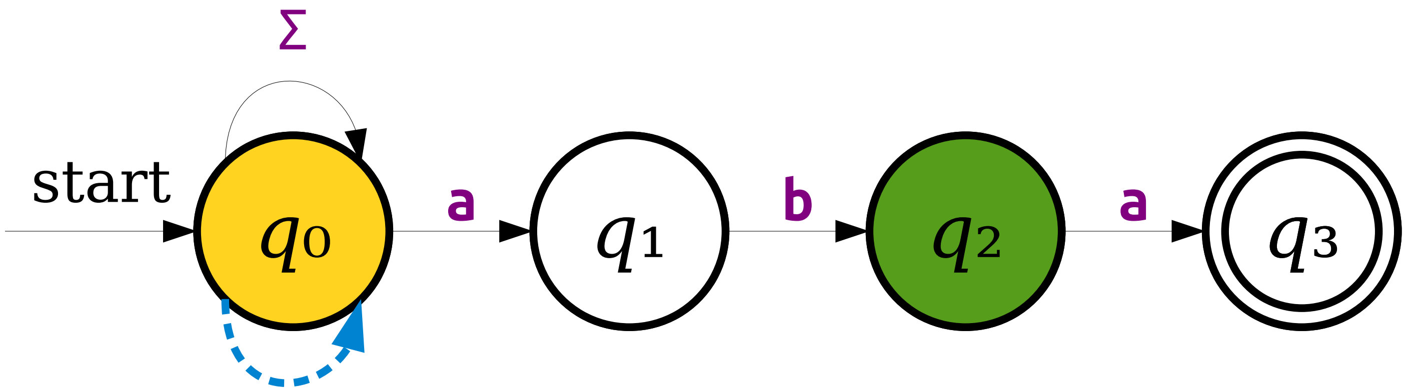
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



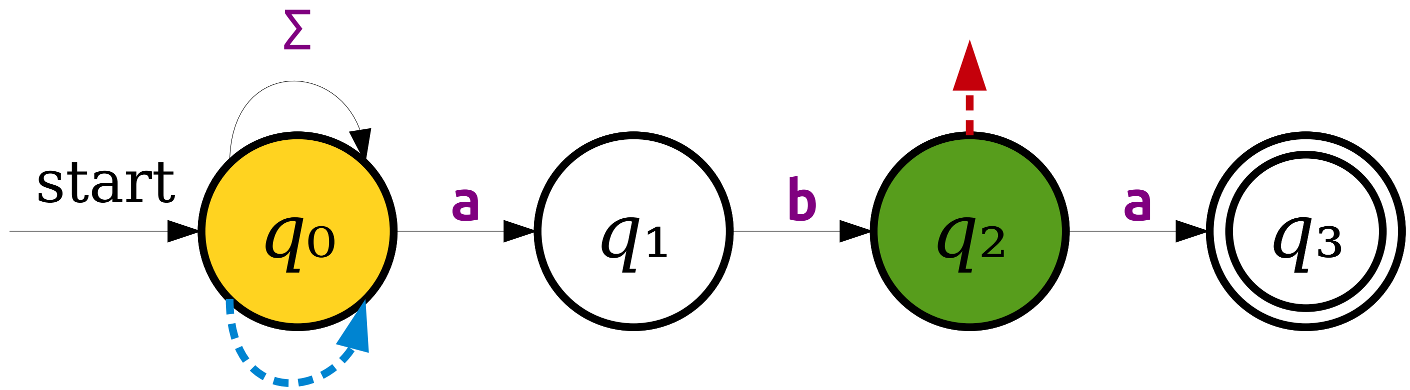
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



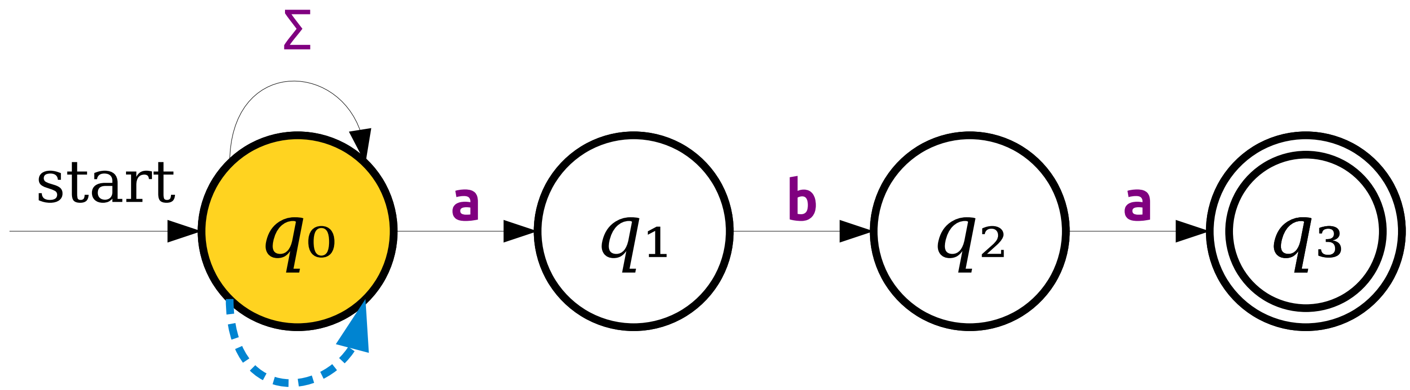
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



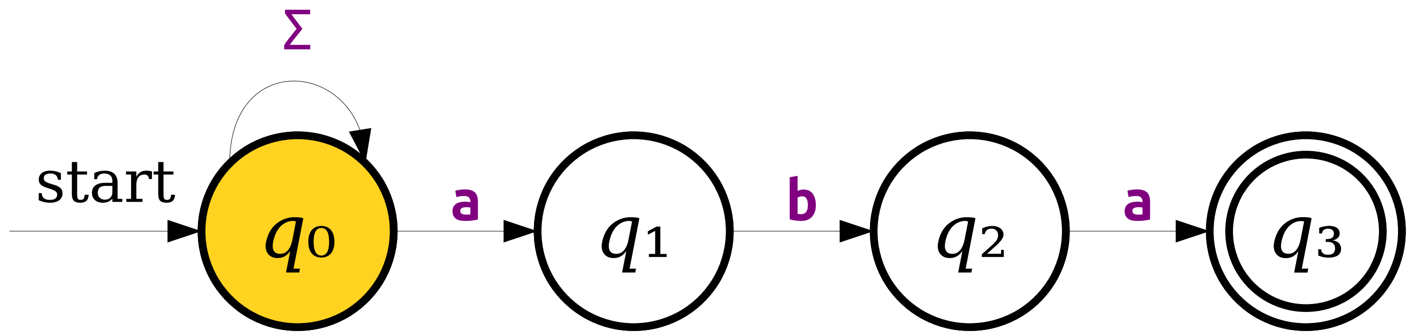
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



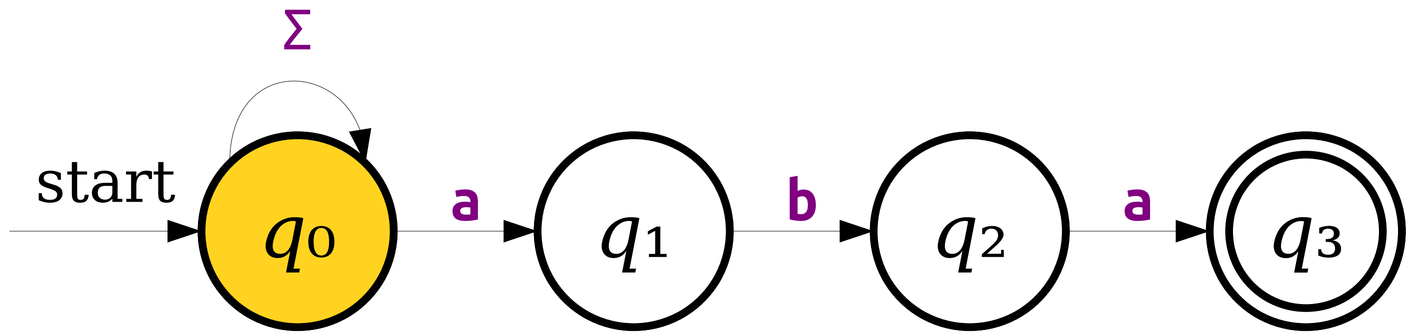
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



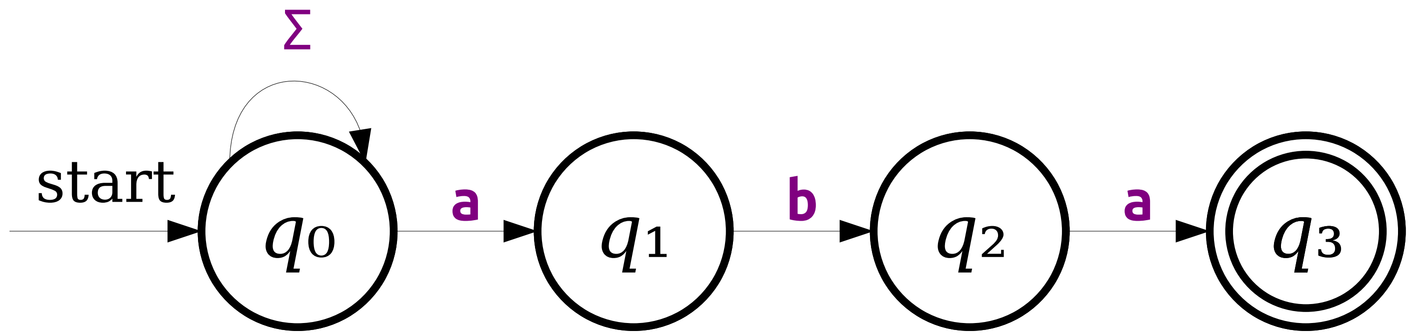
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



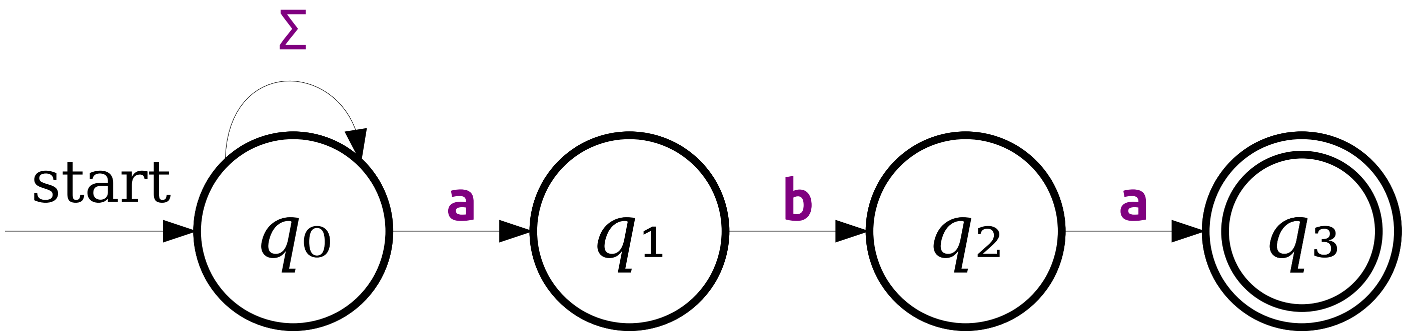
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	



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



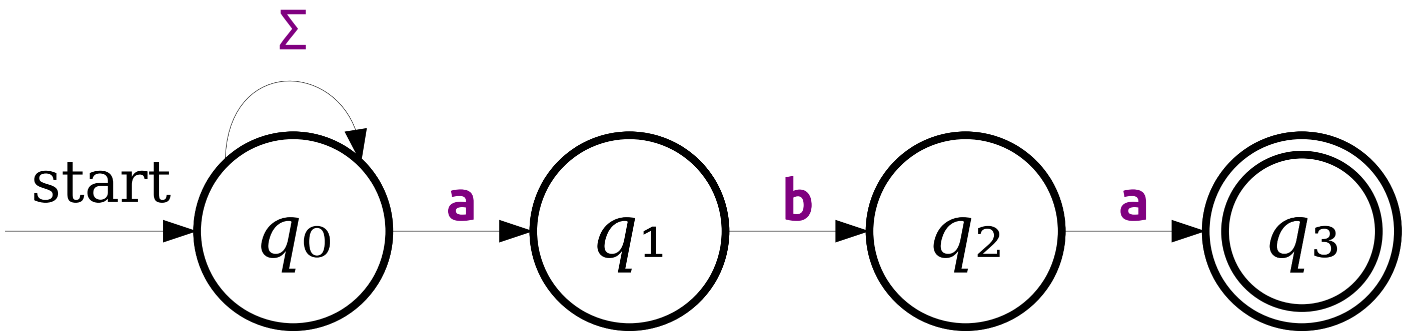
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$



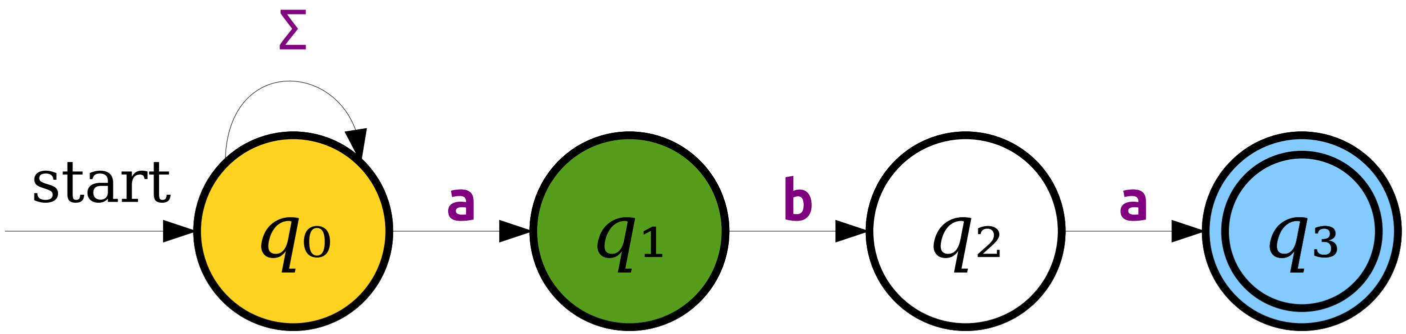
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		

Fill in this row.

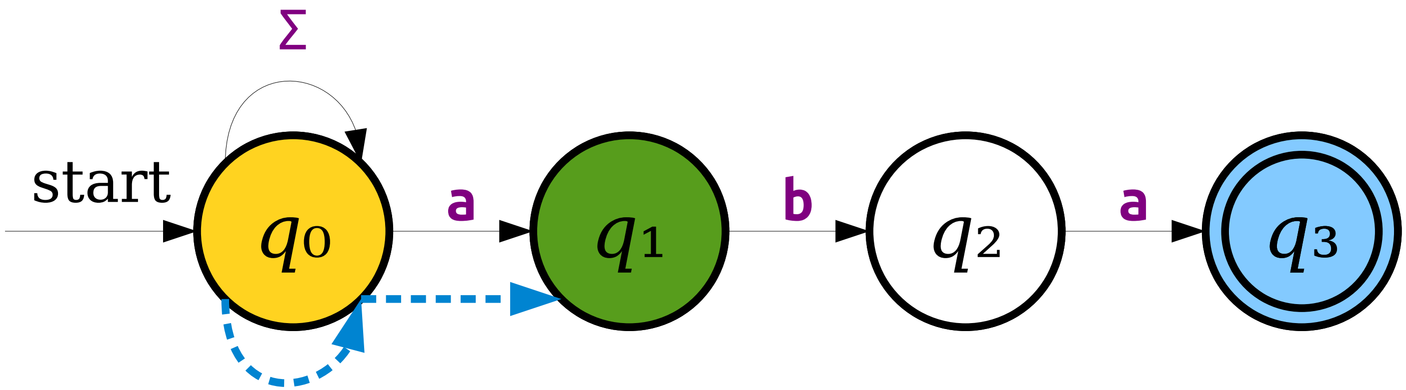
Answer at
https://cs103.stanford.edu/pollev



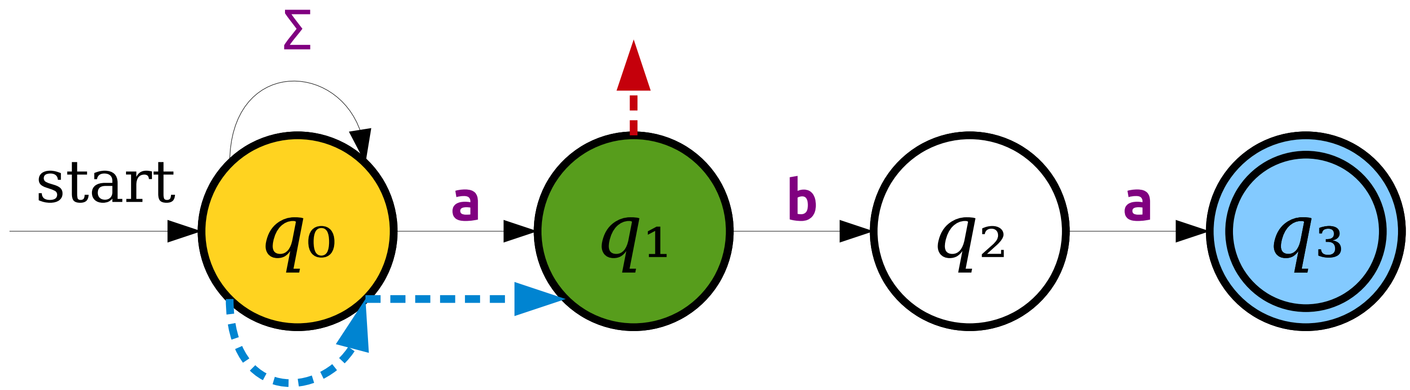
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



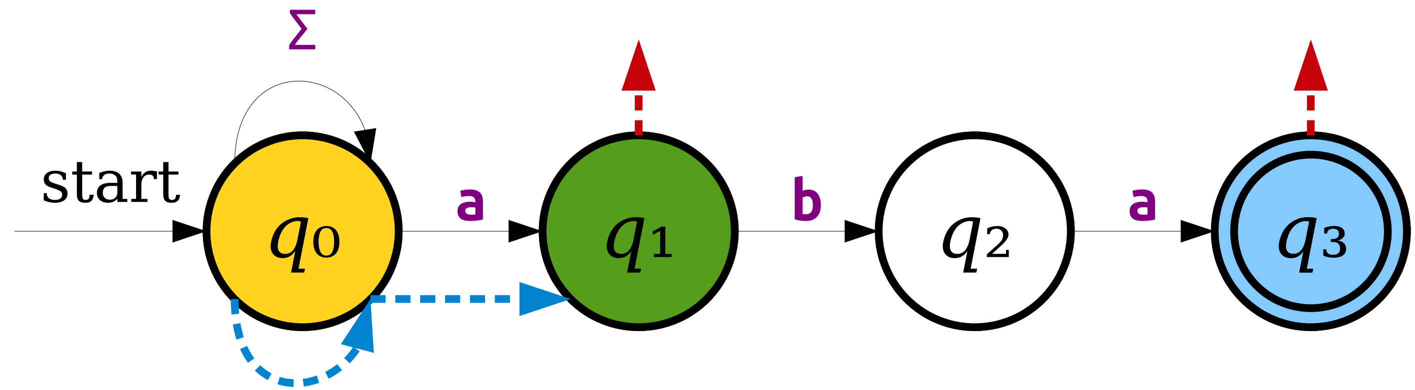
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



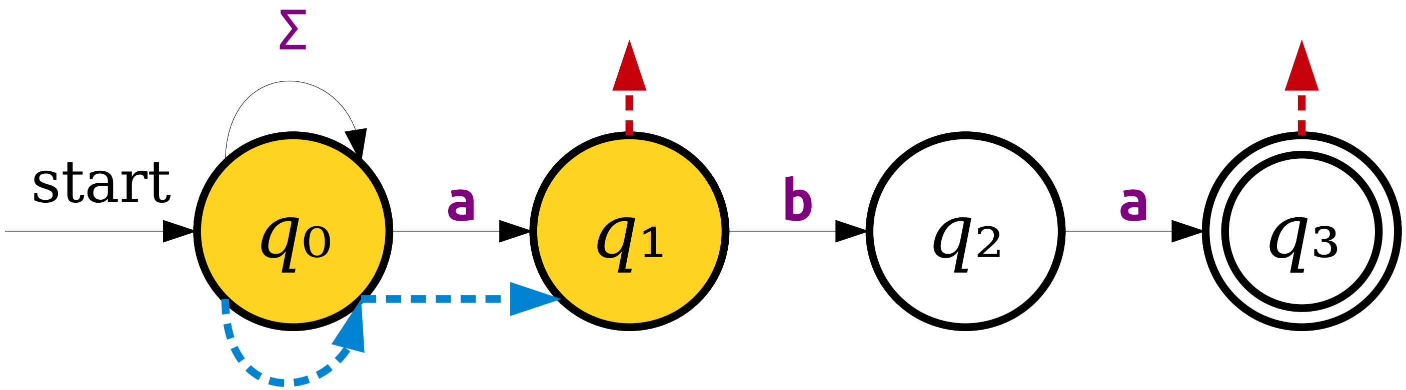
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



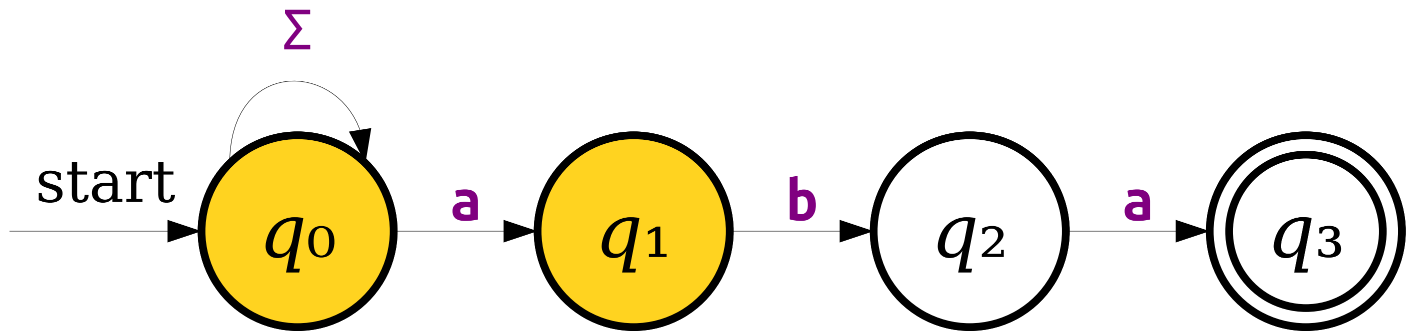
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



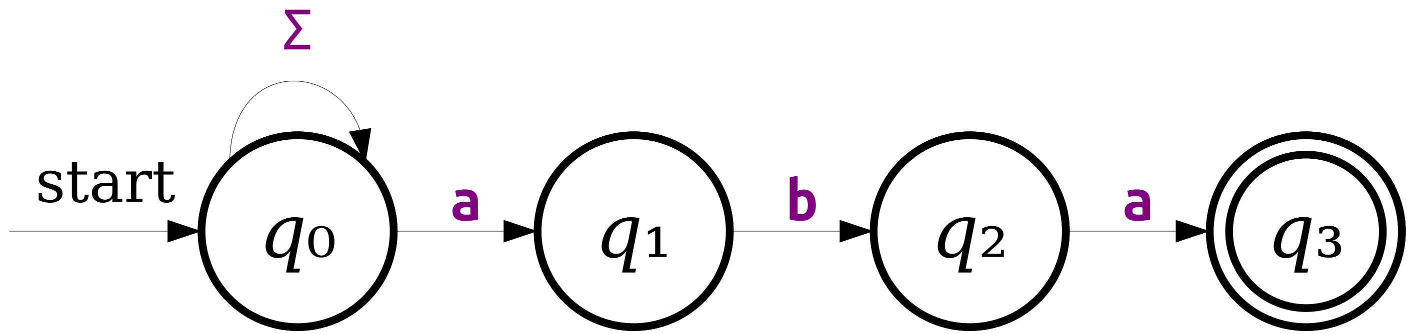
	a	b
$\{q_0\}$	$\{q_0,\ q_1\}$	$\{q_0\}$
$\{q_0,\ q_1\}$	$\{q_0,\ q_1\}$	$\{q_0,\ q_2\}$
$\{q_0,\ q_2\}$	$\{q_0,\ q_1,\ q_3\}$	$\{q_0\}$
$\{q_0,\ q_1,\ q_3\}$		



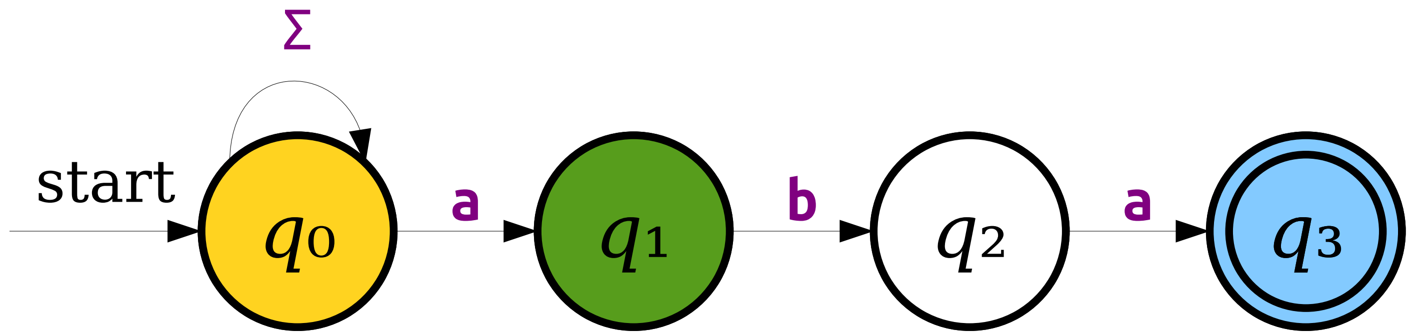
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$		



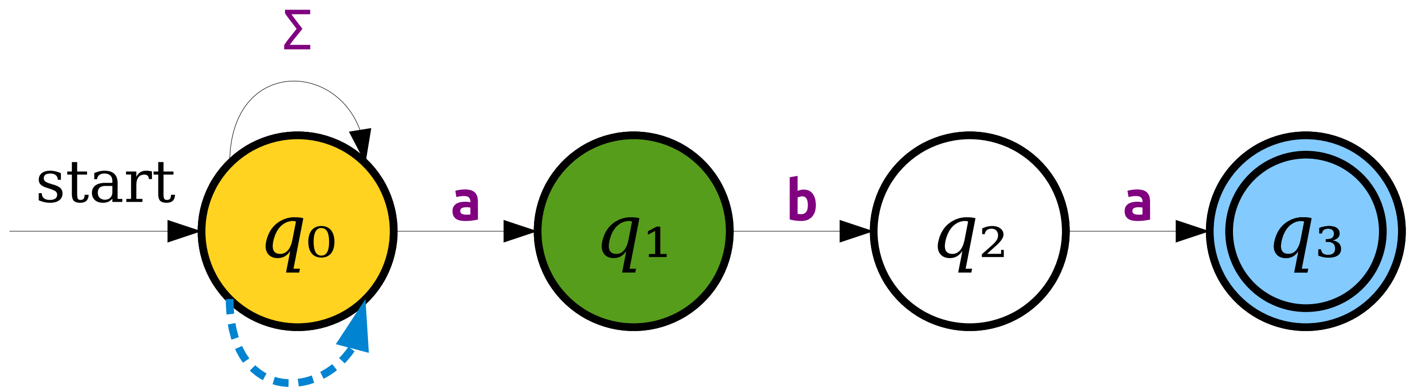
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



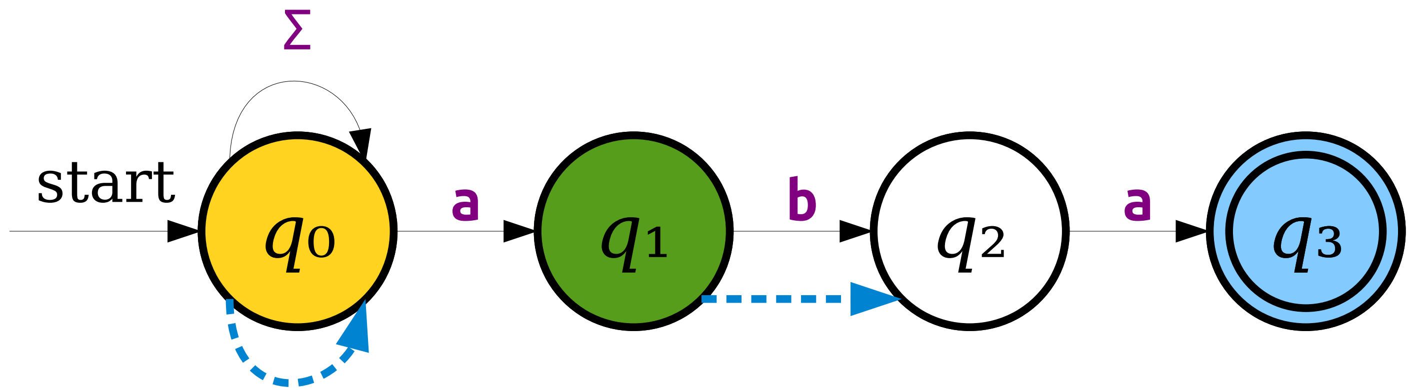
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



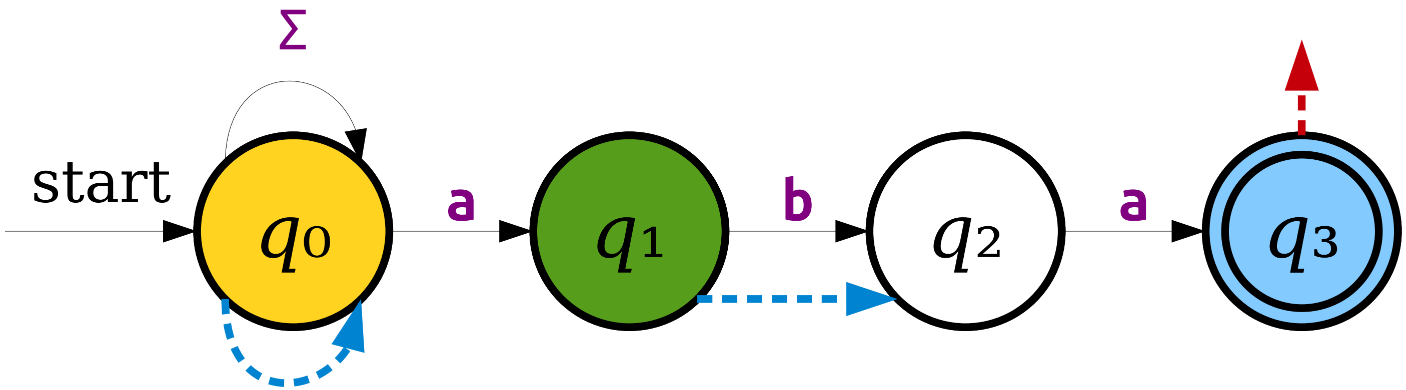
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



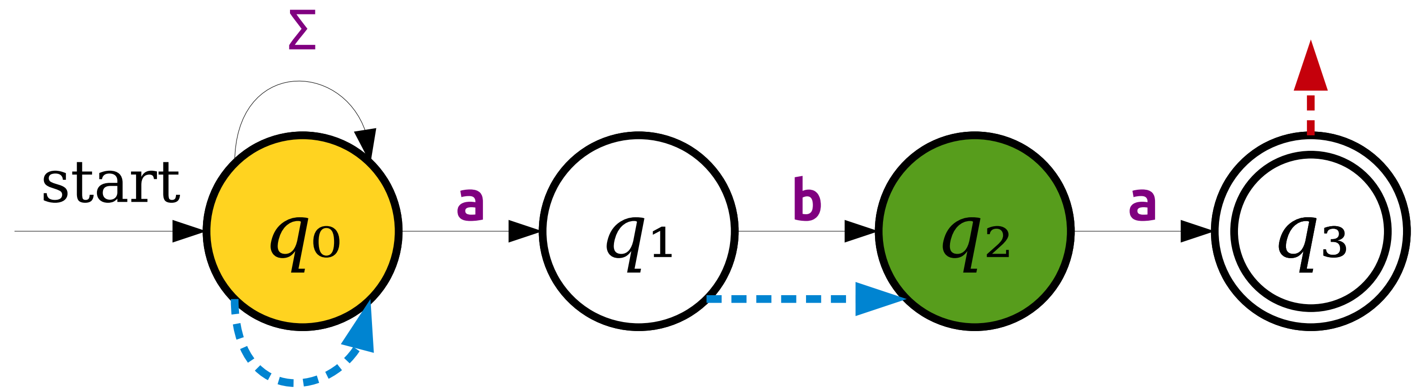
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



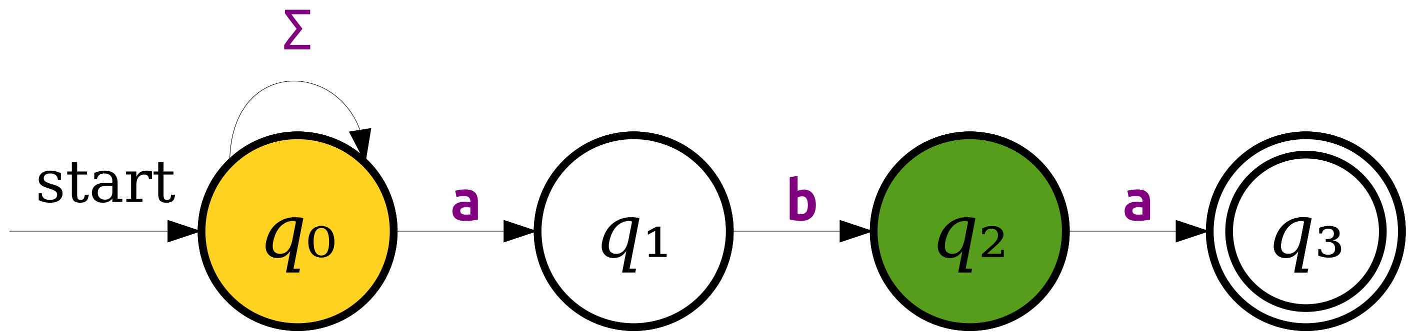
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



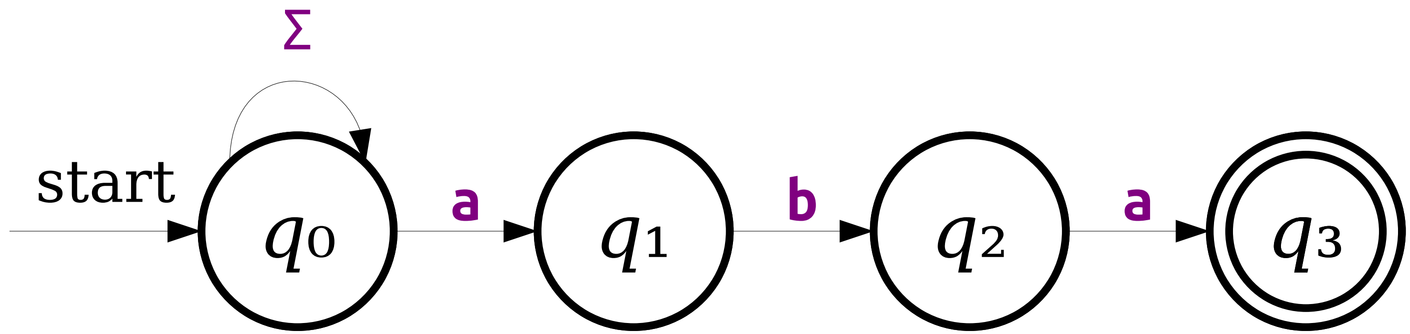
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



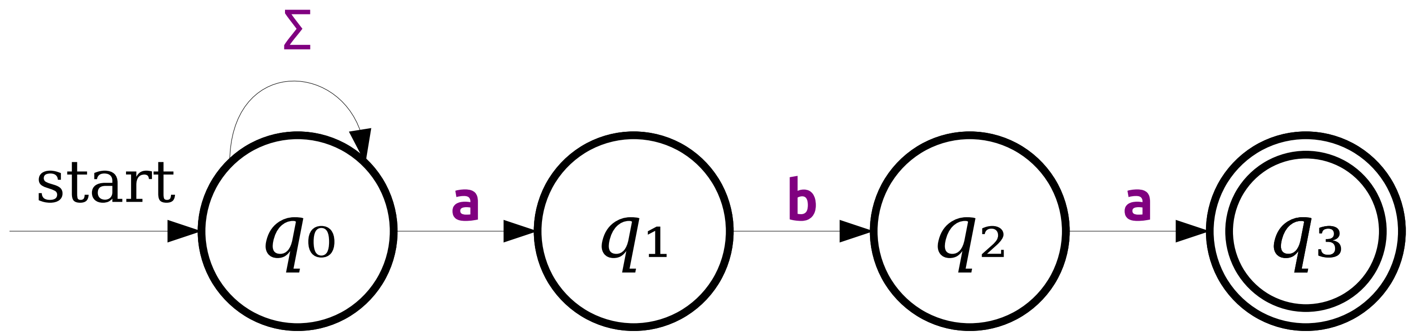
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	



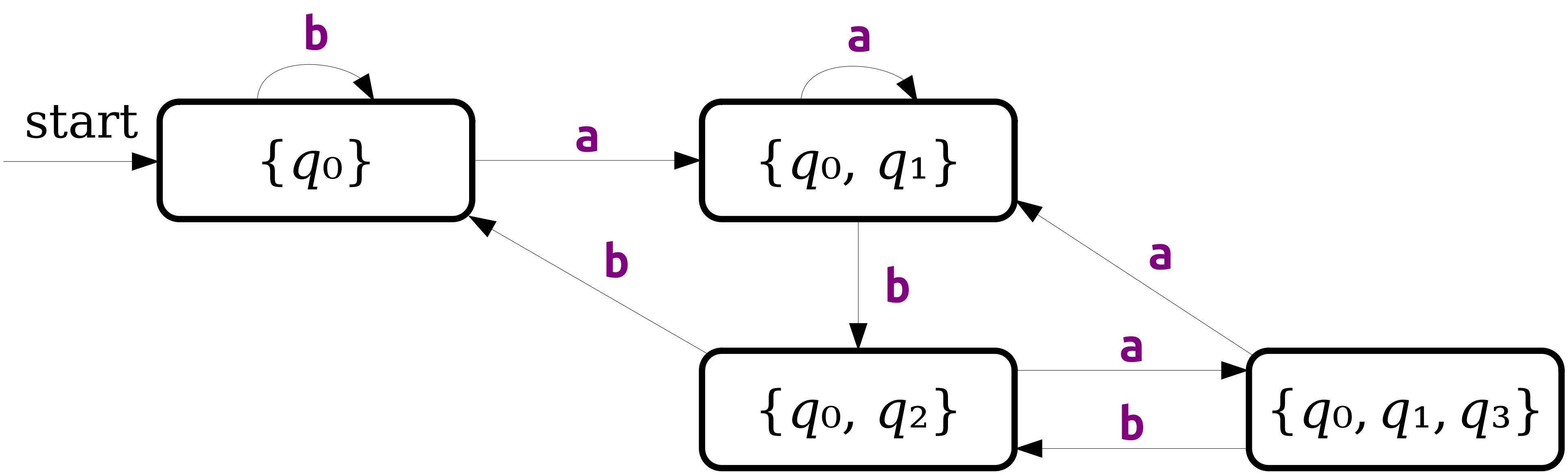
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

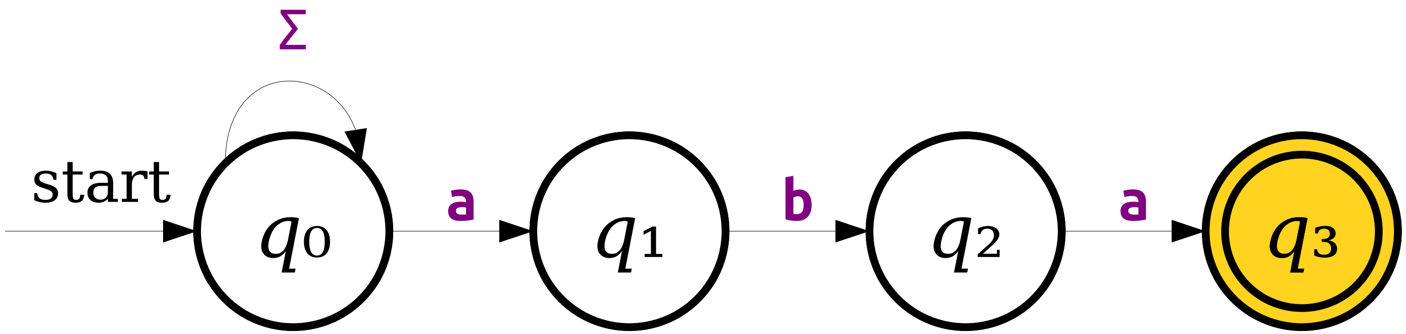


	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

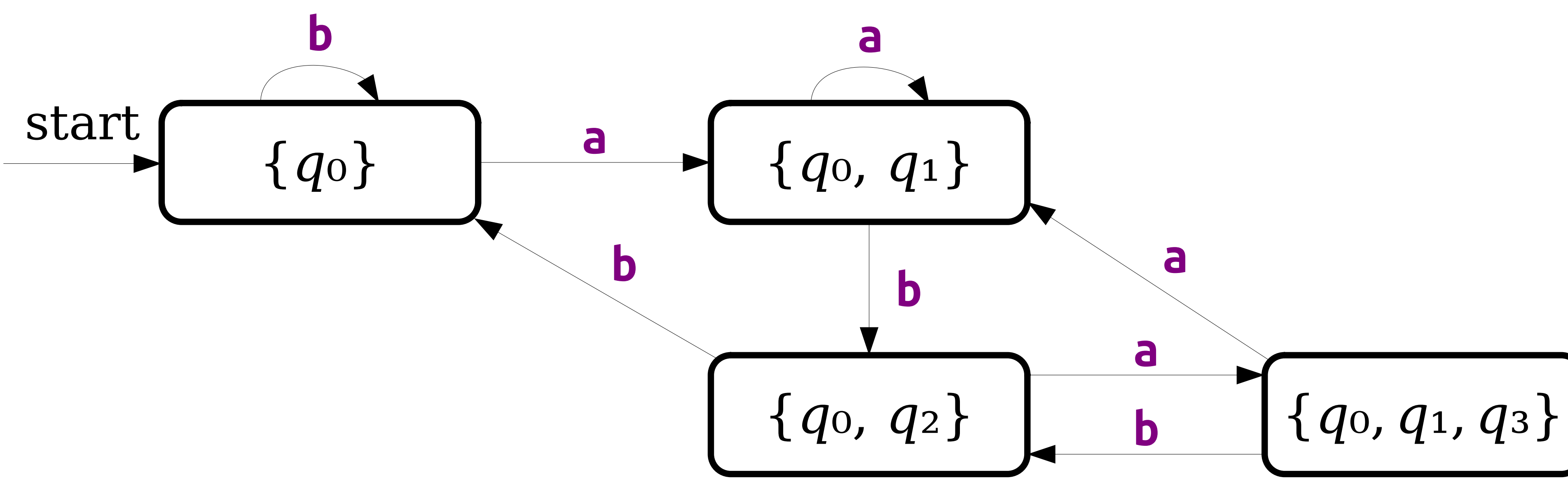


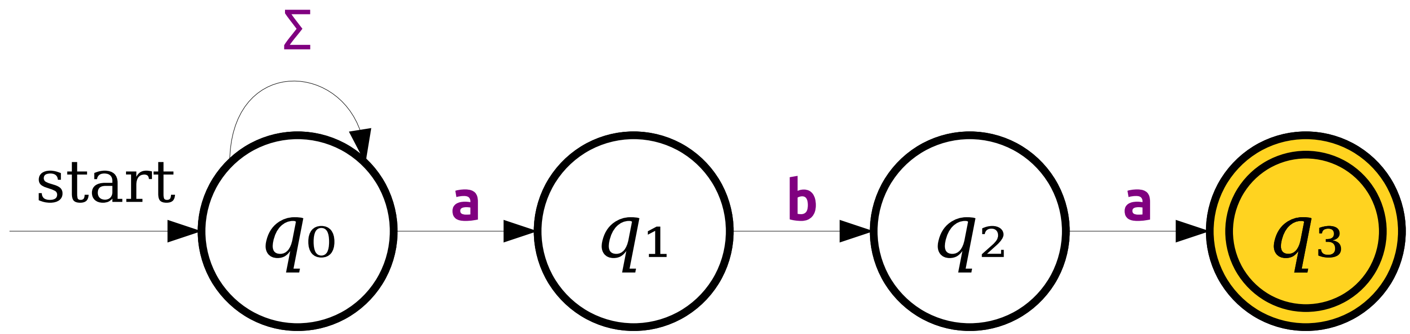
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



