CMPU 240 · Theory of Computation

Nondeterministic Finite Automata

3 February 2026

VASSAR COLLEGE A·D·1861

Assignment 1 due today

Assignment 1 corrections due on Thursday

- Example solutions will be posted after class on Ed www.cs.vassar.edu/~cs240/resources/assignments.pdf

Assignments and corrections

In many courses, homework turns into a painful cycle: If you make mistakes on an assignment, turn it in, get your grade back a week and a half later, throw it in a notebook because the class has moved on to other material, and don't think about those problems again until exam time, you won't learn from your mistakes, and the time and effort you spent on the homework will have been wasted.

In this class, we're making an effort to do better. Instead of submitting assignments (often late) and then waiting for me to grade and comment on your work after all assignments are in, I will be releasing example solutions as soon as the assignment is due and then asking you to correct your own work. This ensures that you will review the exercises while you still remember what you were thinking, and you will get credit for learning from what you did wrong, rather than be punished for not knowing the material perfectly from the start.

It's entirely reasonable to make mistakes when you're first learning material. As such, I want the homework assignments to be low stress. This is my promise: If you make a serious effort to solve the problems, turn them in by the deadline, and carefully review your work to understand how your solutions could be improved, then your homework grade will be very high, even if you initially make mistakes on every problem.

Instructions and examples for correcting assignments

Where are we?

Language theory

An *alphabet* is a set, denoted Σ, whose elements are called *characters*.

A *string over Σ* is a finite sequence of zero or more characters drawn from Σ.

The *empty string*, denoted ε, has no characters.

A *language over Σ* is a set of strings over Σ.

The language Σ^* is the set of all strings over Σ.

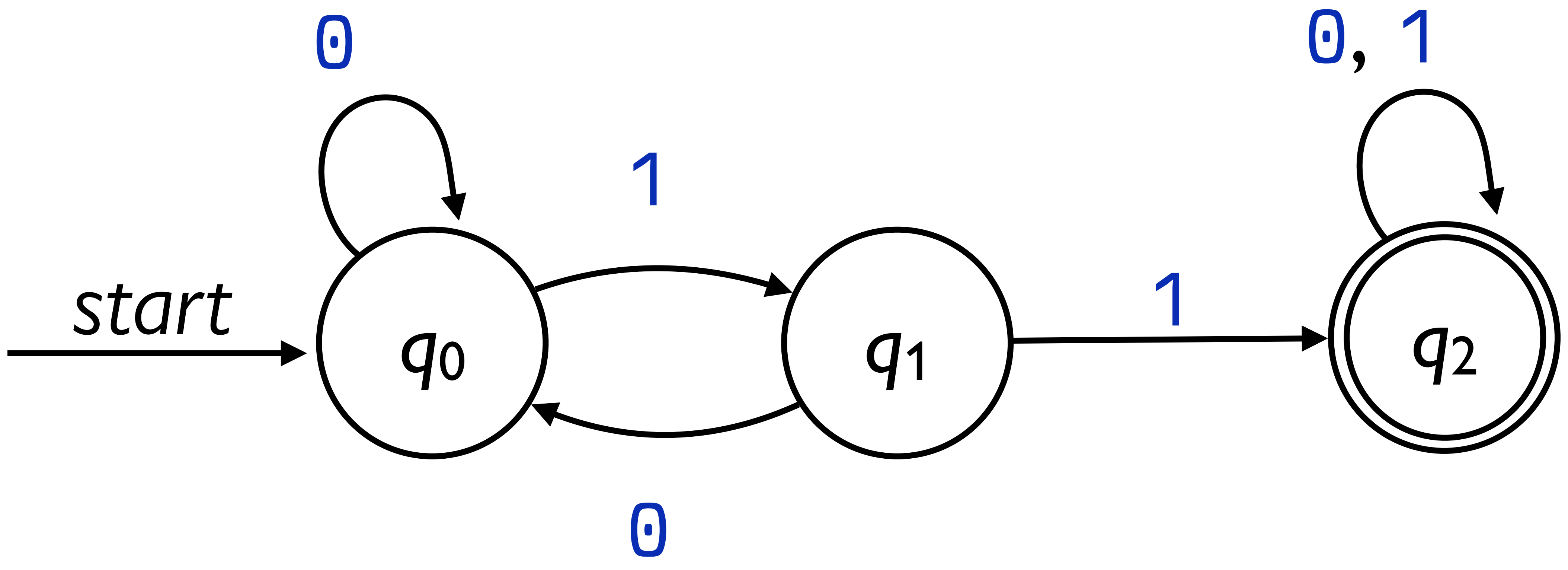
Automata

A *deterministic finite automaton* (DFA) is a simple model of computation, defined relative to some alphabet Σ.

- There is a unique start state.
- There are zero or more accept states.
- For each state in the DFA, there must be *exactly one* transition defined for each symbol in Σ.

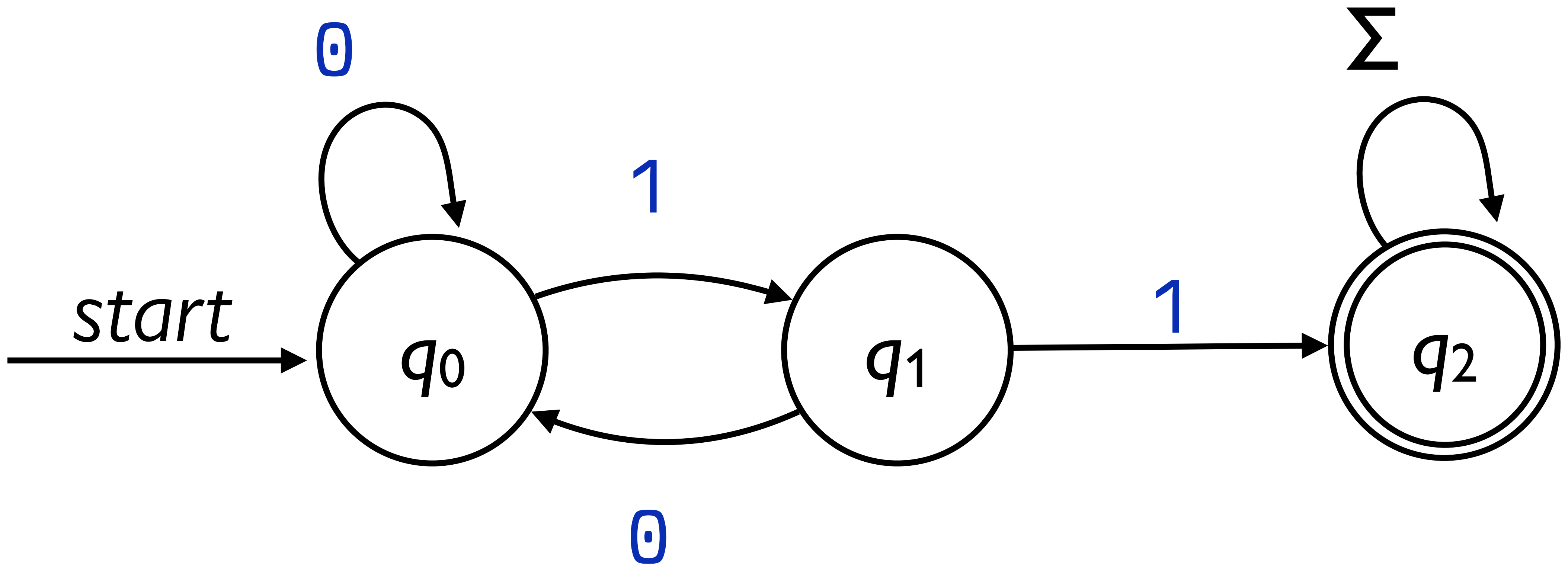
A sample DFA

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



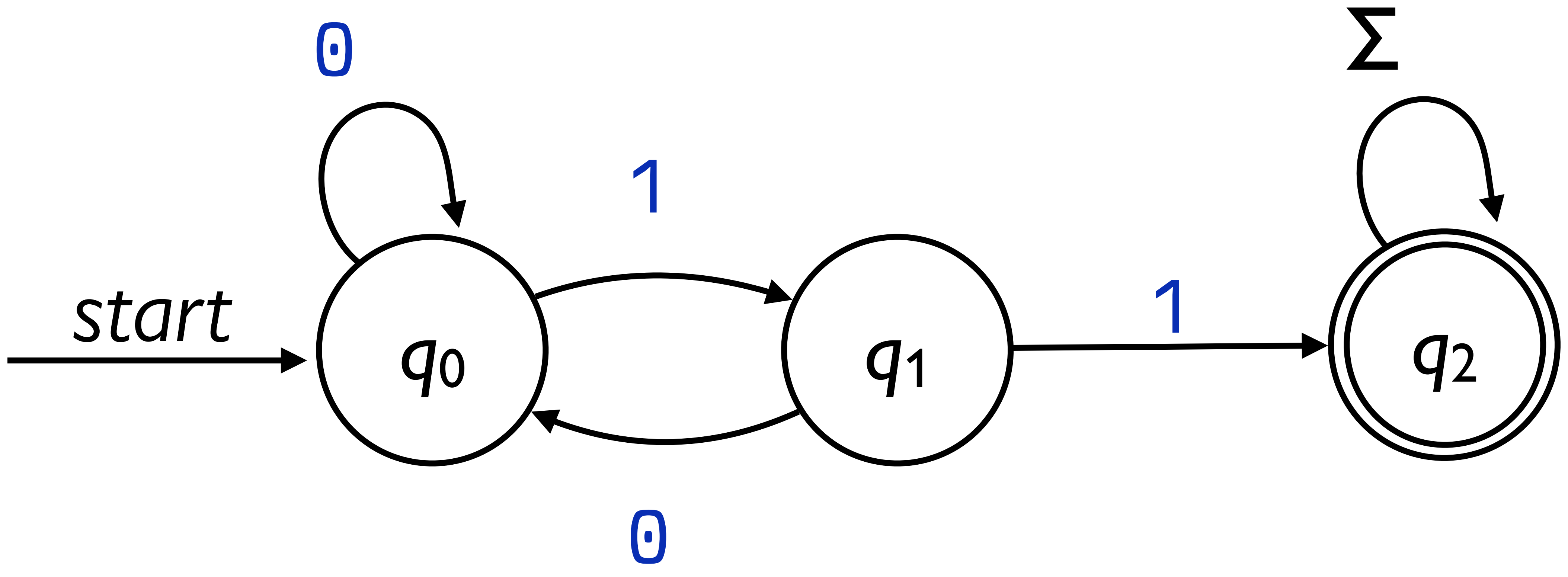
A sample DFA

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



Another way we can write down a DFA is as a transition table:

		0	1
→	q_0	q_0	q_1
	q_1	q_0	q_2
*	q_2	q_2	q_2



Warm-up

EXERCISE Design an automaton to recognize the language of strings that start and end with the same symbol. Let $\Sigma = \{a, b\}$.

Formal definition of a deterministic finite automaton (DFA)

A DFA is represented as a five-tuple $(Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of ***states***,

Σ is the ***alphabet***, a finite set of input symbols,

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

$q_0 \in Q$ is the ***start state***, and

$F \subseteq Q$ is a set of zero or more ***accept states***.

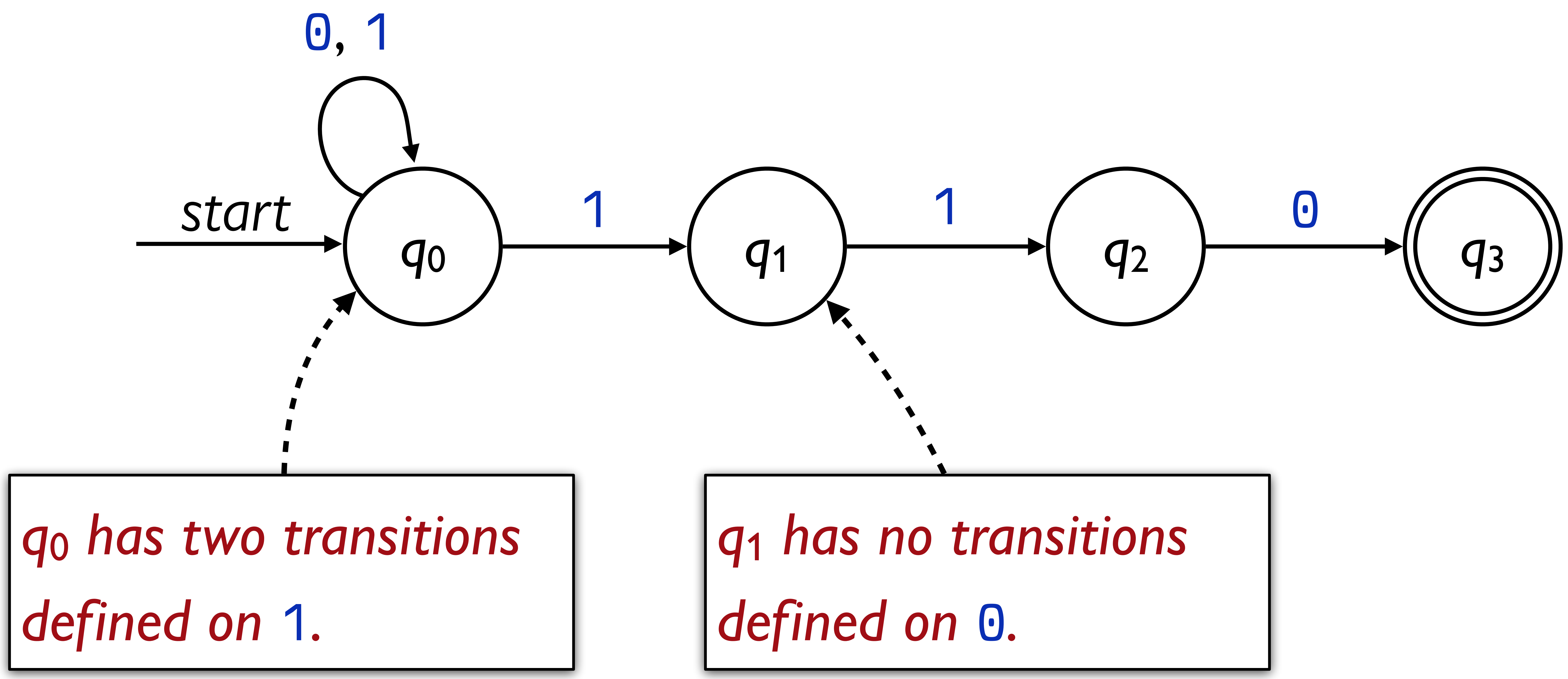
If D is a DFA that processes strings over Σ, the ***language of D***, denoted $L(D)$ is the set of all strings D accepts:

$$L(D) = \{w \in \Sigma^* \mid D \text{ accepts } w\}$$

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

DEFINITION A language L is called a ***regular language*** iff there exists a DFA D such that $L(D) = L$.

*Non*deterministic finite automata


0, 1
start
q_0
1
q_1
1
q_2
0
q_3
q_0 has two transitions defined on 1.
q_1 has no transitions defined on 0.

Nondeterministic finite automata are structurally similar to DFAs, but they represent a fundamental shift in how we'll think about computation.

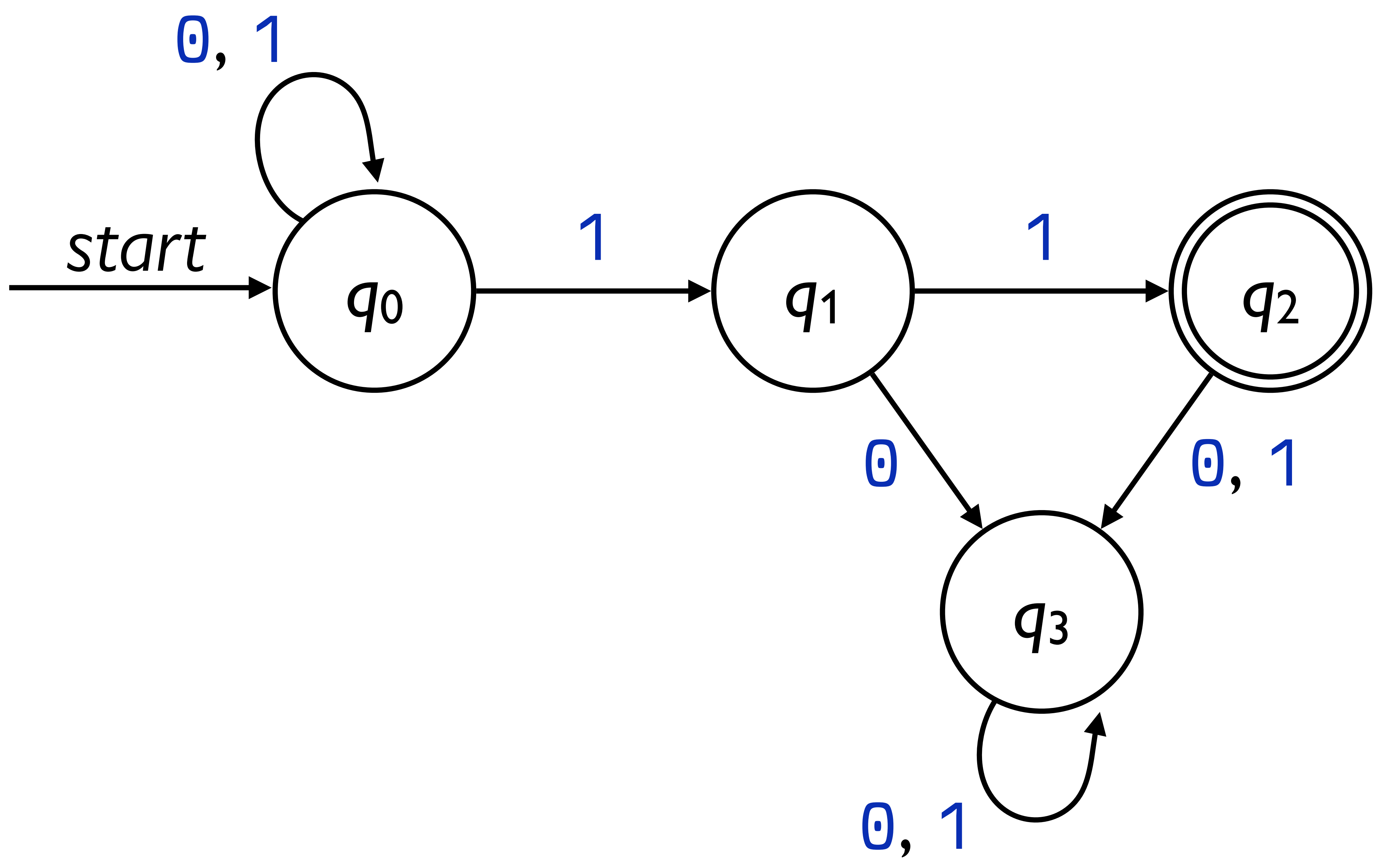
A model of computation is *deterministic* if, at every point in the computation, there is exactly *one choice* it can make.

The machine accepts if that series of choices leads to an accept state.

