CMPU 240 · Theory of Computation

Properties of Regular Languages

10 February 2026

Assignment 2

Due now

Corrections due Thursday

Where are we?

If D is a deterministic finite automaton (DFA), the *language of* D, denoted $L(D)$, is

$$\{w \in \Sigma^* \mid D \text{ accepts } w\}.$$

If L is a language and $L(D) = L$, we say that D *recognizes* the language L.

A language L is called a ***regular language*** if there is some DFA D such that $L(D) = L$.

DFAs have exactly one transition from each state for each input symbol.

NFAs – *non*deterministic finite automata – can have missing transitions, or multiple transitions can be defined on the same input symbol.

NFAs also have a special type of transition called an *ε*-transition. An NFA can follow any number of *ε*-transitions without consuming any input.

An NFA accepts an input if *any possible series of choices* leads to an accept state.

- Using the massive parallelism intuition, an NFA can be thought of as a DFA that can be in many states at once.
- At each point in time, when an NFA needs to follow a transition, it tries all the options at the same time.

Nondeterminism and ε-transitions don't change the power of a finite automaton.

We can convert a DFA to an NFA (trivially) and we can convert an NFA to a DFA (using the subset construction).

Nondeterminism and ε-transitions don't change the power of a finite automaton.

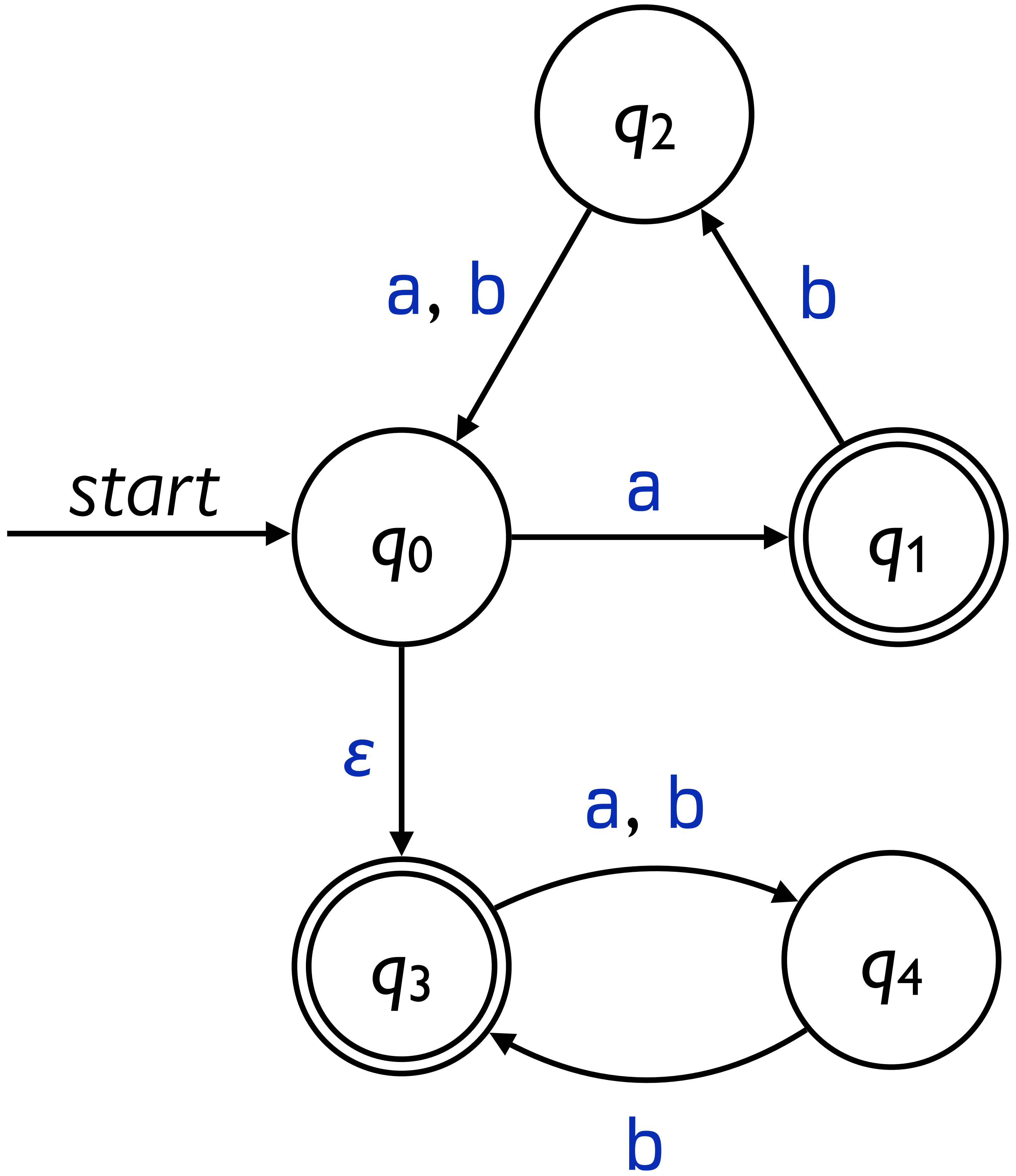
We can convert a DFA to an NFA (trivially) and we can convert an NFA to a DFA (using the subset construction).

$$\delta(q_0, \text{a}) = q_1$$
$$\delta(q_1, \text{a}) = q_0$$

$$\delta(q_0, \text{a}) = \{q_1\}$$
$$\delta(q_1, \text{a}) = \{q_0\}$$

Nondeterminism and ε-transitions don't change the power of a finite automaton.

We can convert a DFA to an NFA (trivially) and we can convert an NFA to a DFA (using the subset construction).



	State	a	b
→ *	$\{q_0, q_3\}$	$\{q_1, q_4\}$	$\{q_4\}$
*	$\{q_1, q_4\}$	$\varnothing$	$\{q_2, q_3\}$
	$\{q_4\}$	$\varnothing$	$\{q_3\}$
*	$\{q_2, q_3\}$	$\{q_0, q_3, q_4\}$	$\{q_0, q_3, q_4\}$
*	$\{q_3\}$	$\{q_4\}$	$\{q_4\}$
*	$\{q_0, q_3, q_4\}$	$\{q_1, q_4\}$	$\{q_3, q_4\}$
*	$\{q_3, q_4\}$	$\{q_4\}$	$\{q_3, q_4\}$
	$\varnothing$	$\varnothing$	$\varnothing$

A language L is called a ***regular language*** if there is some DFA D such that $L(D) = L$.

Equivalently…

A language L is called a ***regular language*** if there is some NFA N such that $L(N) = L$.

But what *are* the regular languages?

time only.

7. <u>Regular Events</u>:

7.1 <u>"Regular events" defined</u>: We shall presently describe a class of events which we will call "regular events." (We would welcome any suggestions as to a more descriptive term.*)

We assume for the purpose that the events refer to the

Stephen Kleene, "Representation of Events in Nerve Nets and Finite Automata", 1951

We know we have a regular language when we design an automaton for it.

Let's consider how much can we change one of these regular languages and still know that it's a regular language.

Closure

If x is a natural number and y is a natural number, what do you get if you

add $x + y$?

multiply $x \cdot y$?

subtract $x - y$?

divide x / y?

If A is a set and B is a set, what do you get if you

take their union, $A \cup B$?

take their intersection, $A \cap B$?

take their difference, $A - B$?

A class of objects is *closed* under an operation if applying that operation to one or more elements of the class produces another of the elements.

The *natural numbers* are closed under *addition* and *multiplication*.

The *integers* are closed under *addition*, *multiplication*, and *subtraction*.

Sets are closed under *union*, *intersection*, and *set difference*.

Propositional logic is closed under *conjunction*, *disjunction*, and *negation*.

The *ε-closure* of a set of states is closed under the operation of *following ε-transitions*.

Since languages are sets, we can use standard set operations on them, including

union (∪),

intersection (∩), and

complement.

Complement

The complement of a language

Given a language $L \subseteq \Sigma^*$, the *complement* of that language, denoted $\overline{L}$, is the language of all strings in Σ^* that *aren't* in L:

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

The complement of a language

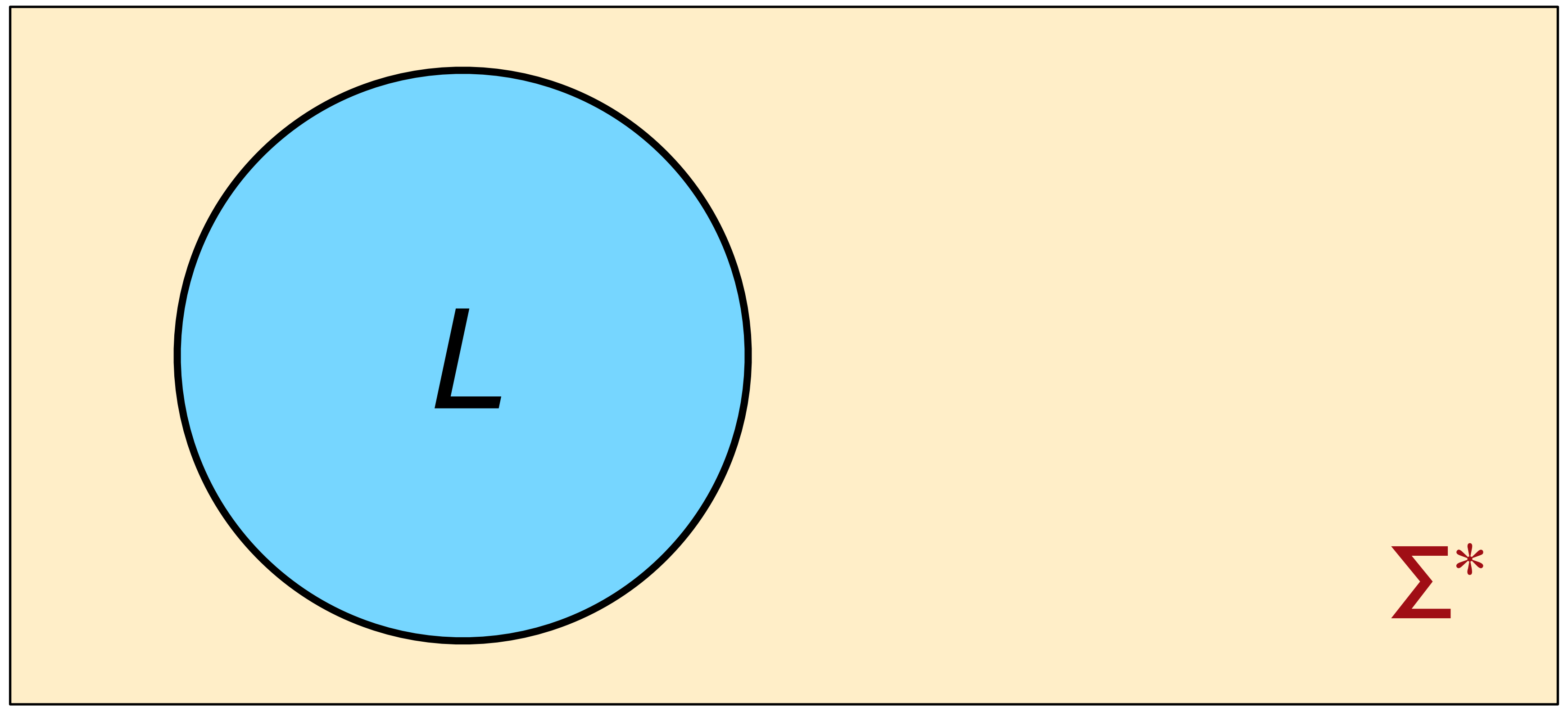
Given a language $L \subseteq \Sigma^*$, the ***complement*** of that language, denoted $\overline{L}$, is the language of all strings in Σ^* that ***aren't*** in L:

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

The complement of a language

Given a language $L \subseteq \Sigma^*$, the ***complement*** of that language, denoted $\overline{L}$, is the language of all strings in Σ^* that ***aren't*** in L:

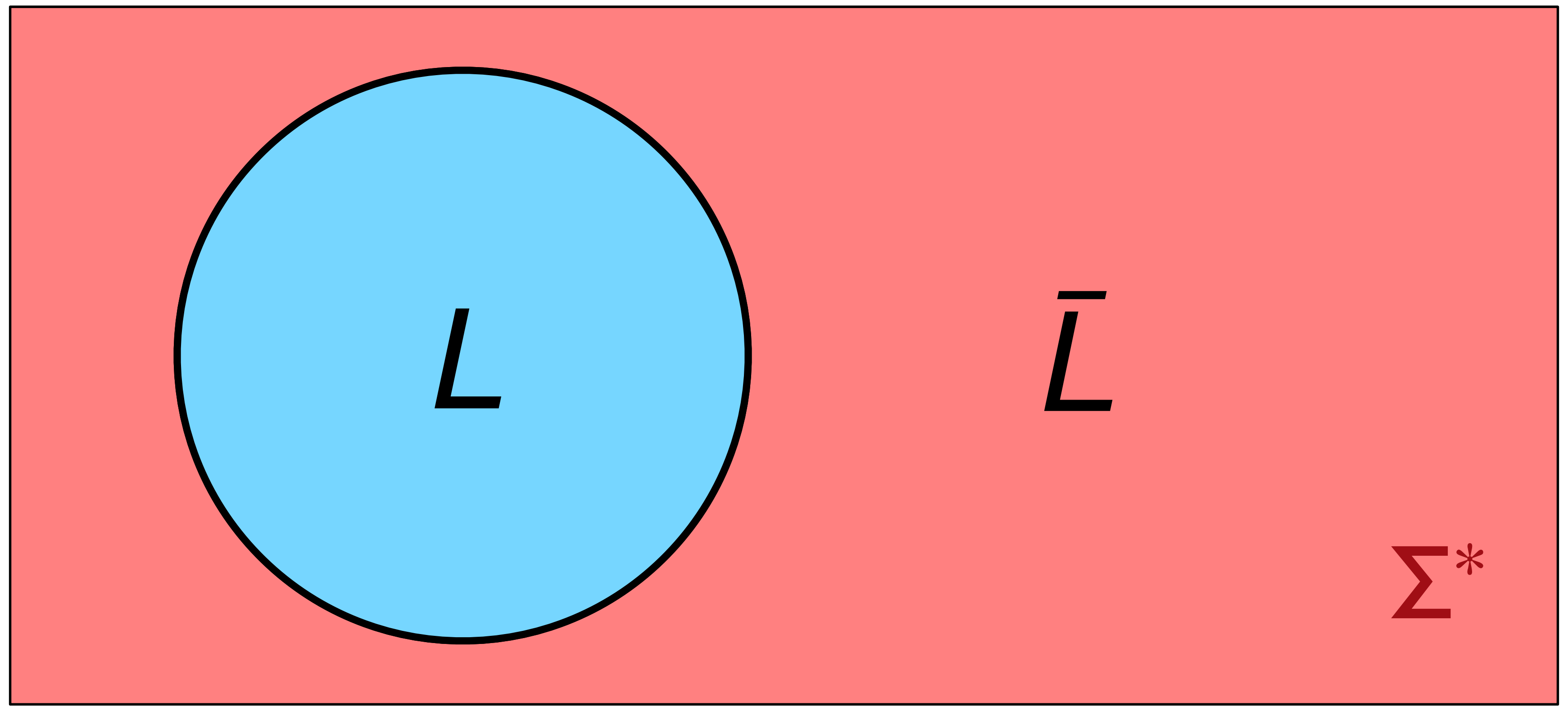
$$\overline{L} = \Sigma^* - L$$



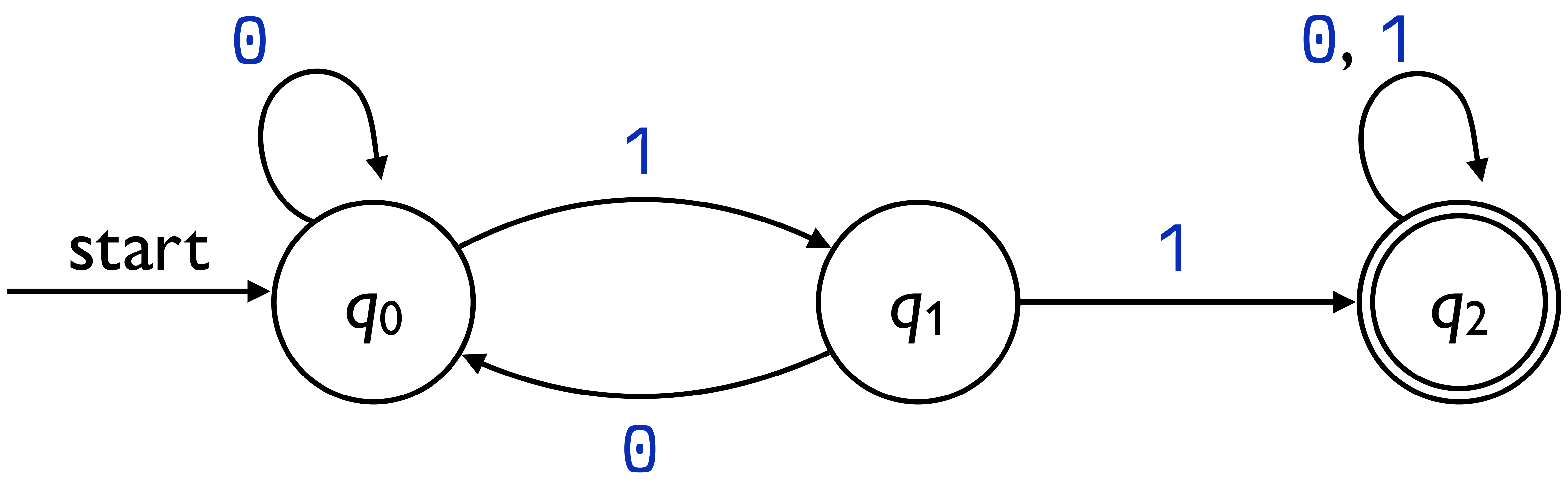
The complement of a language

Given a language $L \subseteq \Sigma^*$, the *complement* of that language, denoted $\overline{L}$, is the language of all strings in Σ^* that ***aren't*** in L:

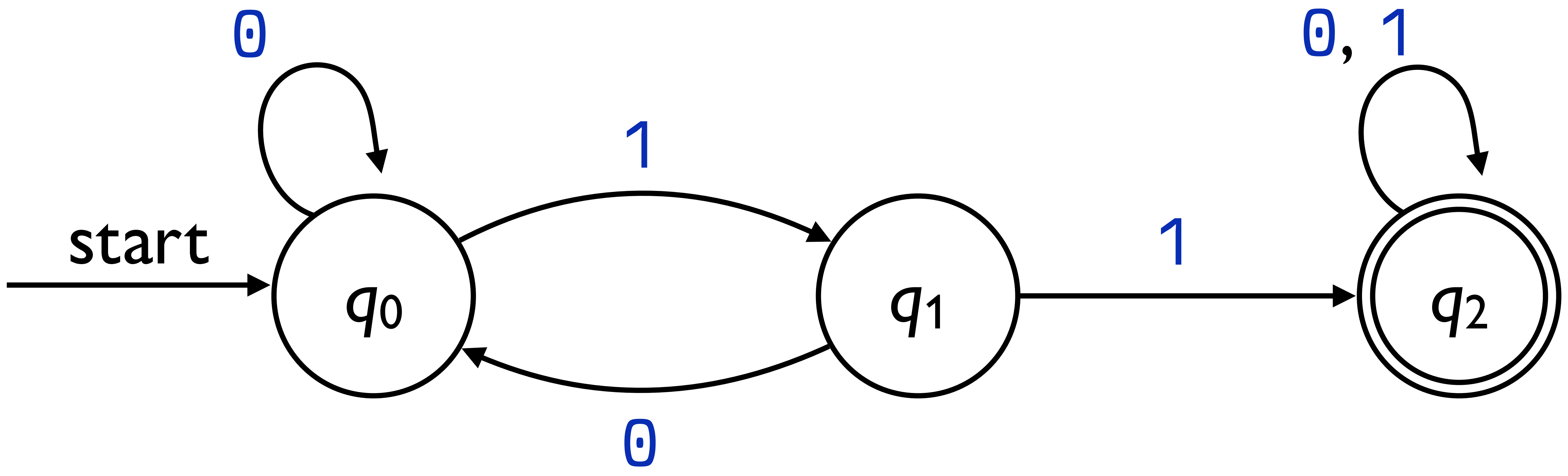
$$\overline{L} = \Sigma^* - L$$



$L = \{w \in \{0, 1\}^* \mid w \text{ contains } 11 \text{ as a substring}\}$

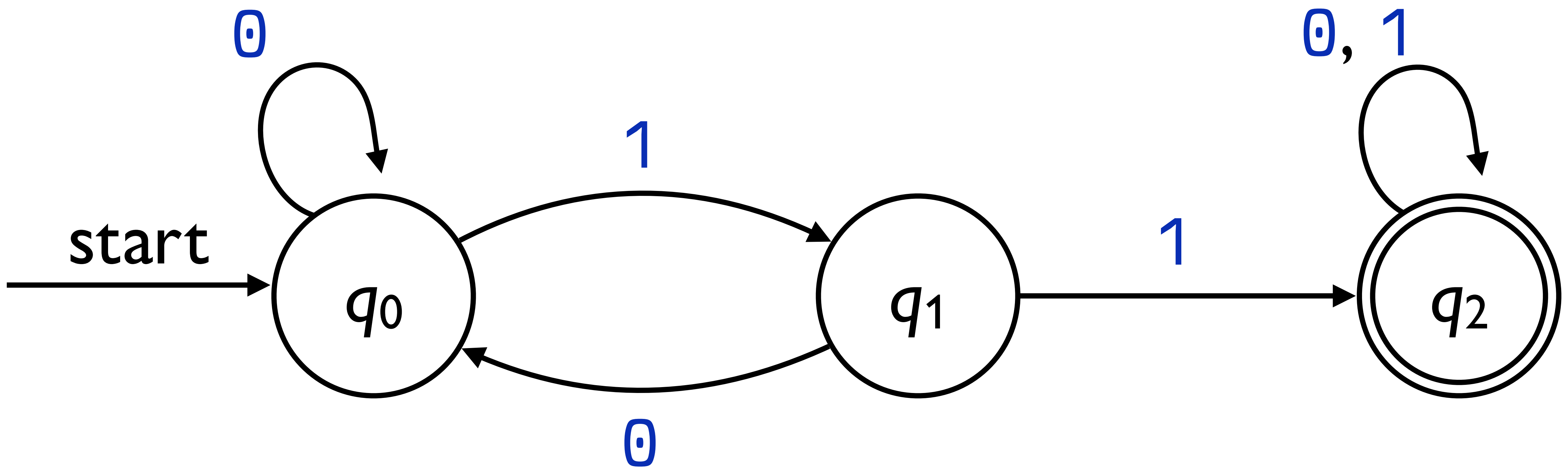


$L = \{w \in \{0, 1\}^* \mid w \text{ contains } 11 \text{ as a substring}\}$

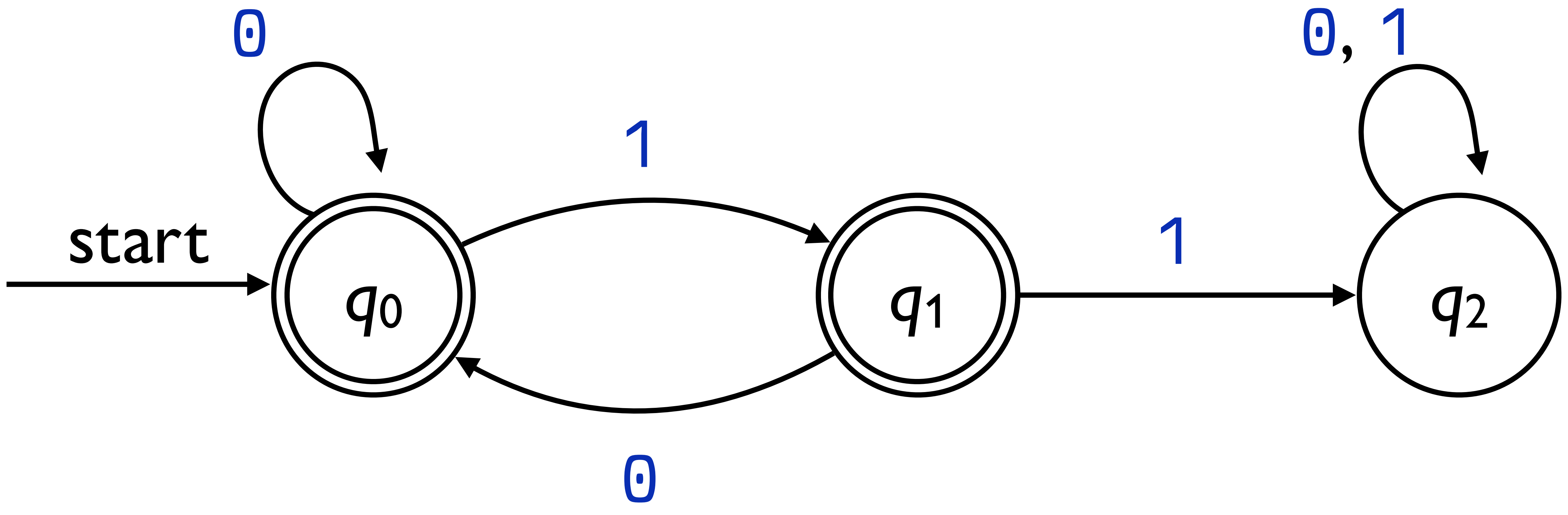


$\overline{L} = \{w \in \{0, 1\}^* \mid w$ *does not* contain 11 as a substring$\}$

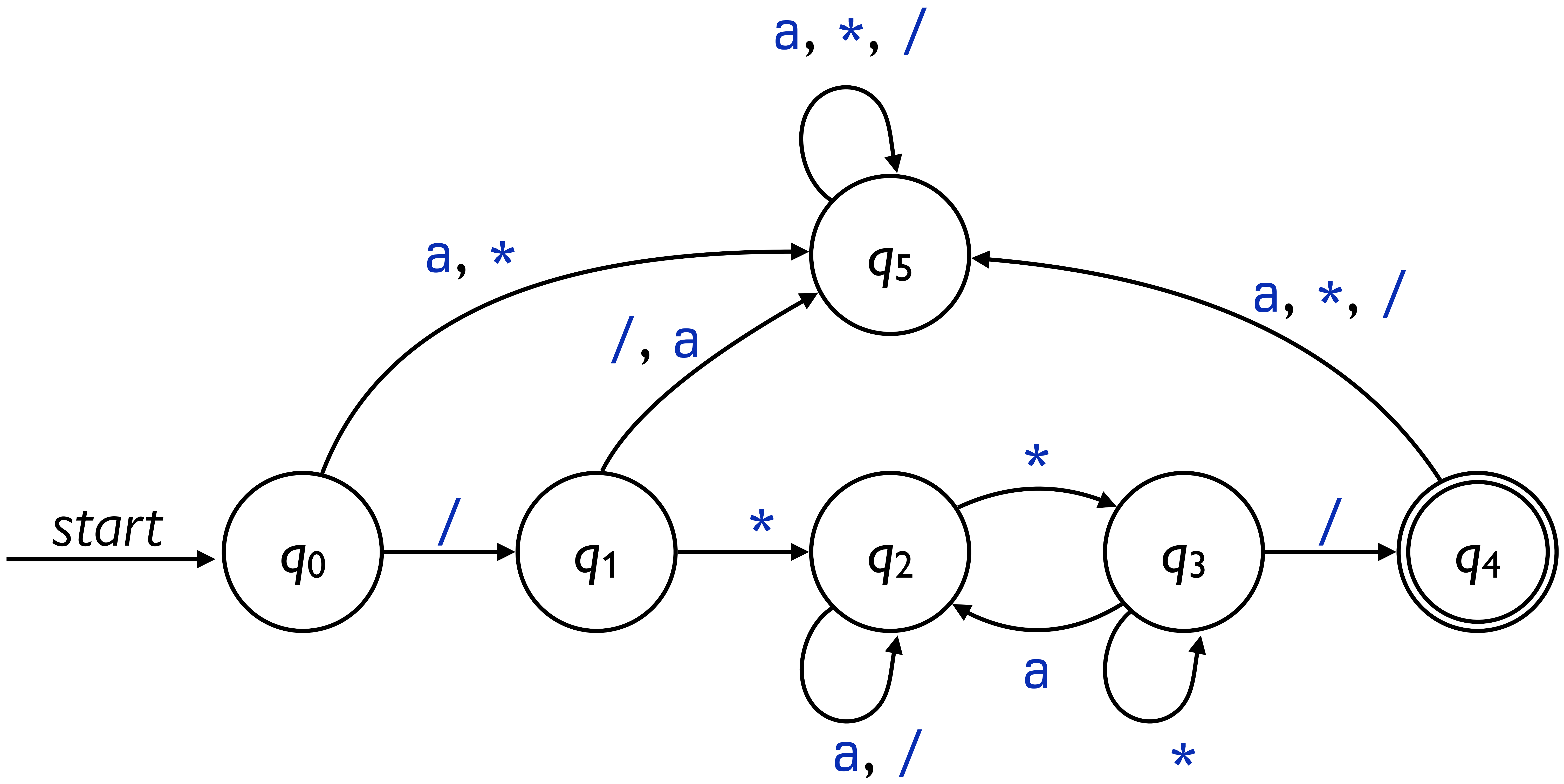
$L = \{w \in \{0, 1\}^* \mid w \text{ contains } 11 \text{ as a substring}\}$



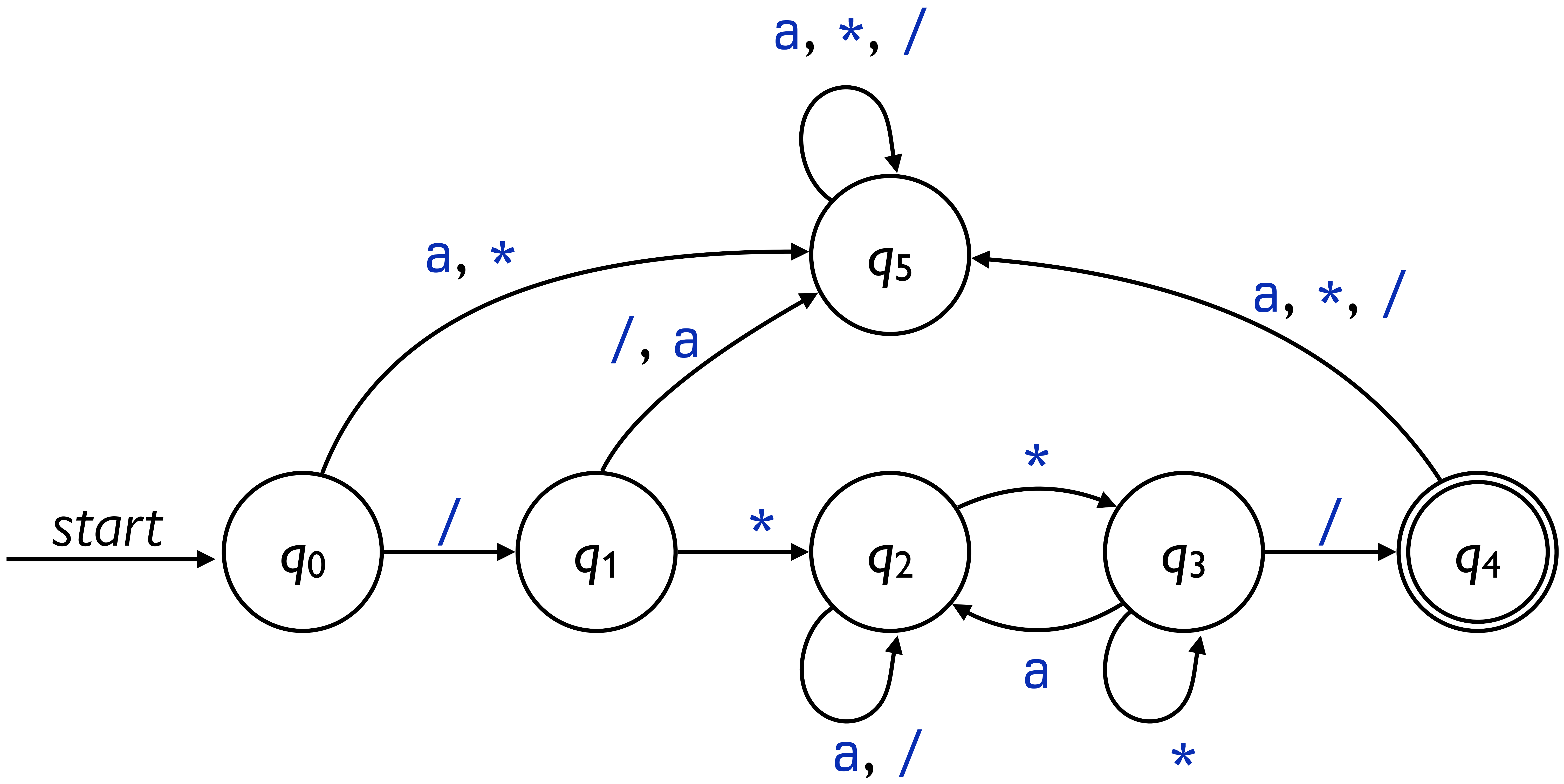
$\overline{L} = \{w \in \{0, 1\}^* \mid w \textit{ does not} \text{ contain } 11 \text{ as a substring}\}$



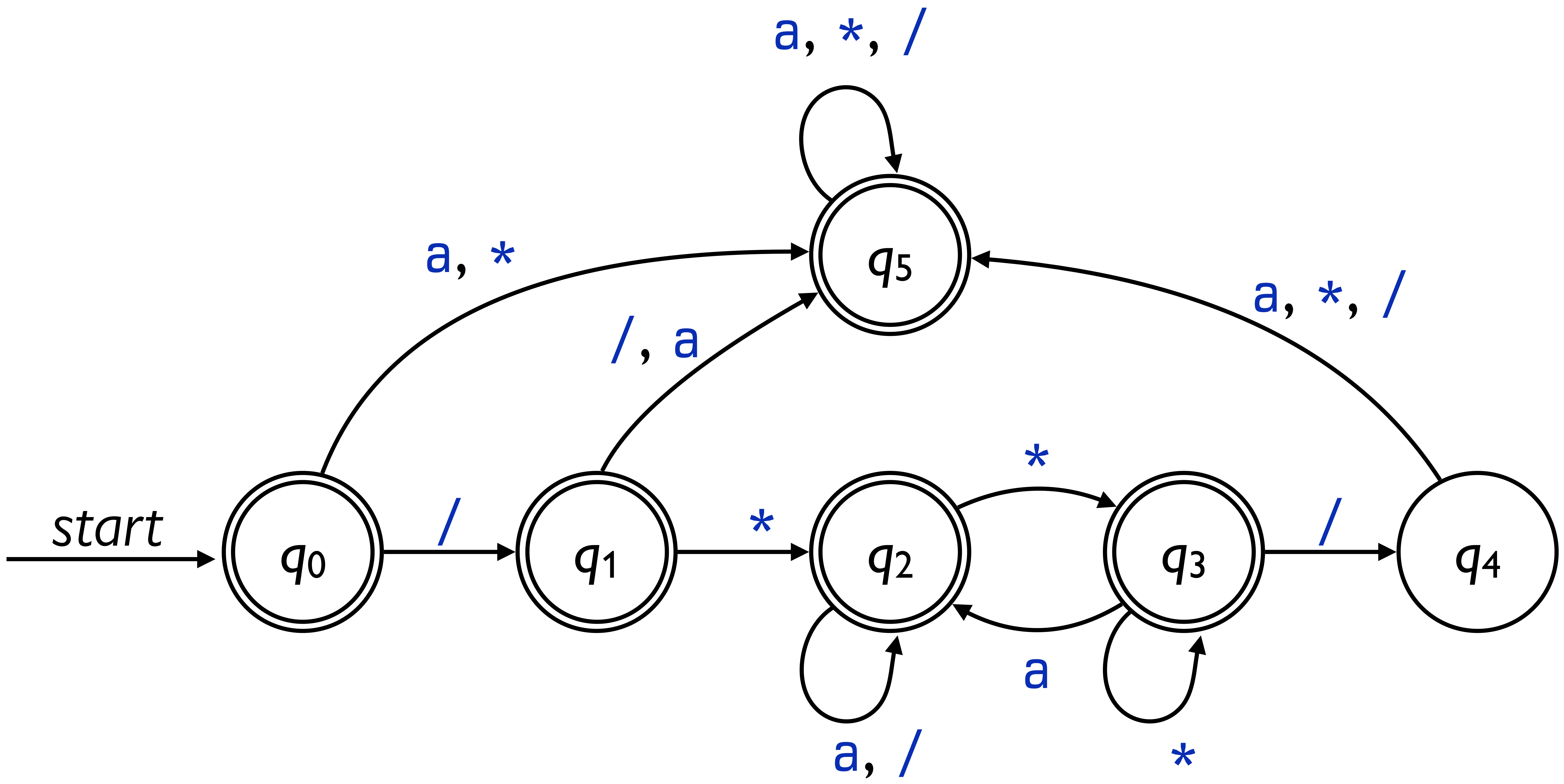
$L = \{w \in \{a, *, /\}^* \mid w \text{ represents a C-style comment}\}$



$\overline{L}$ = {$w \in$ {a, *, /}* | w ***doesn't*** represent a C-style comment}



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



THEOREM If L is a regular language, then $\overline{L}$ is also a regular language.

PROOF BY CONSTRUCTION Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognizing L, we can construct a DFA $M' = (Q, \Sigma, \delta, q_0, Q - F)$. Then M' recognizes $\overline{L}$; we omit a proof of correctness.

In proofs that involve constructions like this one, it's usually pretty clear that the construction works, so I'm usually okay with you omitting the proof of correctness.

THEOREM If L is a regular language, then $\overline{L}$ is also a regular language.

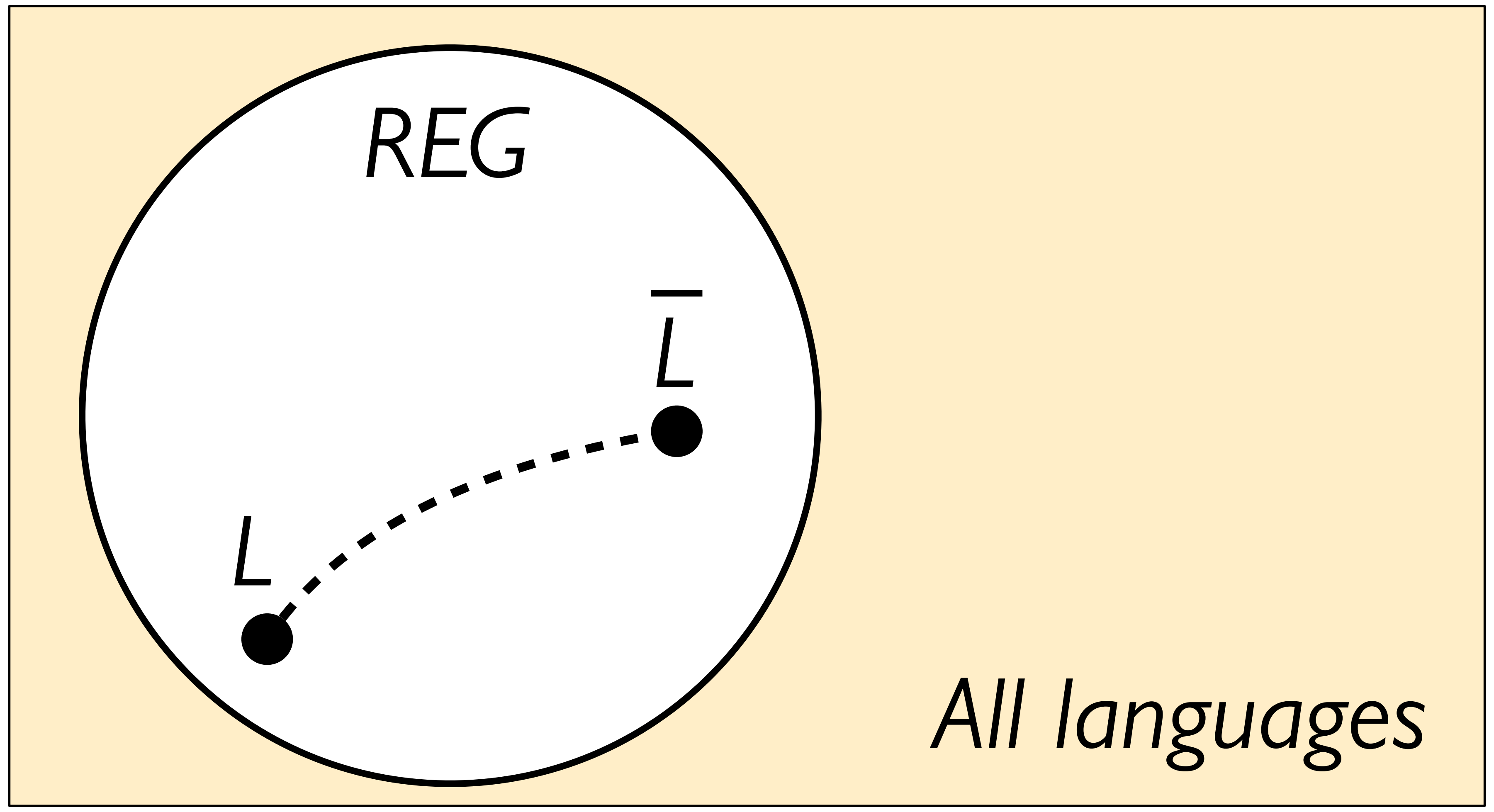
PROOF BY CONSTRUCTION Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognizing L, we can construct a DFA $M' = (Q, \Sigma, \delta, q_0, Q - F)$.

Then M' recognizes $\overline{L}$. For if $w \in L$, then M, after reading w, is in state $q \in F$, so M', after reading w, is also in state q, which is not in $Q - F$, so M' rejects w. On the other hand, if $w \notin L$, then M, after reading w, is in state $q \notin F$, so M' after reading w, is also in state q, which is in $Q - F$, so M' accepts w.

But if you wanted to add a proof of correctness, it would look like this.

THEOREM If L is a regular language, then $\overline{L}$ is also a regular language.

As a result, we say that the regular languages are *closed under complementation*.



Union

Union of two languages

If L_1 and L_2 are languages over the alphabet Σ, 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 $L_1 \cup L_2$?

THEOREM If L_1 and L_2 are regular, then $L_1 \cup L_2$ is regular.

PROOF BY CONSTRUCTION Given the automata

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizing L_1 and

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizing L_2,

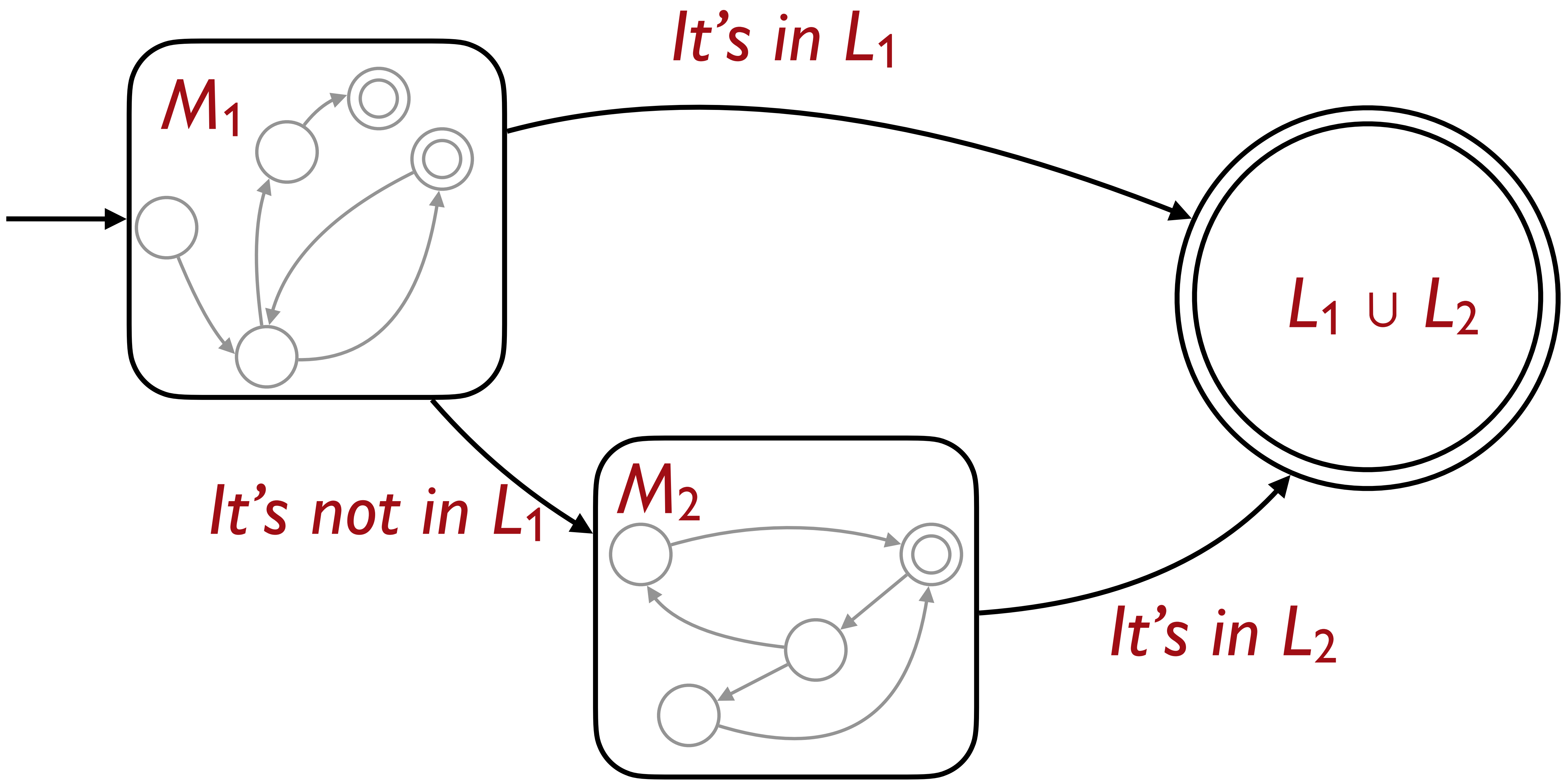
construct $M = (Q, \Sigma, \delta, q_0, F)$ recognizing $L_1 \cup L_2$.

For simplicity, let the alphabets be the same.

…

You might think this:

Run the string through M_1, and see whether M_1 accepts it. If not, run the string through M_2 and see whether M_2 accepts it.