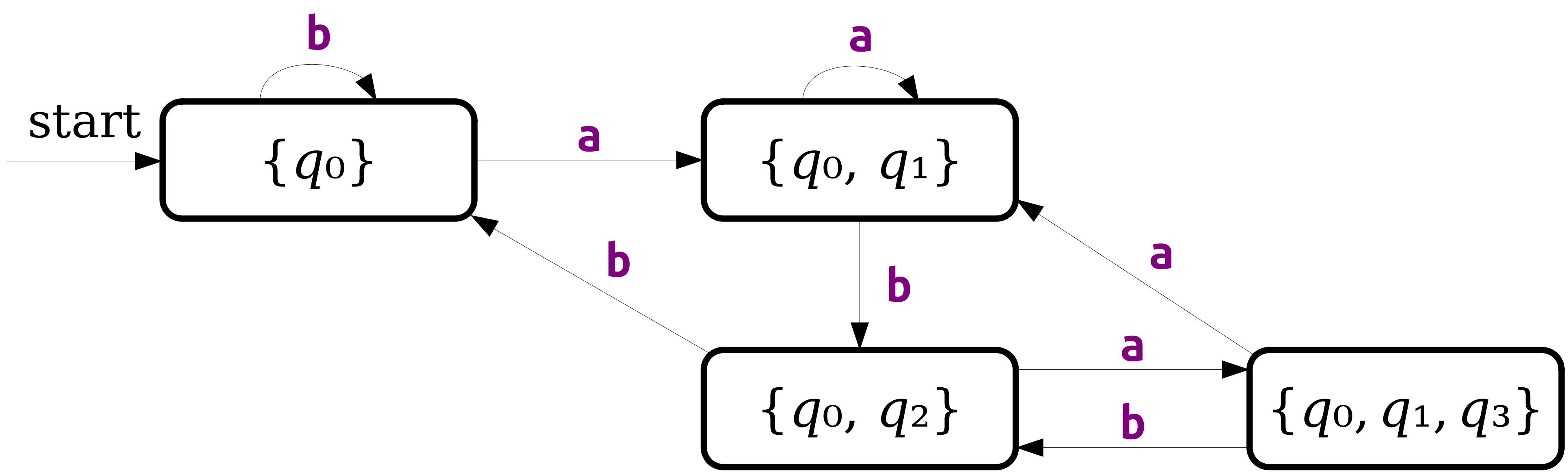


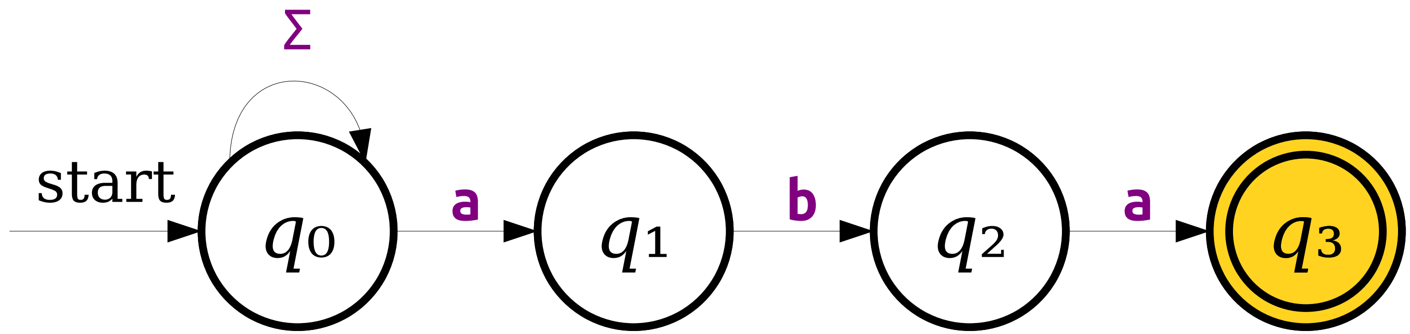
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



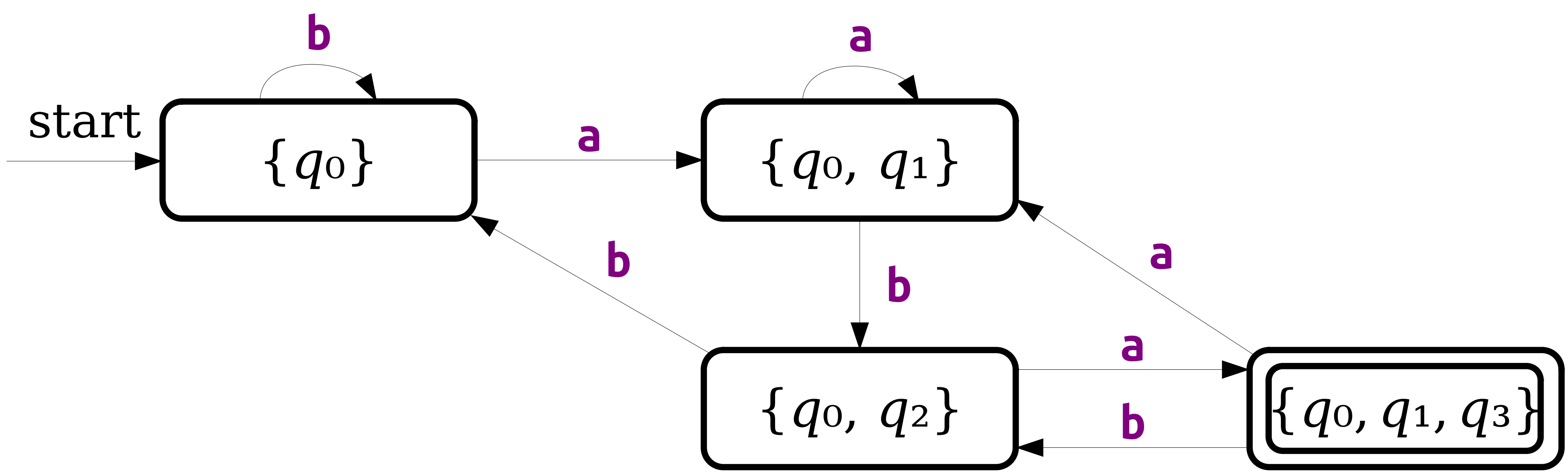


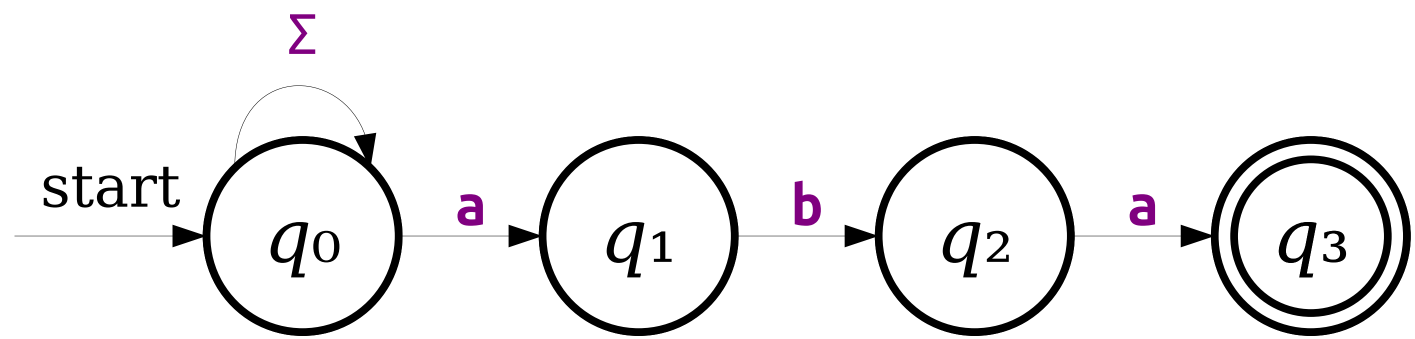
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



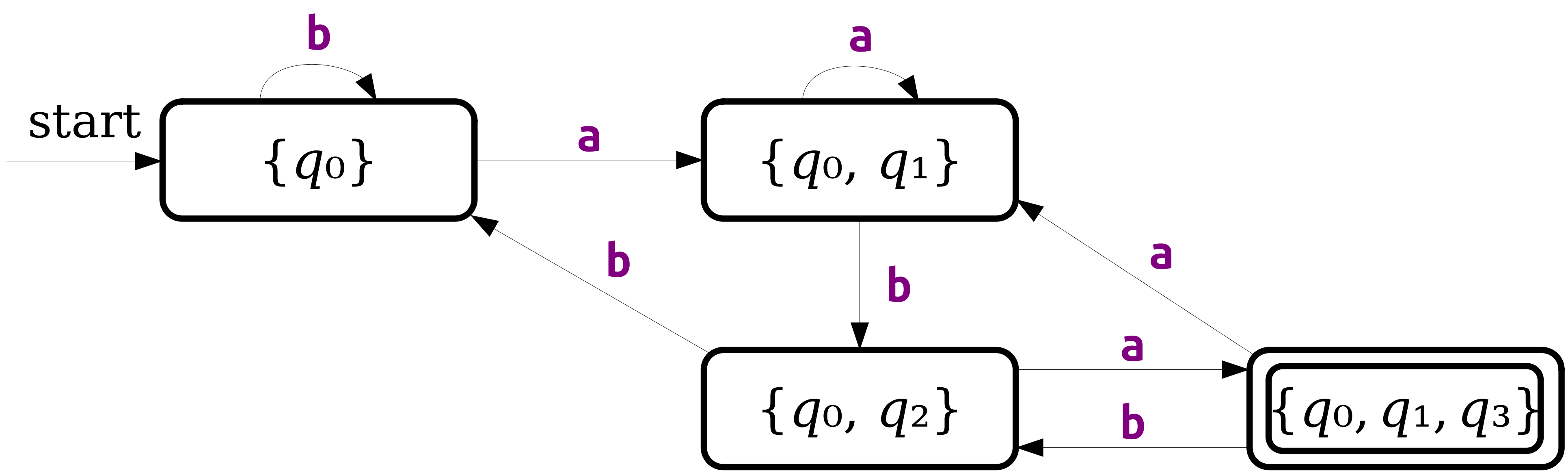


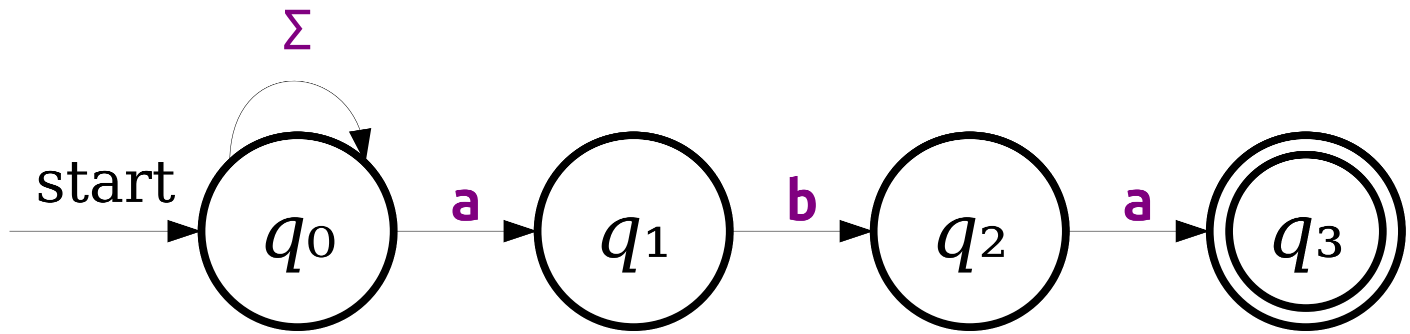
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



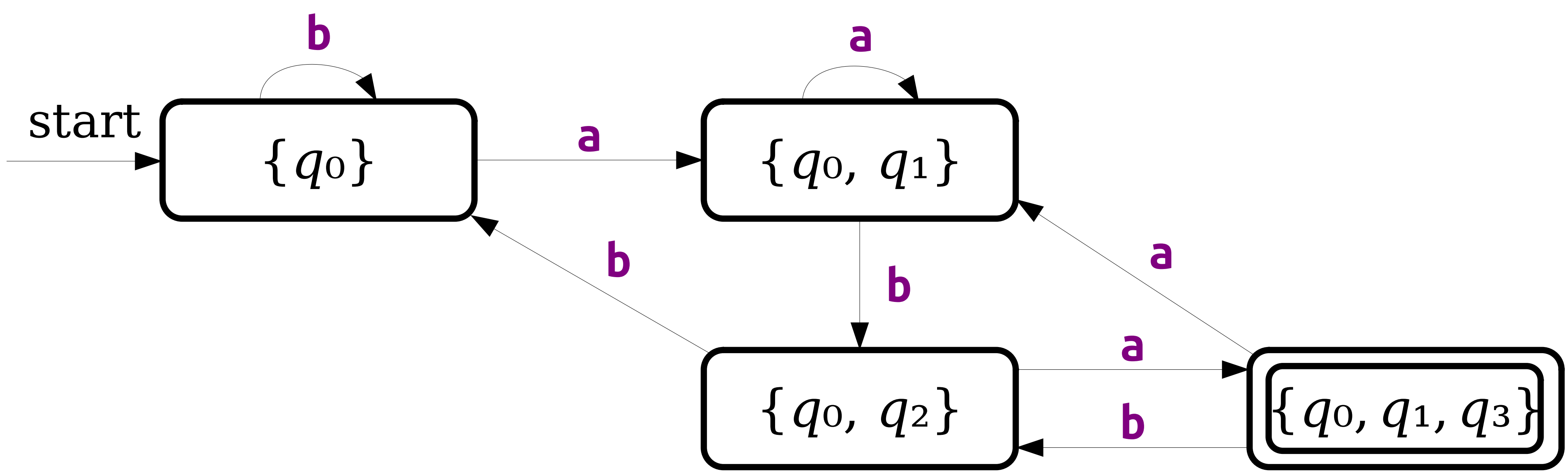


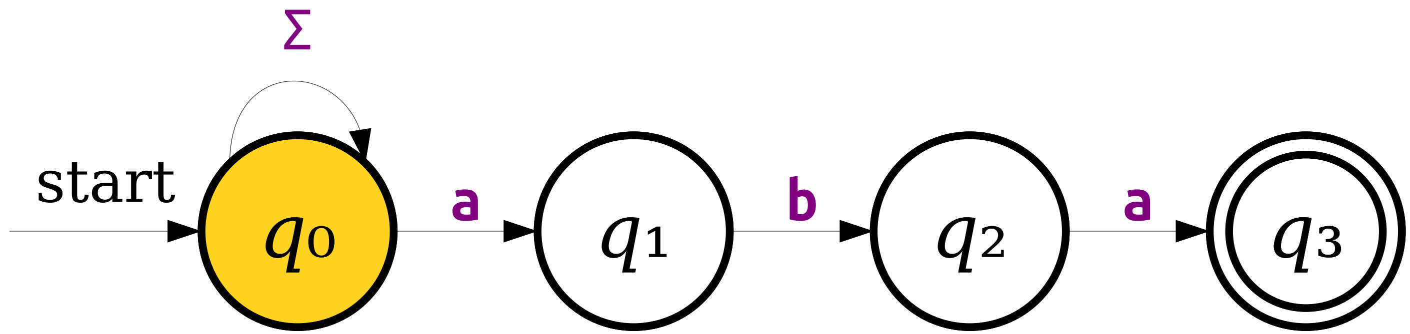
	a	b
$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

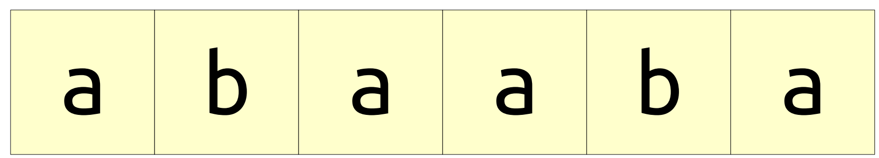


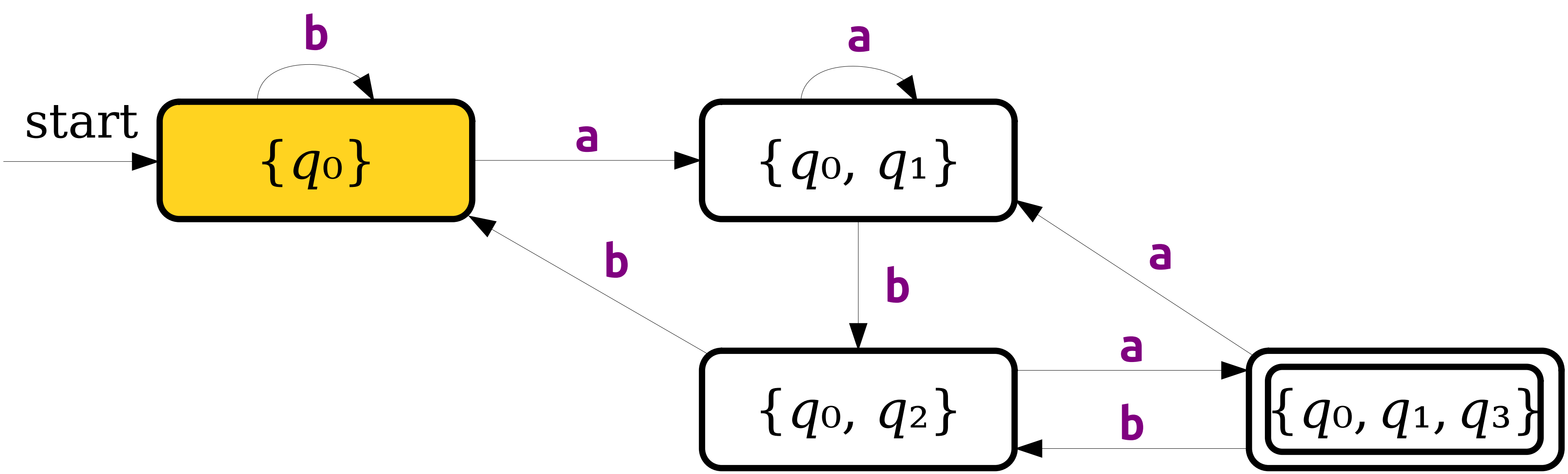


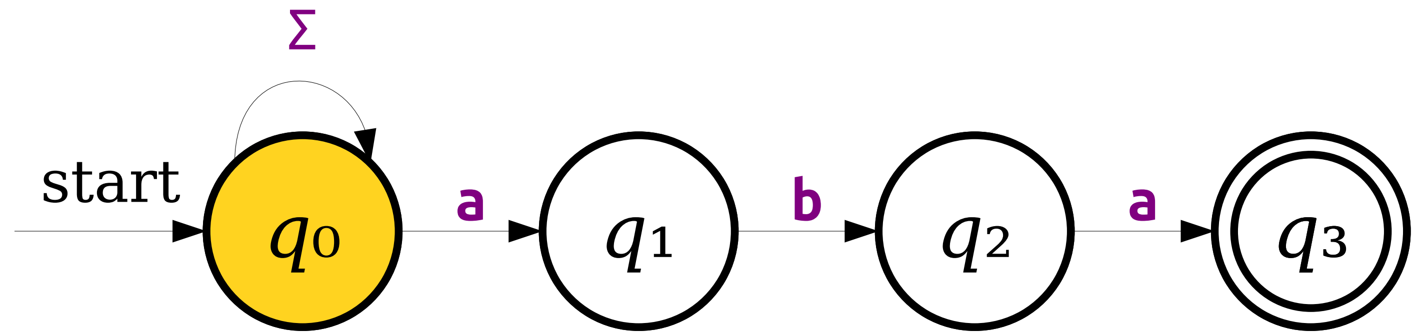
a	b	a	a	b	a

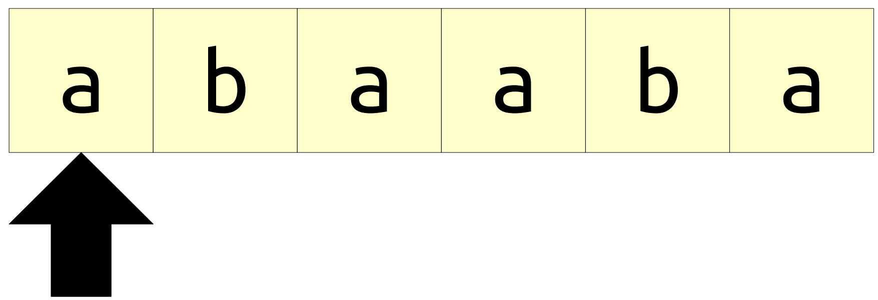


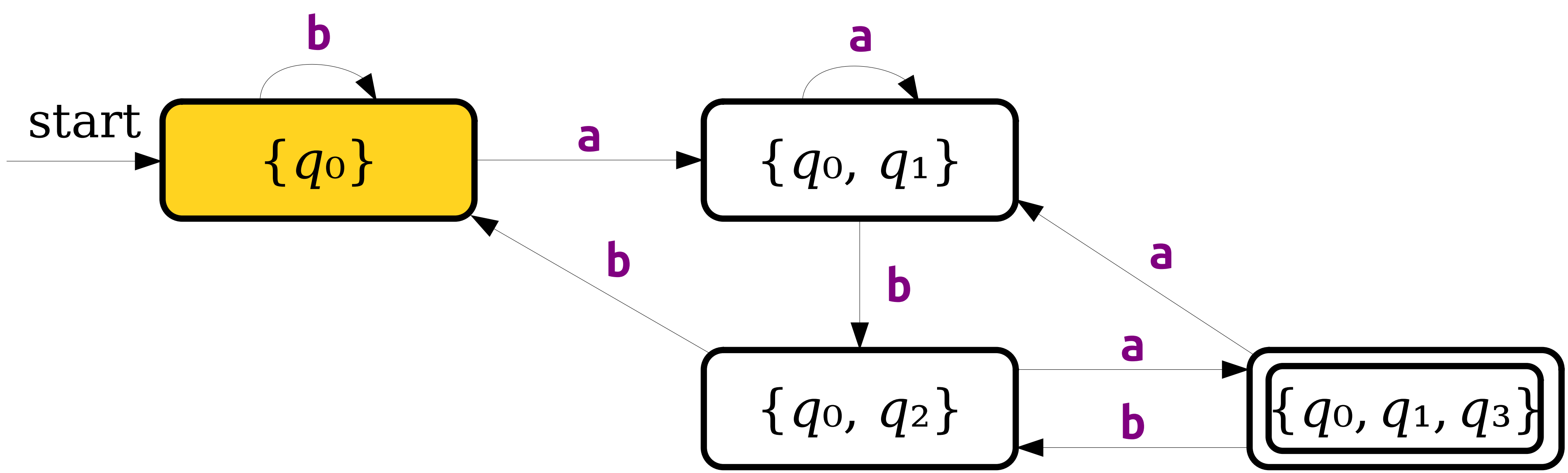

Σ
start
q_0
a
q_1
b
q_2
a
q_3

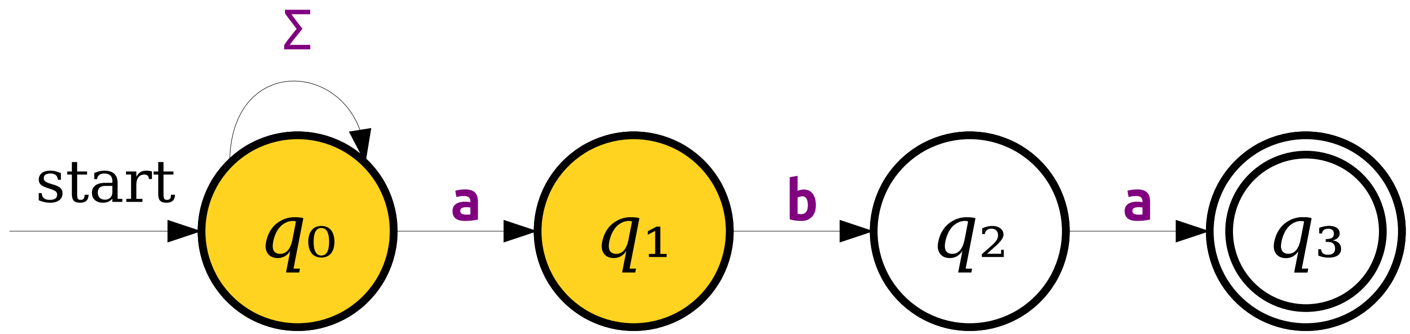

a
b
a
a
b
a

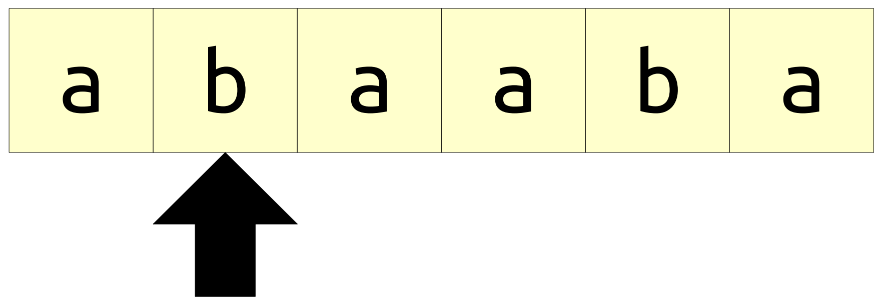

b
start
$\{q_0\}$
a
a
$\{q_0, q_1\}$
b
b
a
a
$\{q_0, q_2\}$
b
$\{q_0, q_1, q_3\}$

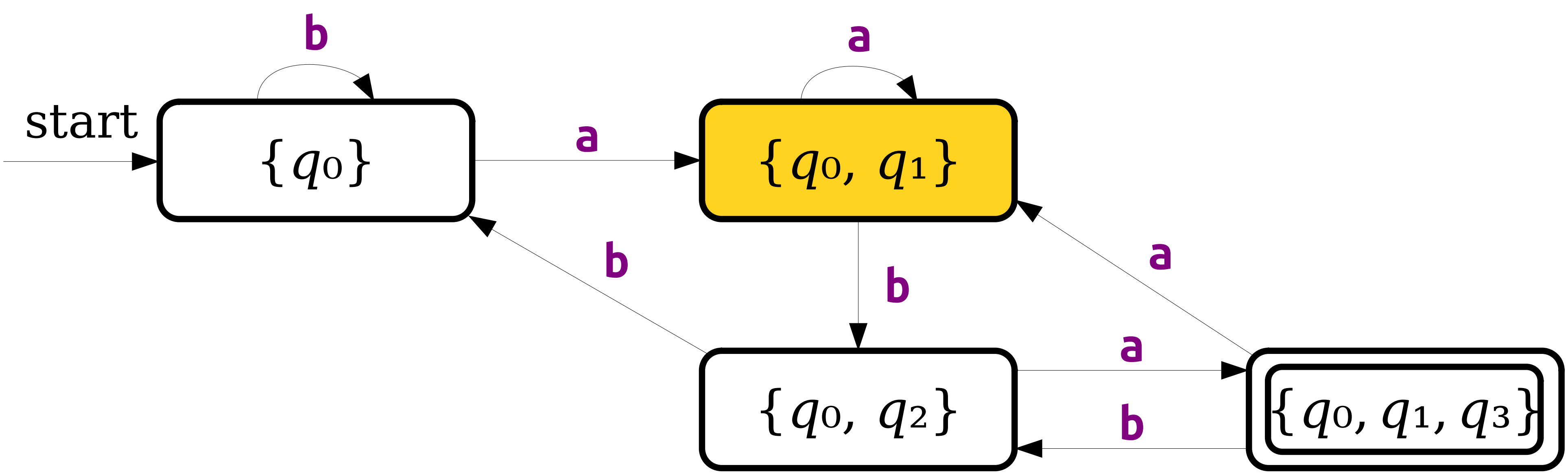

Σ
start
q_0
a
q_1
b
q_2
a
q_3

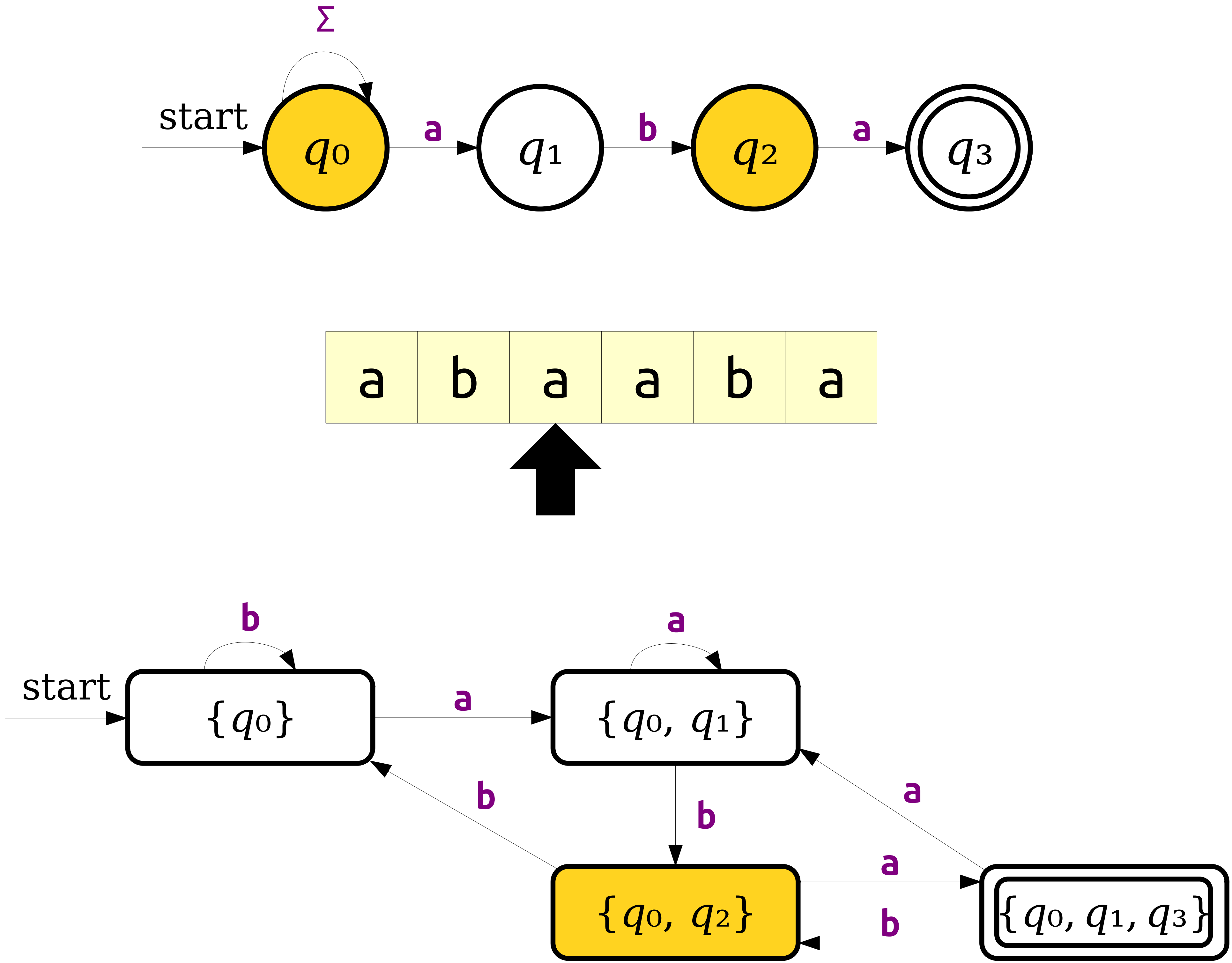

a
b
a
a
b
a

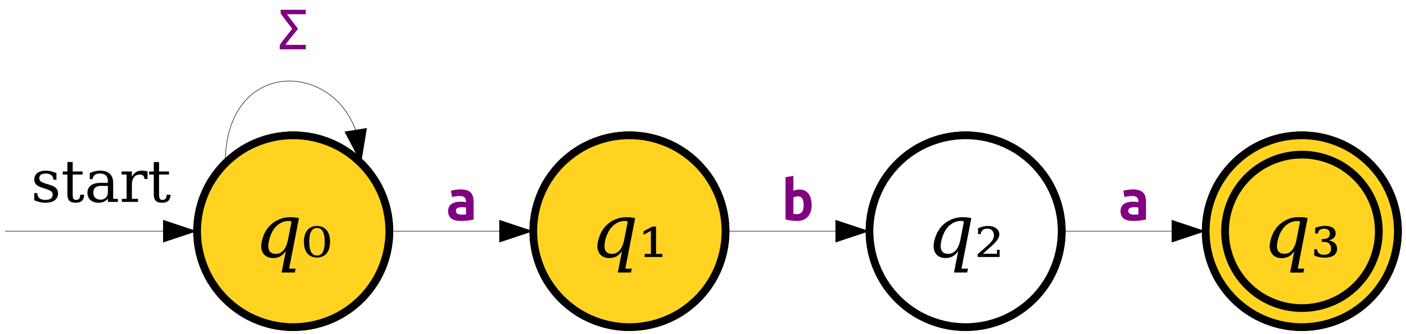

b
start
$\{q_0\}$
a
$\{q_0, q_1\}$
a
b
b
a
a
b
$\{q_0, q_2\}$
$\{q_0, q_1, q_3\}$


Σ
start
q_0
a
q_1
b
q_2
a
q_3

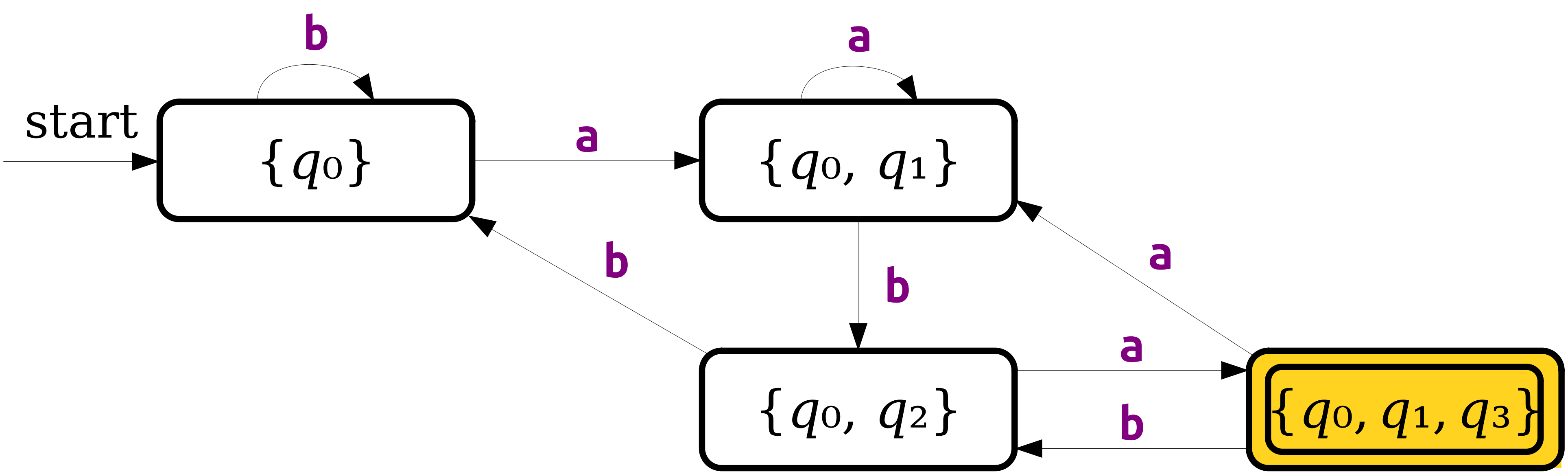

a b a a b a

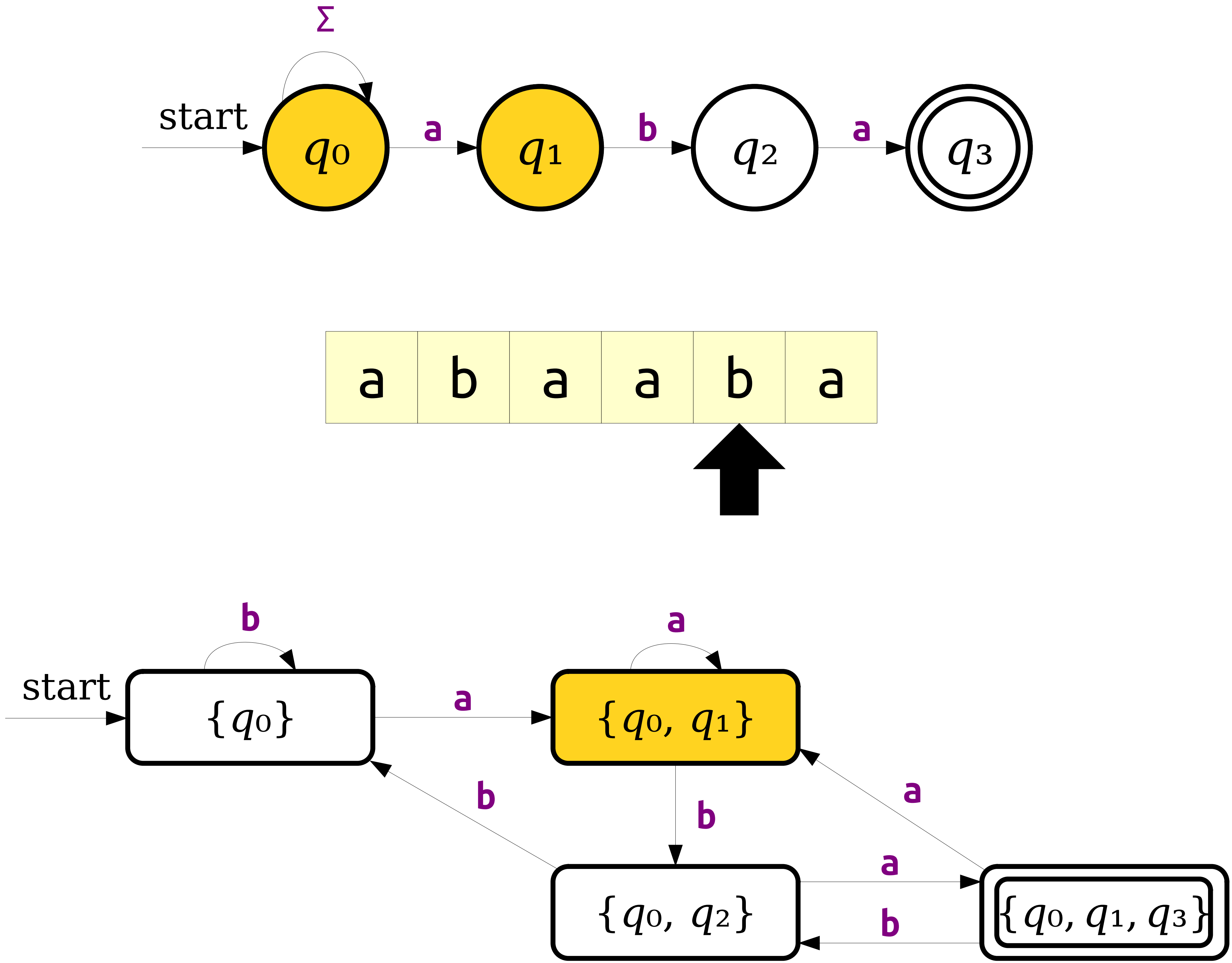

b
a
start
$\{q_0\}$
a
$\{q_0, q_1\}$
b
b
a
a
b
$\{q_0, q_2\}$
$\{q_0, q_1, q_3\}$

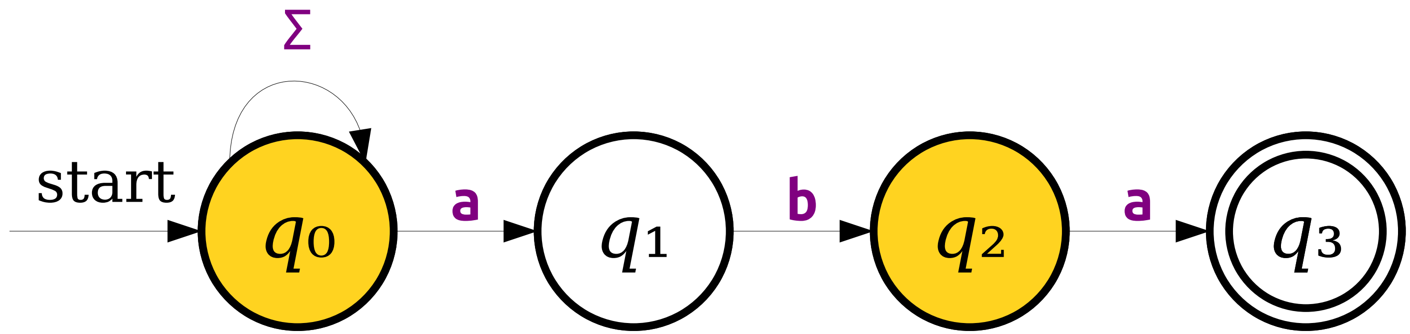

Σ
start
q_0
a
q_1
b
q_2
a
q_3
a b a a b a
b
a
start
$\{q_0\}$
a
$\{q_0, q_1\}$
b
b
a
a
b
$\{q_0, q_2\}$
$\{q_0, q_1, q_3\}$

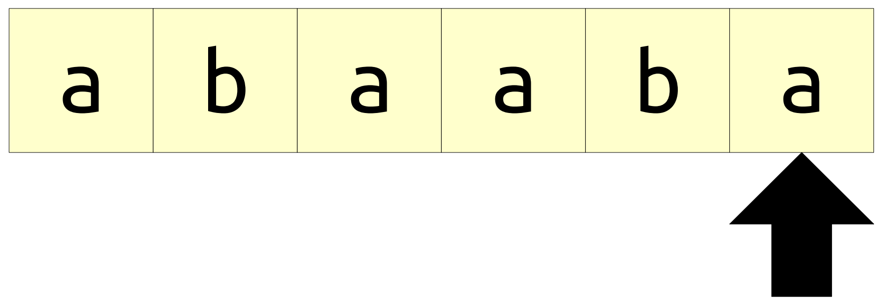


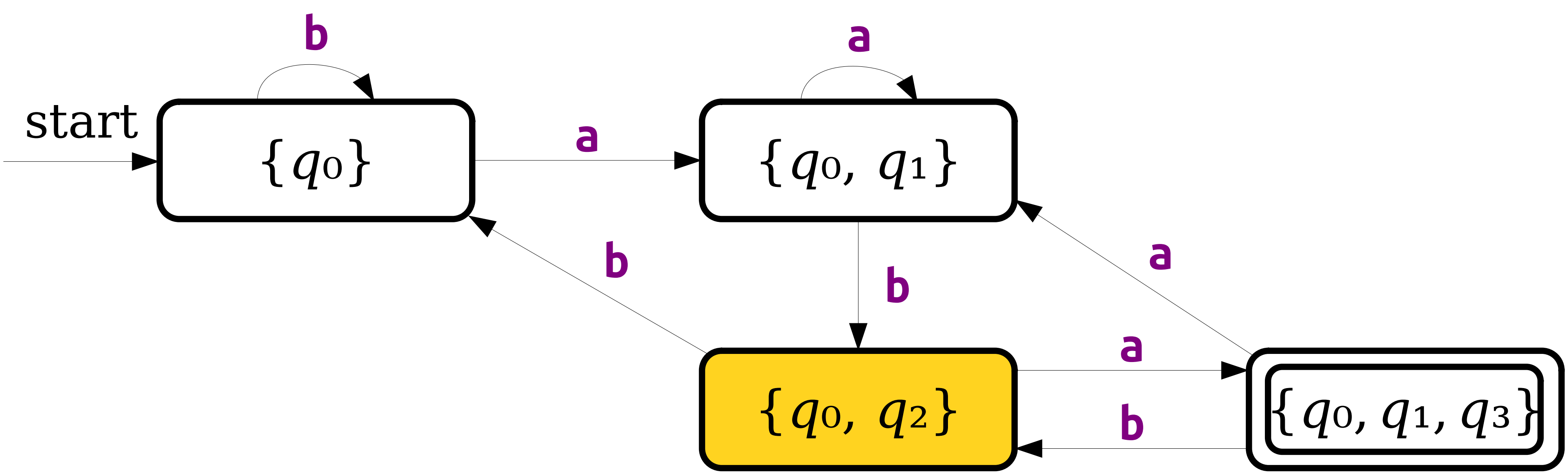
a	b	a	a	b	a

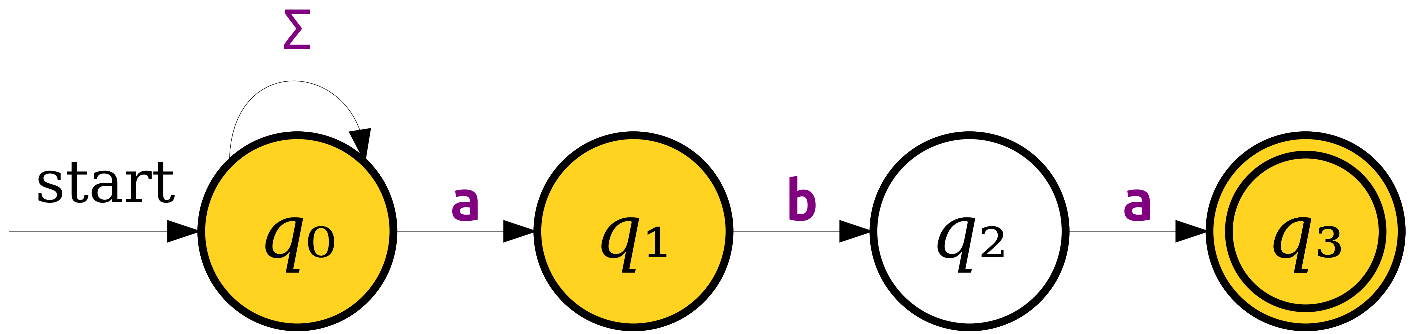



Σ
start
q_0
a
q_1
b
q_2
a
q_3
a b a a b a
b
a
start
$\{q_0\}$
a
$\{q_0, q_1\}$
b
b
a
a
$\{q_0, q_2\}$
b
$\{q_0, q_1, q_3\}$

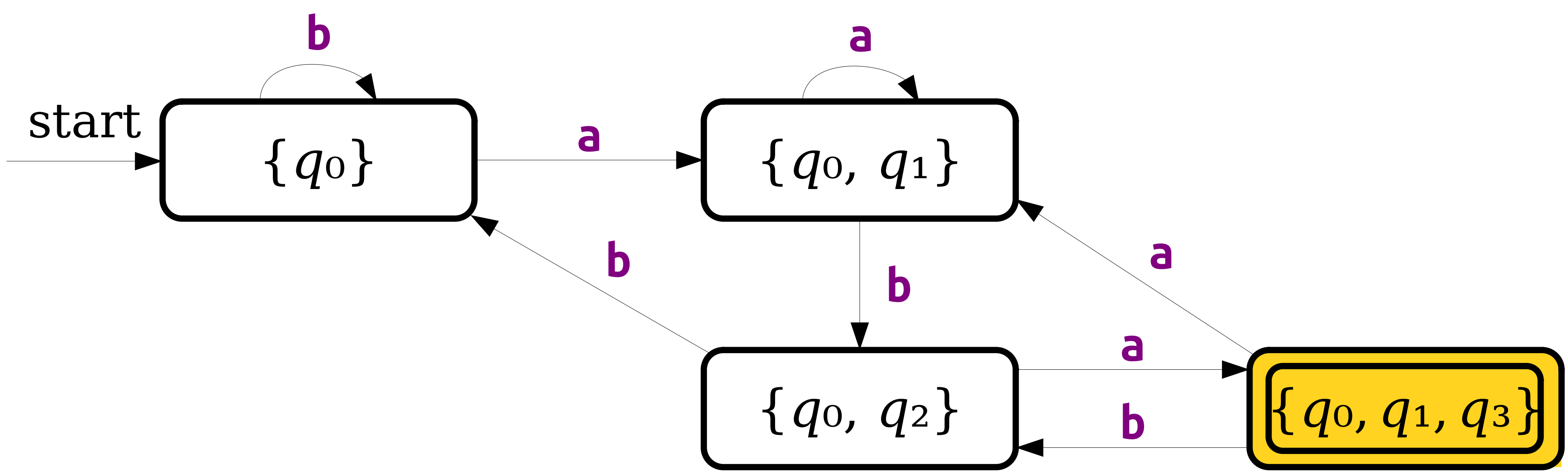

Σ
start
a
b
a
q_0
q_1
q_2
q_3


a b a a b a


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



a	b	a	a	b	a



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 ε-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?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. **The Subset Construction**
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

Part 3

Regular Languages

- A language L is called ***regular*** when there's a DFA D that recognizes L (that is, $\mathscr{L}(D) = L$).
- ***Theorem:*** A language L is regular if and only if there's an NFA N that recognizes it (that is, $\mathscr{L}(N) = L$).
- This fact makes it possible to explore regular languages by considering either DFAs or NFAs.