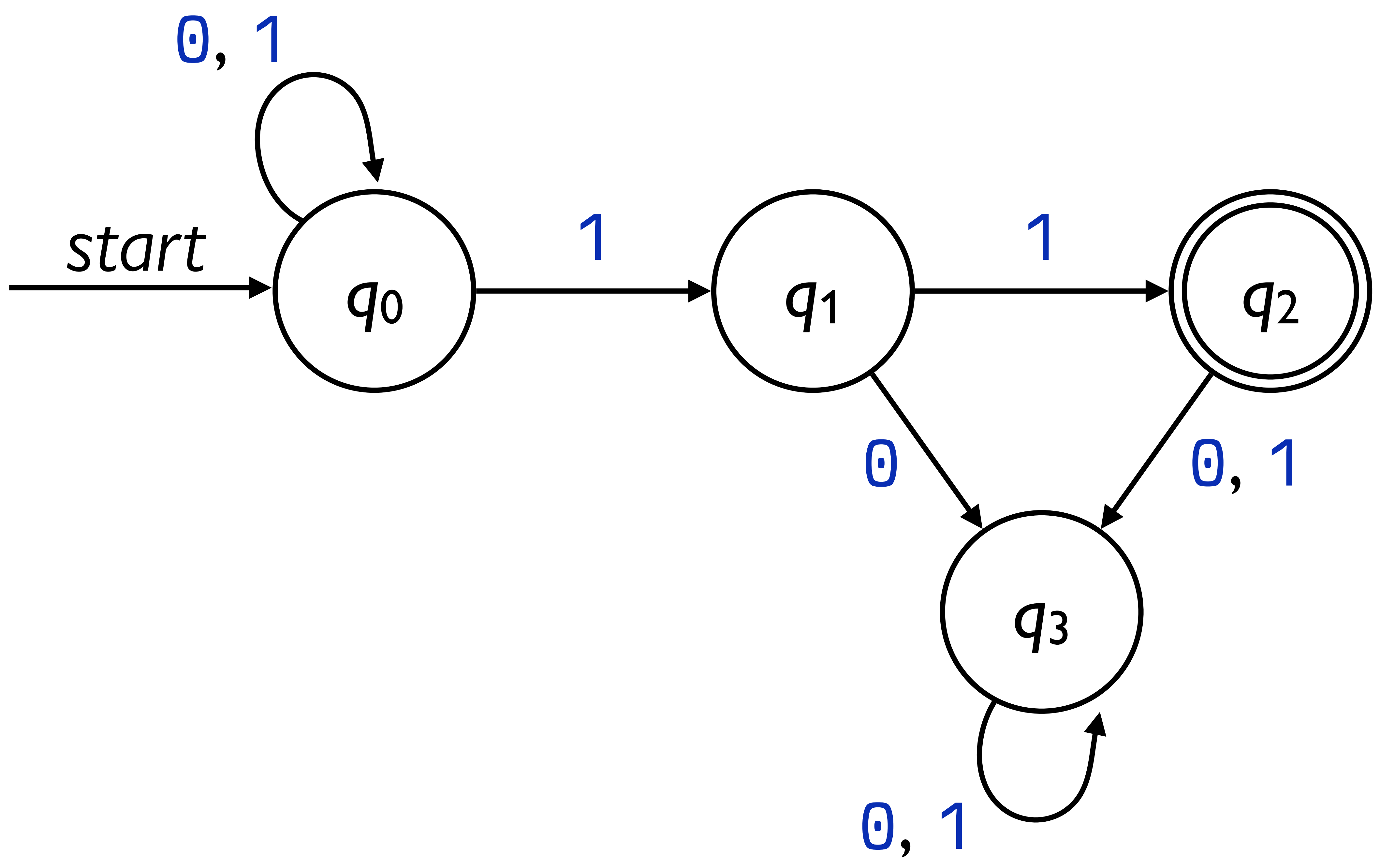
A model of computation is *nondeterministic* if the machine has *zero or more* decisions it can make at one point.

The machine accepts if any series of choices leads to an accept state.