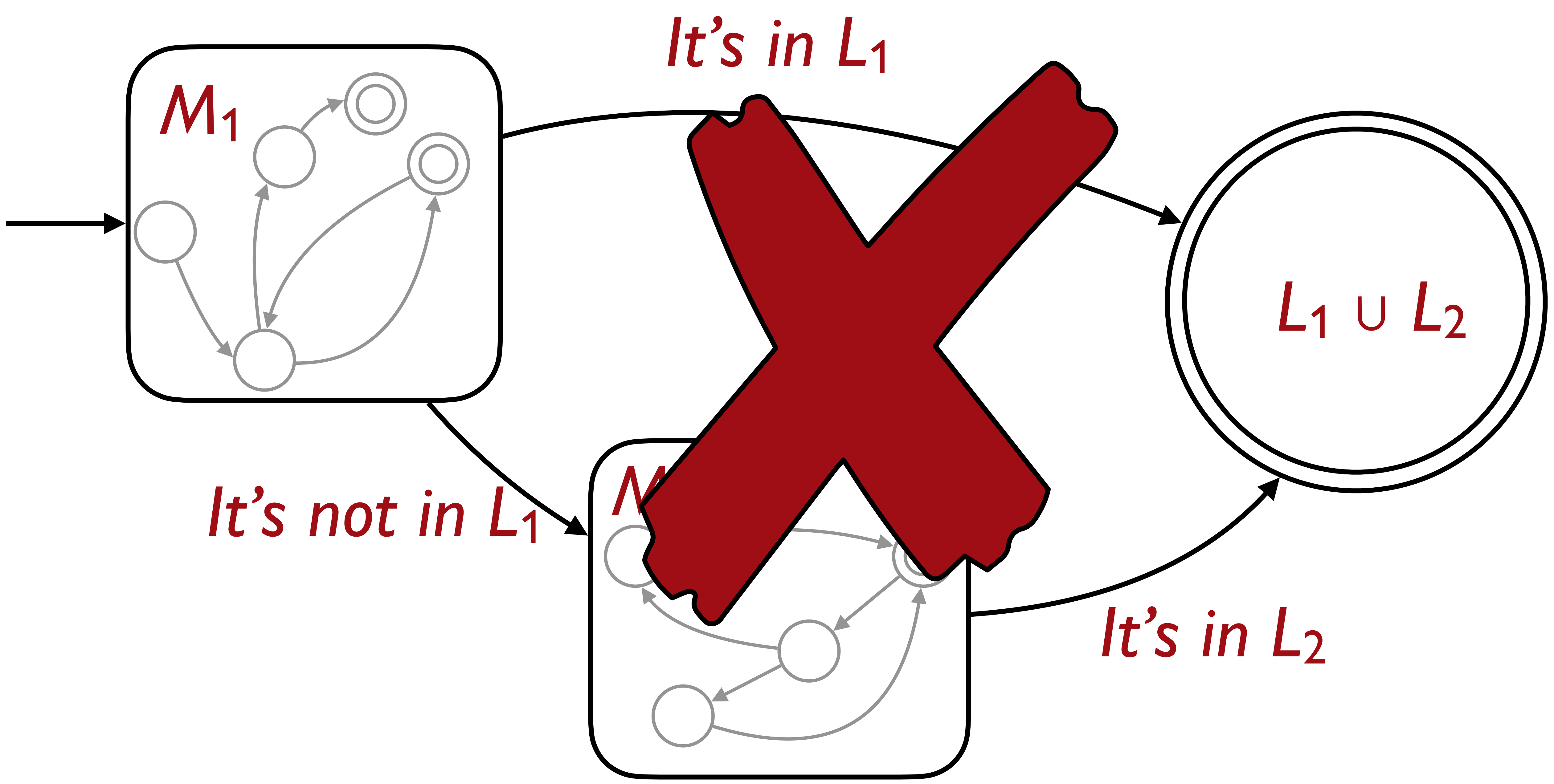


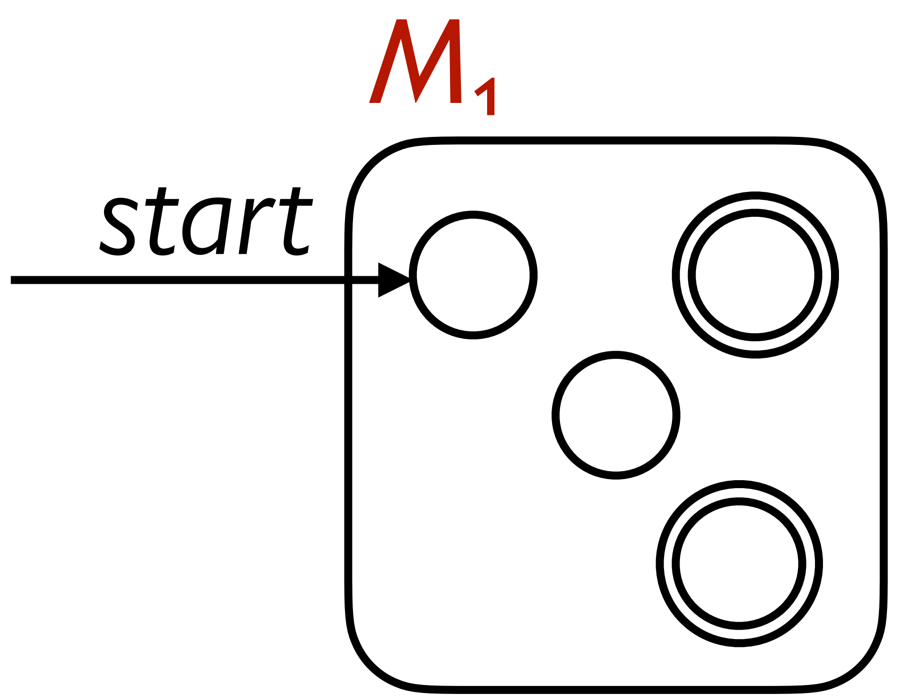
You might think this:

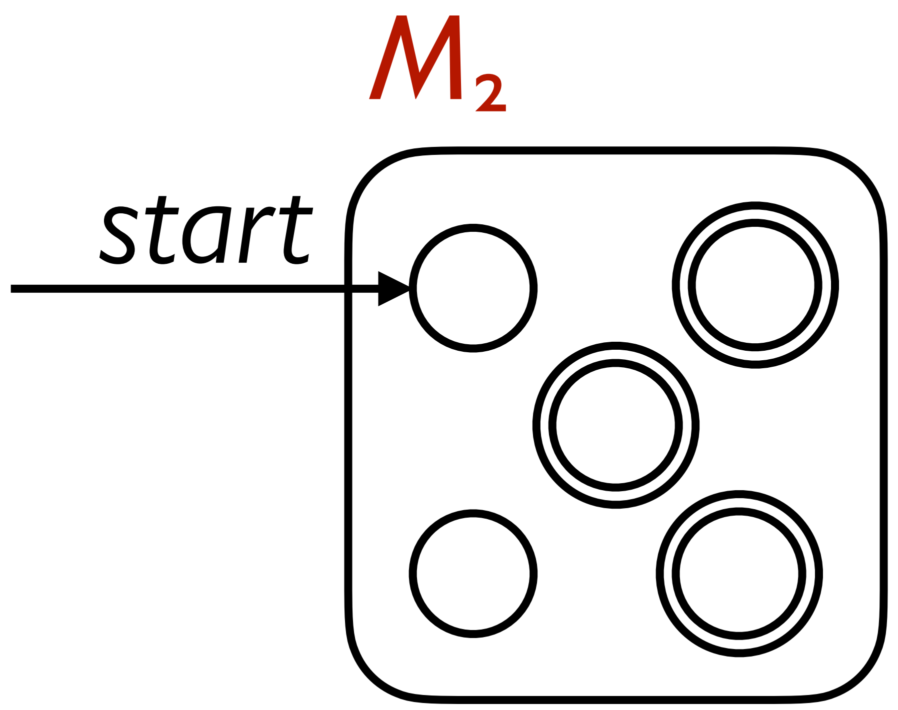
Run the string through M_1, and see whether M_1 accepts it. If not, run the string through M_2 and see whether M_2 accepts it.

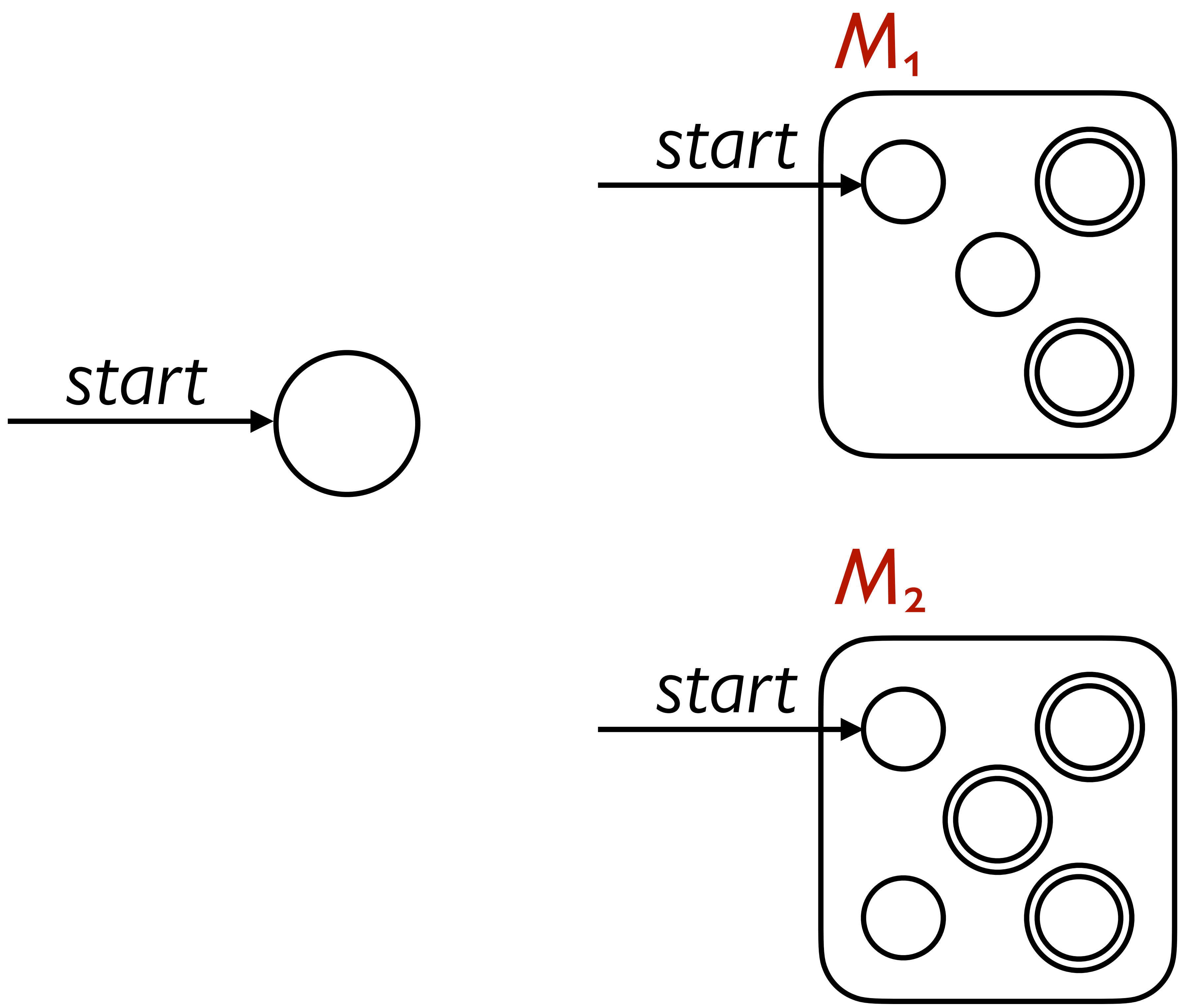
But you only get one pass!

A DFA / NFA can't try something on the whole input string, and then try another thing on the whole input string.

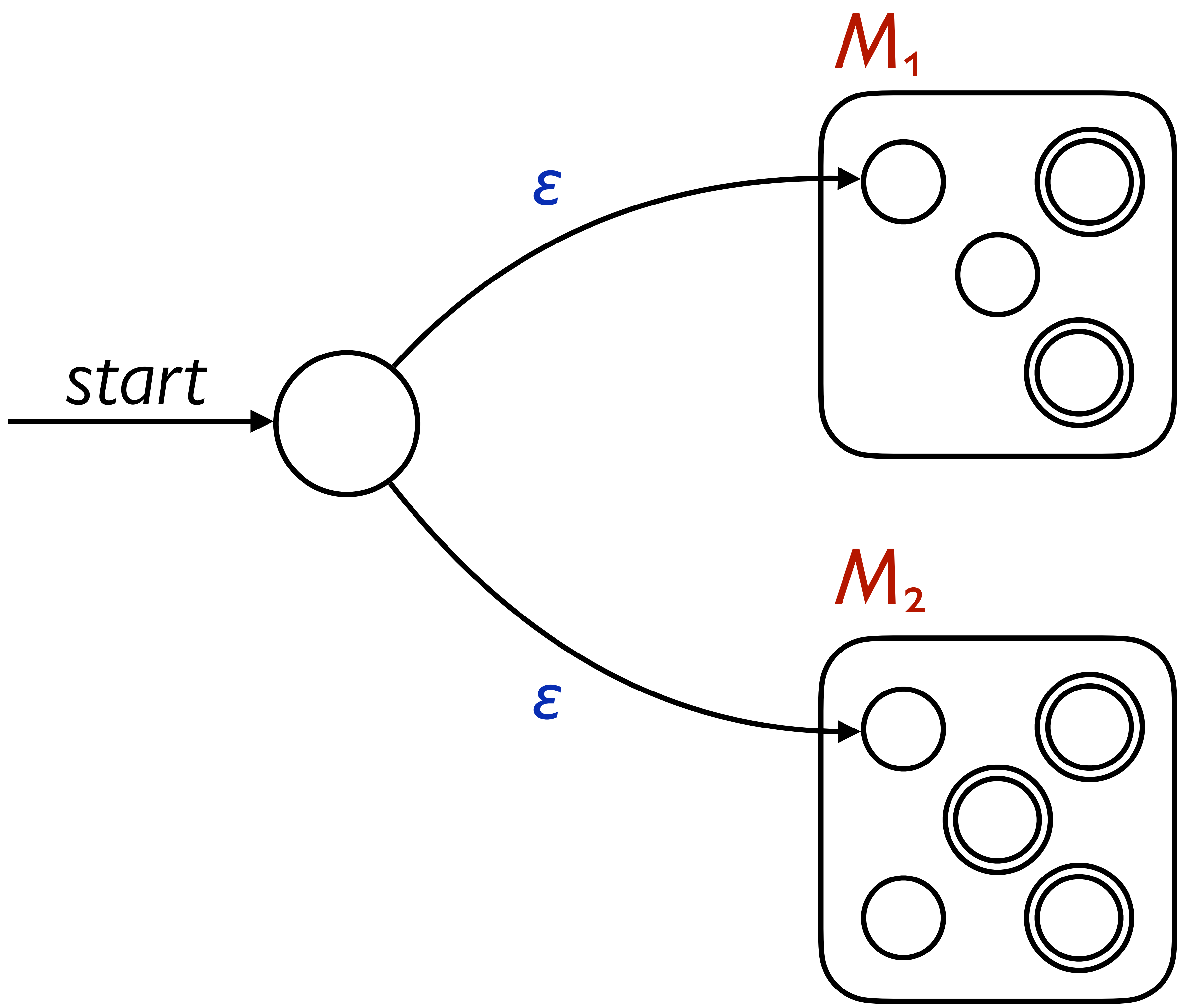



M1
start

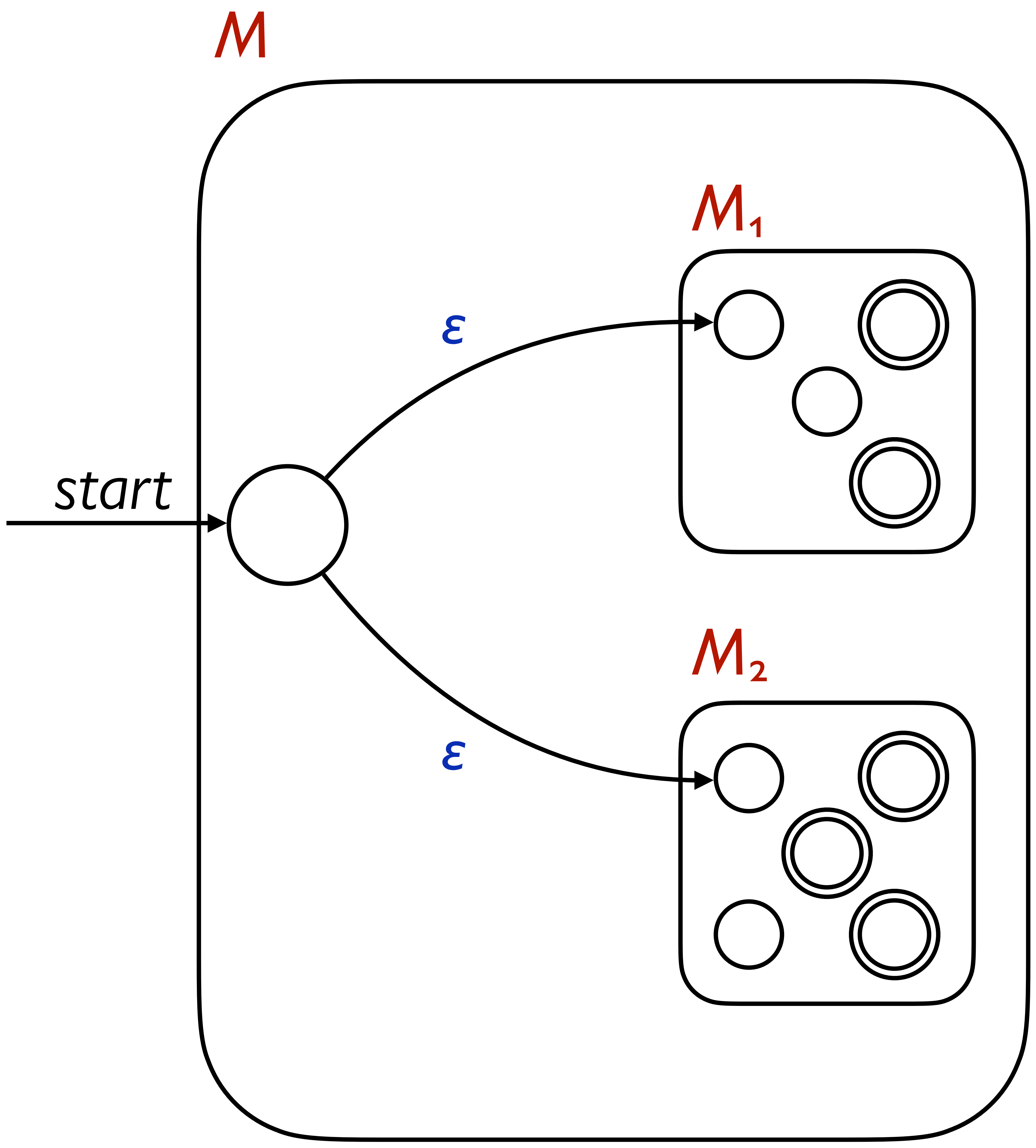

M2
start



start
M1
start
M2
start



The new machine guesses non-deterministically which of the two machines accepts the input



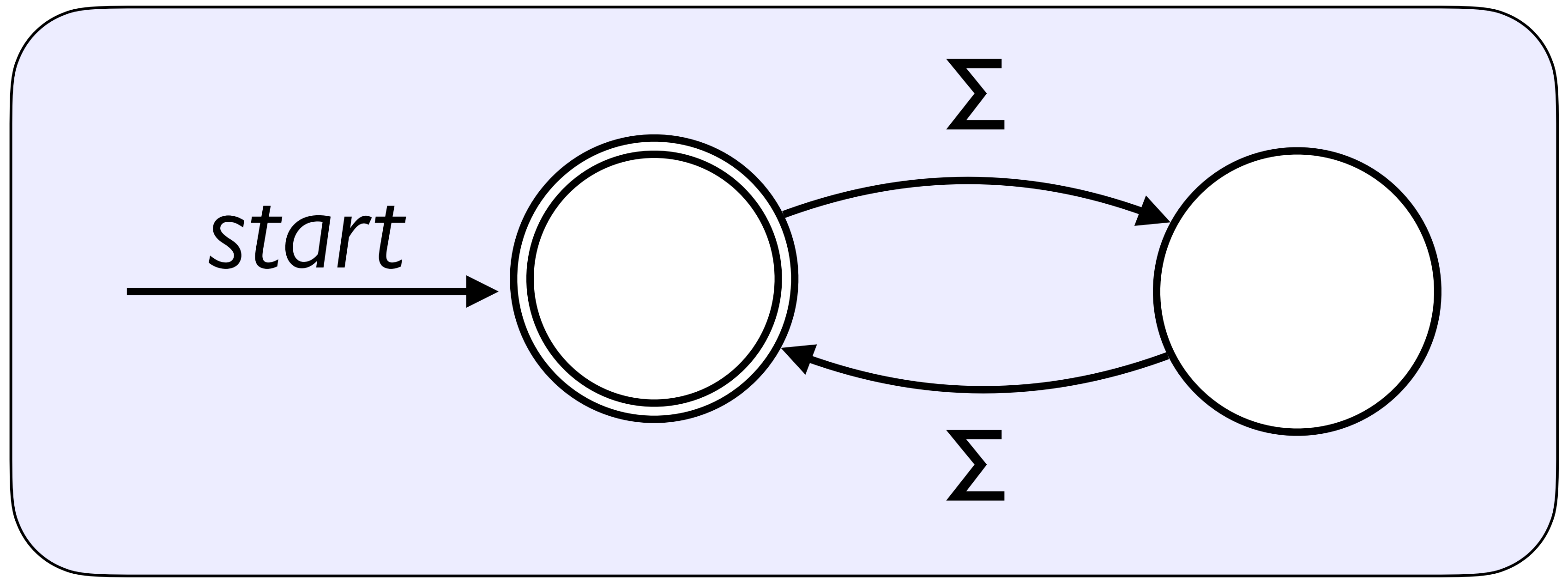
$$L(M) = L_1 \cup L_2$$

This construction proves the class of regular languages is closed under the union operation.

Example

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

Construct an NFA for $L_1 \cup L_2$.

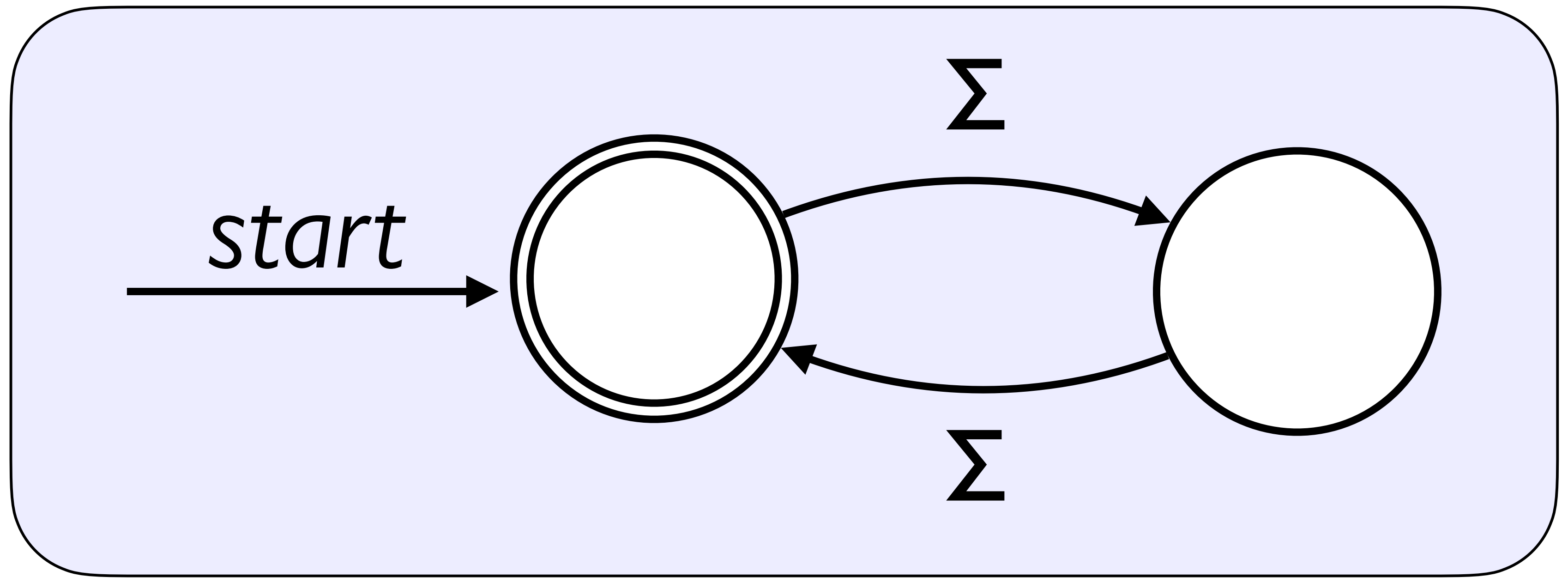


DFA for L_1

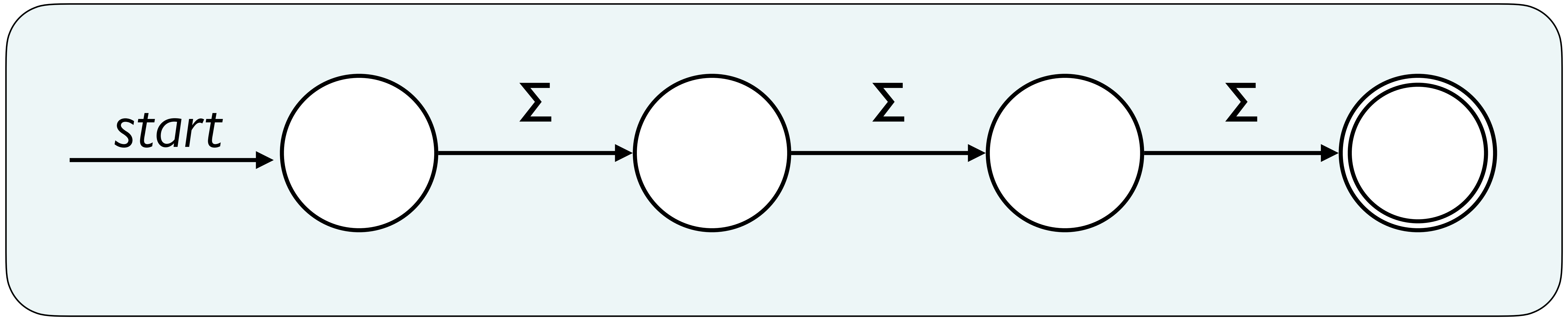
$L_1 = \{w \in \{a, b\}^* \mid w \text{ has even length}\}$

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

Construct an NFA for $L_1 \cup L_2$.



DFA for L_1

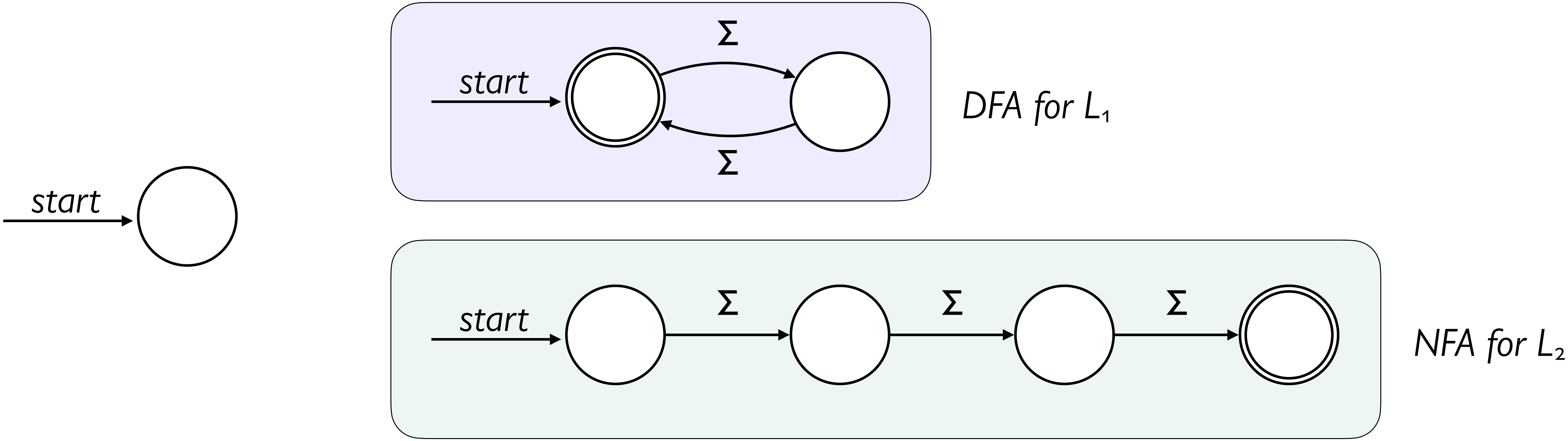


NFA for L_2

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

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

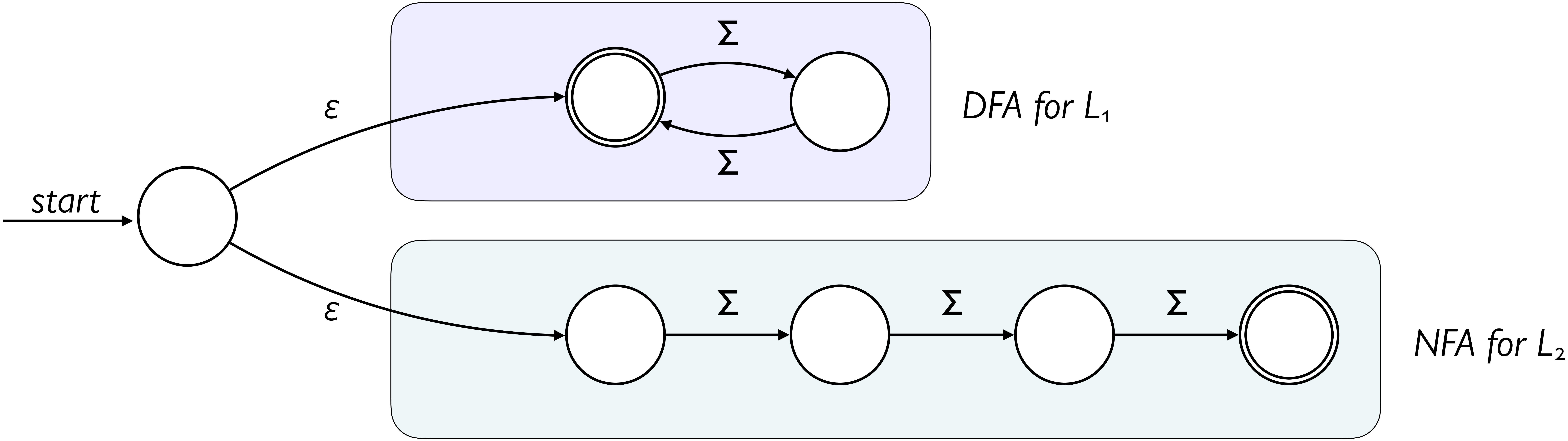
Construct an NFA for $L_1 \cup L_2$.



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

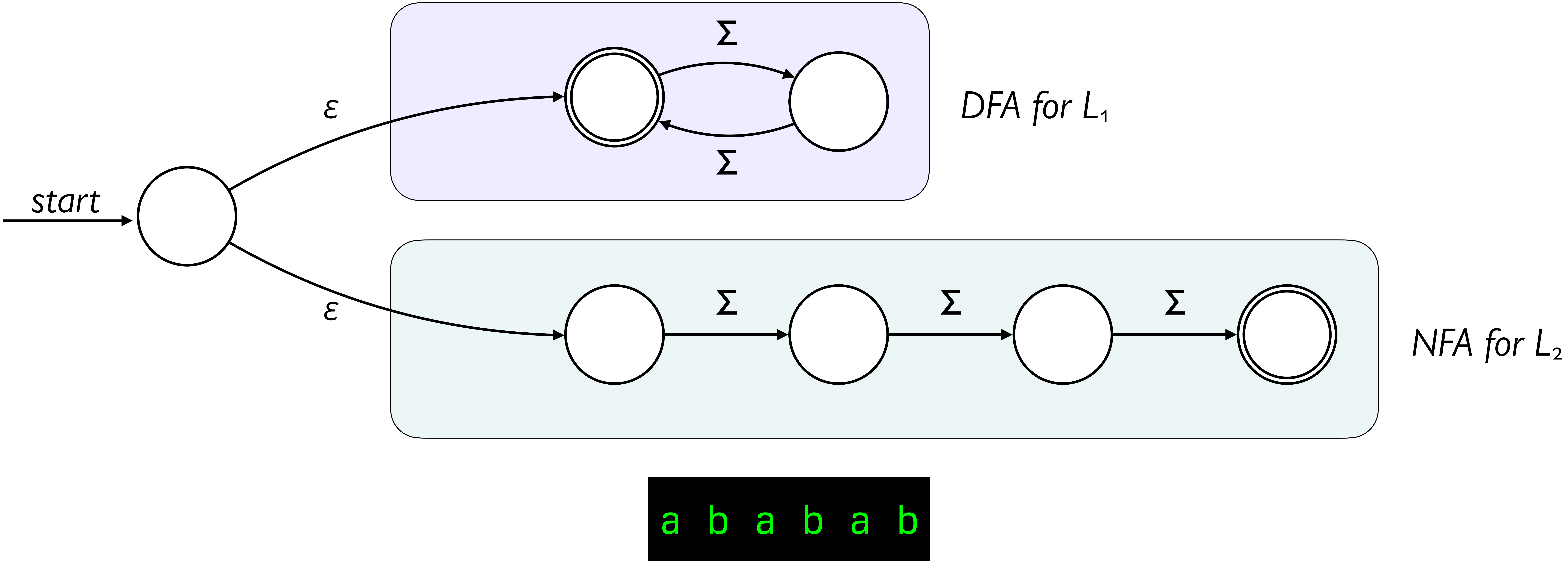
$L_2 = \{w \in \{a, b\}^* \mid w \text{ has length exactly three}\}$

Construct an NFA for $L_1 \cup L_2$.



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

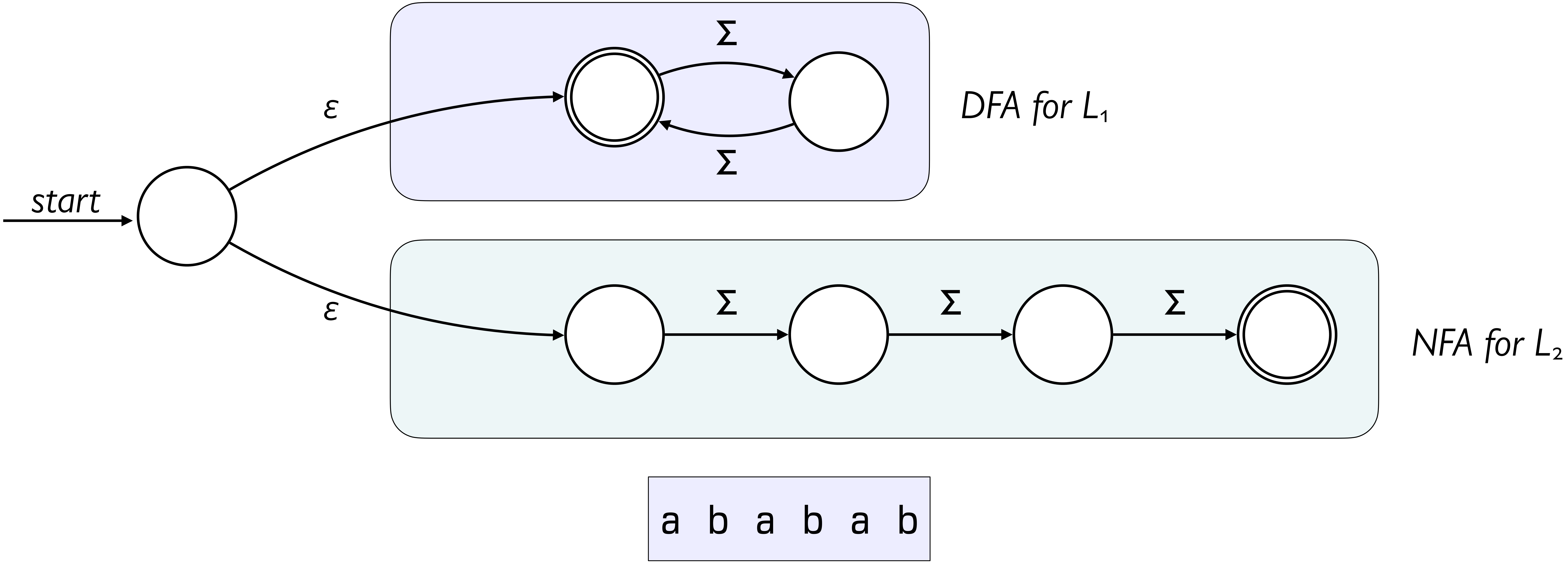
Construct an NFA for $L_1 \cup L_2$.



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

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

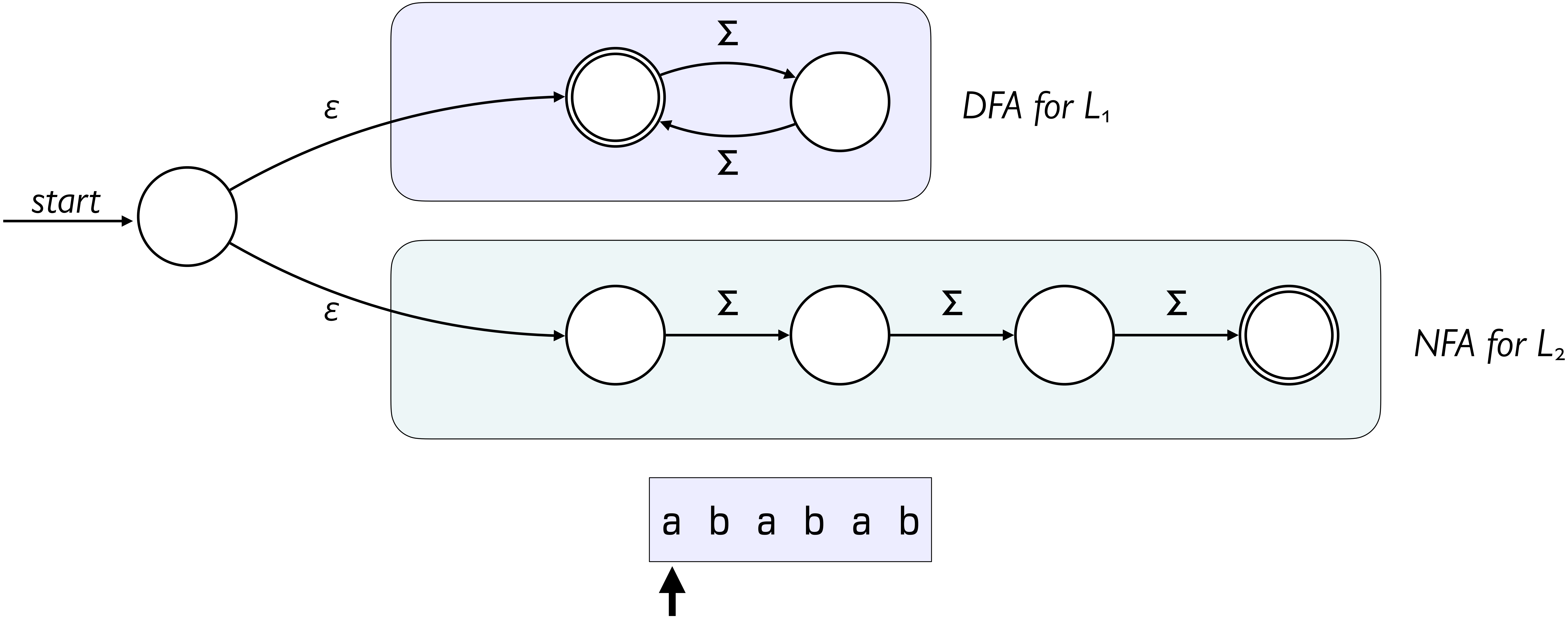
Construct an NFA for $L_1 \cup L_2$.



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

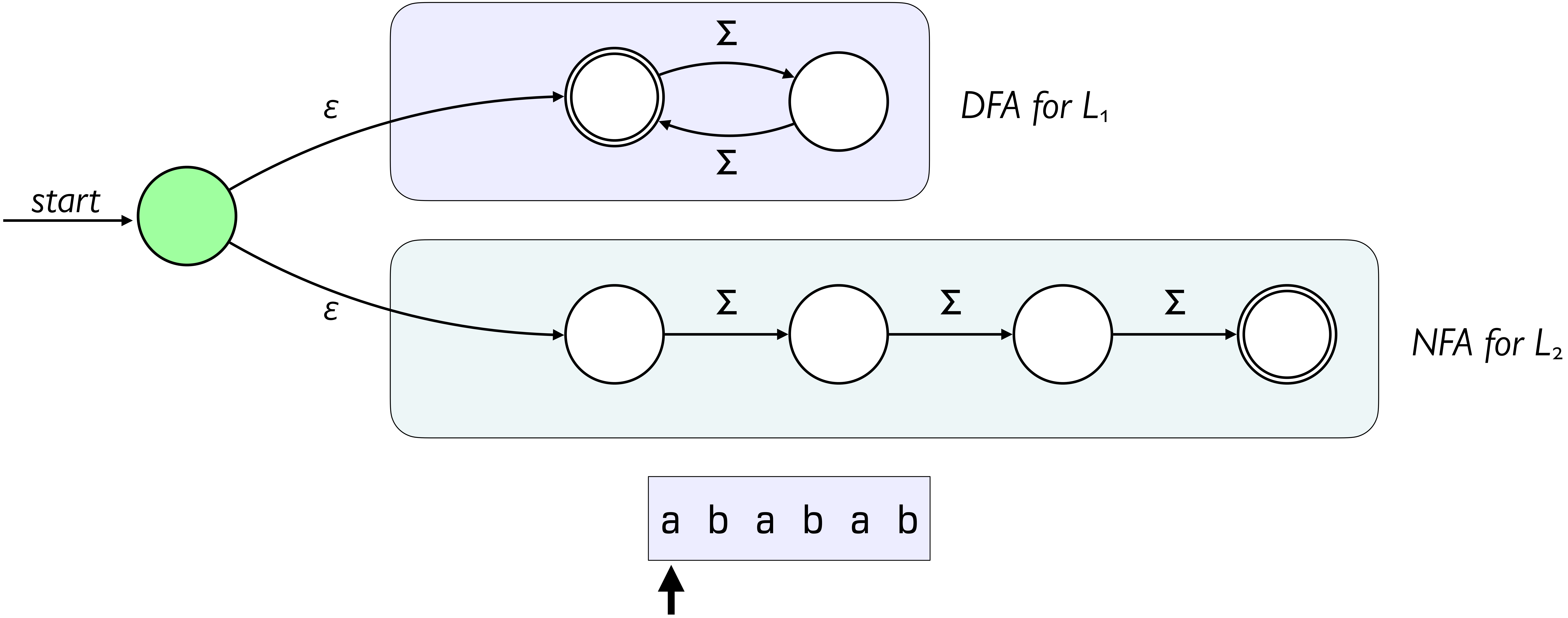
$L_2 = \{w \in \{a, b\}^* \mid w \text{ has length exactly three}\}$

Construct an NFA for $L_1 \cup L_2$.



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

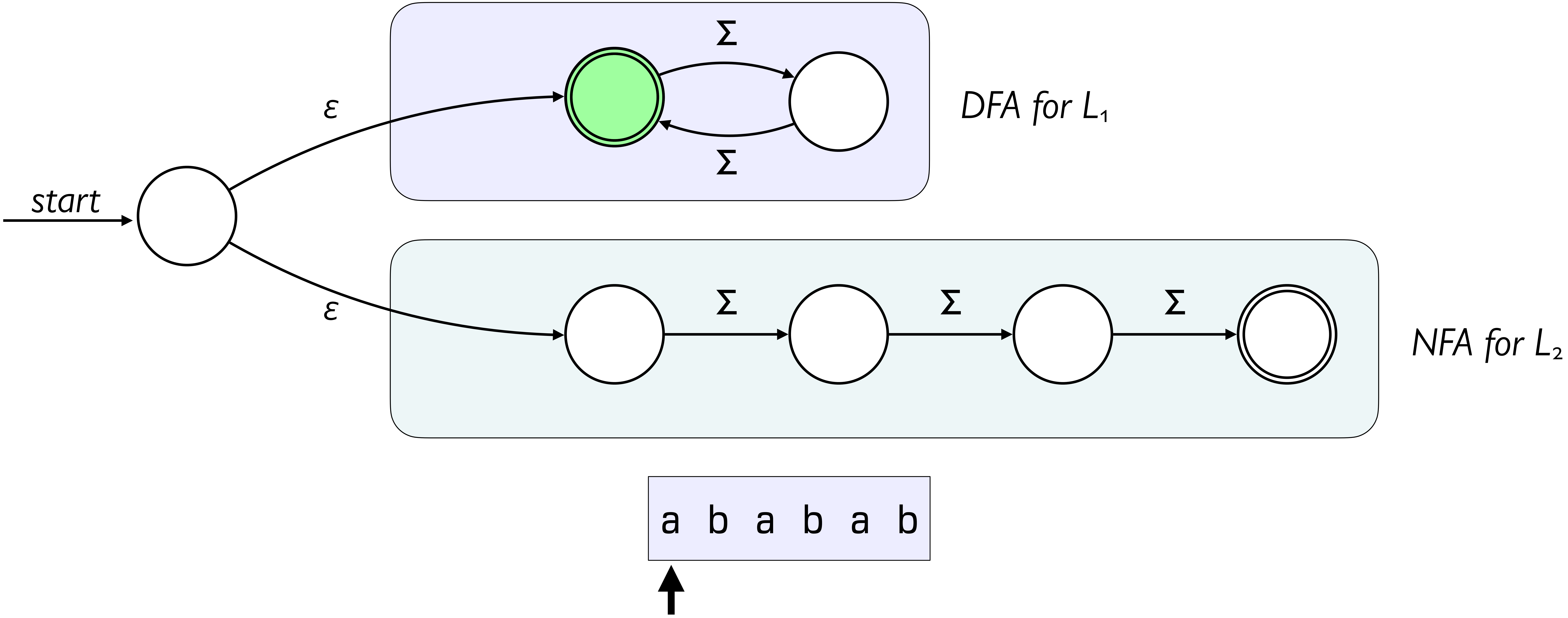
Construct an NFA for $L_1 \cup L_2$.



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

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

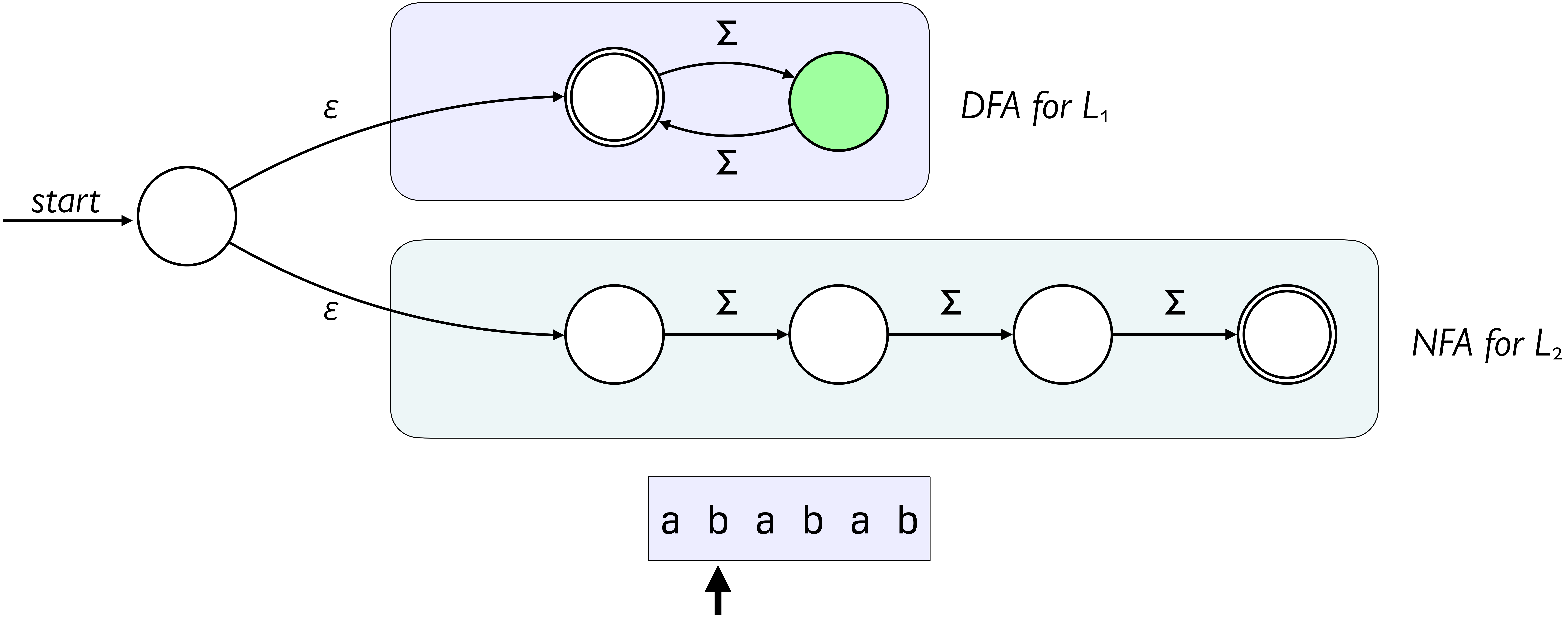
Construct an NFA for $L_1 \cup L_2$.