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. **Regular Languages Revisited**
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What's Next?

Finite Automata

Part 3

Problem Set Six

- Problem Set Five was due today at 1:00 PM.
  - You can use a late day to extend the deadline to Saturday at 1:00 PM.
- Problem Set Six goes out today. It's due next Friday at 1:00 PM.
  - Play around with automata!
  - Explore properties of languages!
  - See some cool applications!

Other Things

- Please read my post on Ed about regrade requests. Regrade requests that don’t conform to the guidelines articulated there will likely be dismissed without review.
- The Grade Cruncher is posted on the course homepage.

Finite Automata

Part 3

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. **Union and Intersection**
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Motivating Example: ***Numbers***

Numbers

- Numbers can be written in many ways:

2718

2,718

2.718×10^3

MMDCCXVIII

二千七百一十八

ב'תשי"ח

൨൭൧൮

ՍՉԺԸ

etc.

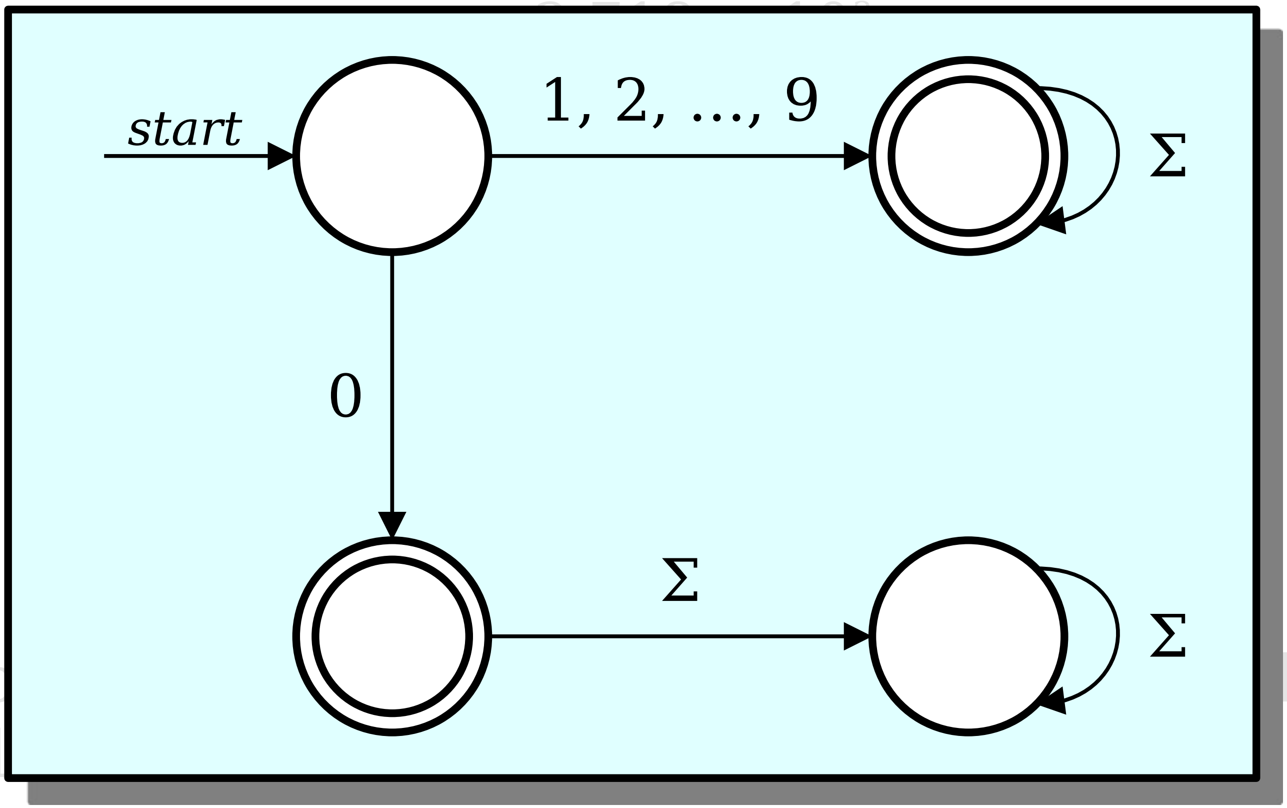
- How would we design a DFA or NFA that checks if a particular string is a number in some numeral system?

Numbers

- Numbers can be written in many ways:

2718

2,718



- How wo … ks if a particul … system?

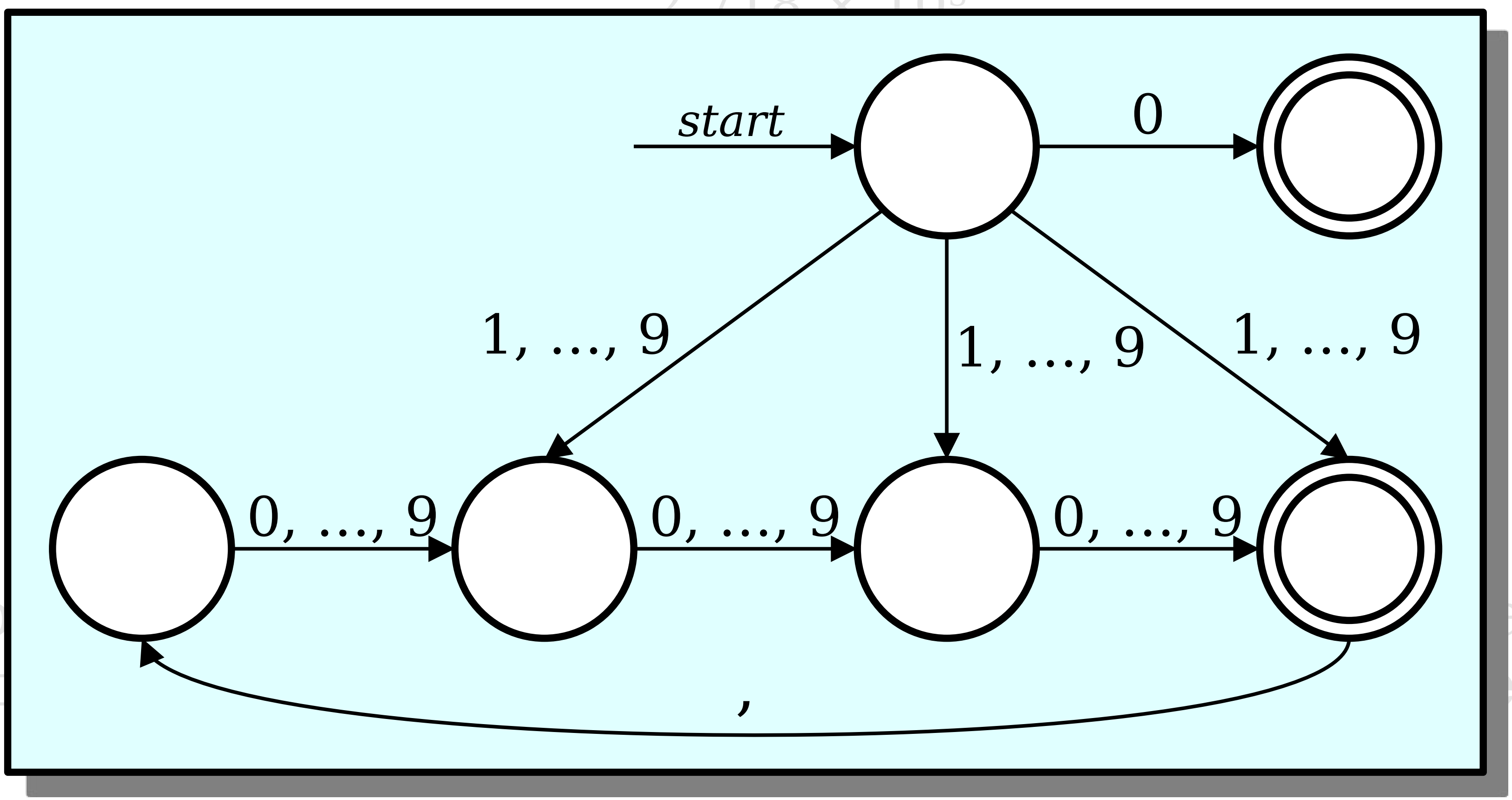
Numbers

- Numbers can be written in many ways:

2718

2,718

$2,718 \times 10^3$



- Ho … a
pa … em?

Practical Question: If we can build a bunch of finite automata that all recognize certain patterns, can we build a single finite automaton that recognizes all of those patterns?

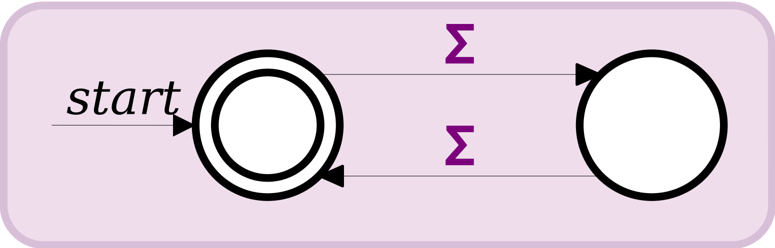
Closure Under Union

- If L_1 and L_2 are languages over the alphabet Σ, the language $\boldsymbol{L_1 \cup L_2}$ is the language of all strings in at least one of the two languages.
- Intuitively, if L_1 and L_2 correspond to languages of strings with one of two different patterns, then $L_1 \cup L_2$ is the language of strings with at least one of those patterns.
- ***Theorem:*** If L_1 and L_2 are regular, so is $L_1 \cup L_2$.

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

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

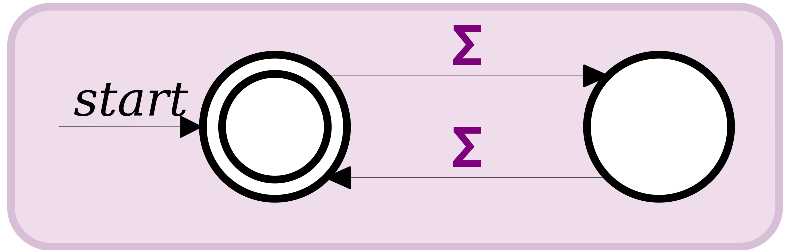
Construct an NFA for $L_1 \cup L_2$.



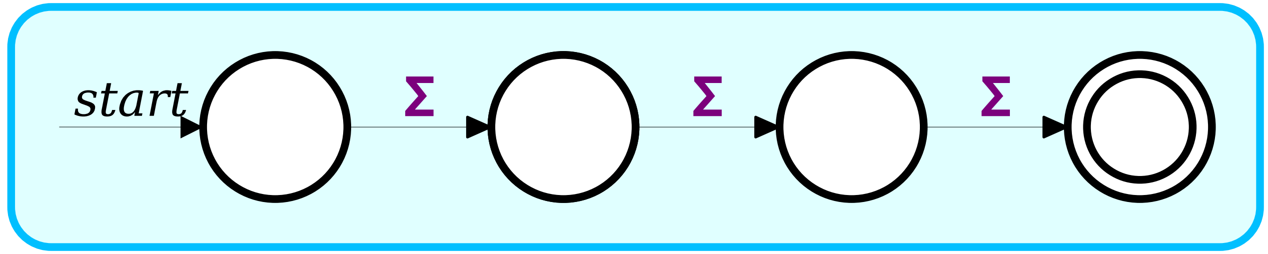
DFA for L_1

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

Construct an NFA for $L_1 \cup L_2$.



DFA for L_1

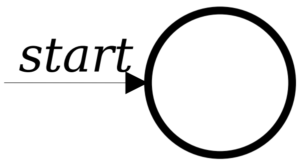


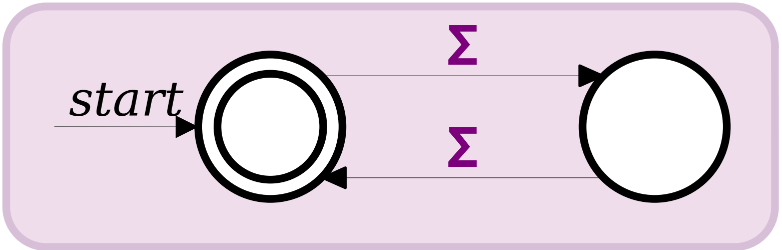
NFA for L_2

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

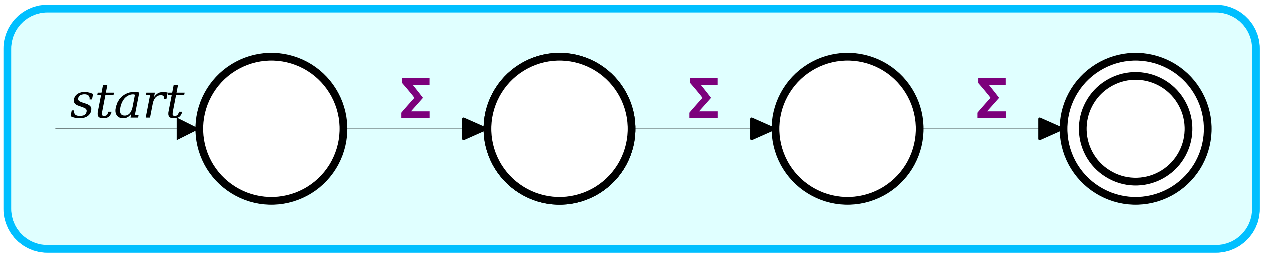
$L_2 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.





DFA for L_1

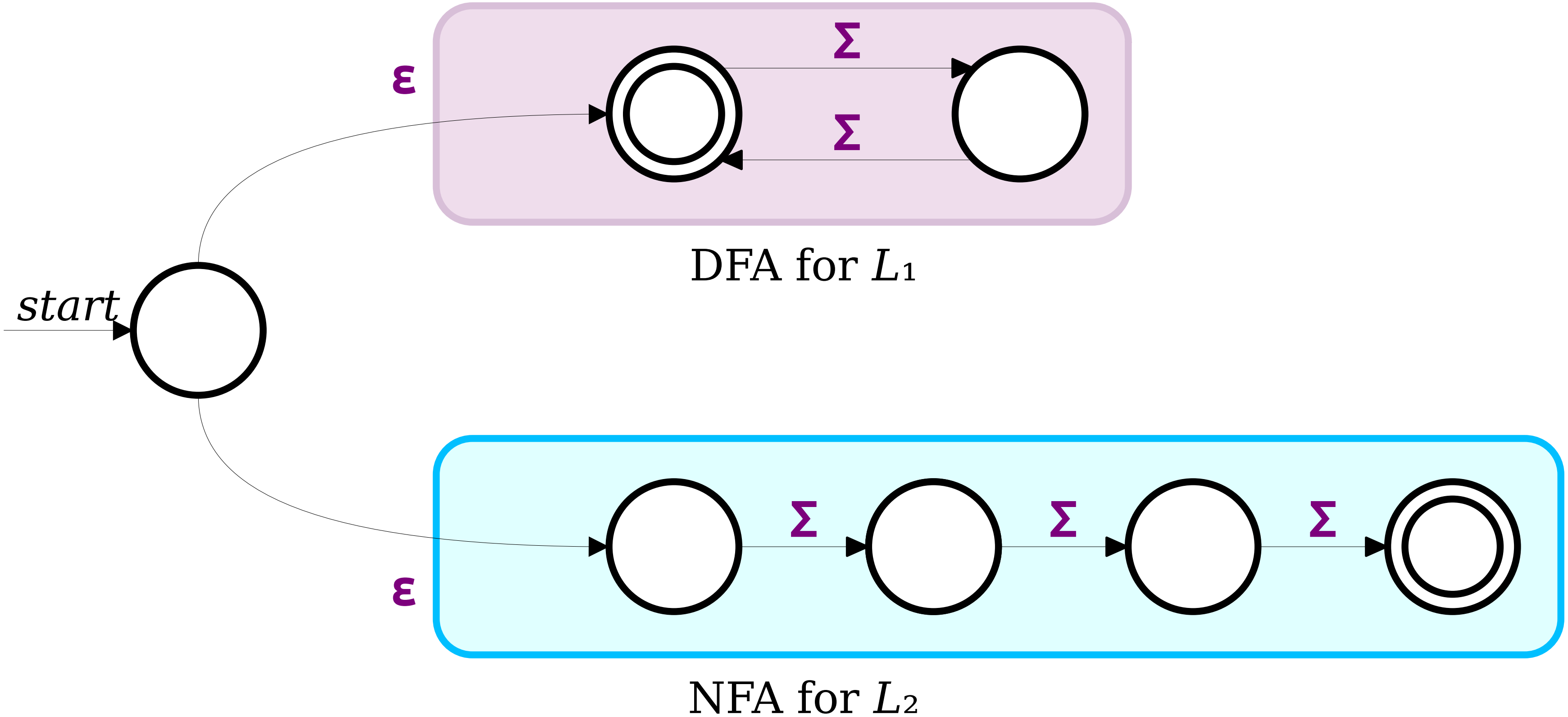


NFA for L_2

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

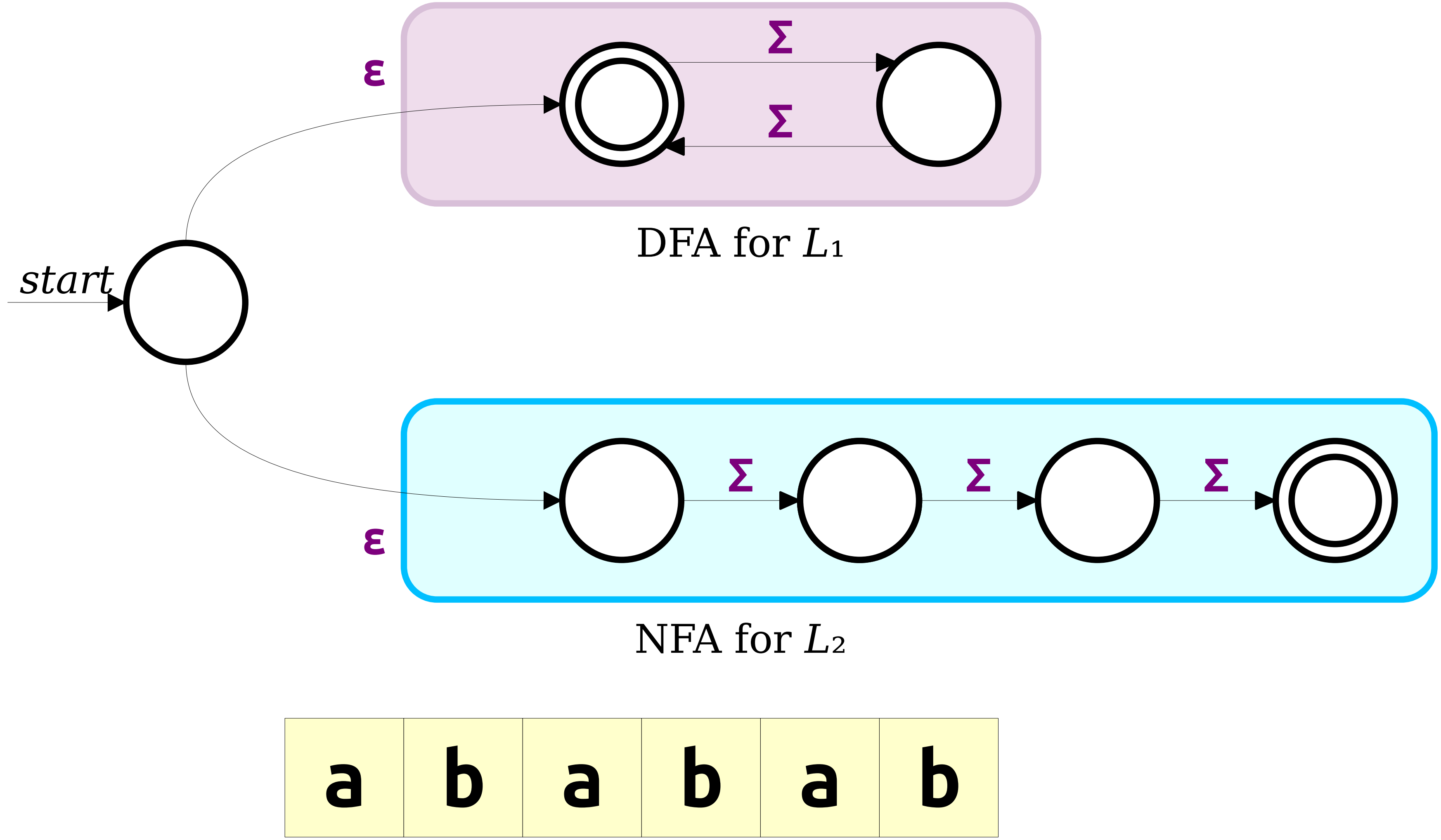
$L_2 = \{\ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.



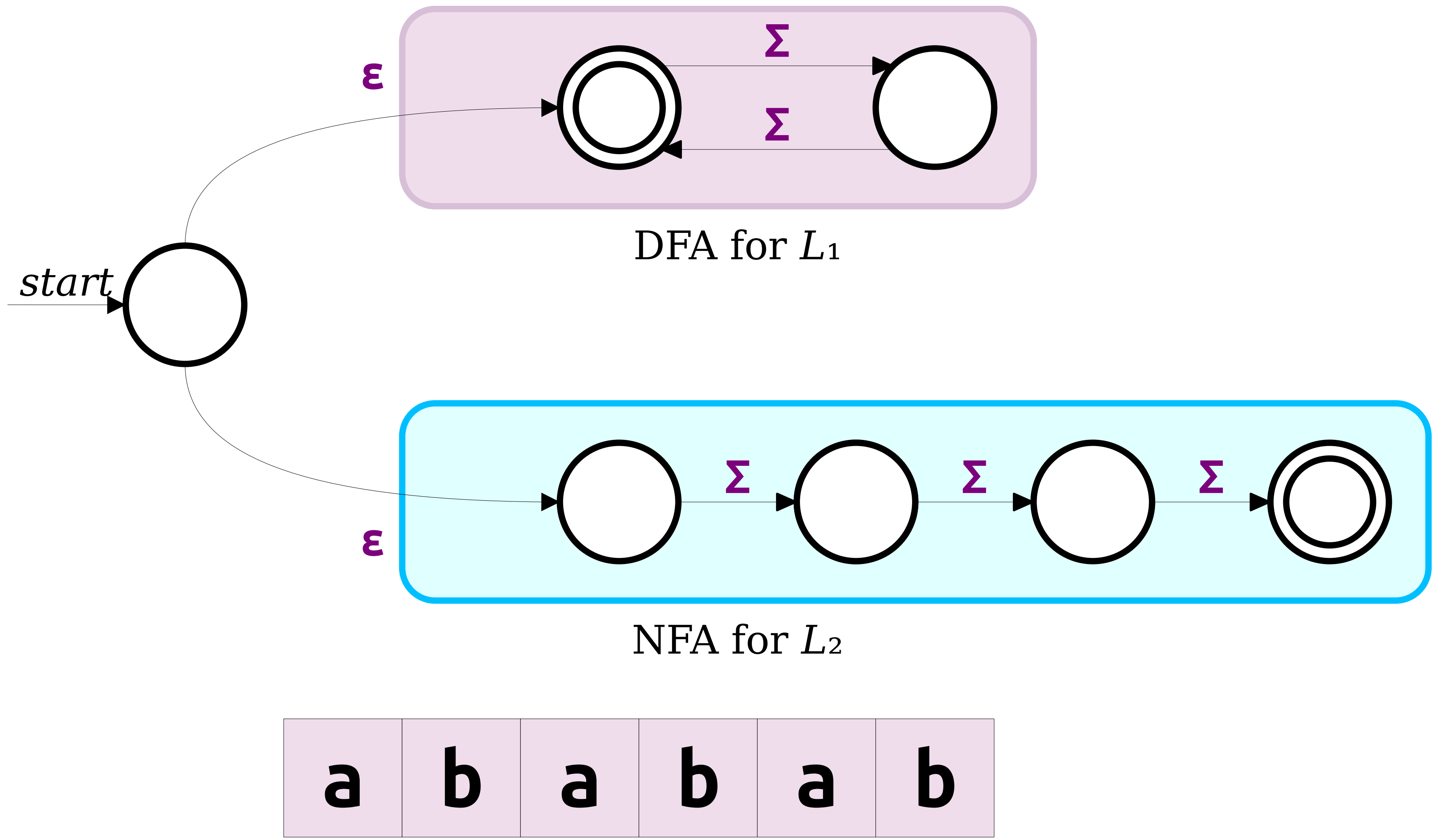
$L_1 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has even length } \}$
$L_2 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.



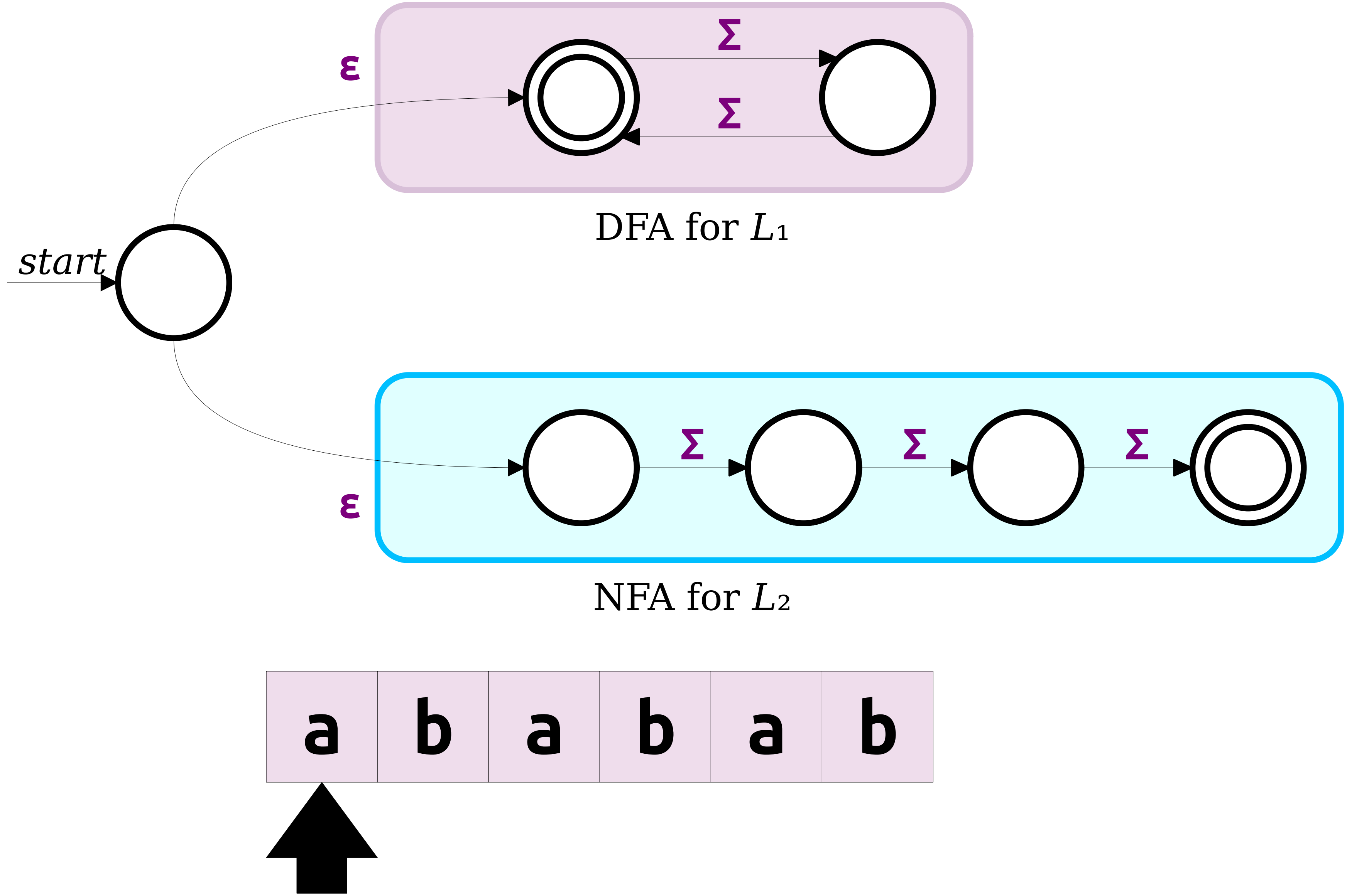
$L_1 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has even length } \}$
$L_2 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.



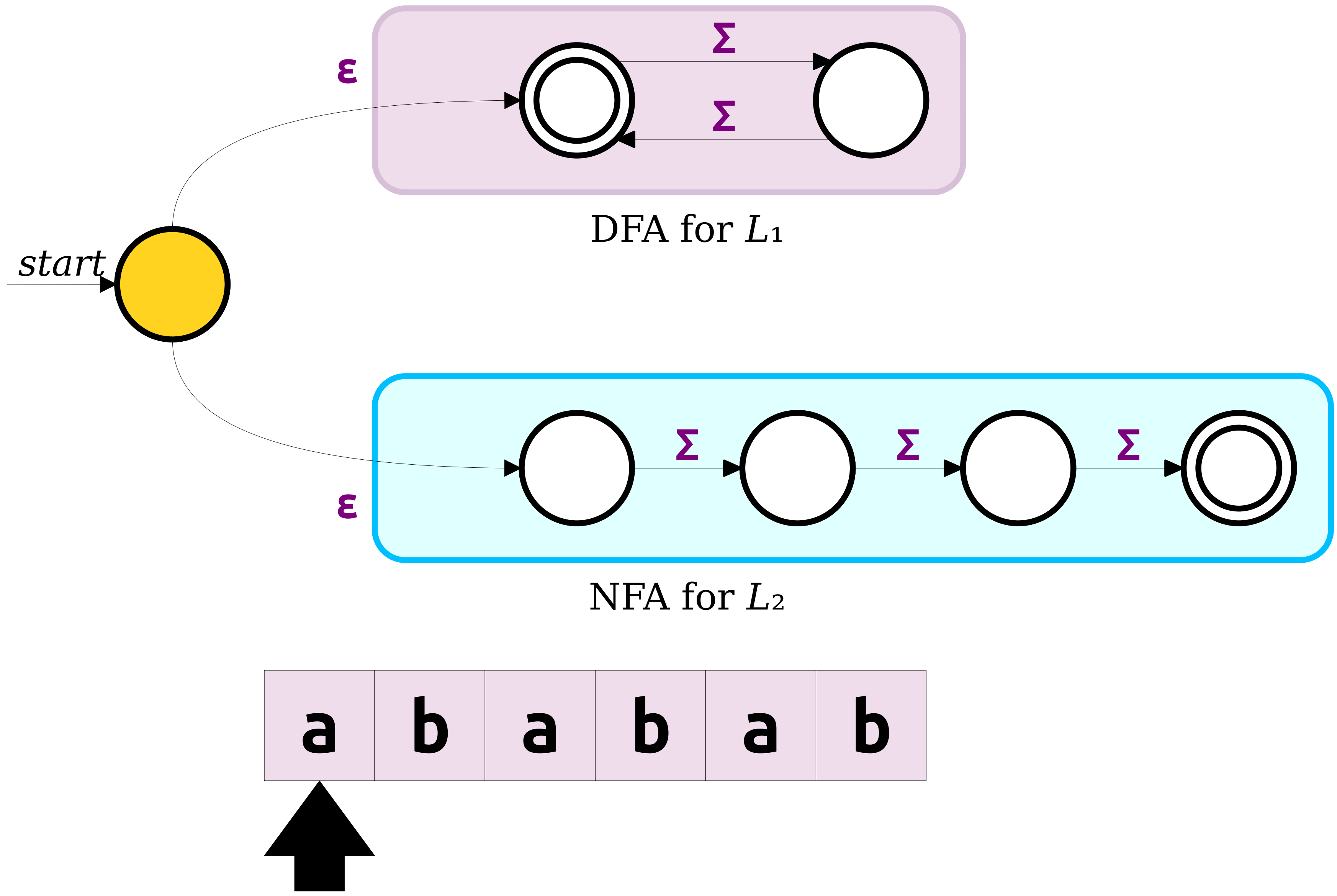
$L_1 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has even length } \}$
$L_2 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.



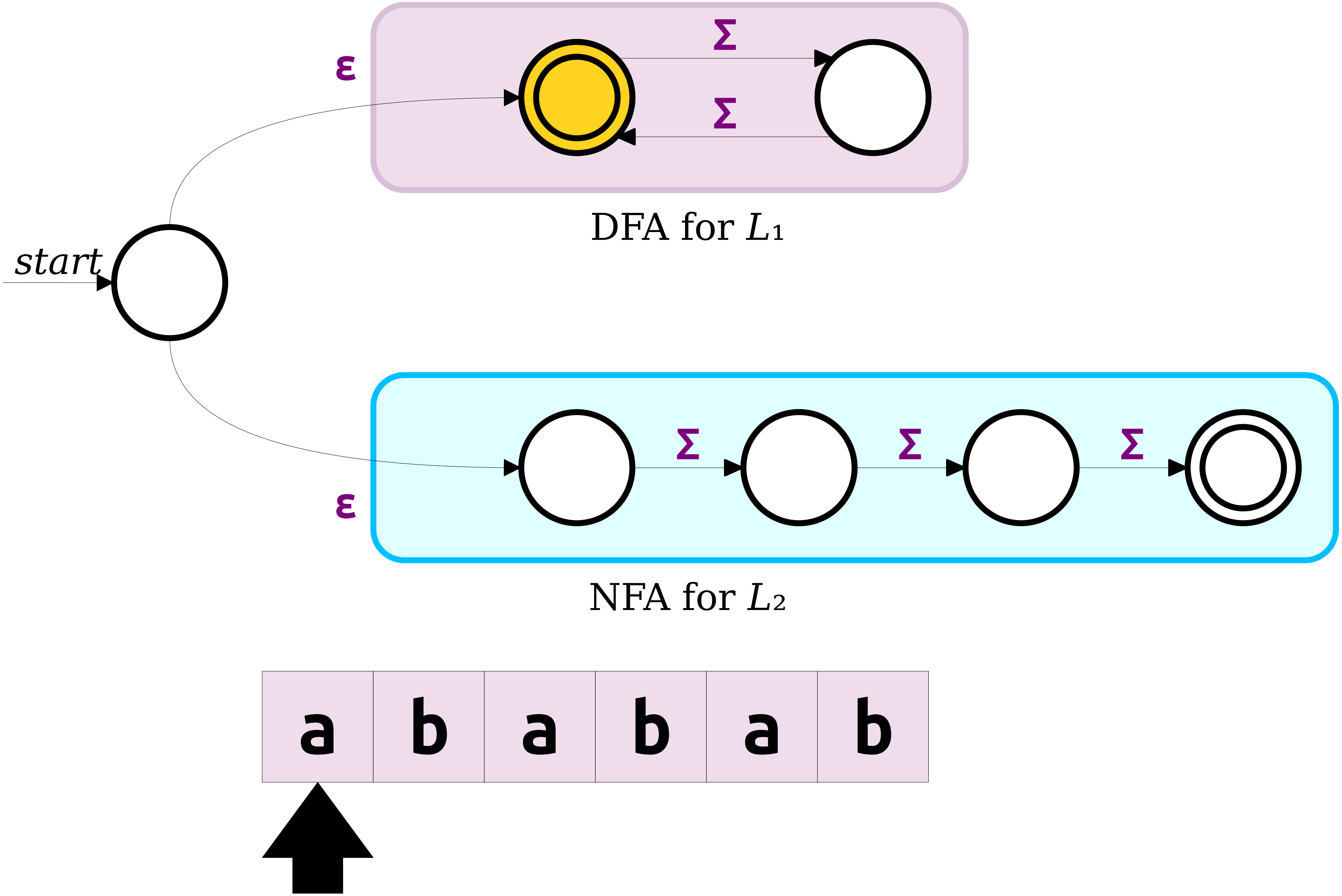
$L_1 = \{ w \in \{a, b\}^* \mid w$ has even length $\}$
$L_2 = \{ w \in \{a, b\}^* \mid w$ has length exactly three $\}$

Construct an NFA for $L_1 \cup L_2$.