A simple NFA

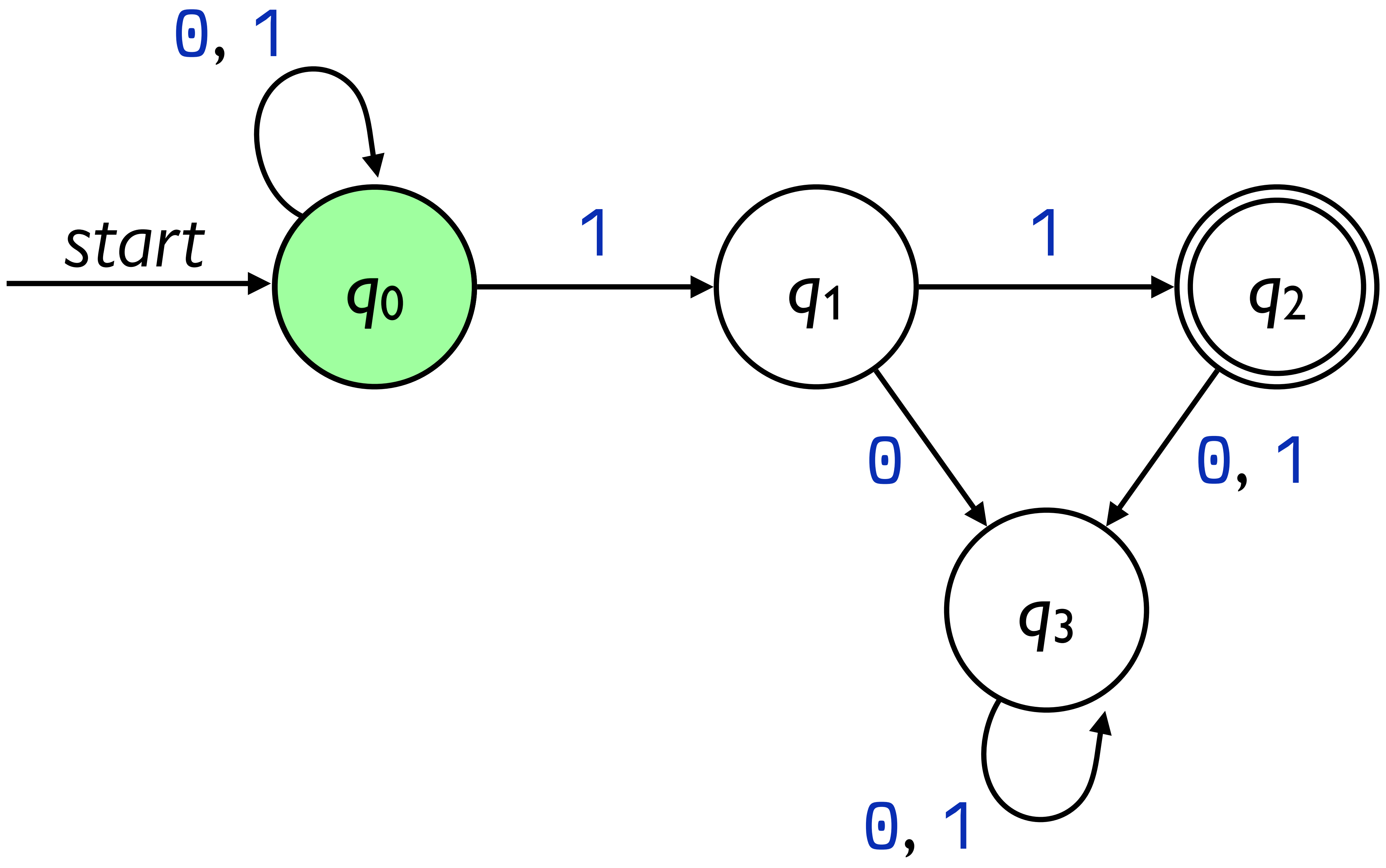


A simple NFA

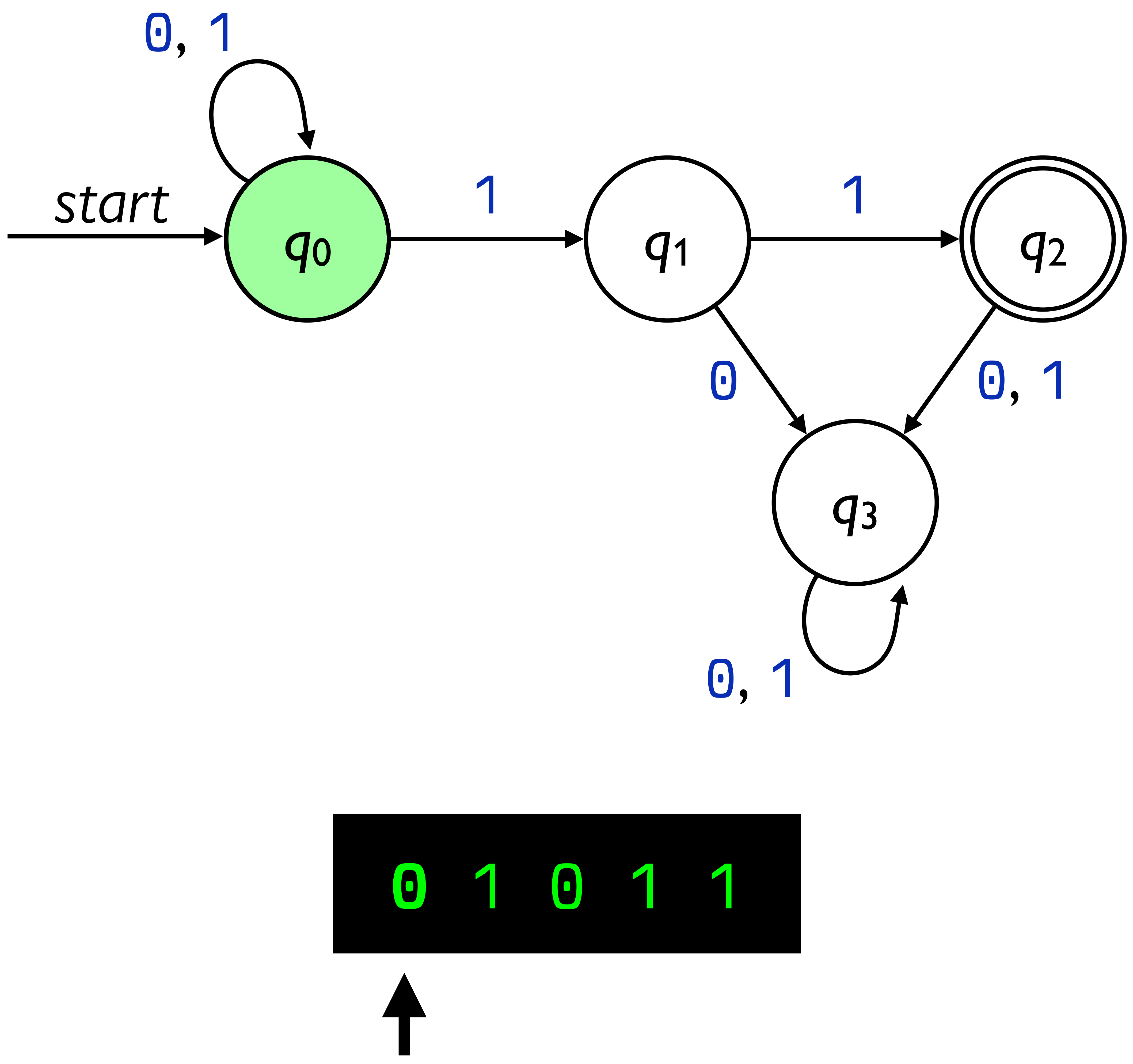


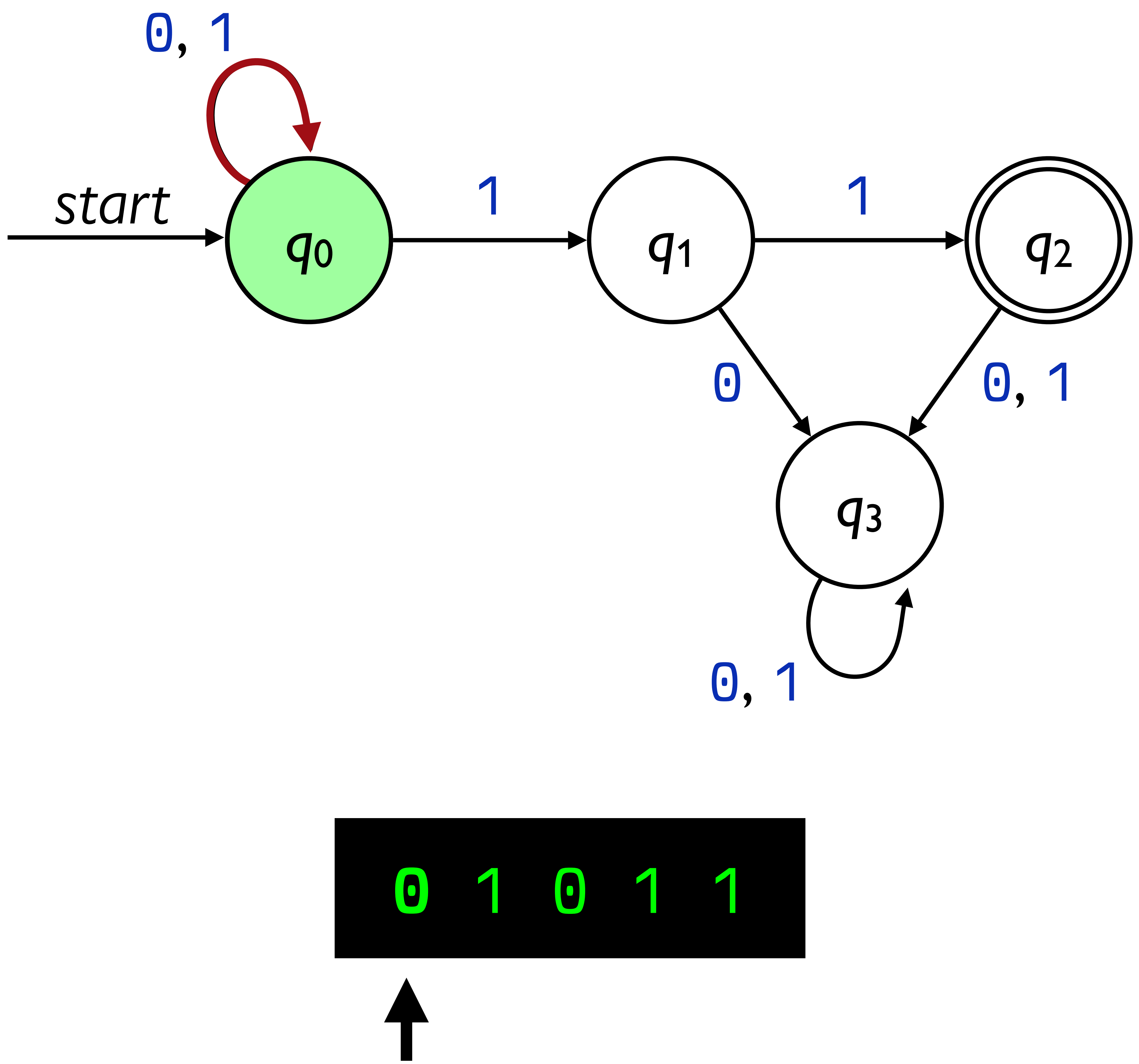
0 1 0 1 1

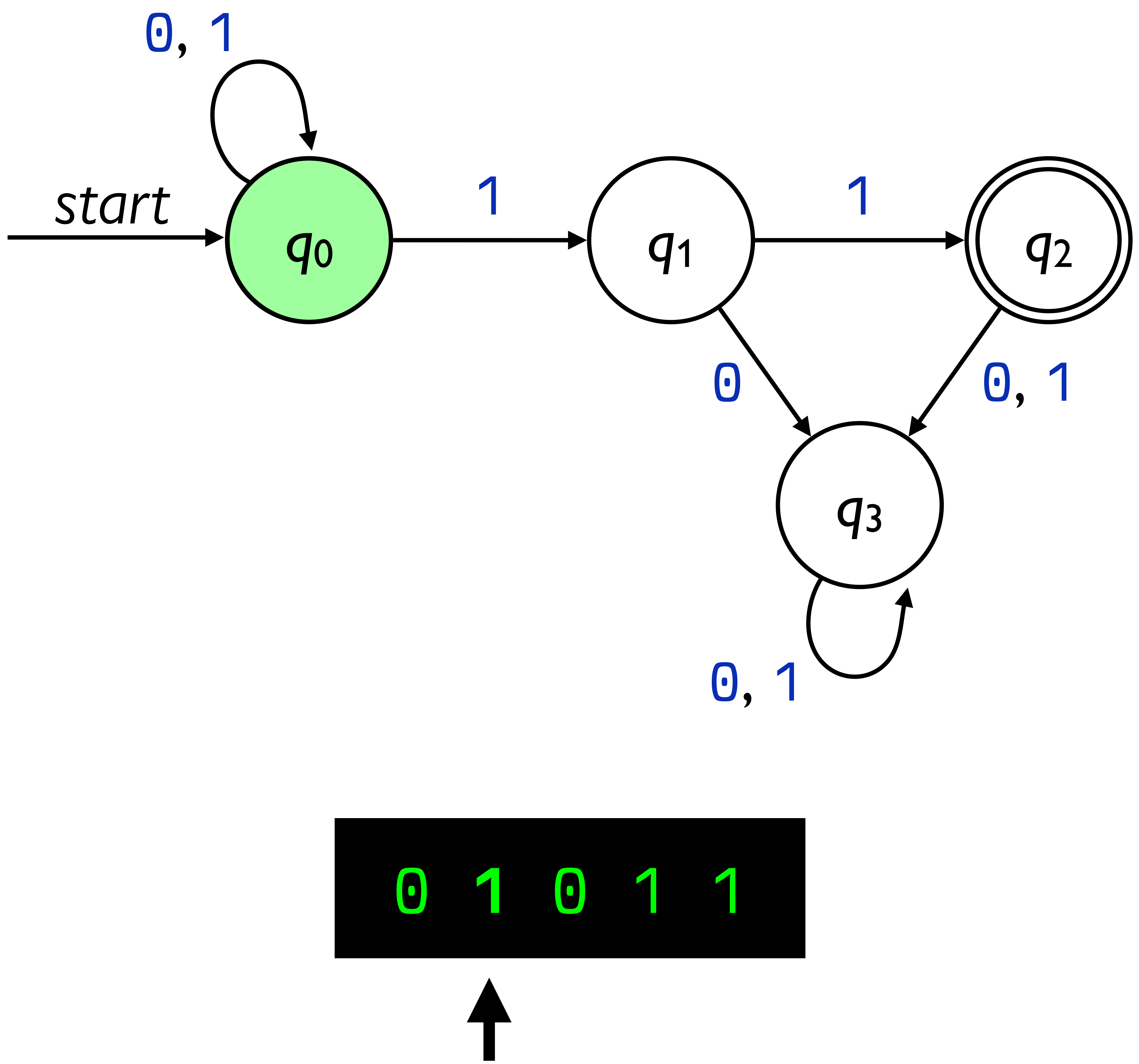
A simple NFA

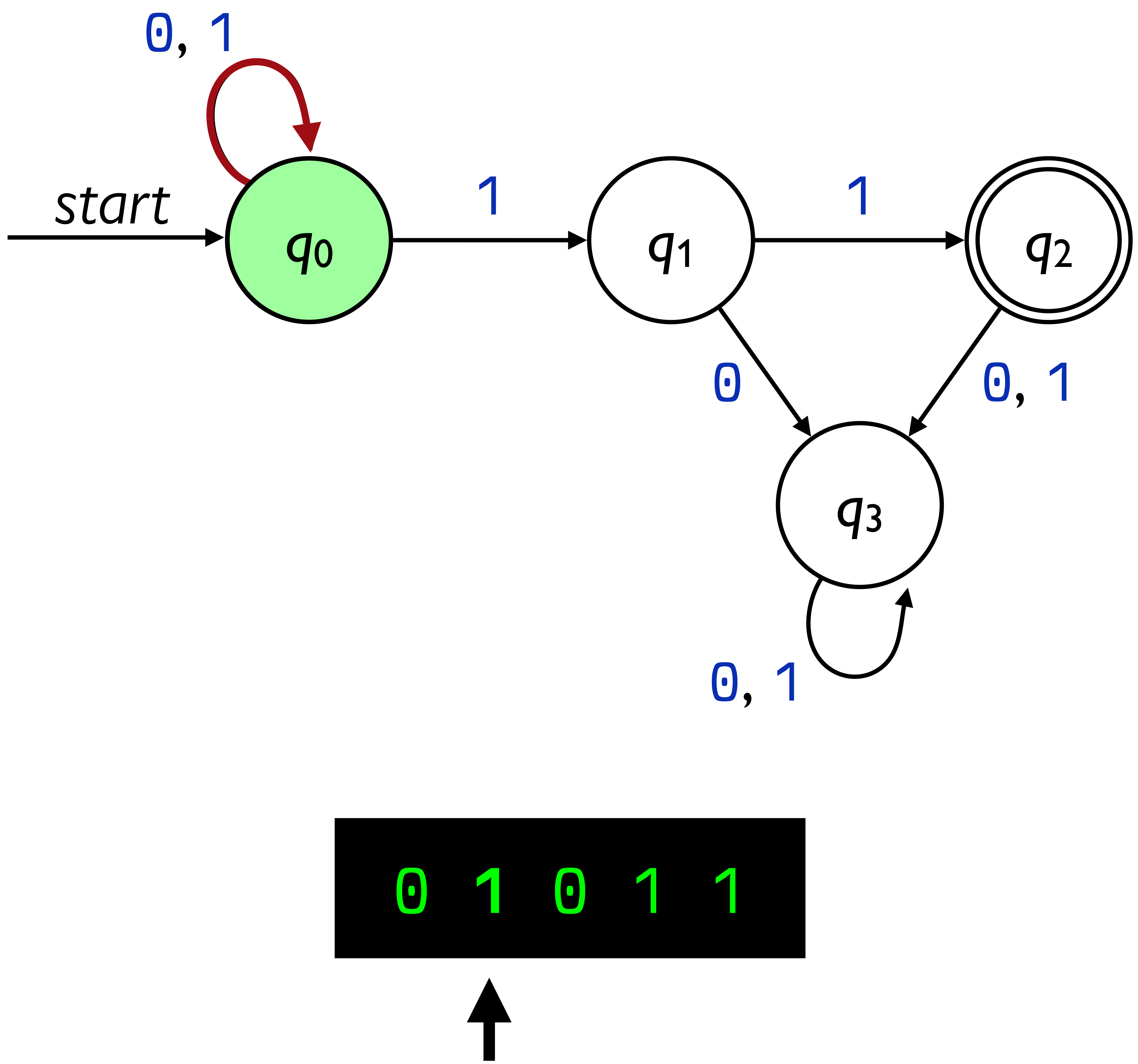


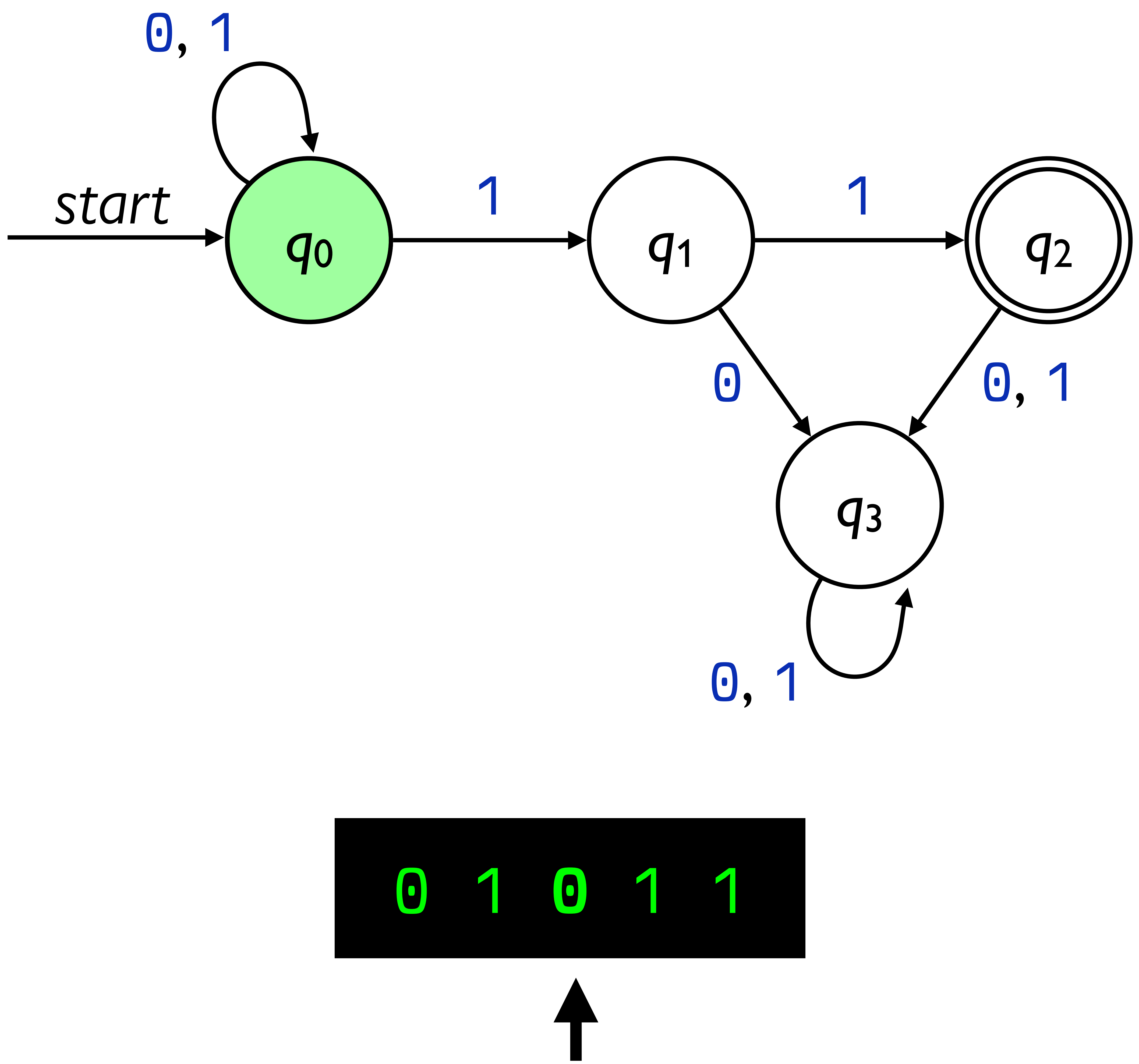
0 1 0 1 1

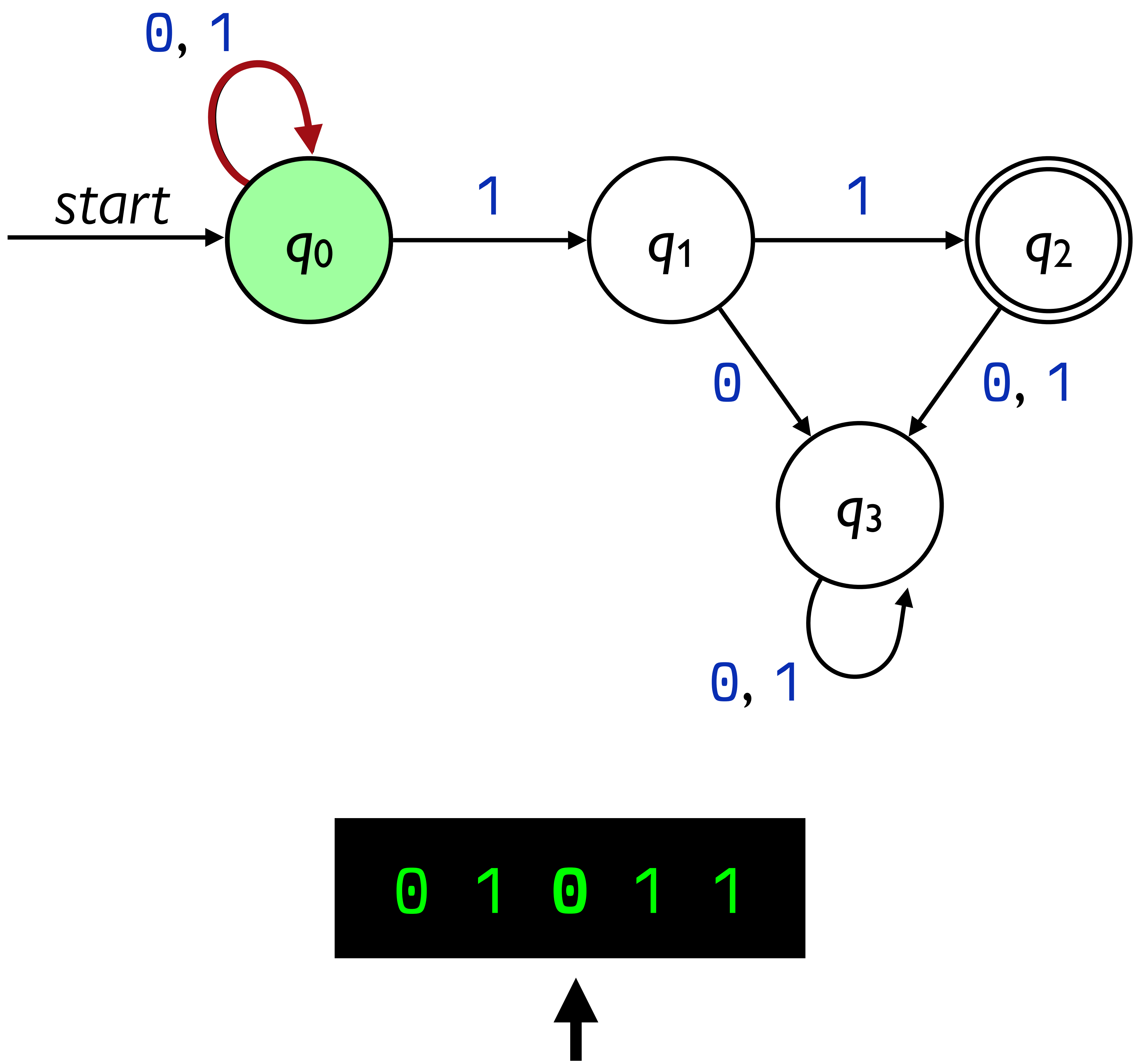

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

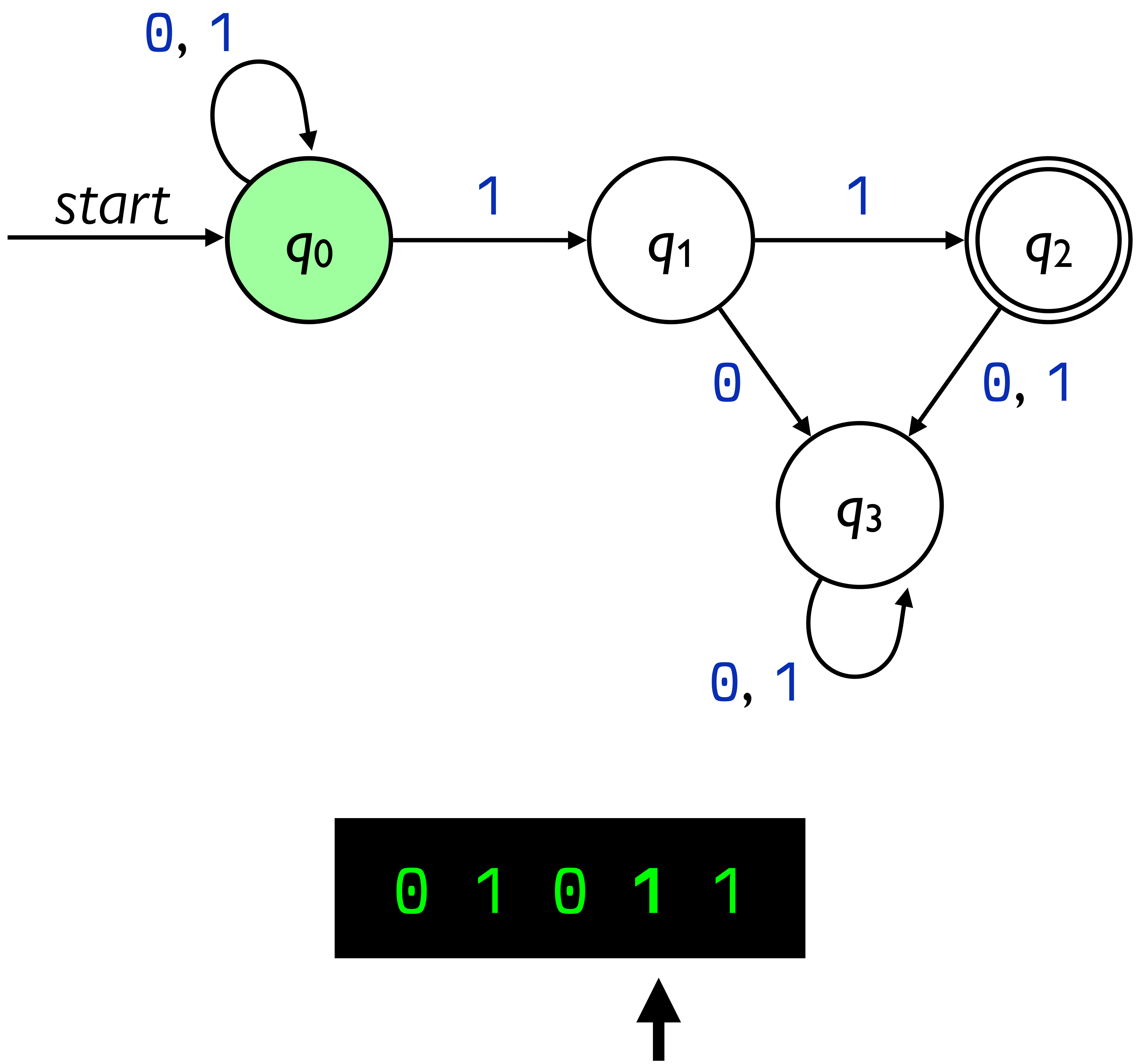