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

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

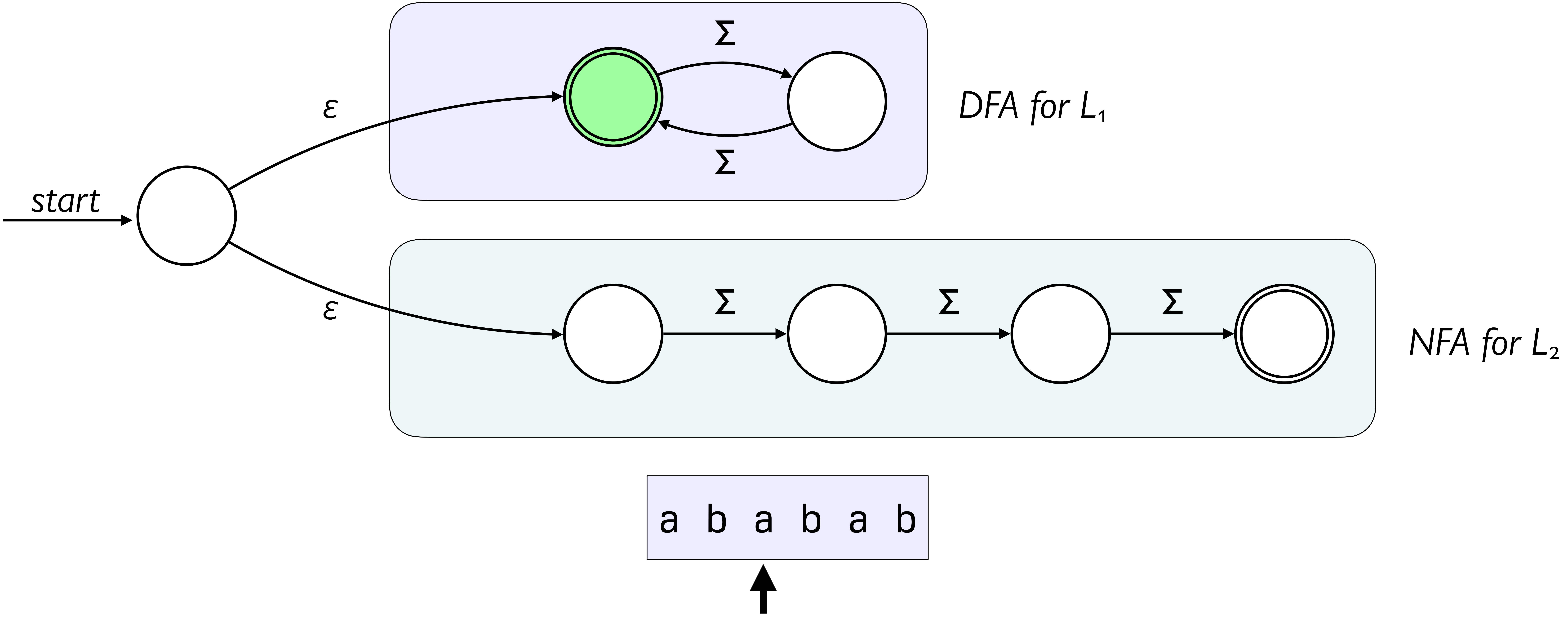
Construct an NFA for $L_1 \cup L_2$.



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

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

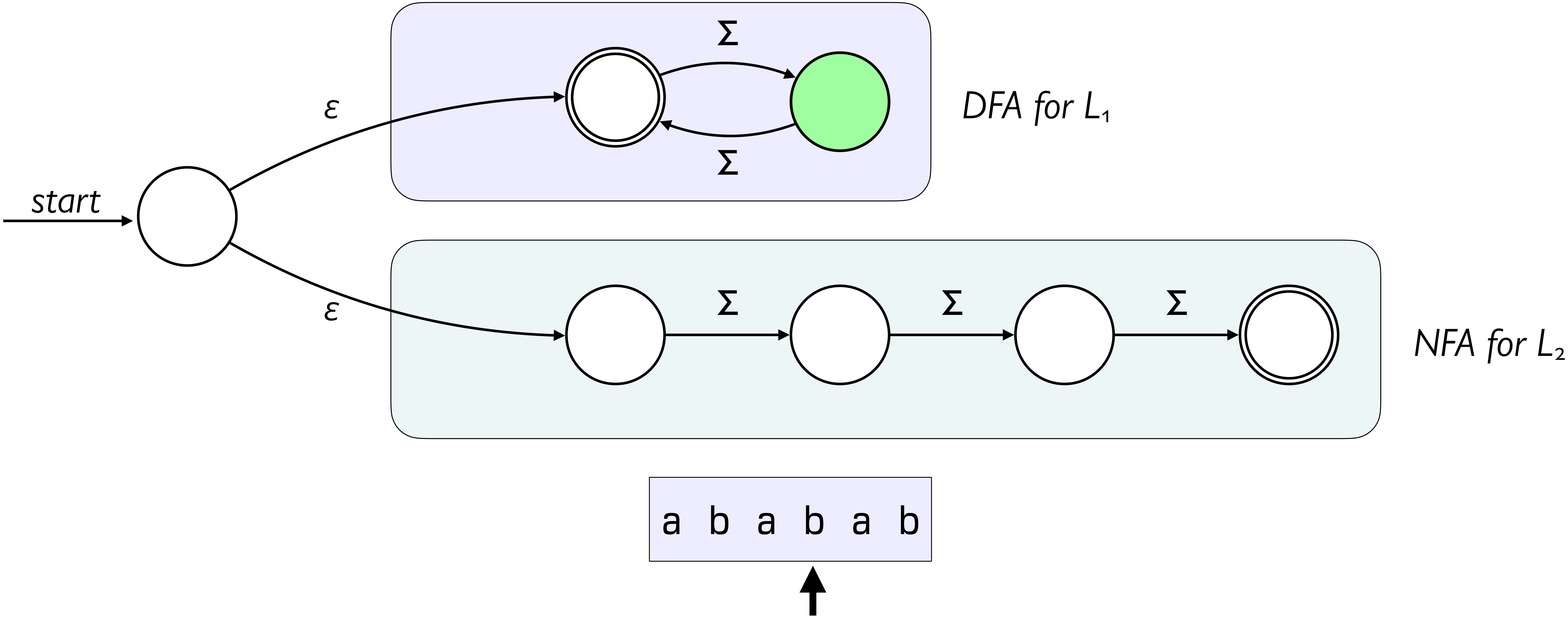
Construct an NFA for $L_1 \cup L_2$.



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

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

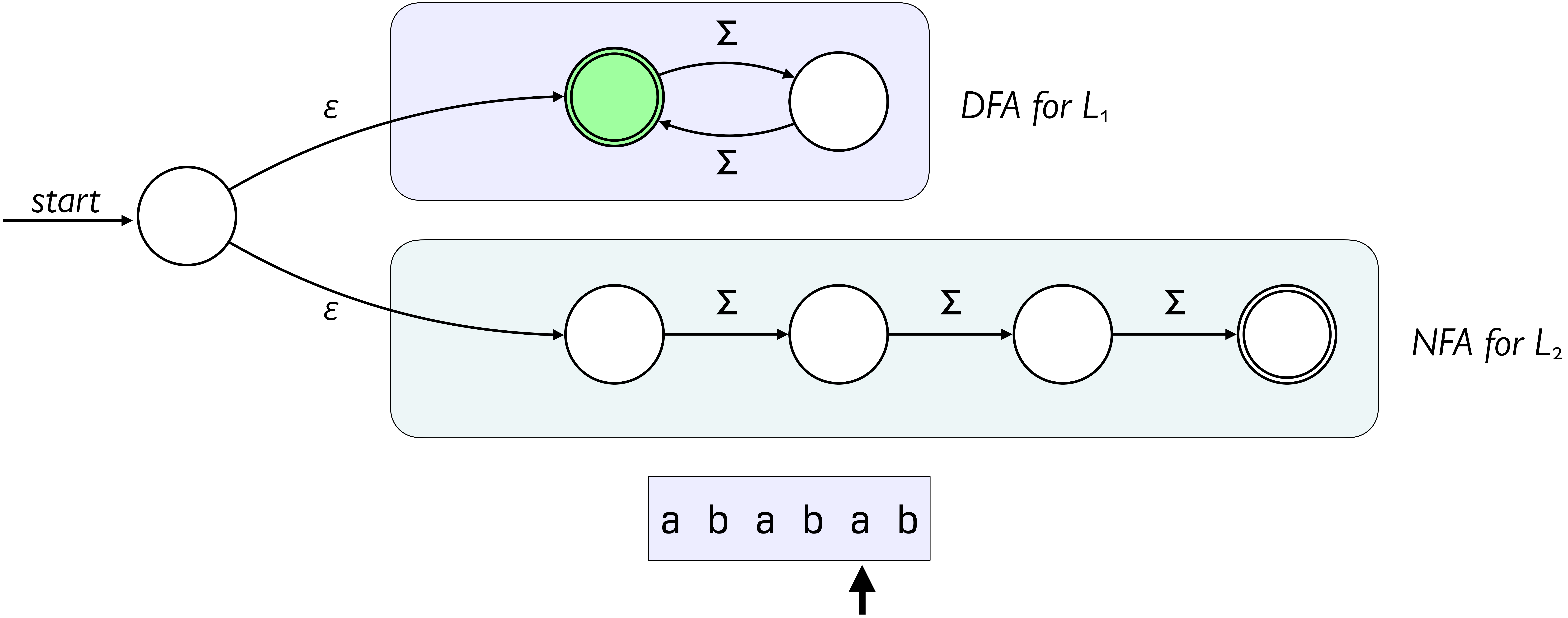
Construct an NFA for $L_1 \cup L_2$.



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

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

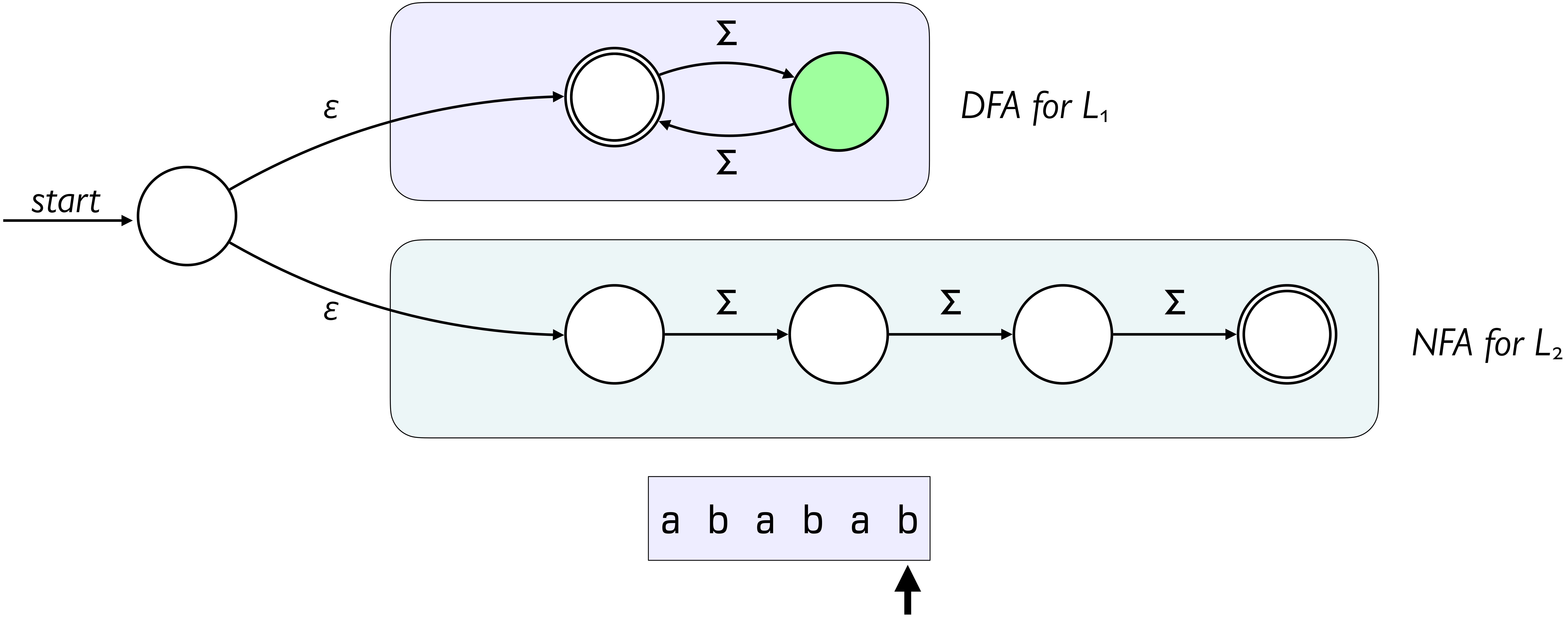
Construct an NFA for $L_1 \cup L_2$.



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

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

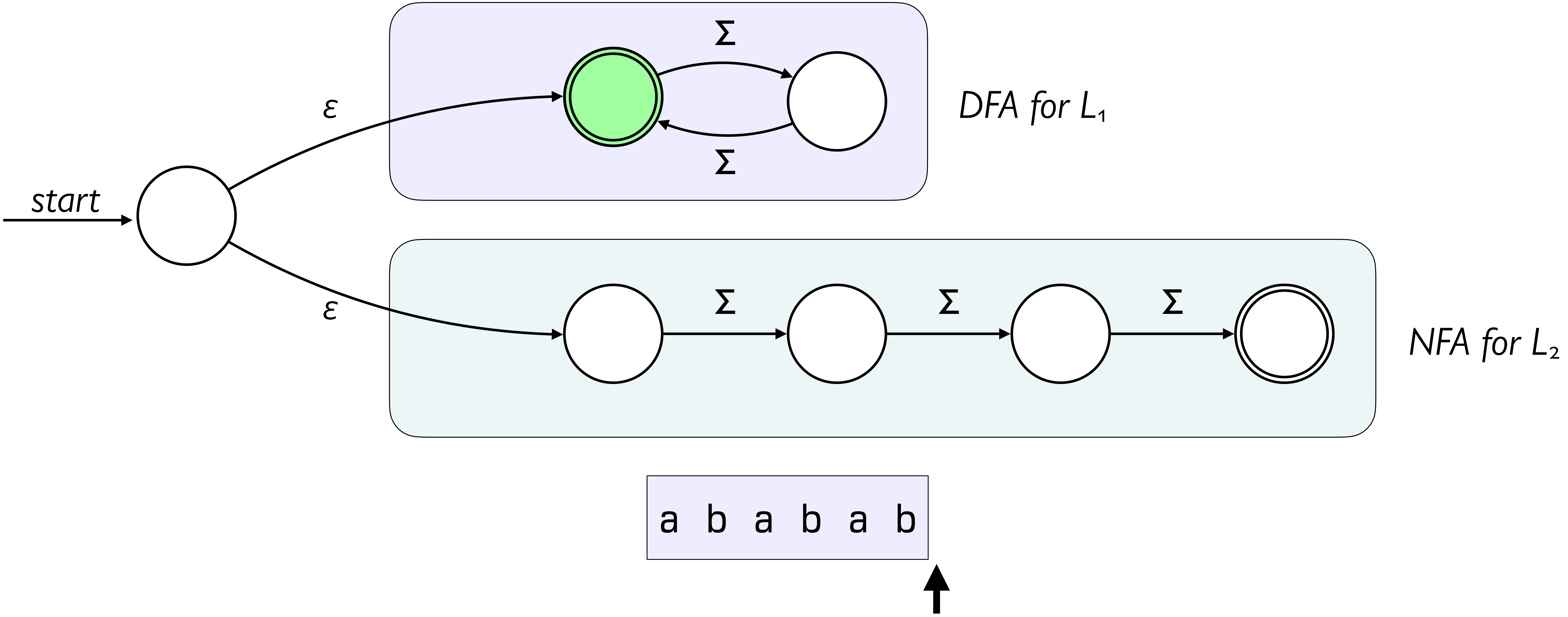
Construct an NFA for $L_1 \cup L_2$.



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

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

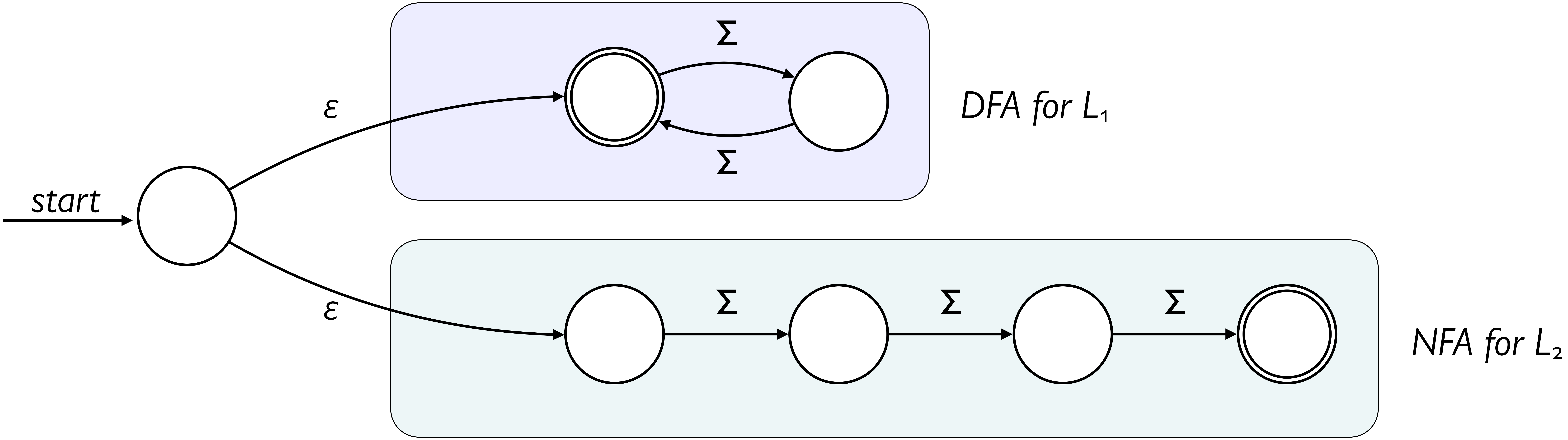
Construct an NFA for $L_1 \cup L_2$.



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

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

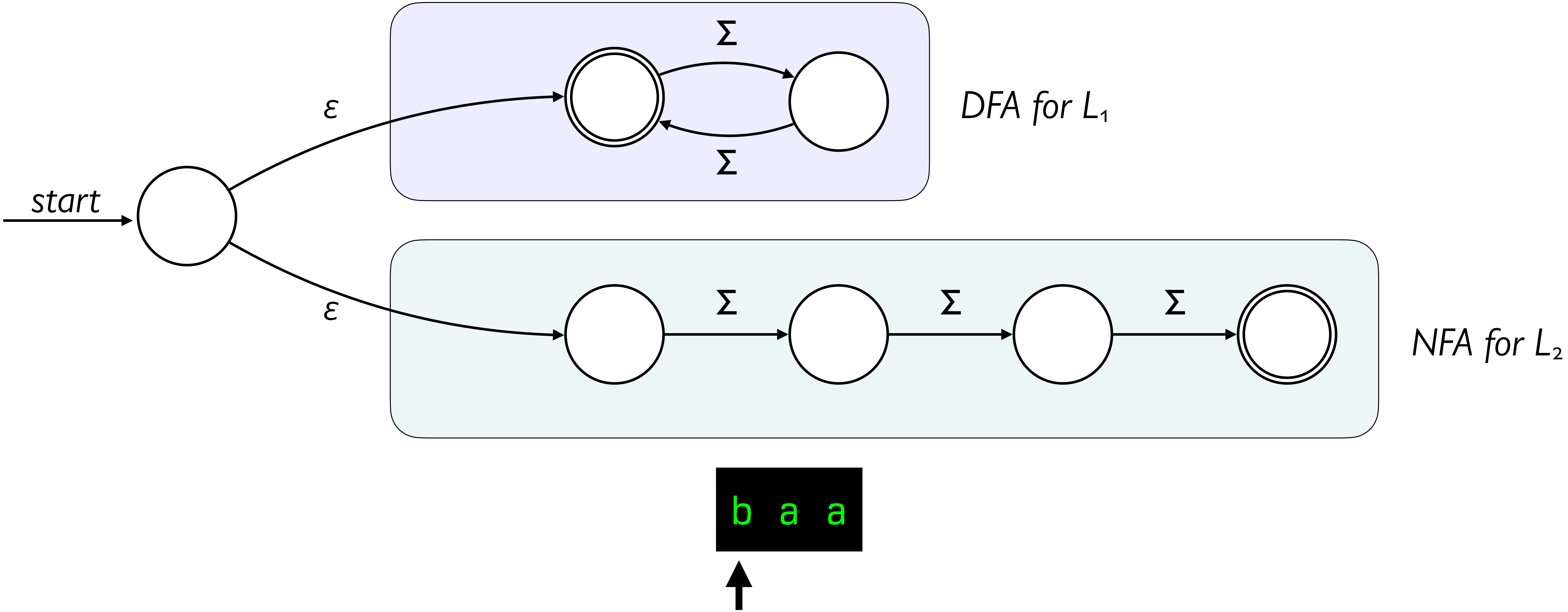
Construct an NFA for $L_1 \cup L_2$.



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

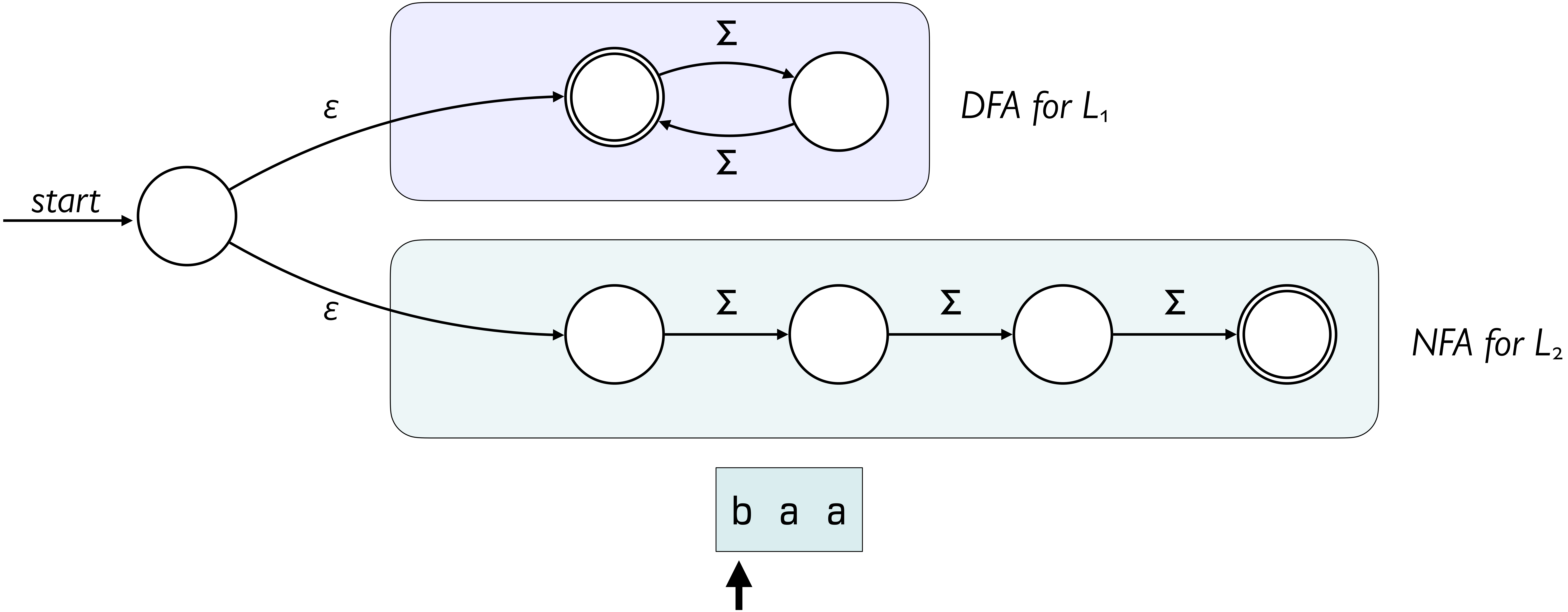
$L_2 = \{w \in \{a, b\}^* \mid w \text{ has length exactly three}\}$

Construct an NFA for $L_1 \cup L_2$.



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

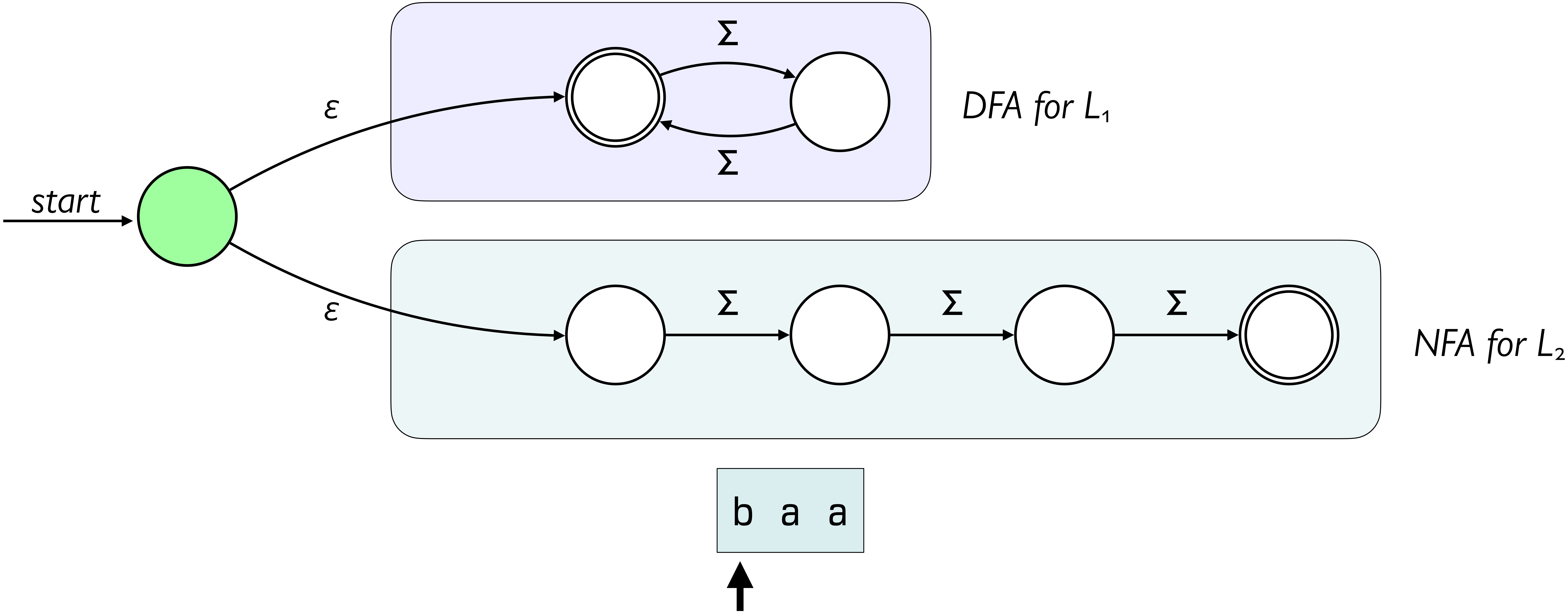
Construct an NFA for $L_1 \cup L_2$.



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

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

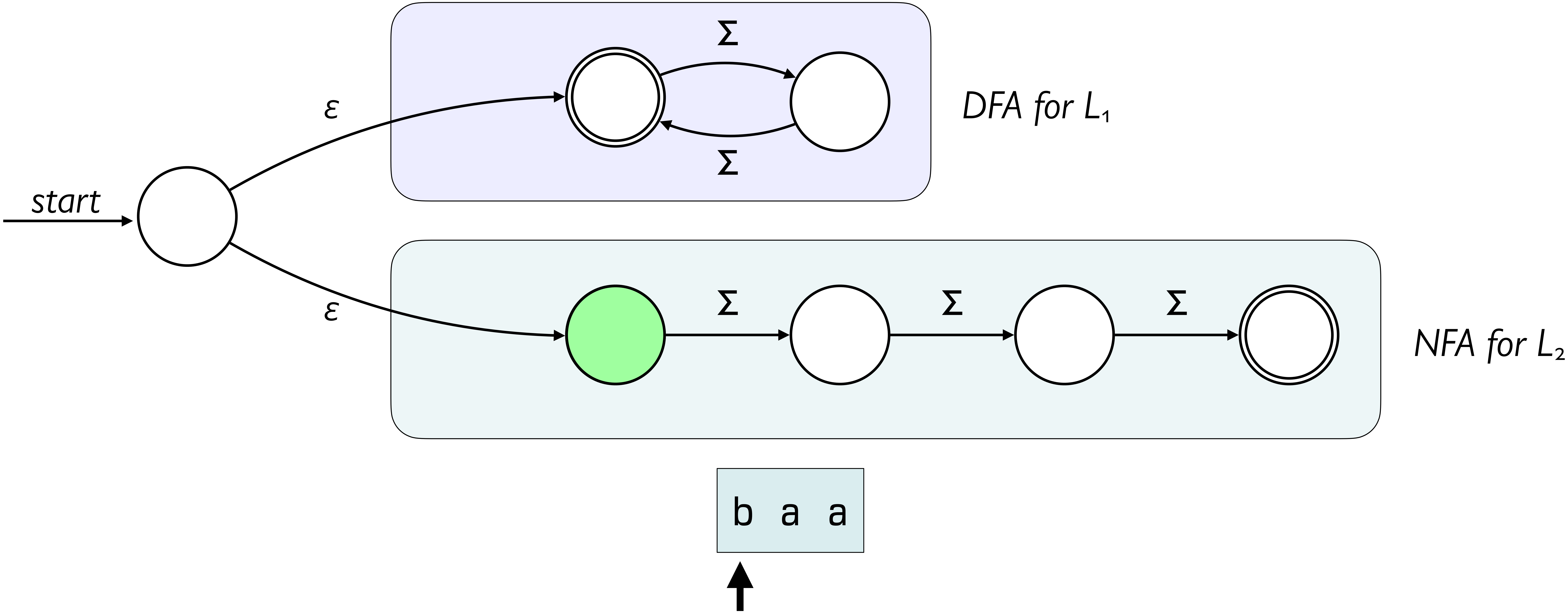
Construct an NFA for $L_1 \cup L_2$.



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

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

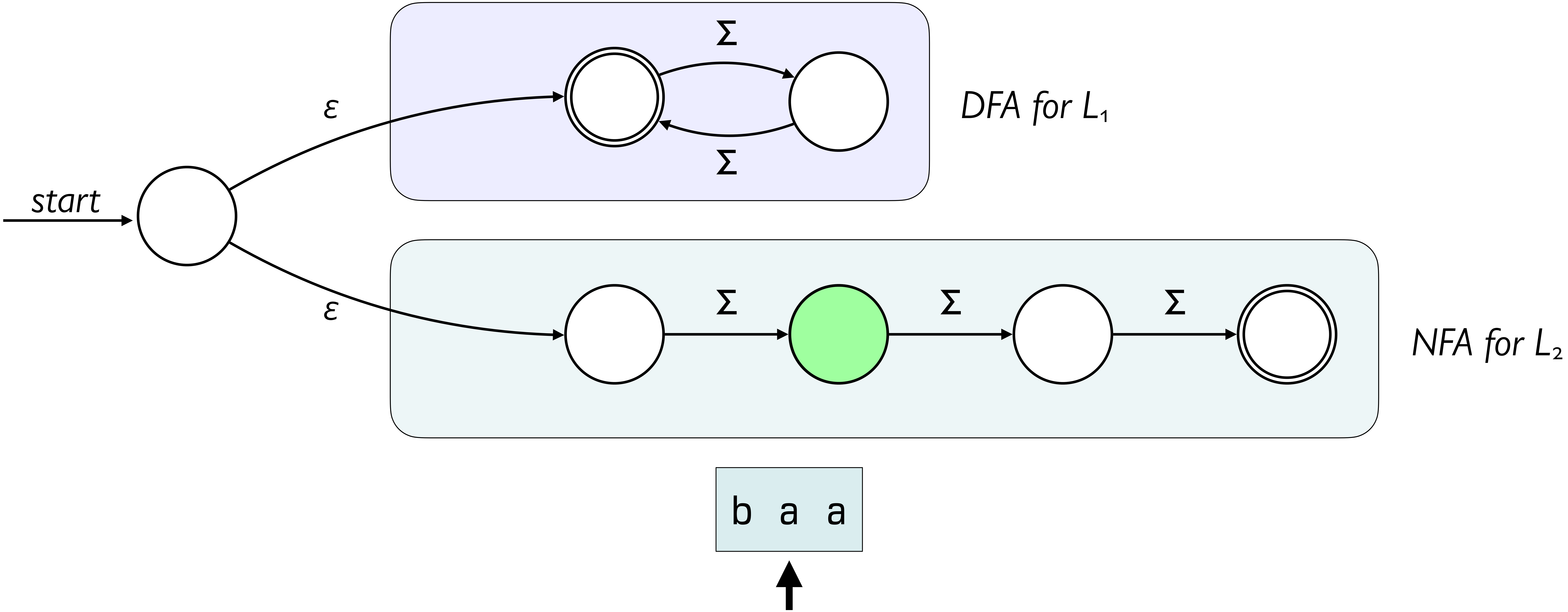
Construct an NFA for $L_1 \cup L_2$.



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

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

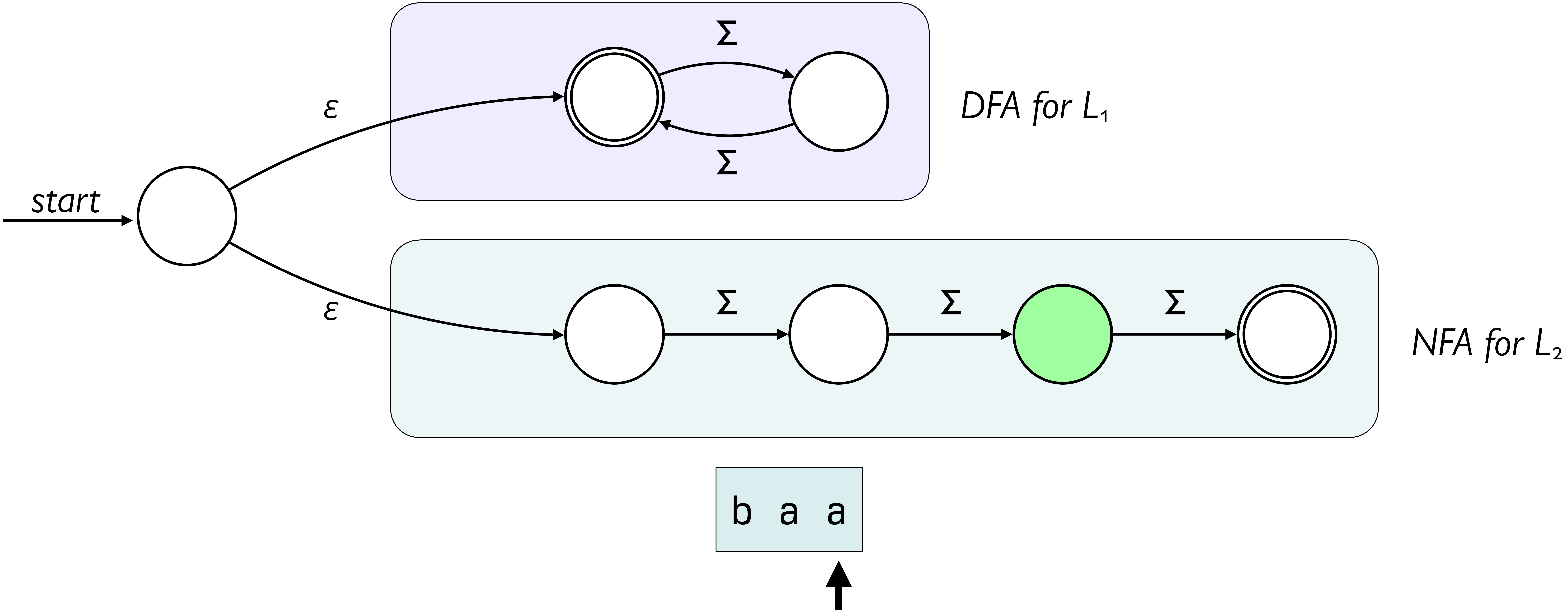
Construct an NFA for $L_1 \cup L_2$.



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

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

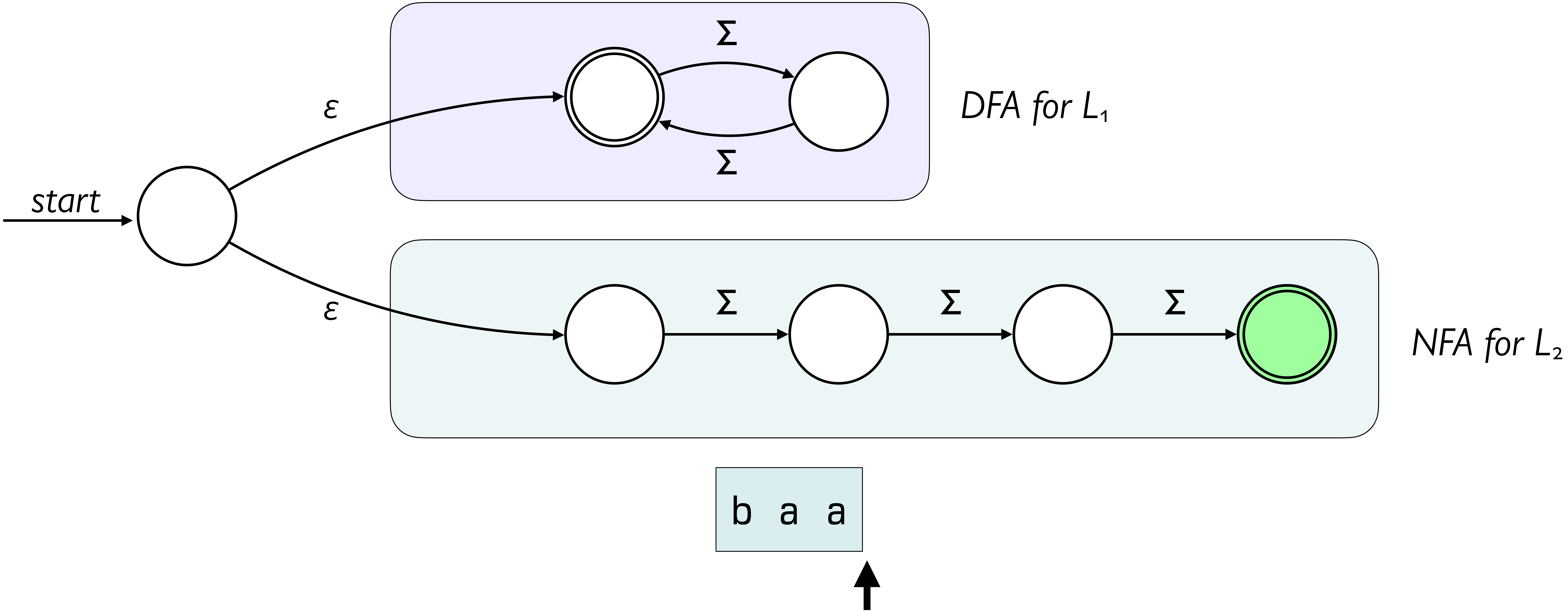
Construct an NFA for $L_1 \cup L_2$.



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

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

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 $L_1 \cup L_2$.

Intersection

The intersection of two languages

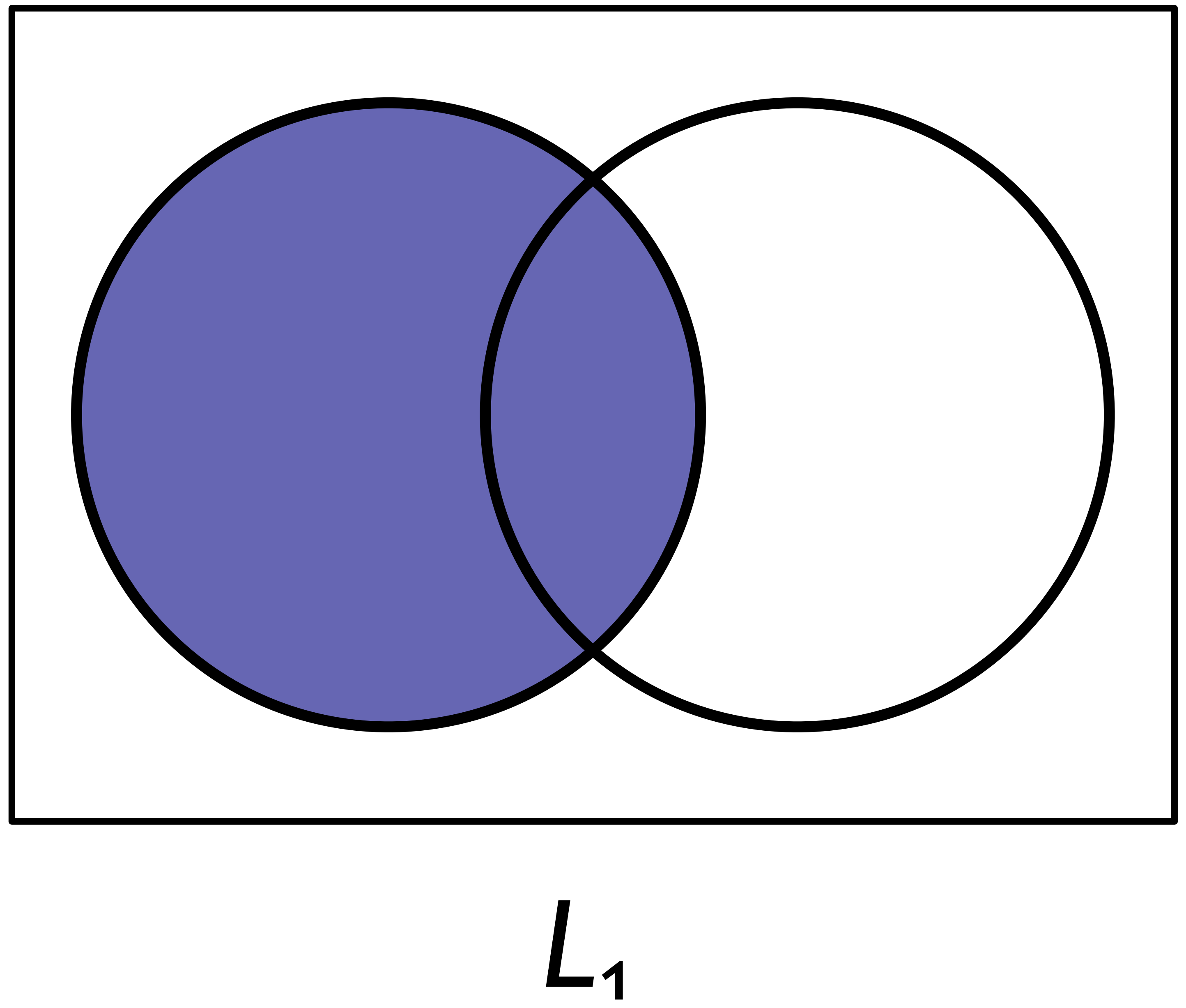
If L_1 and L_2 are languages over Σ, then $L_1 \cap L_2$ is the language of strings in both L_1 and L_2.

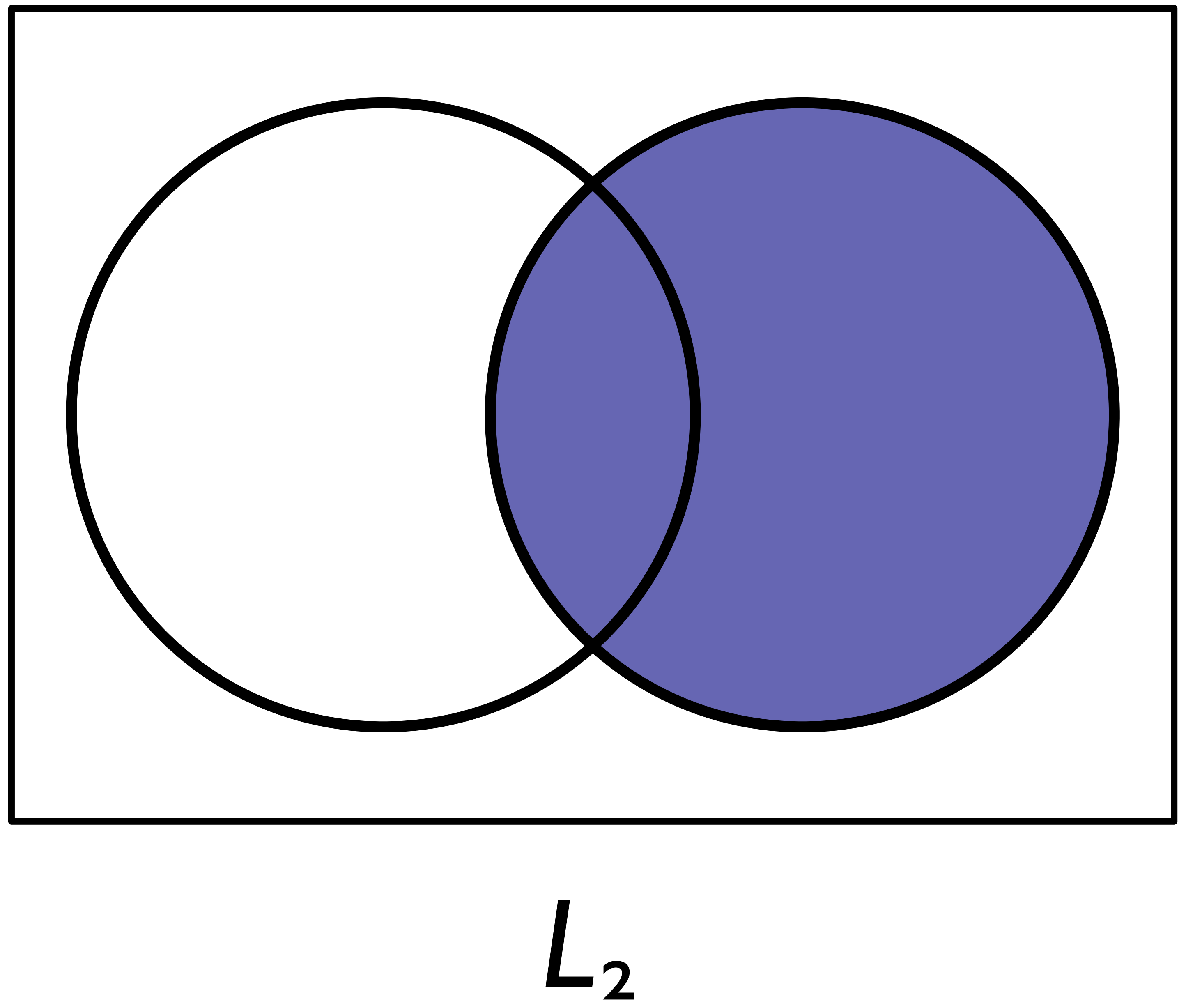
If L_1 and L_2 are both regular, is $L_1 \cap L_2$ regular?

The intersection of two languages

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.

If L_1 and L_2 are both regular, is $L_1 \cap L_2$ regular?