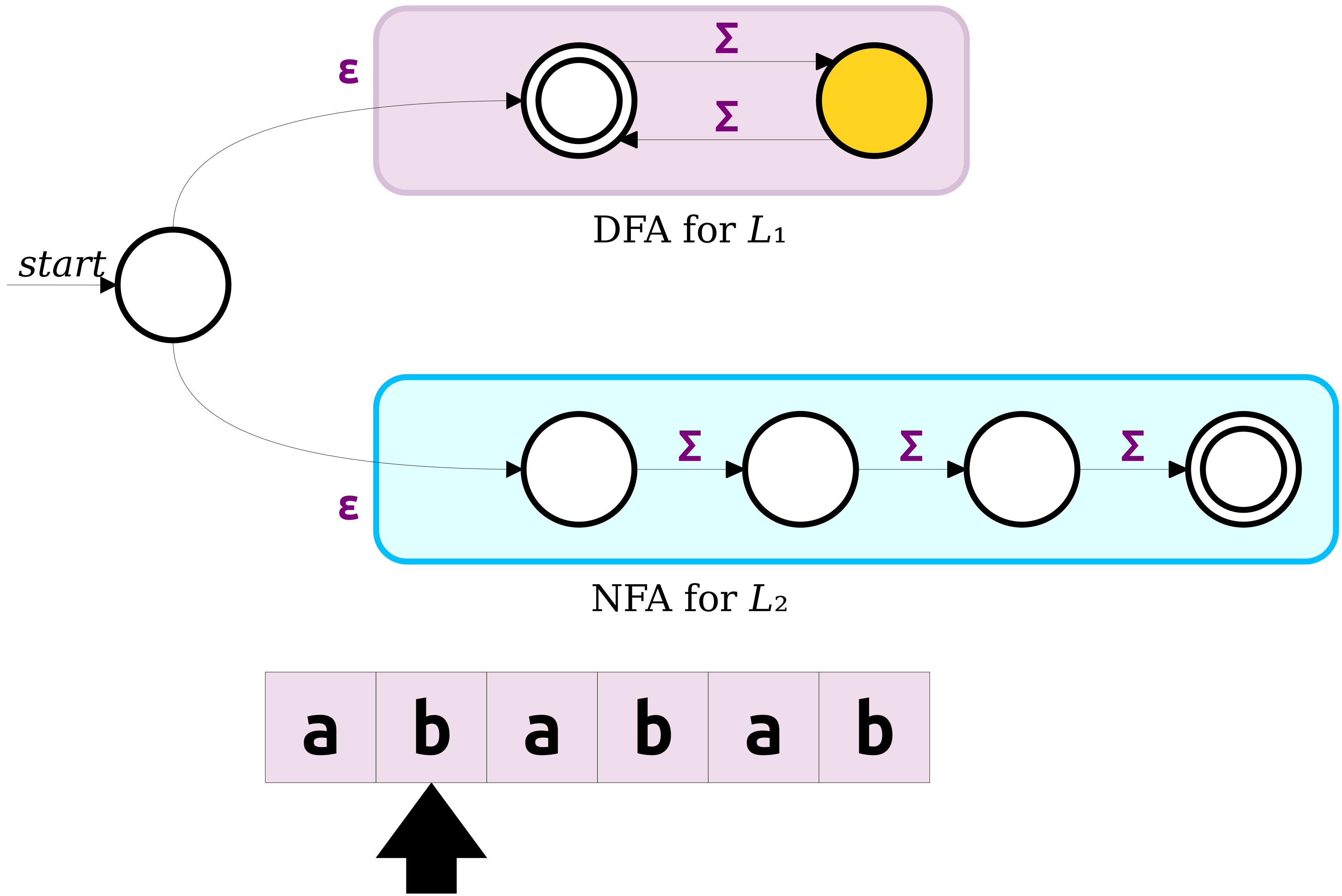
$L_1 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has even length } \}$
$L_2 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.



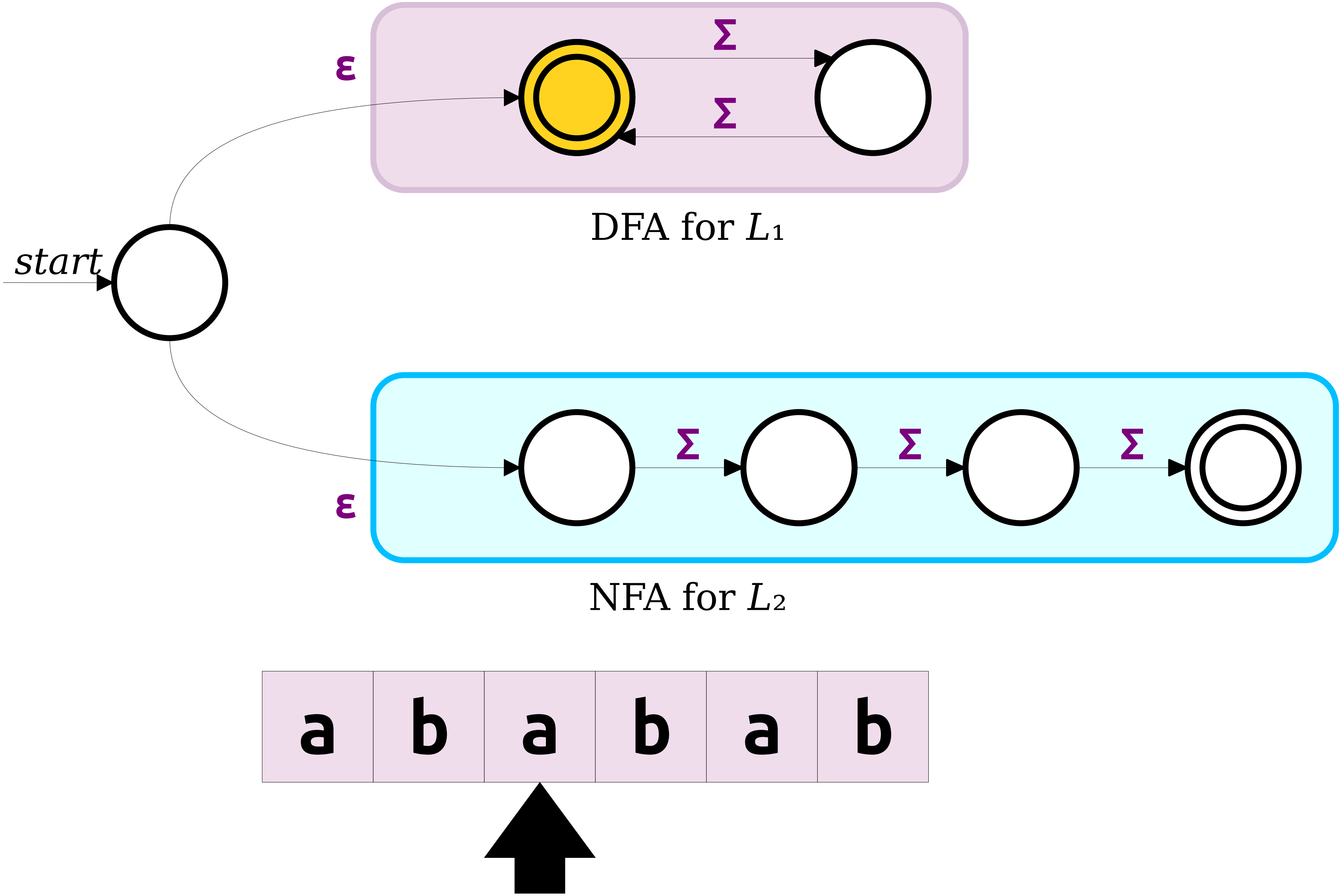
$L_1 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has even length } \}$
$L_2 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.



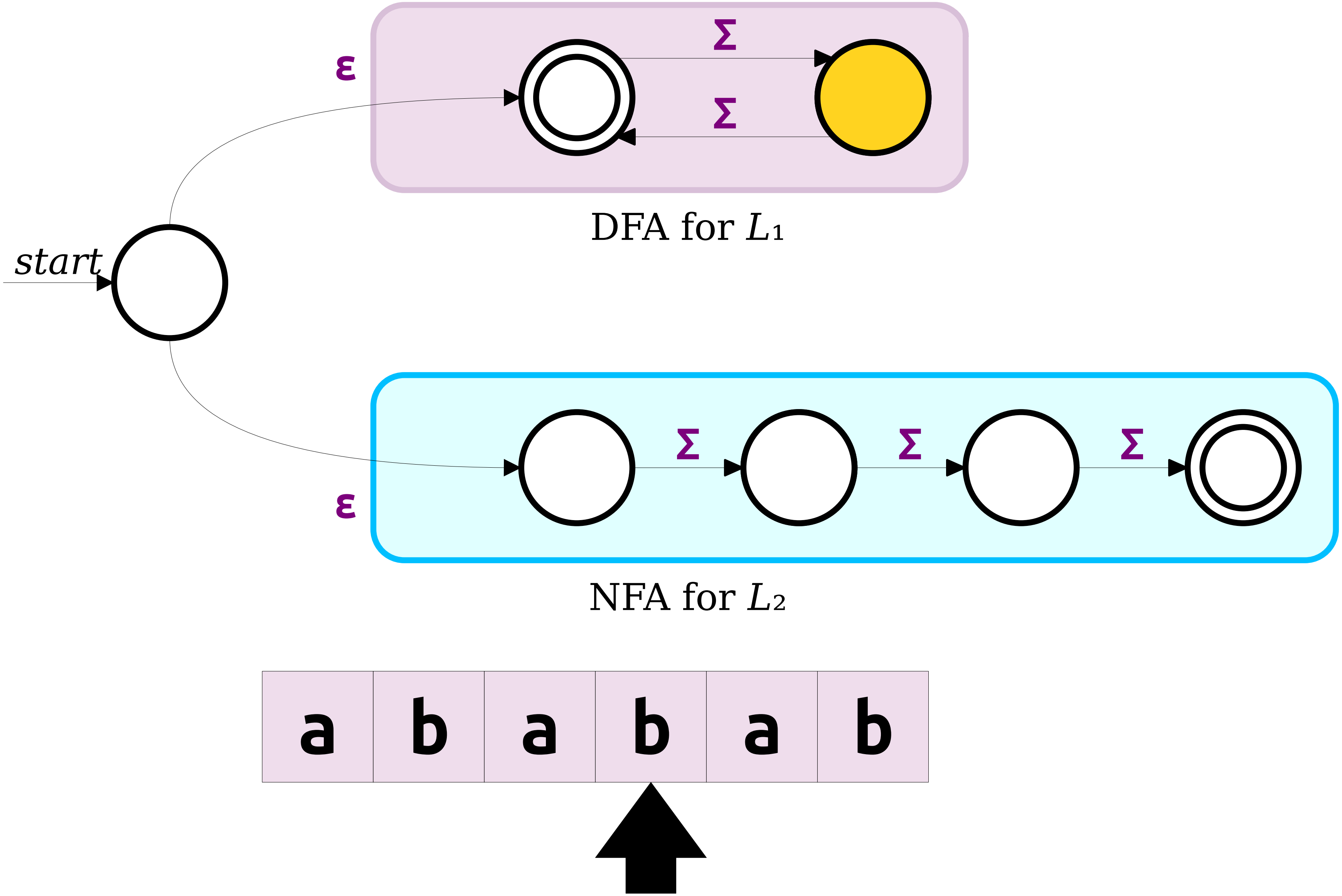
$L_1 = \{ w \in \{a, b\}^* \mid w$ has even length $\}$
$L_2 = \{ w \in \{a, b\}^* \mid w$ has length exactly three $\}$

Construct an NFA for $L_1 \cup L_2$.



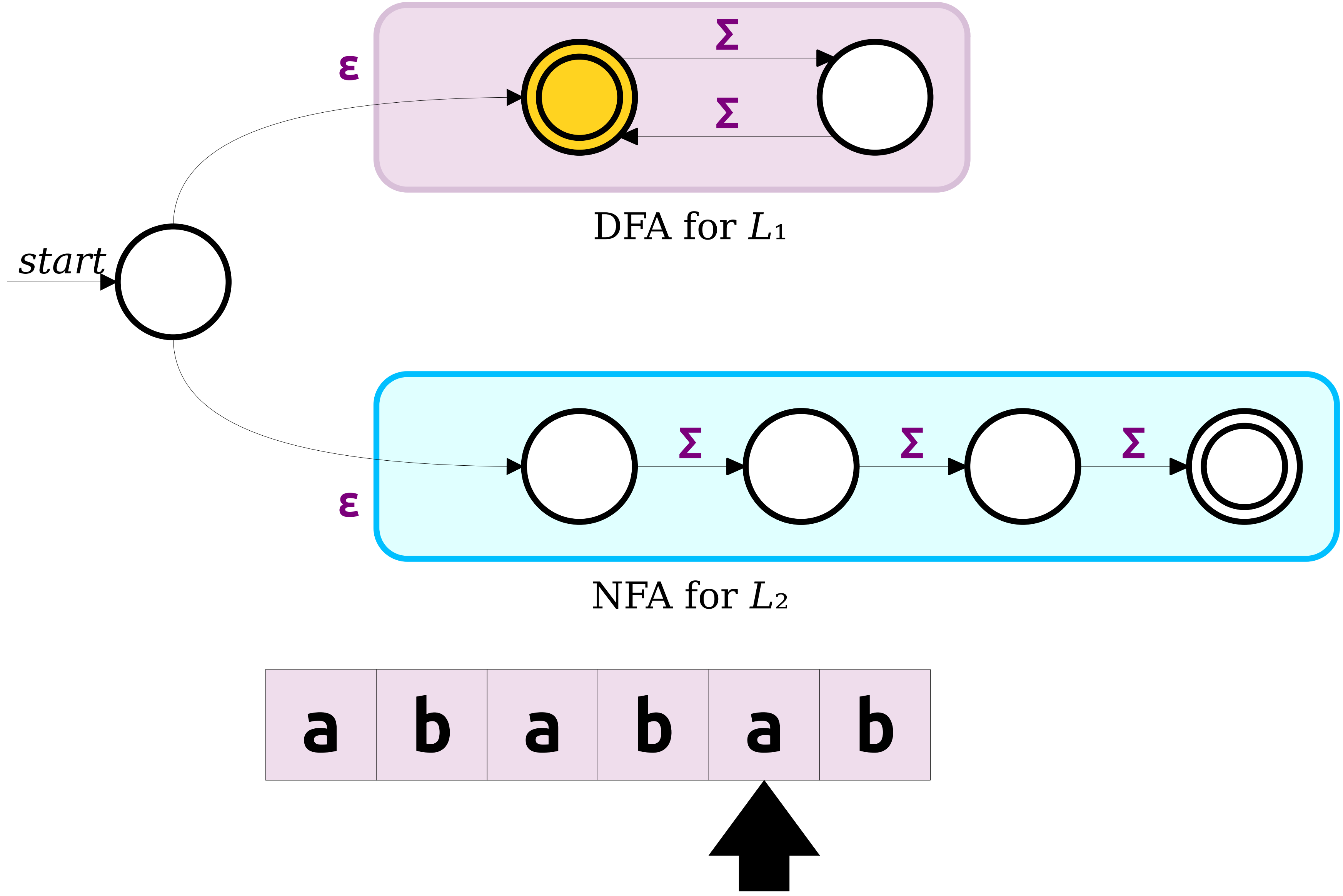
L_1 = { $w \in$ {**a**, **b**}* | w has even length }
L_2 = { $w \in$ {**a**, **b**}* | w has length exactly three }

Construct an NFA for $L_1 \cup L_2$.



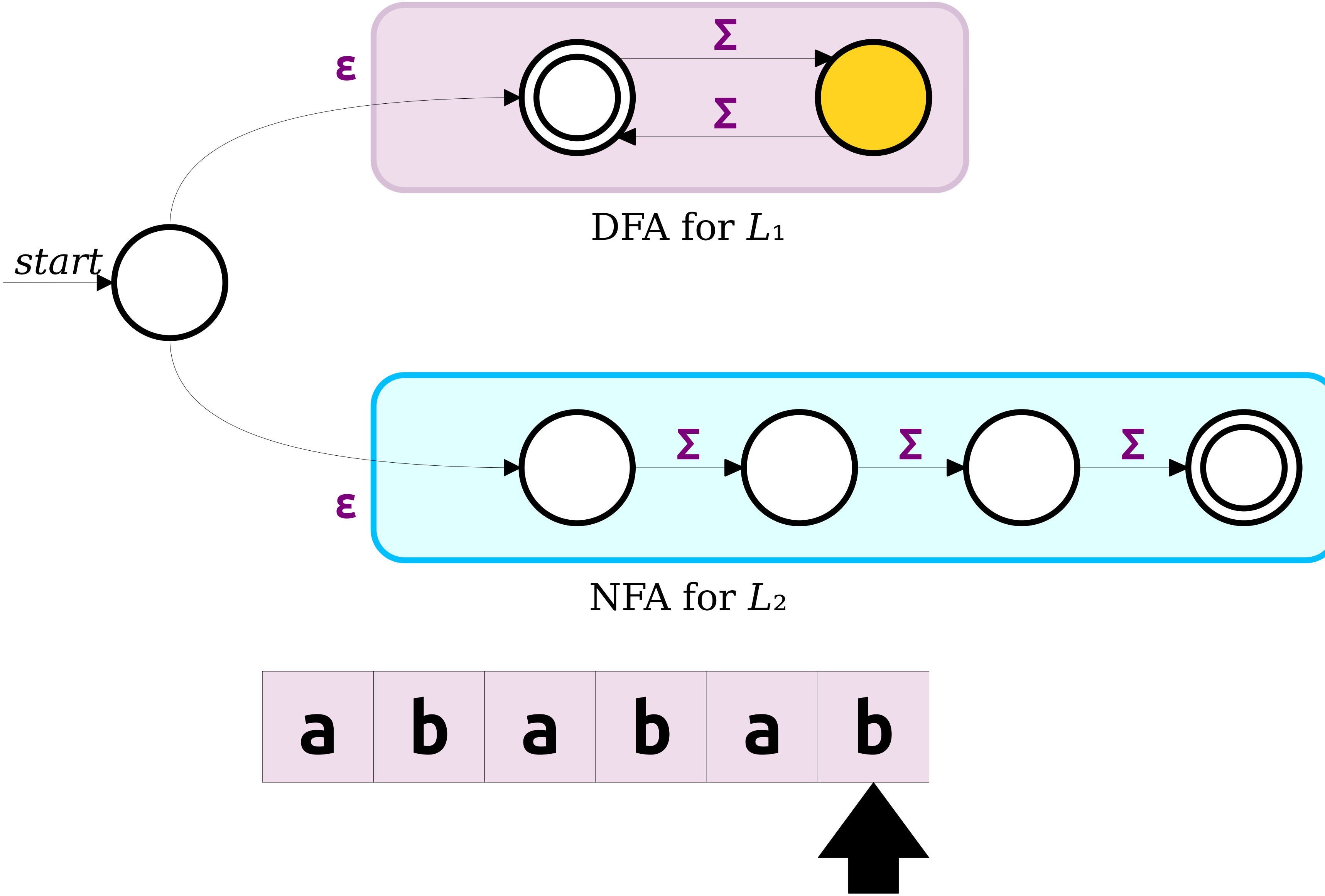
L_1 = { w ∈ {**a**, **b**}* | w has even length }
L_2 = { w ∈ {**a**, **b**}* | w has length exactly three }

Construct an NFA for $L_1 \cup L_2$.



L_1 = { w ∈ {**a**, **b**}* | w has even length }
L_2 = { w ∈ {**a**, **b**}* | w has length exactly three }

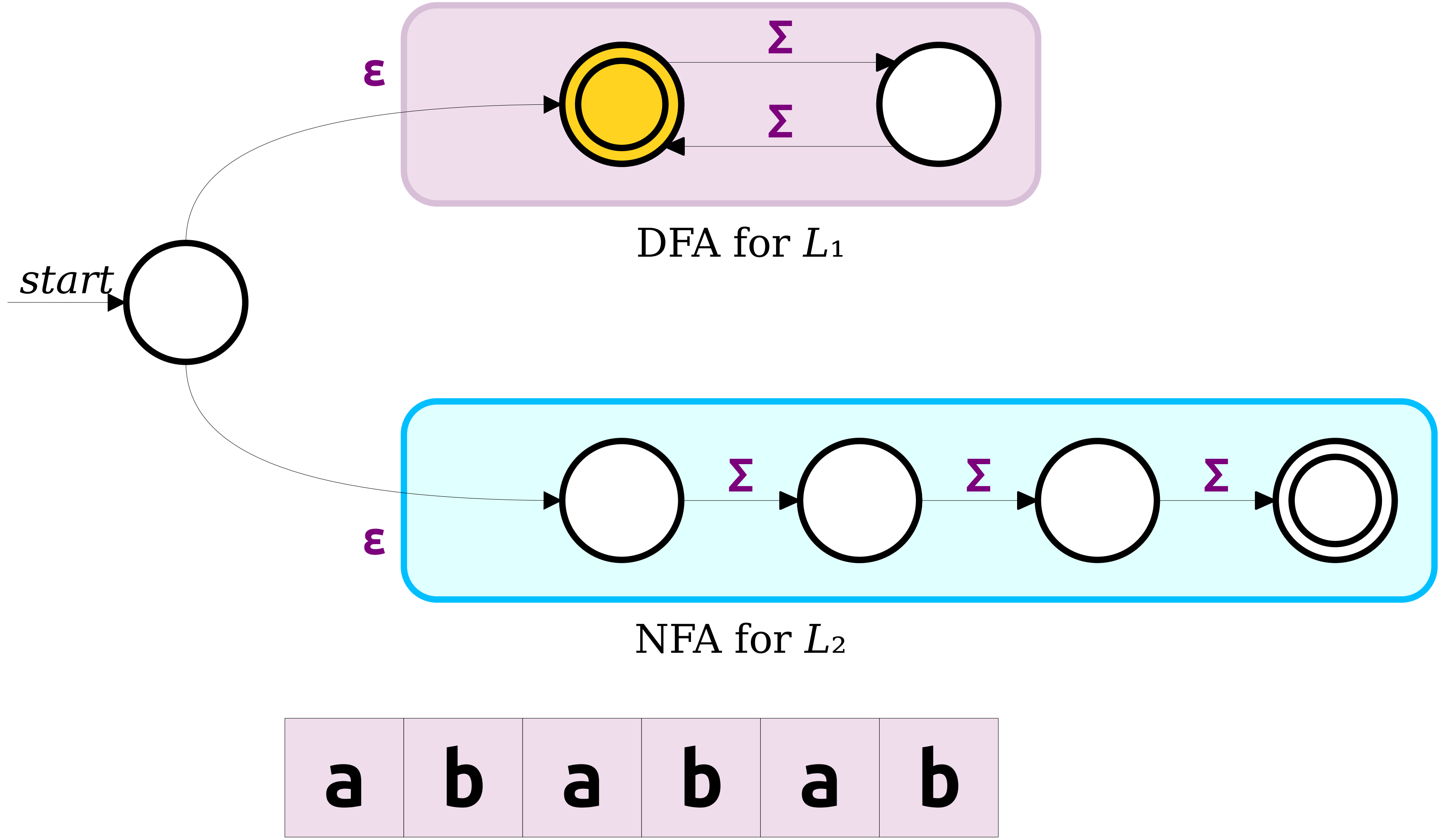
Construct an NFA for $L_1 \cup L_2$.



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

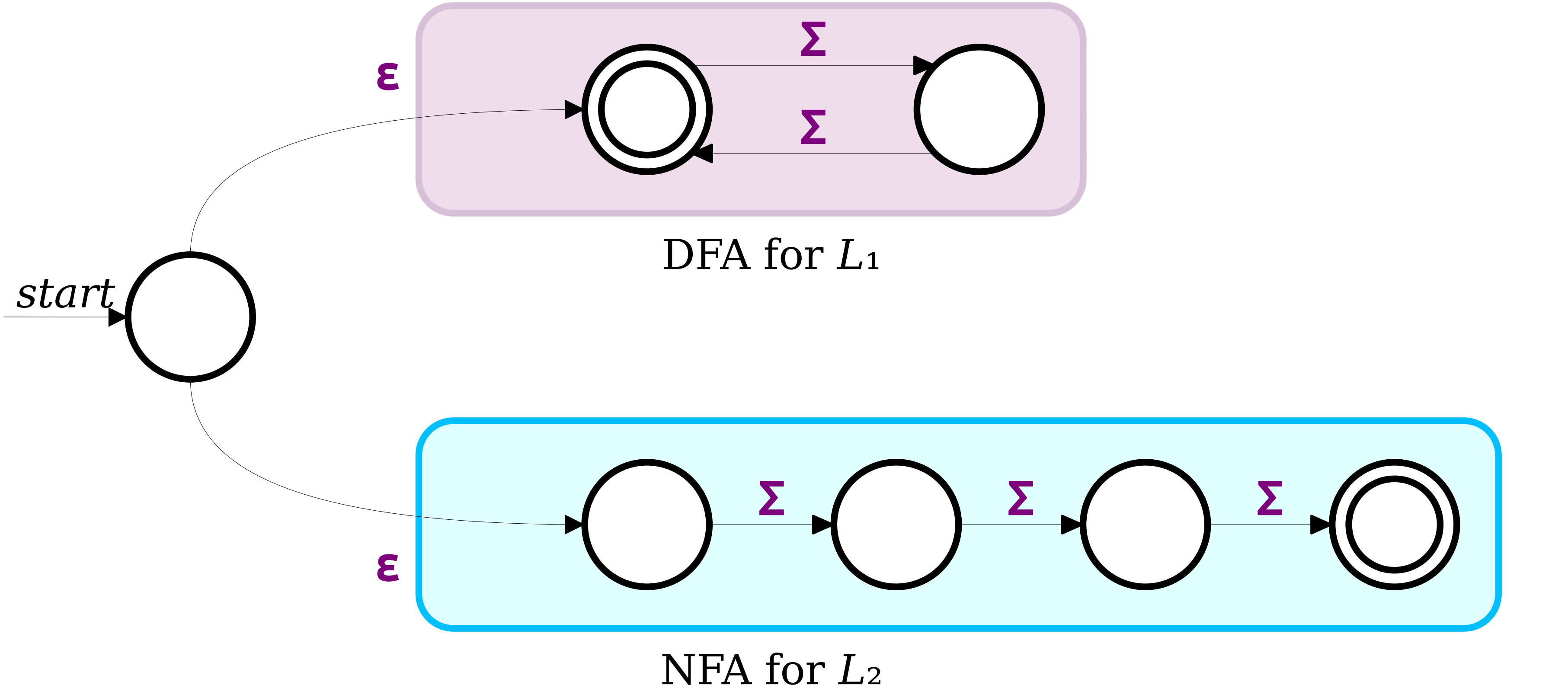
$L_2 = \{ w \in \{$**a**, **b**$\}^* \mid w$ has length exactly three $\}$

Construct an NFA for $L_1 \cup L_2$.



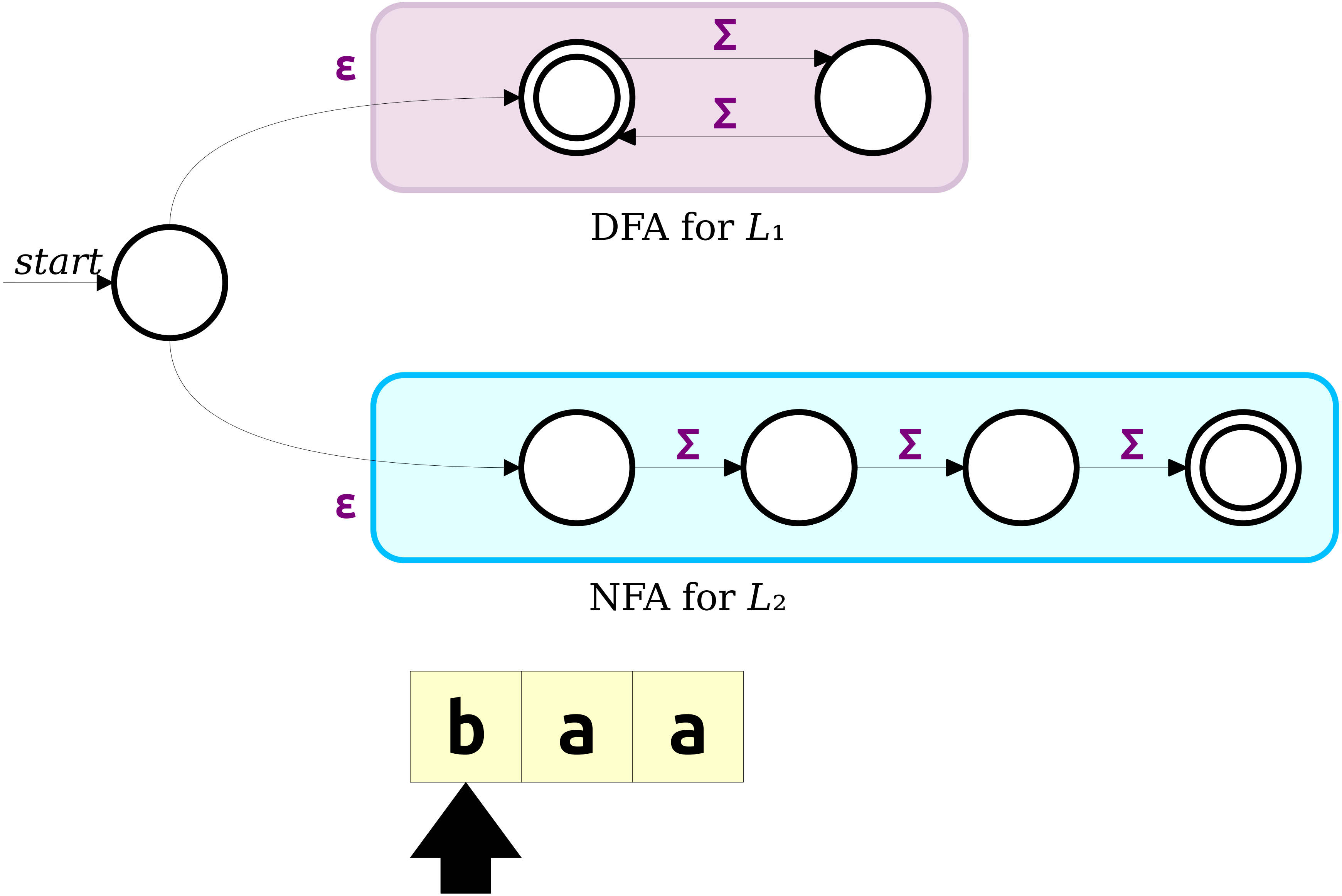
L_1 = { $w \in$ {a, b}* | w has even length }
L_2 = { $w \in$ {a, b}* | w has length exactly three }

Construct an NFA for $L_1 \cup L_2$.



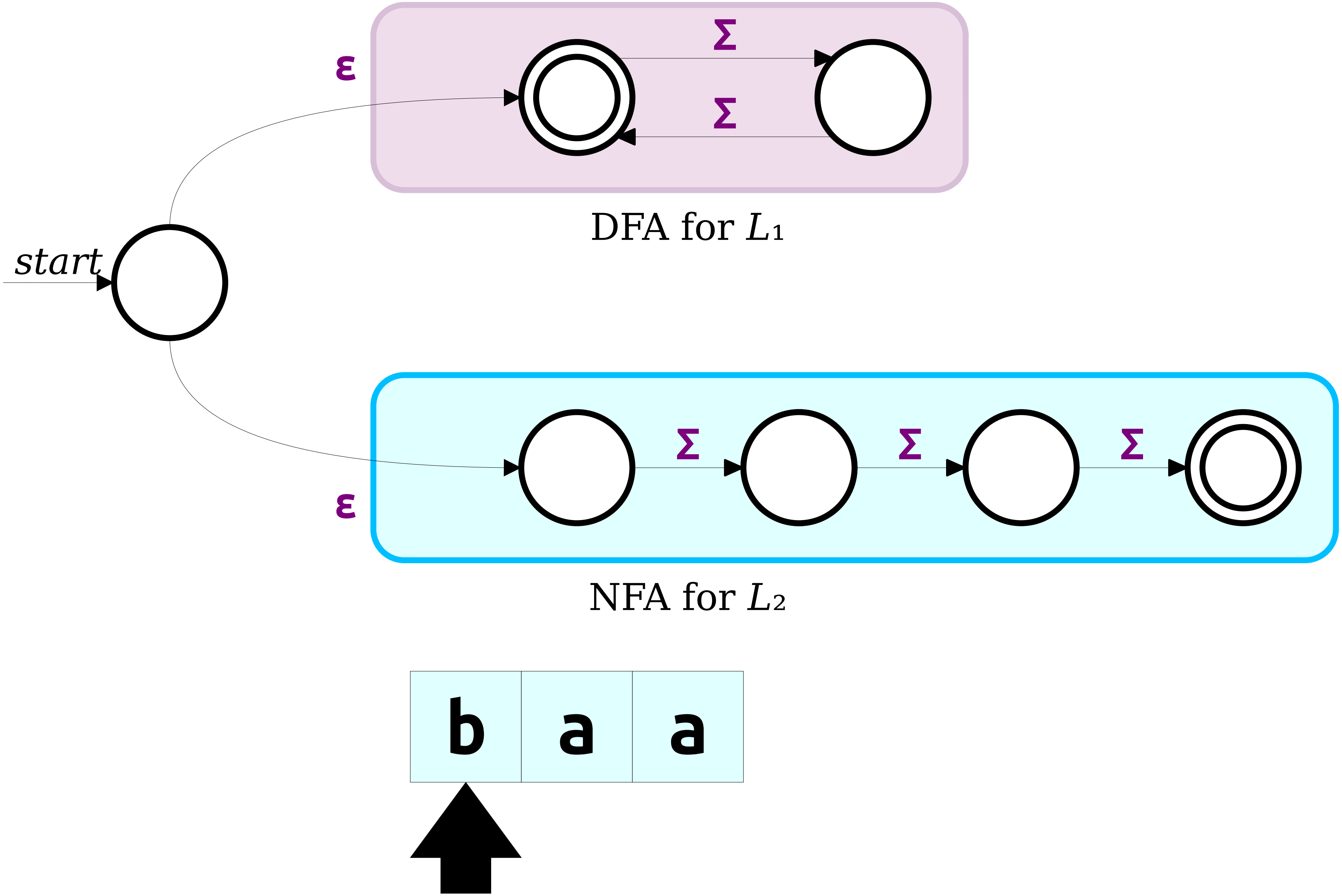
$L_1 = \{ w \in \{$**a**, **b**$\}^* \mid w$ has even length $\}$
$L_2 = \{ w \in \{$**a**, **b**$\}^* \mid w$ has length exactly three $\}$

Construct an NFA for $L_1 \cup L_2$.



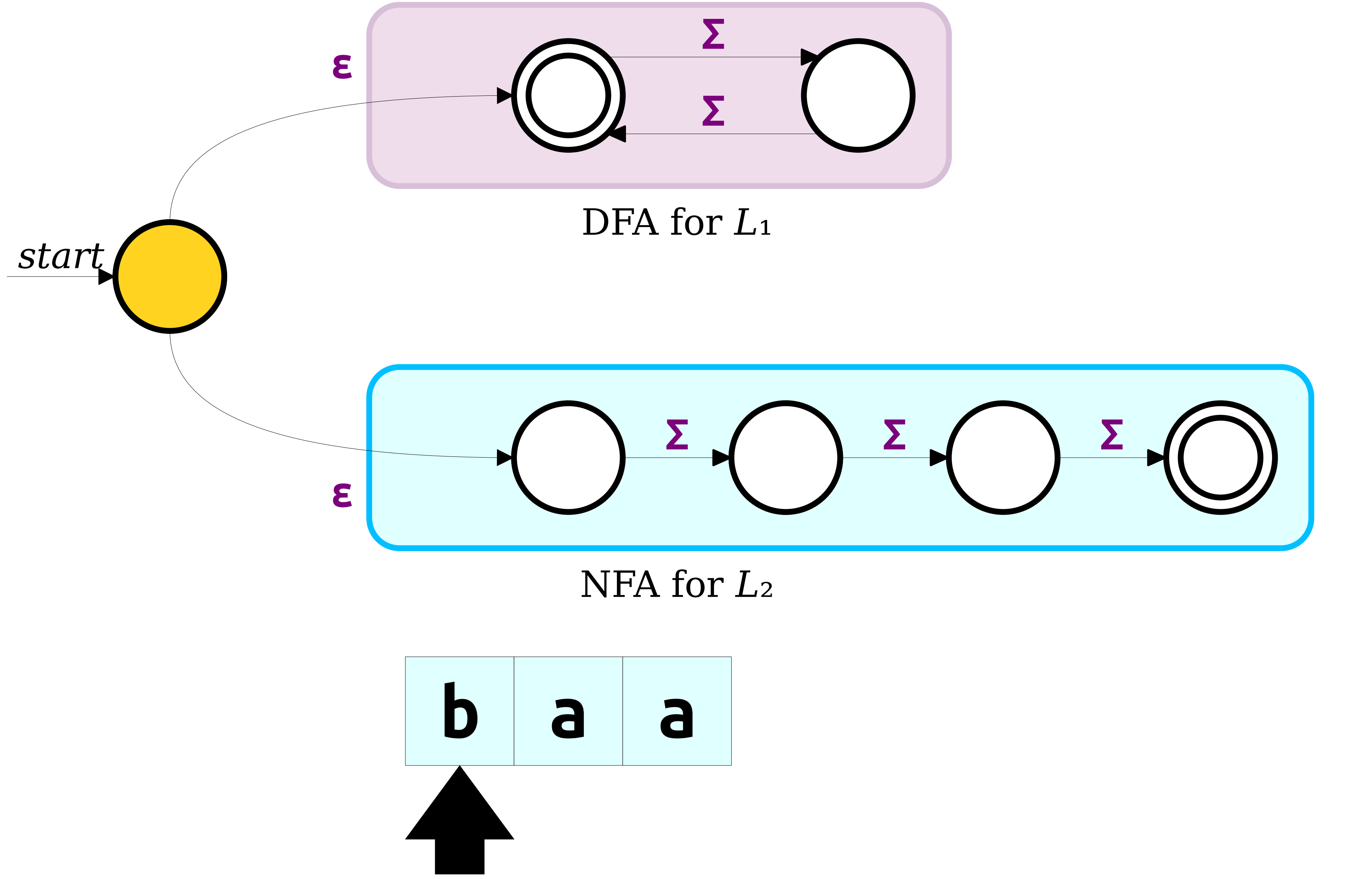
L_1 = { w ∈ {a, b}* | w has even length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

Construct an NFA for $L_1 \cup L_2$.



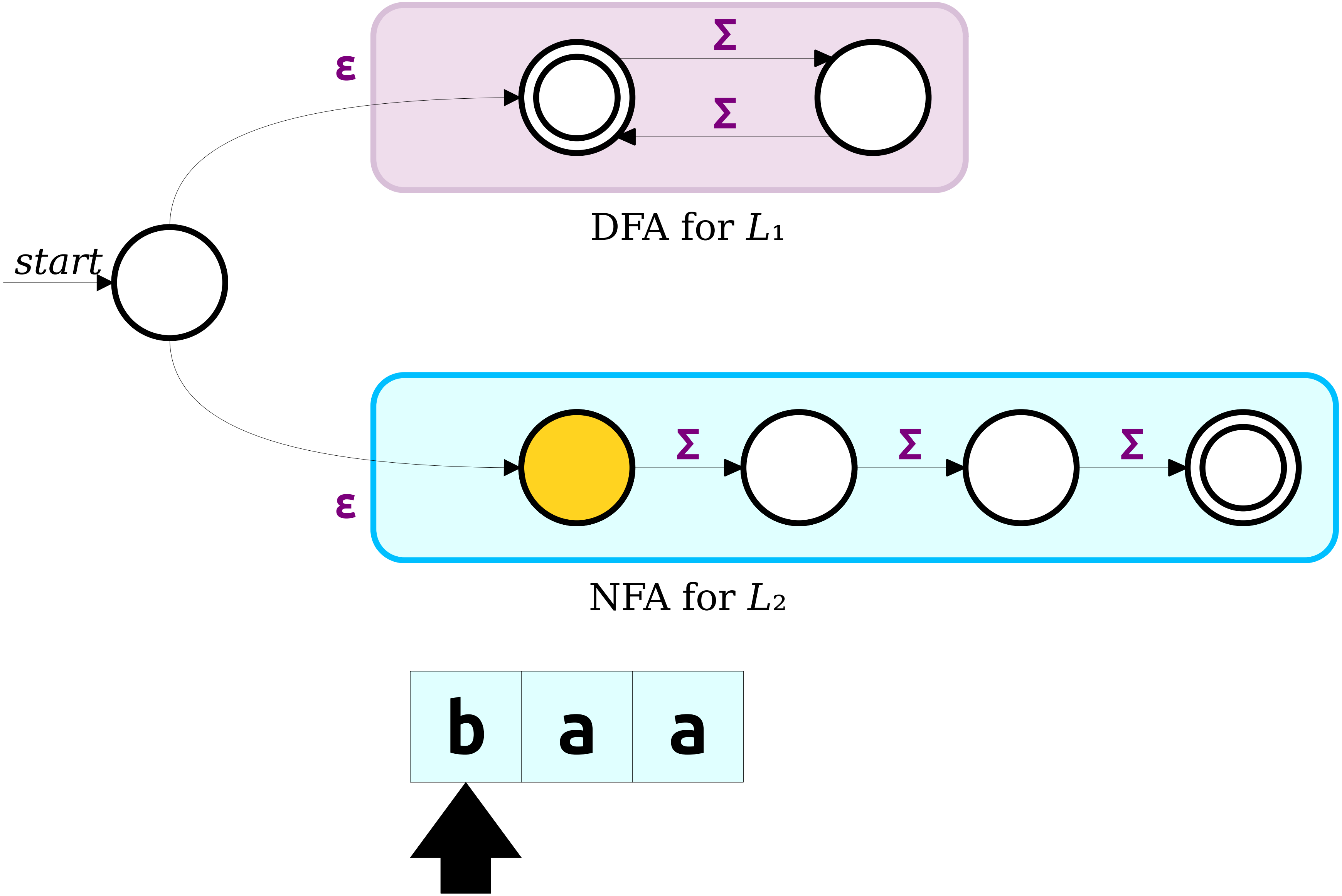
L_1 = { w ∈ {**a**, **b**}* | w has even length }
L_2 = { w ∈ {**a**, **b**}* | w has length exactly three }

Construct an NFA for $L_1 \cup L_2$.



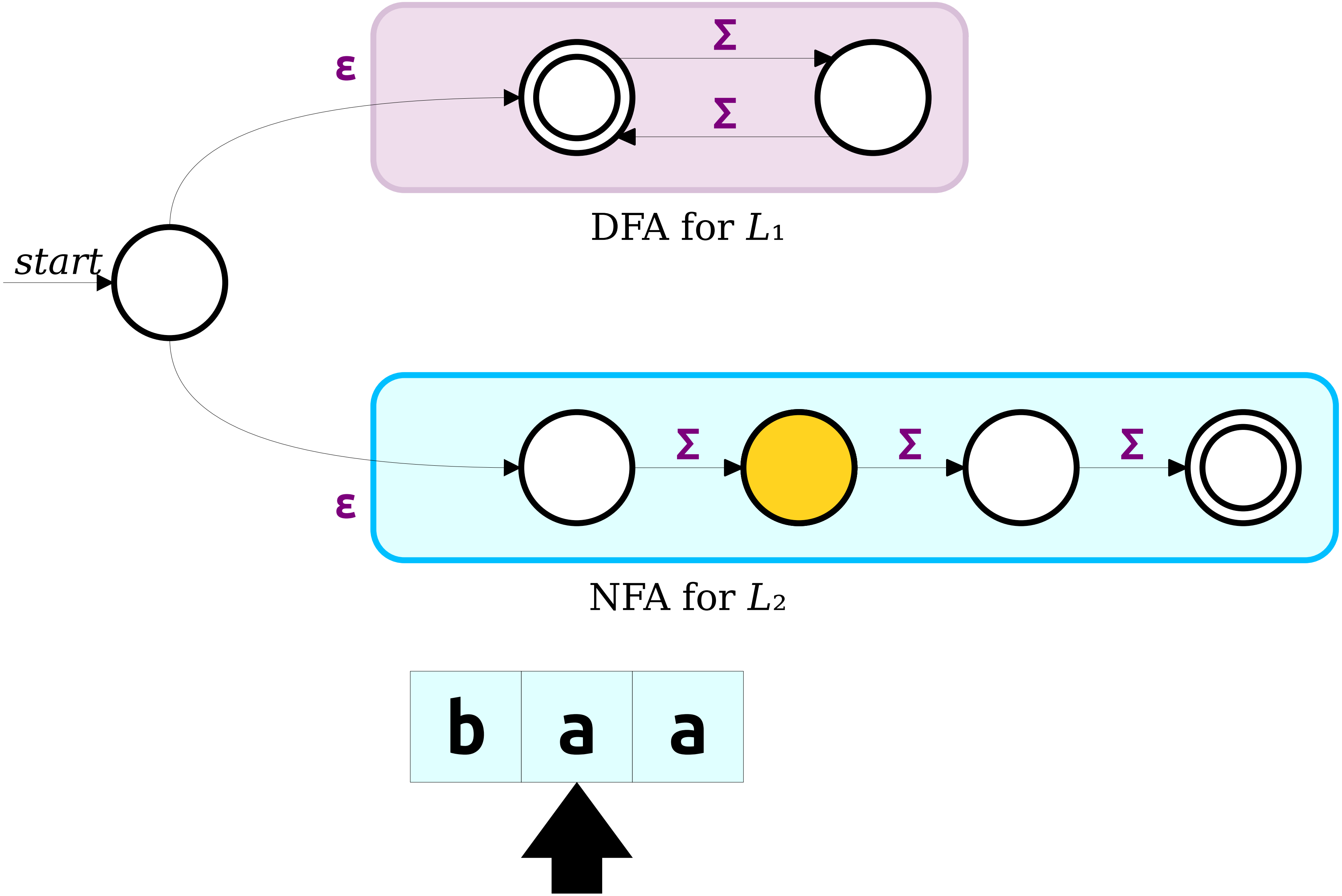
$L_1 = \{\ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has even length } \}$
$L_2 = \{\ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.