0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1
0 1 0 1 1

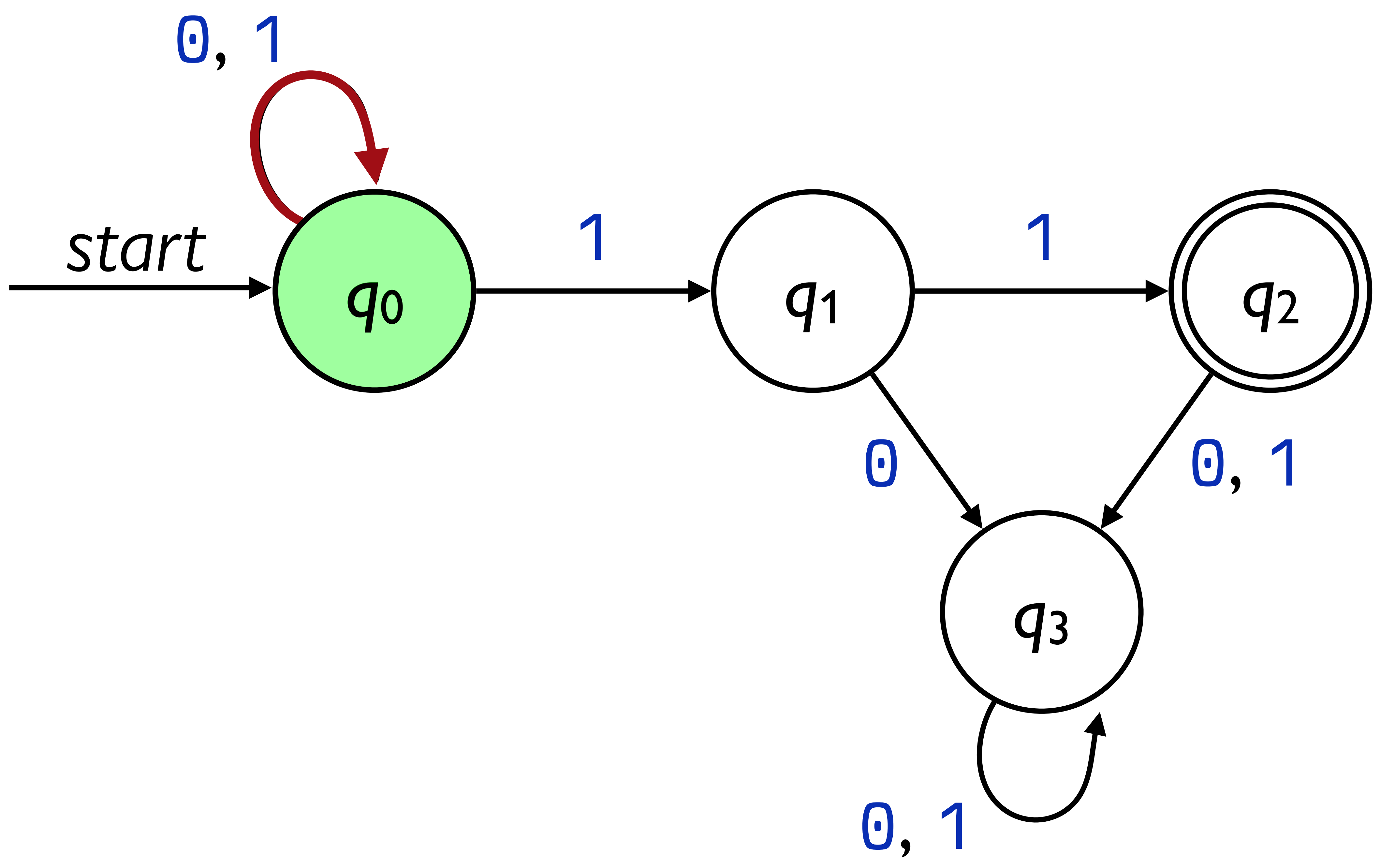

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

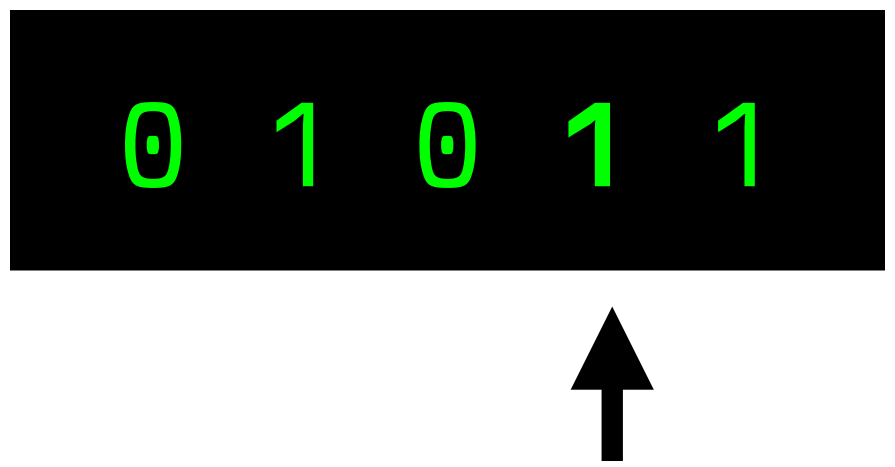

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

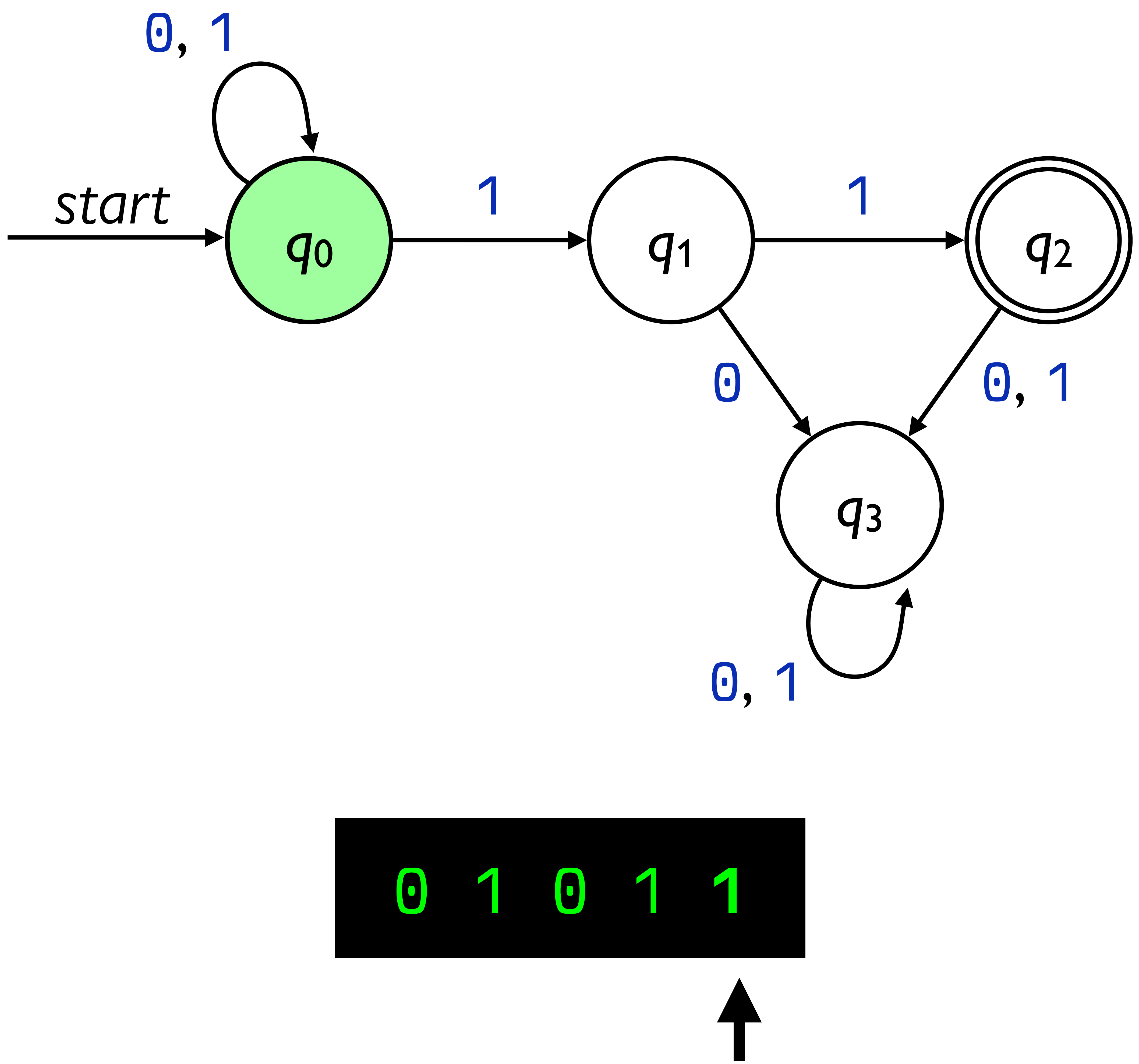

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

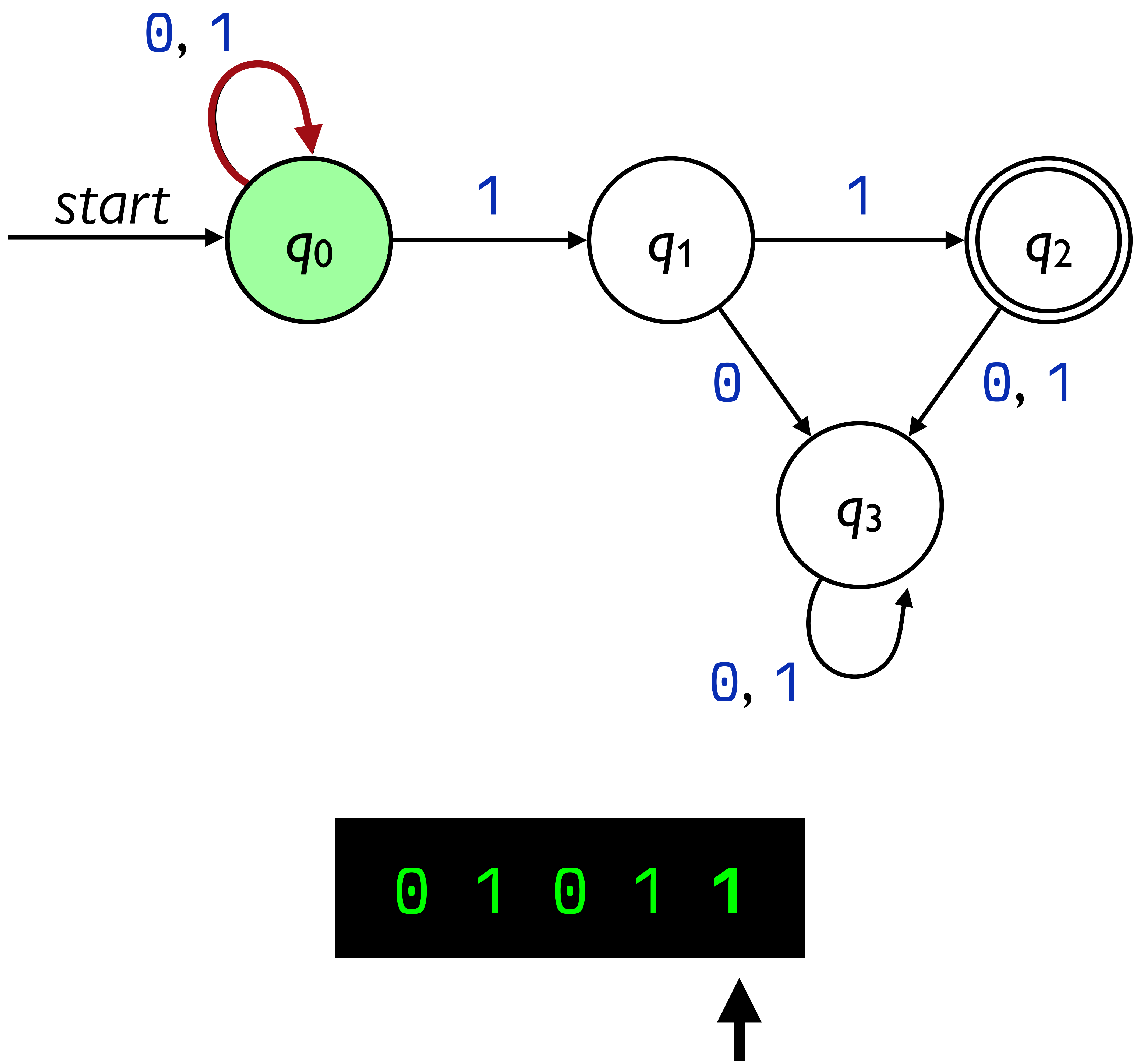

0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1