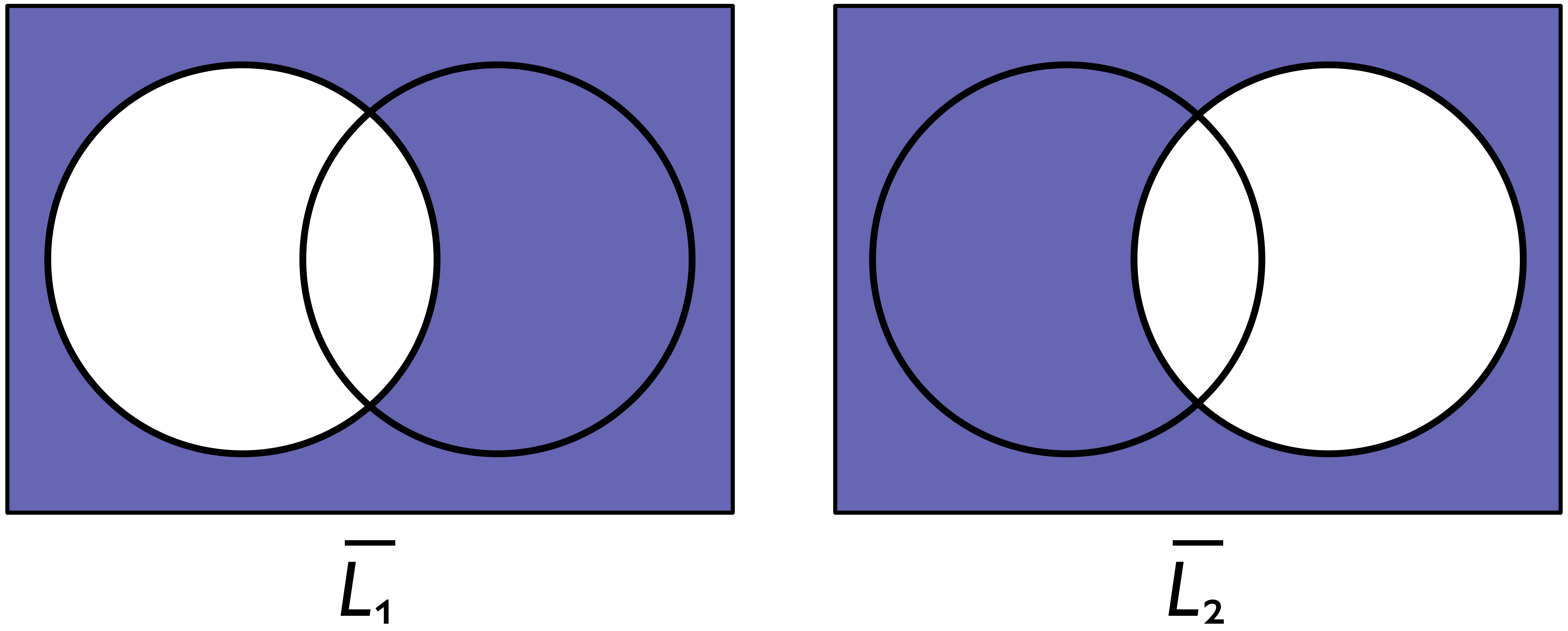




The intersection of two languages

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.

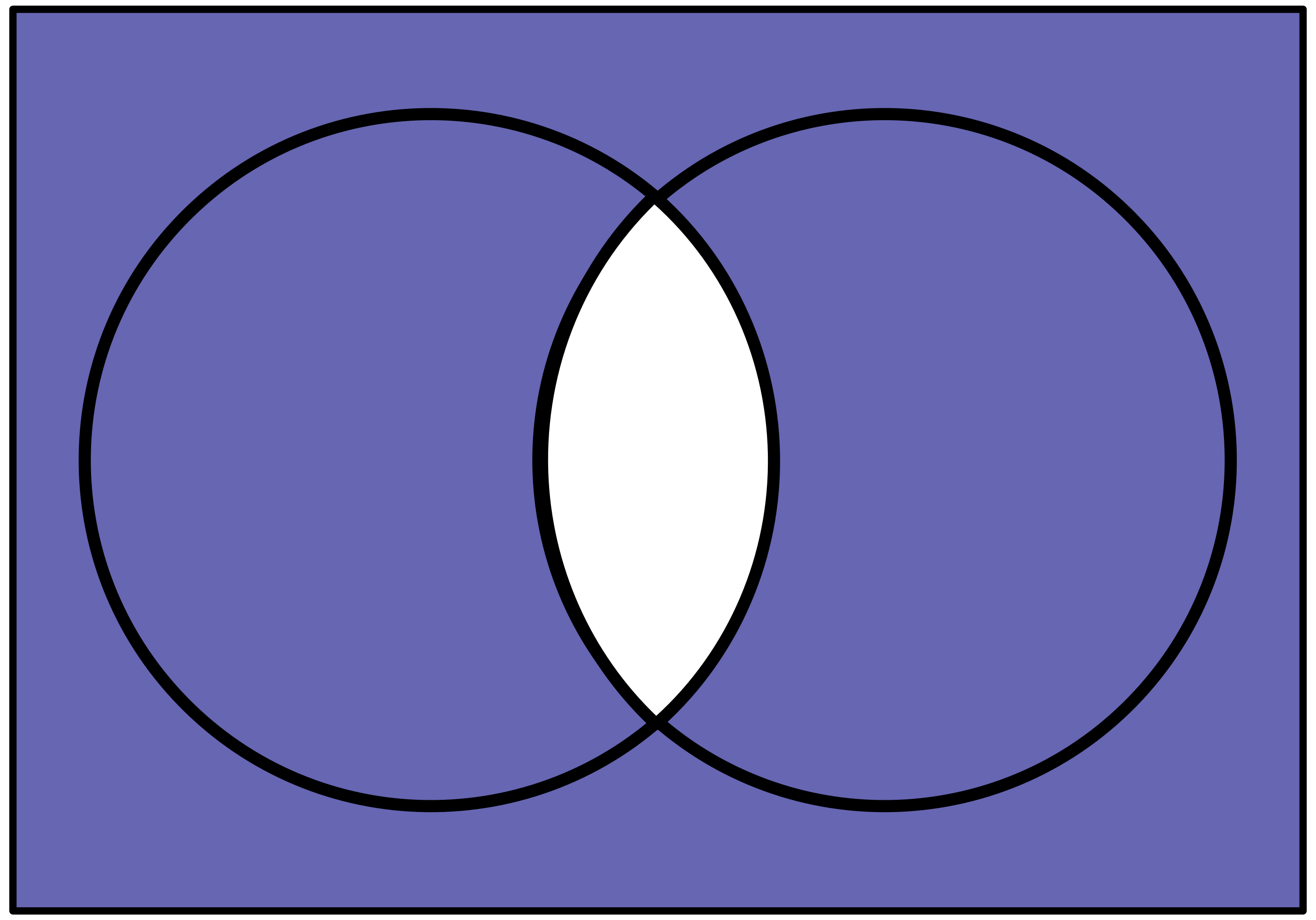
If L_1 and L_2 are both regular, is $L_1 \cap L_2$ regular?



The intersection of two languages

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.

If L_1 and L_2 are both regular, is $L_1 \cap L_2$ regular?

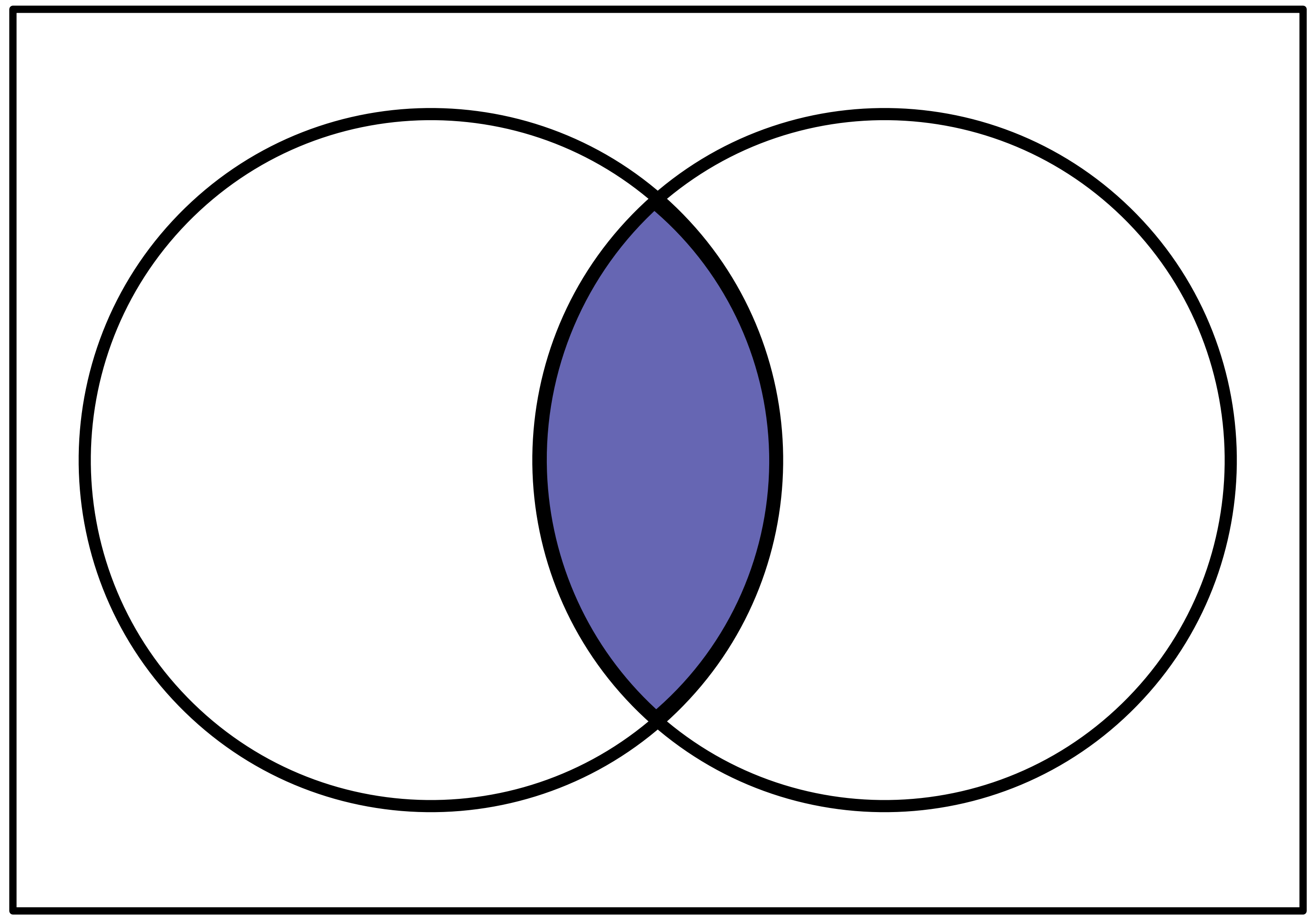


$\overline{L_1} \cup \overline{L_2}$

The intersection of two languages

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.

If L_1 and L_2 are both regular, is $L_1 \cap L_2$ regular?



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

De Morgan's law!

De Morgan's Law

In set theory,

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

or, in propositional logic,

$$\neg(A \vee B) = \neg A \wedge \neg B$$

which you've probably encountered when programming! For example, in Python,

`if not (A or B): ...` *is equivalent to* `if not A and not B: ...`

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

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

$$= L_1 \cap L_2$$

Double complement – like double negation – cancels out.

Concatenation

Concatenation of strings

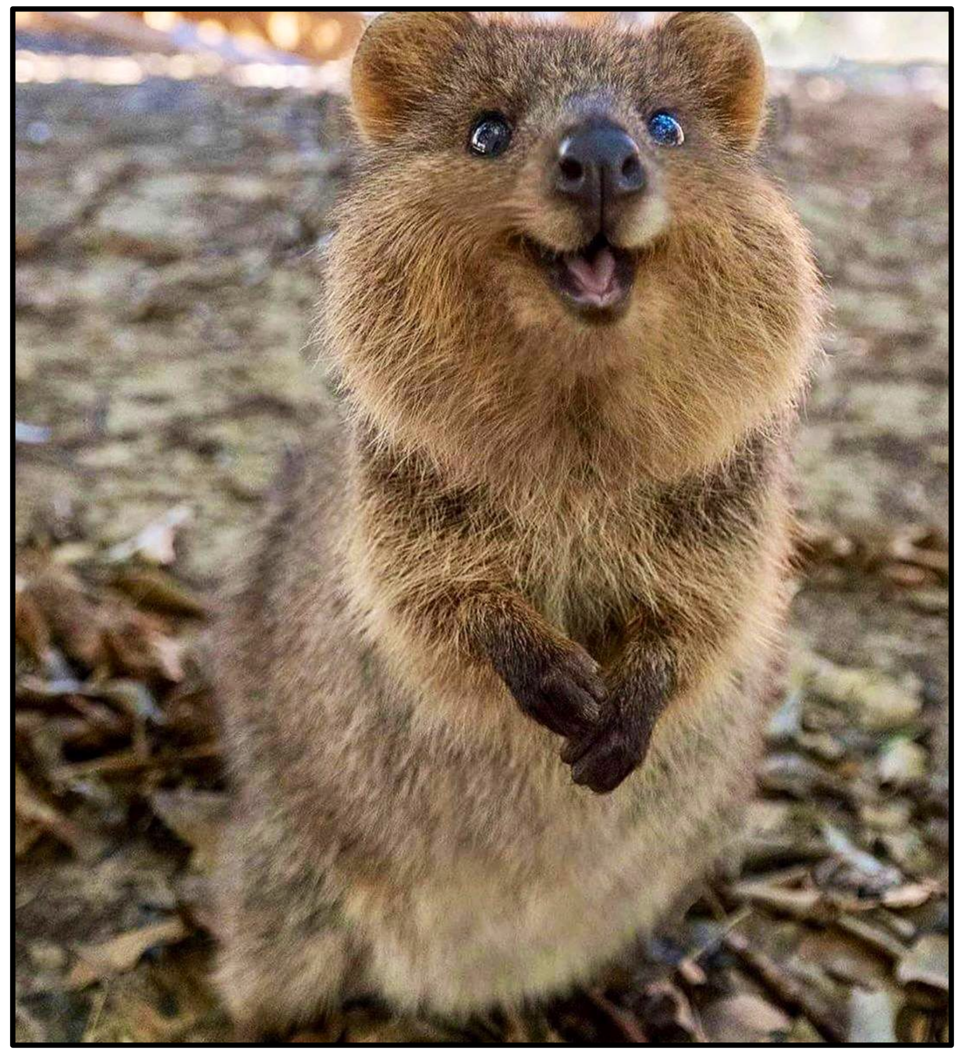
Recall: If $w \in \Sigma^*$ and $x \in \Sigma^*$, the *concatenation* of w and x, denoted $w \circ x$ or just wx, is the string formed by tacking all characters in x onto the end of w.

E.g., if w = `quo` and x = `kka`, the concatenation wx = `quokka`

Concatenation of strings

Recall: If $w \in \Sigma^*$ and $x \in \Sigma^*$, the *concatenation* of w and x, denoted $w \circ x$ or just wx, is the string formed by tacking all characters in x onto the end of w.

E.g., if w = quo and x = kka, the concatenation wx = quokka



A quokka, happy just to be mentioned

Concatenation of strings

The empty string, ε, is the *identity element* for concatenation:

$w\varepsilon = \varepsilon w = w$

Concatenation is *associative*:

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

Concatenation of languages

The *concatenation* of two languages L_1 and L_2 over the alphabet Σ is the language

$$L_1L_2 = \{wx \in \Sigma^* \mid w \in L_1 \text{ and } x \in L_2\}$$

E.g., consider the languages

$Noun$ = {`Puppy`, `Rainbow`, `Whale`, …}

$Verb$ = {`Hugs`, `Juggles`, `Loves`, …}

Det = {`A`, `The`}

The language $Det\,Noun\,Verb\,Det\,Noun$ is

{`APuppyHugsTheWhale`, `TheRainbowJugglesTheRainbow`, `TheWhaleLovesAPuppy`, …}

Concatenation of languages

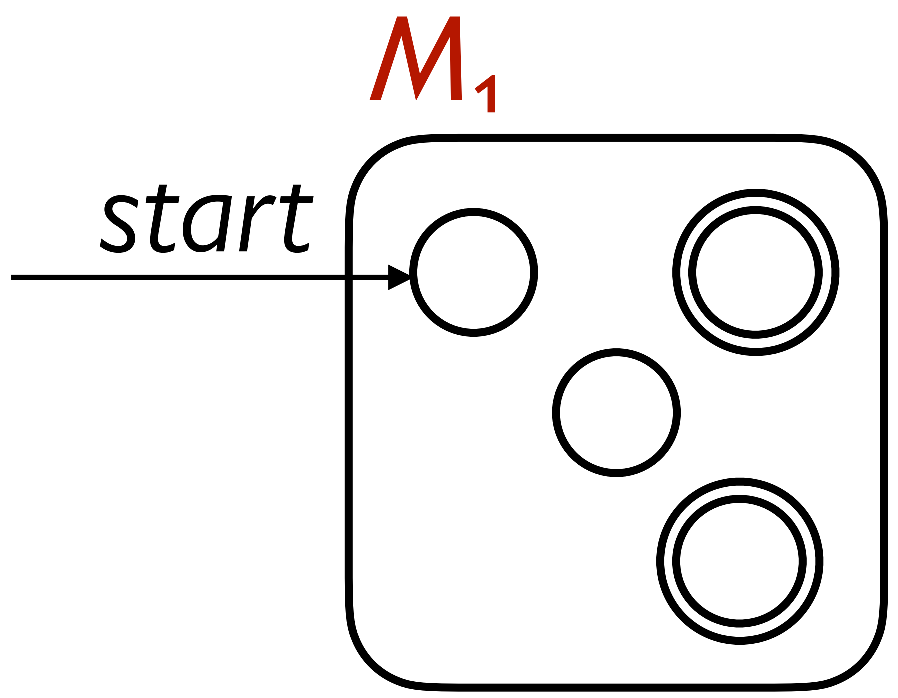
Two views of L_1L_2:

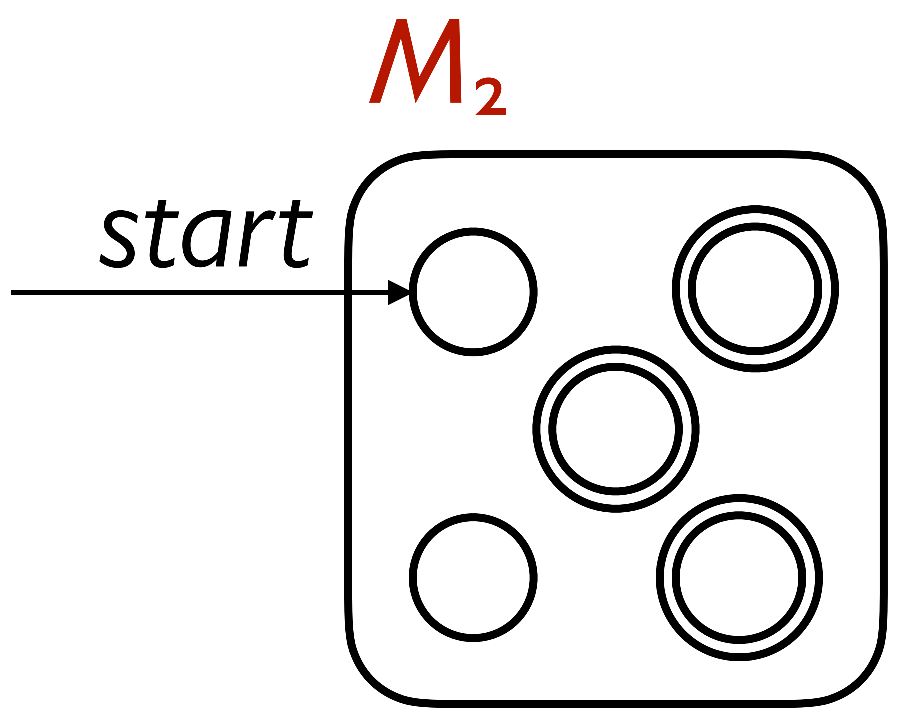
The set of all strings that can be *made* by concatenating a string in L_1 with a string in L_2.

The set of strings that can be *split* into two pieces: a piece from L_1 followed by a piece from L_2.

Conceptually it's similar to the Cartesian product of two sets, only with strings.

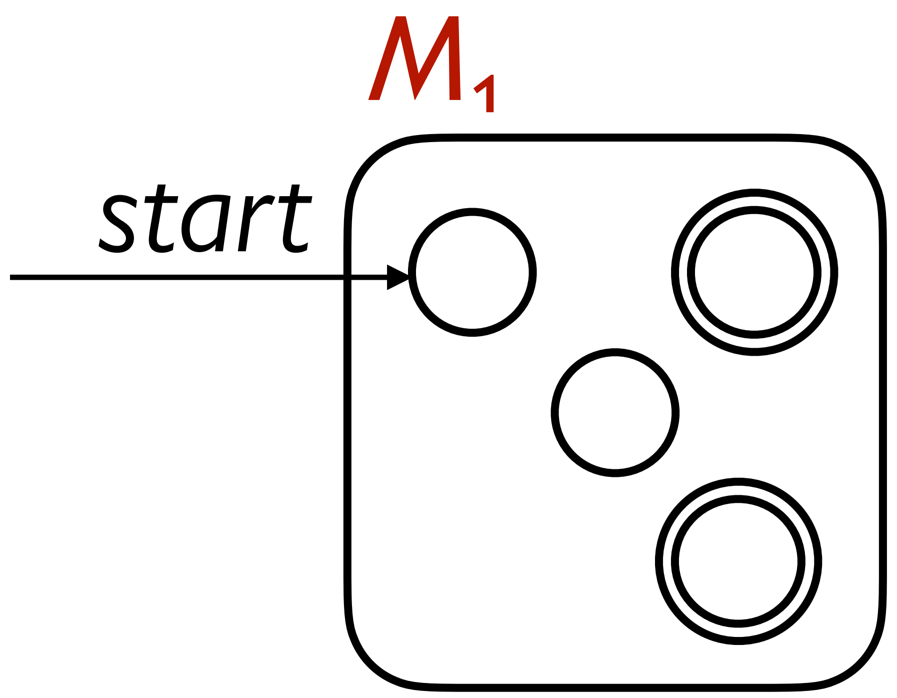
If L_1 and L_2 are regular languages, is L_1L_2?

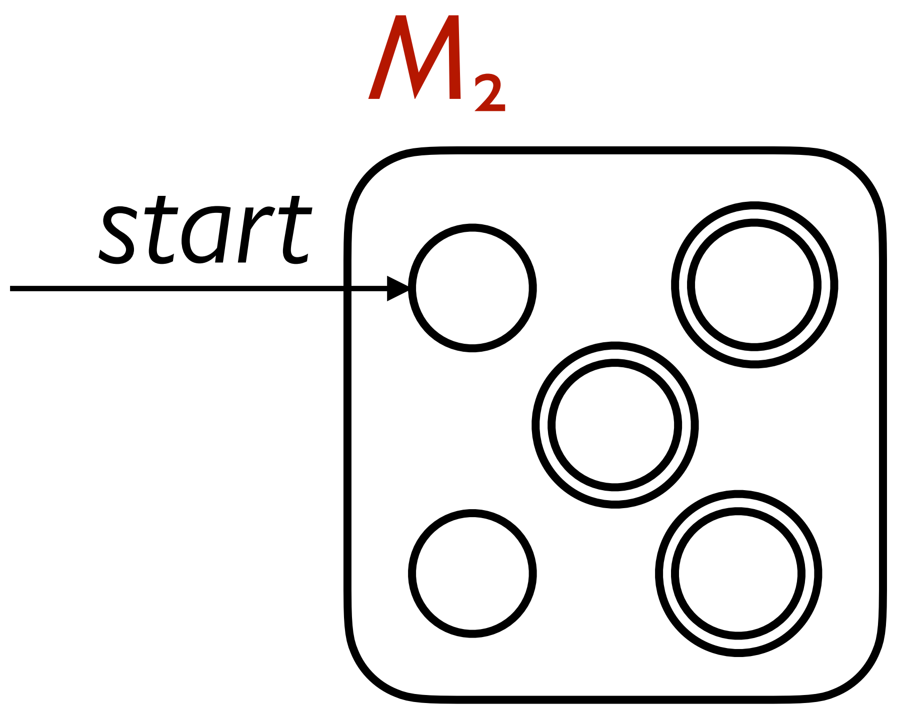




b o o k k e e p e r

If L_1 and L_2 are regular languages, is L_1L_2?





b o o k

k e e p e r

How could we know where the first string ends and the second begins?

There isn't a straightforward way to do this with a DFA; our model makes it too hard to keep track of the possibilities.

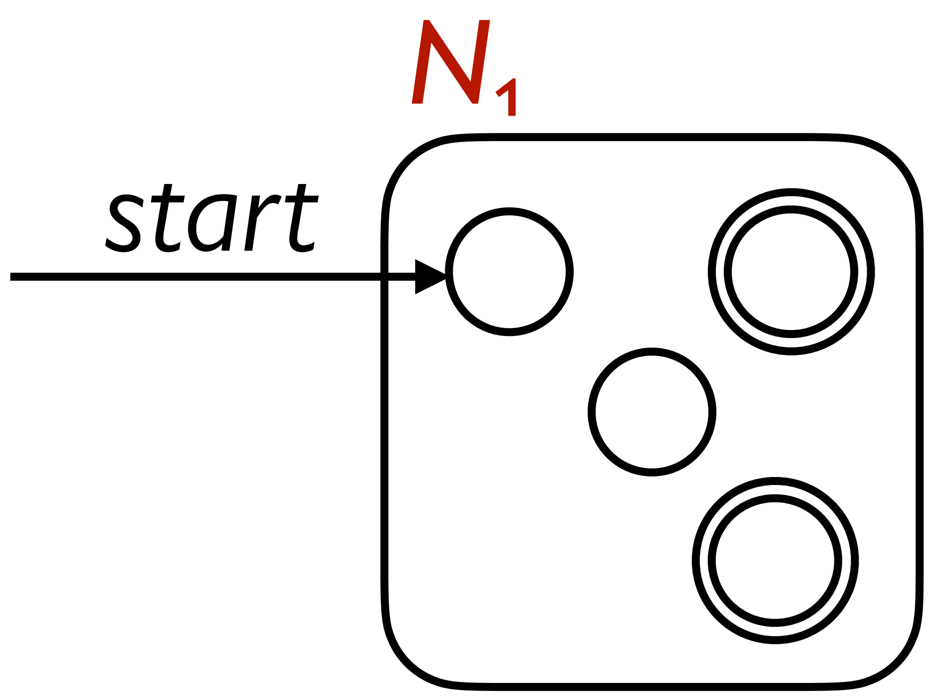
With NFAs, it's easy!

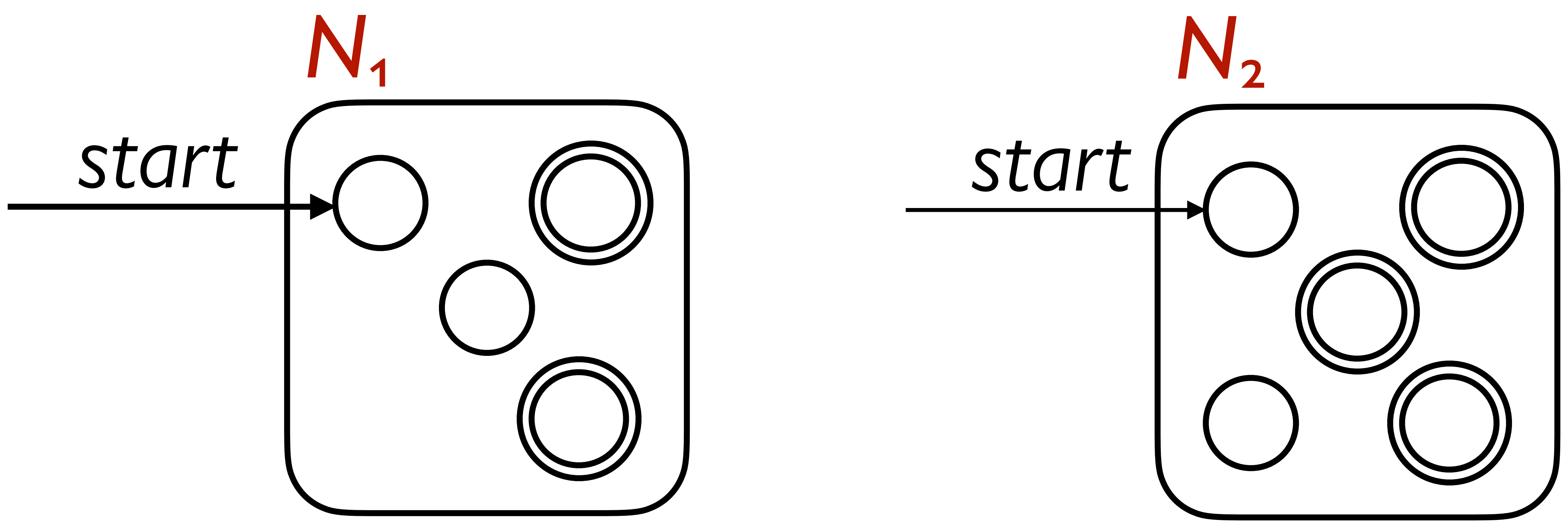
Given a string w, run a finite automaton for L_1 on w.

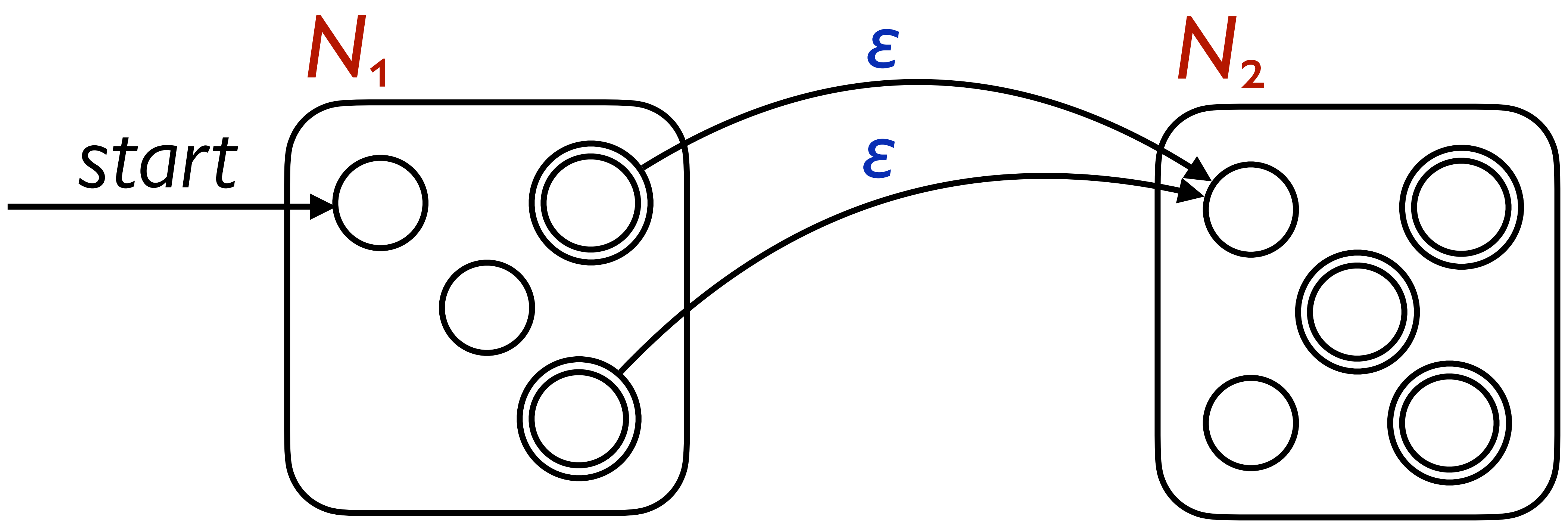
Whenever it reaches an accept state, optionally hand the rest of w to the finite automaton for L_2.

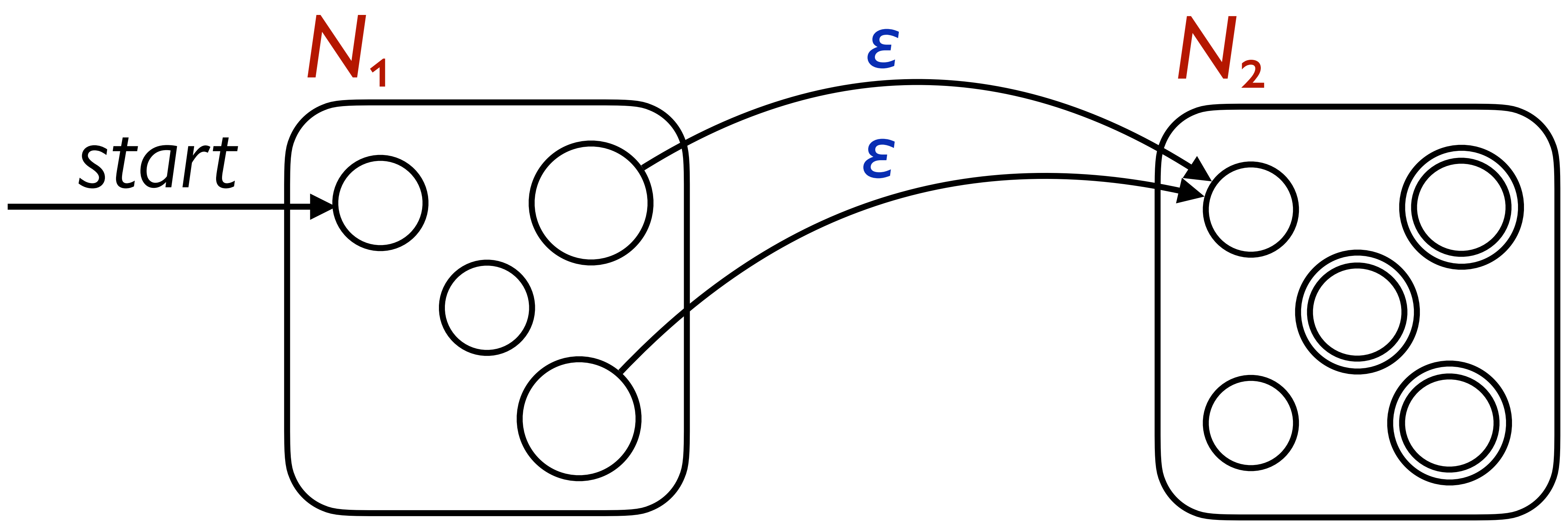
If the automaton for L_2 accepts the rest, $w \in L_1L_2$.

If the automaton for L_2 rejects the remainder, either $w \notin L_1L_2$ or the split was incorrect.

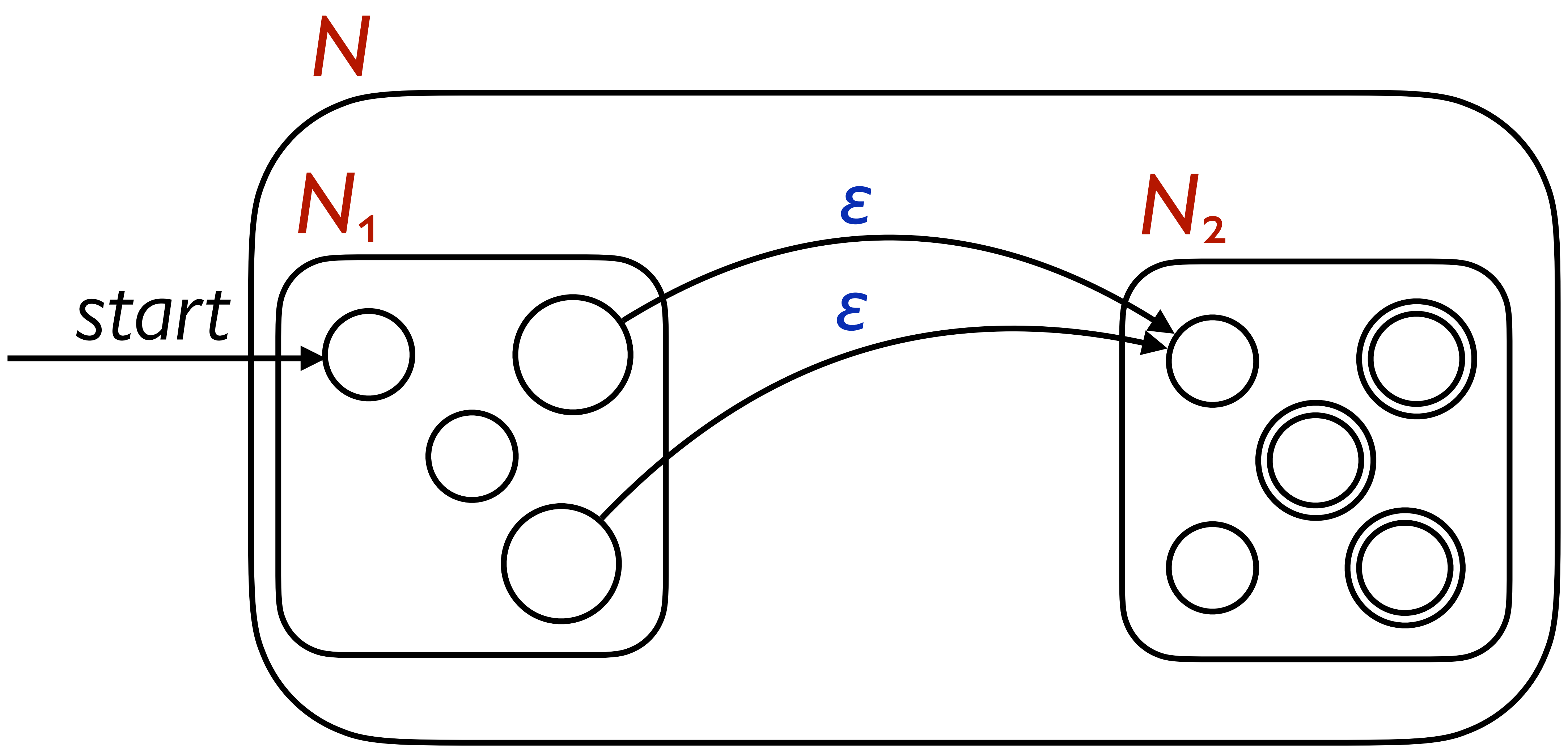

N_1
start


N1
start
N2
start


N1
ε
N2
start
ε



The new machine guesses non-deterministically where to split the input in order to have a first part accepted by N_1 and a second part accepted by N_2.



$L(N) = L_1L_2$

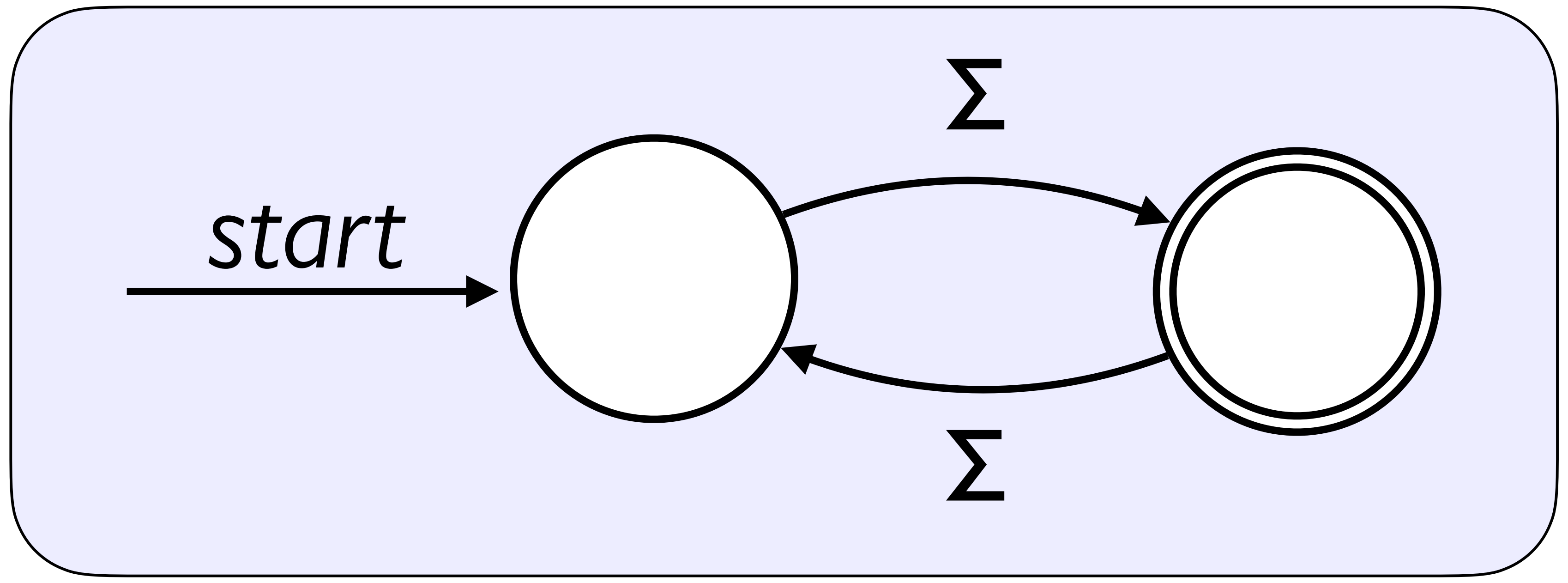
This construction proves the class of regular languages is closed under concatenation.

Example

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

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

Construct an NFA for L_1L_2.

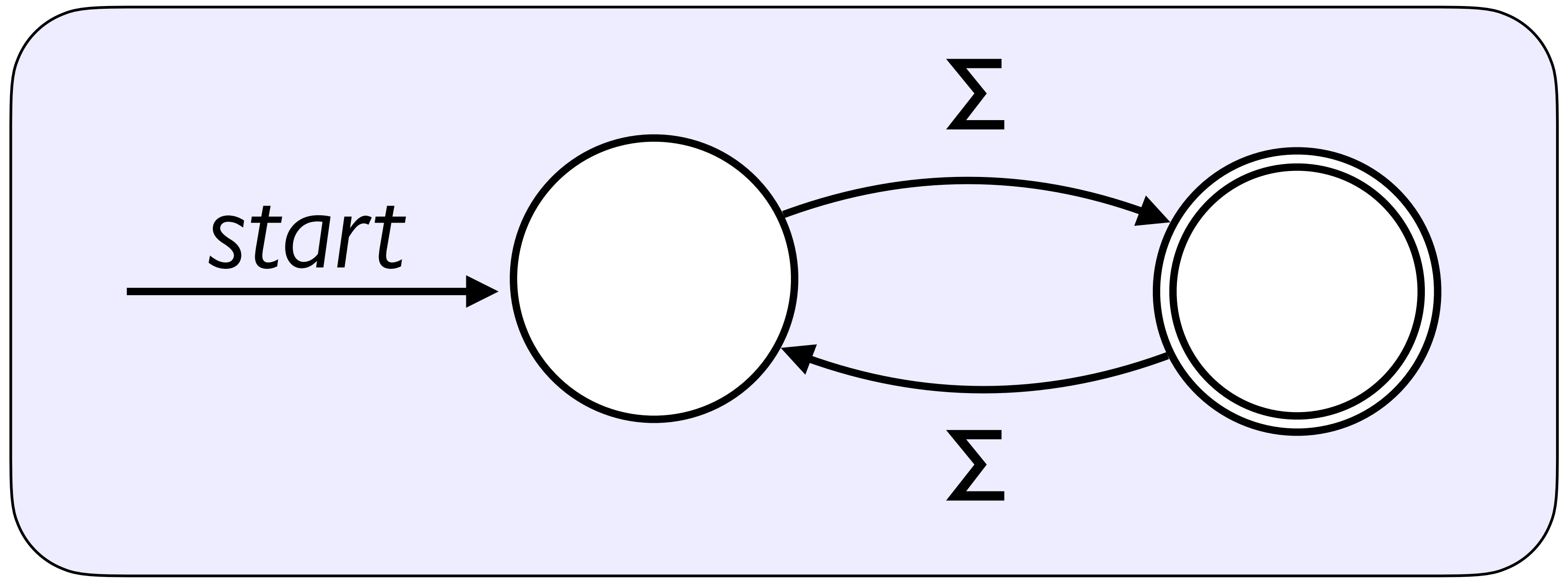


DFA for L_1

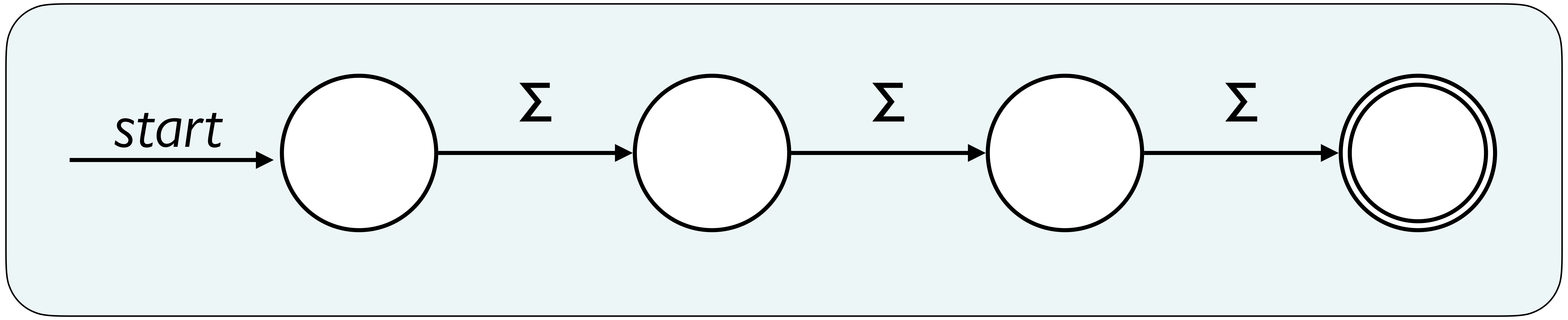
$L_1 = \{w \in \{a, b\}^* \mid w \text{ has } odd \text{ length}\}$

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

Construct an NFA for L_1L_2.



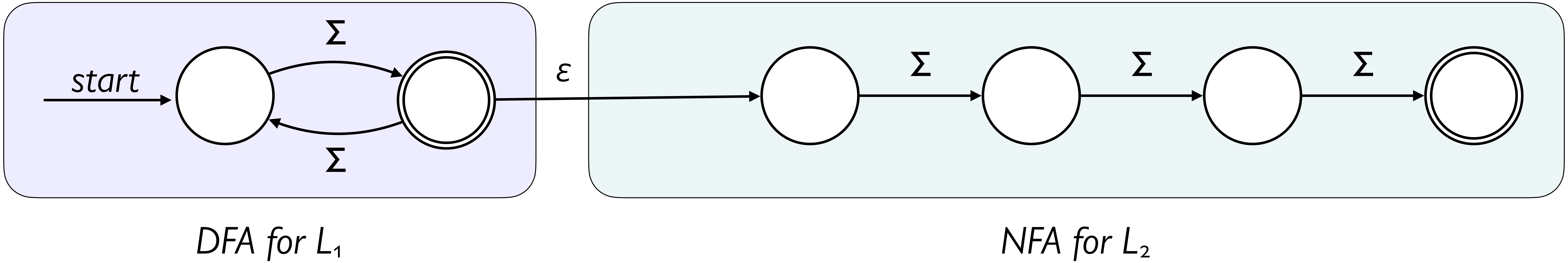
DFA for L_1



NFA for L_2

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

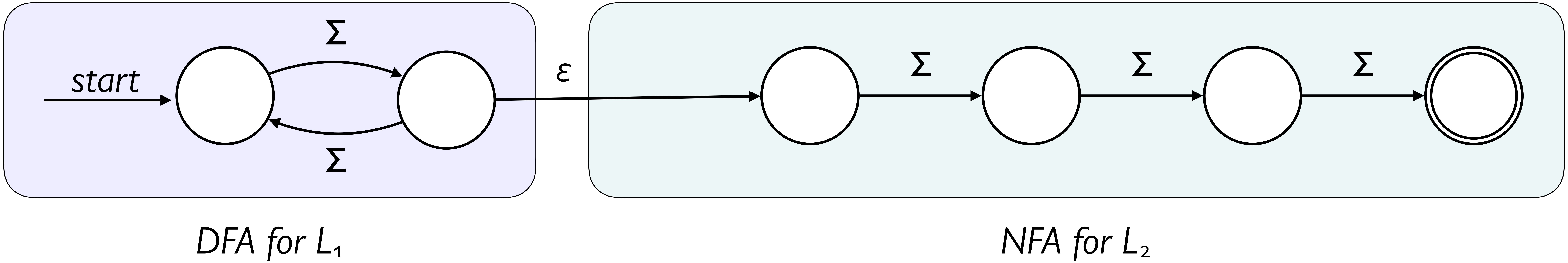
Construct an NFA for L_1L_2.