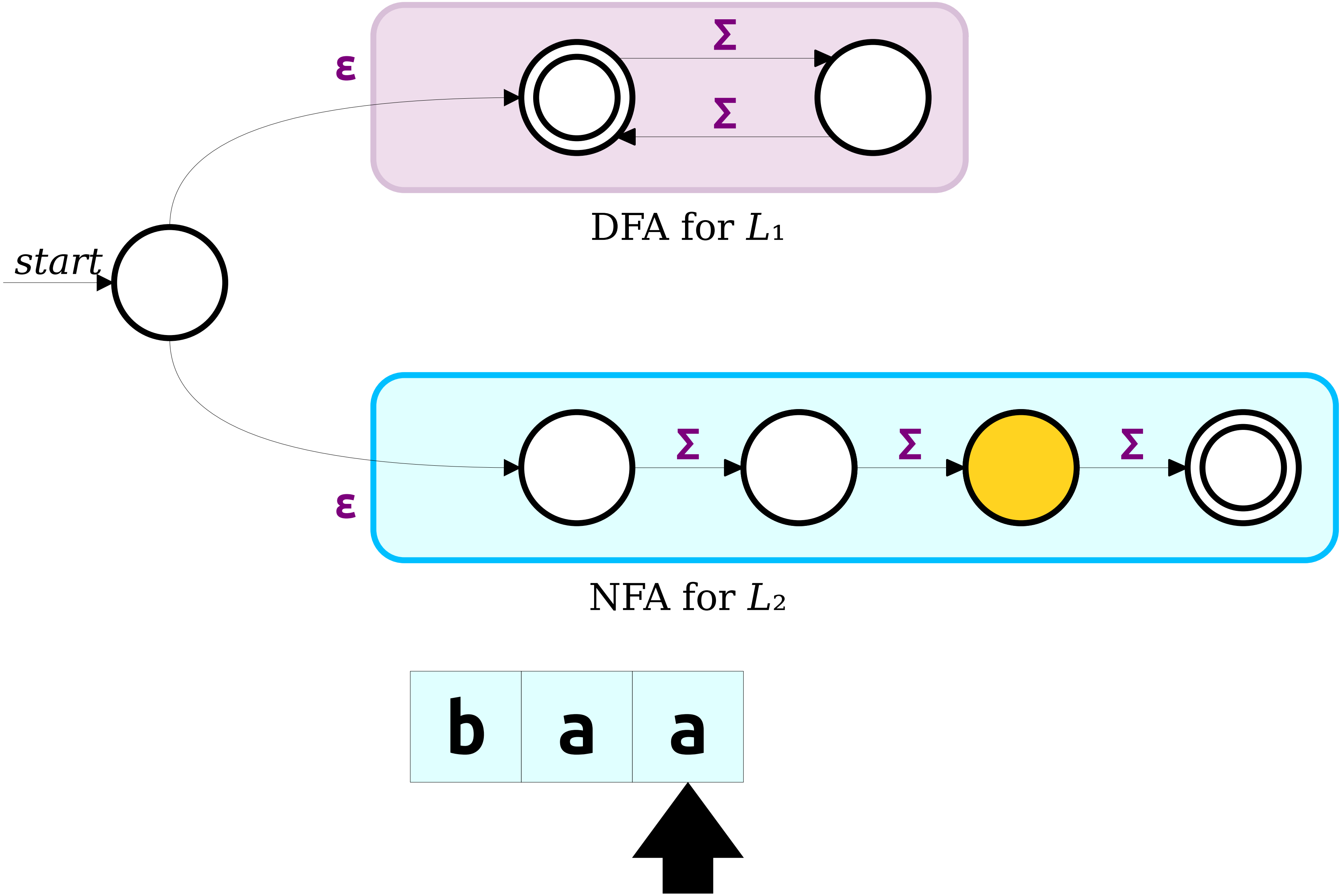
$L_1 = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ has even length } \}$
$L_2 = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for $L_1 \cup L_2$.



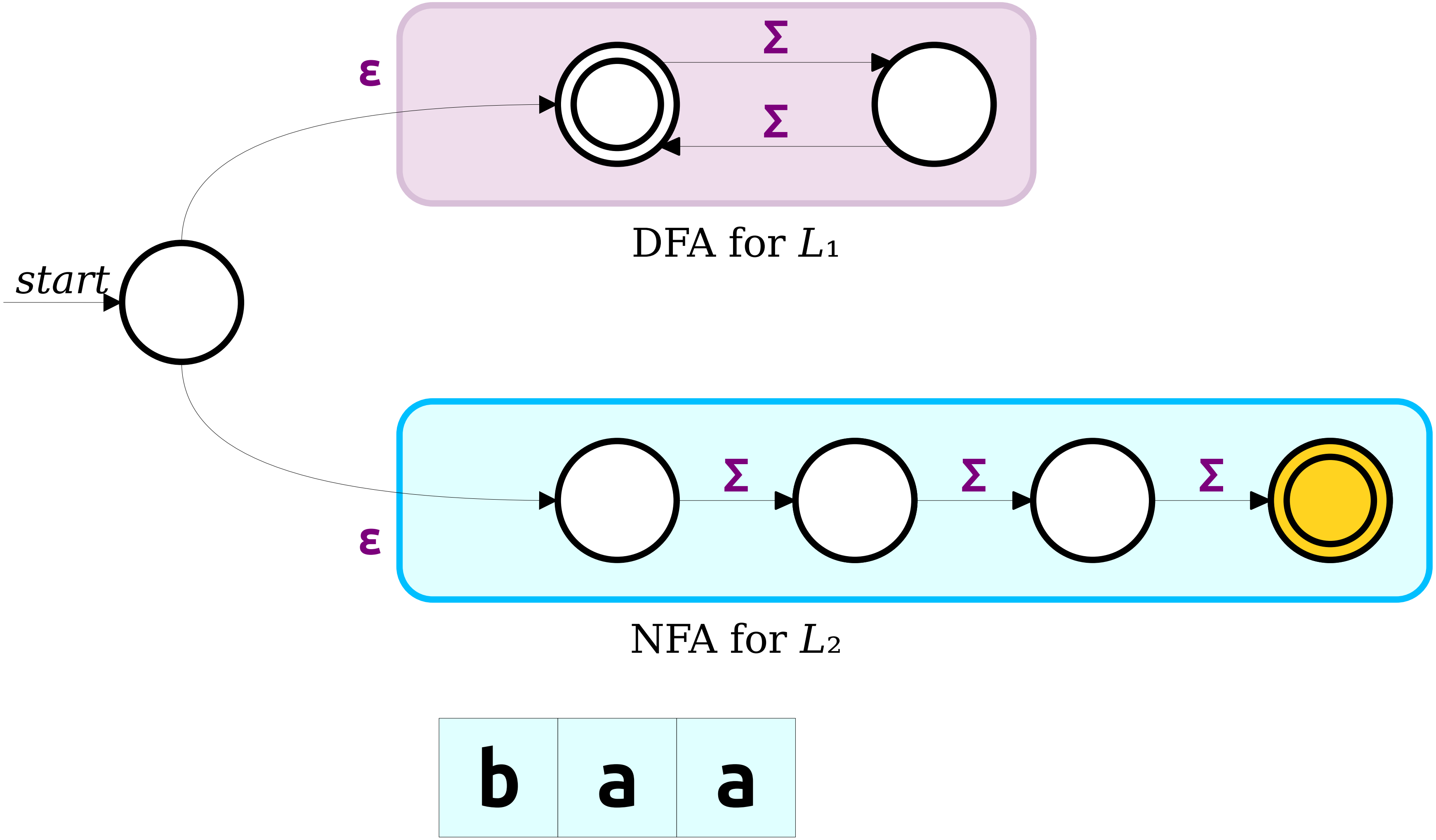
L_1 = { $w \in$ {a, b}* | w has even length }
L_2 = { $w \in$ {a, b}* | w has length exactly three }

Construct an NFA for $L_1 \cup L_2$.



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

Construct an NFA for $L_1 \cup L_2$.



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

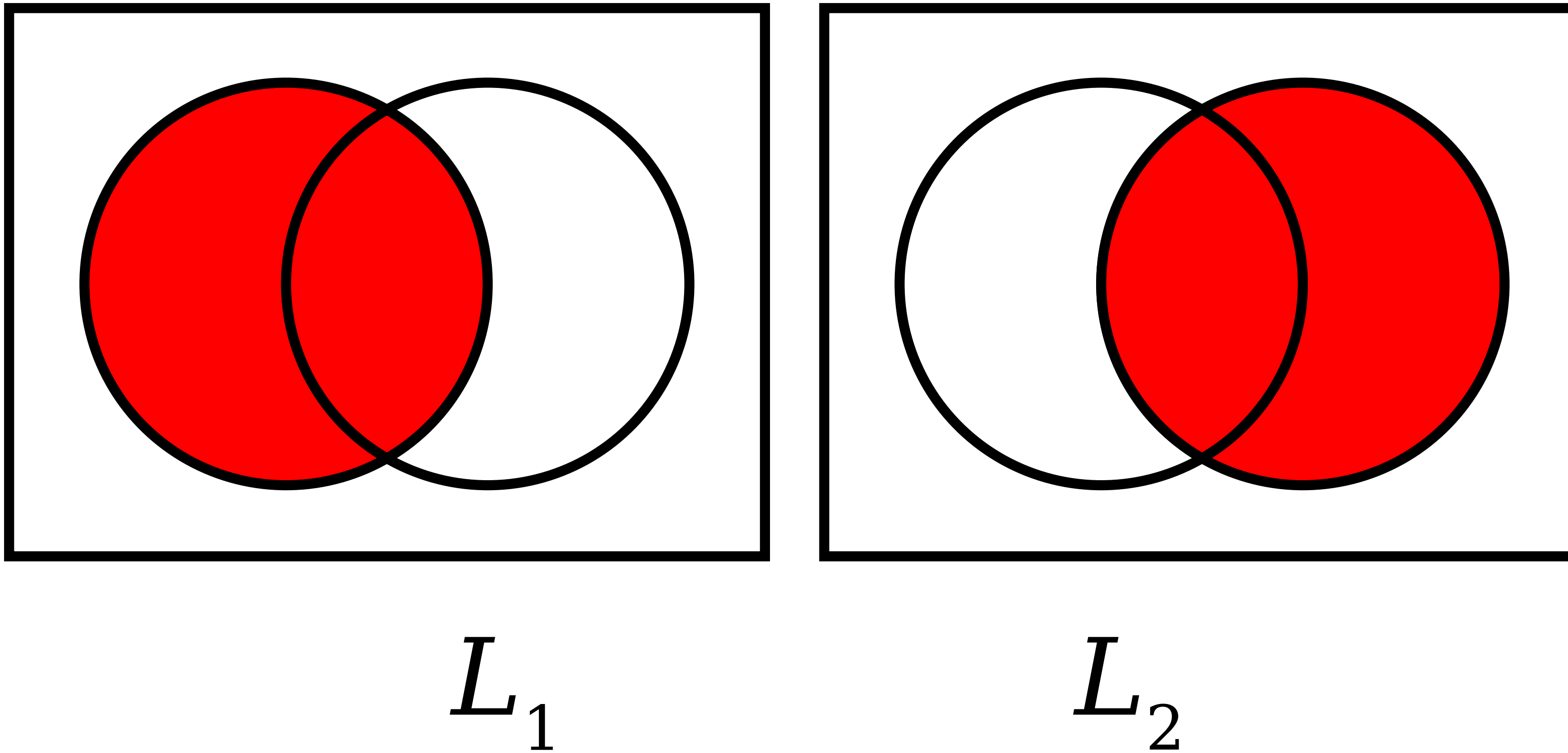
Construct an NFA for $L_1 \cup L_2$.

Closure Under Intersection

- If L_1 and L_2 are languages over Σ, then $\boldsymbol{L_1 \cap L_2}$ is the language of strings in both L_1 and L_2.
- Intuitively, $L_1 \cap L_2$ is the set of strings meeting the requirements of each language.
- ***Theorem:*** If L_1 and L_2 are regular, so is $L_1 \cap L_2$.

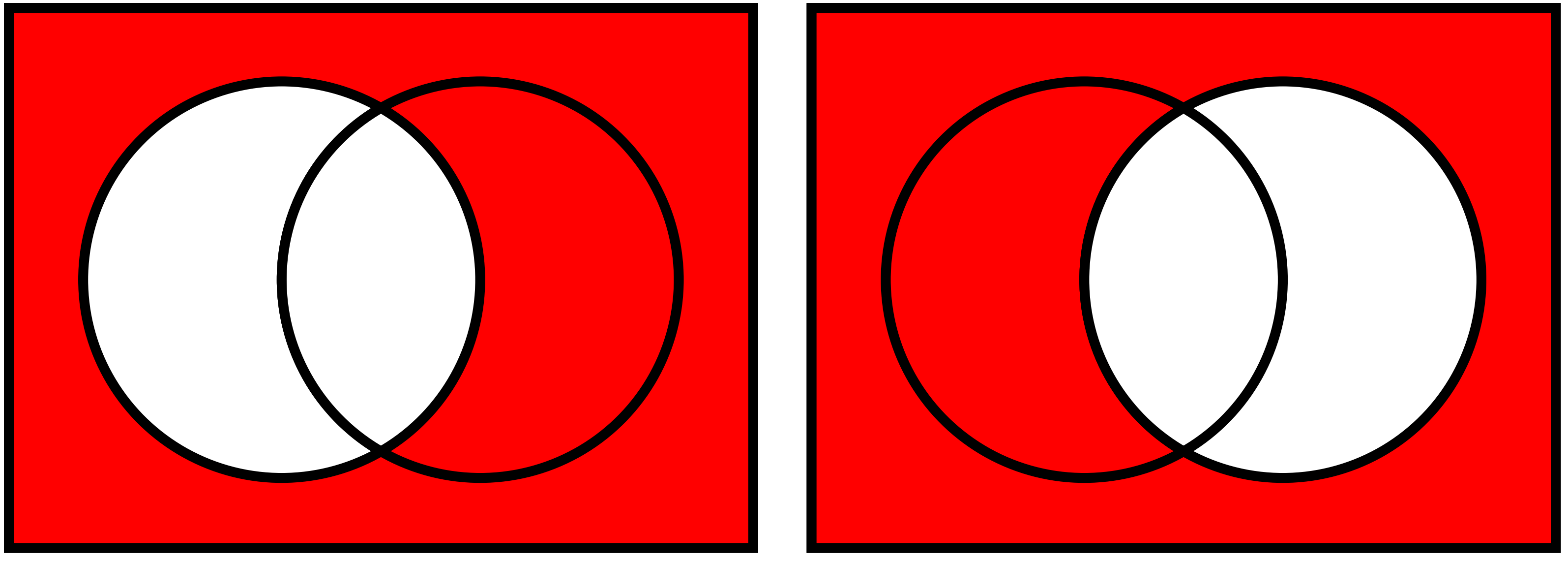
Closure Under Intersection

- If L_1 and L_2 are languages over Σ, then ***$L_1 \cap L_2$*** is the language of strings in both L_1 and L_2.
- Intuitively, $L_1 \cap L_2$ is the set of strings meeting the requirements of each language.
- ***Theorem:*** If L_1 and L_2 are regular, so is $L_1 \cap L_2$.



Closure Under Intersection

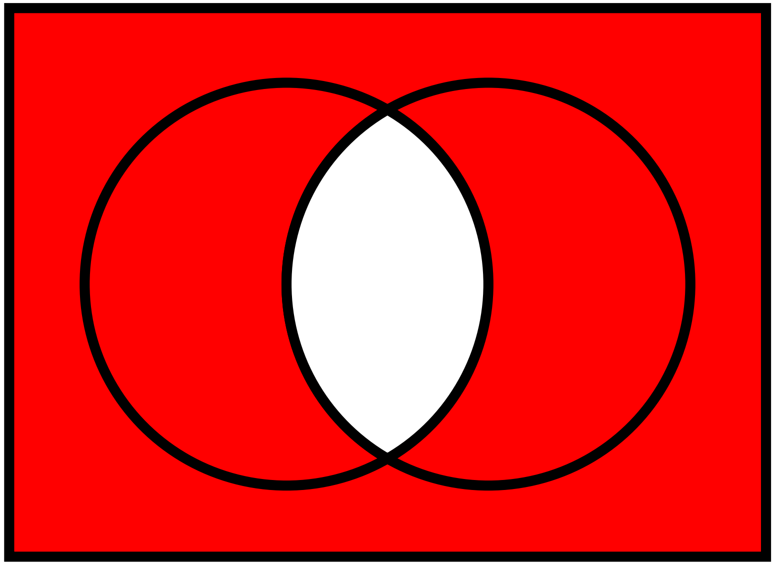
- If L_1 and L_2 are languages over Σ, then ***$L_1 \cap L_2$*** is the language of strings in both L_1 and L_2.
- Intuitively, $L_1 \cap L_2$ is the set of strings meeting the requirements of each language.
- ***Theorem:*** If L_1 and L_2 are regular, so is $L_1 \cap L_2$.



$\overline{L}_1$ $\overline{L}_2$

Closure Under Intersection

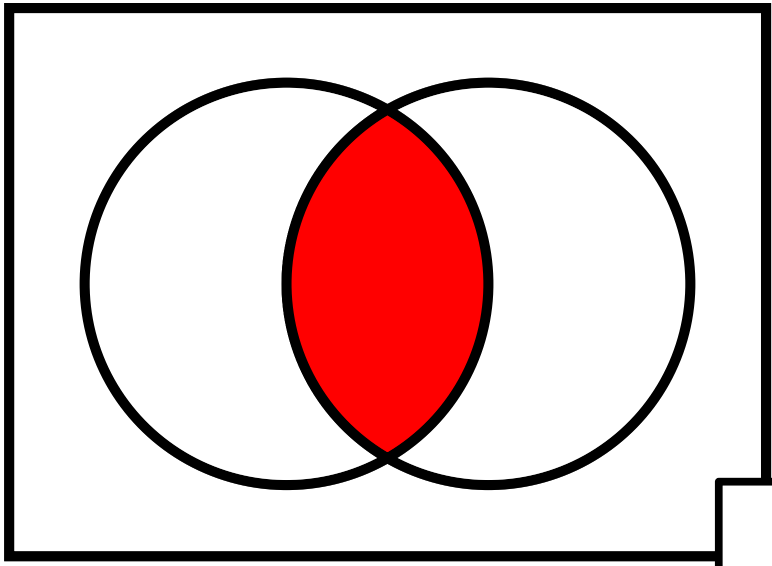
- If L_1 and L_2 are languages over Σ, then $\mathbf{L_1 \cap L_2}$ is the language of strings in both L_1 and L_2.
- Intuitively, $L_1 \cap L_2$ is the set of strings meeting the requirements of each language.
- ***Theorem:*** If L_1 and L_2 are regular, so is $L_1 \cap L_2$.



$$\overline{L}_1 \cup \overline{L}_2$$

Closure Under Intersection

- If L_1 and L_2 are languages over Σ, then $\boldsymbol{L_1 \cap L_2}$ is the language of strings in both L_1 and L_2.
- Intuitively, $L_1 \cap L_2$ is the set of strings meeting the requirements of each language.
- ***Theorem:*** If L_1 and L_2 are regular, so is $L_1 \cap L_2$.



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

Hey, it's De Morgan's laws!

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. **Union and Intersection**
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

Part 3

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. **String Concatenation**
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Numbers

- Numbers can be written in many ways:

  2718

  2,718

  2.718×10^3

  MMDCCXVIII

  二千七百一十八

  ב'תשי"ח

  ൨൭൧൮

  ՍՉԺԸ

  etc.

- How would we design a DFA or NFA that checks if a particular string is a number in some numeral system?

Numbers

- Numbers can be written in many ways:

2718

2,718

2.718 × 10^3

MMDCCXVIII

二千七百一十八

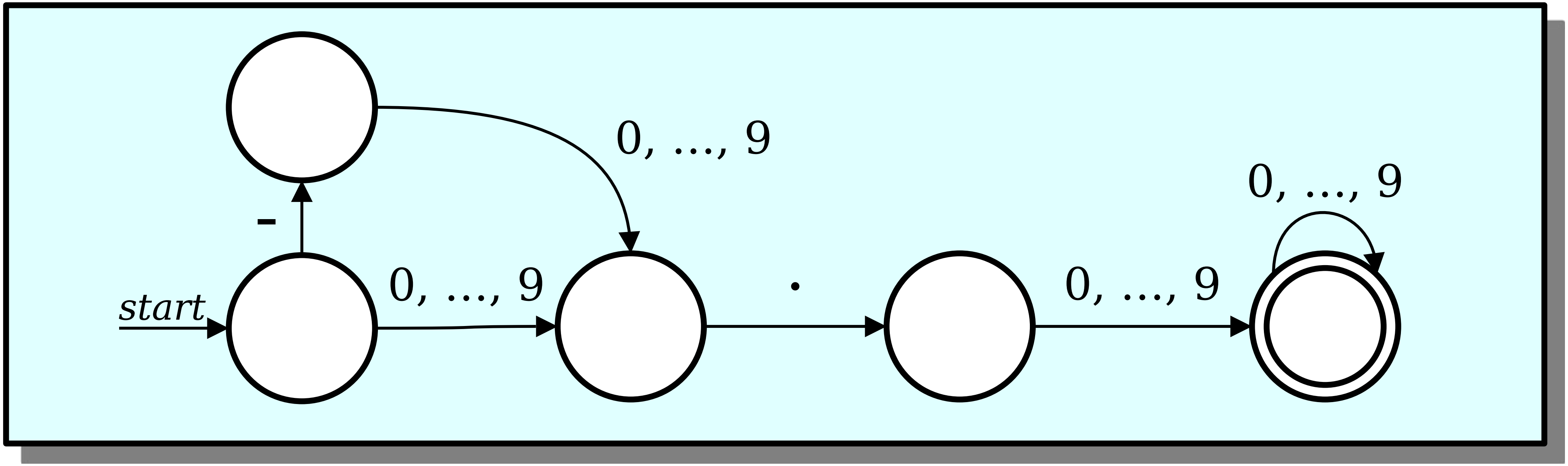
ב'תשי"ח

២៧១៨

ՍՉԺԸ

etc.

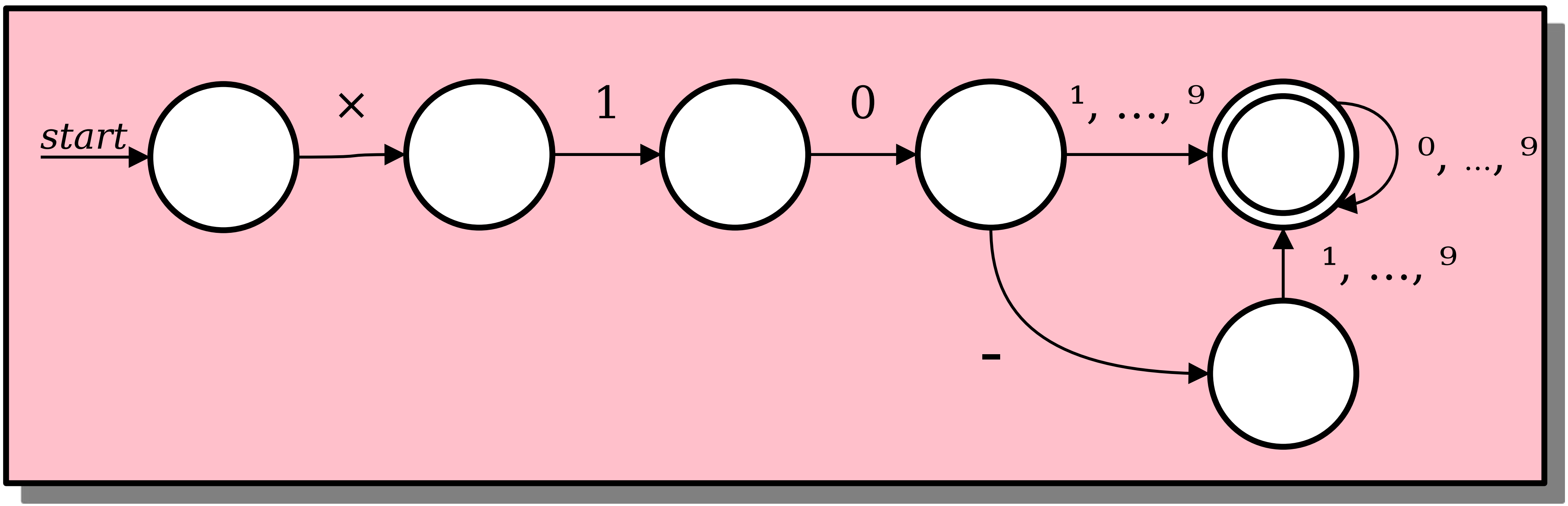
- How would we design a DFA or NFA that checks if a particular string is a number in some numeral system?



2.718 × 10^3

MMDCCXVIII

二千七百一十八



Question: If you can build finite automata to match the first and second halves of a pattern, can you build a single finite automaton that matches the full pattern?

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 = **quo** and x = **kka**, the concatenation wx = **quokka**.
- This is analogous to the + operator for strings in many programming languages.
- Some facts about concatenation:
  - The empty string ε 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 Σ is the language

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

- Let L_1 = { **ab**, **ba** } and L_2 = { **aa**, **bb** }. What is L_1L_2?

Answer at
https://cs103.stanford.edu/pollev

Concatenation Example

- Let Σ = { **a**, **b**, …, **z**, **A**, **B**, …, **Z** } and consider these languages over Σ:
  - ***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 Σ is the language

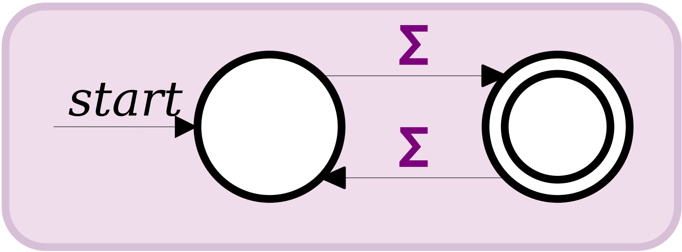
$$\boldsymbol{L_1L_2 = \{\ x \mid \exists w_1 \in L_1.\ \exists w_2 \in L_2.\ x = w_1w_2\ \}}$$

- 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.
- ***Theorem:*** If L_1 and L_2 are regular languages, then so is L_1L_2.

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

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

Construct an NFA for L_1L_2.

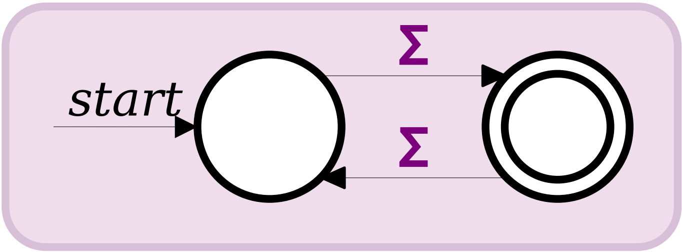


DFA for L_1

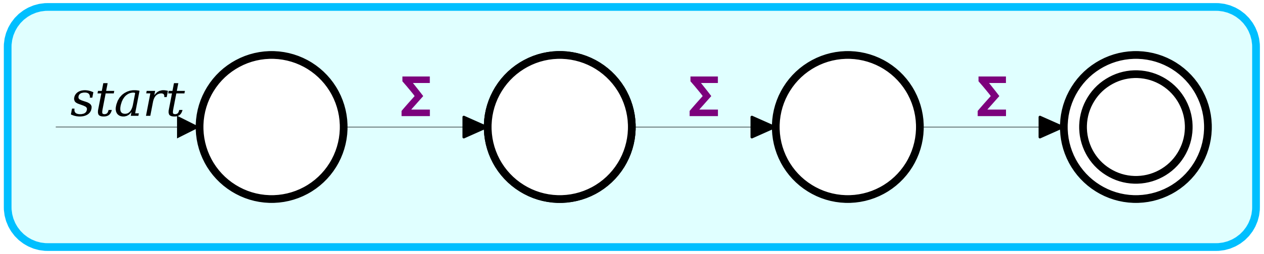
$L_1 = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ has } odd \text{ length } \}$

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

Construct an NFA for L_1L_2.



DFA for L_1

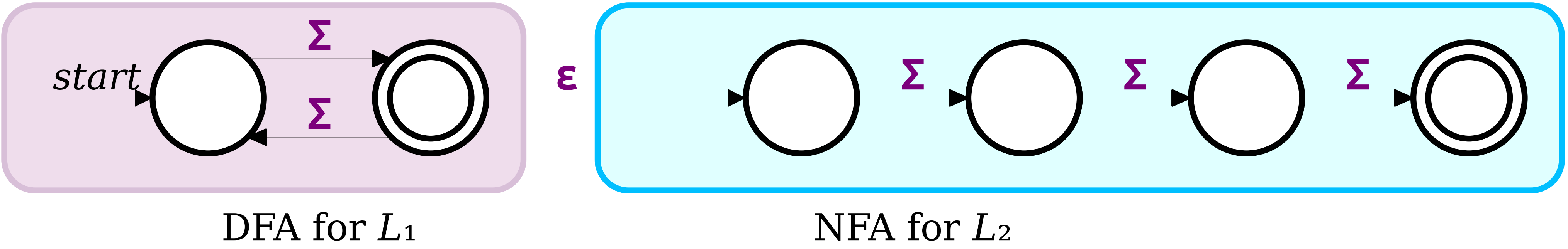


NFA for L_2

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

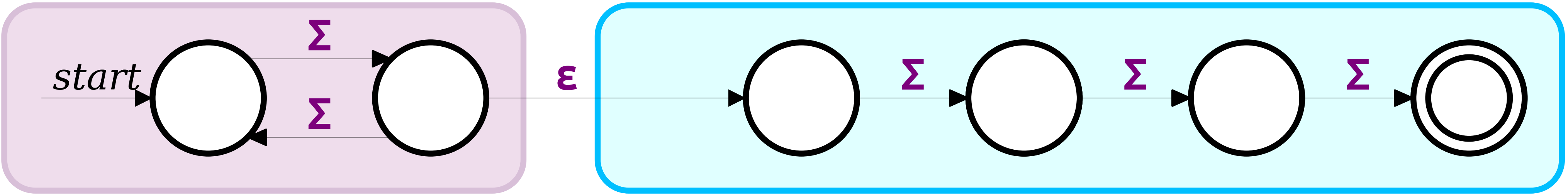
$L_2 = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ has length exactly three } \}$

Construct an NFA for L_1L_2.



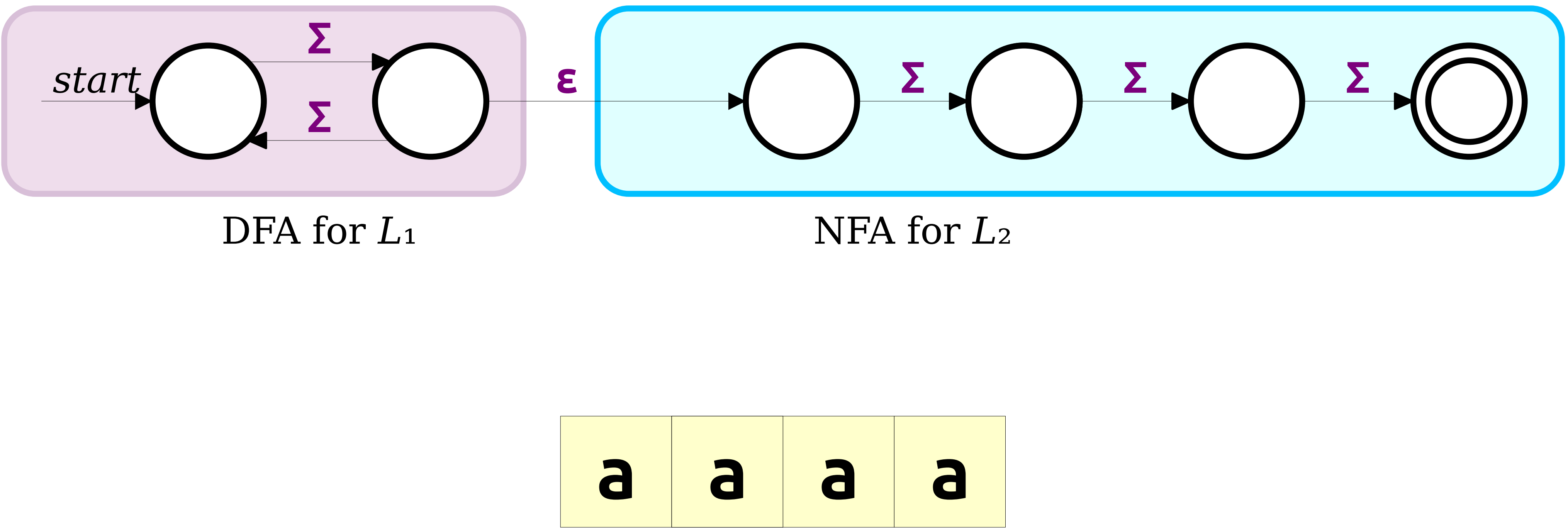
L_1 = { w ∈ {**a**, **b**}* | w has *odd* length }
L_2 = { w ∈ {**a**, **b**}* | w has length exactly three }

Construct an NFA for L_1L_2.



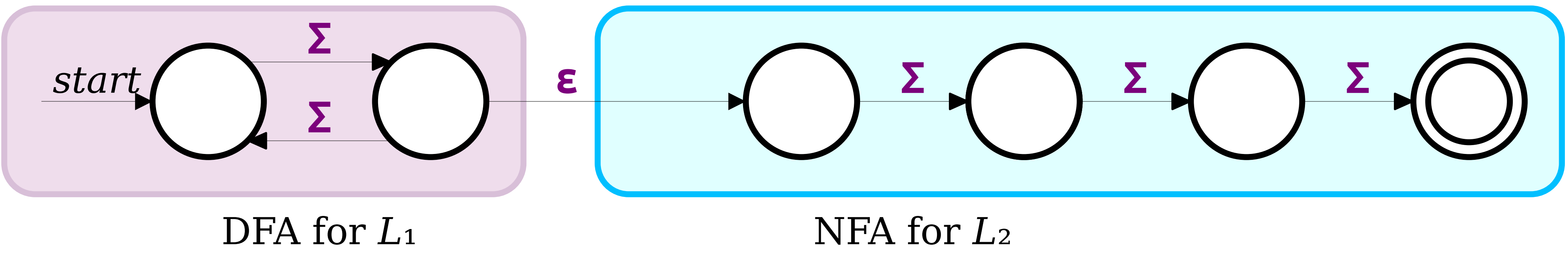
L_1 = { w ∈ {**a**, **b**}* | w has *odd* length }
L_2 = { w ∈ {**a**, **b**}* | w has length exactly three }

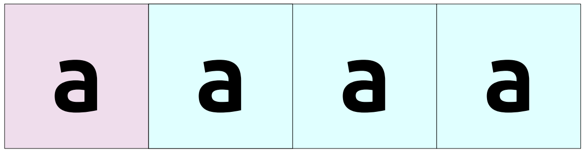
Construct an NFA for L_1L_2.



L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

Construct an NFA for L_1L_2.

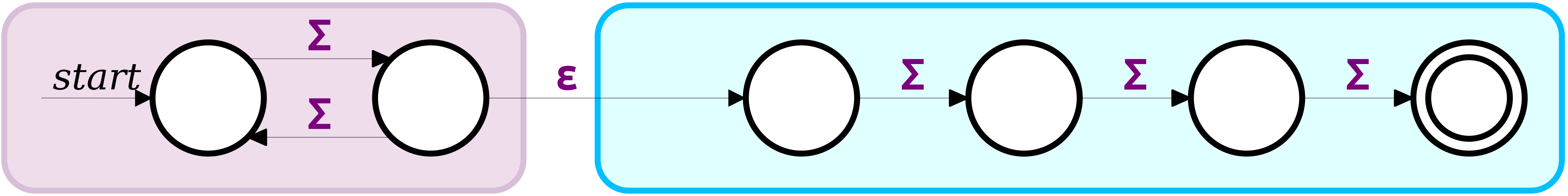




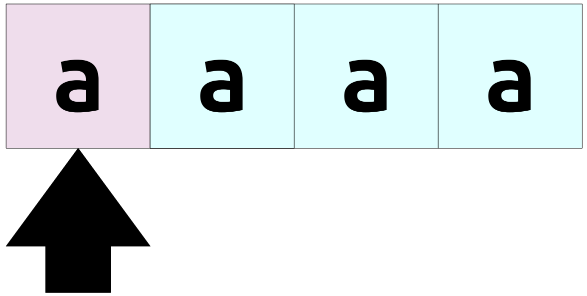
$L_1 = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ has } \textit{odd} \text{ length } \}$

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

Construct an NFA for L_1L_2.



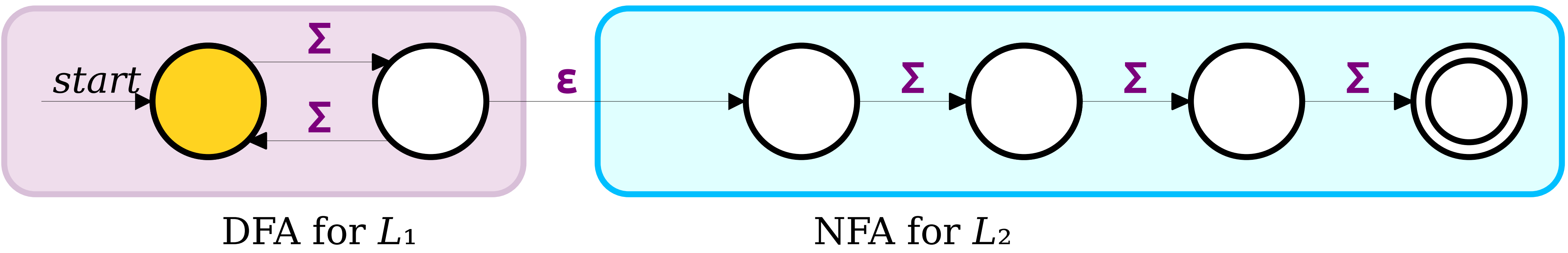
DFA for L_1 NFA for L_2

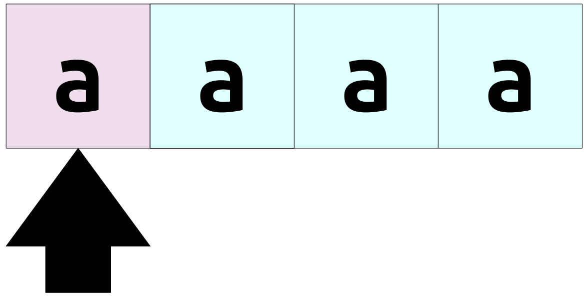


$L_1 = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ has } \textit{odd} \text{ length } \}$

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

Construct an NFA for L_1L_2.

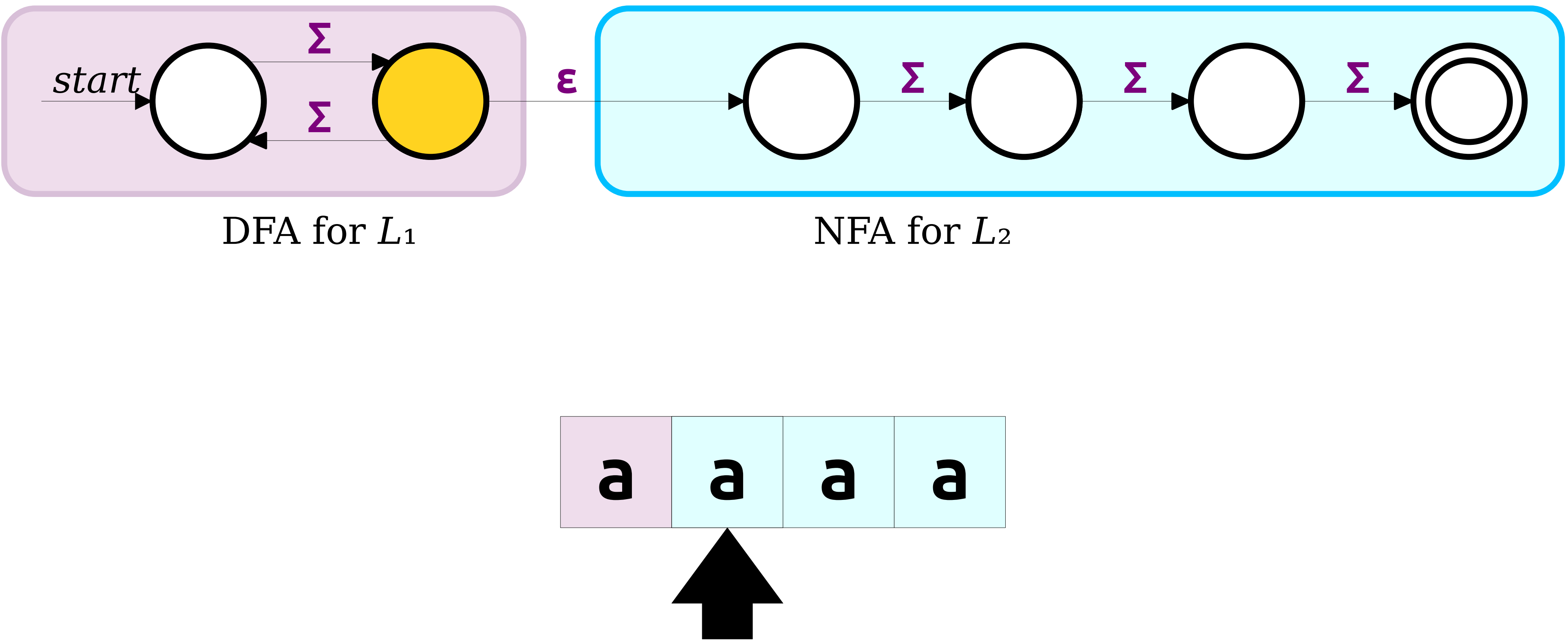




$L_1 = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ has } \textit{odd} \text{ length } \}$

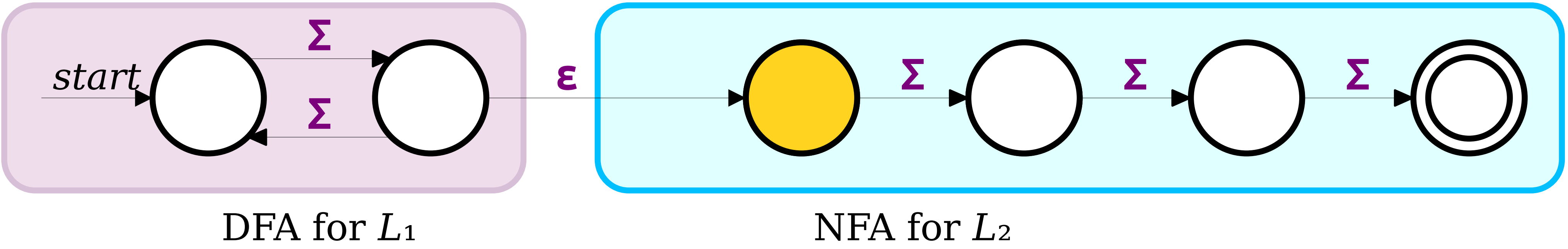
$L_2 = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ has length exactly three } \}$

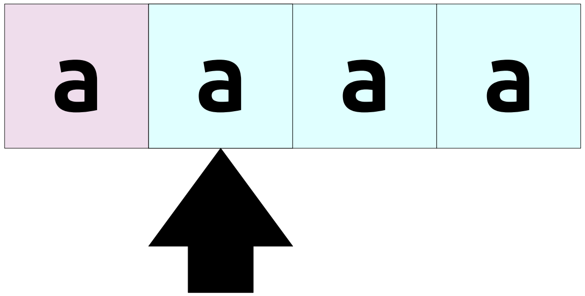
Construct an NFA for L_1L_2.