0 1 0 1 1


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

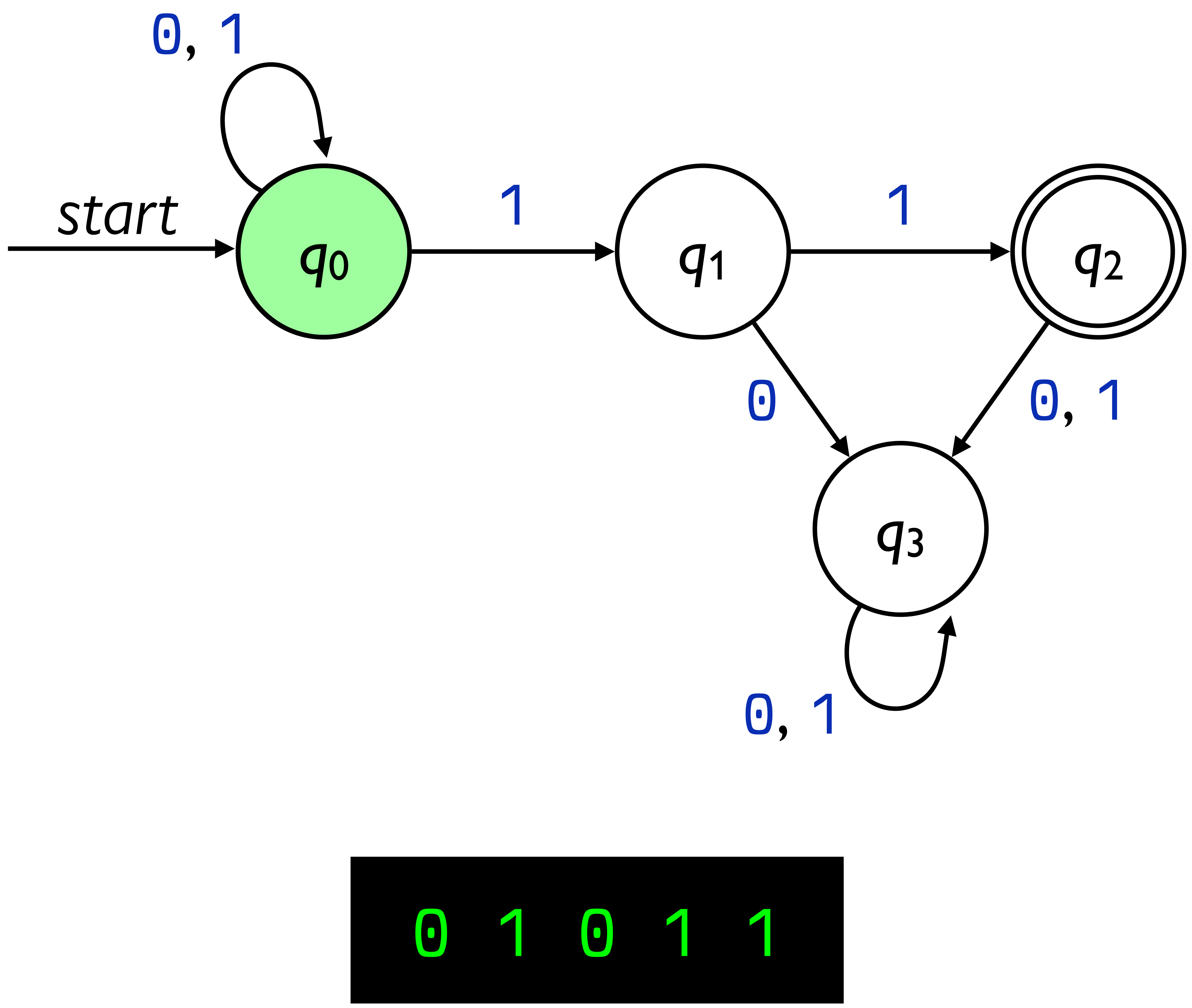

0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1

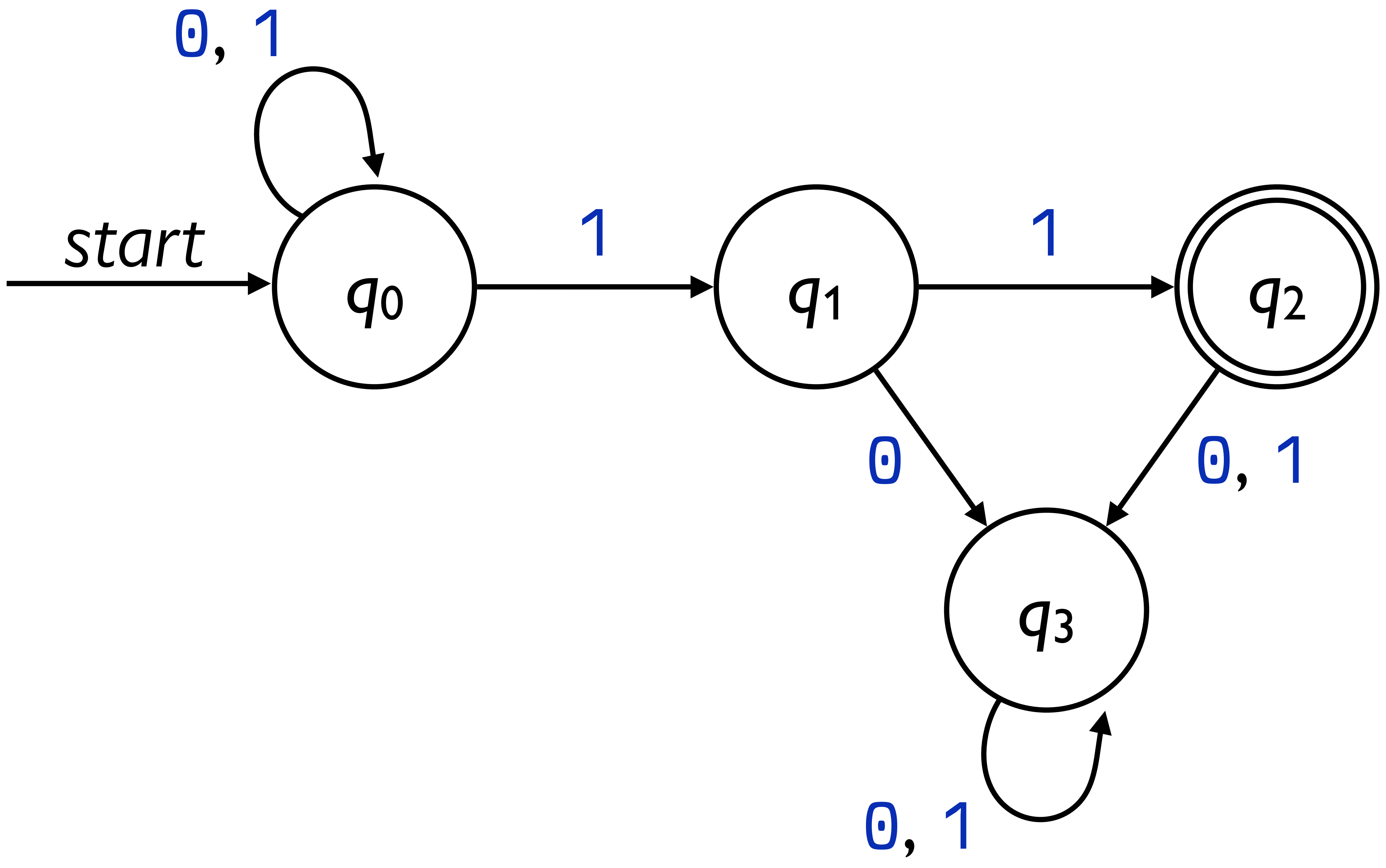

0 1 0 1 1


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

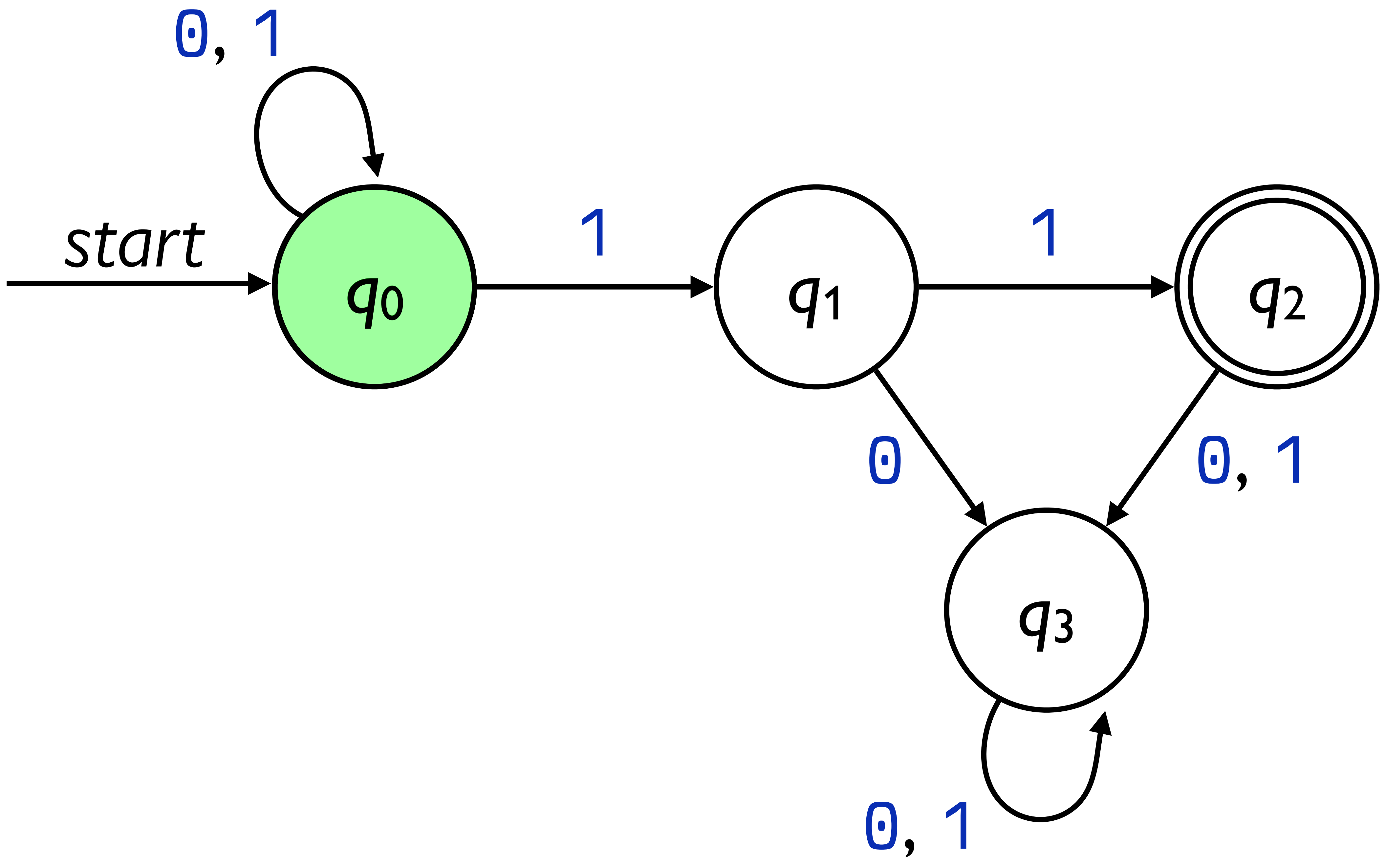


0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1
0 1 0 1 1

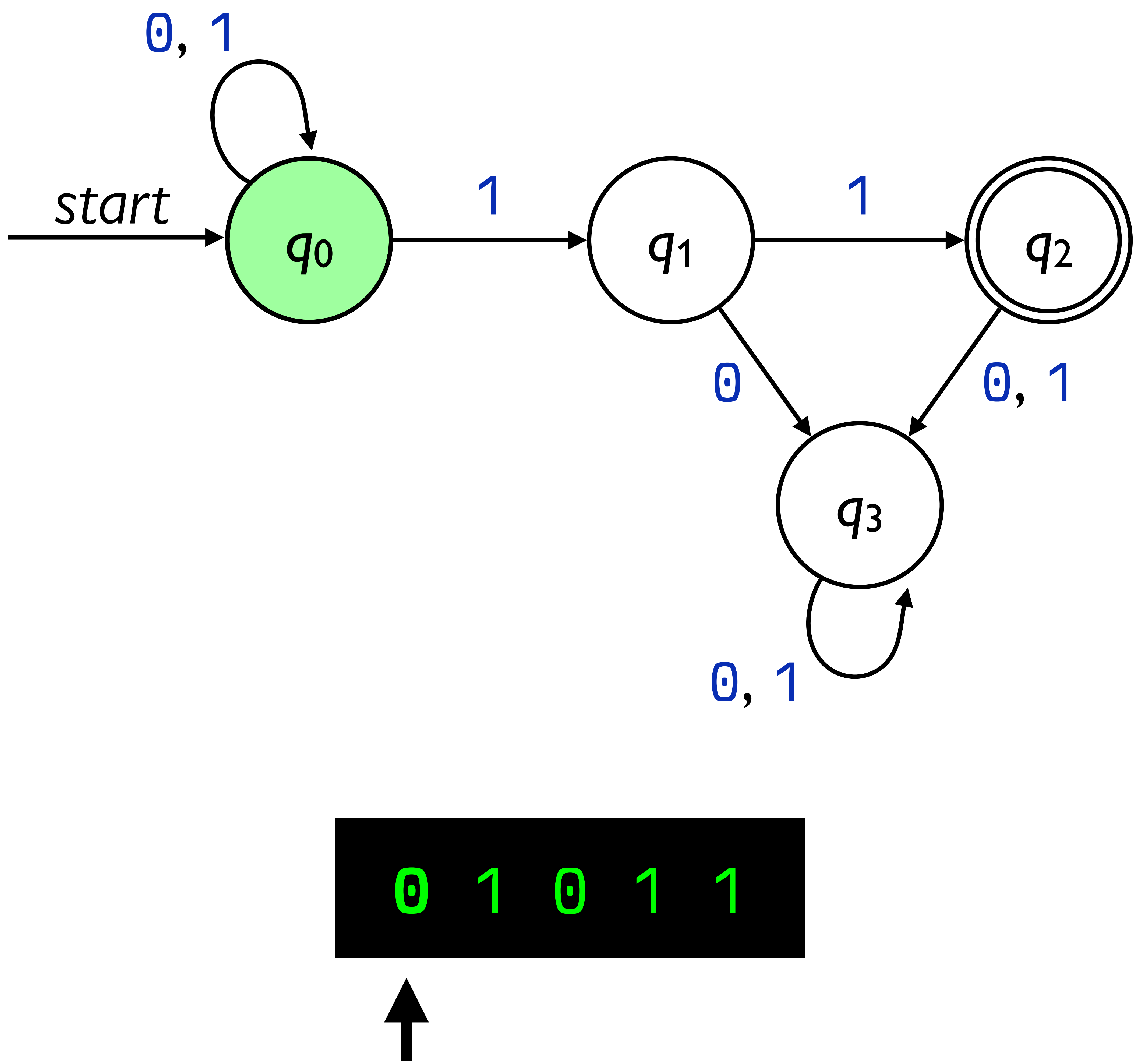