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

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

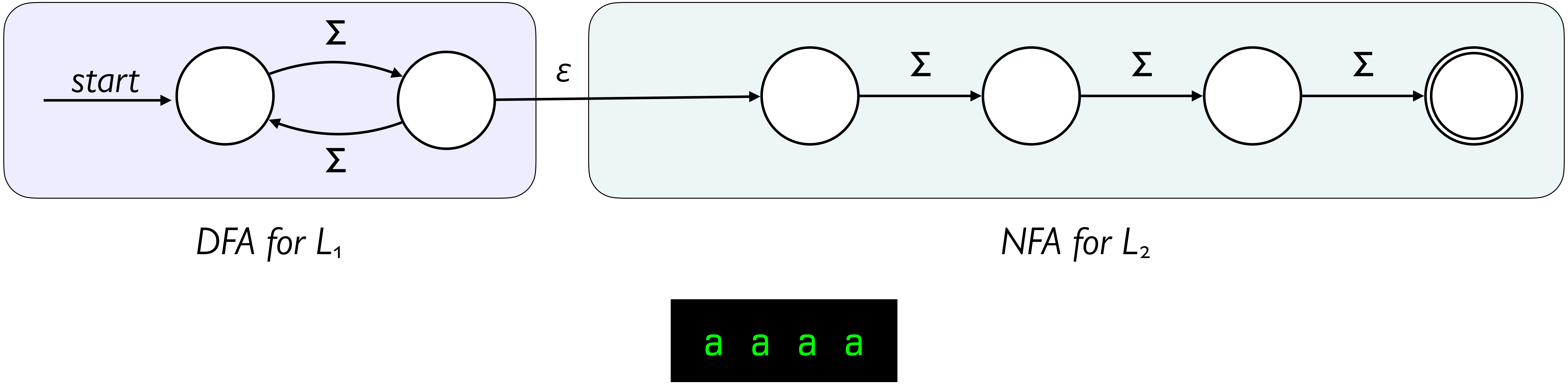
Construct an NFA for L_1L_2.



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

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

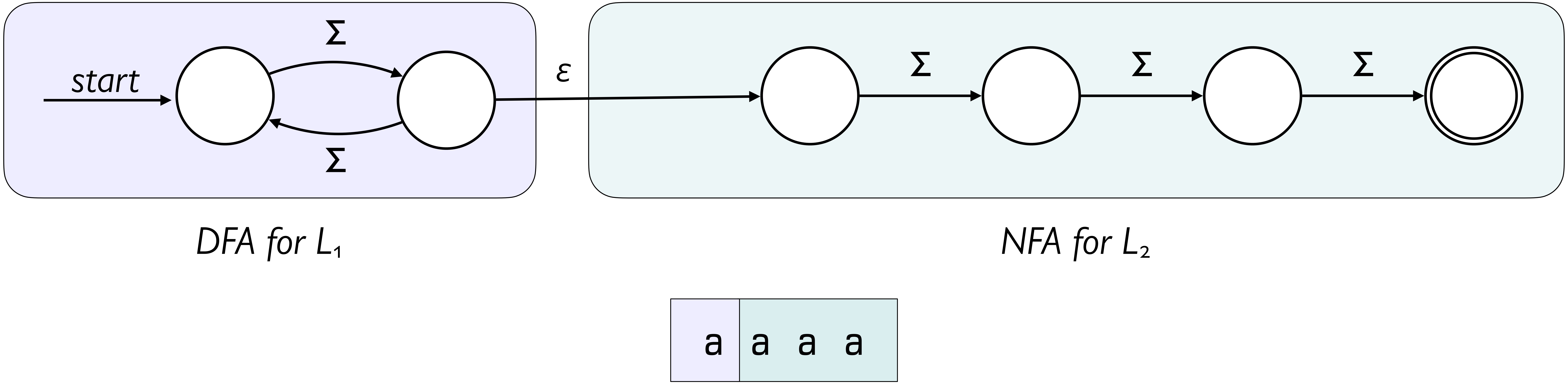
Construct an NFA for L_1L_2.



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

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

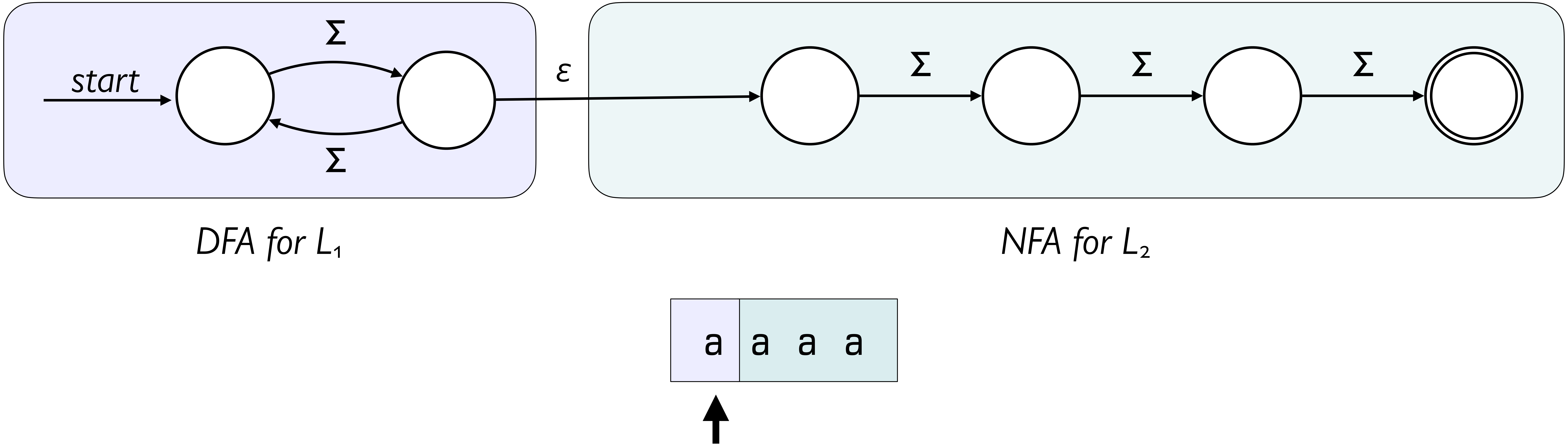
Construct an NFA for L_1L_2.



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

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

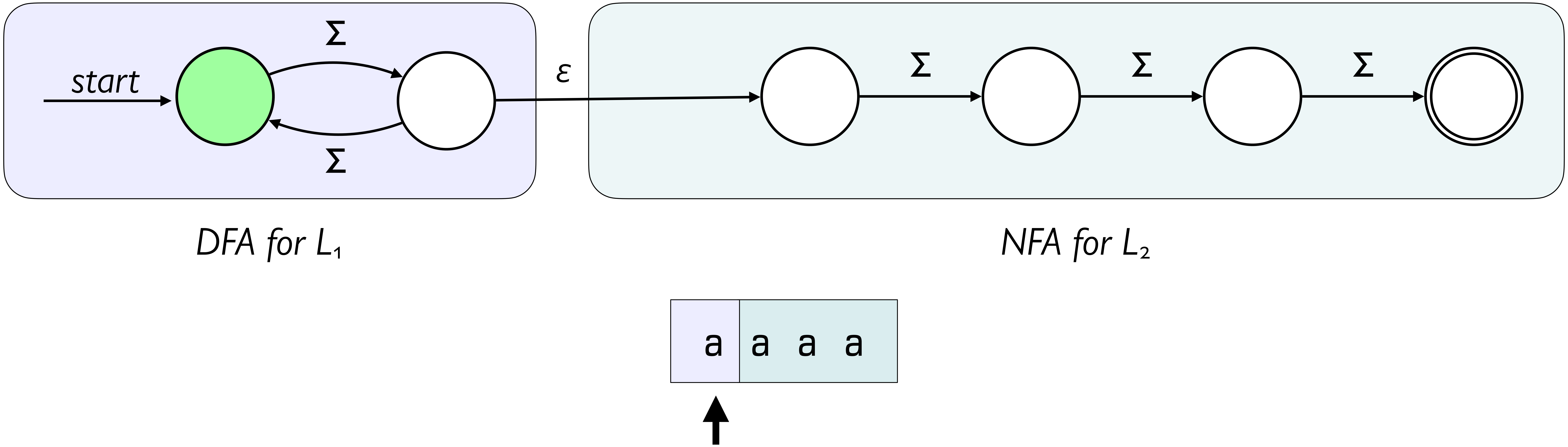
Construct an NFA for L_1L_2.



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

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

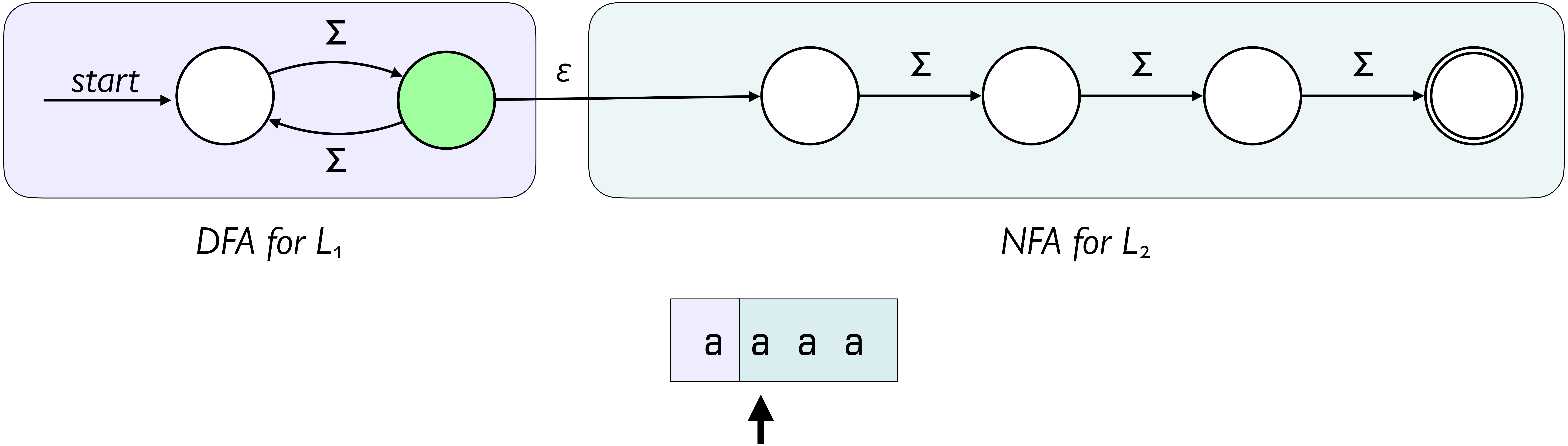
Construct an NFA for L_1L_2.



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

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

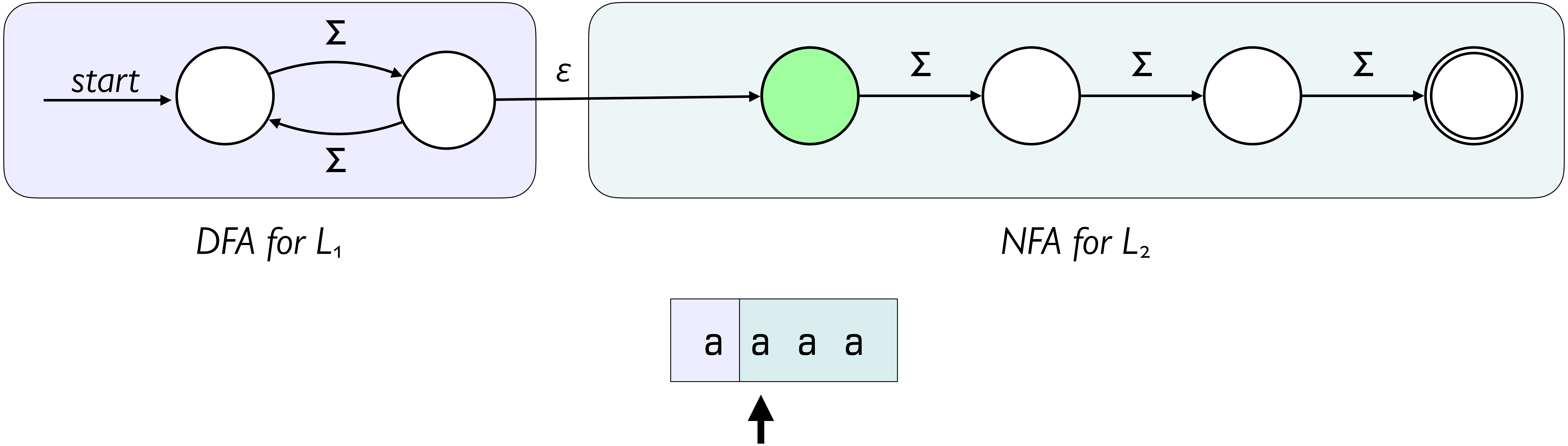
Construct an NFA for L_1L_2.



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

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

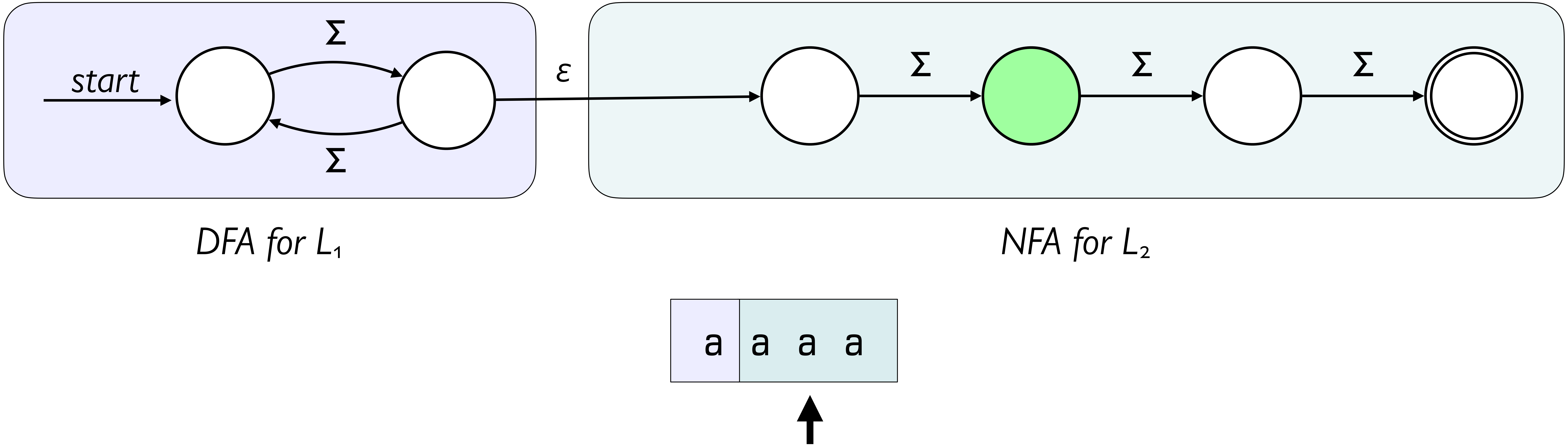
Construct an NFA for L_1L_2.



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

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

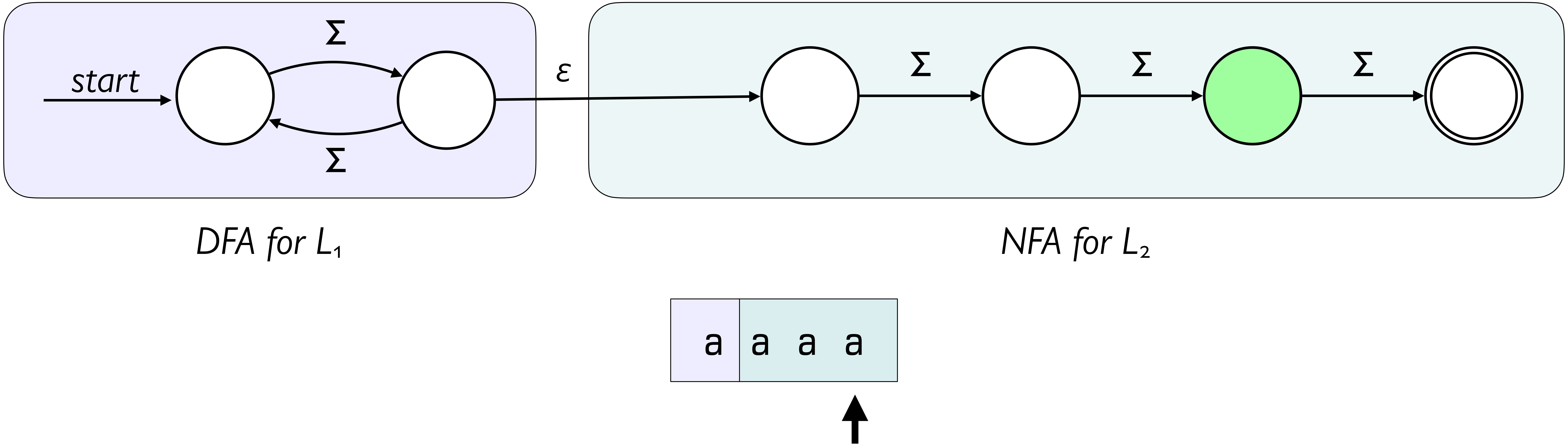
Construct an NFA for L_1L_2.



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

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

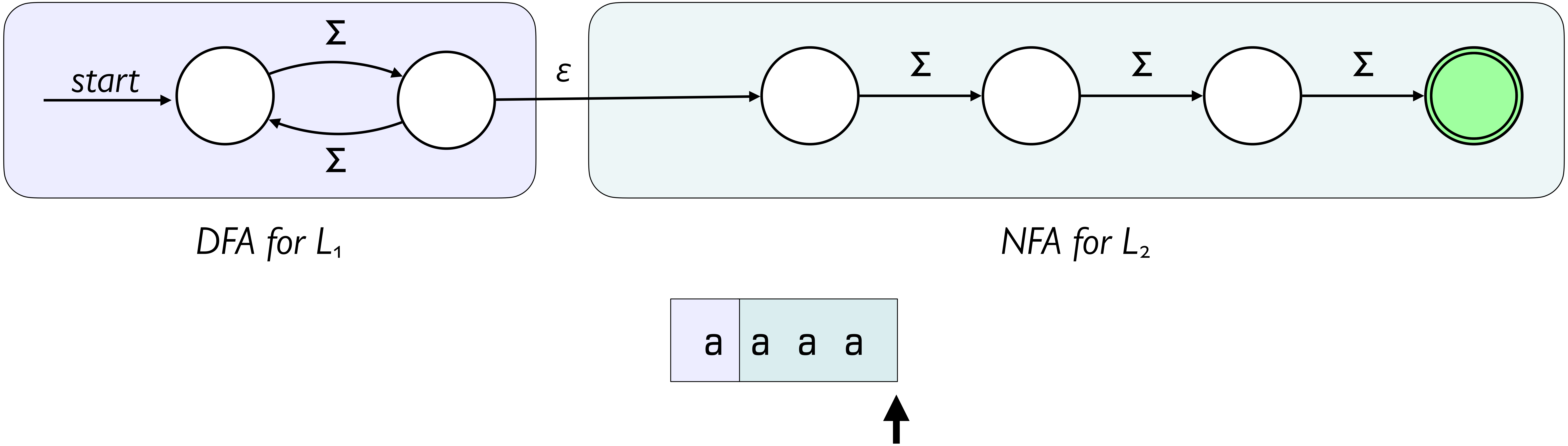
Construct an NFA for L_1L_2.



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

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

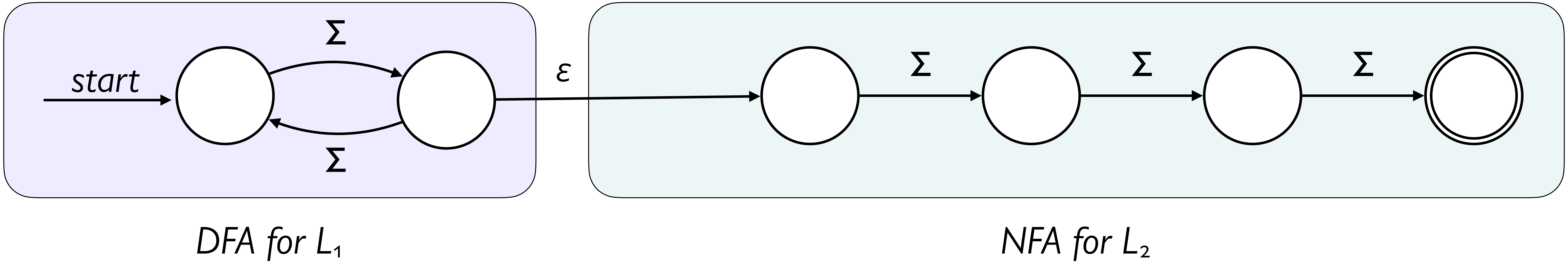
Construct an NFA for L_1L_2.



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

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

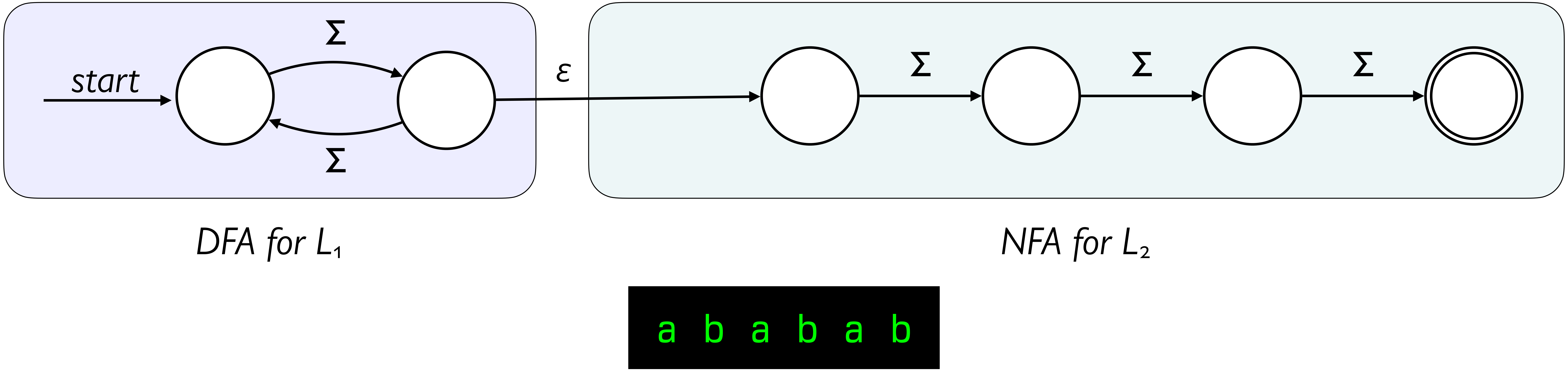
Construct an NFA for L_1L_2.



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

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

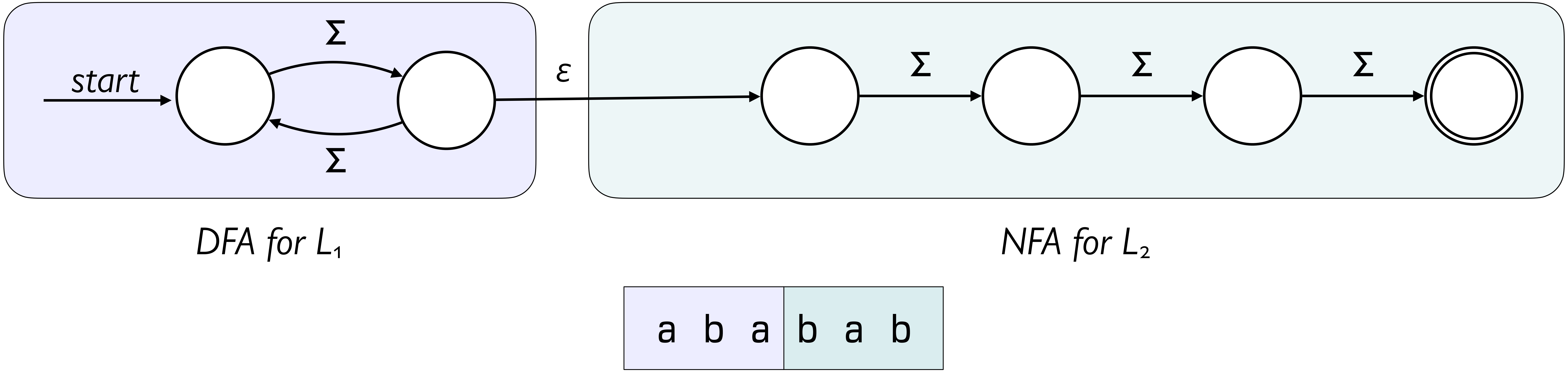
Construct an NFA for $L_1 L_2$.



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

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

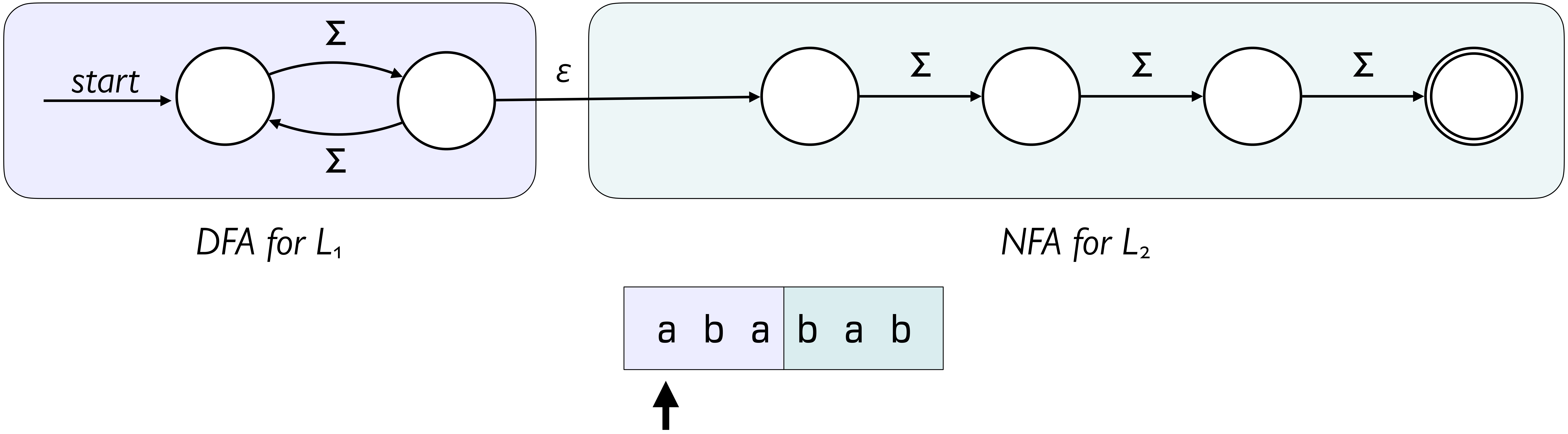
Construct an NFA for L_1L_2.



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

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

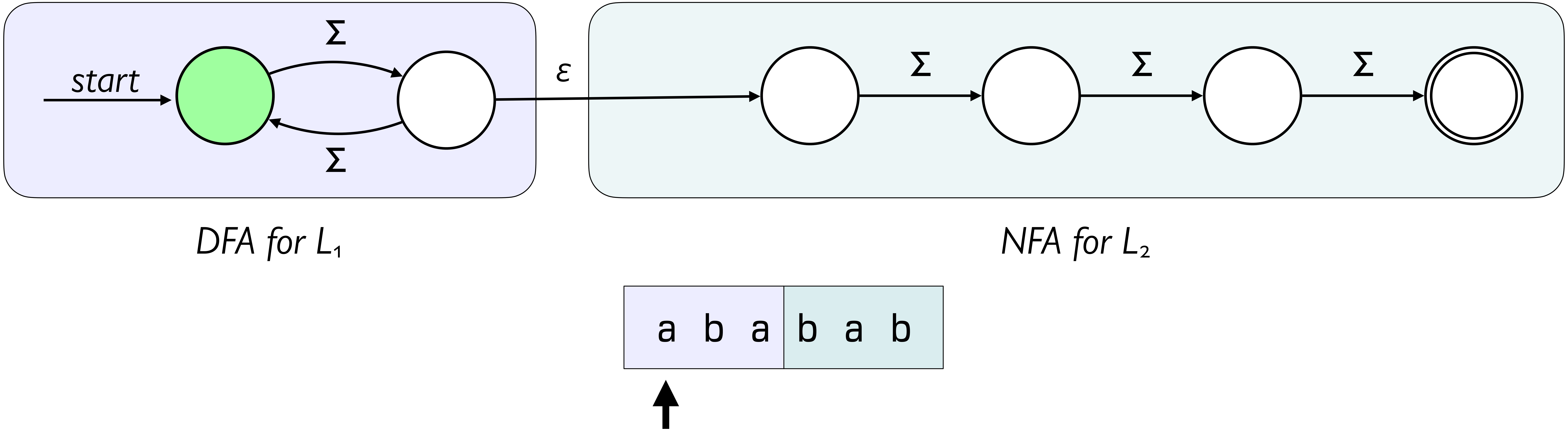
Construct an NFA for L_1L_2.



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

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

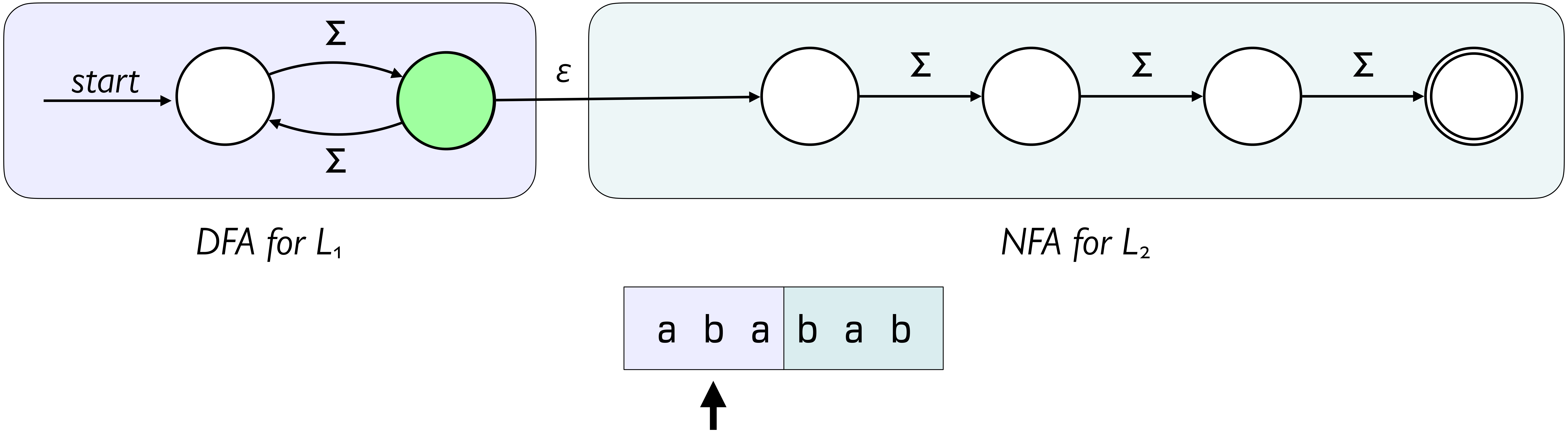
Construct an NFA for L_1L_2.



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

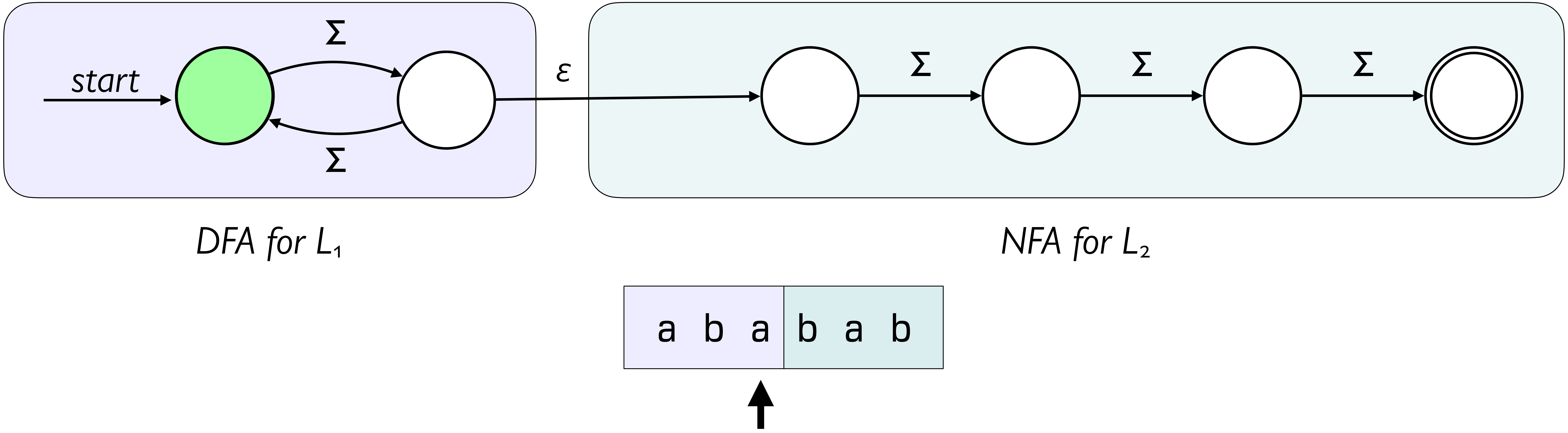
$L_2 = \{w \in \{a, b\}^* \mid w \text{ has length exactly three}\}$

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}

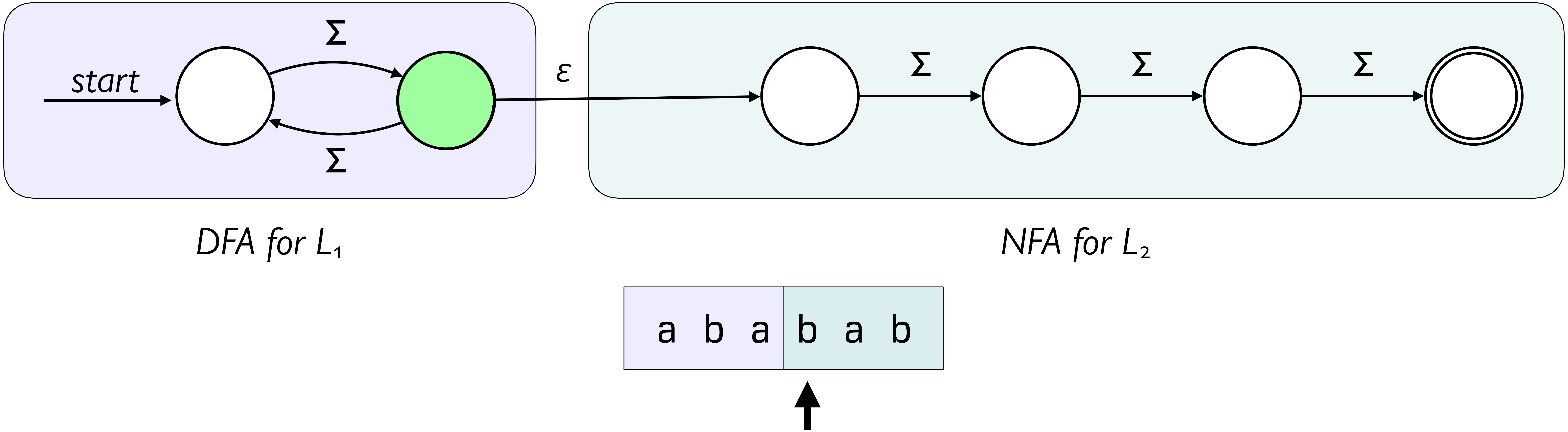
Construct an NFA for L_1L_2.



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

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

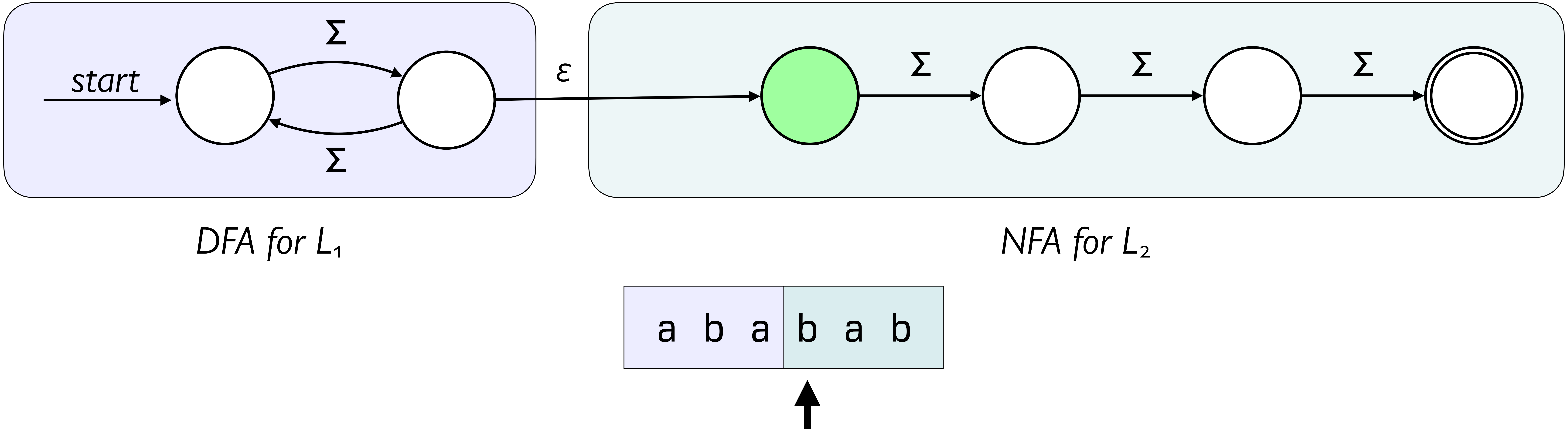
Construct an NFA for L_1L_2.



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

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

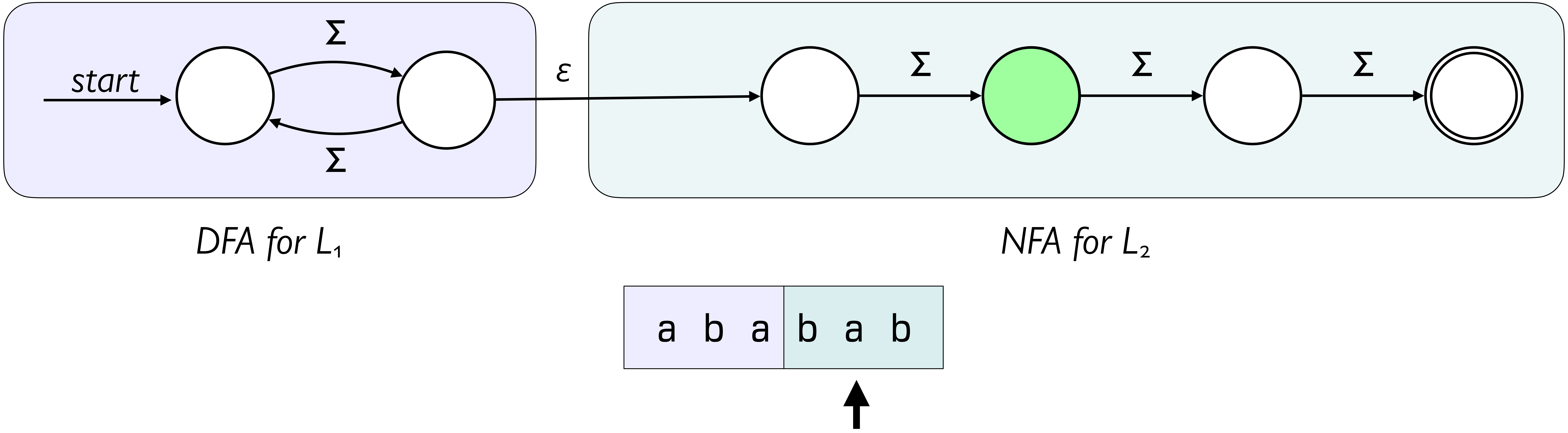
Construct an NFA for L_1L_2.



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

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

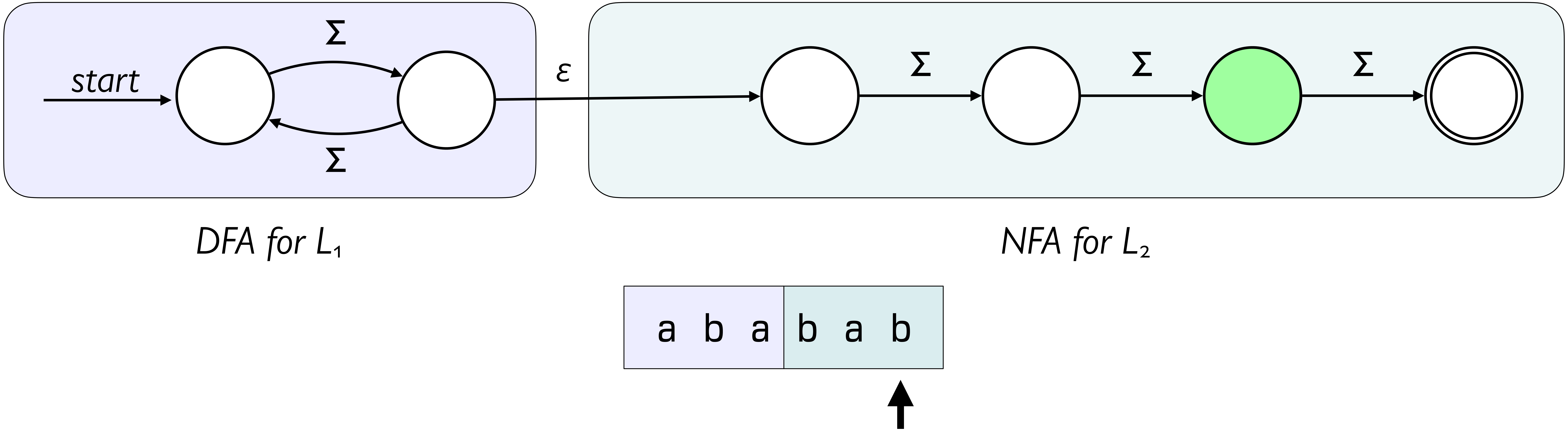
Construct an NFA for L_1L_2.



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

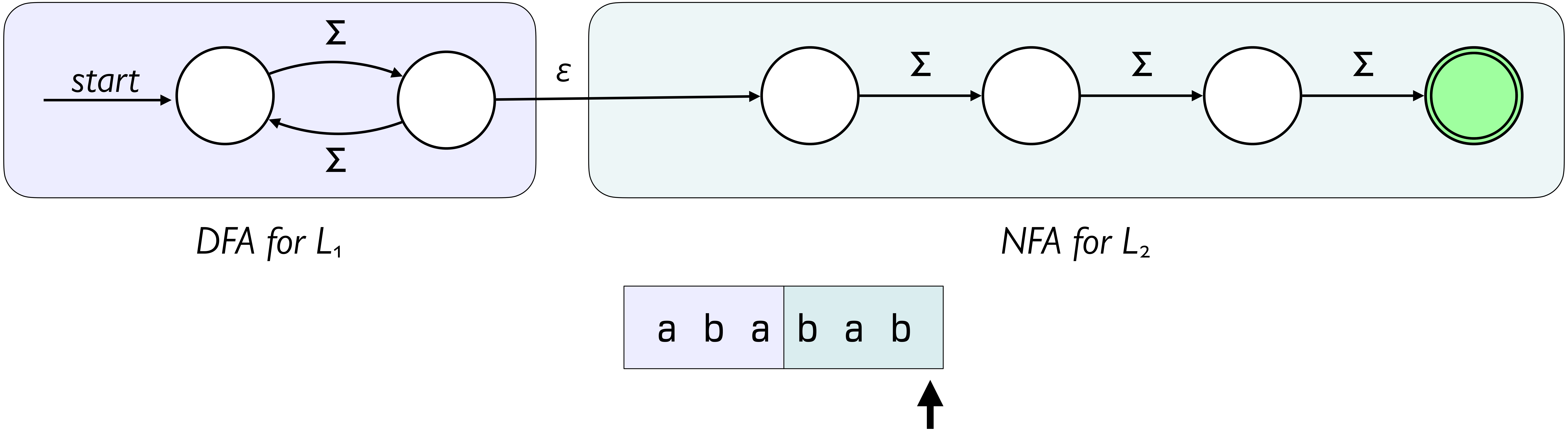
$L_2 = \{w \in \{a, b\}^* \mid w$ has length exactly three$\}$

Construct an NFA for L_1L_2.



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

Construct an NFA for L_1L_2.



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

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

Construct an NFA for L_1L_2.

Kleene star

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*…

Language exponentiation

We can define what it means to "exponentiate" a language as follows:

$L^0 = \{\varepsilon\}$

Base case: Any string formed by concatenating zero strings together is just the empty string.

$L^{n+1} = L L^n$

Recursive case: Concatenating $n+1$ strings together works by concatenating n strings, then concatenating one more.

Kleene (star) closure

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

$$L^* = \{w \in \Sigma^* \mid \exists n \in \mathbb{N}_0 \,.\, w \in L^n\}$$

A word is in L^* iff it's in one of the languages L^0, L^1, L^2, …

That is, L^* consists of all the possible ways of concatenating zero or more strings in L.

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

ε,

a, bb,

aa, abb, bba, bbbb,

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

…

}

If L is a regular language, is L^* necessarily regular?

A *bad* line of reasoning

If L is regular,

$L^0 = \{\varepsilon\}$ is regular.

$L^1 = L$ is regular.

$L^2 = LL$ is regular.

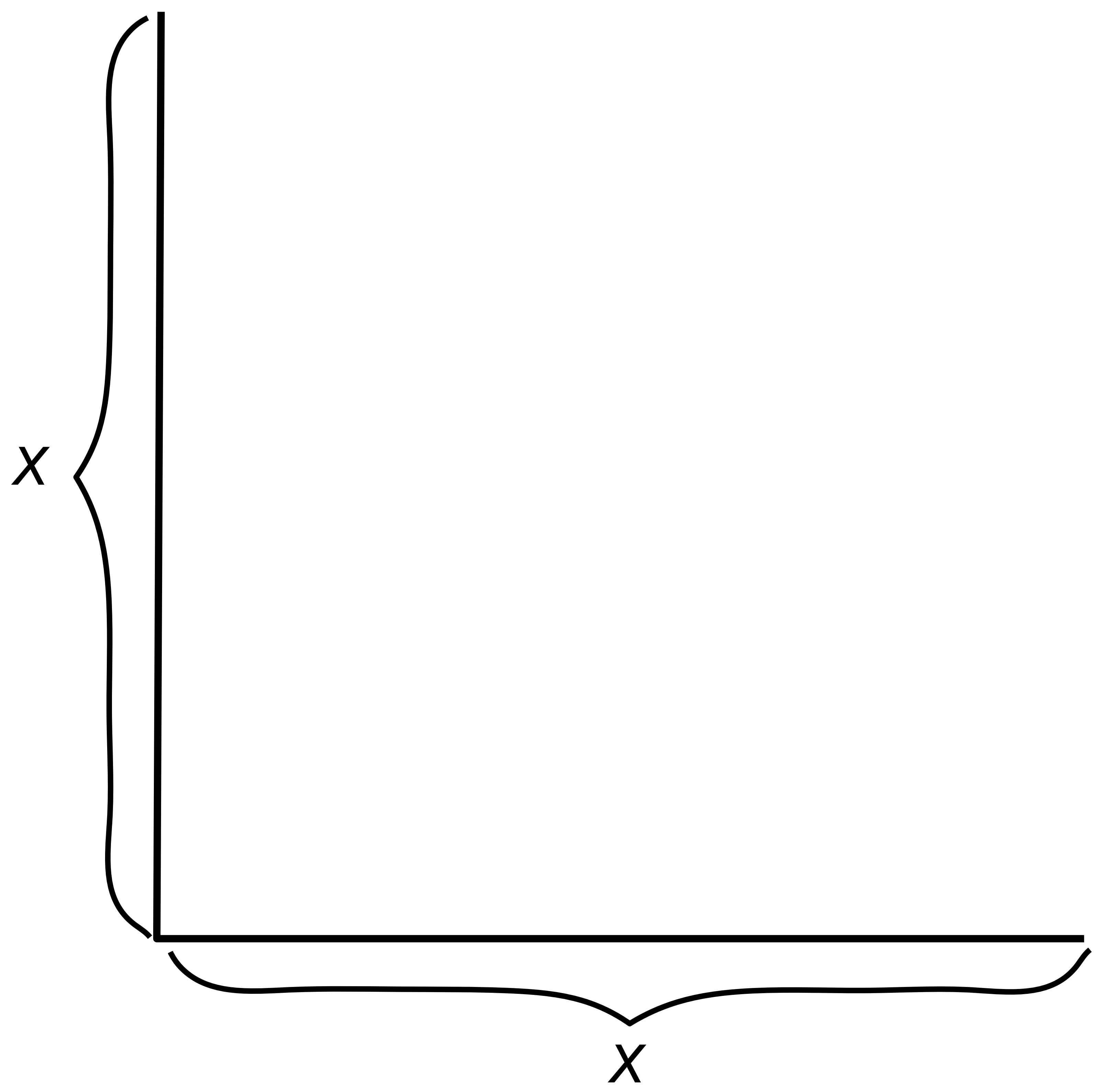
$L^3 = L(LL)$ is regular.