L_1 = { w ∈ {**a**, **b**}* | w has *odd* length }
L_2 = { w ∈ {**a**, **b**}* | w has length exactly three }

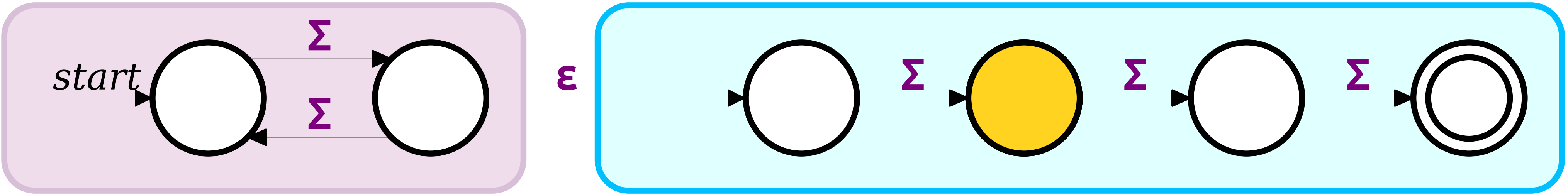
Construct an NFA for L_1L_2.



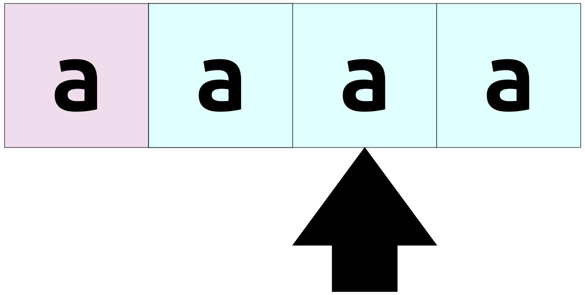


L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

Construct an NFA for L_1L_2.

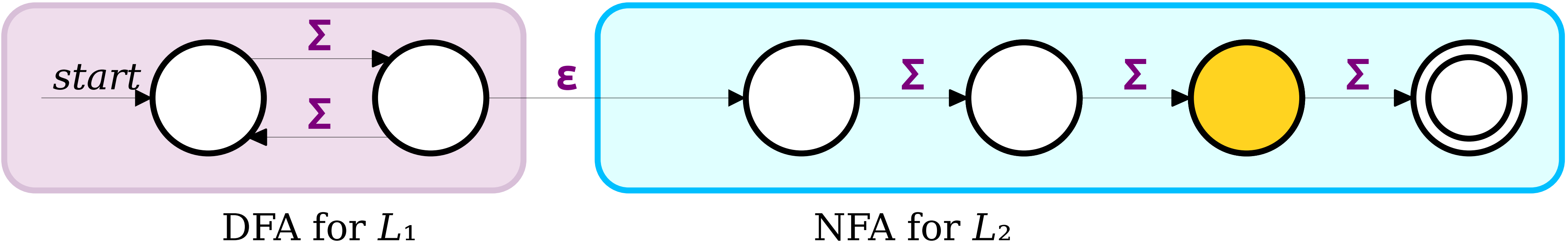


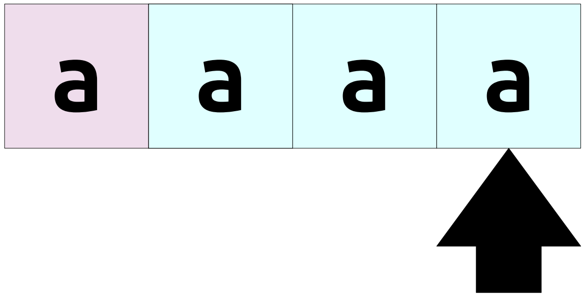
DFA for L_1 NFA for L_2



L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

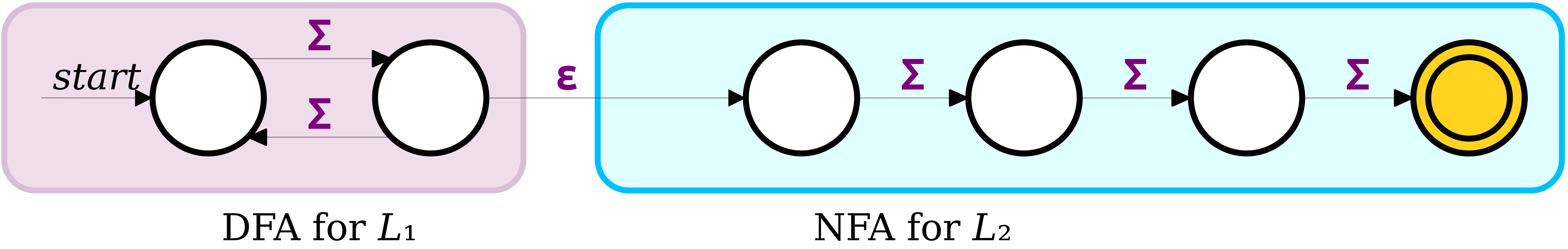
Construct an NFA for L_1L_2.

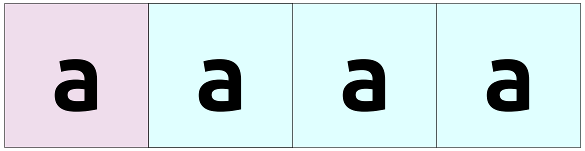




L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

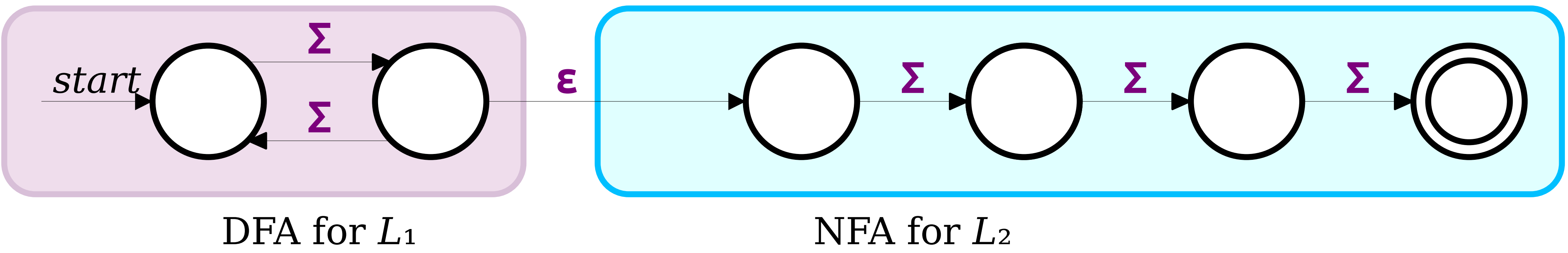
Construct an NFA for L_1L_2.

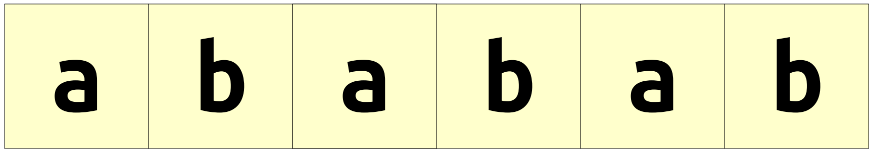




L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

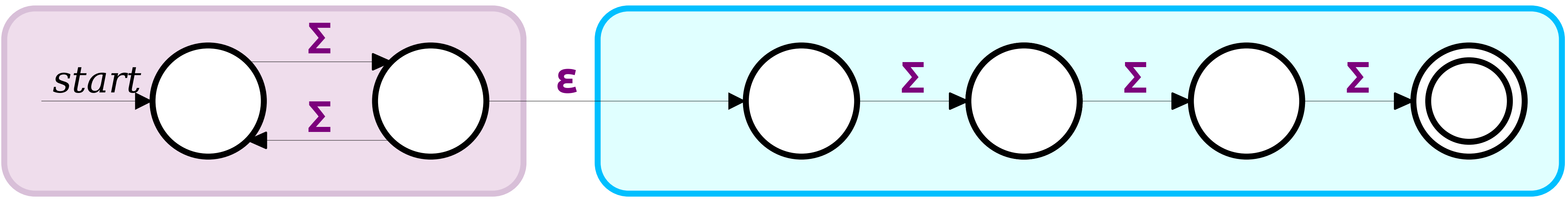
Construct an NFA for L_1L_2.





L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

Construct an NFA for L_1L_2.



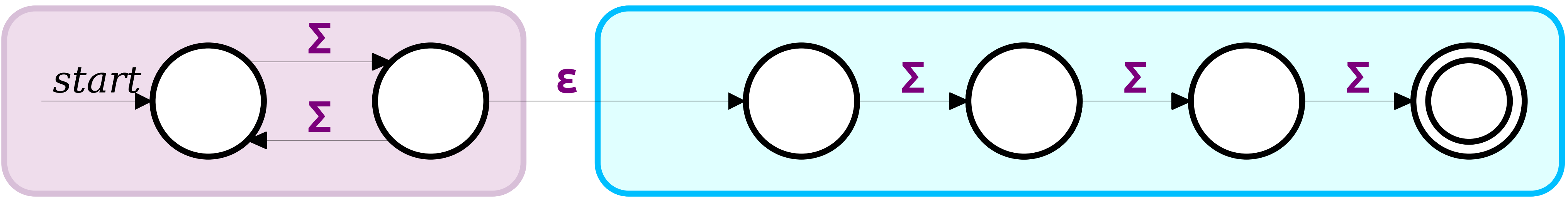
DFA for L_1 NFA for L_2

a	b	a	b	a	b

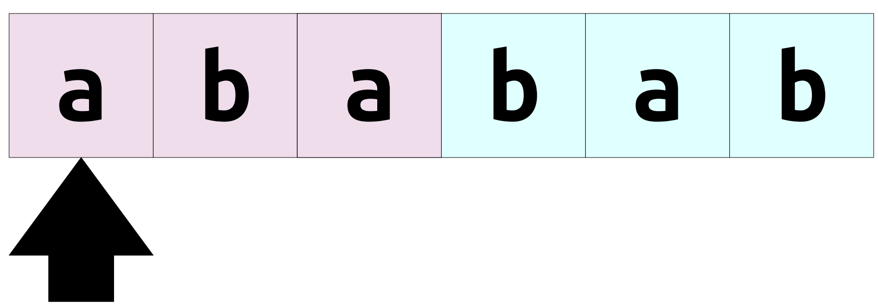
$L_1 = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ has } \textit{odd} \text{ length } \}$

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

Construct an NFA for L_1L_2.



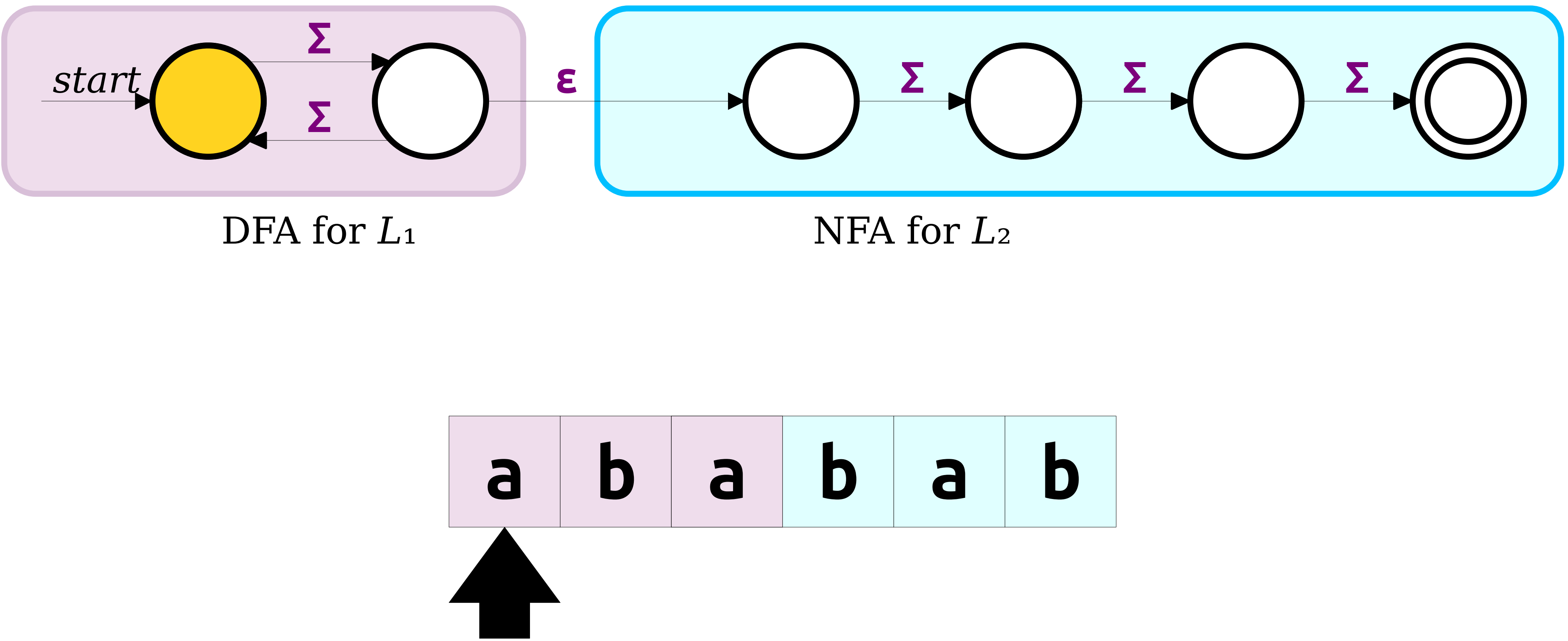
DFA for L_1 NFA for L_2



L_1 = { w ∈ {**a**, **b**}* | w has *odd* length }

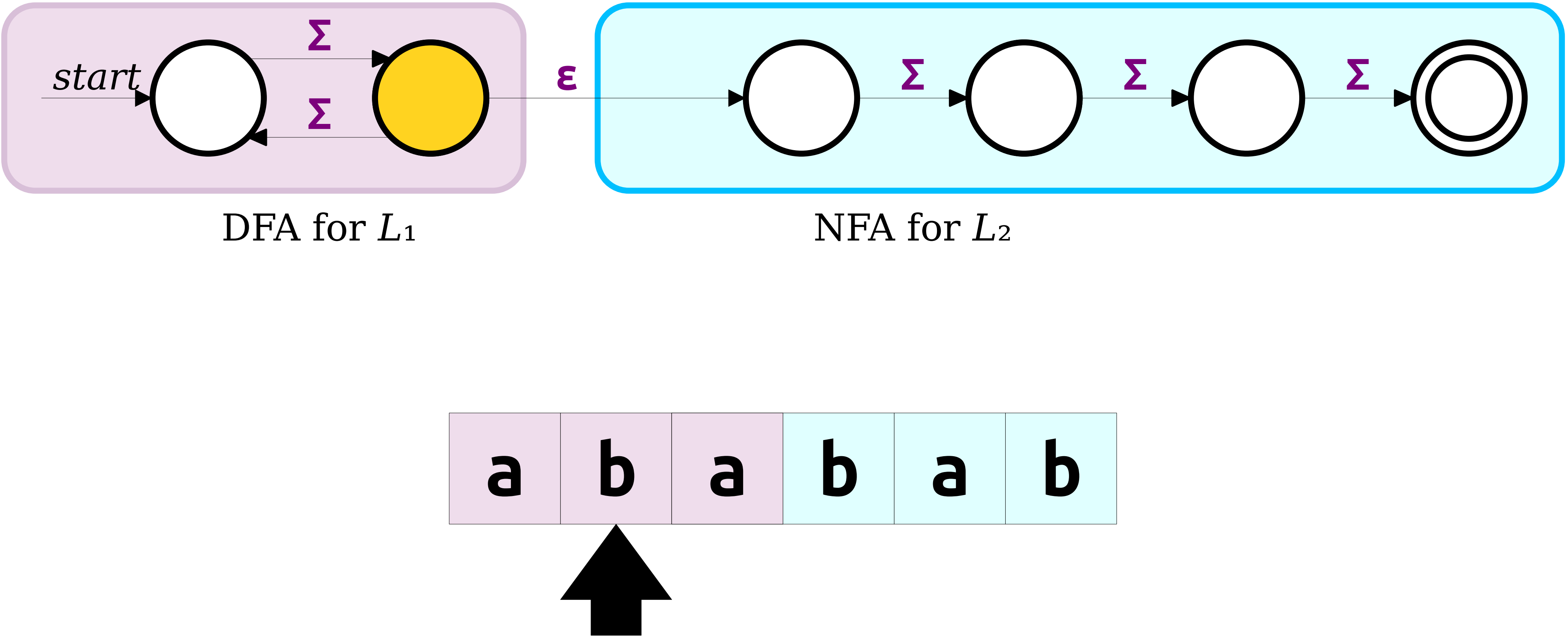
L_2 = { w ∈ {**a**, **b**}* | w has length exactly three }

Construct an NFA for L_1L_2.



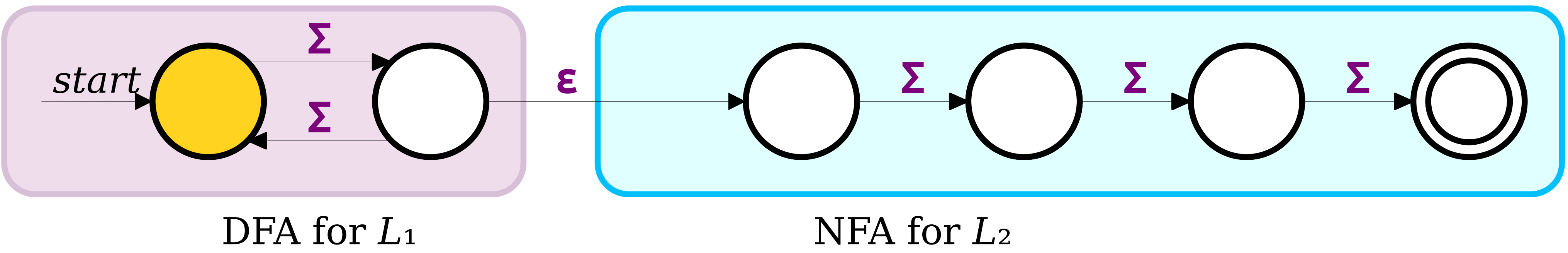
L_1 = { $w \in$ {**a**, **b**}* | w has *odd* length }
L_2 = { $w \in$ {**a**, **b**}* | w has length exactly three }

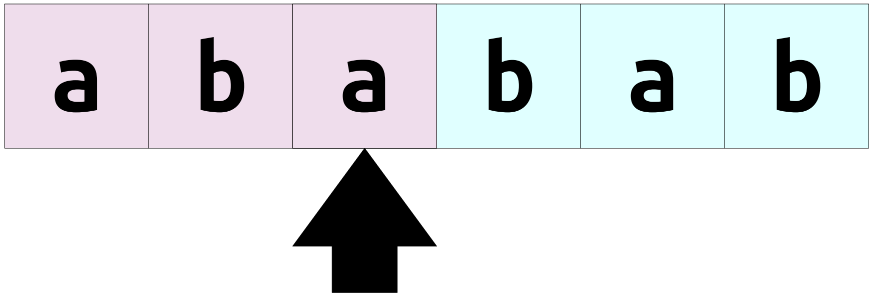
Construct an NFA for L_1L_2.



L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

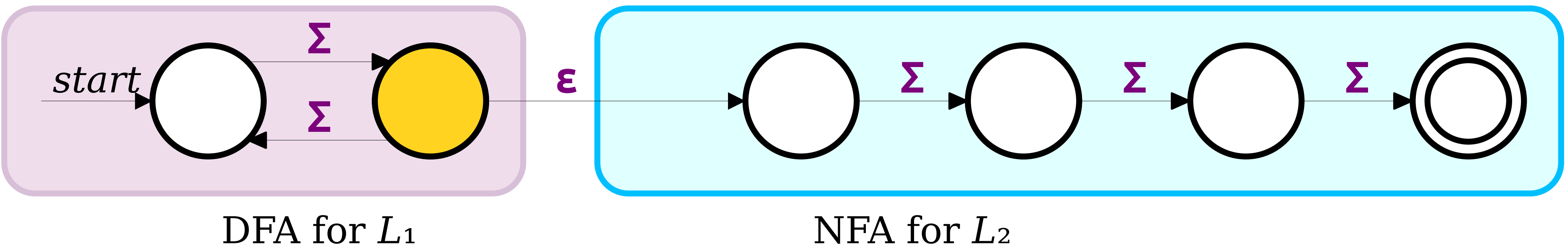
Construct an NFA for L_1L_2.

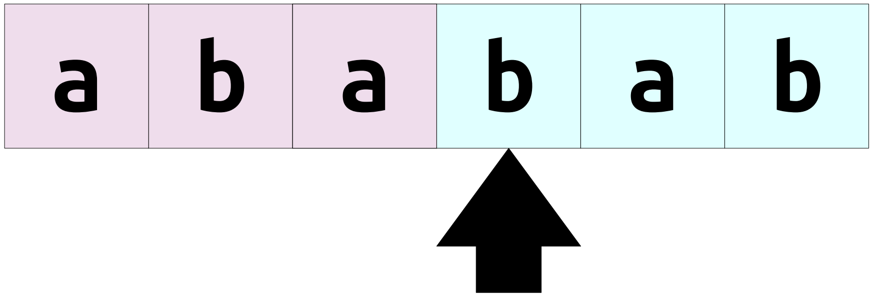




L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

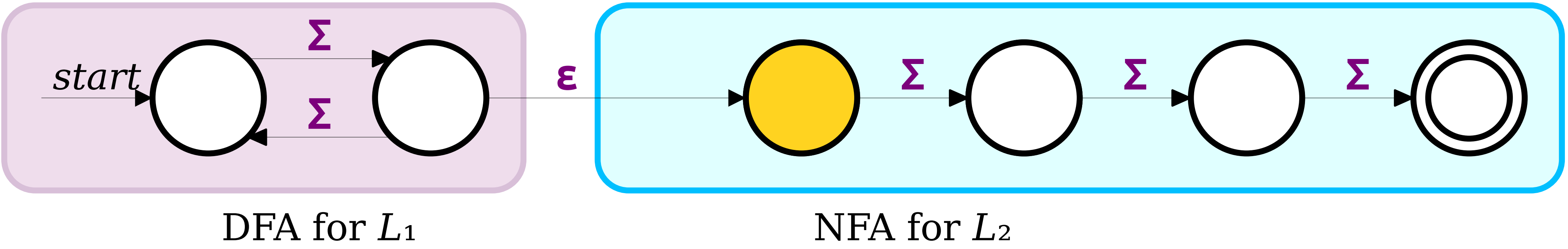
Construct an NFA for L_1L_2.

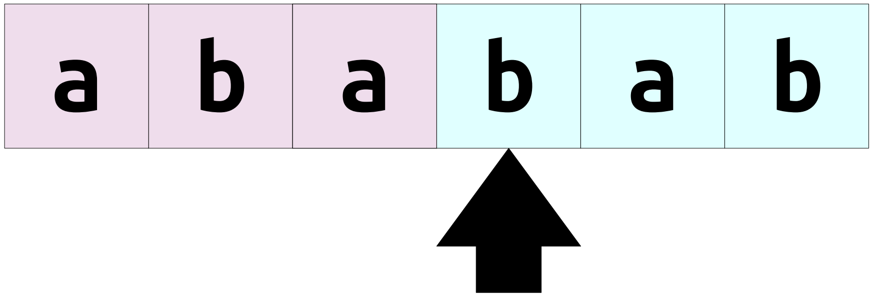




L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

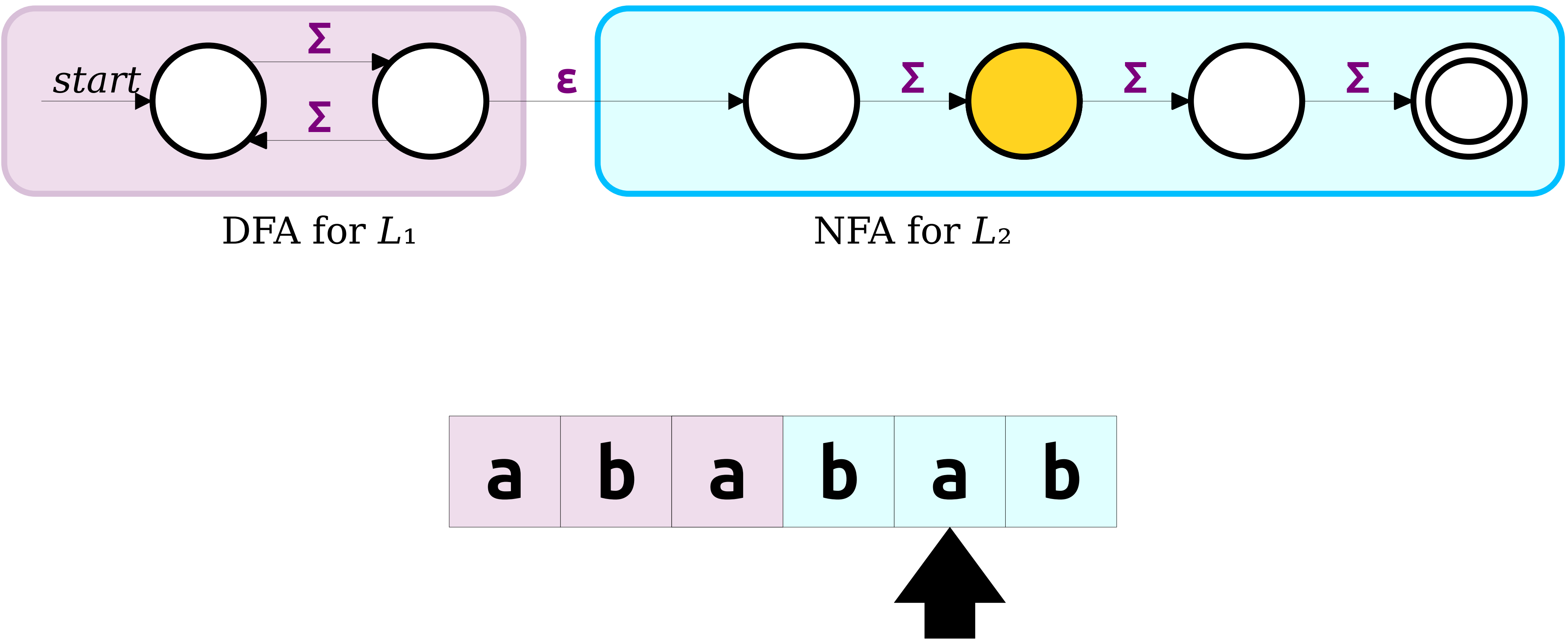
Construct an NFA for L_1L_2.





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

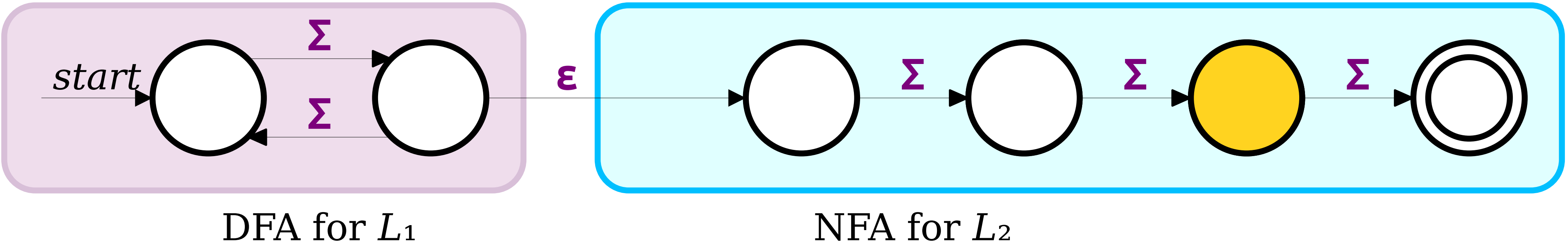
Construct an NFA for L_1L_2.

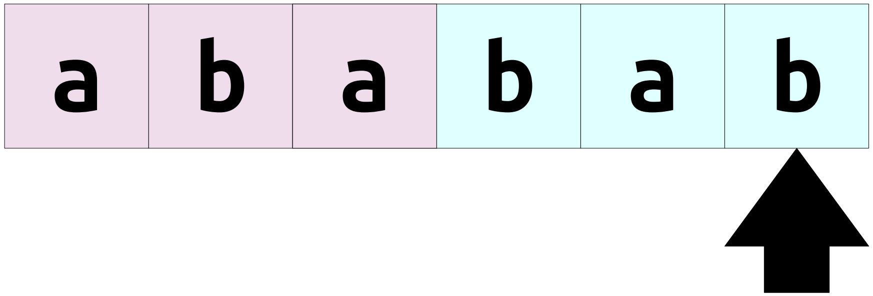


L_1 = { w ∈ {a, b}* | w has *odd* length }

L_2 = { w ∈ {a, b}* | w has length exactly three }

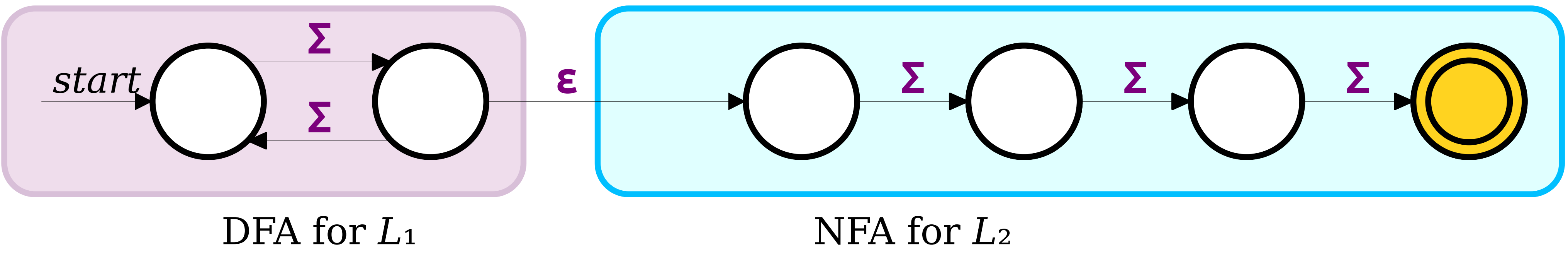
Construct an NFA for L_1L_2.





L_1 = { w ∈ {**a**, **b**}* | w has *odd* length }
L_2 = { w ∈ {**a**, **b**}* | w has length exactly three }

Construct an NFA for L_1L_2.



a	b	a	b	a	b

L_1 = { w ∈ {a, b}* | w has *odd* length }
L_2 = { w ∈ {a, b}* | w has length exactly three }

Construct an NFA for L_1L_2.

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. **String Concatenation**
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What's Next?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. **Language Exponentiation and Kleene Star**
9. Summary of Closure Properties
10. What’s Next?

Numbers

- Suppose we successfully build a finite automaton that checks if a string is a numbers.
- Now, we want to make a new automaton that checks if a string consists of a *series* of numbers.
  - Perhaps we're parsing a data file, for example.
- Do we have to start from scratch? Or could we reuse what we have?

Lots and Lots of Concatenation

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

{ aaaa, aab, baa, bb }

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

{ aaaaaa, aaaab, aabaa, aabb, baaaa, baab, bbaa, bbb}

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

{ aaaaaaaa, aaaaaab, aaaabaa, aaaabb, aabaaaa, aabaab, aabbaa, aabbb, baaaaaa, baaaab, baabaa, baabb, bbaaaa, bbaab, bbbaa, 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 Closure

- An important operation on languages is the ***Kleene closure***, or ***Kleene star***, 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 Ø*?

The Kleene Closure

If L = { **a**, **bb** }, then L^* = {

ε,

a, **bb**,

aa, **abb**, **bba**, **bbbb**,

aaa, **aabb**, **abba**, **abbbb**, **bbaa**, **bbabb**, **bbbba**, **bbbbbb**,

…

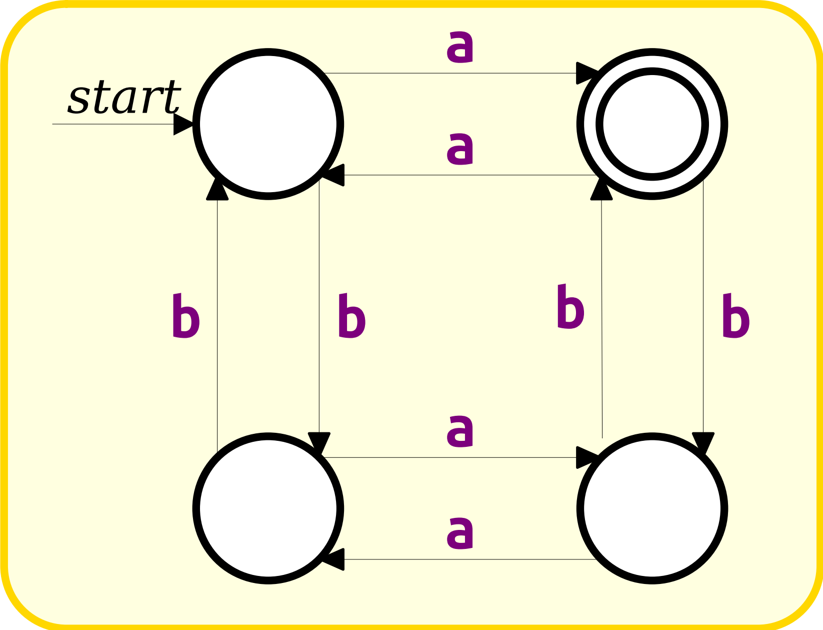
}

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.

Theorem: If L is a regular language, so is L^*.

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

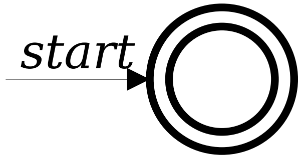
Construct an NFA for L^*.

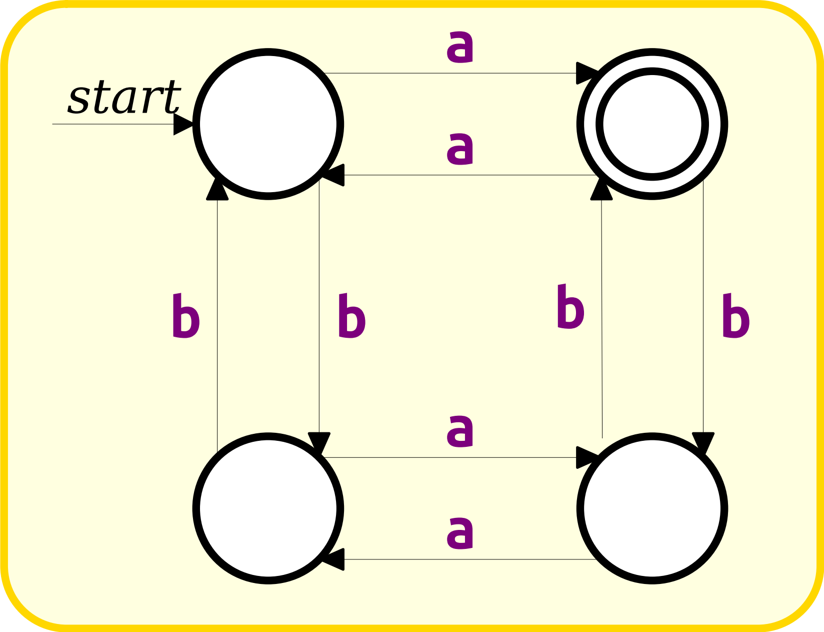


DFA for L

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

Construct an NFA for L^*.

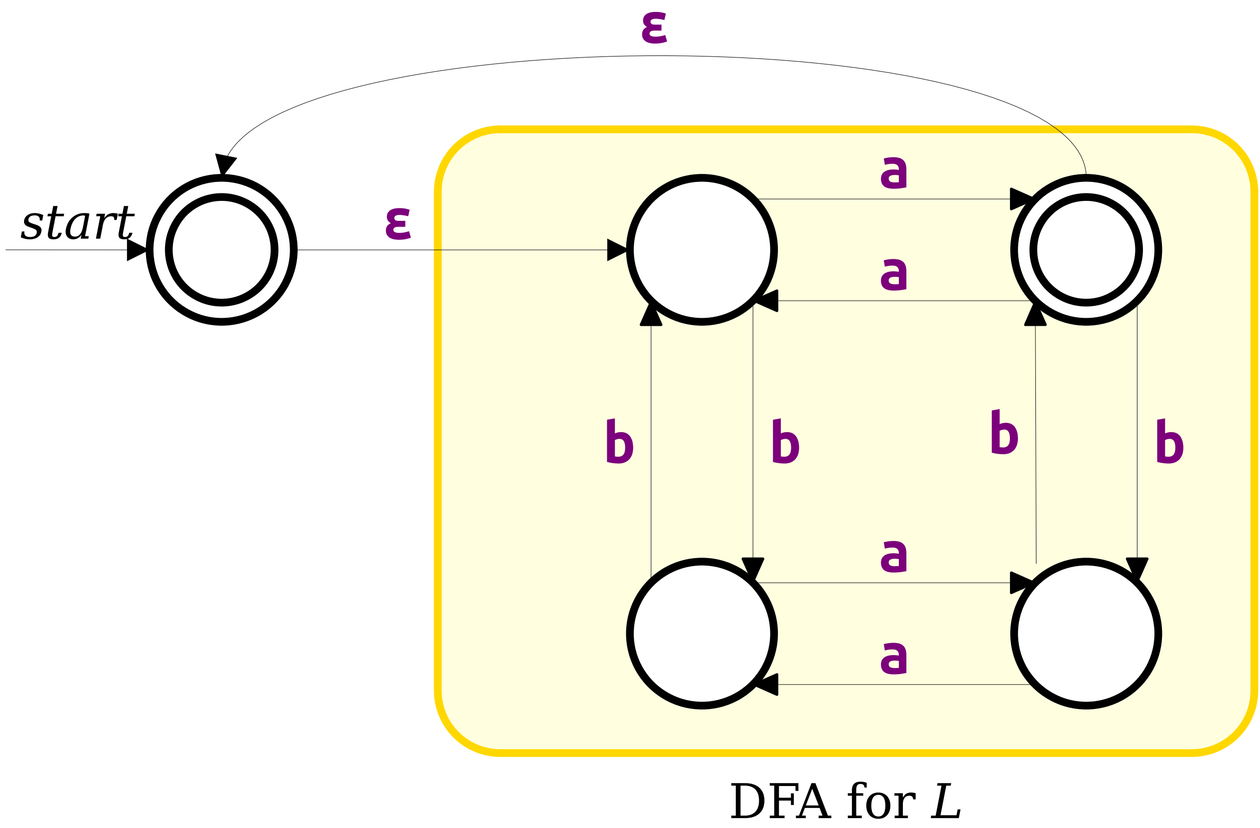




DFA for L

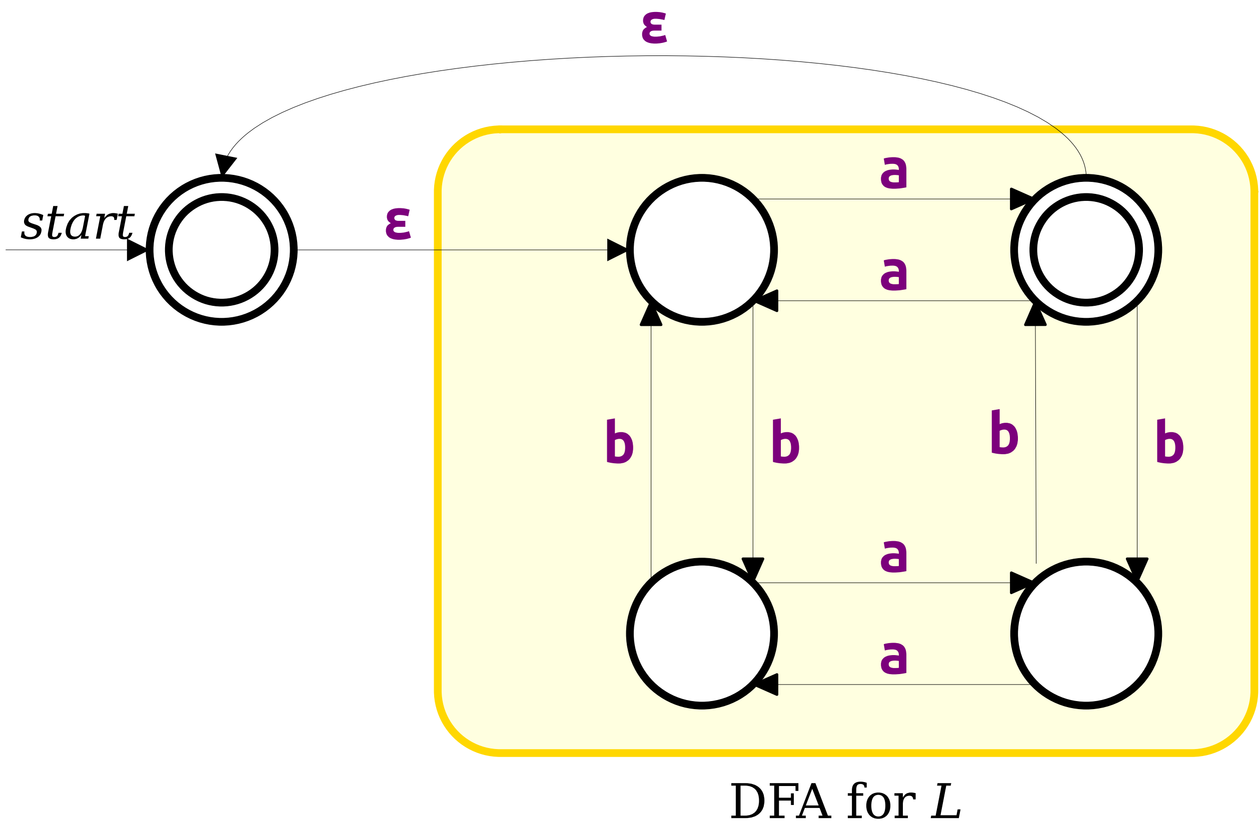
L = { w ∈ {**a**, **b**}* | w has an odd number of **a**'s and an even number of **b**'s }

Construct an NFA for L*.