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

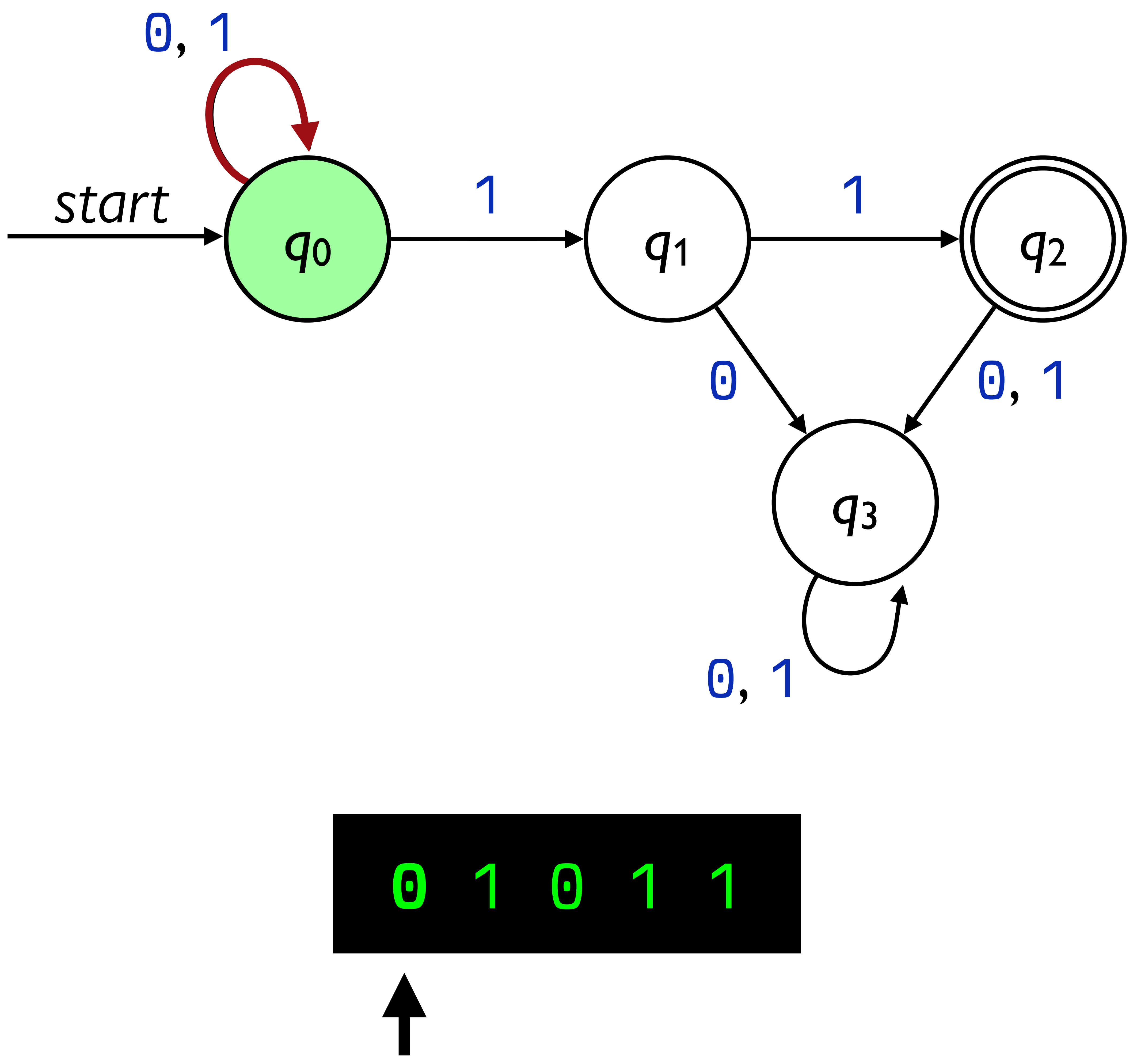


0 1 0 1 1

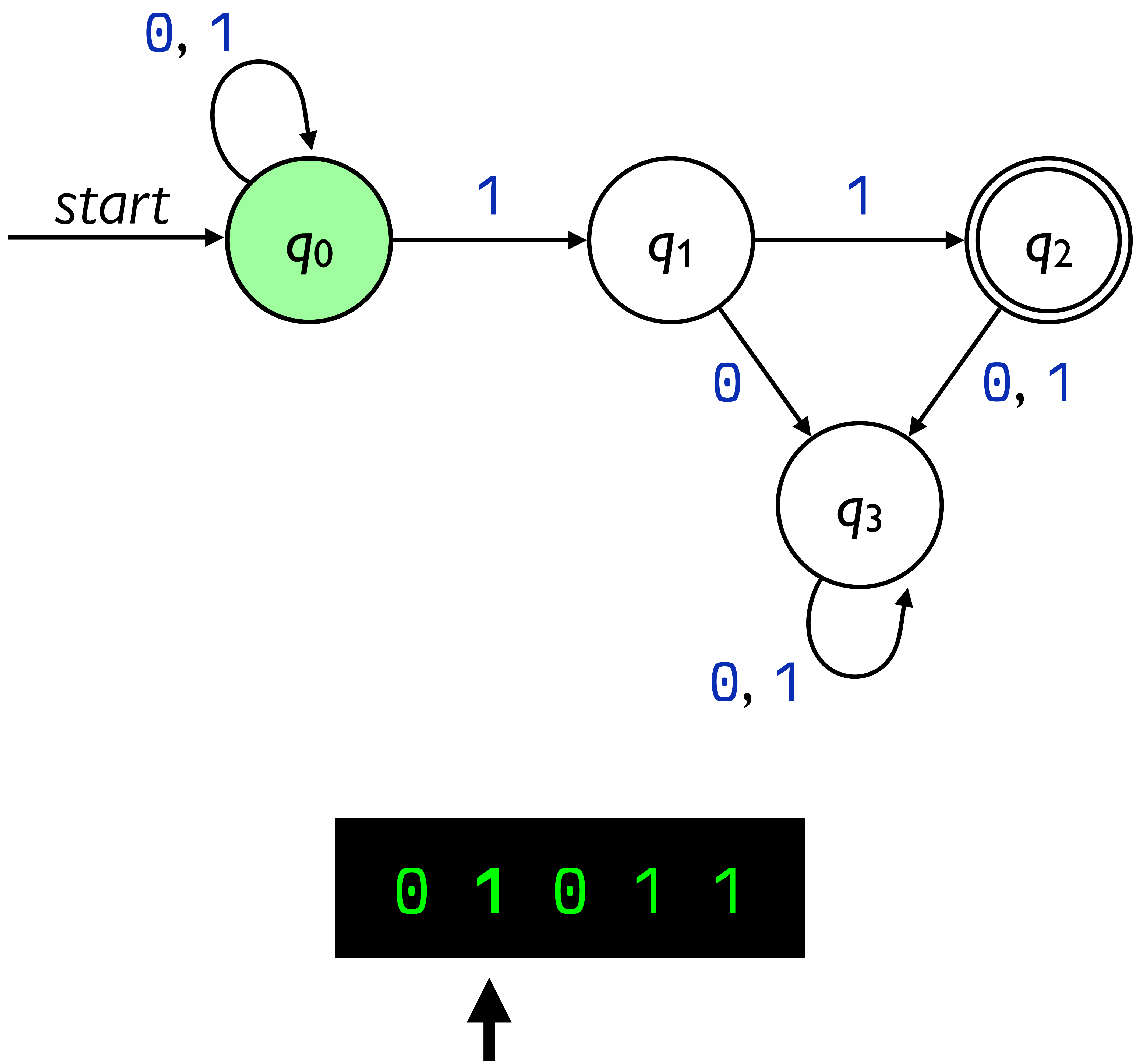


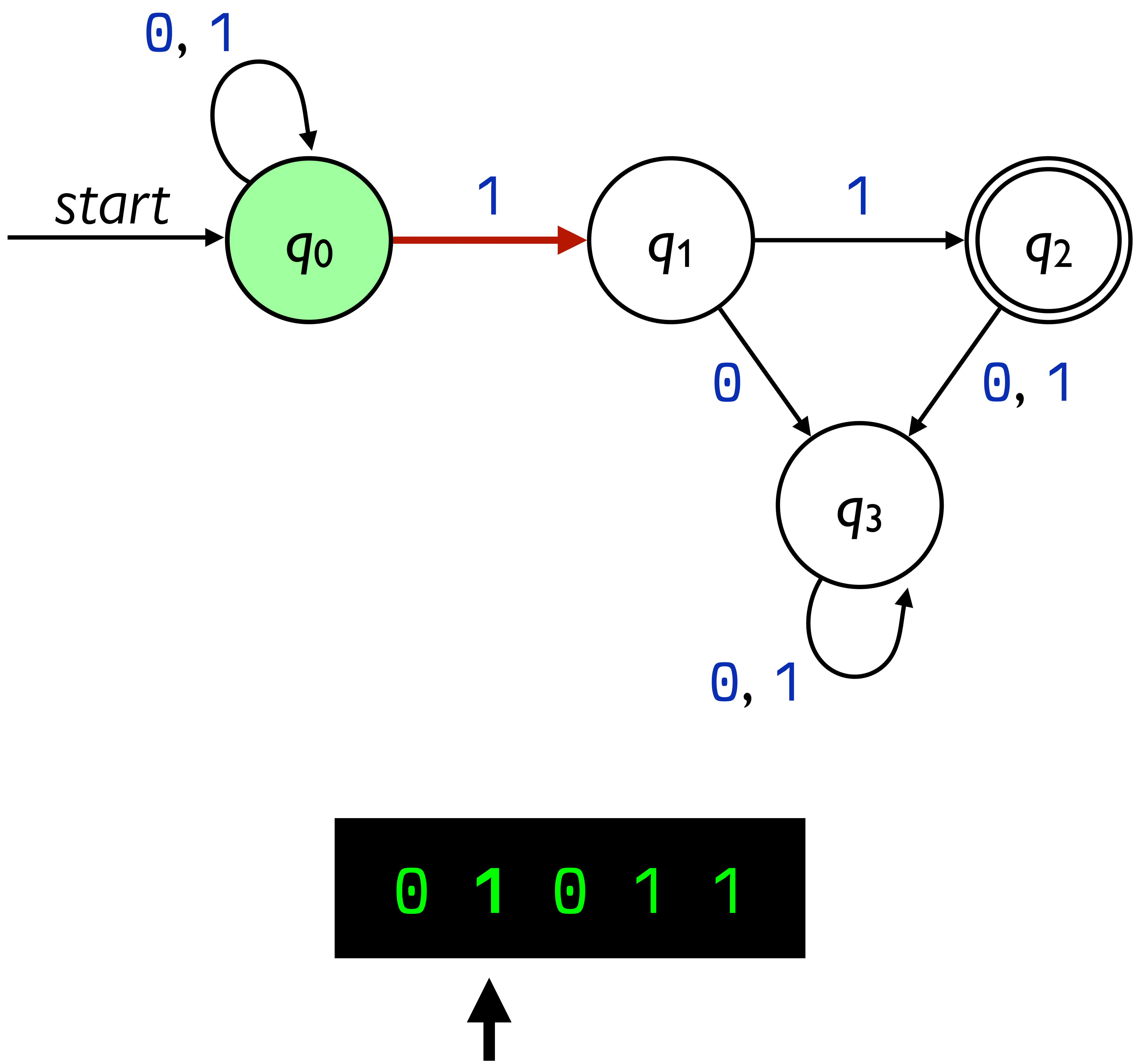
0 1 0 1 1

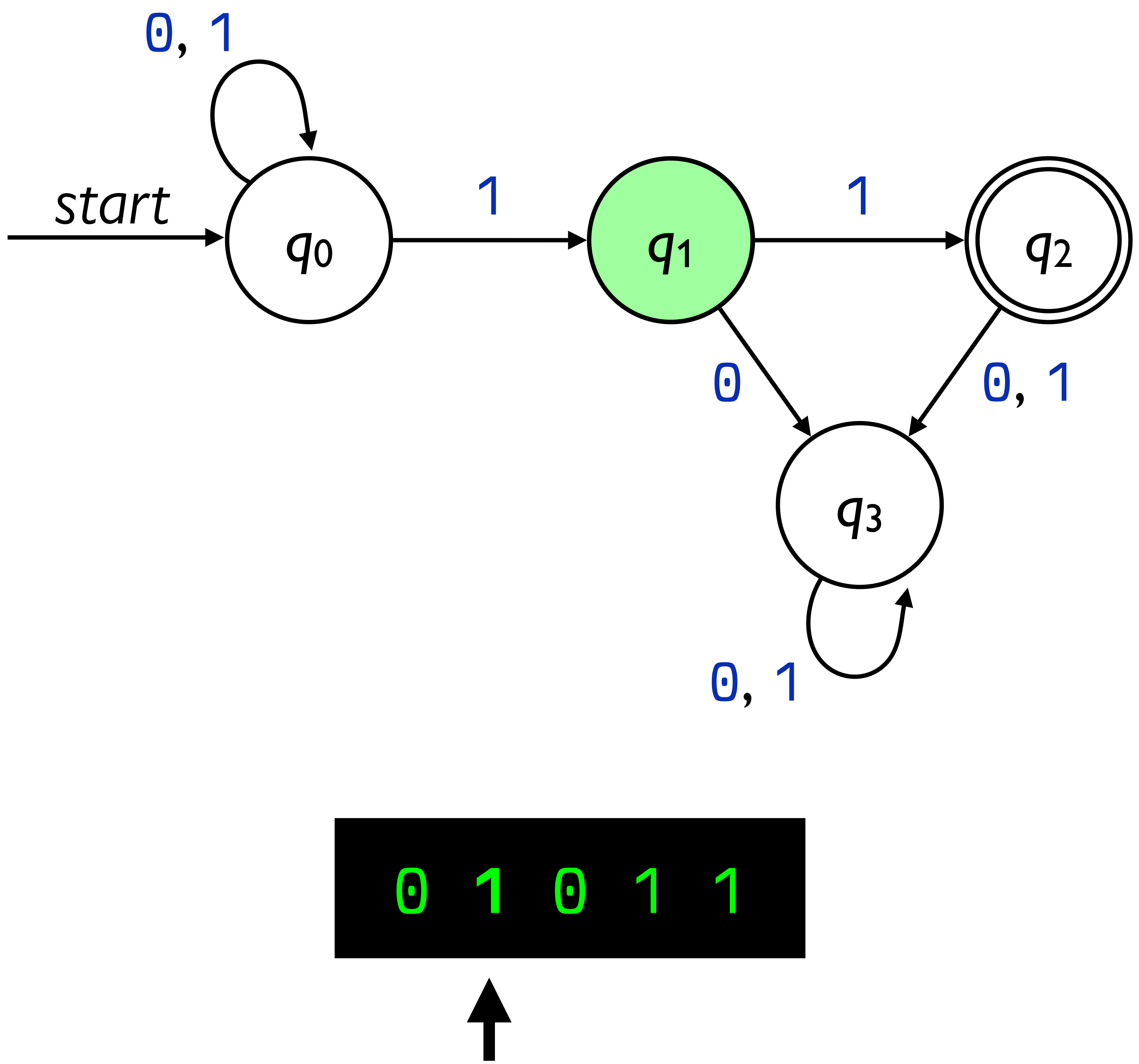

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

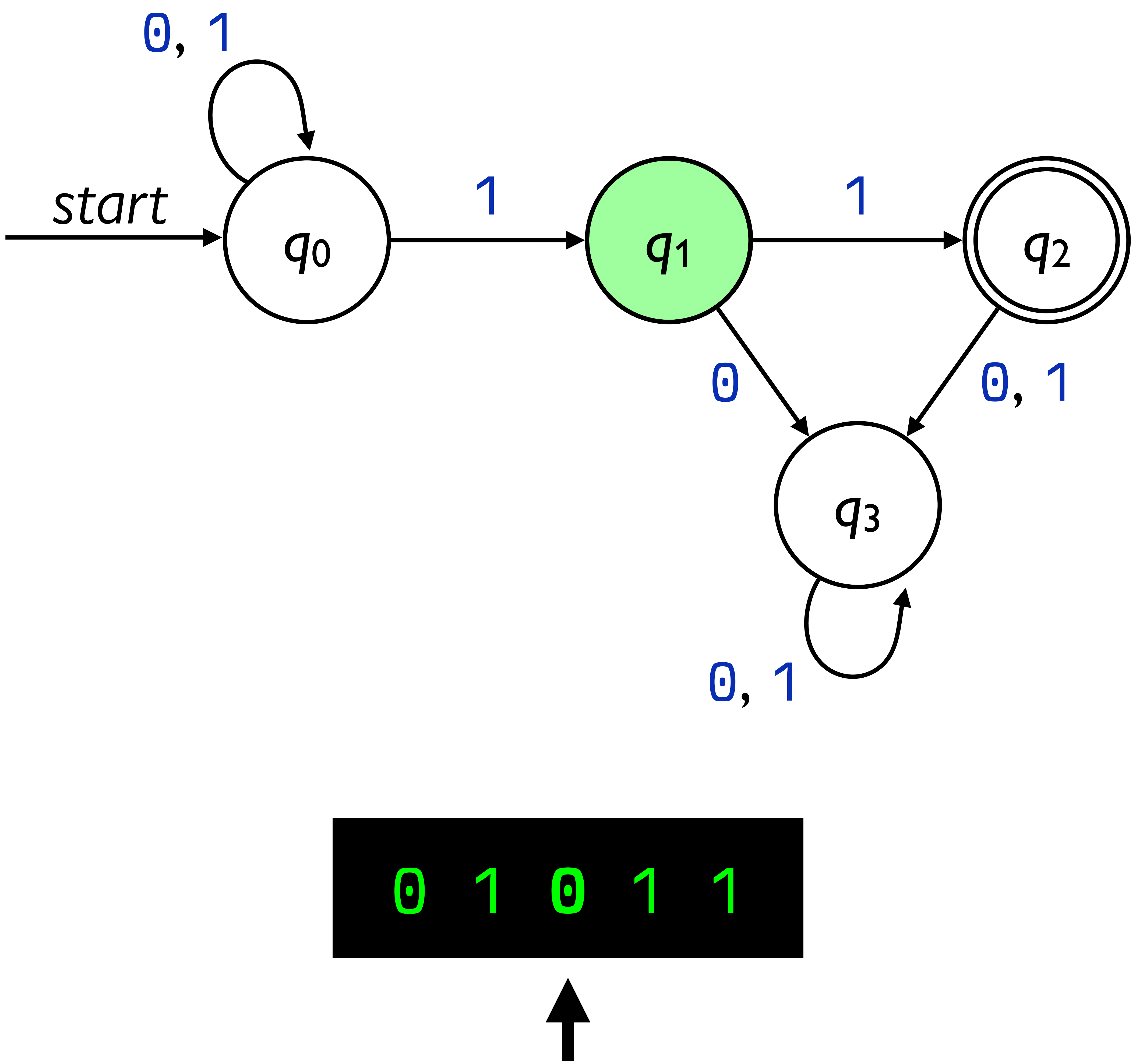


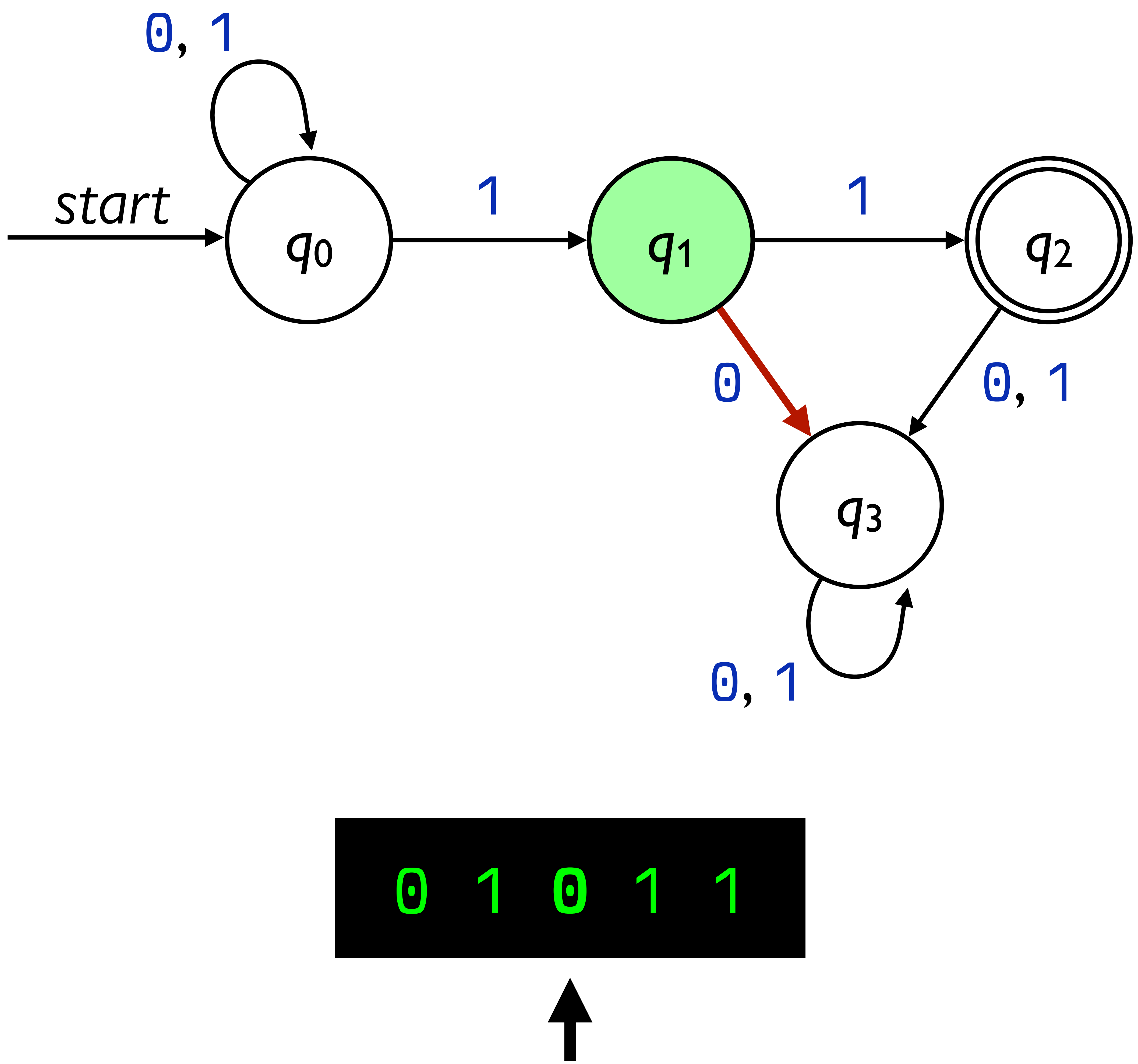
0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