…

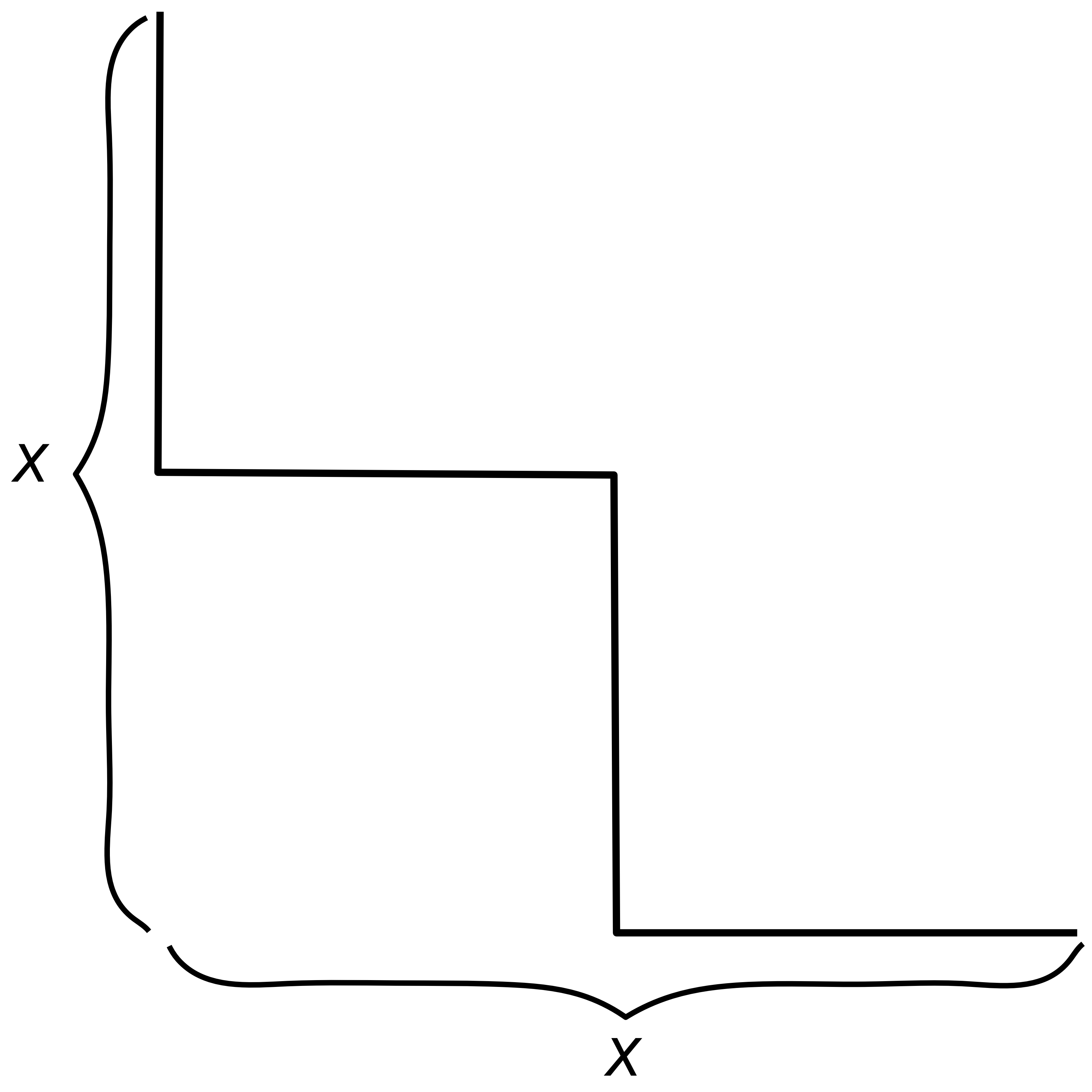
Regular languages are closed under union.

So, the union of all these languages is regular.

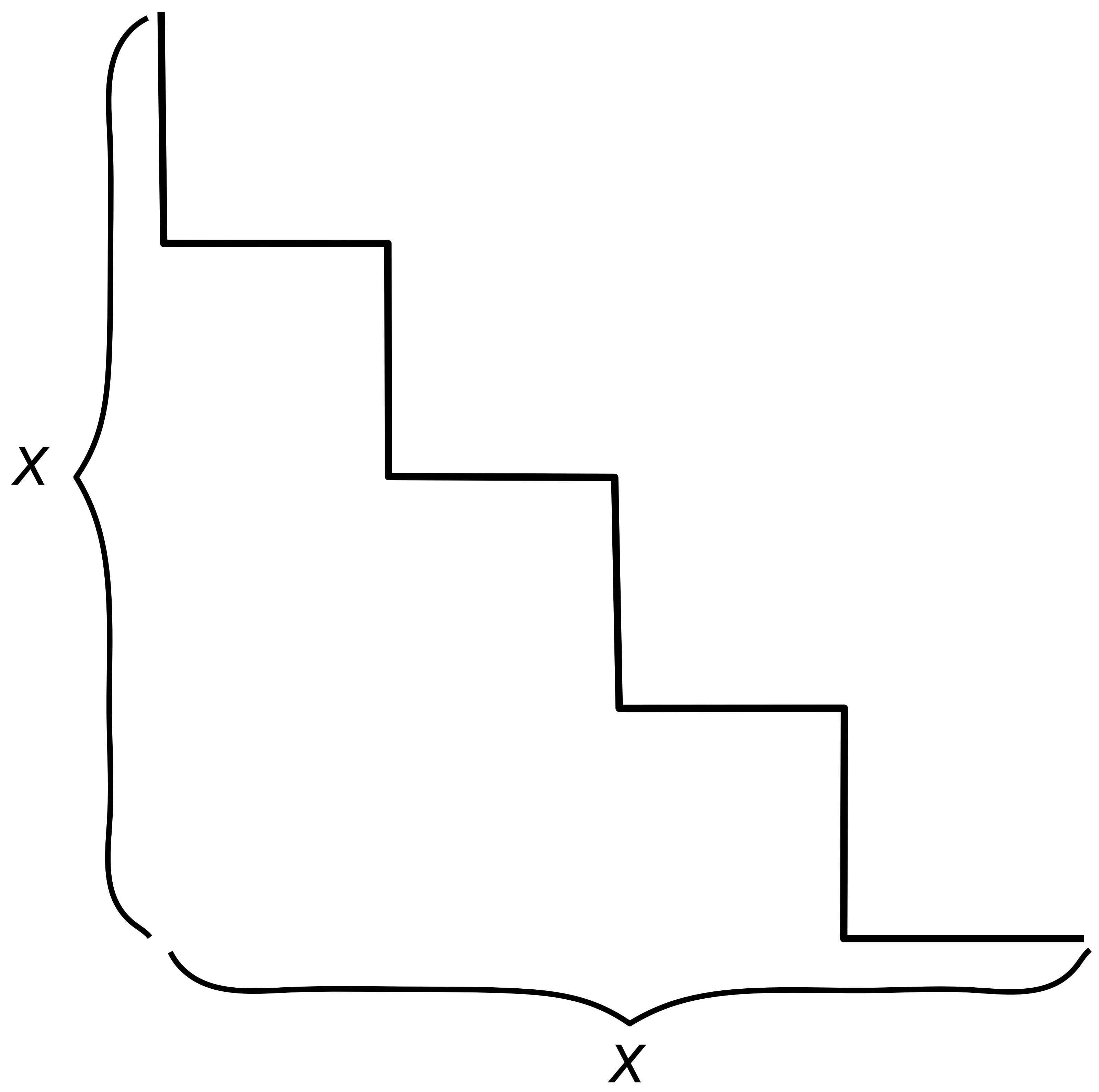
Reasoning about infinity



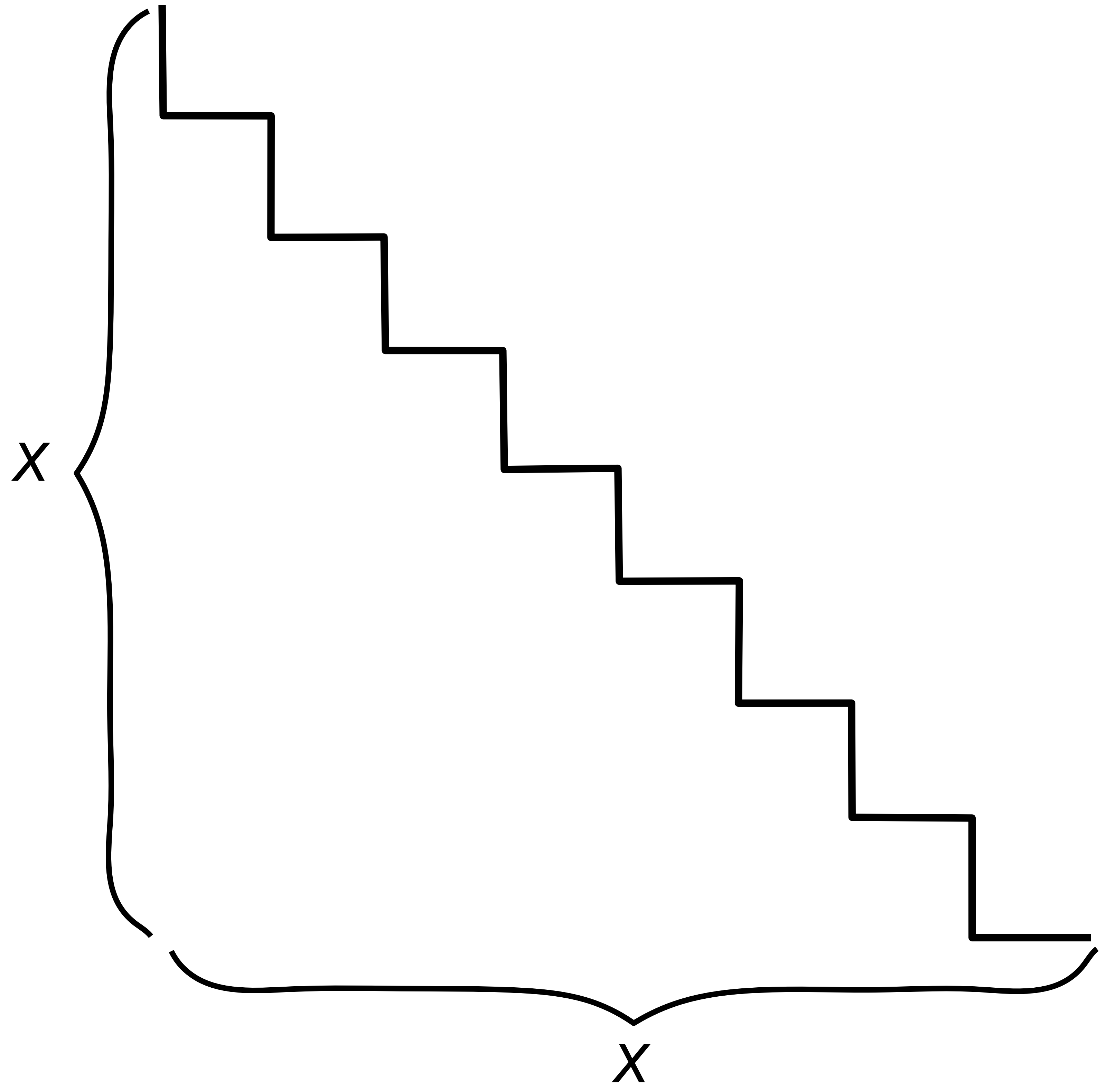
Reasoning about infinity



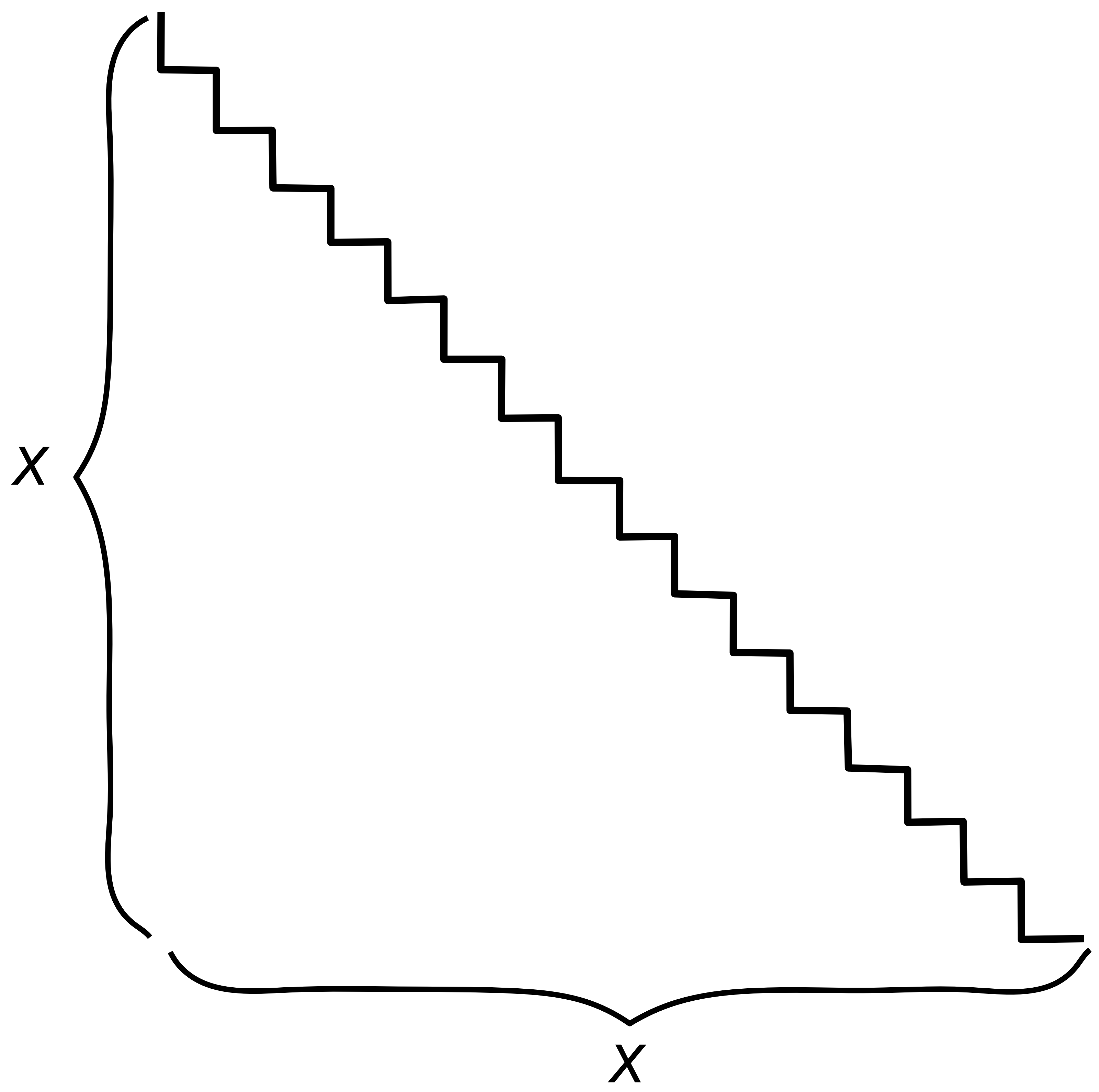
Reasoning about infinity



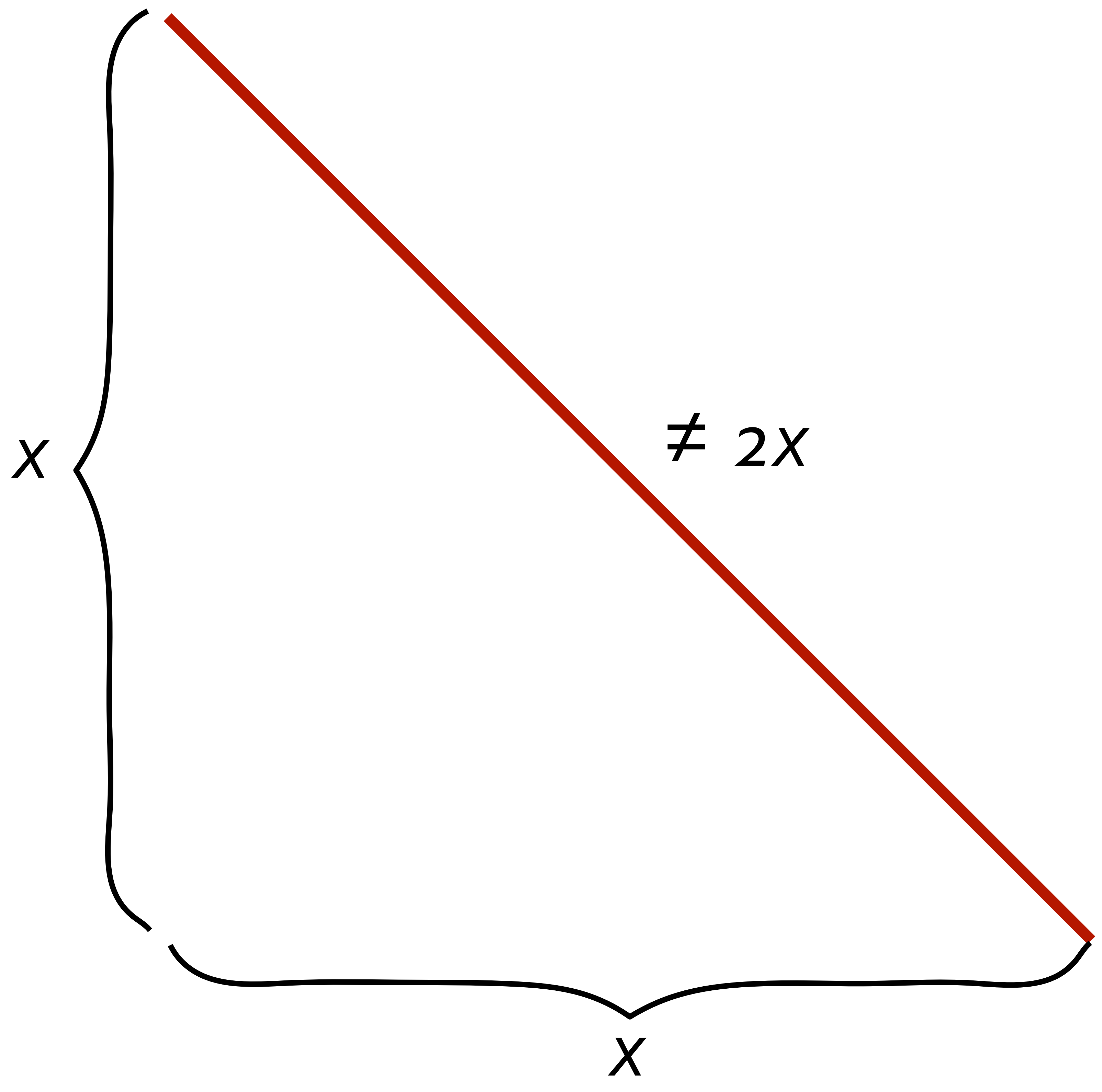
Reasoning about infinity



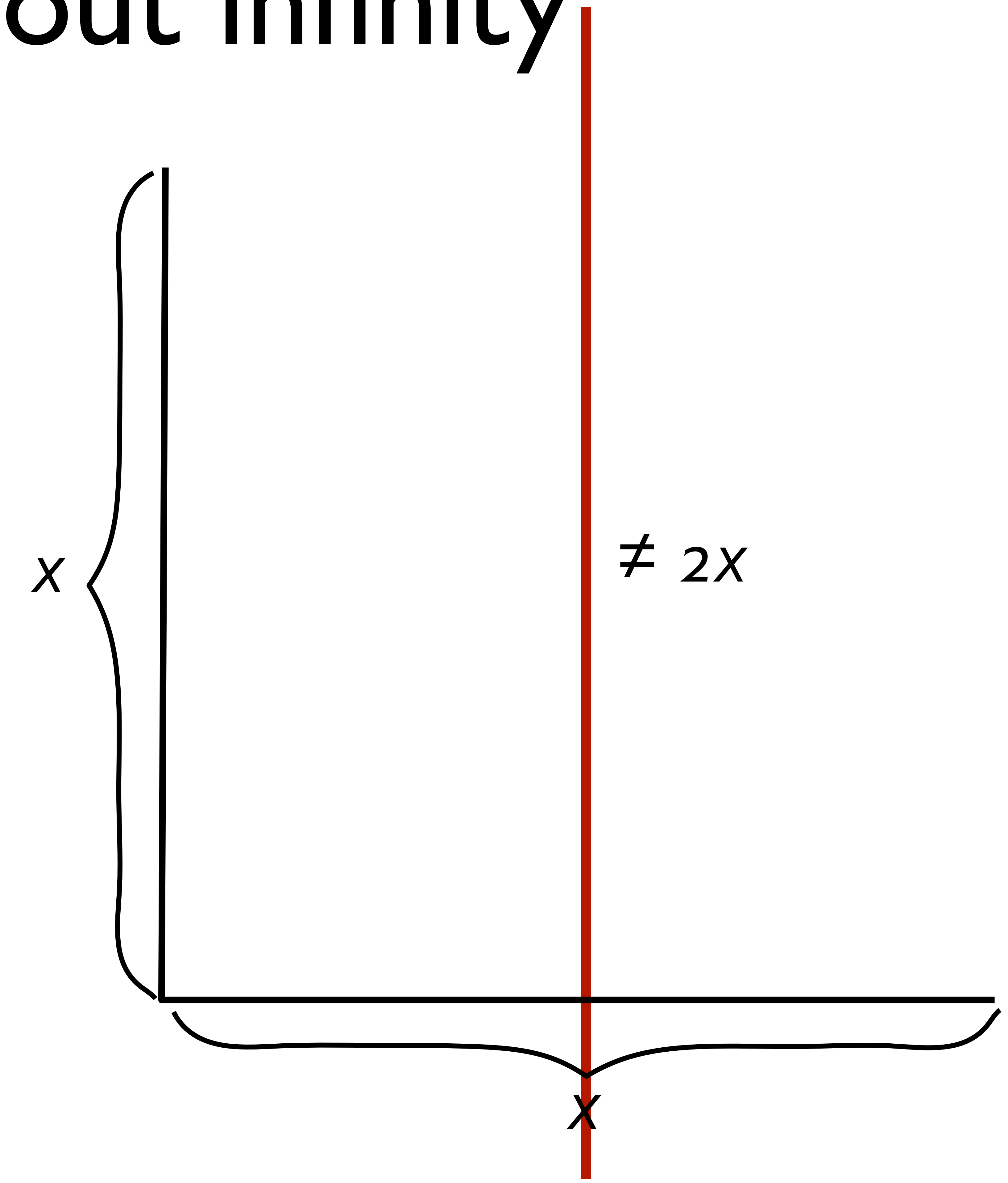
Reasoning about infinity



Reasoning about infinity



Reasoning about infinity



Reasoning about infinity

$$0.9 < 1$$

Reasoning about infinity

$$0.99 < 1$$

Reasoning about infinity

$$0.999 < 1$$

Reasoning about infinity

$$0.9999 < 1$$

Reasoning about infinity

$$0.99999 < 1$$

Reasoning about infinity

$$0.\overline{9} \nless 1$$

Reasoning about infinity

Strange but true!

$$0.\overline{9} = 1$$

$$x = 0.\overline{9}$$

$$10x = 9.\overline{9}$$ *Multiply both sides by 10*

$$9x = 9$$ *Subtract x from both sides*

$$x = 1$$ *Divide both sides by 9*

Reasoning about infinity

o is finite

Reasoning about infinity

1 is finite

Reasoning about infinity

2 is finite

Reasoning about infinity

3 is finite

Reasoning about infinity

4 is finite

Reasoning about infinity

∞ is ***not*** finite

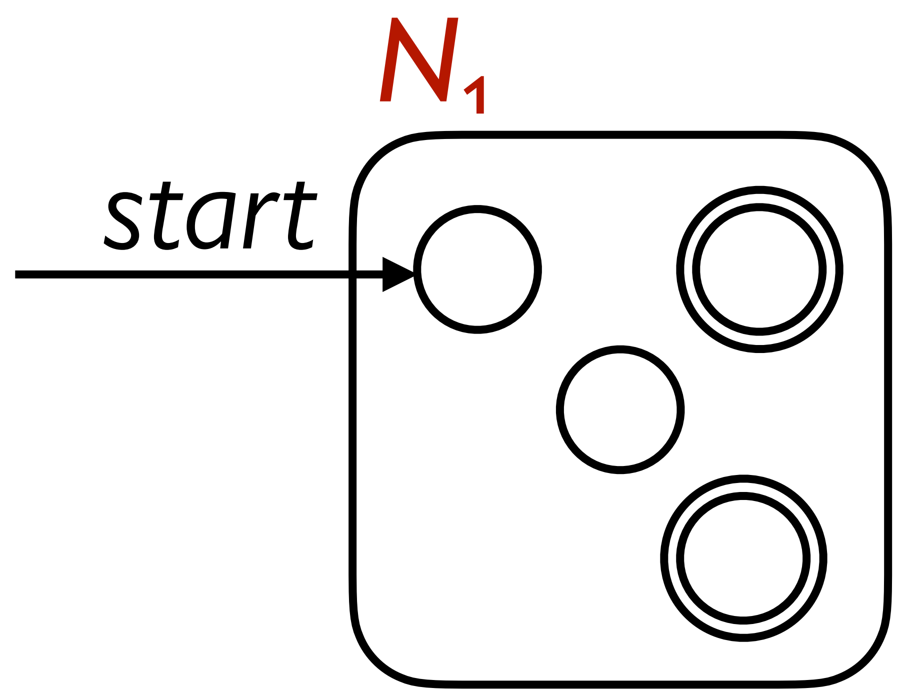
Even if a series of finite objects all have some property, the "limit" of that process *doesn't* necessarily have that property.

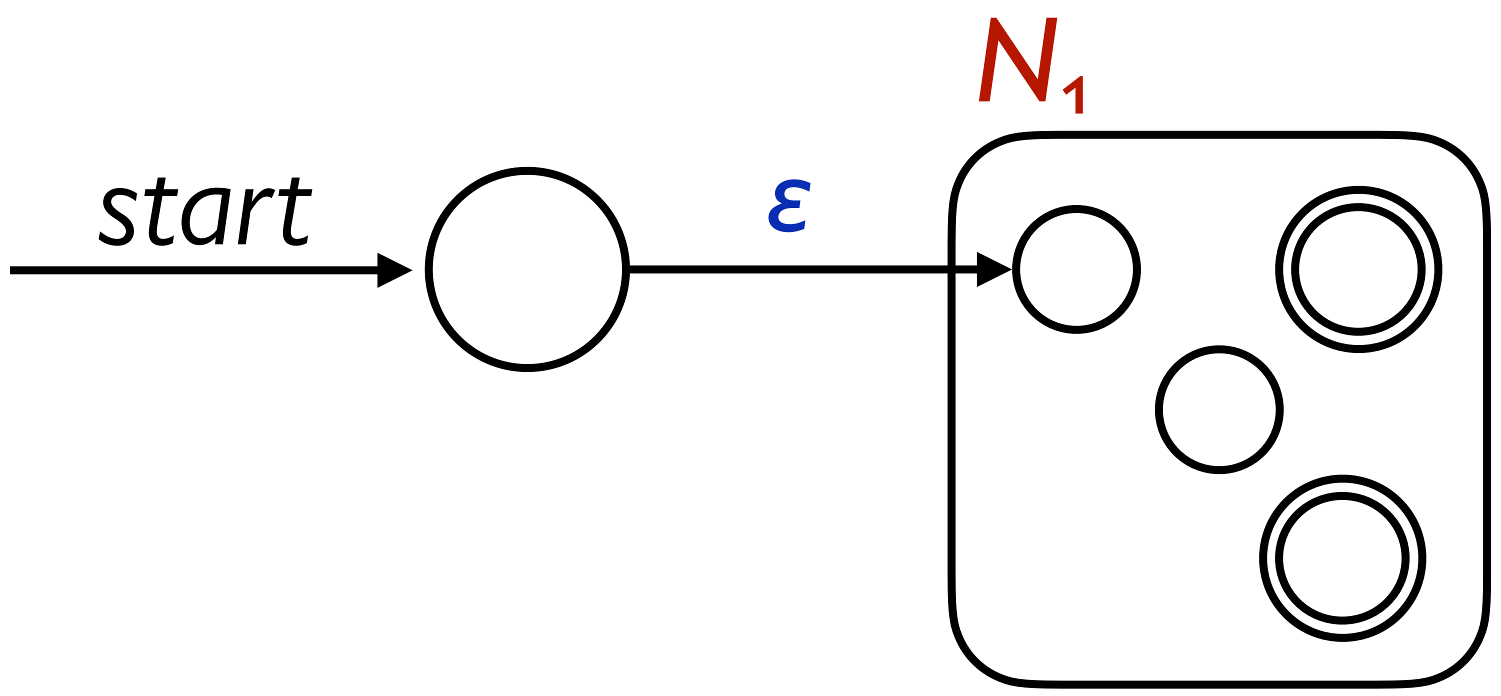
In general, it's not safe to conclude that some property that holds in the finite case must hold in the infinite case.

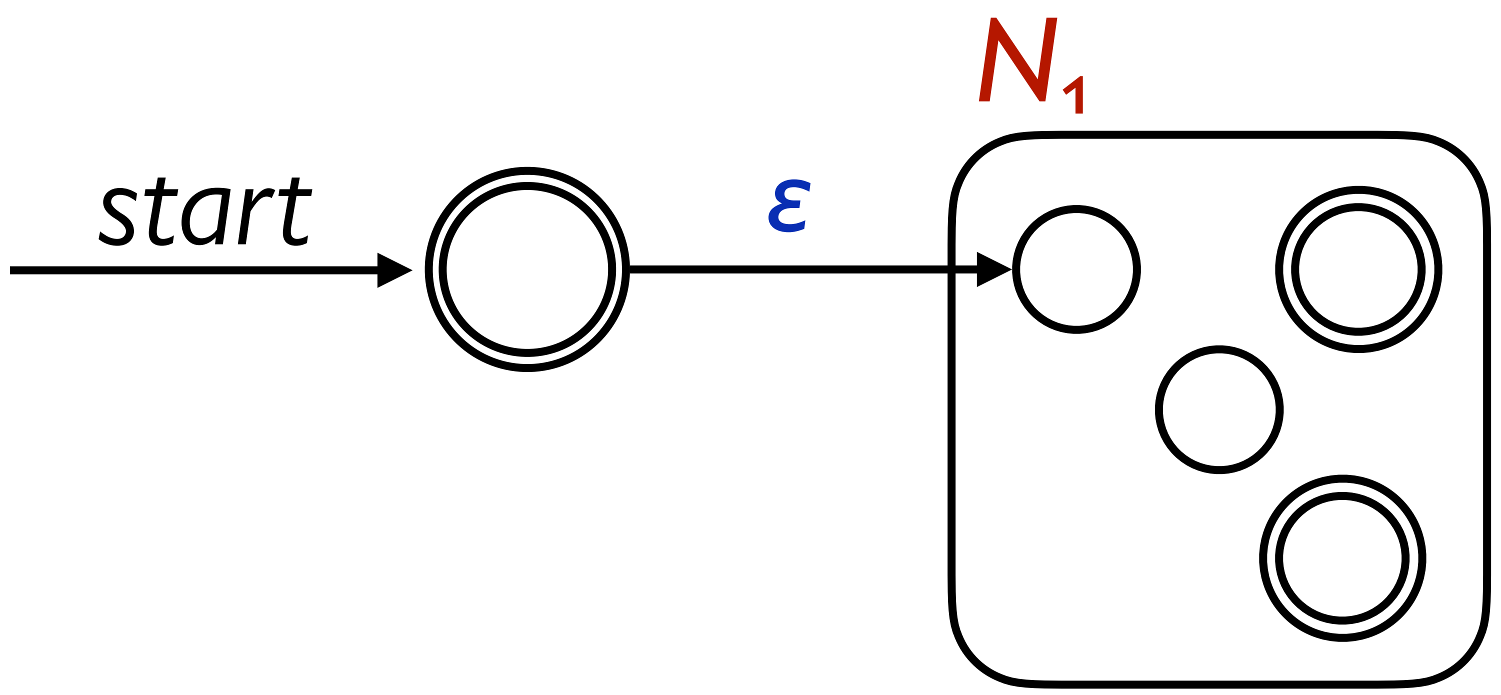
So, an argument based on $L^* = L^0 \cup L^1 \cup \cdots$ isn't going to work.

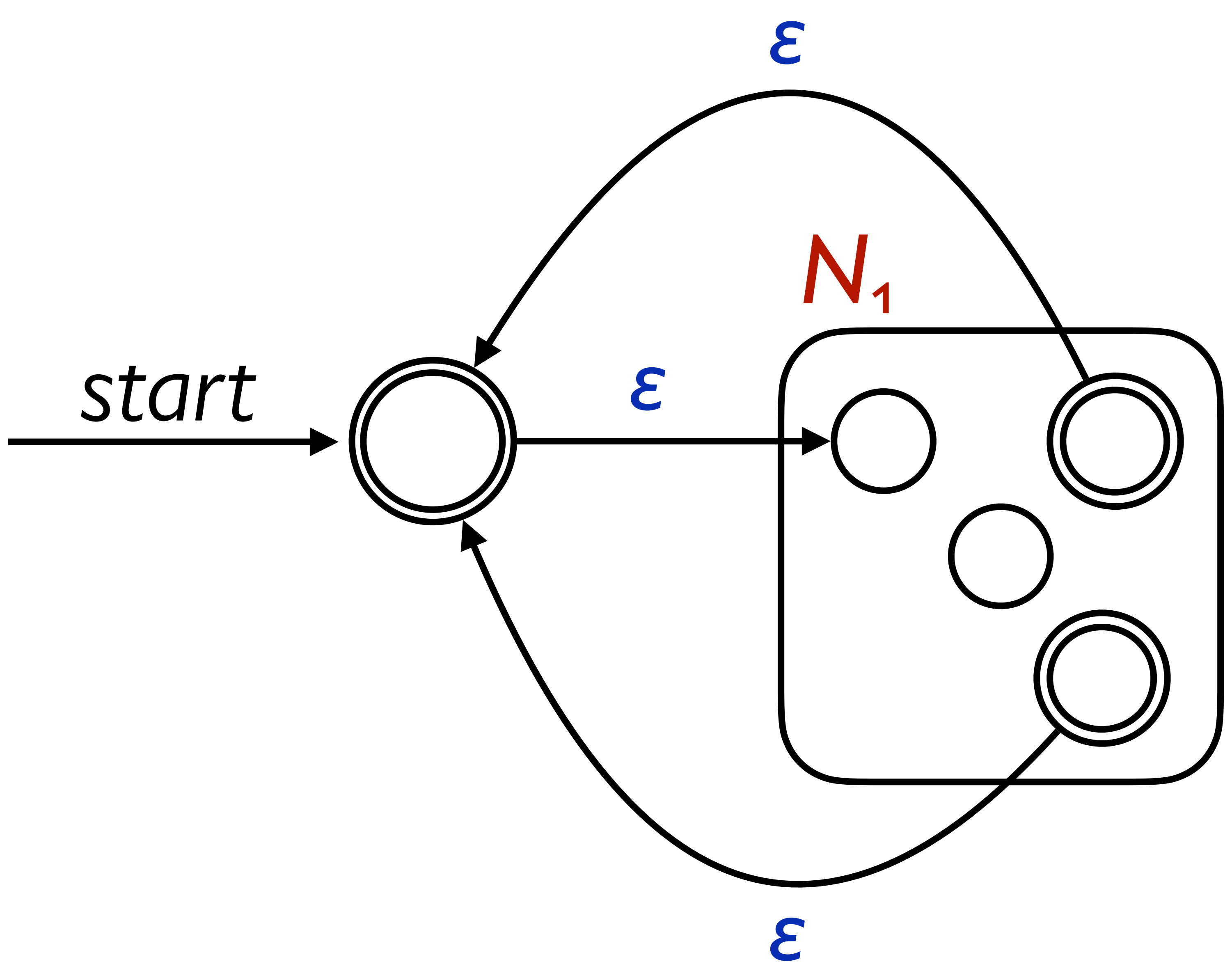
We need a different line of reasoning.

Can we convert an NFA for a language L into an NFA for L^*?

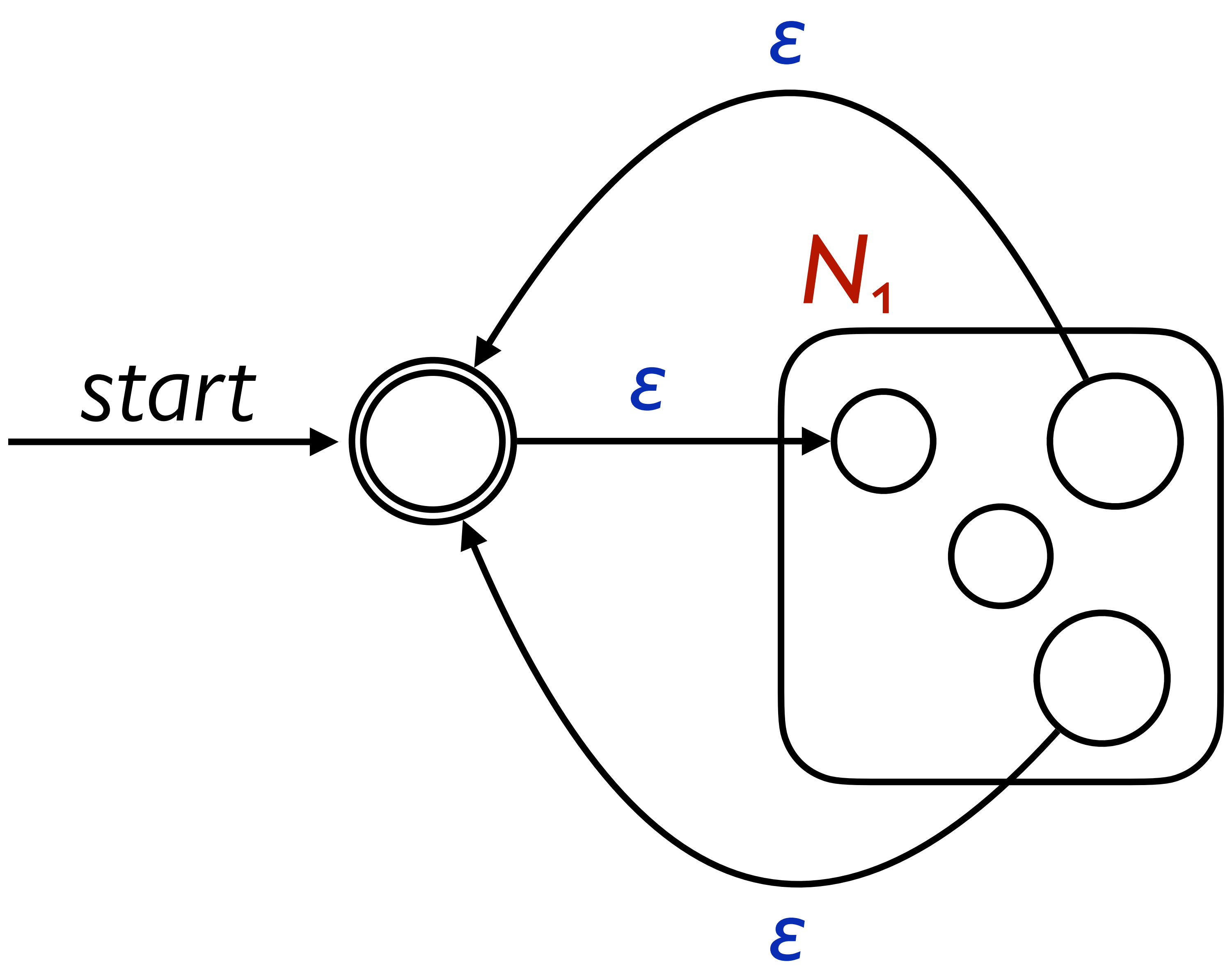

N1
start


start
ε
N_1

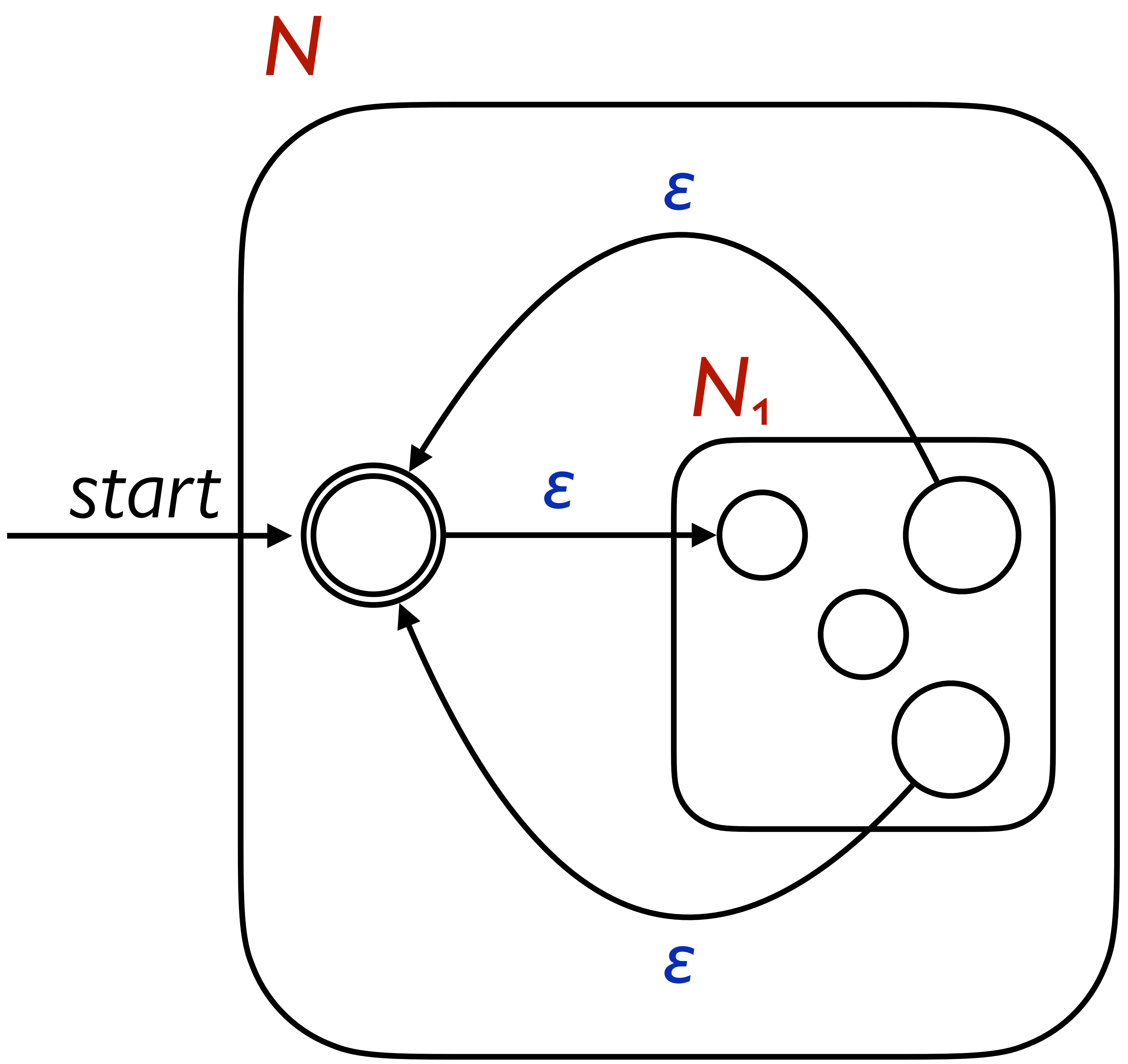

start
ε
N_1



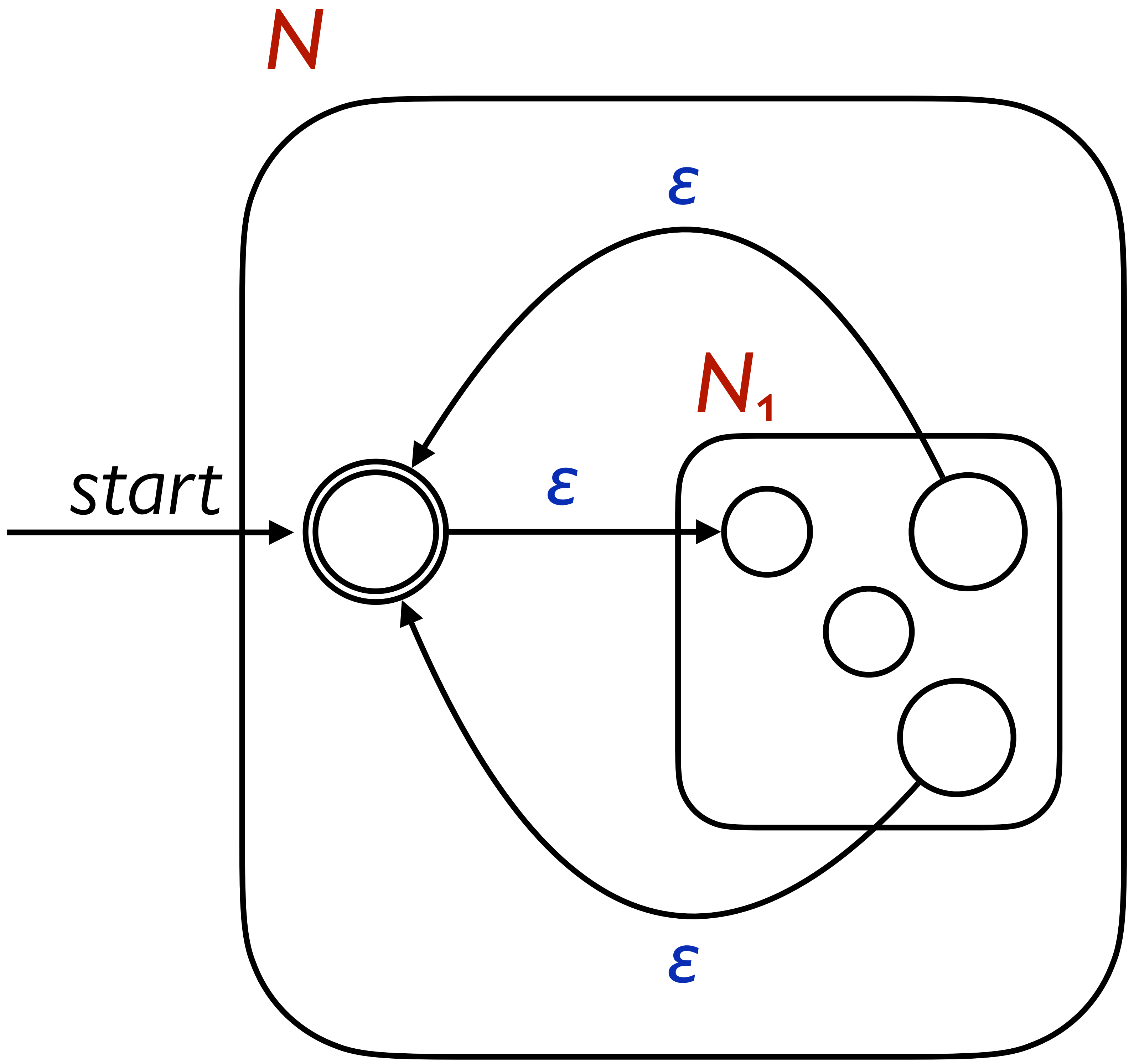
The new machine has the option of jumping back to the start state to read another piece that N_1 accepts.