L = { w ∈ {**a**, **b**}* | w has an odd number of **a**'s and an even number of **b**'s }

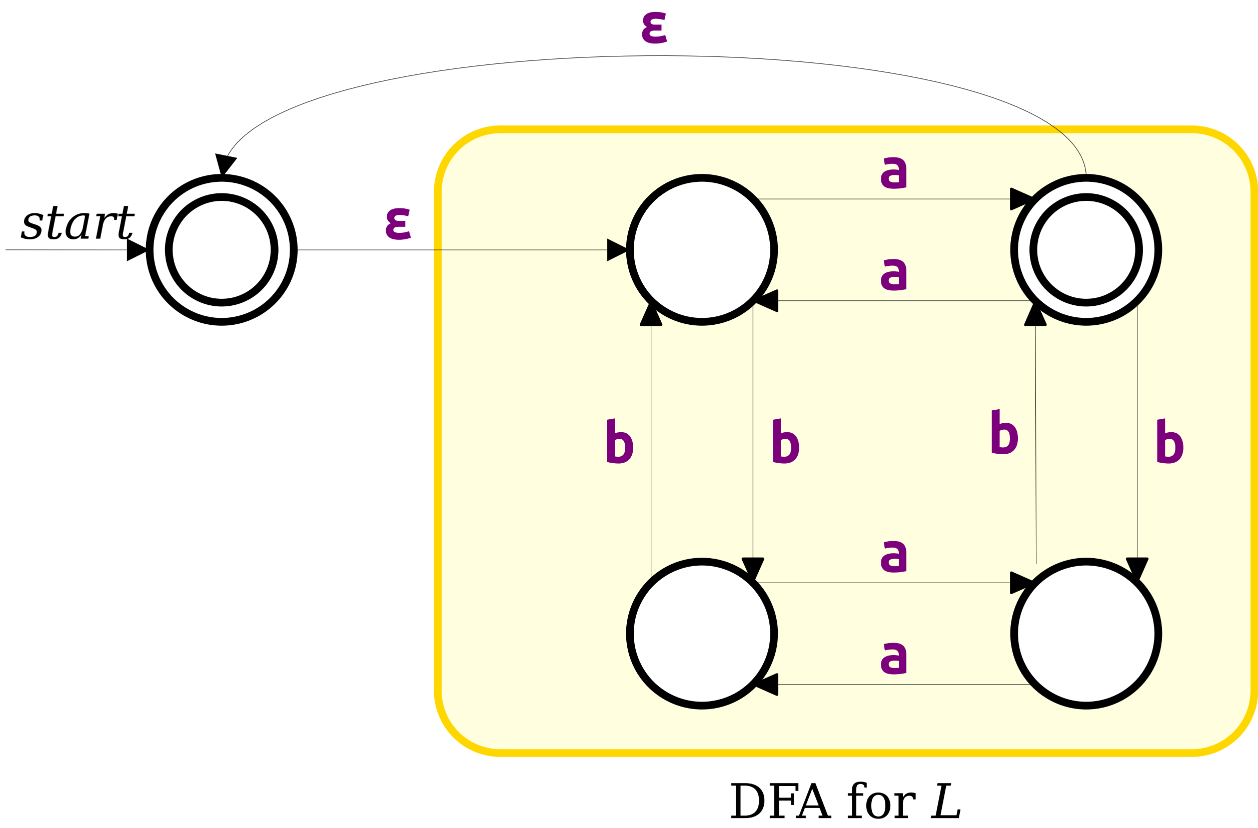
Construct an NFA for L^*.

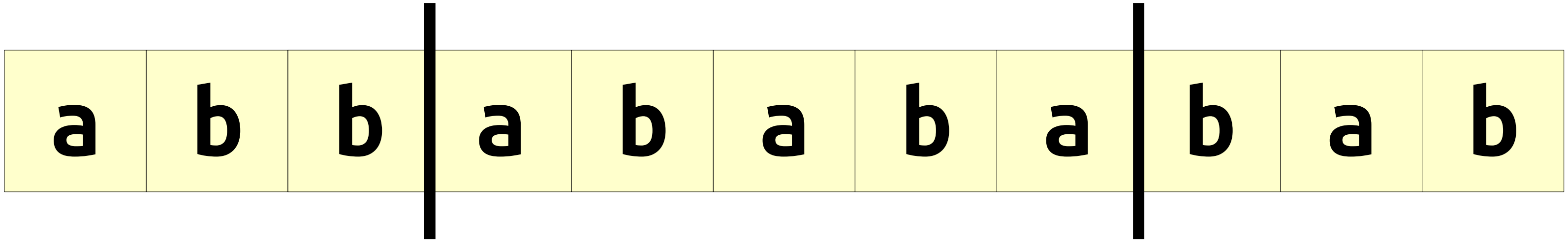


a	b	b	a	b	a	b	a	b	a	b

L = { w ∈ {a, b}* | w has an odd number of a's and an even number of b's }

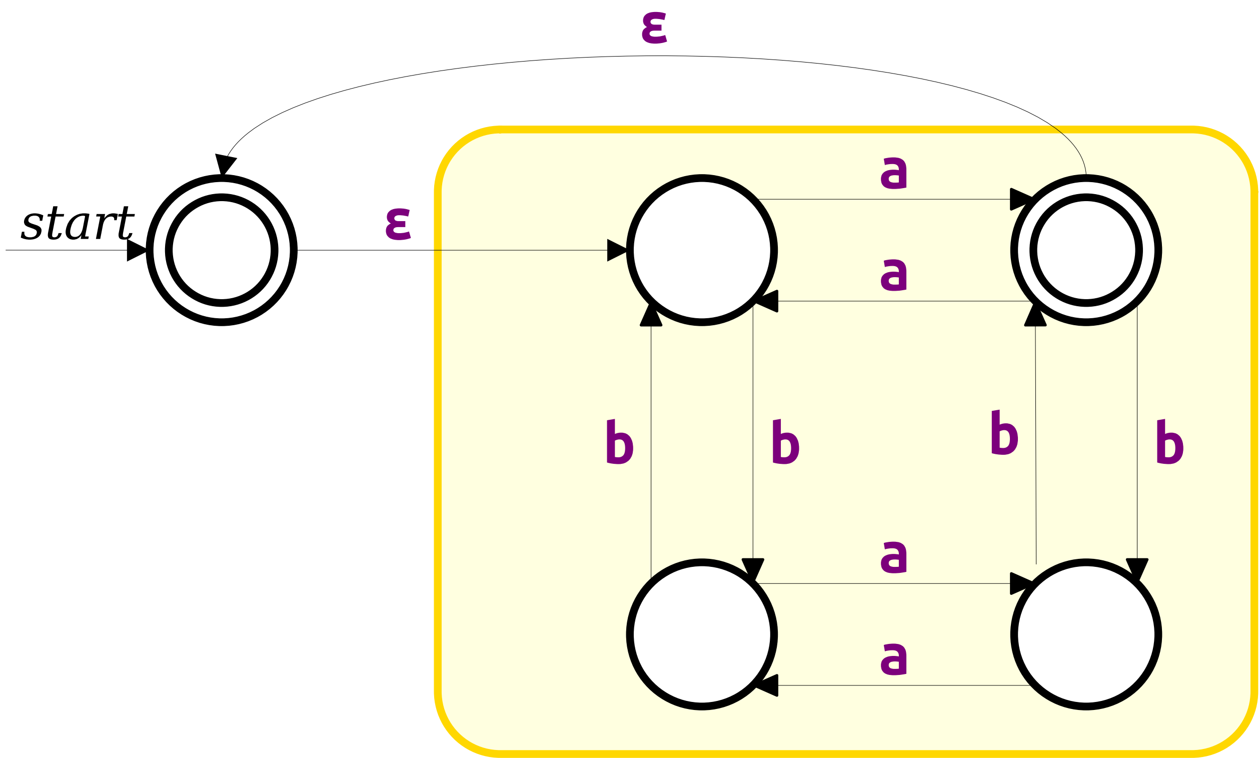
Construct an NFA for L*.



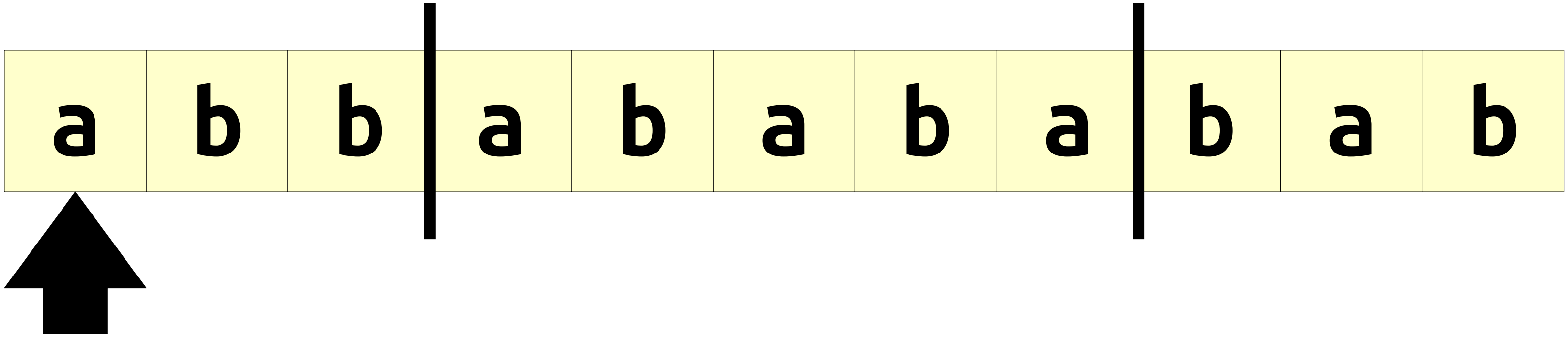


$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ has an odd number of **a**'s and an even number of **b**'s $\}$

Construct an NFA for L^*.

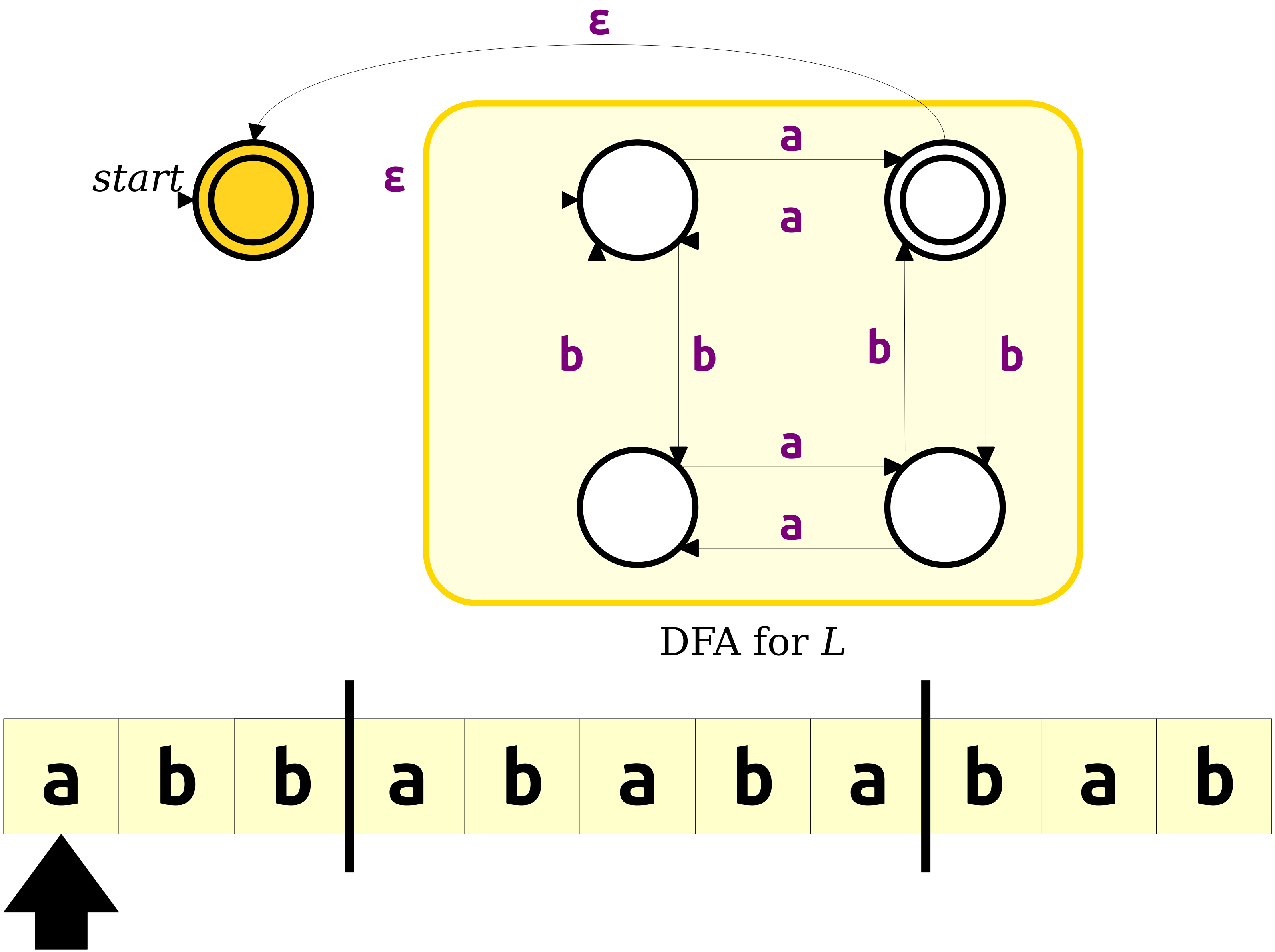


DFA for *L*



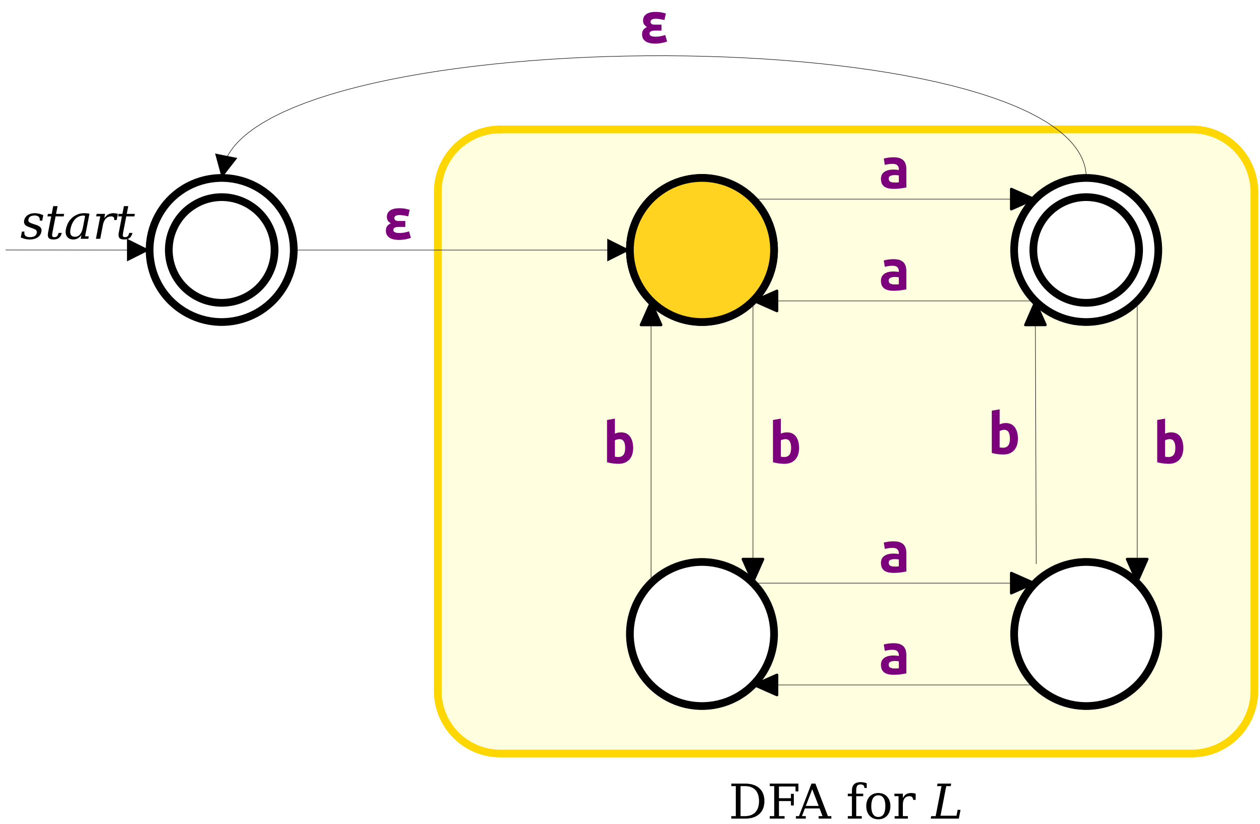
L = { $w \in$ {**a**, **b**}* | w has an odd number of **a**'s and an even number of **b**'s }

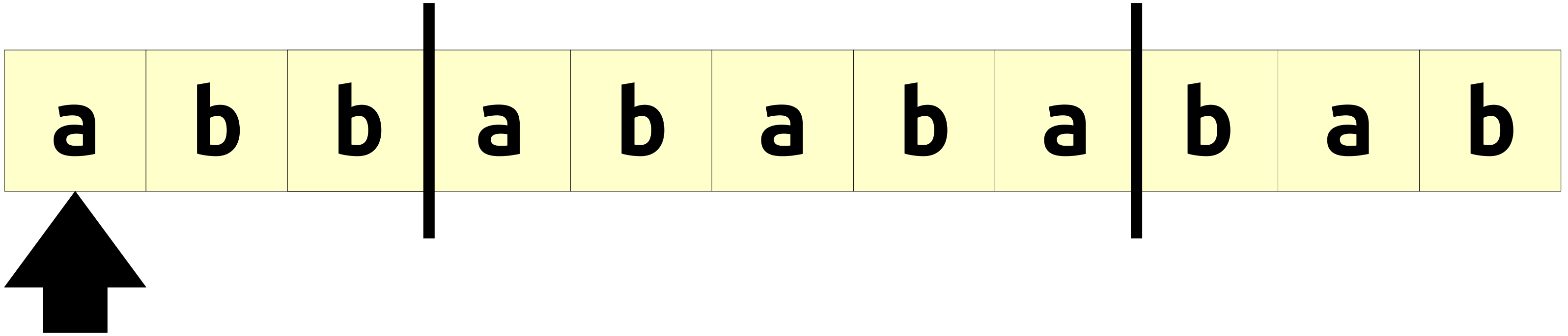
Construct an NFA for L*.



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

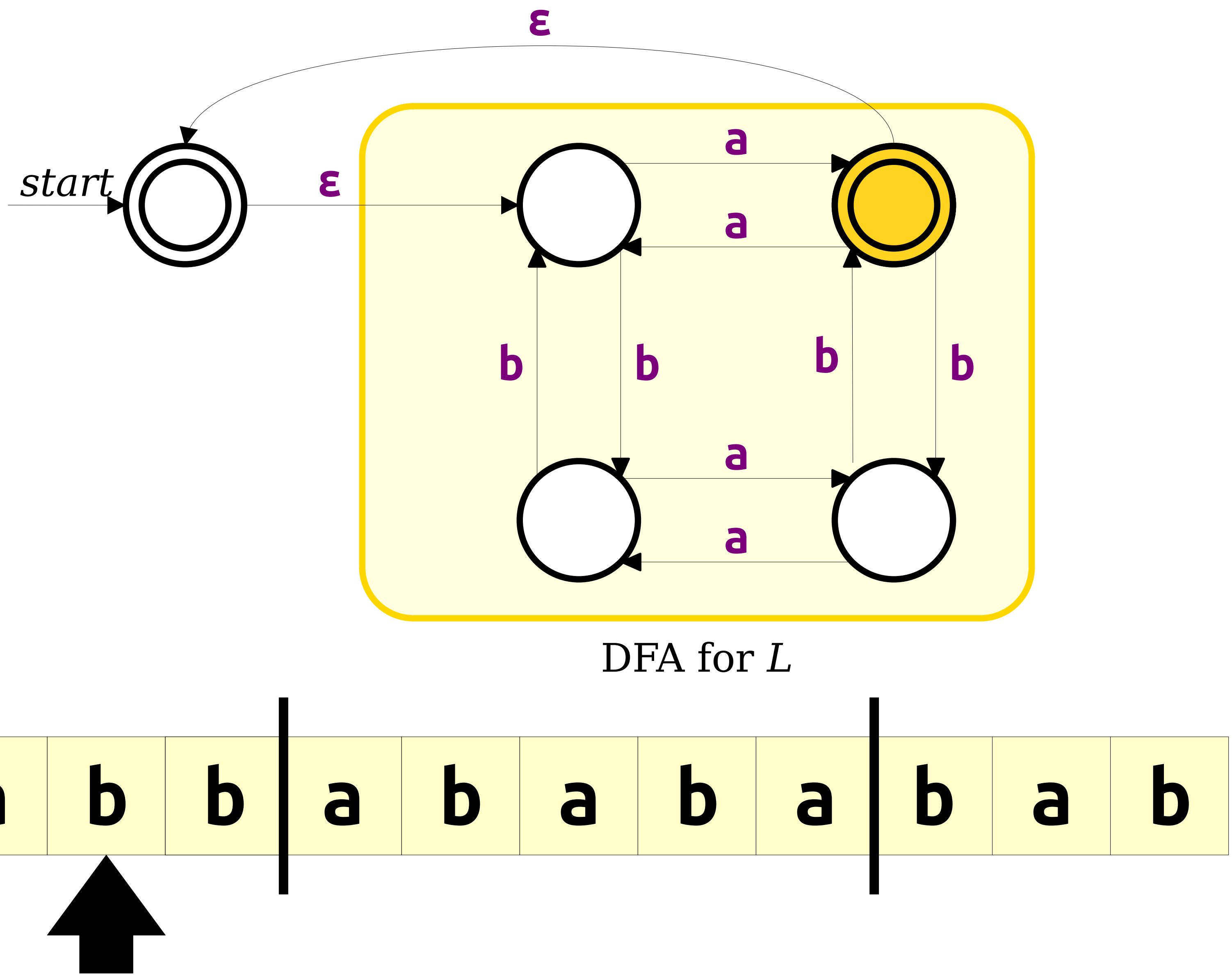
Construct an NFA for L^*.





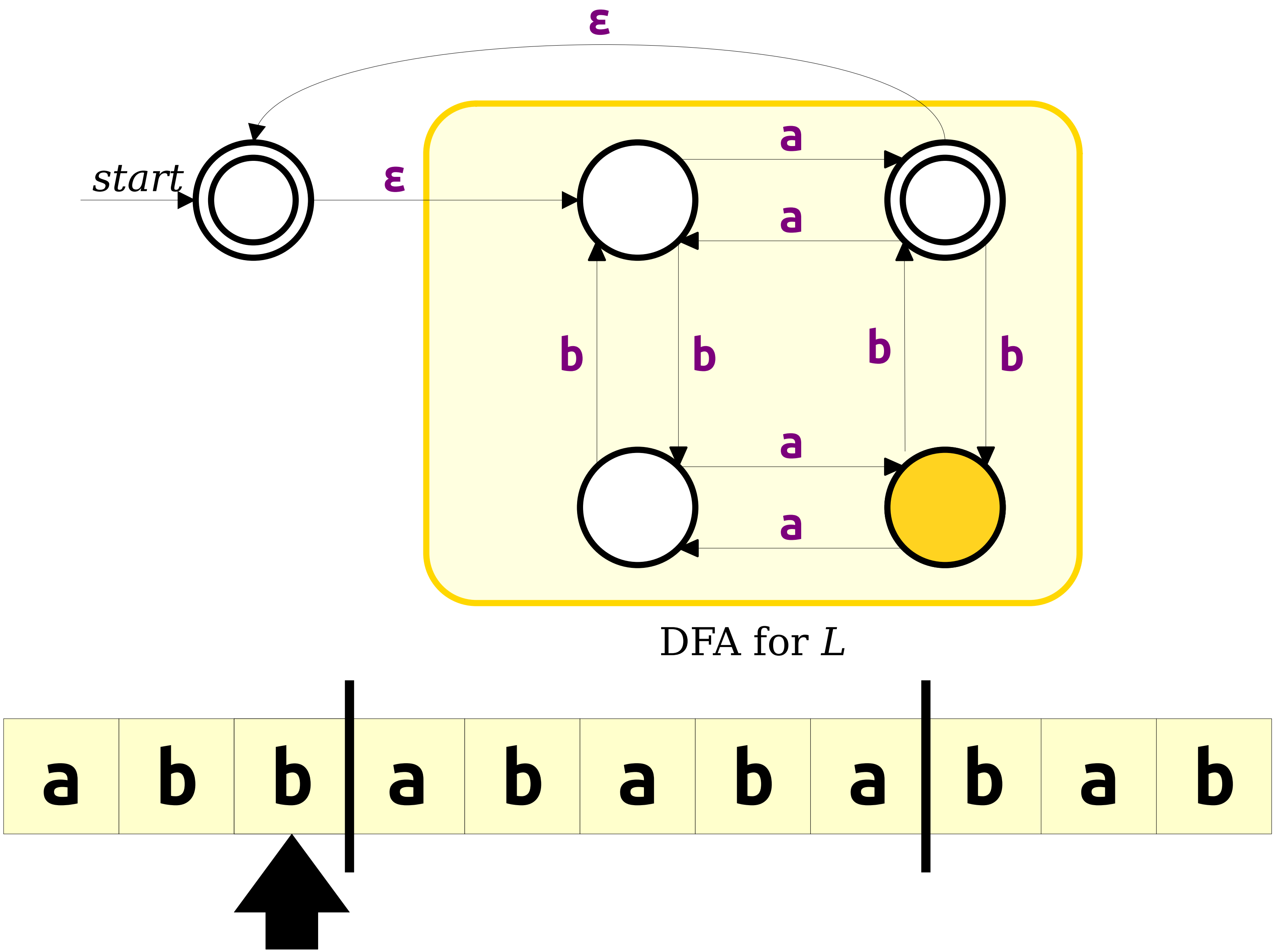
$L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w$ has an odd number of **a**'s and an even number of **b**'s $\}$

Construct an NFA for L^*.



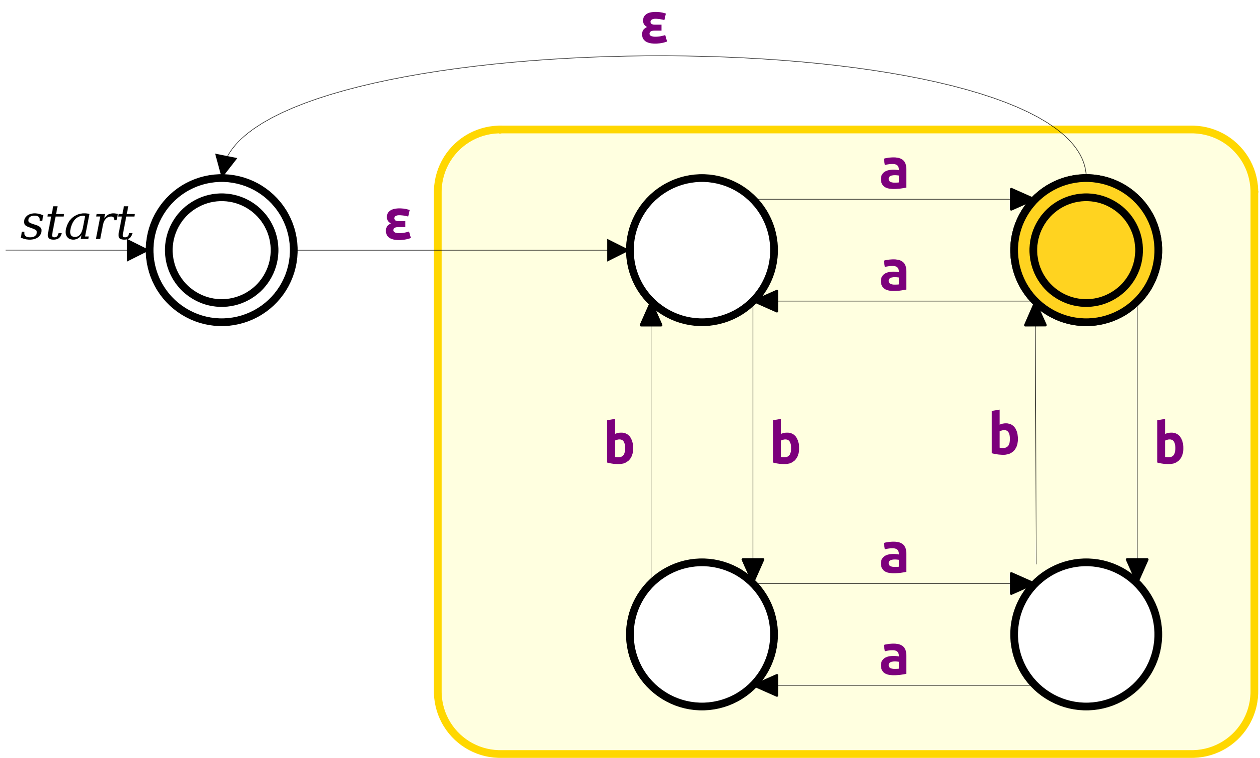
$L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w$ has an odd number of **a**'s and an even number of **b**'s $\}$

Construct an NFA for L^*.

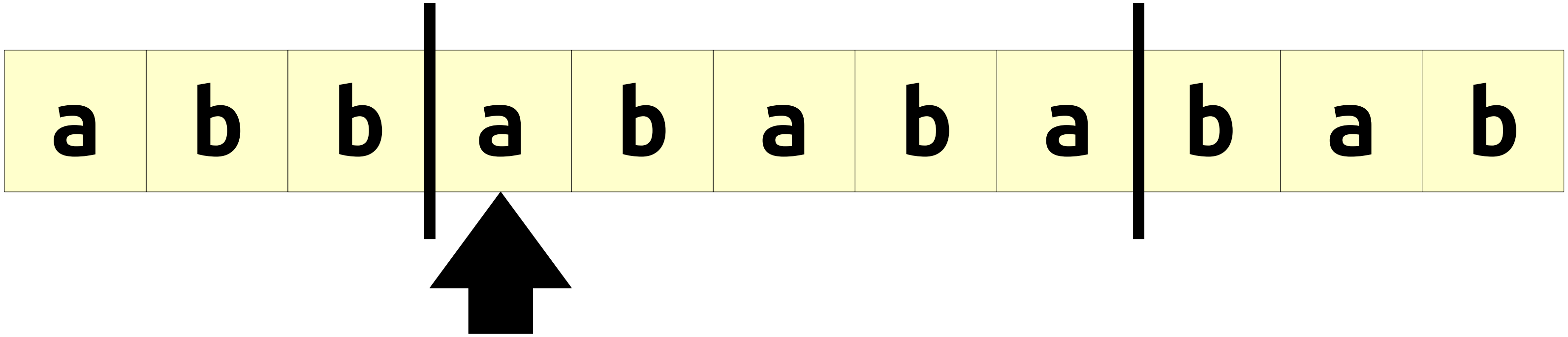


L = { w ∈ {**a**, **b**}* | w has an odd number of **a**'s and
an even number of **b**'s }

Construct an NFA for L*.

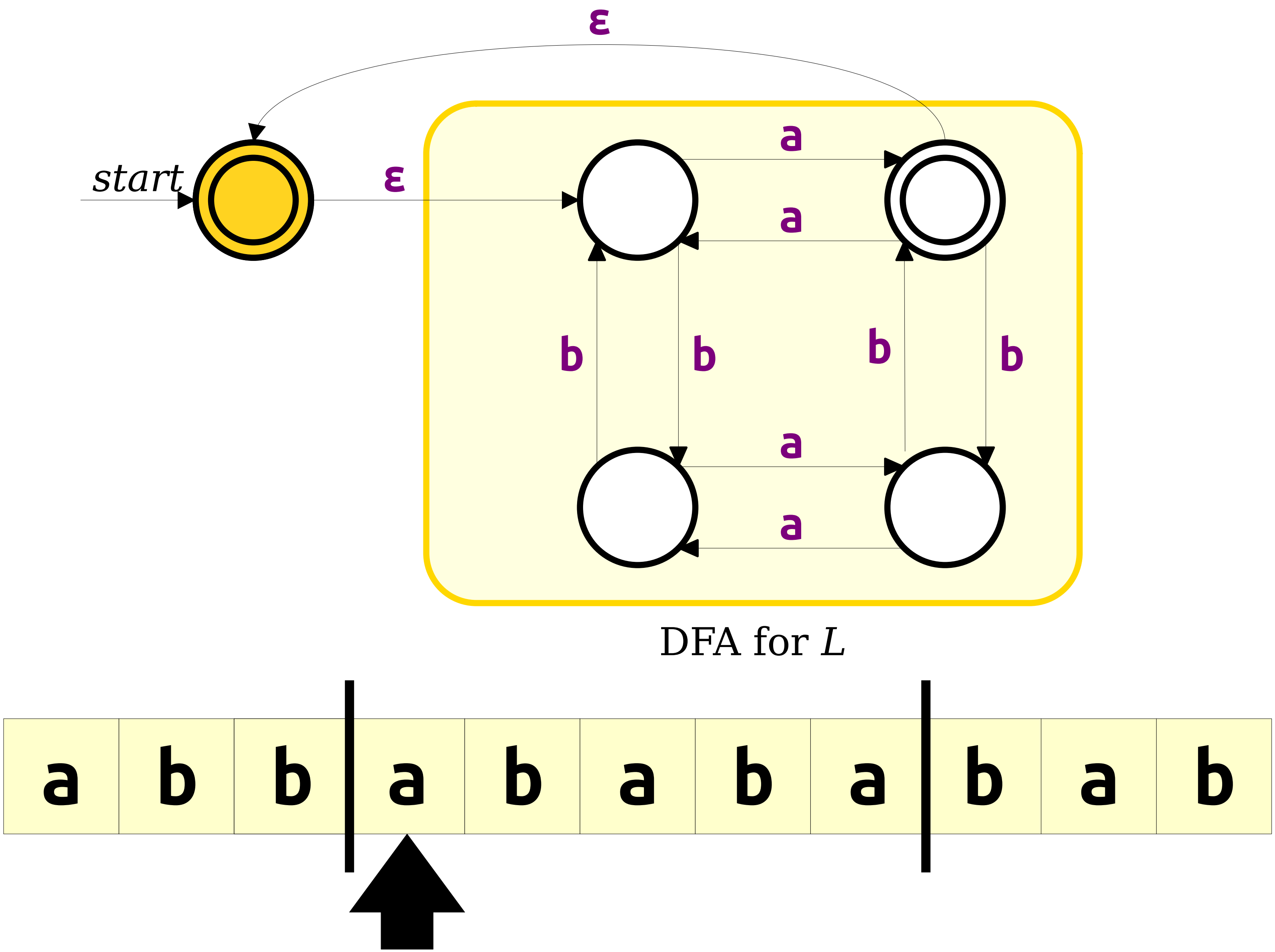


DFA for L



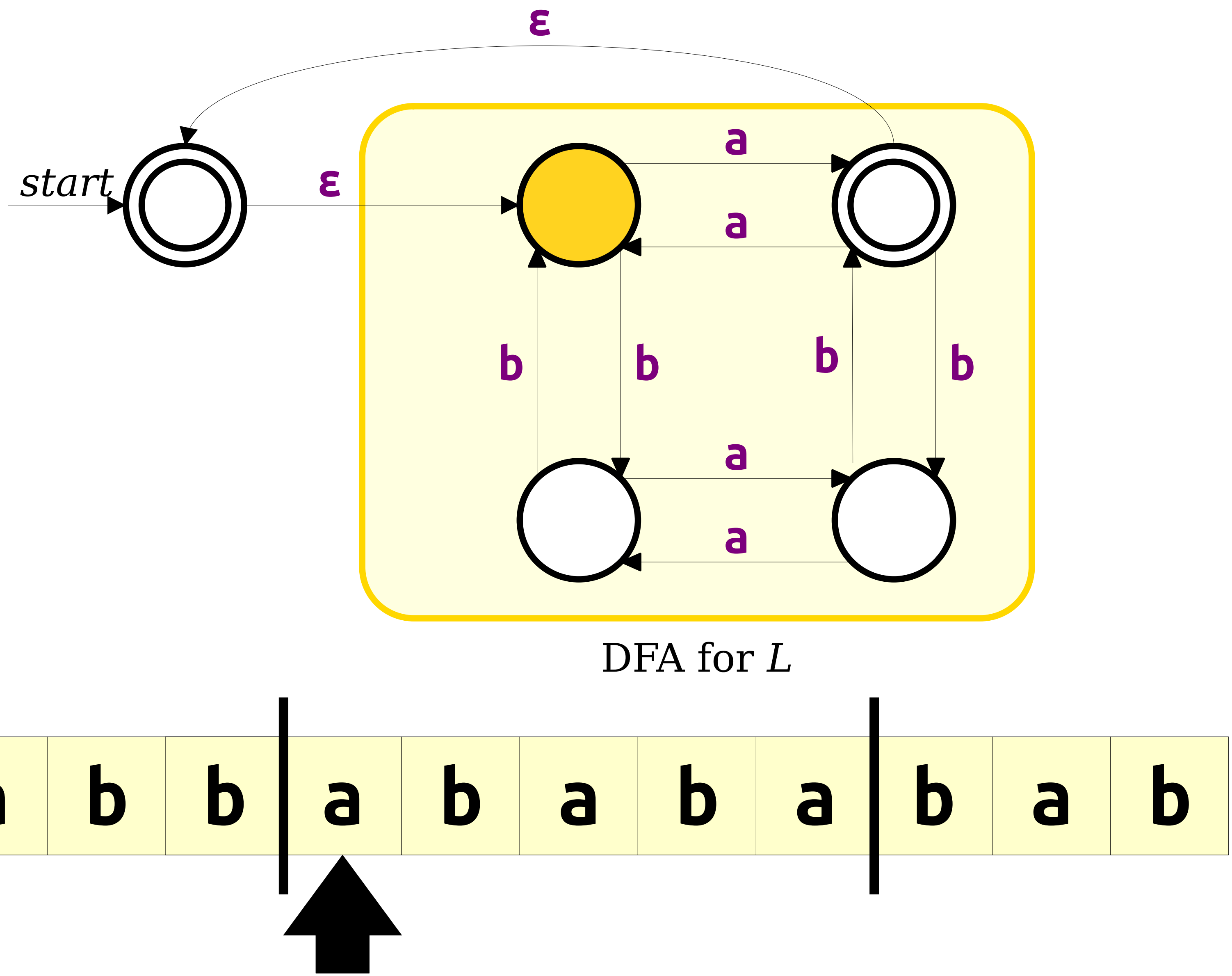
$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ has an odd number of **a**'s and an even number of **b**'s $\}$

Construct an NFA for L^*.



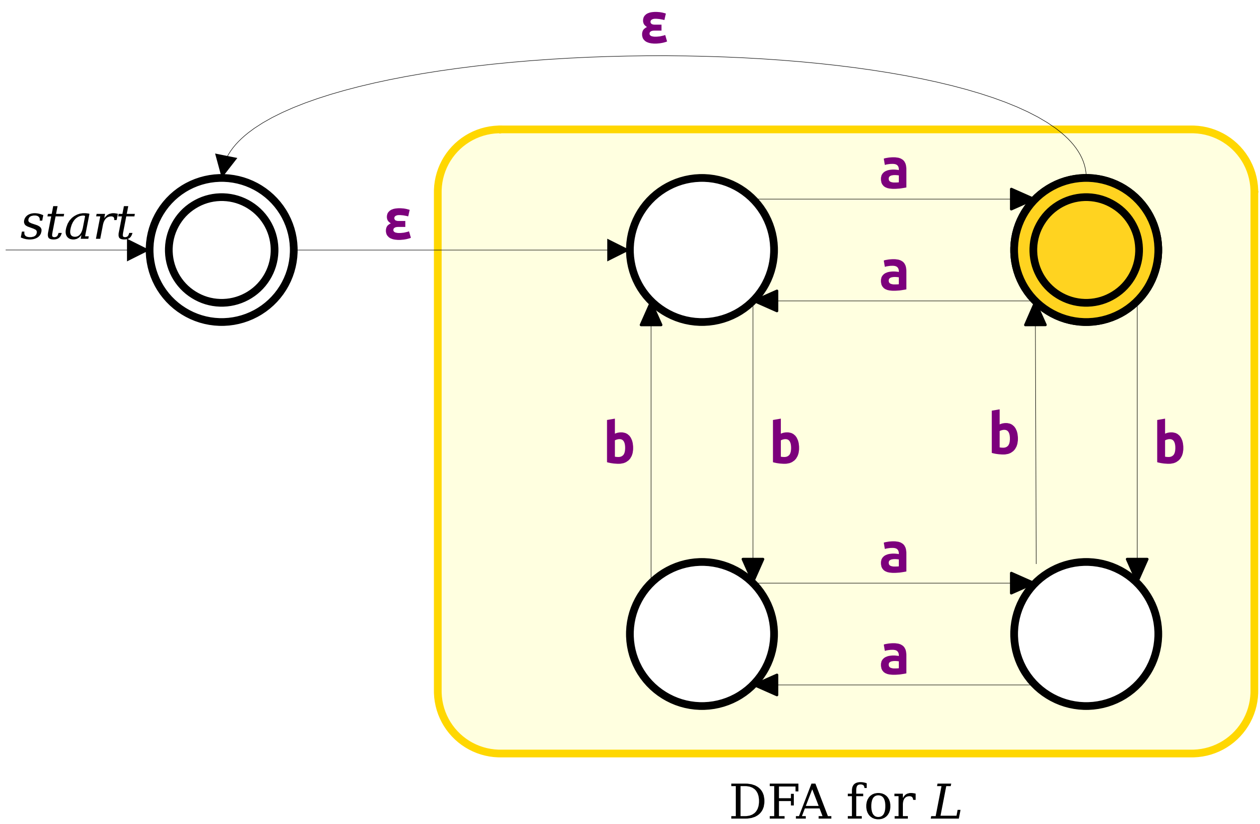
L = { *w* ∈ {a, b}* | *w* has an odd number of a's and an even number of b's }

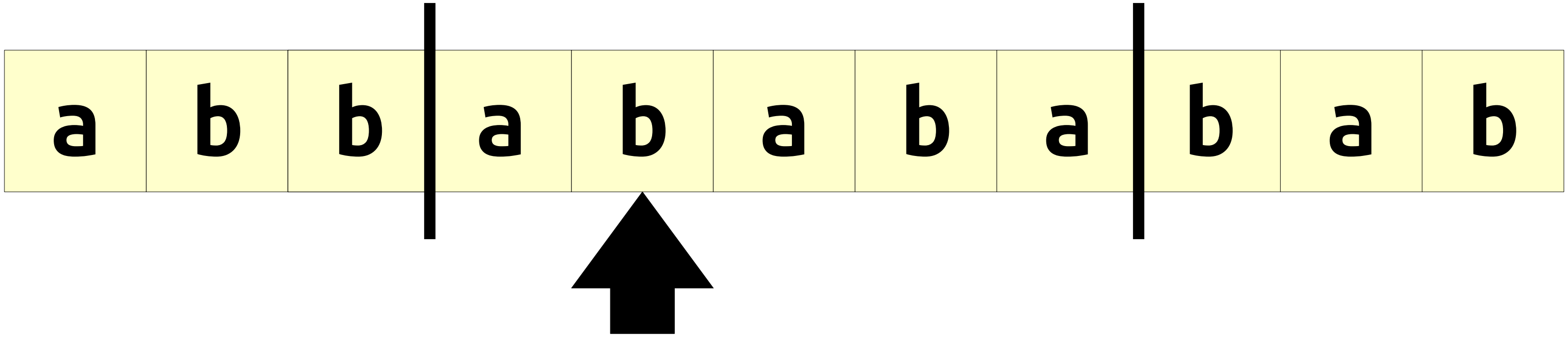
Construct an NFA for *L**.