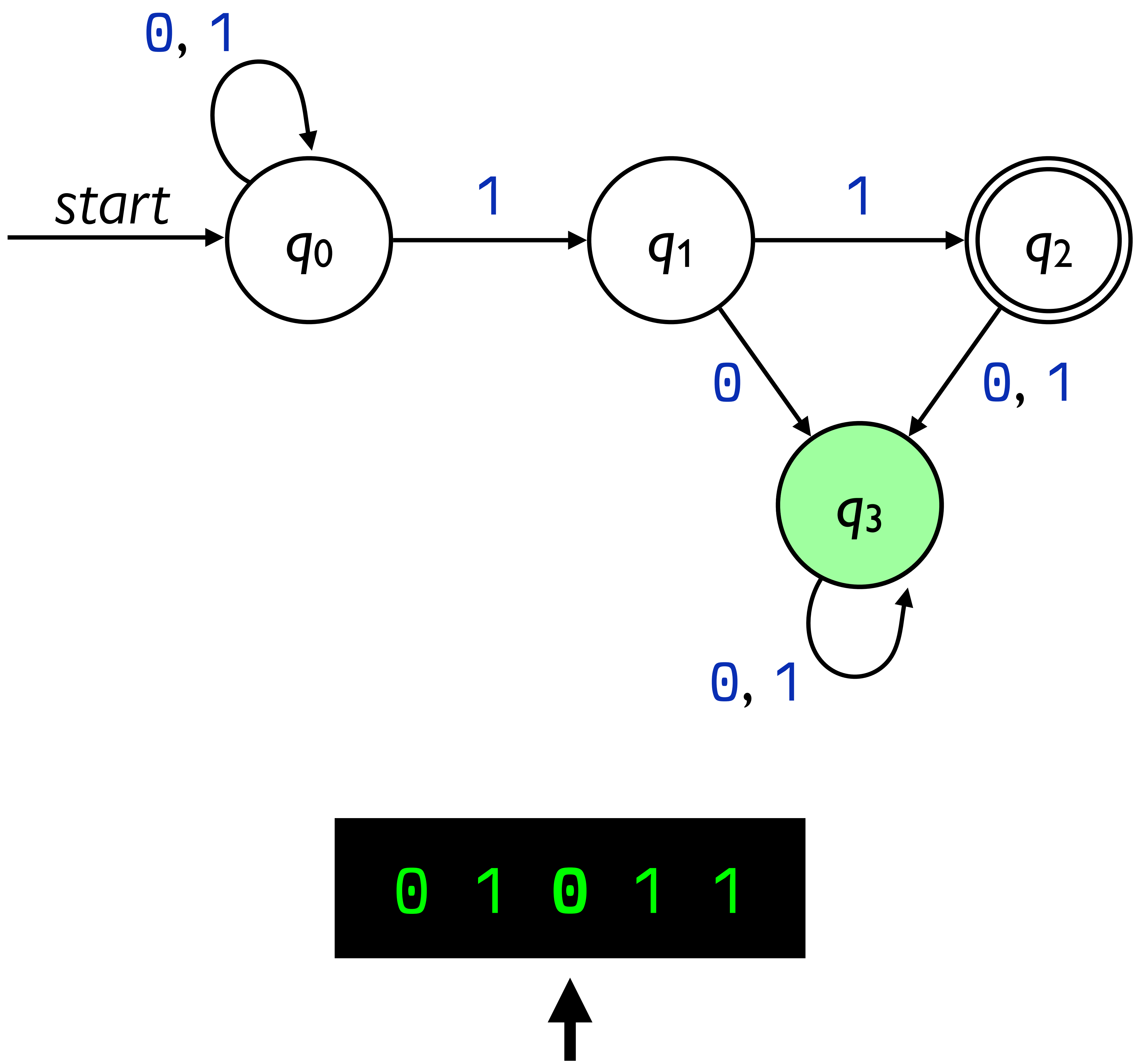

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

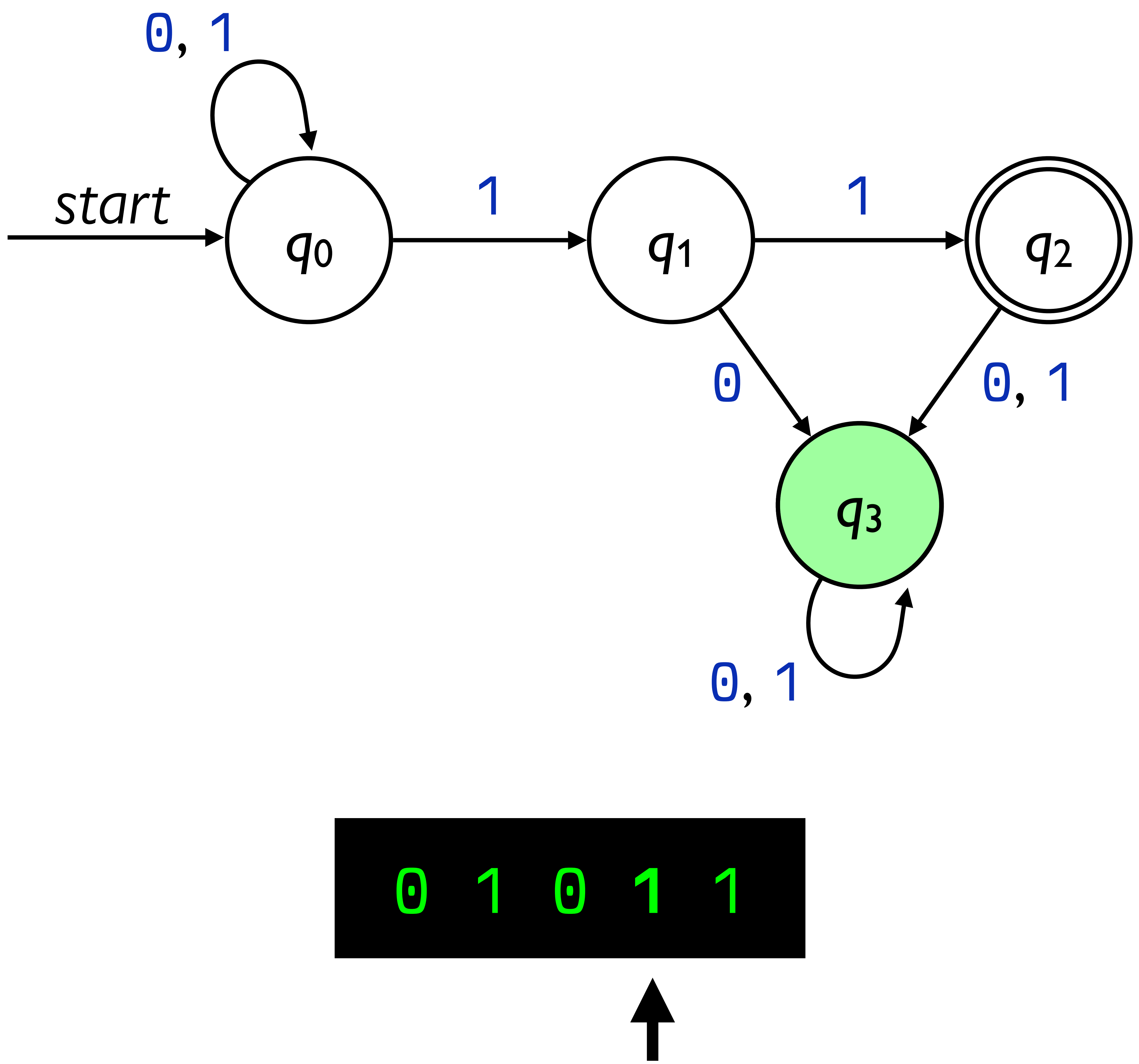

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

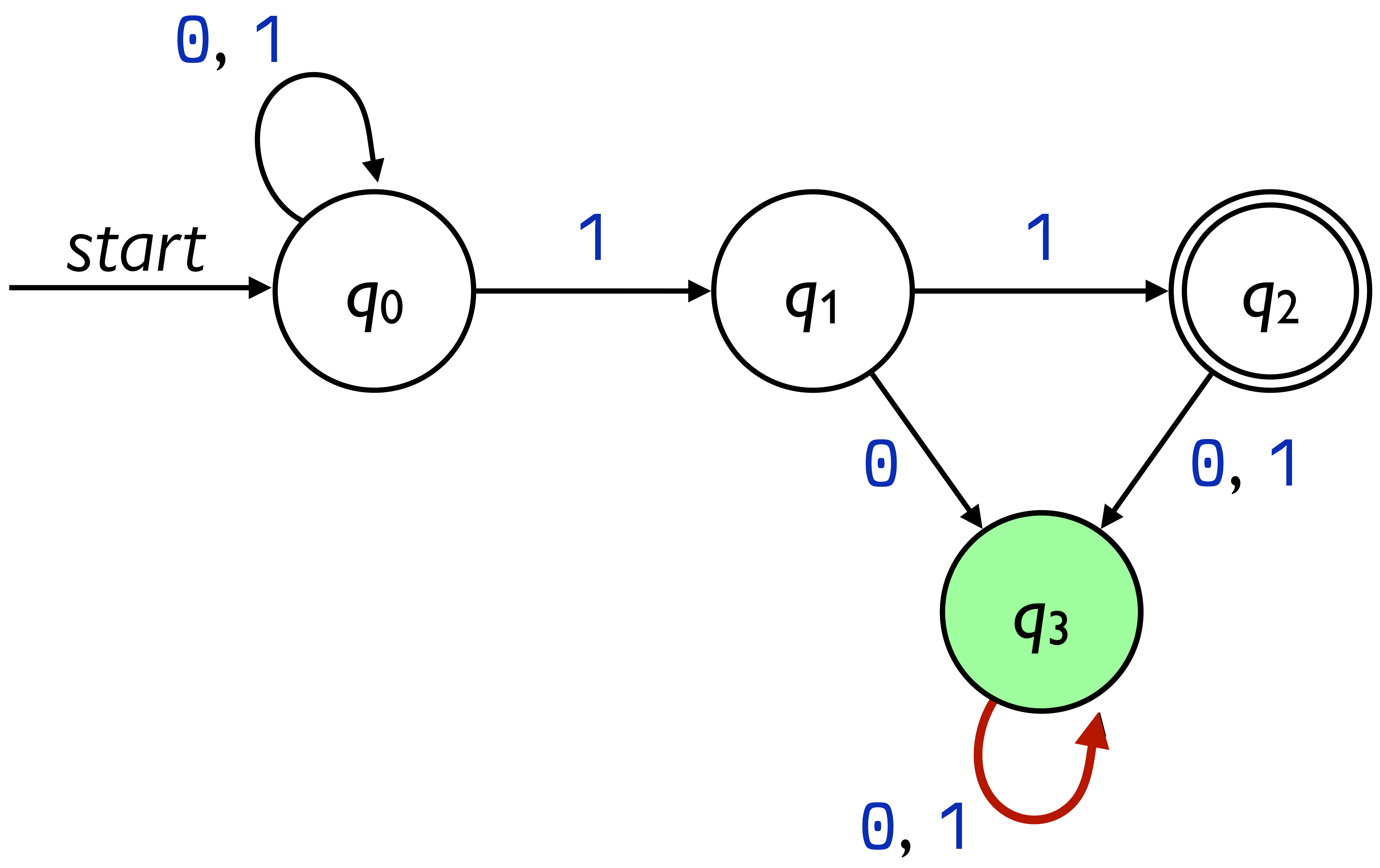

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

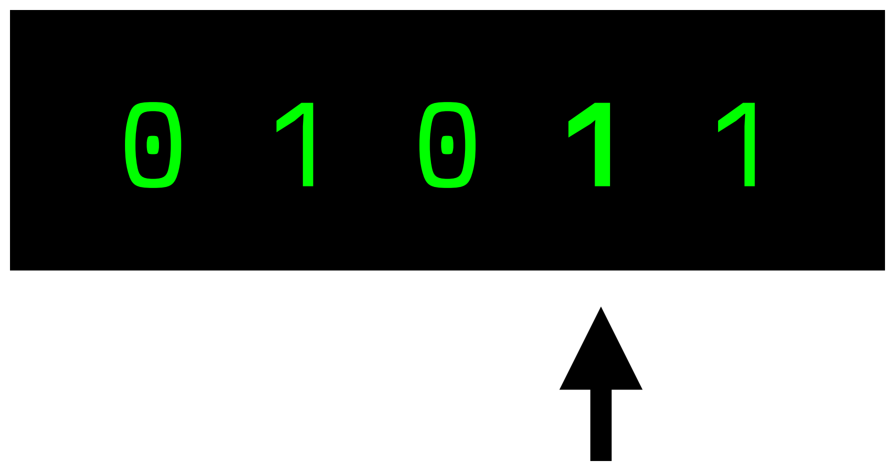


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

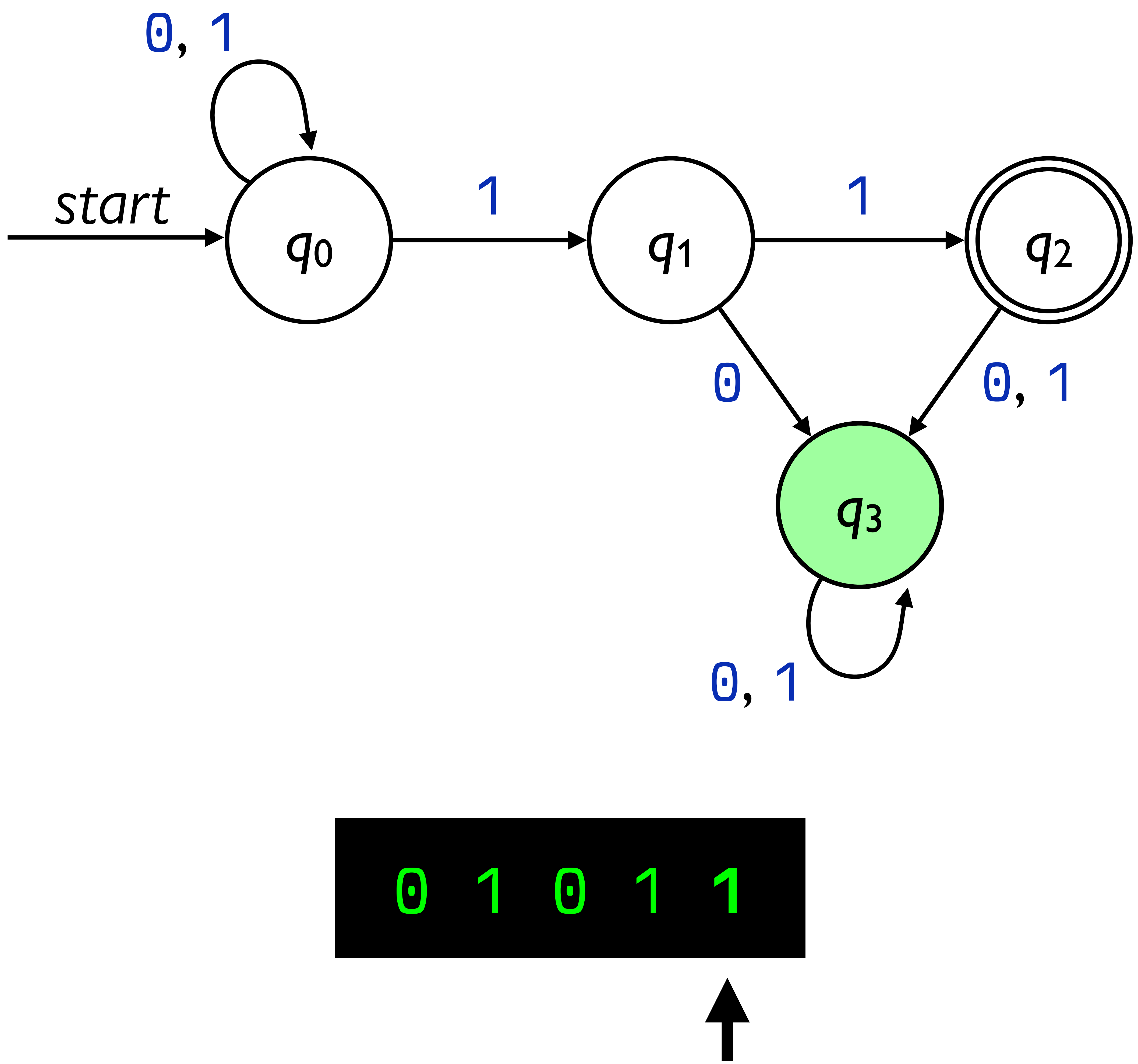

0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1
0 1 0 1 1

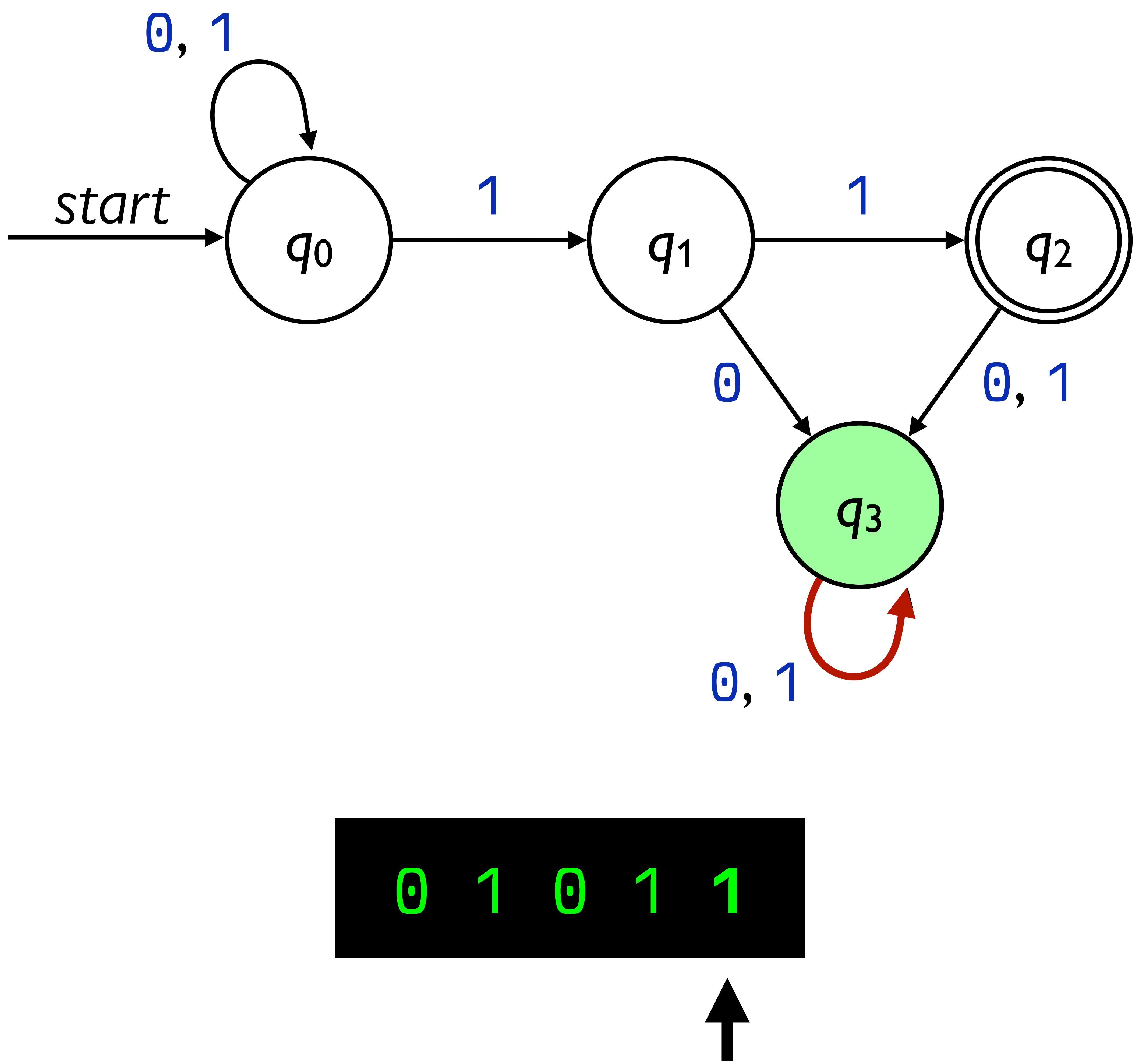


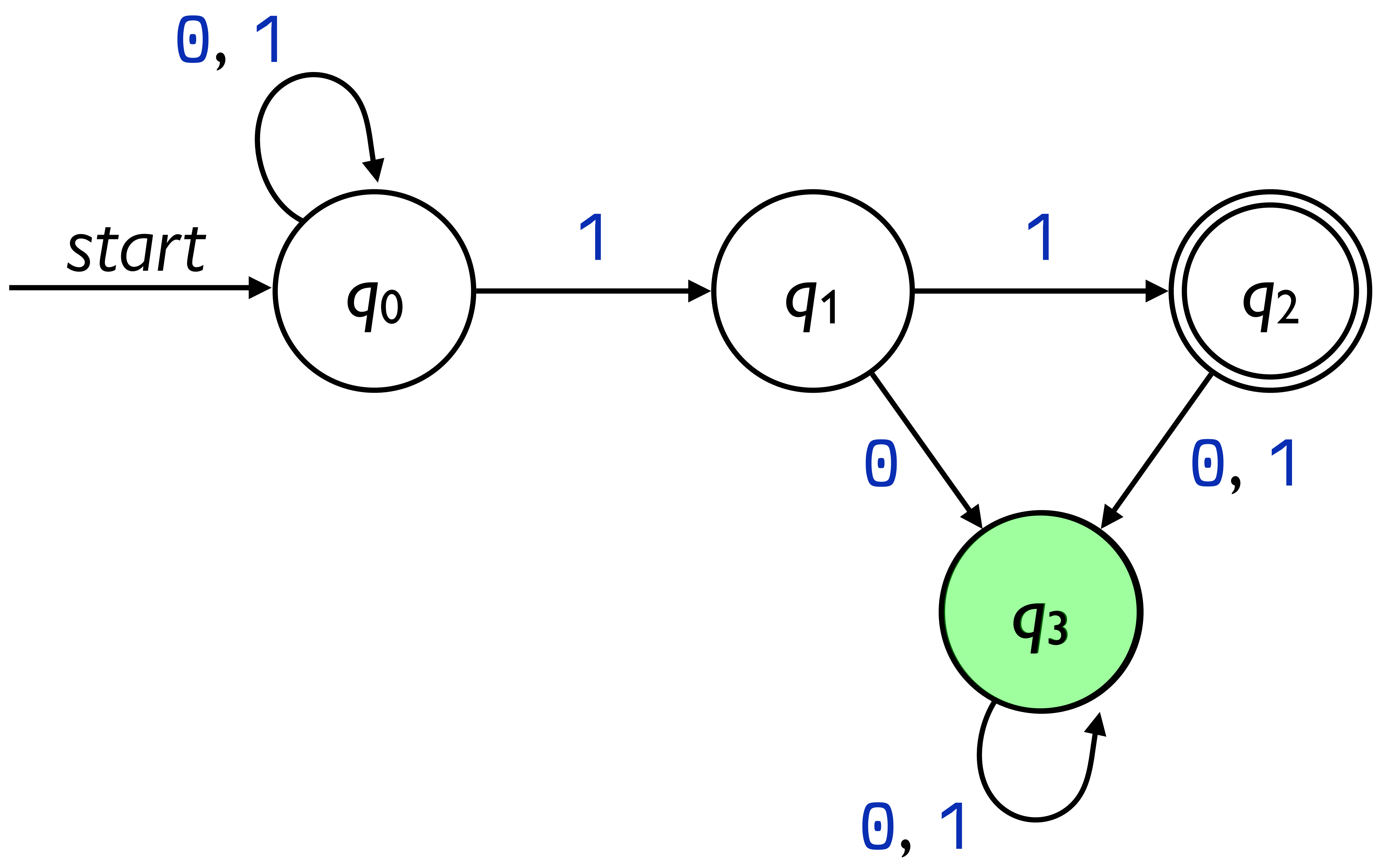
0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1