The new machine has the option of jumping back to the start state to read another piece that N_1 accepts.



$$L(N) = L(N_1)^*$$



$L(N) = L(N_1)^*$

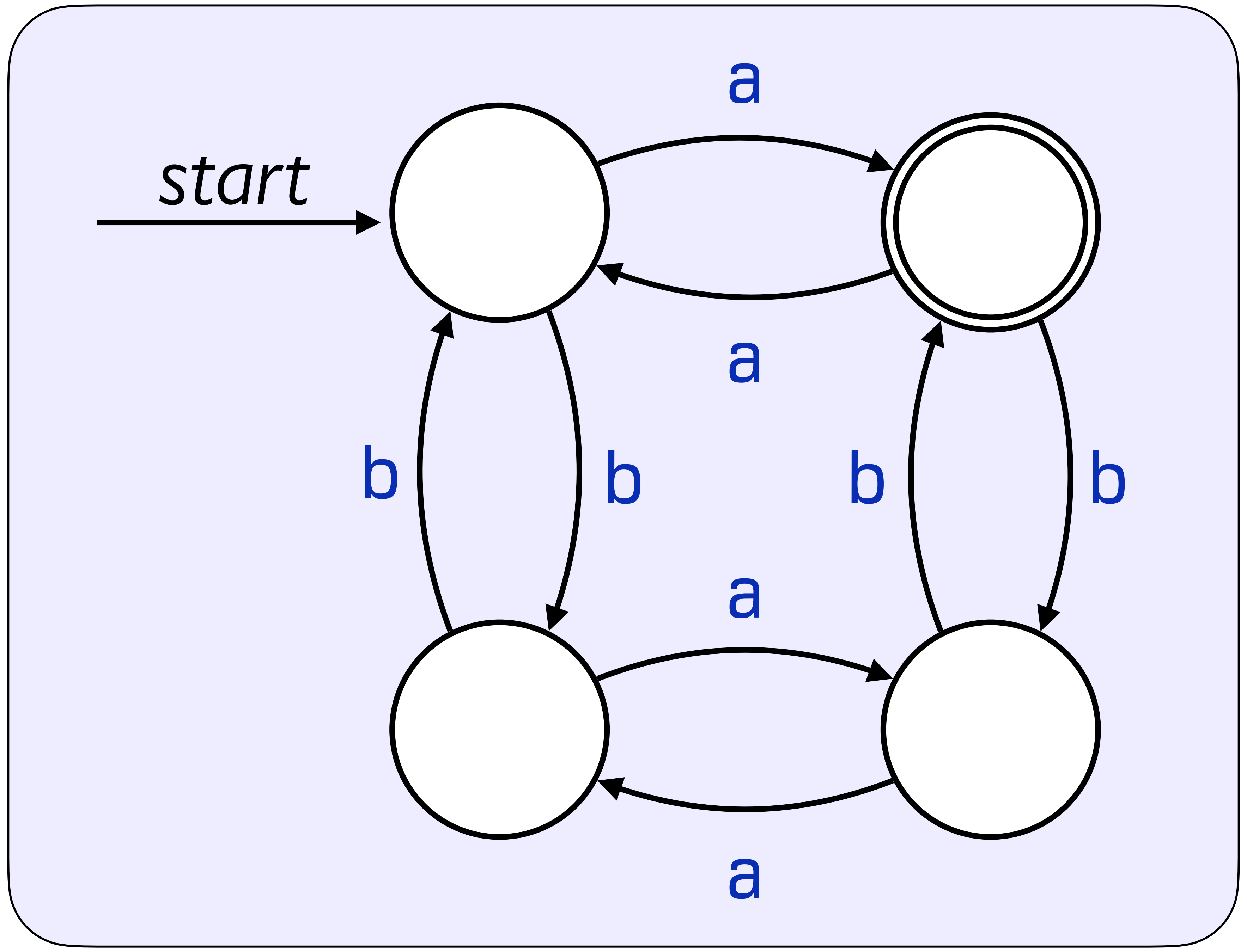
Why add a new start state instead of making N_1's a final state?

This construction proves the class of regular languages is closed under Kleene star.

Example

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

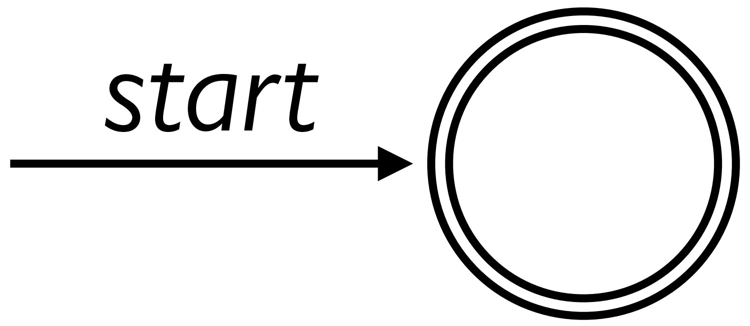
Construct an NFA for L^*.

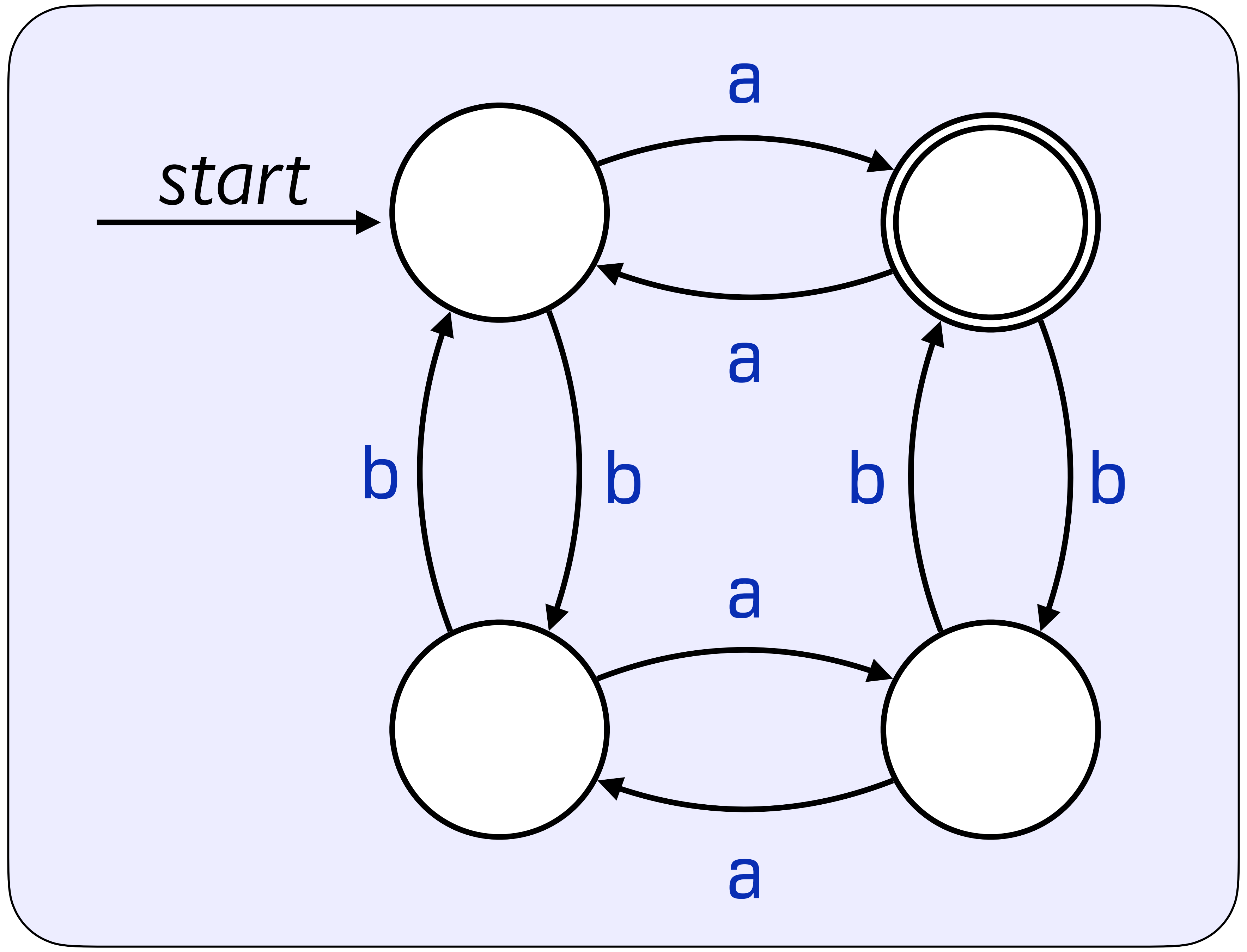


DFA for L

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

Construct an NFA for L^*.

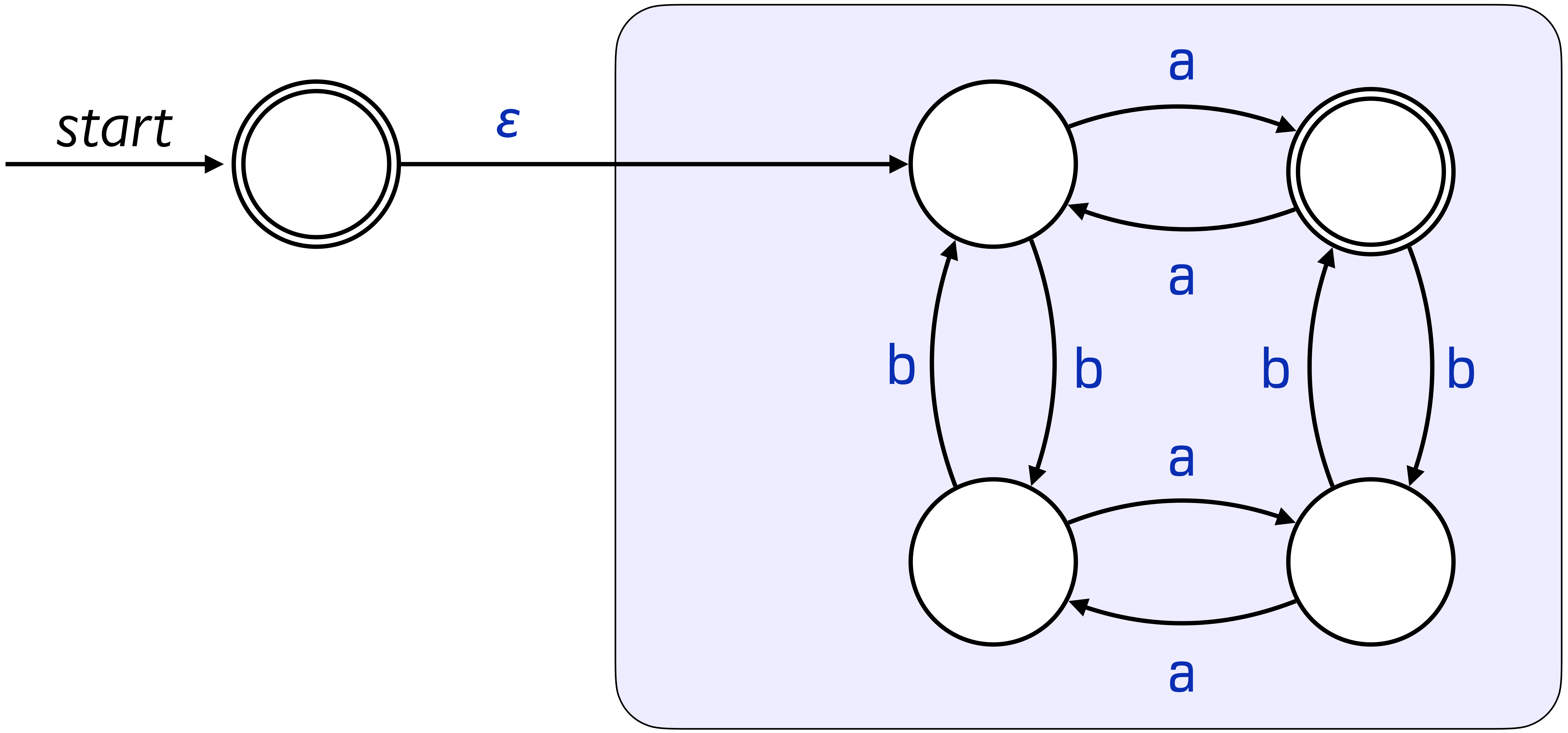




DFA for L

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

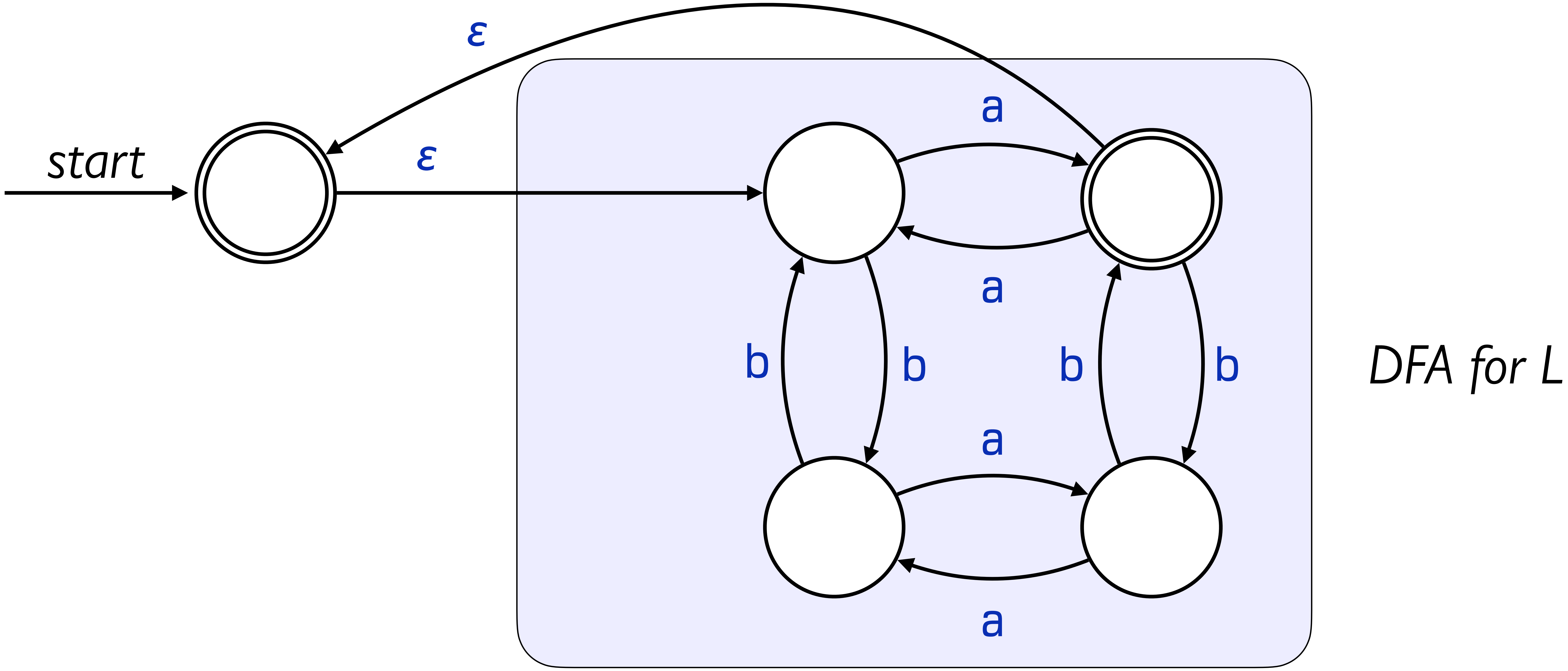
Construct an NFA for L^*.



DFA for L

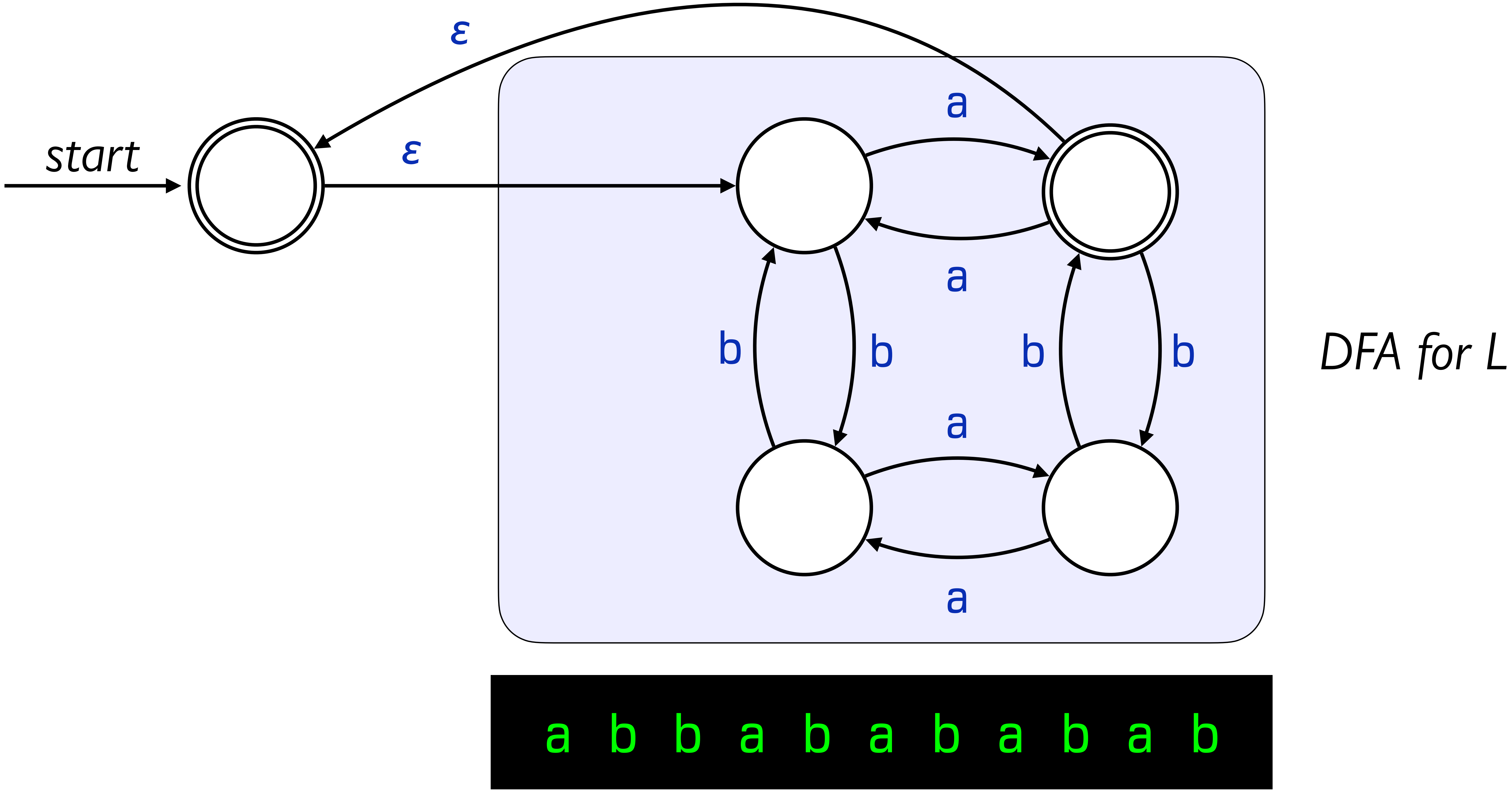
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



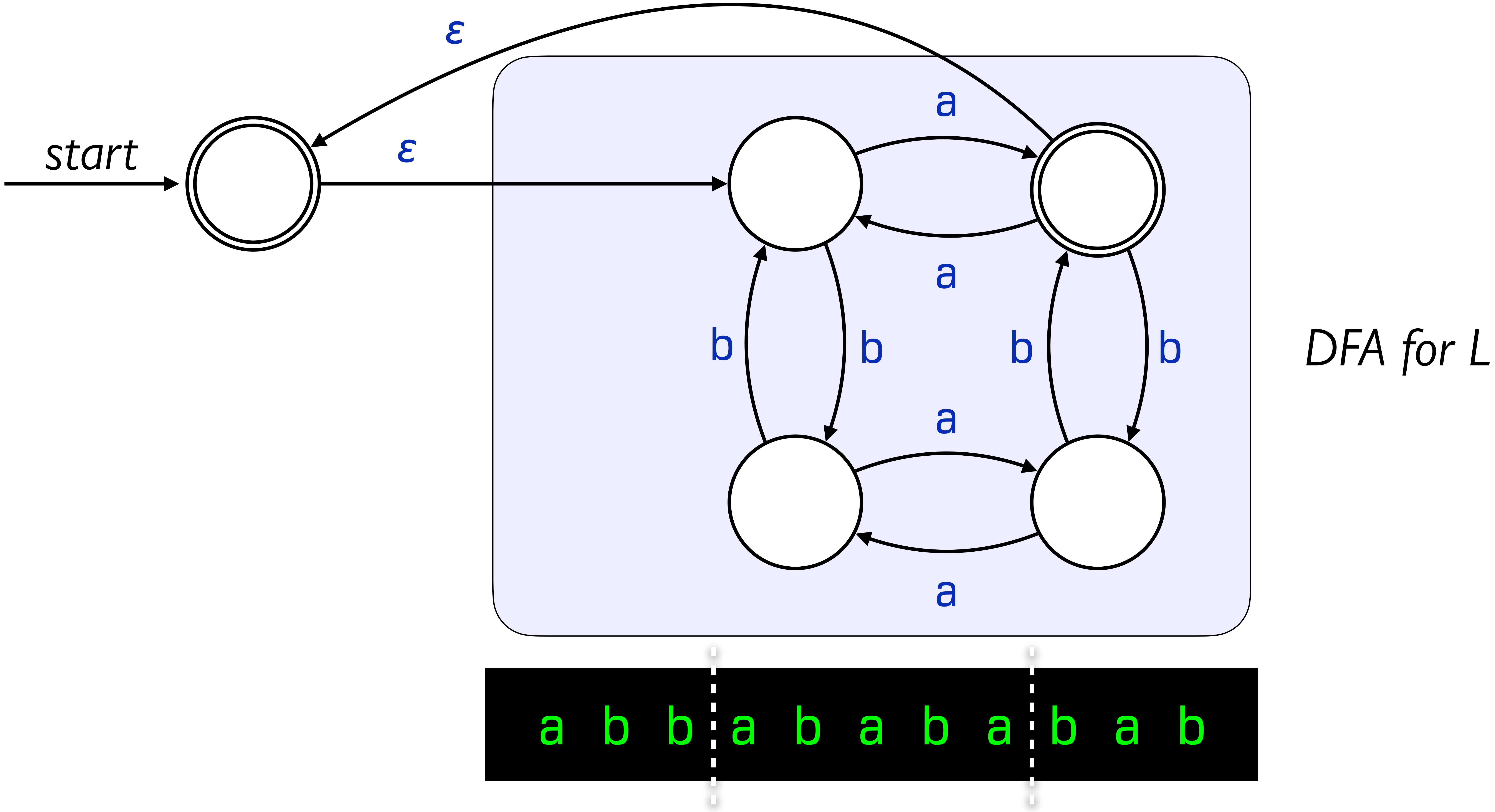
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



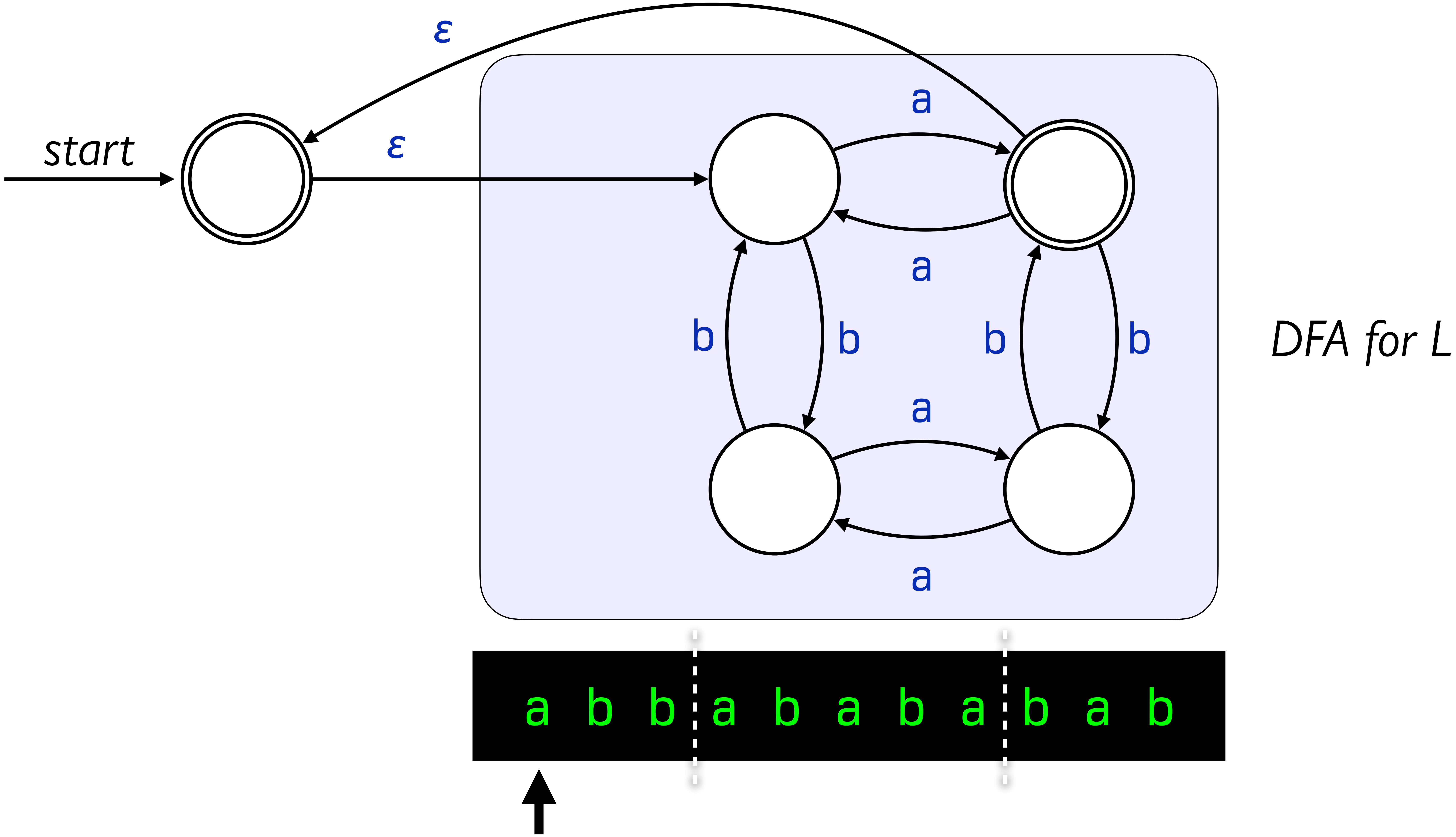
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



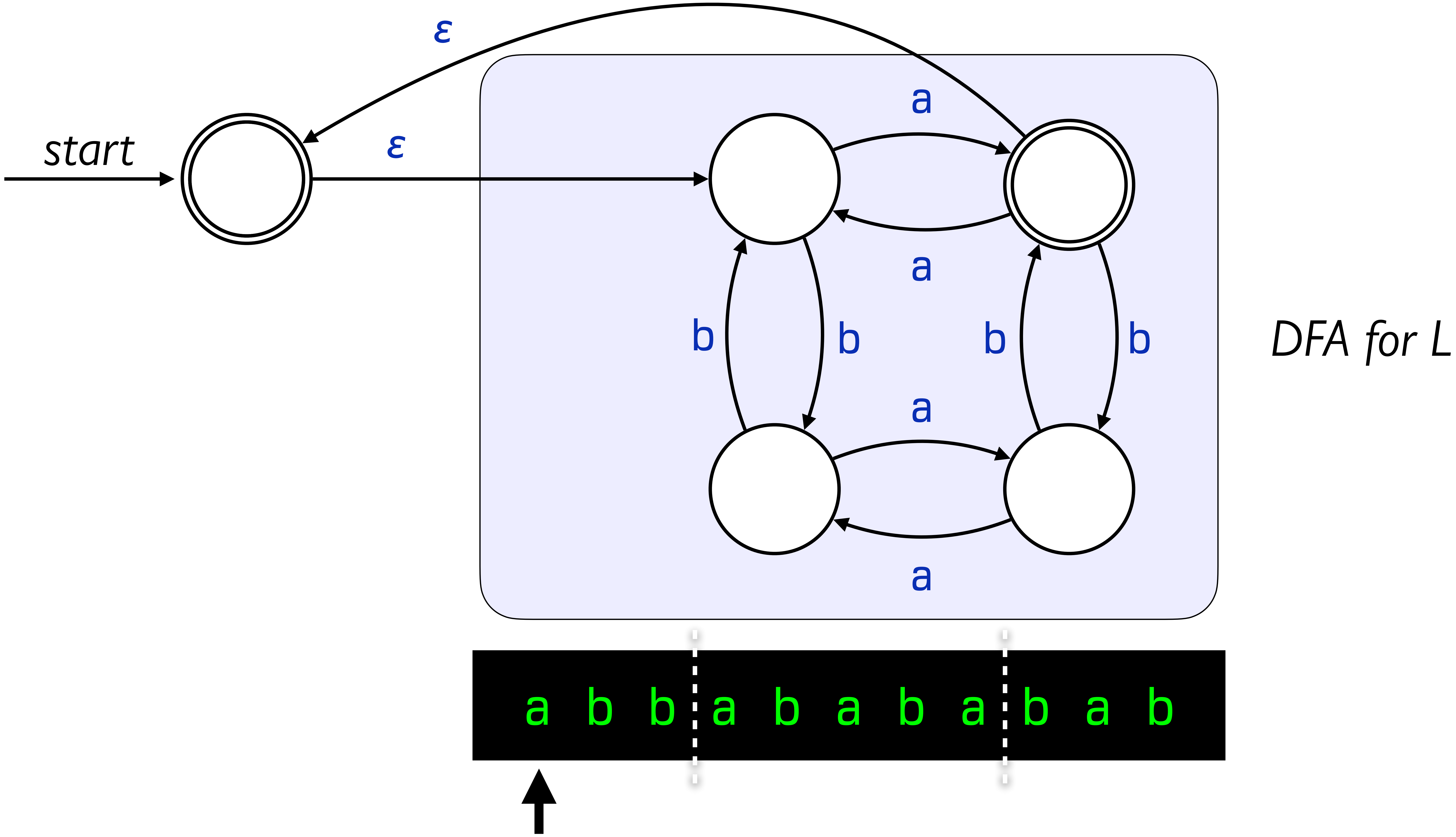
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



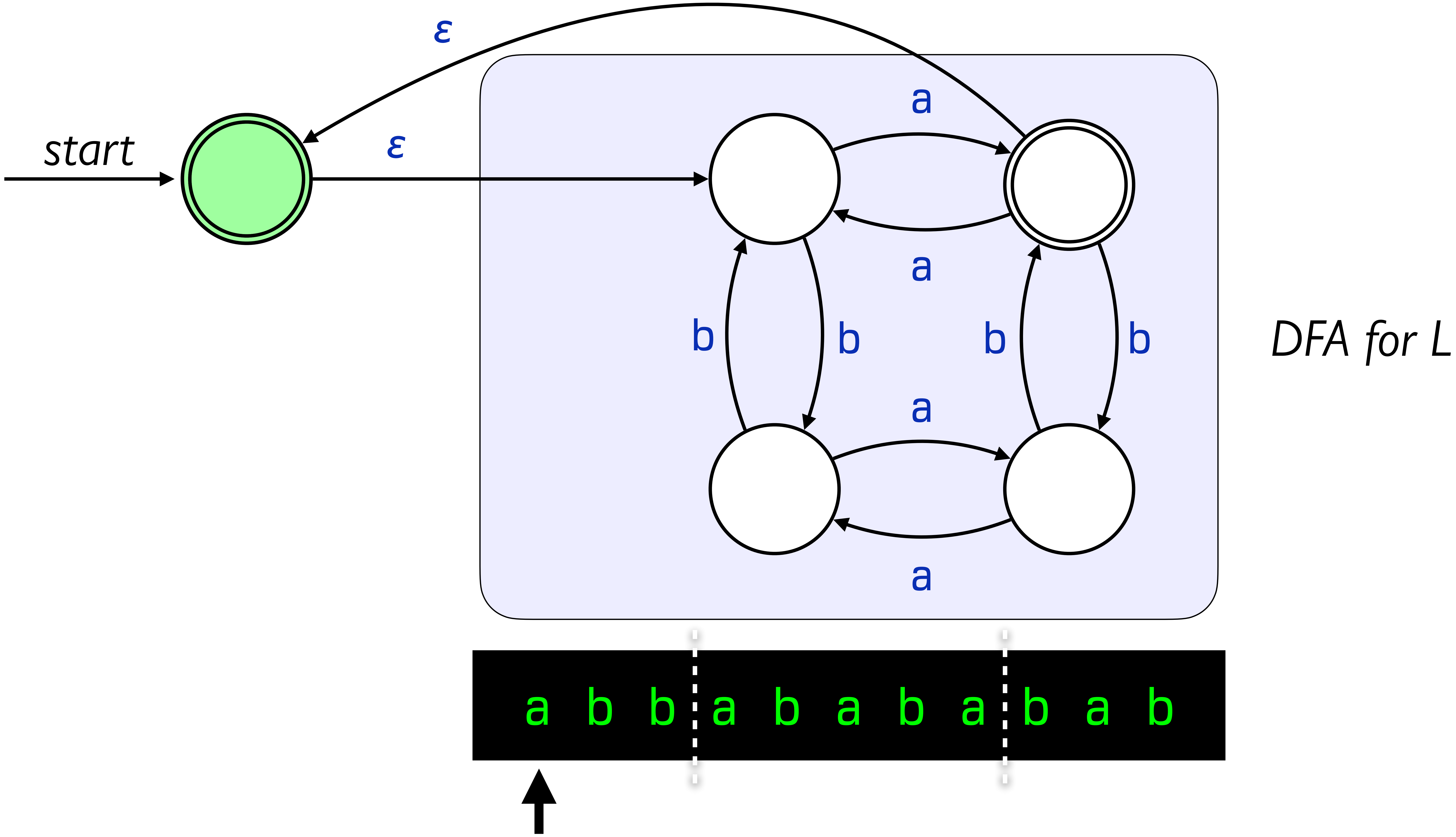
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



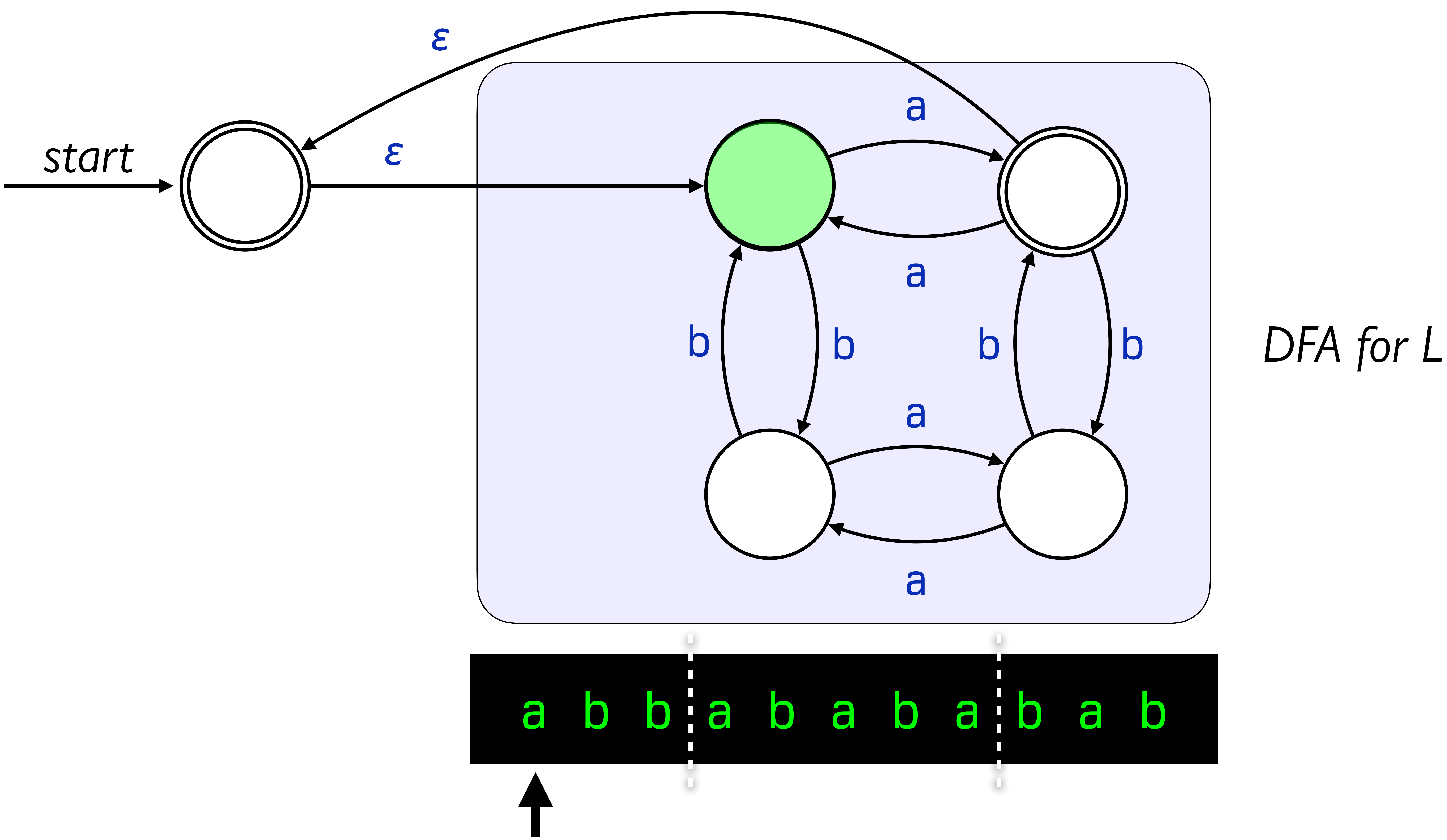
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



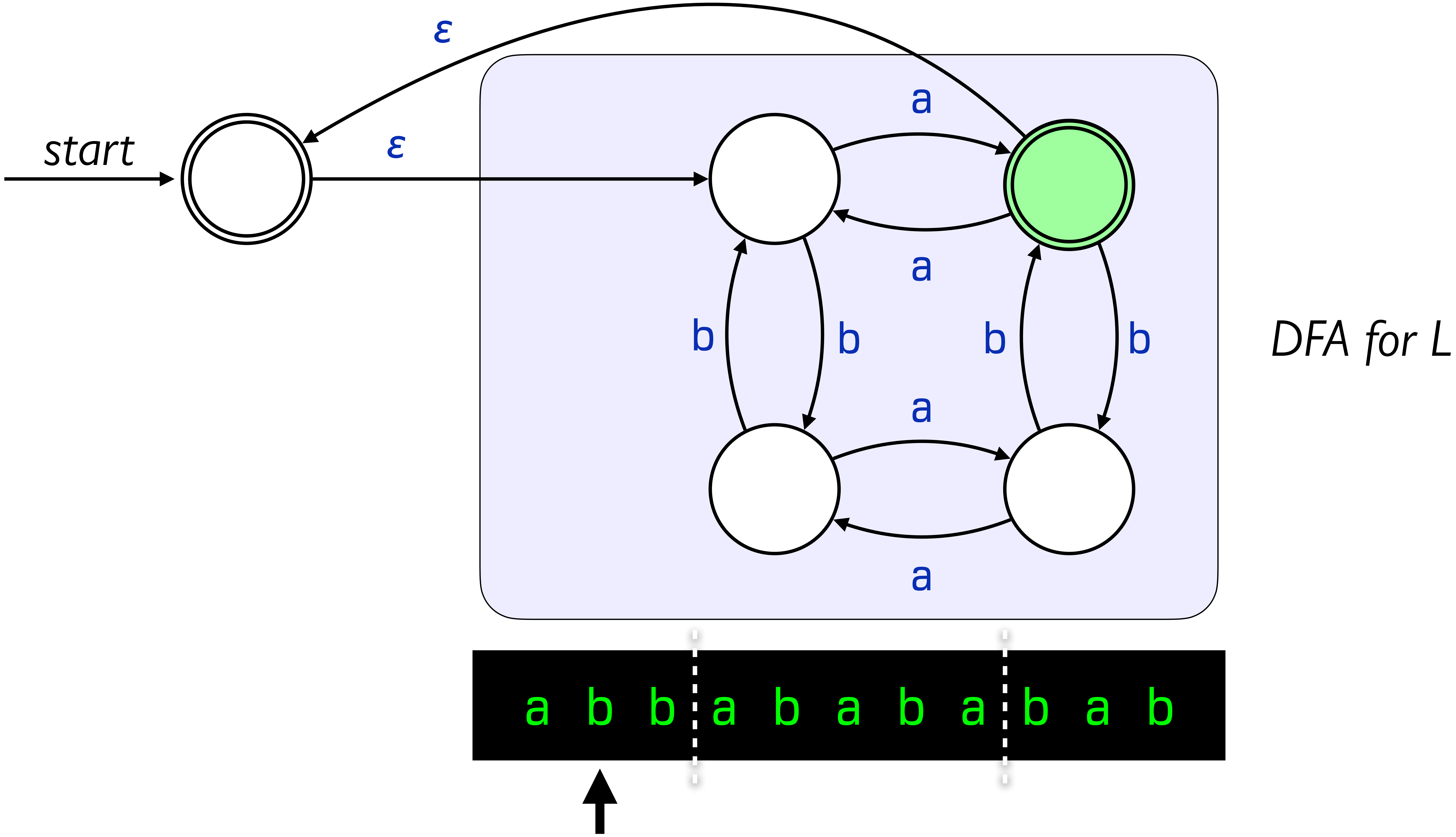
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



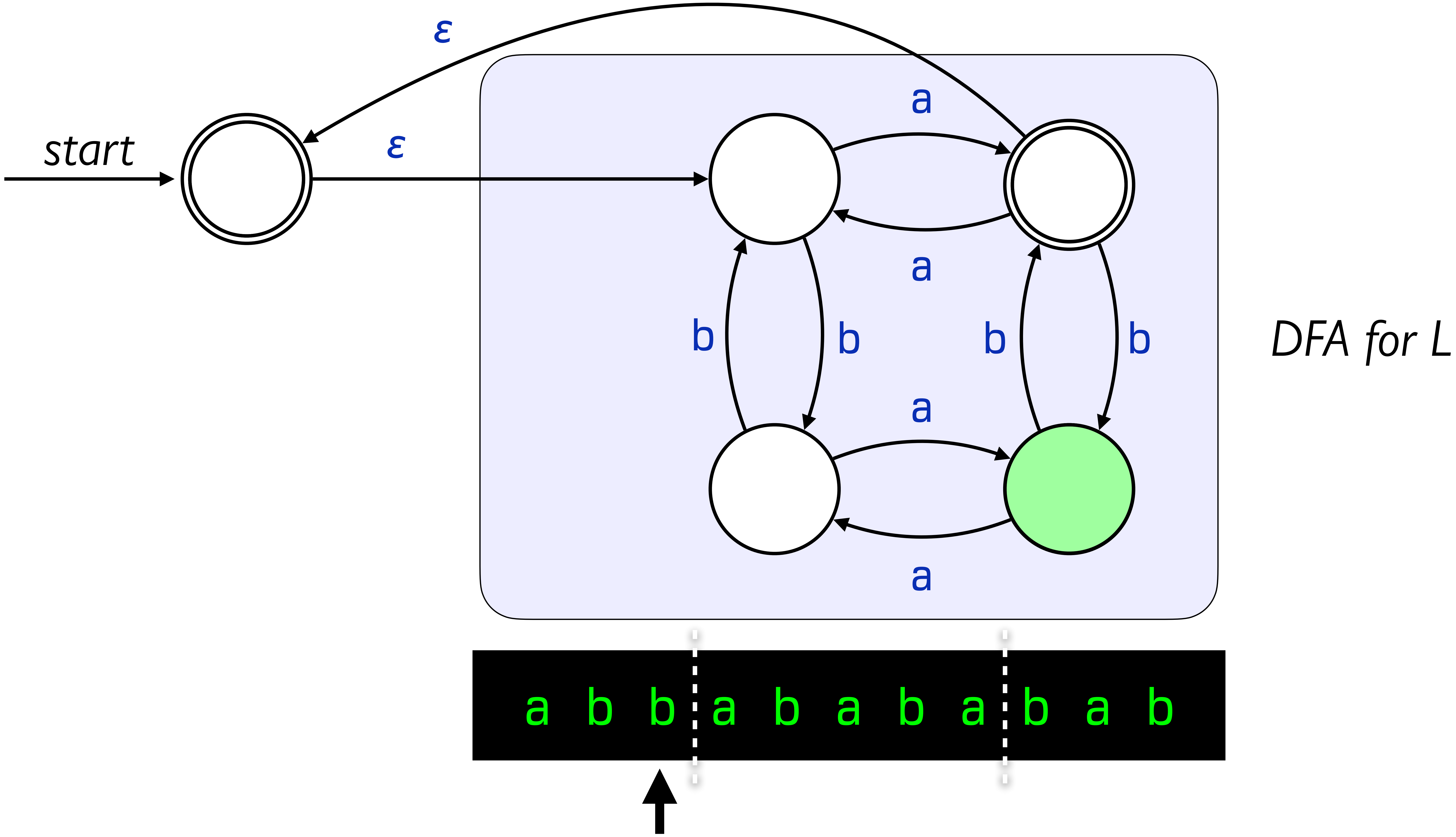
L = {w ∈ {a, b}* | w has an odd number of as and an even number of bs}

Construct an NFA for L*.



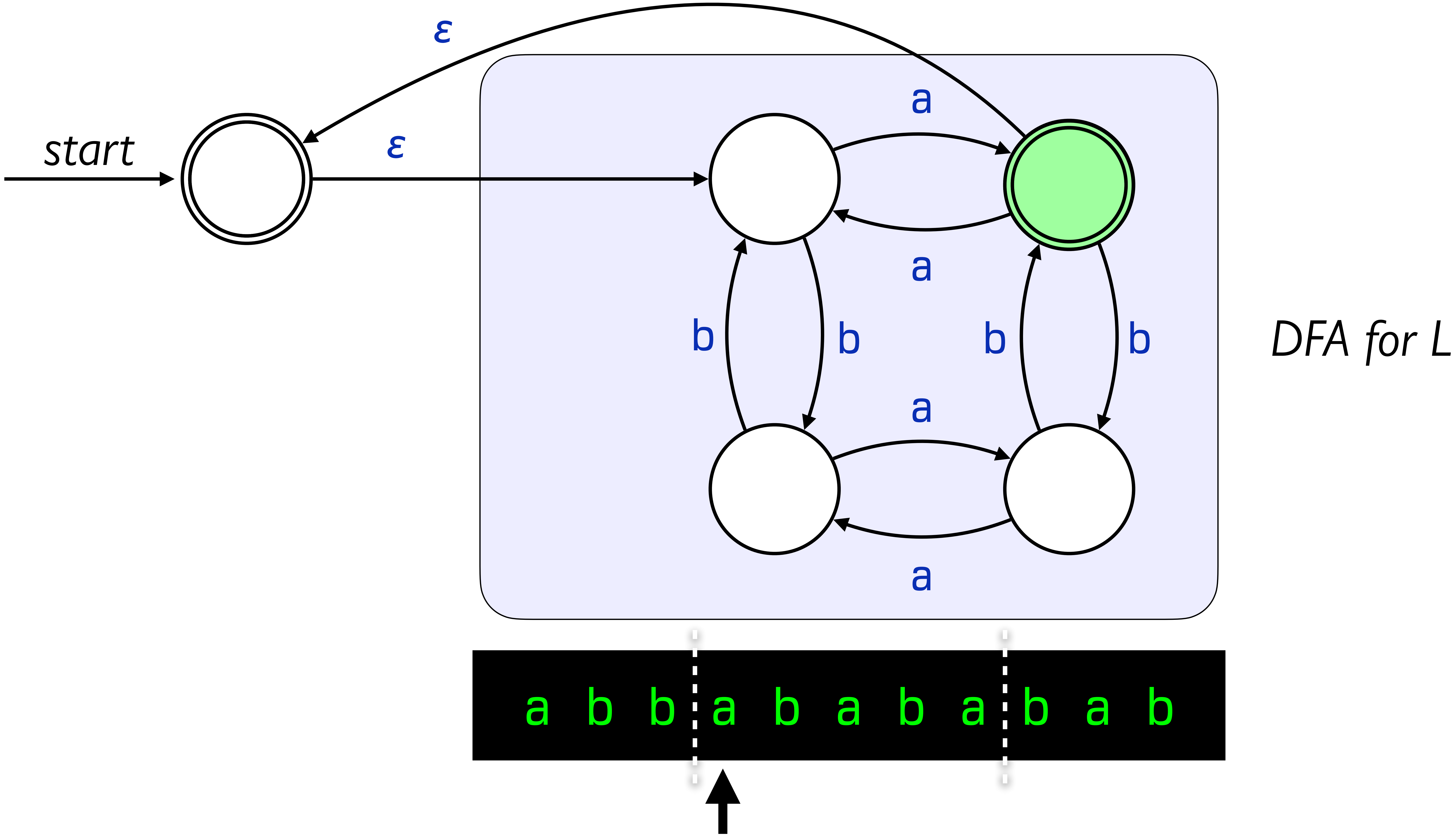
L = {$w \in$ {a, b}* | w has an odd number of as and an even number of bs}

Construct an NFA for L^*.