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

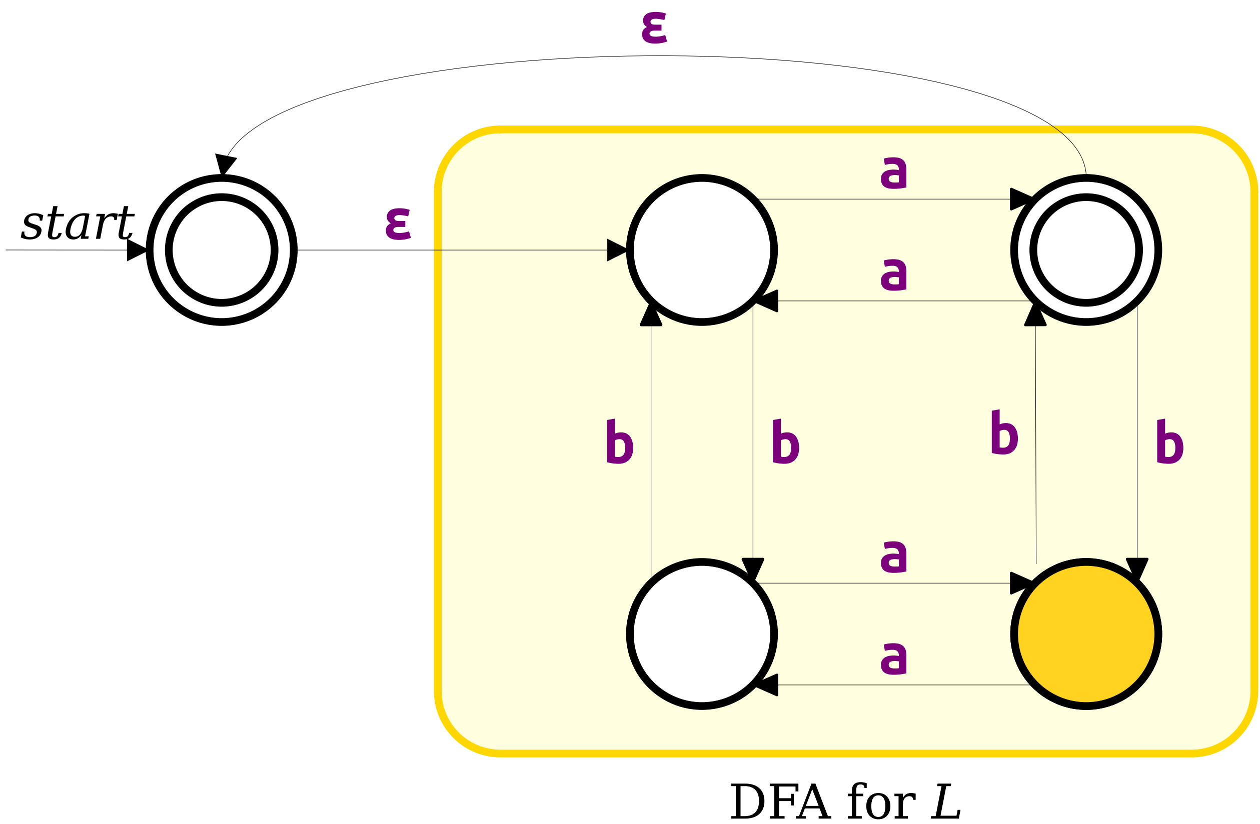
Construct an NFA for L^*.

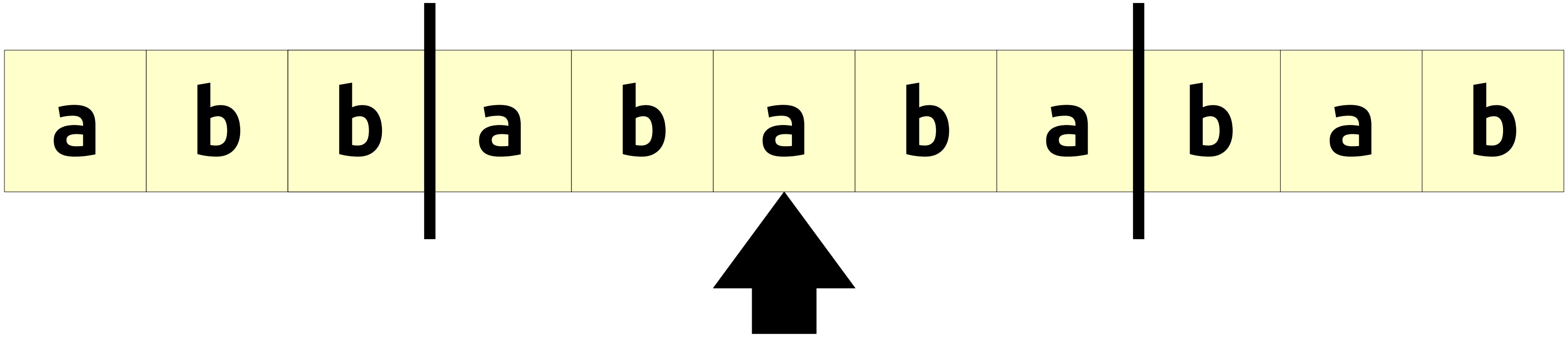




$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ has an odd number of **a**'s and an even number of **b**'s $\}$

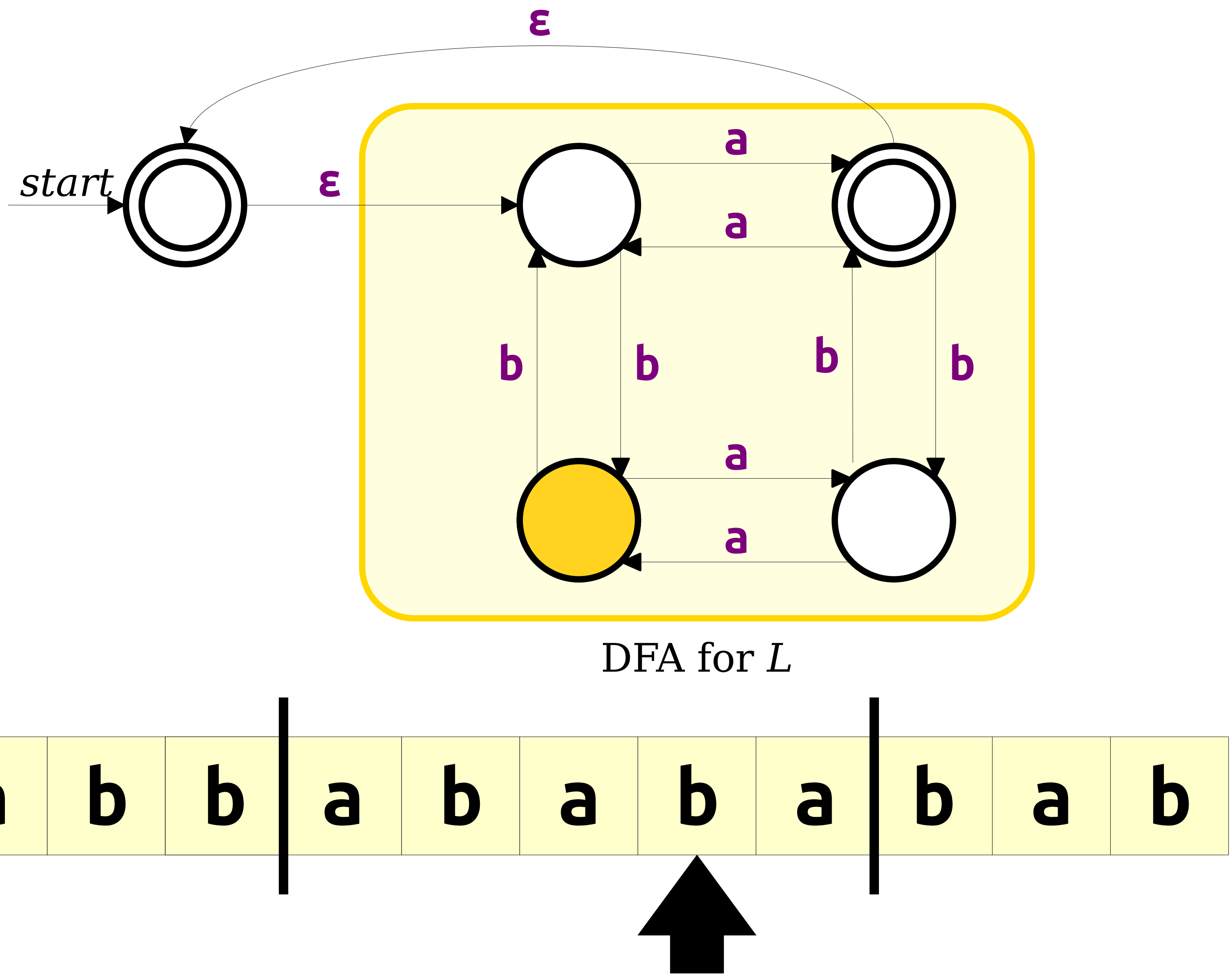
Construct an NFA for L^*.





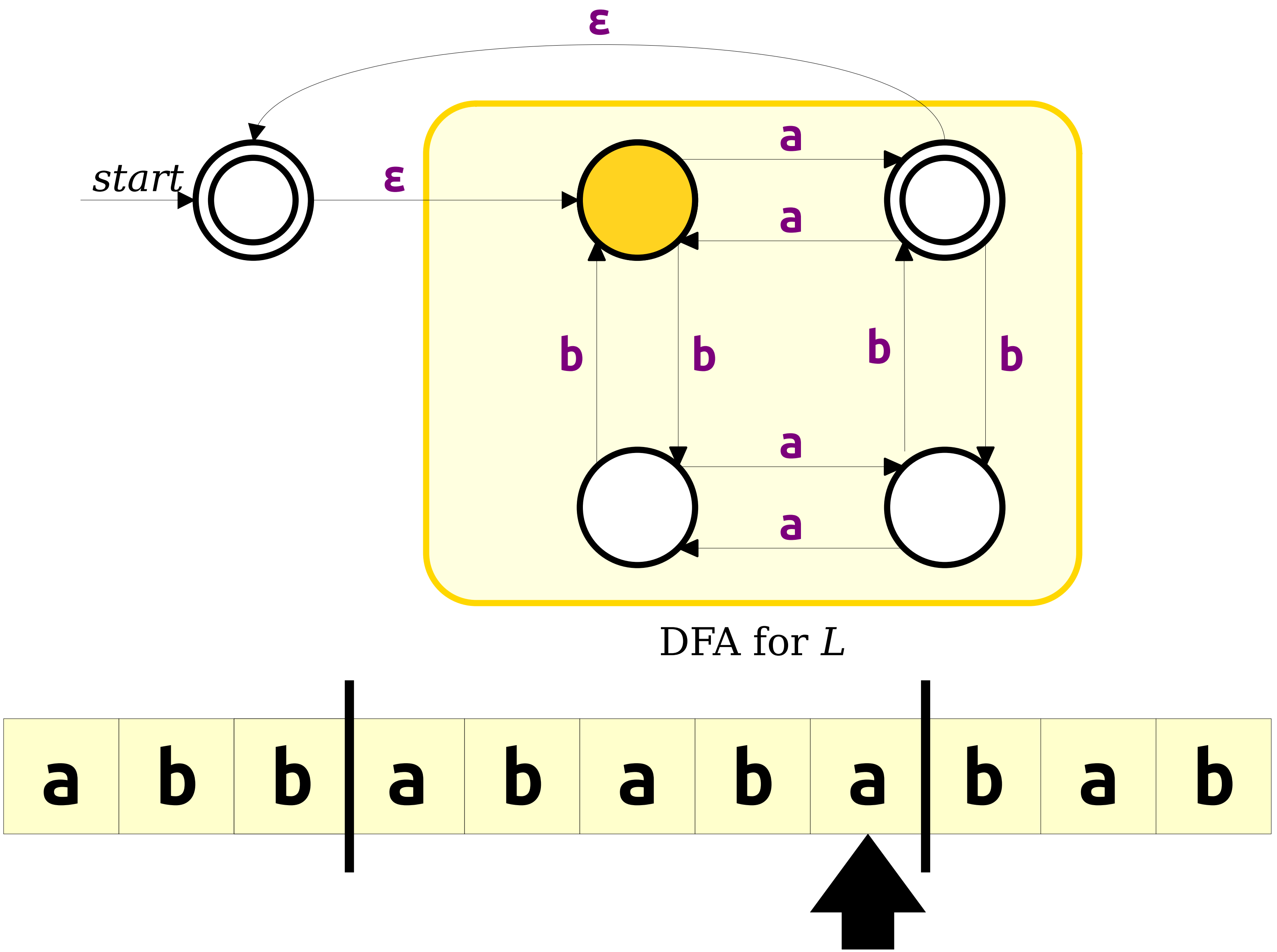
L = { w ∈ {**a**, **b**}* | w has an odd number of **a**’s and an even number of **b**’s }

Construct an NFA for L*.



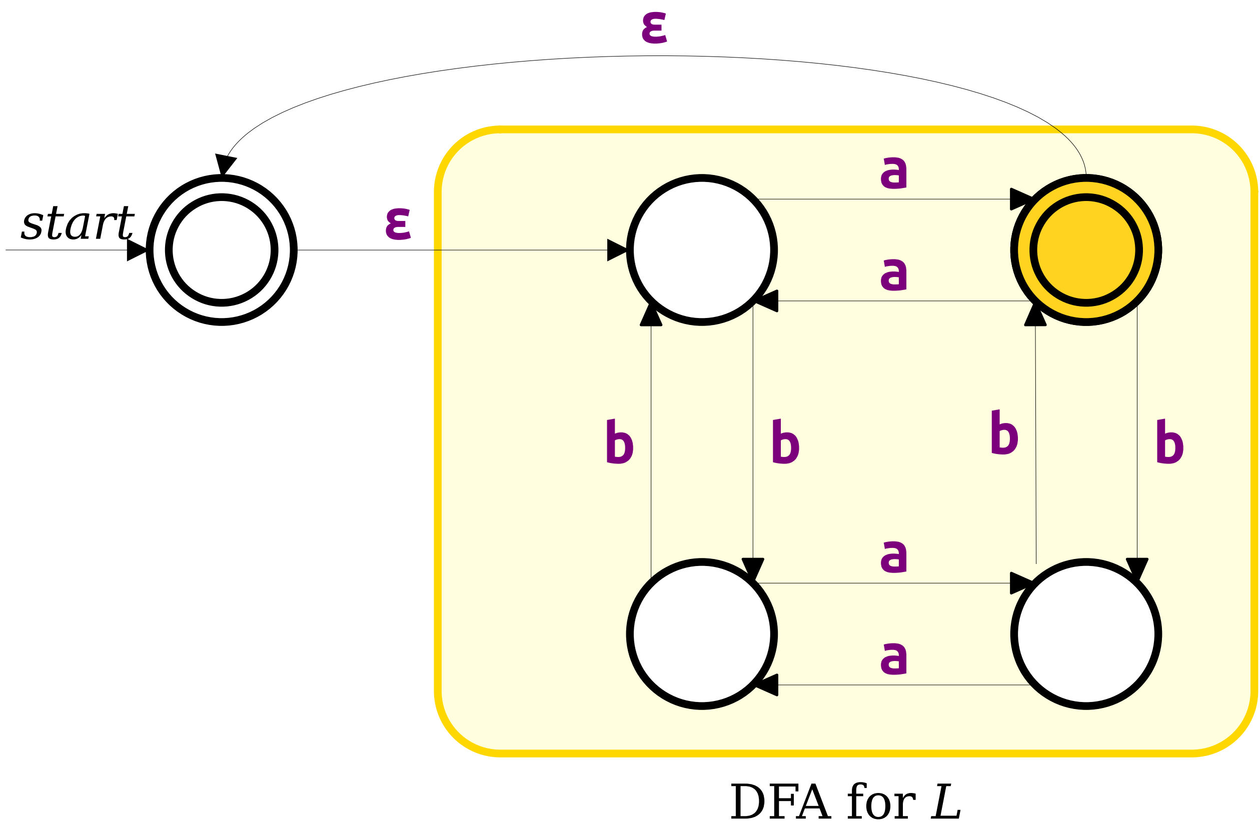
L = { $w \in \{a, b\}^*$ | w has an odd number of a's and an even number of b's }

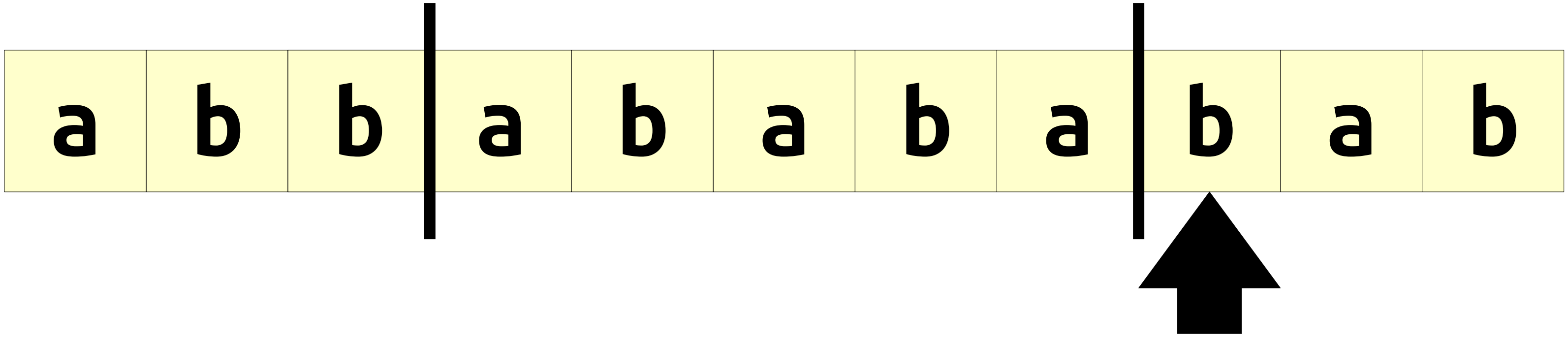
Construct an NFA for L^*.



L = { w ∈ {**a**, **b**}* | w has an odd number of **a**'s and an even number of **b**'s }

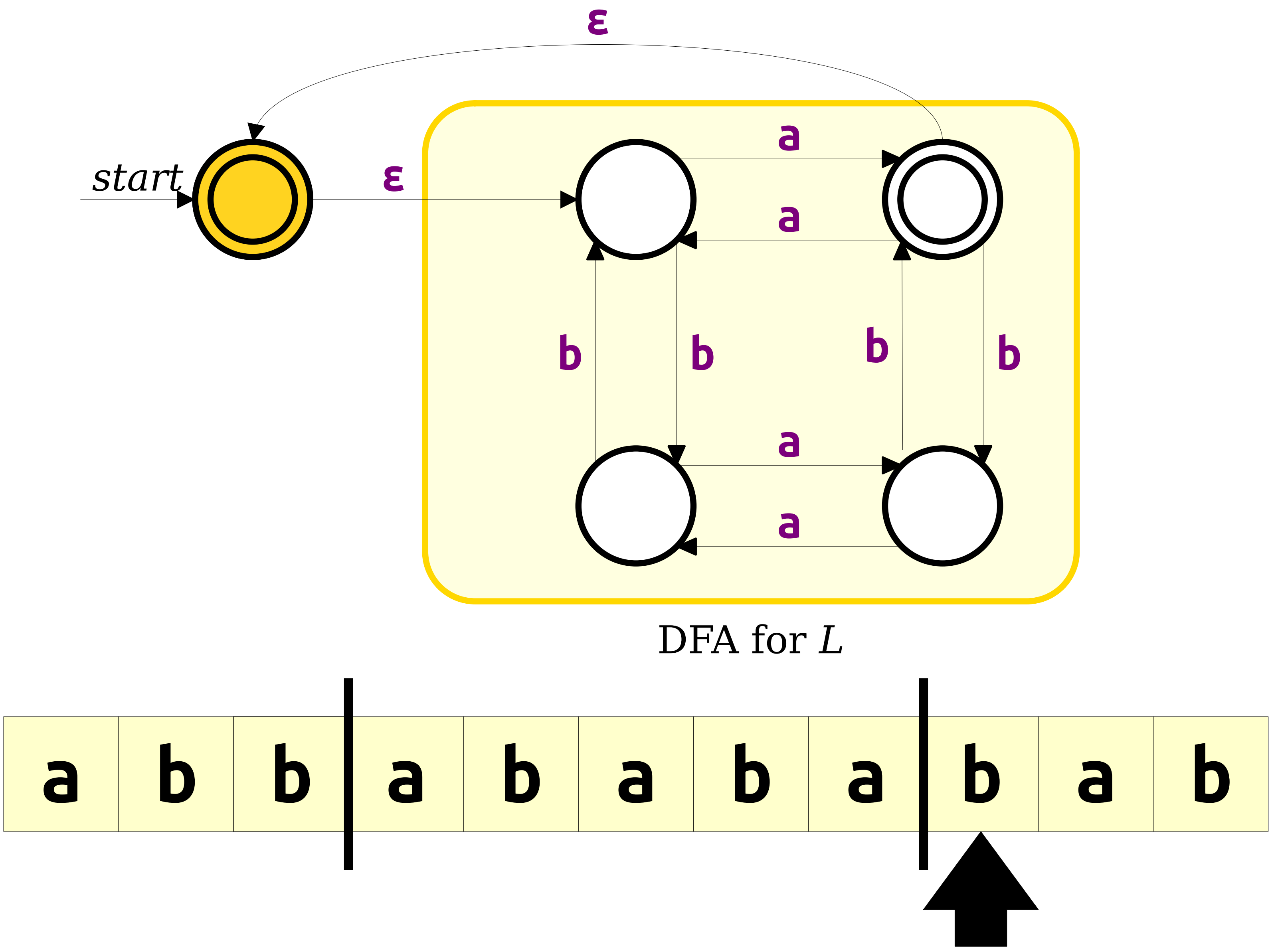
Construct an NFA for L*.





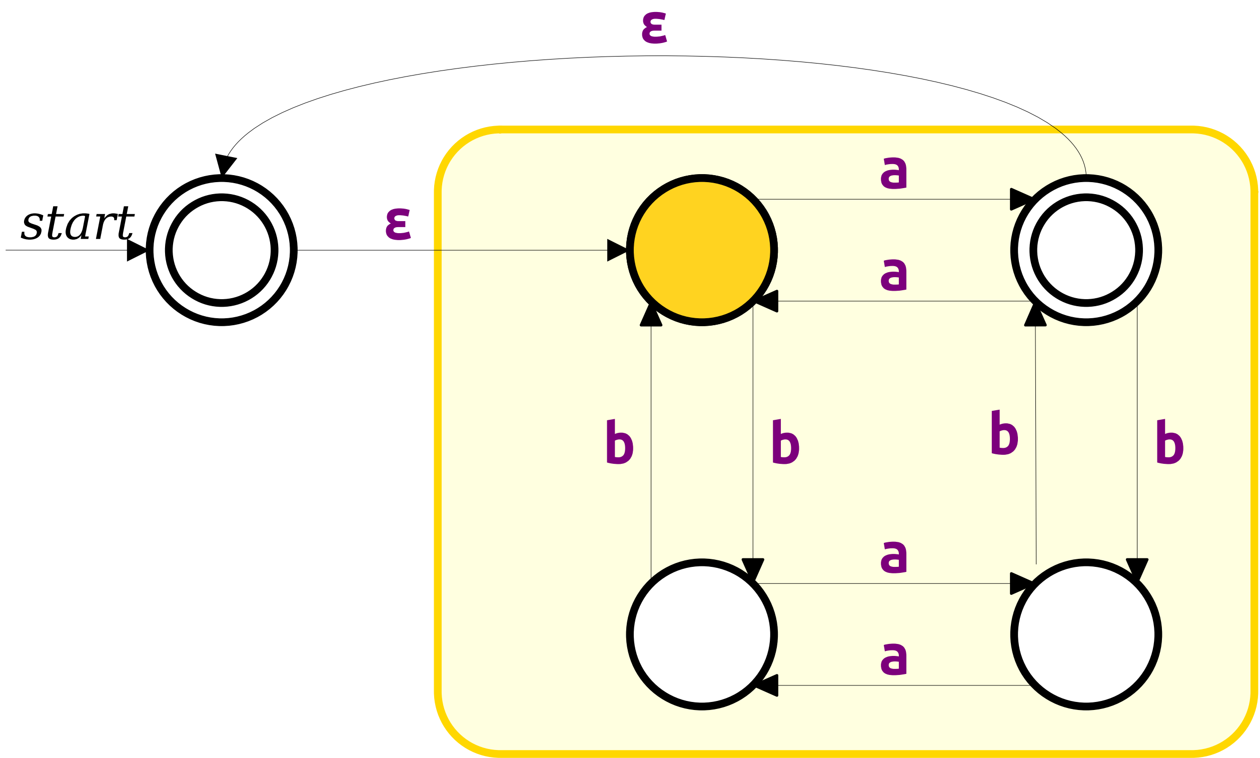
$L = \{ w \in \{$**a**, **b**$\}^* \mid w$ has an odd number of **a**'s and an even number of **b**'s $\}$

Construct an NFA for L^*.

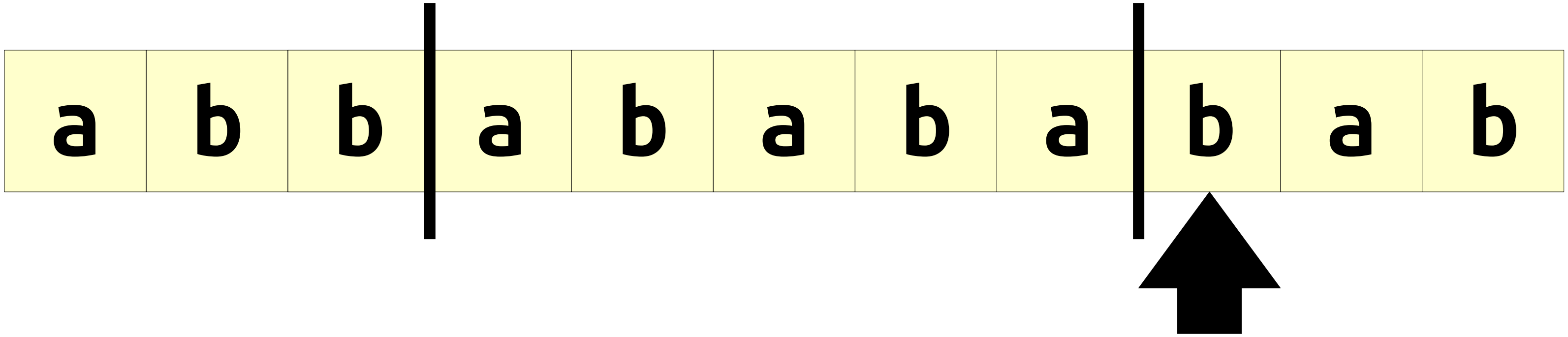


$L = \{\ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w$ has an odd number of **a**'s and an even number of **b**'s $\}$

Construct an NFA for L^*.

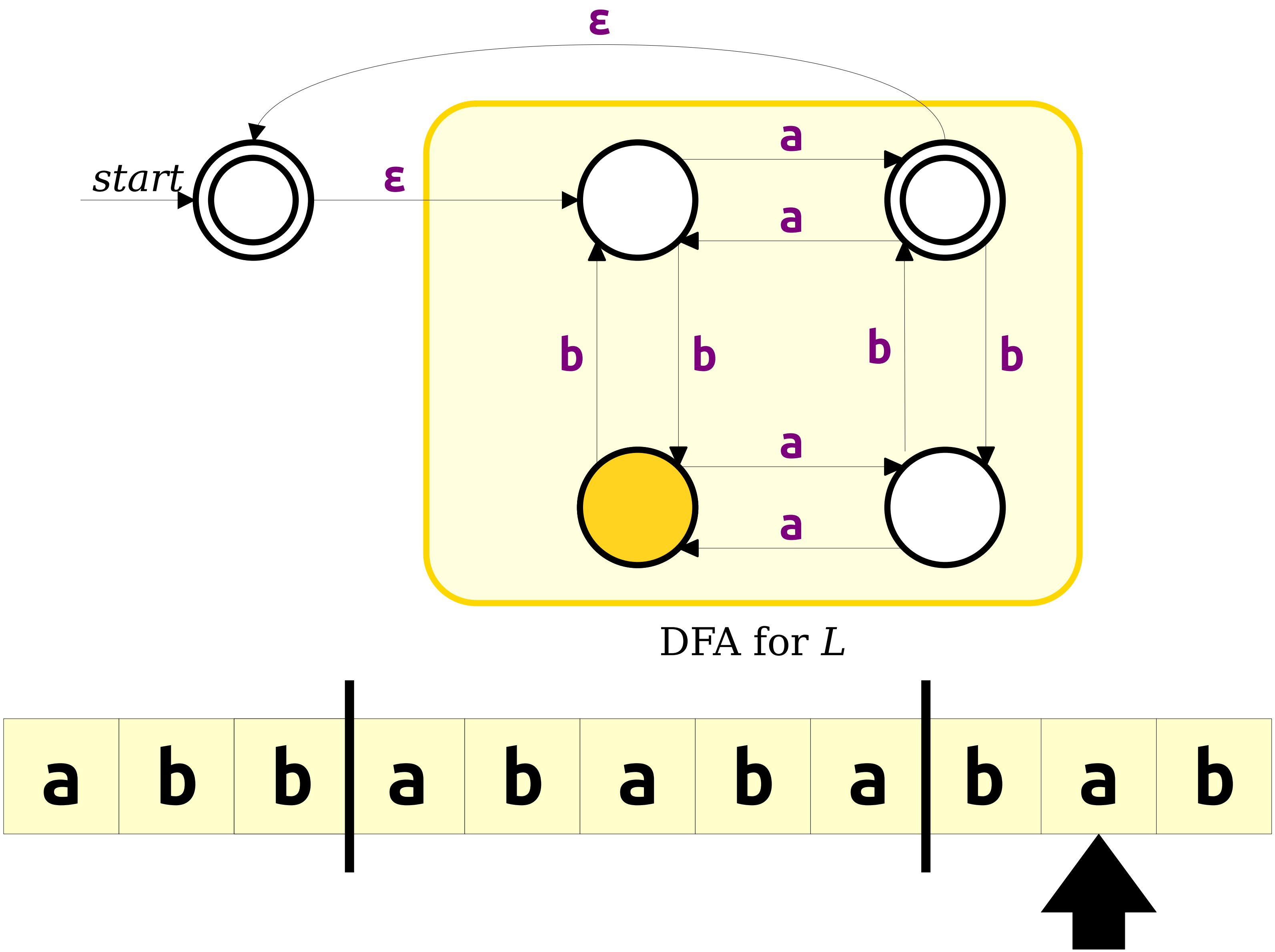


DFA for L



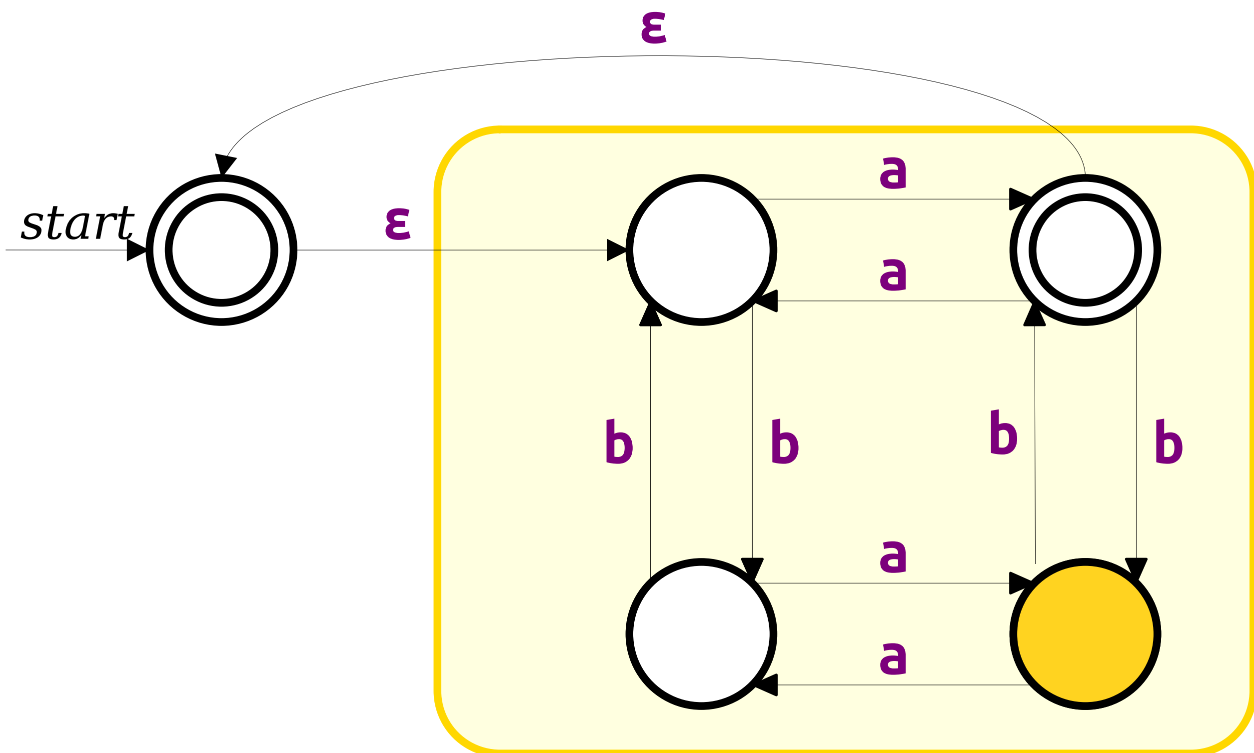
$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ has an odd number of **a**'s and an even number of **b**'s $\}$

Construct an NFA for L^*.

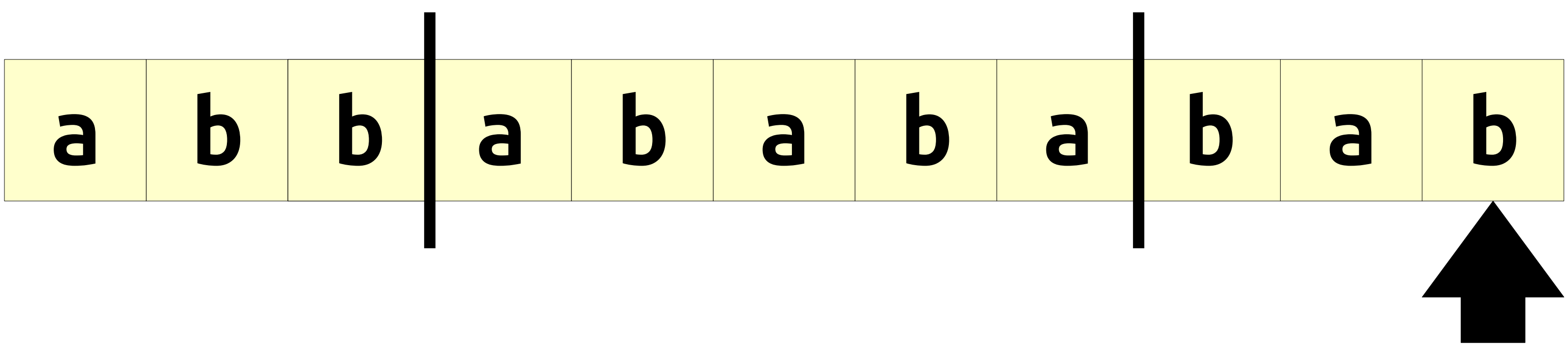


$L = \{\ w \in \{$**a**, **b**$\}^* \mid w$ has an odd number of **a**’s and an even number of **b**’s $\}$

Construct an NFA for L^*.

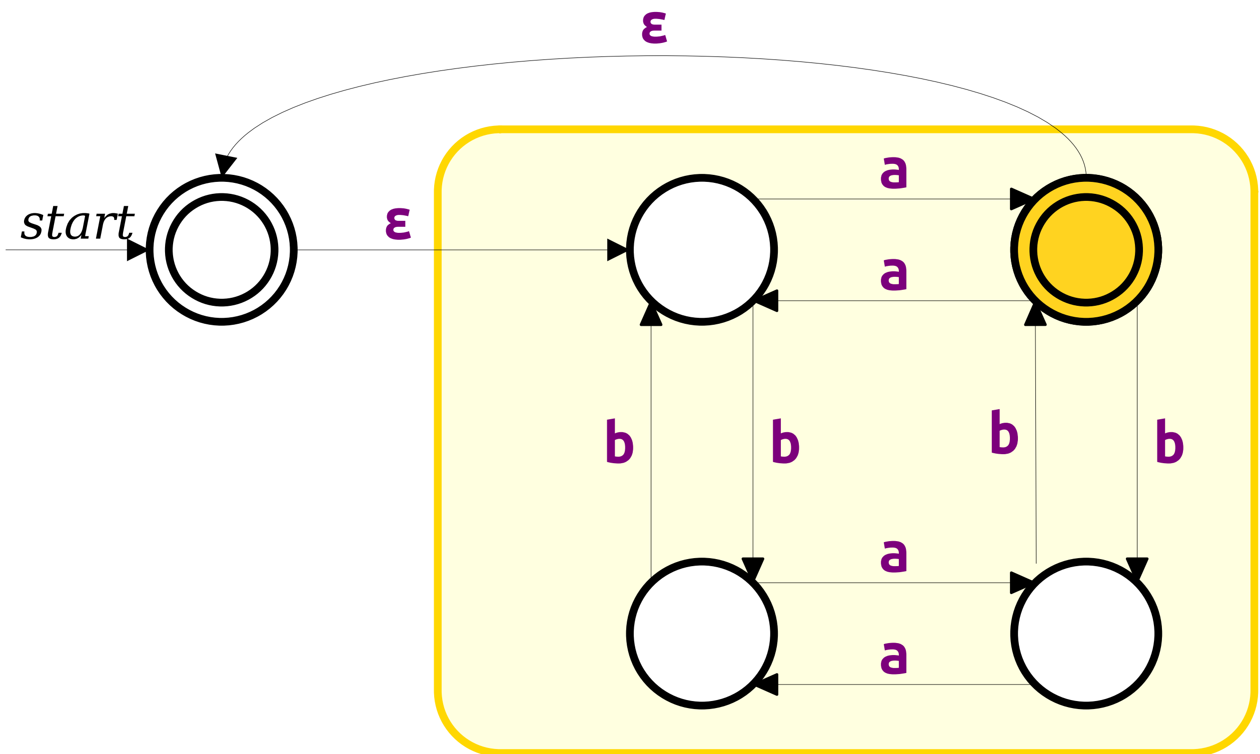


DFA for L

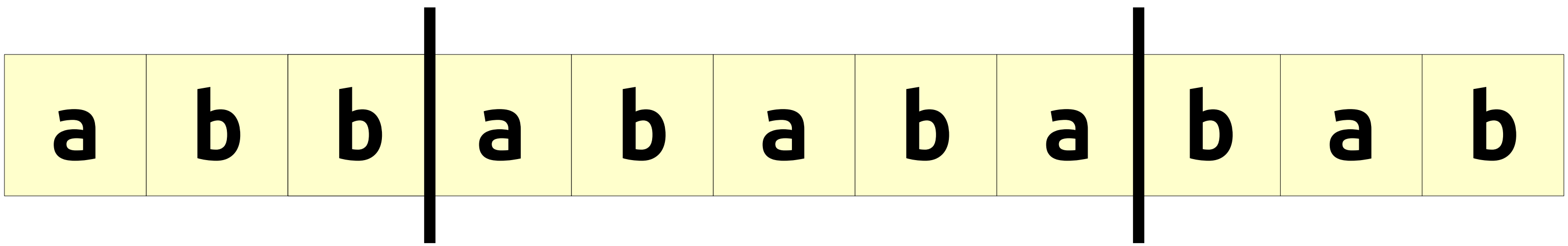


L = { w ∈ {**a**, **b**}* | w has an odd number of **a**'s and
an even number of **b**'s }

Construct an NFA for L^*.

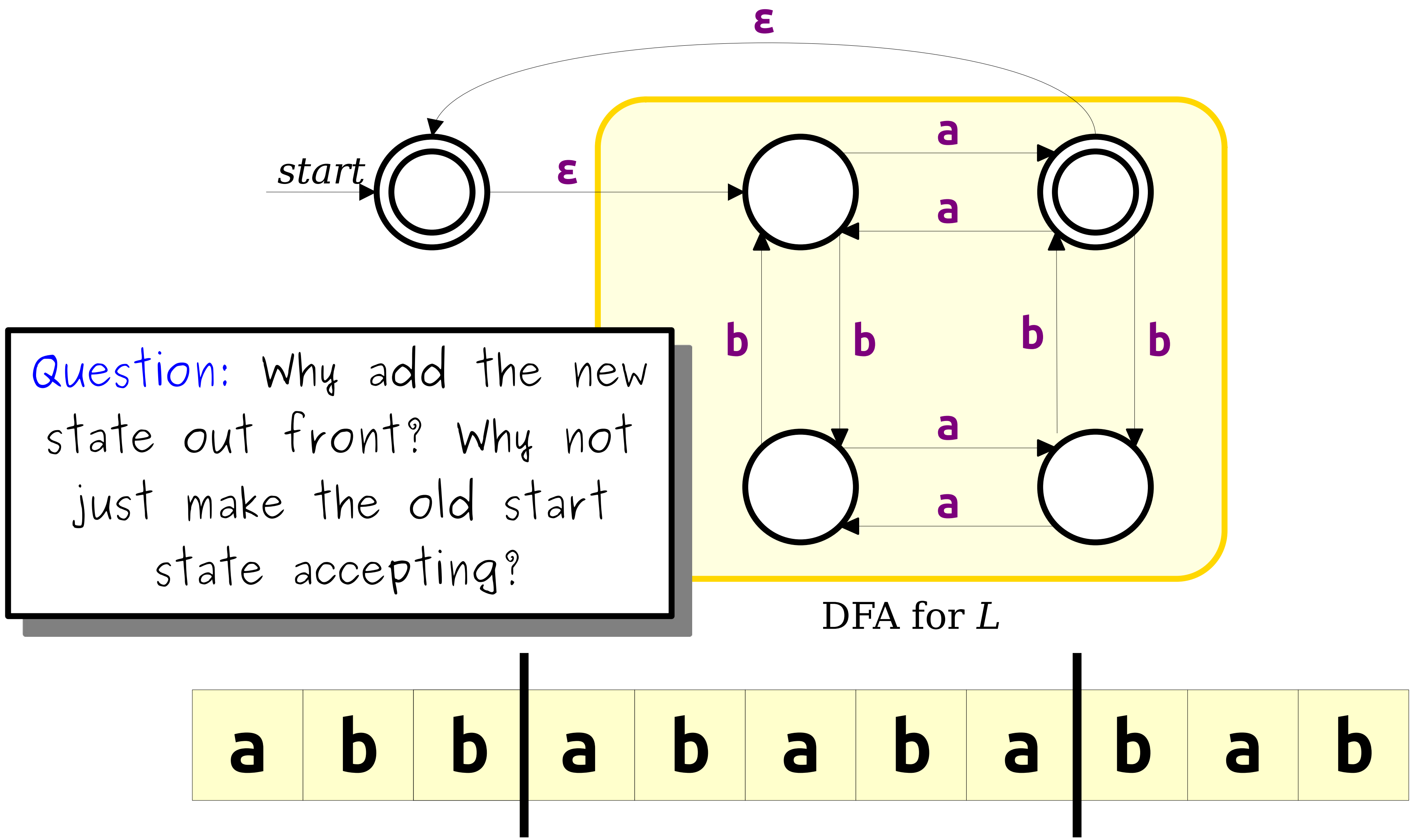


DFA for L



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

Construct an NFA for L^*.



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

Construct an NFA for L^*.

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. **Language Exponentiation and Kleene Star**
9. Summary of Closure Properties
10. What's Next?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. What’s Next?

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. **Summary of Closure Properties**
10. What’s Next?

Closure Properties

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

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. **Summary of Closure Properties**
10. What’s Next?

Finite Automata

Part 3

Finite Automata

Part 3

1. Recap from Last Time
2. How Powerful Are NFAs?
3. The Subset Construction
4. Regular Languages Revisited
5. Announcements
6. Union and Intersection
7. String Concatenation
8. Language Exponentiation and Kleene Star
9. Summary of Closure Properties
10. **What's Next?**

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!