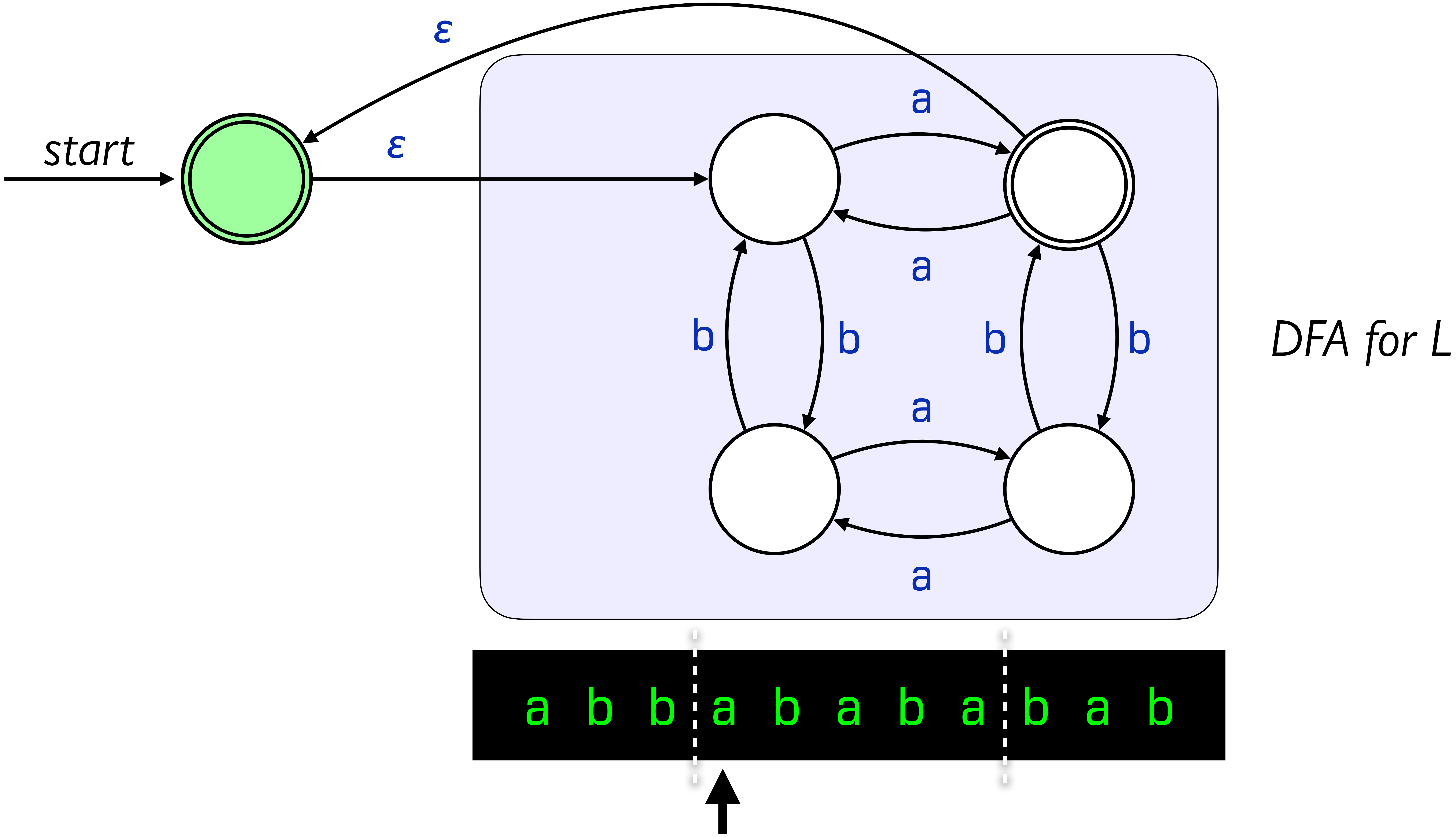
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



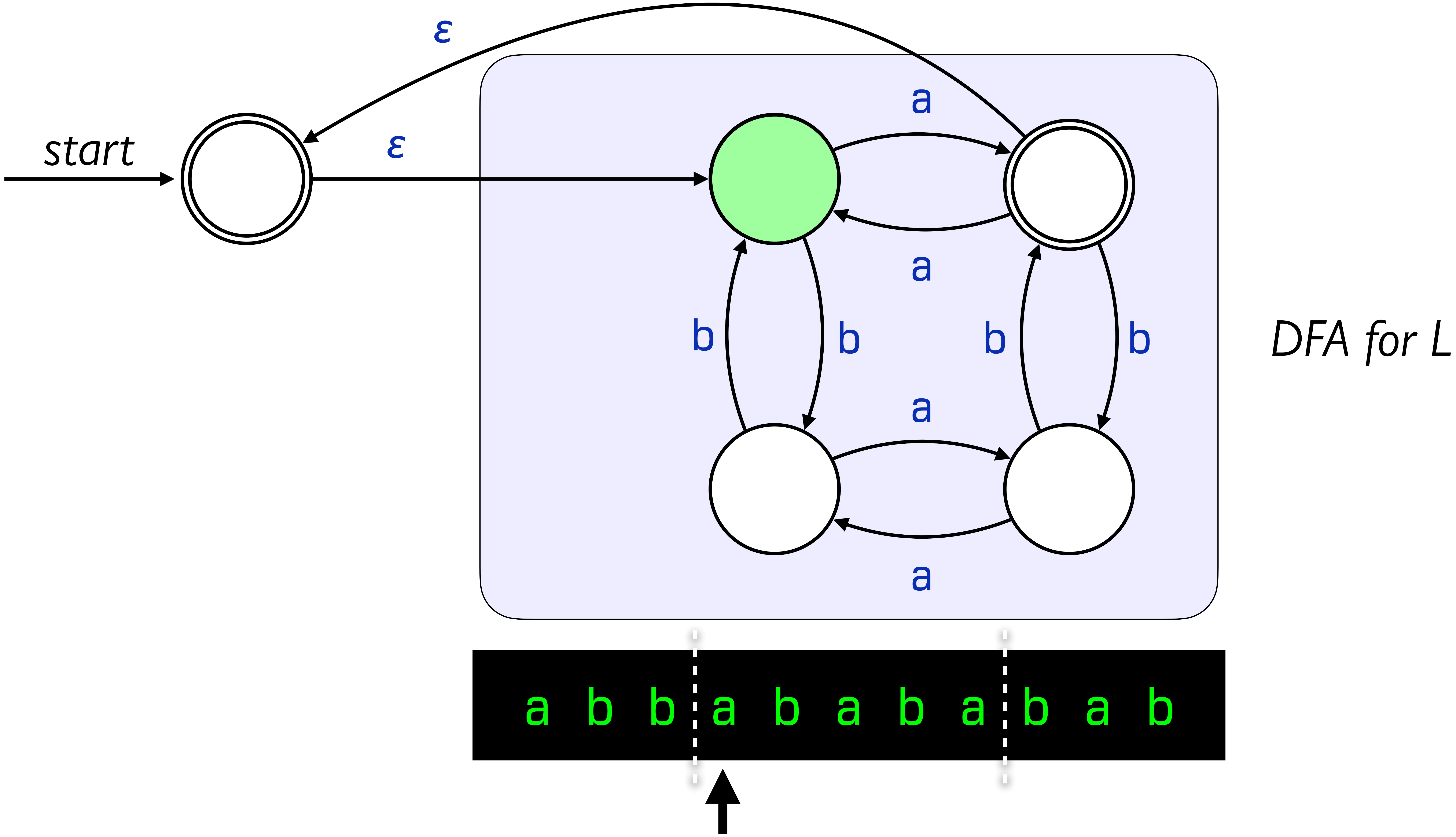
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



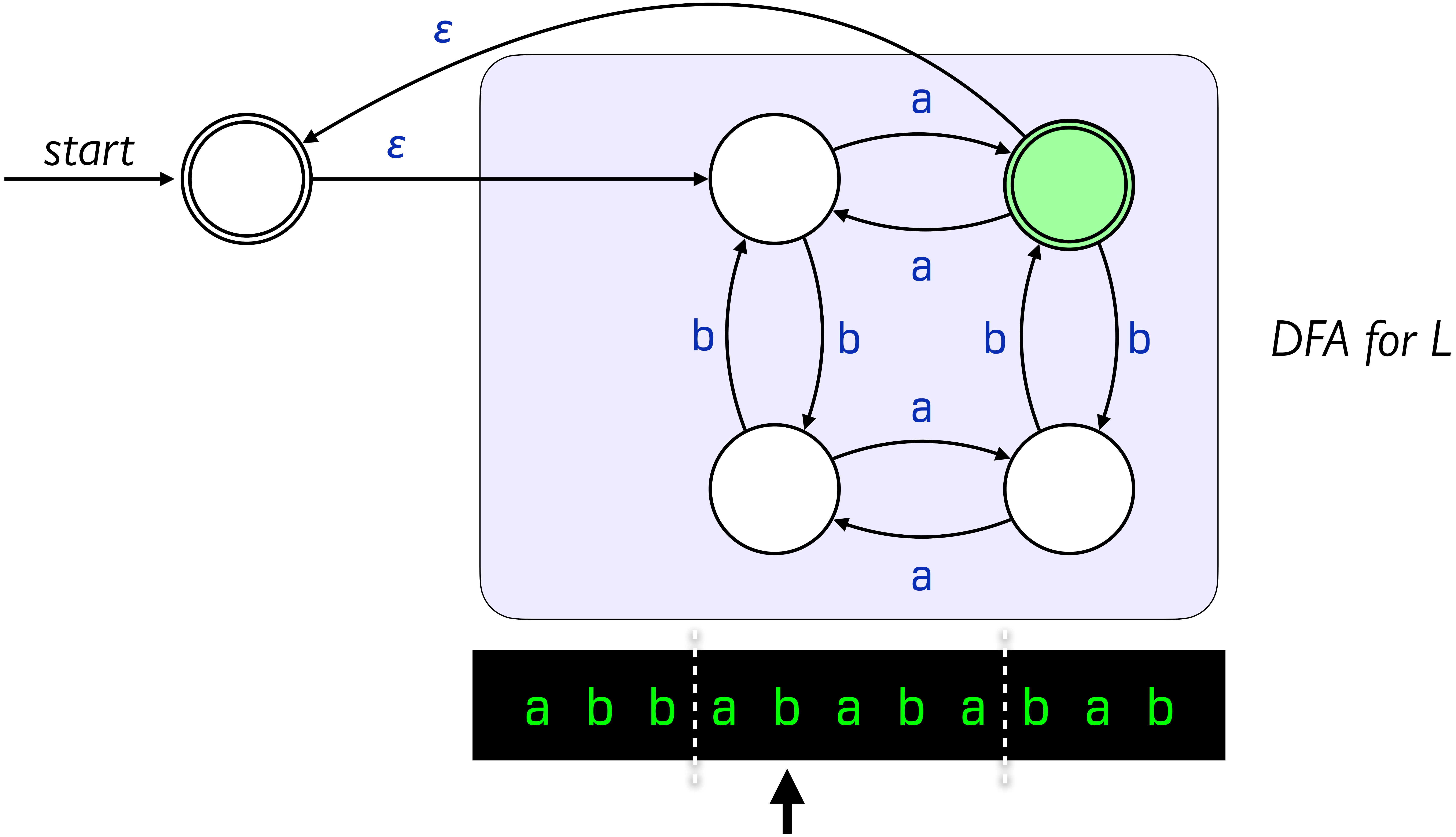
L = {w ∈ {a, b}* | w has an odd number of as and an even number of bs}

Construct an NFA for L^*.



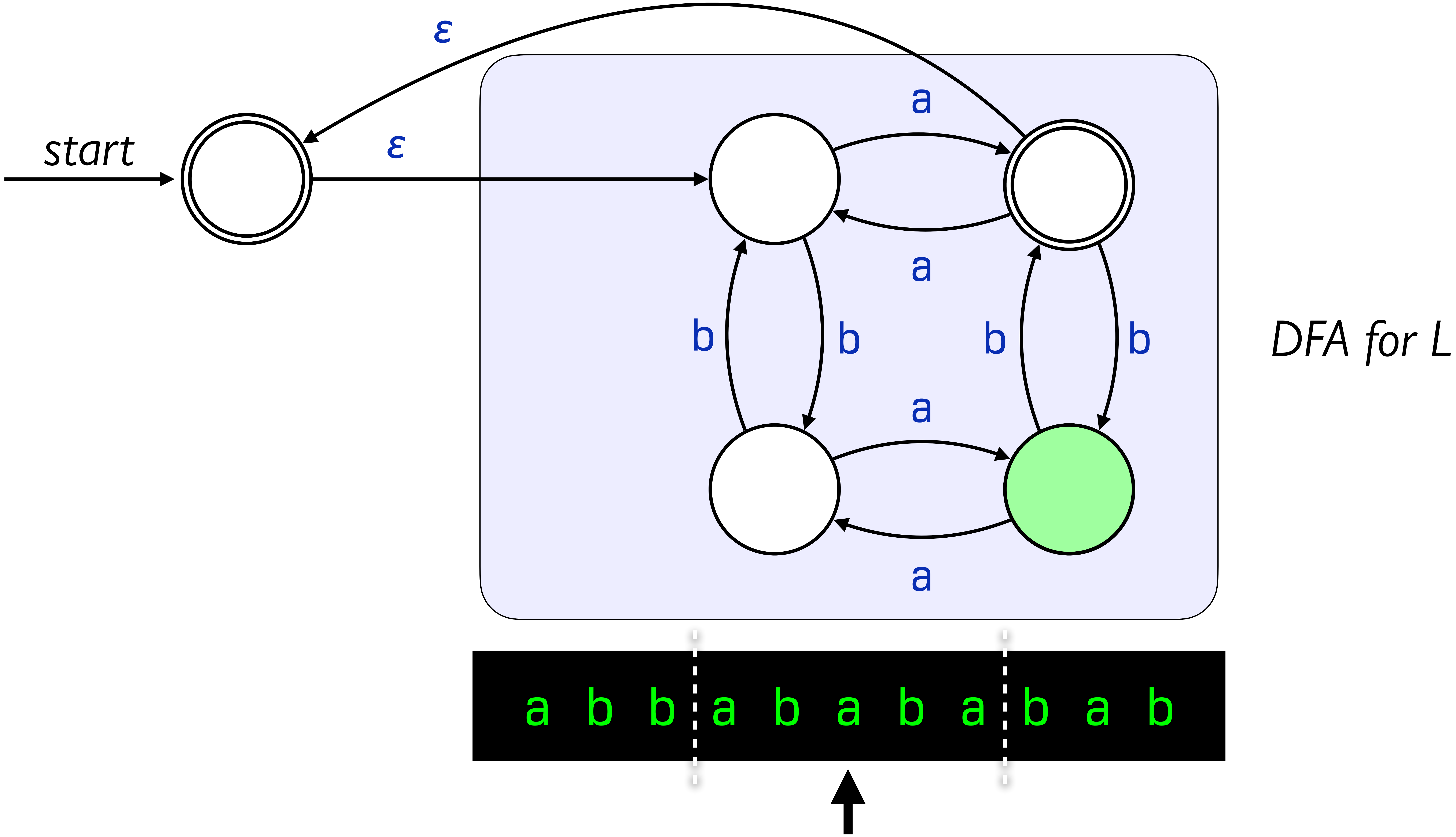
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



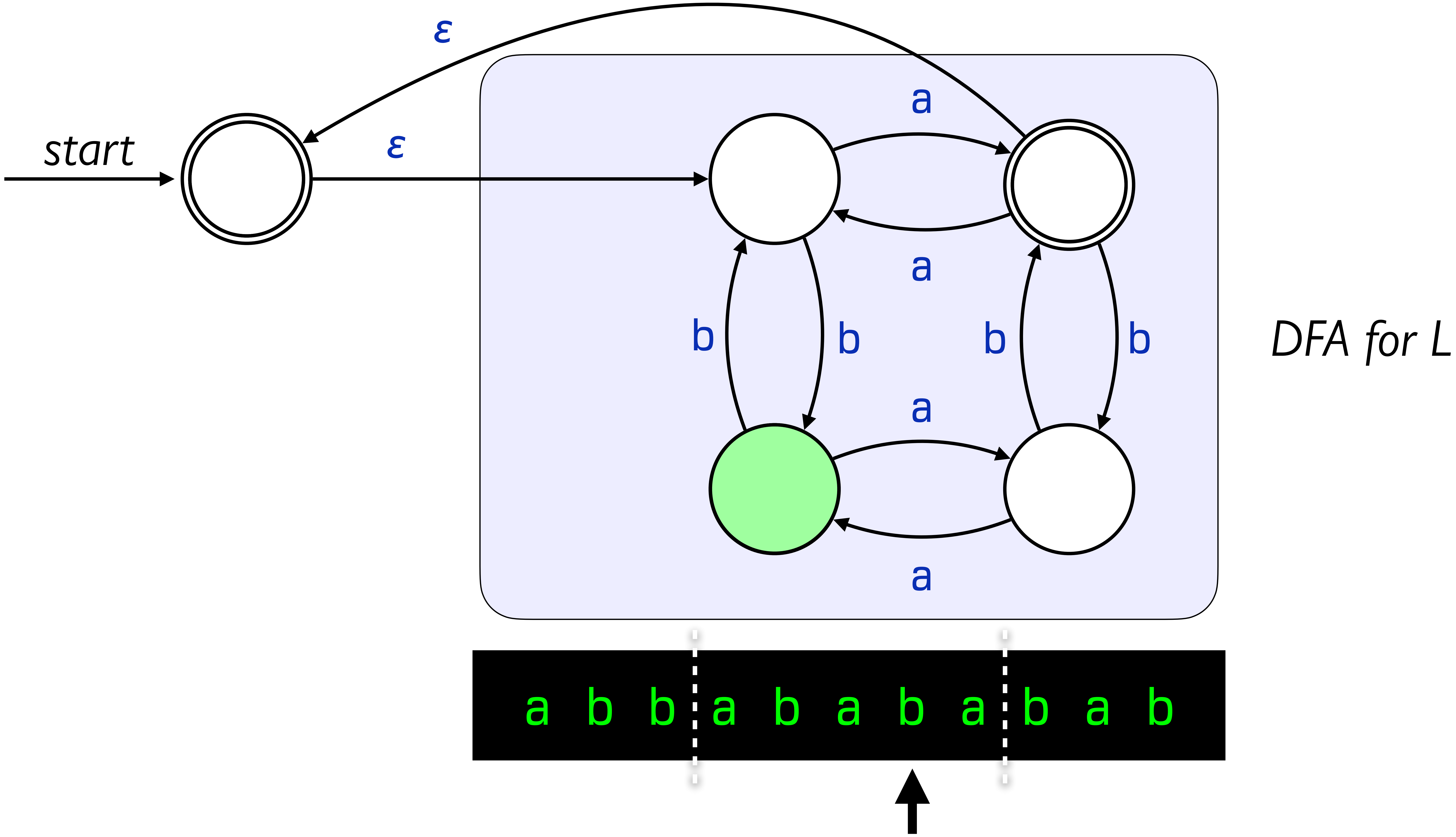
L = {$w \in$ {a, b}* | w has an odd number of as and an even number of bs}

Construct an NFA for L^*.



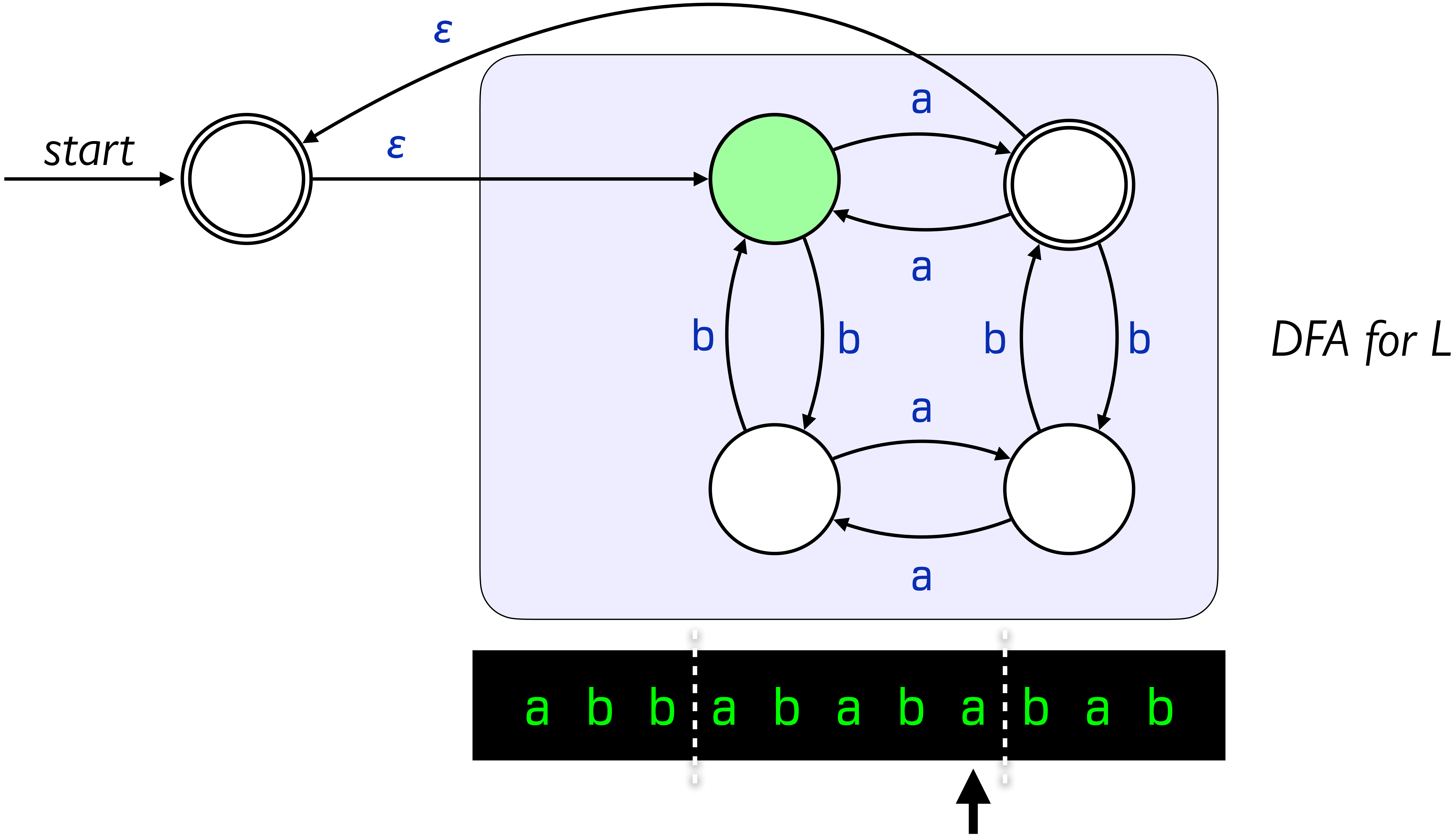
L = {$w \in$ {a, b}* | w has an odd number of as and an even number of bs}

Construct an NFA for L^*.



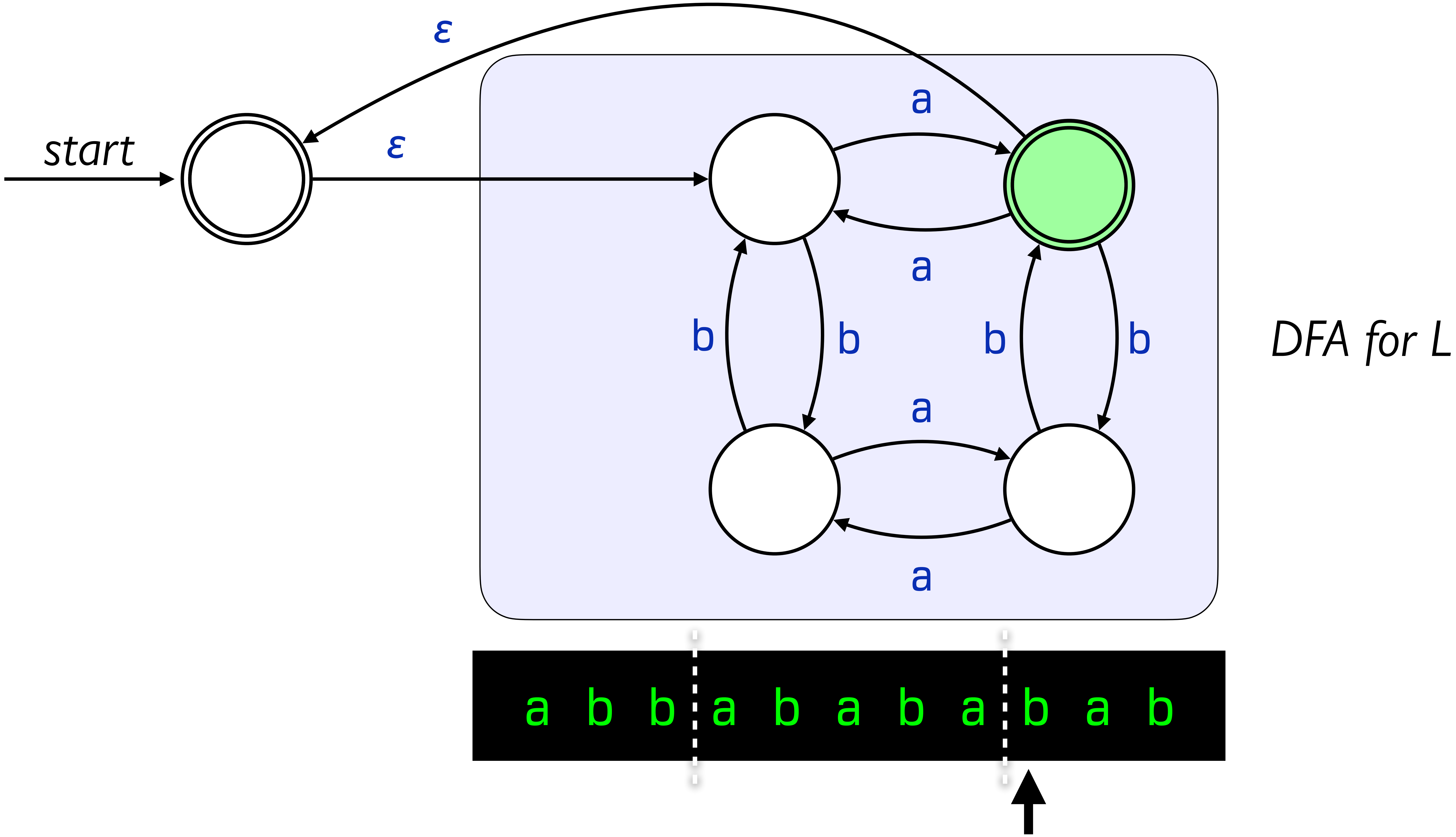
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



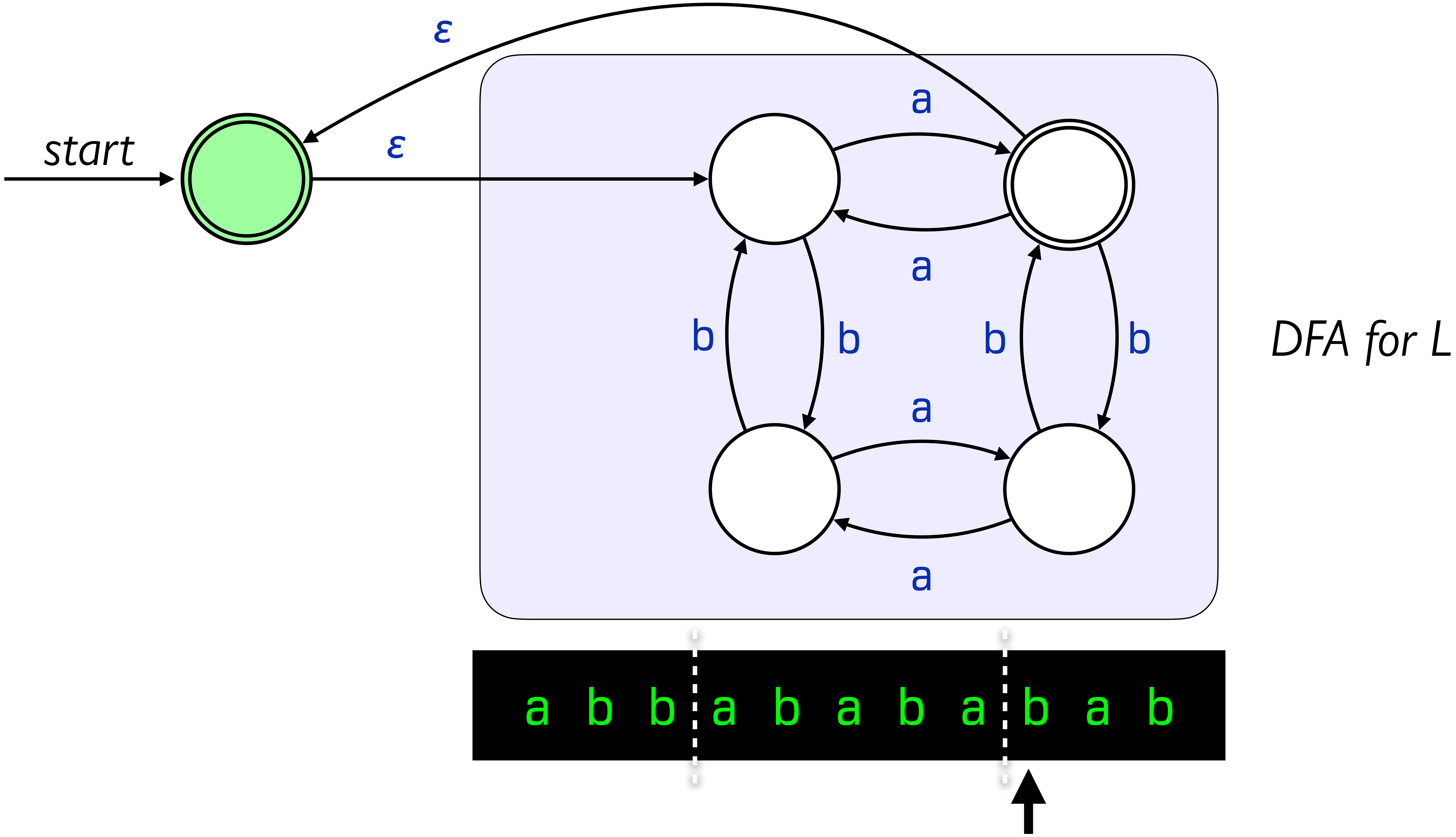
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



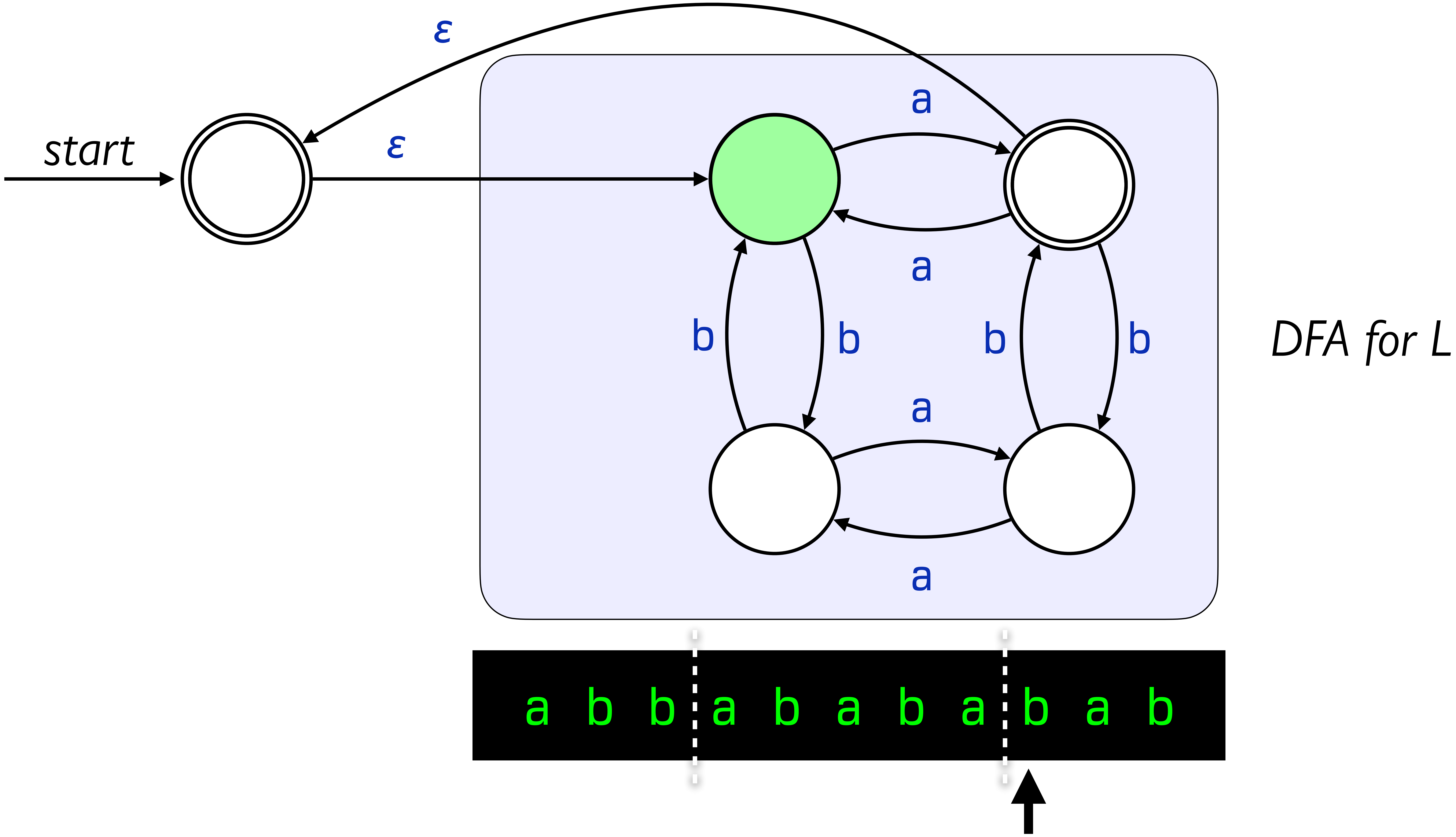
L = {w ∈ {a, b}* | w has an odd number of as and an even number of bs}

Construct an NFA for L^*.



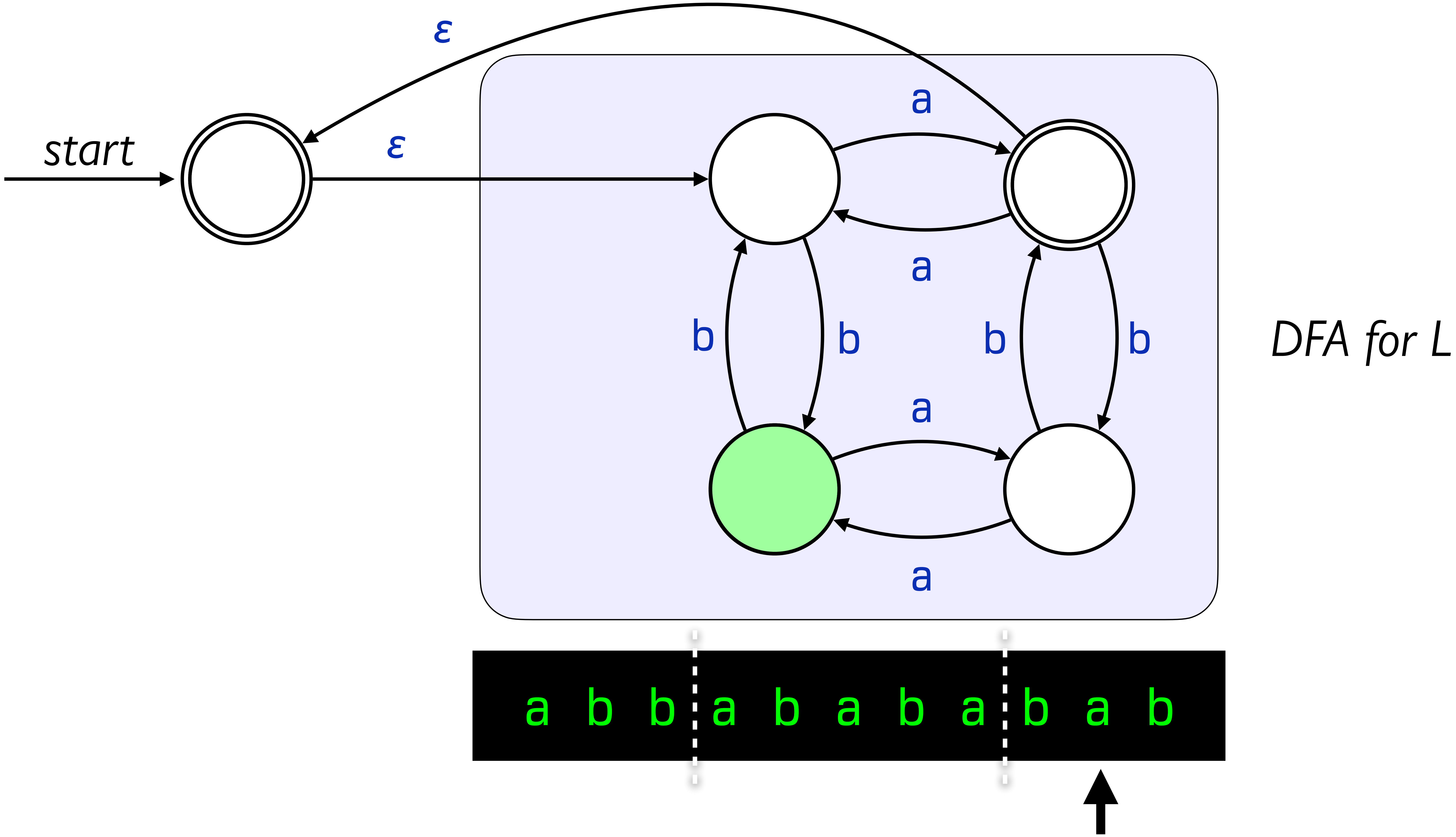
L = {w ∈ {a, b}* | w has an odd number of as and an even number of bs}

Construct an NFA for L^*.



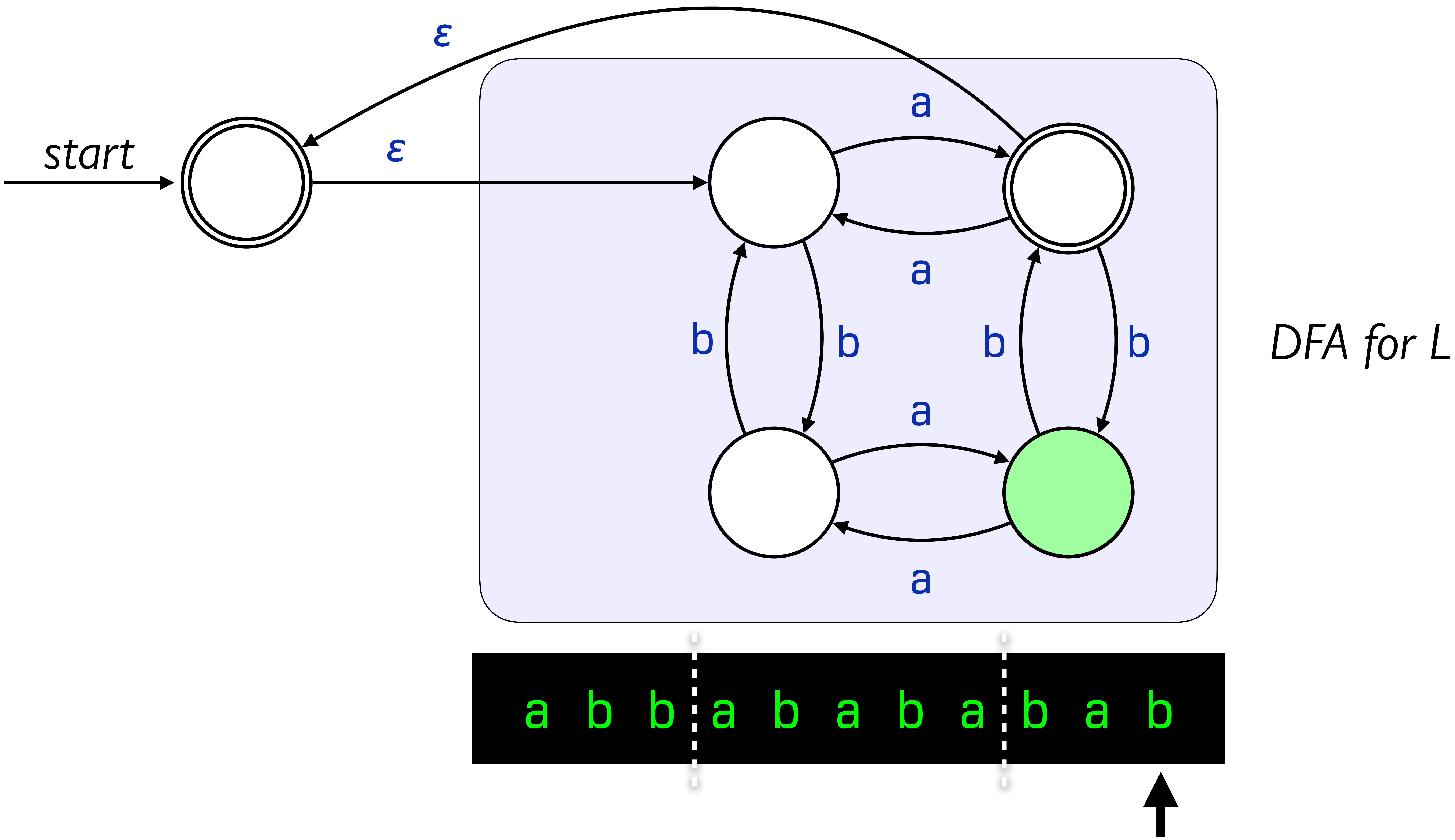
$L = \{w \in \{a, b\}^* \mid w$ has an odd number of as and an even number of bs$\}$

Construct an NFA for L^*.



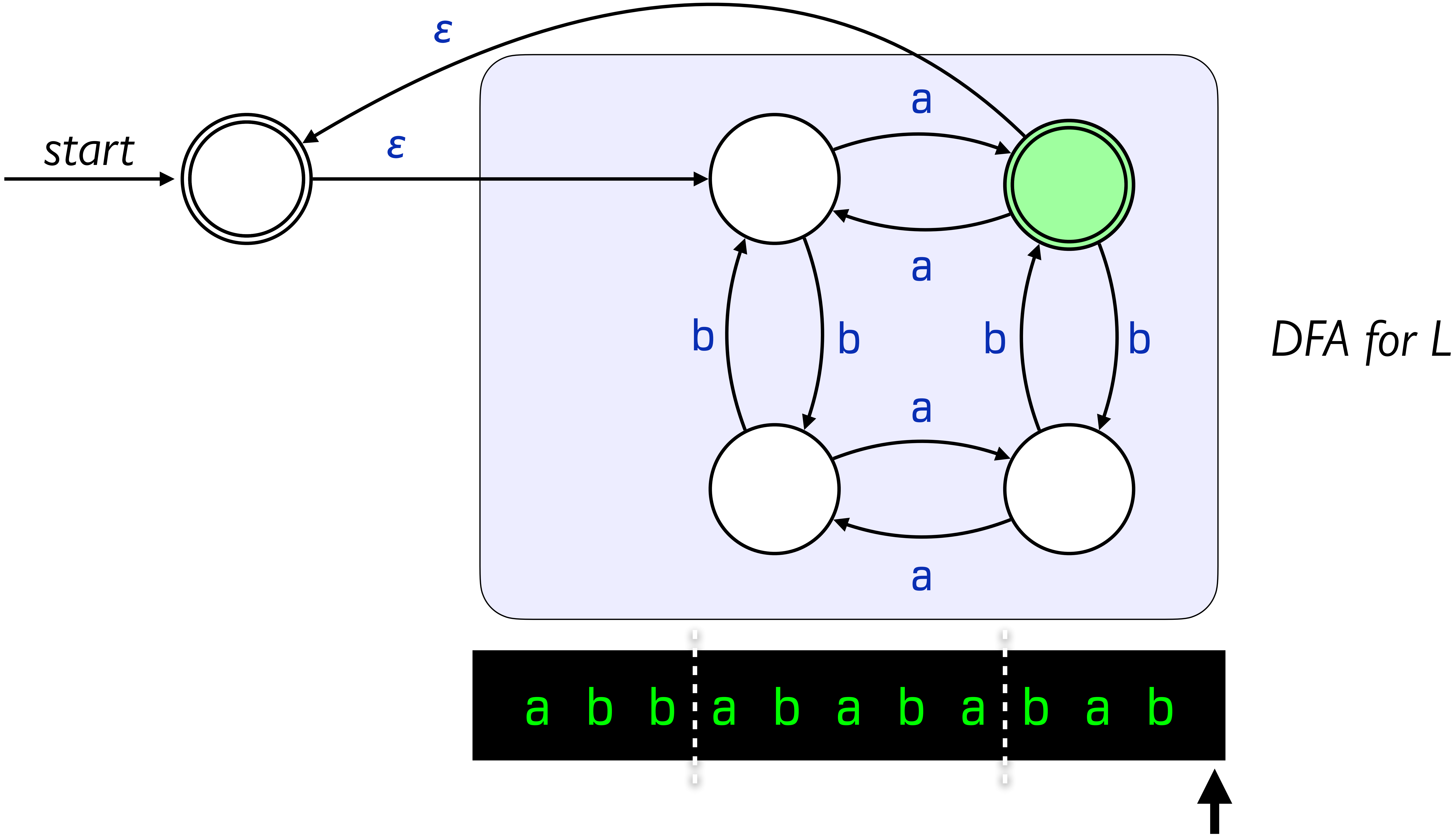
L = {w ∈ {a, b}* | w has an odd number of as and an even number of bs}

Construct an NFA for L^*.



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

Construct an NFA for L^*.



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

Construct an NFA for L^*.

So what?

Closure properties

THEOREM If L_1 and L_2 are regular languages over an alphabet Σ, then so are the following languages:

$\overline{L_1}$

$L_1 \cup L_2$

$L_1 \cap L_2$

$L_1 L_2$

L_1^*

These properties are *closure properties of the regular languages*.

Using three of these closure properties – concatenation, union, and Kleene star – we can build our third and final view of the regular languages: *regular expressions*.

Next time!

Acknowledgments

This lecture incorporates material from:

David Chiang, University of Notre Dame

Nancy Ide, Vassar College

Keith Schwarz, Stanford University

Michael Sipser, *Introduction to the Theory of Computation*

Jeffrey Ullman, Stanford University