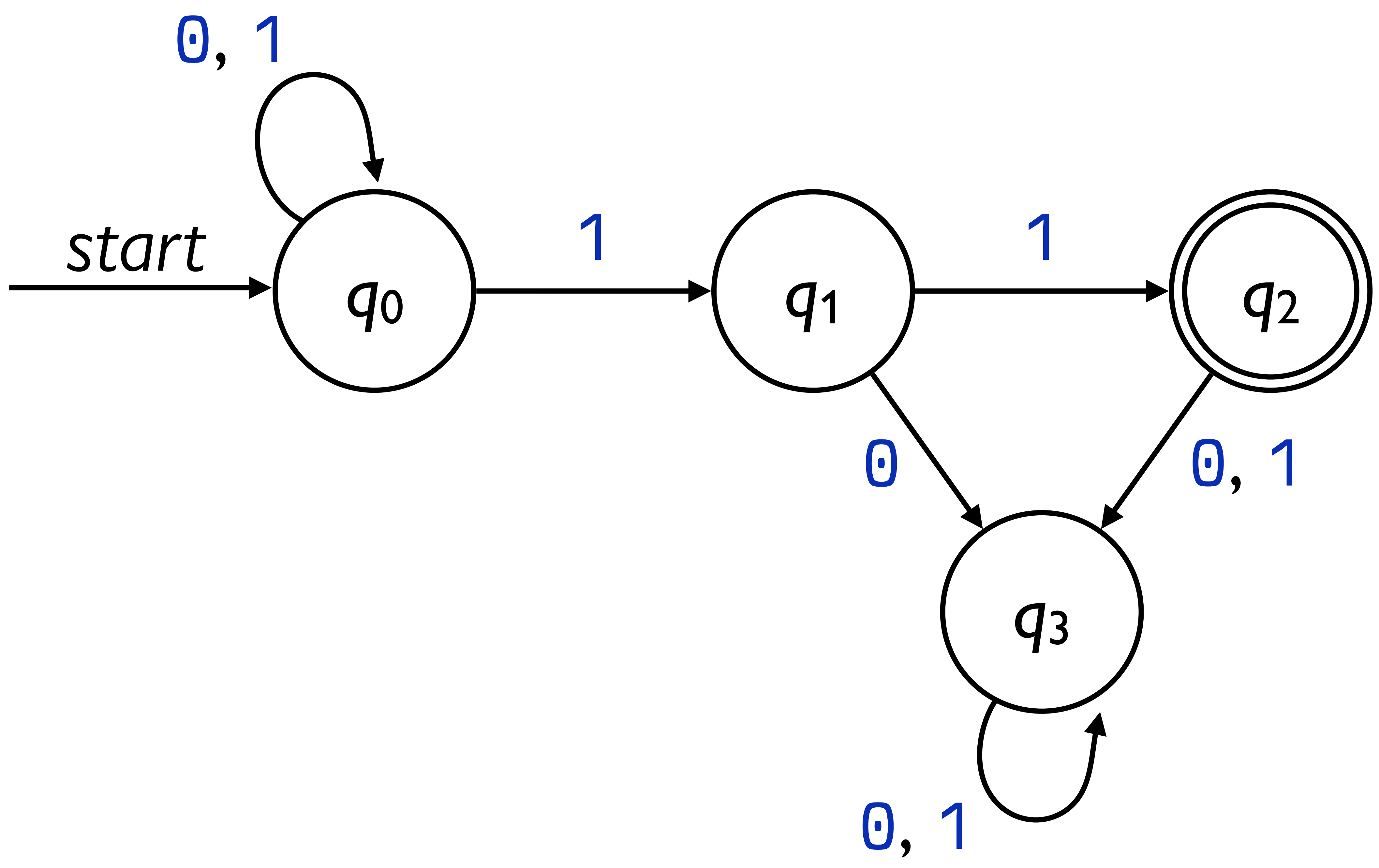
0 1 0 1 1


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

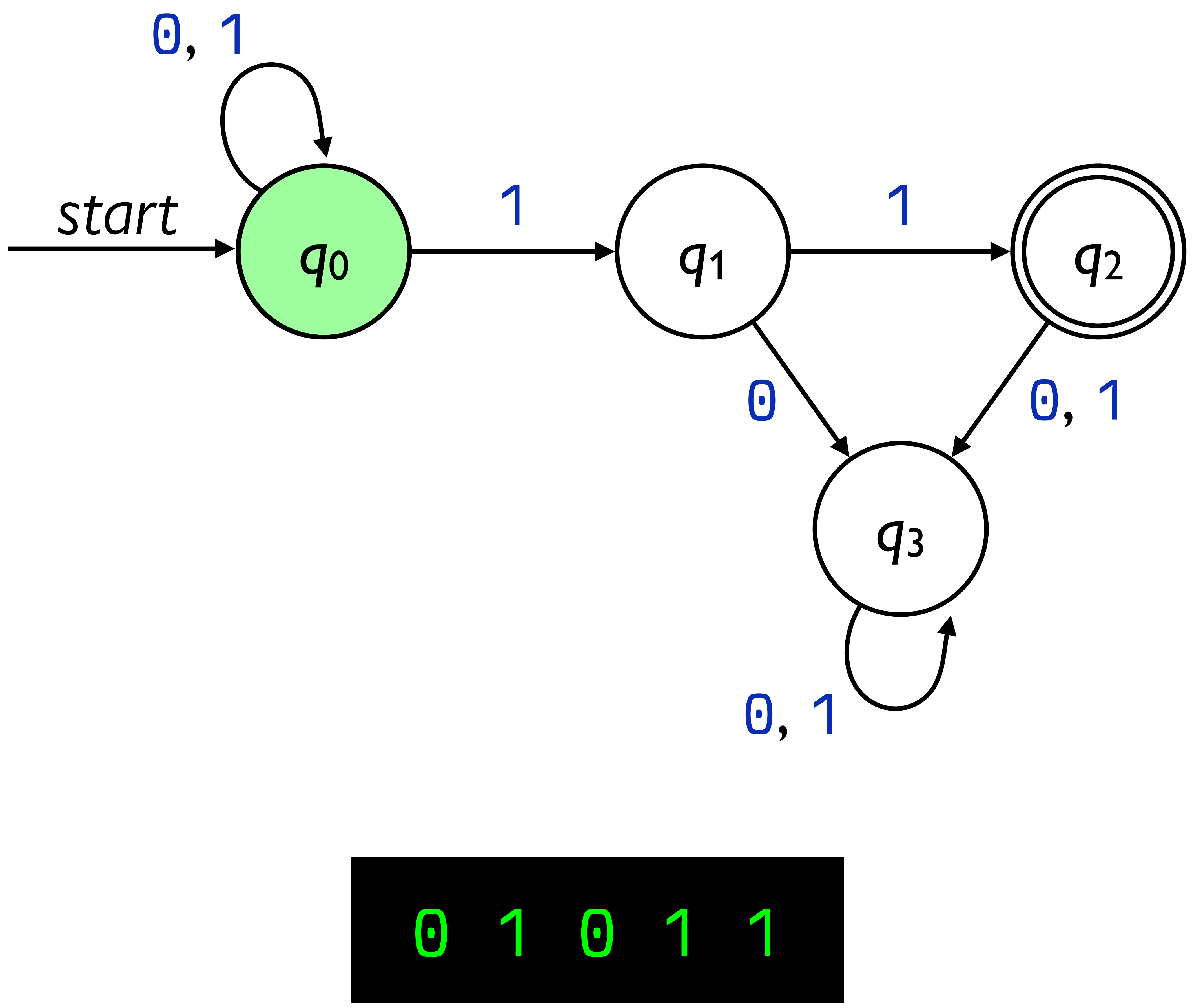

0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1
0 1 0 1 1

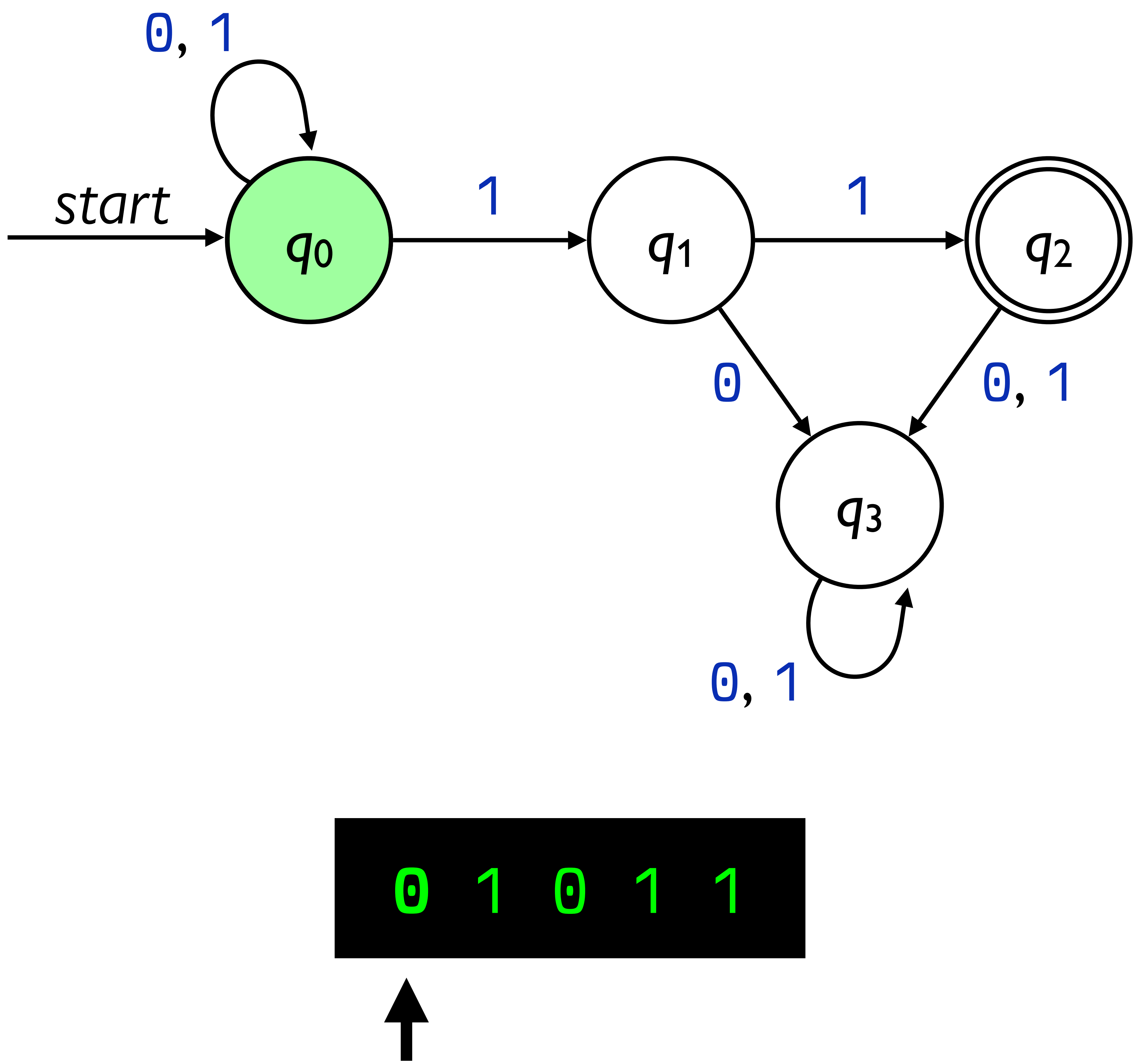


0 1 0 1 1

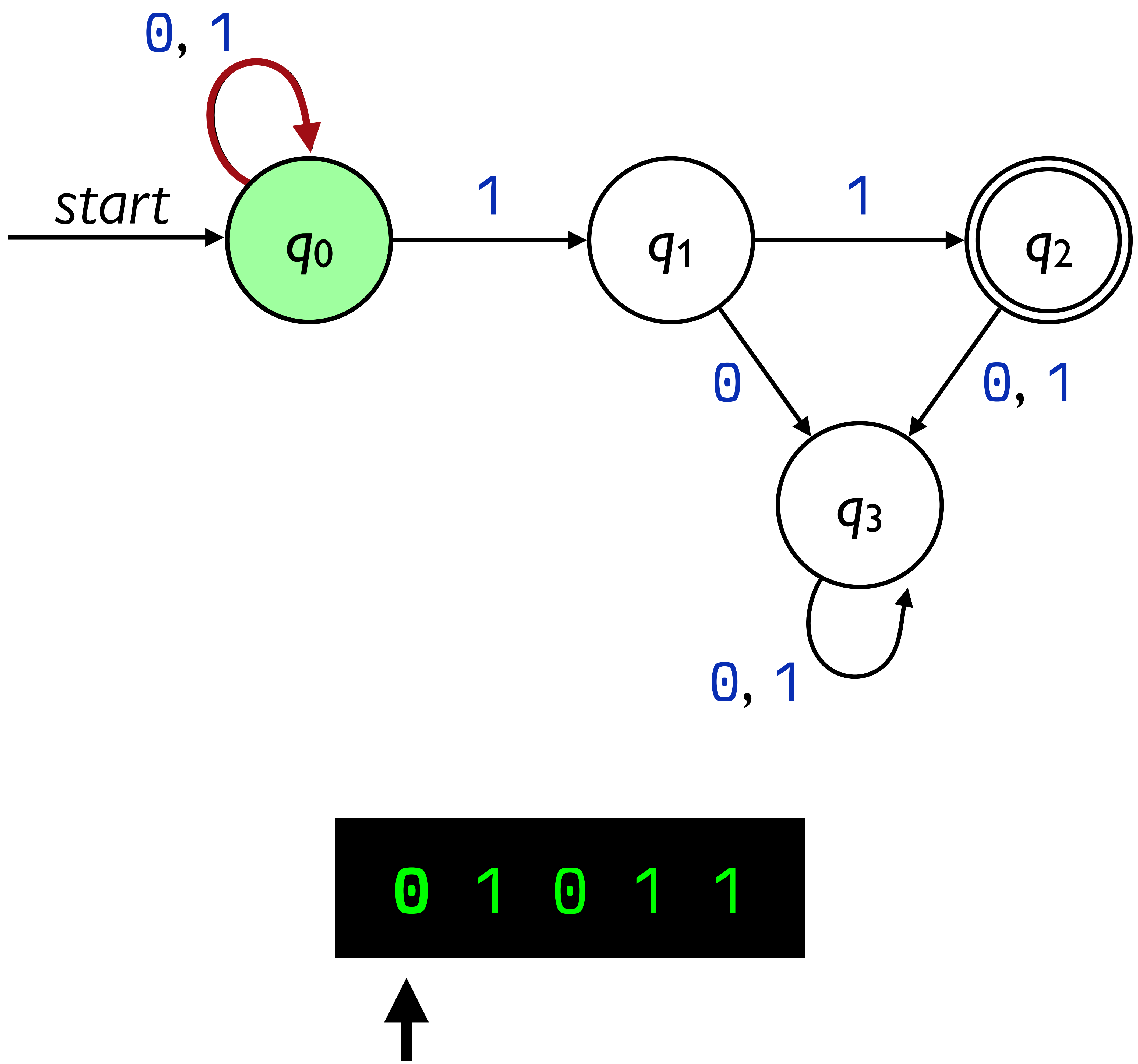


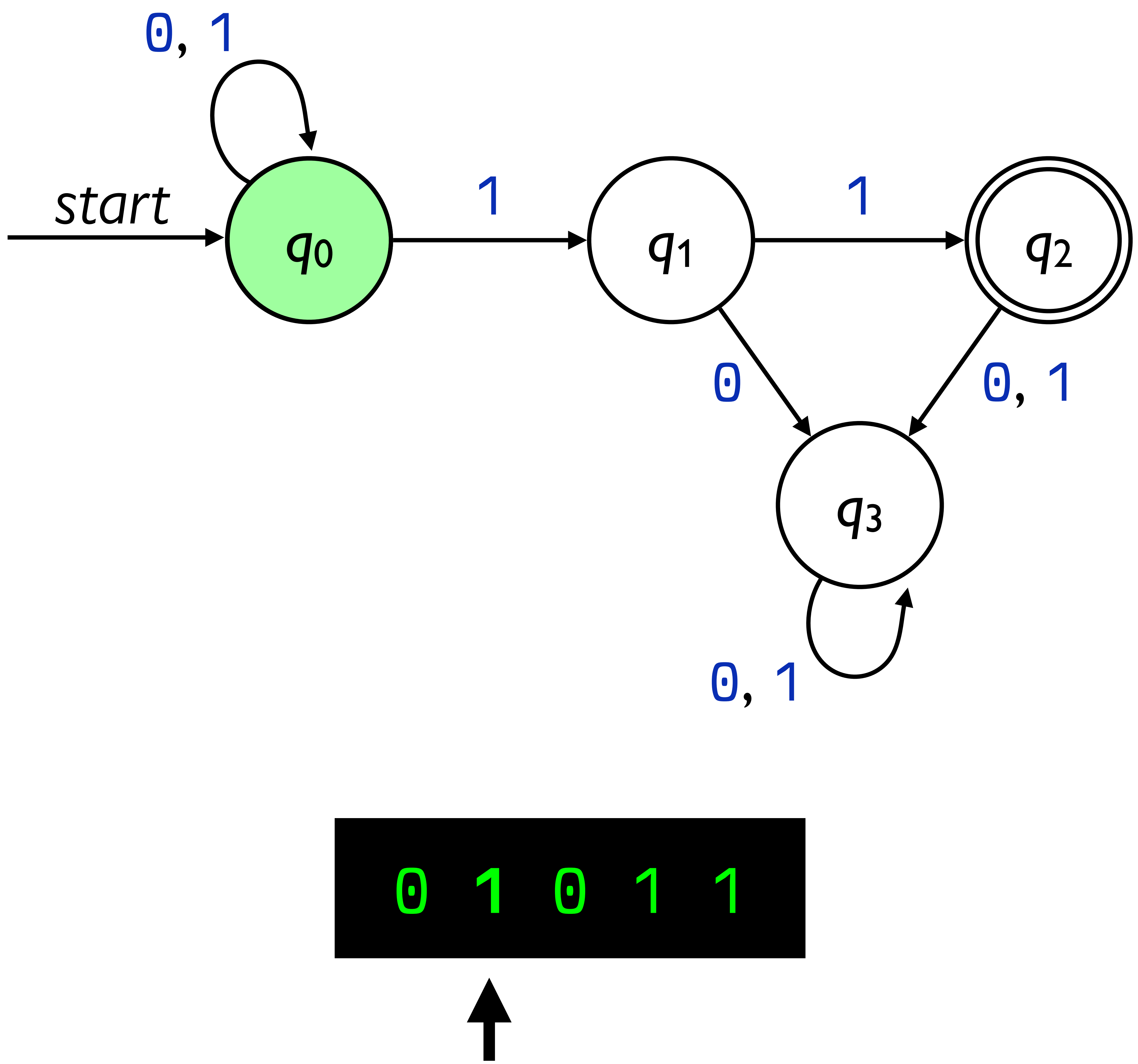
`0 1 0 1 1`

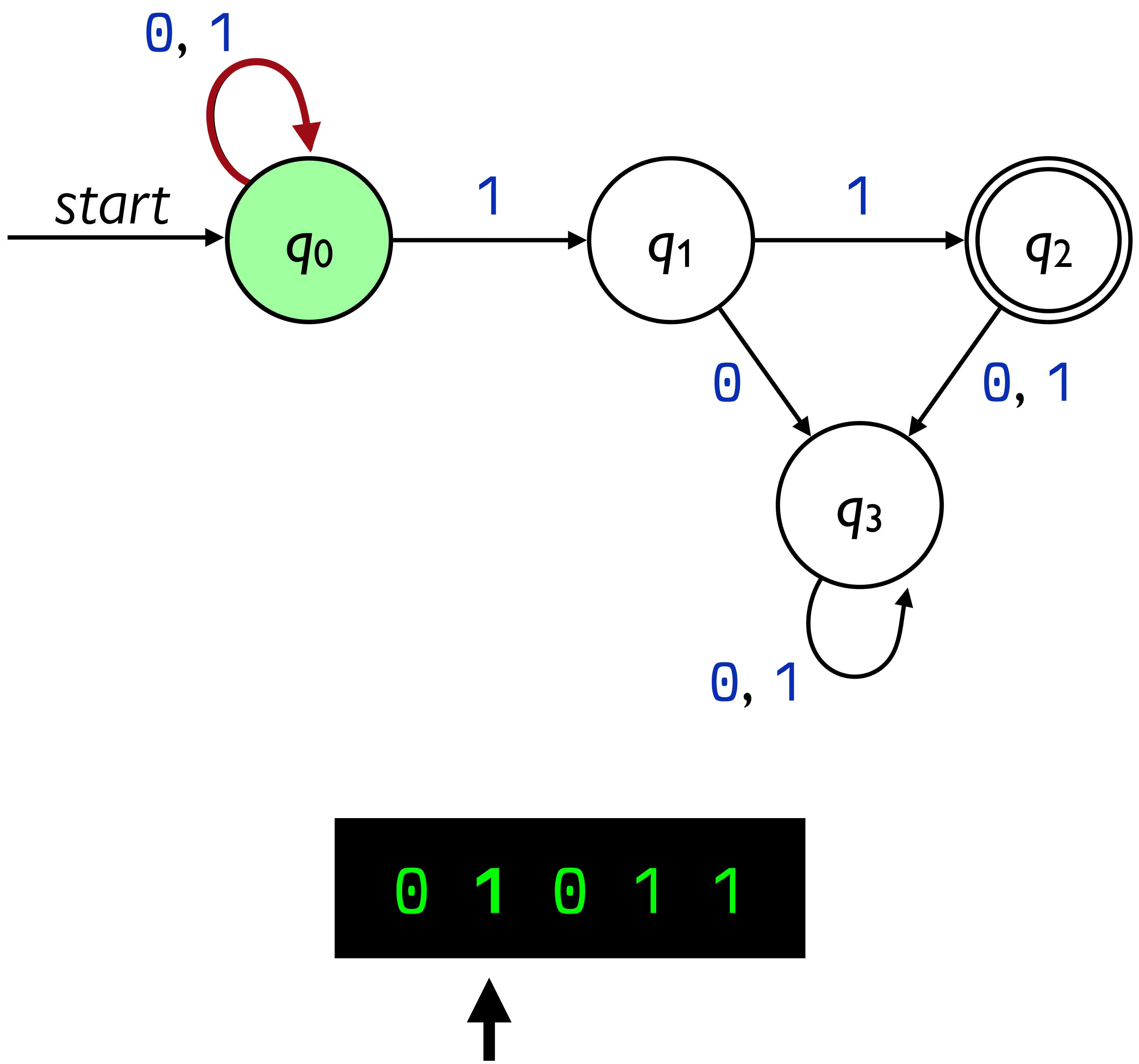

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1



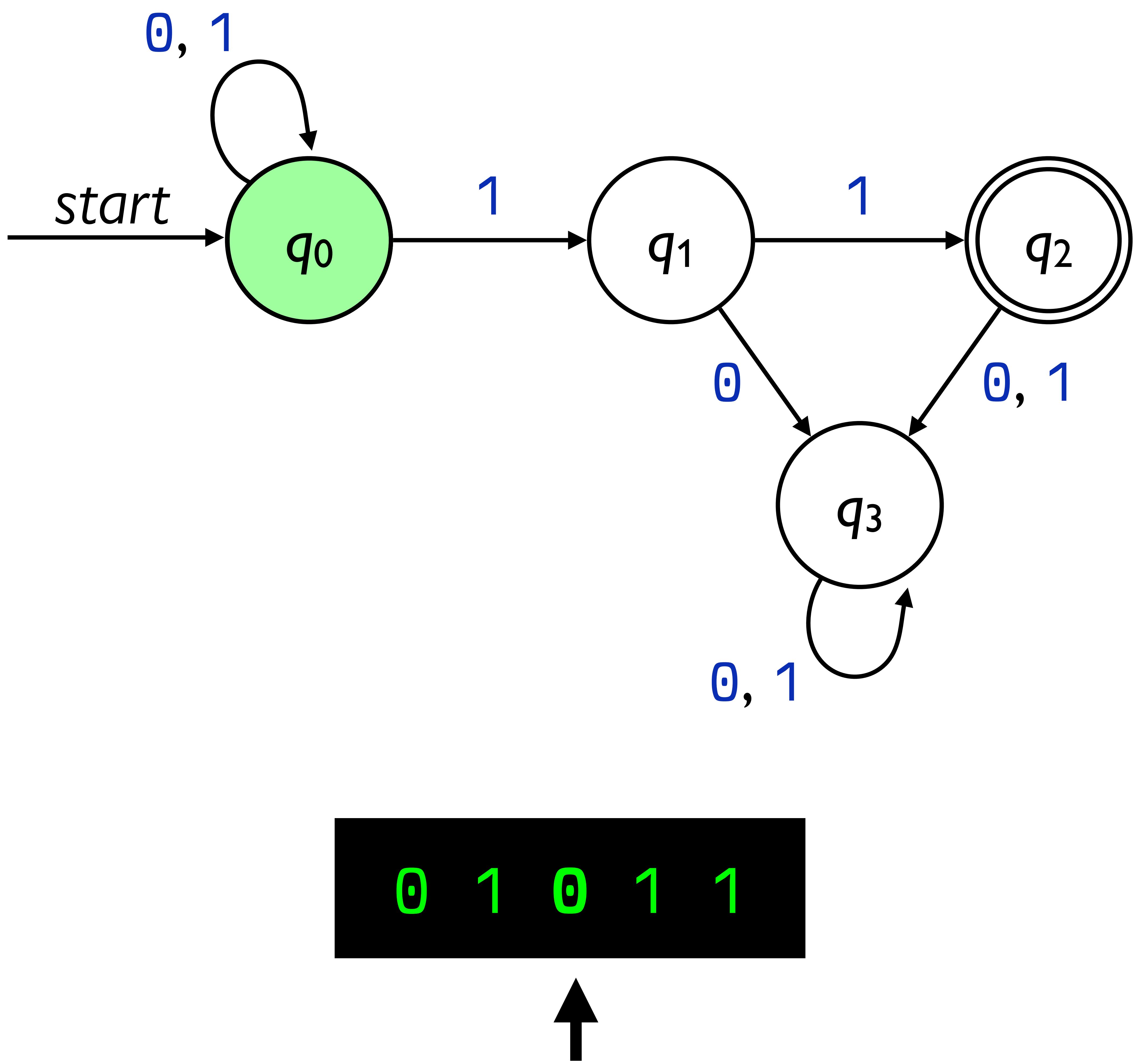
0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1


0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1
0 1 0 1 1

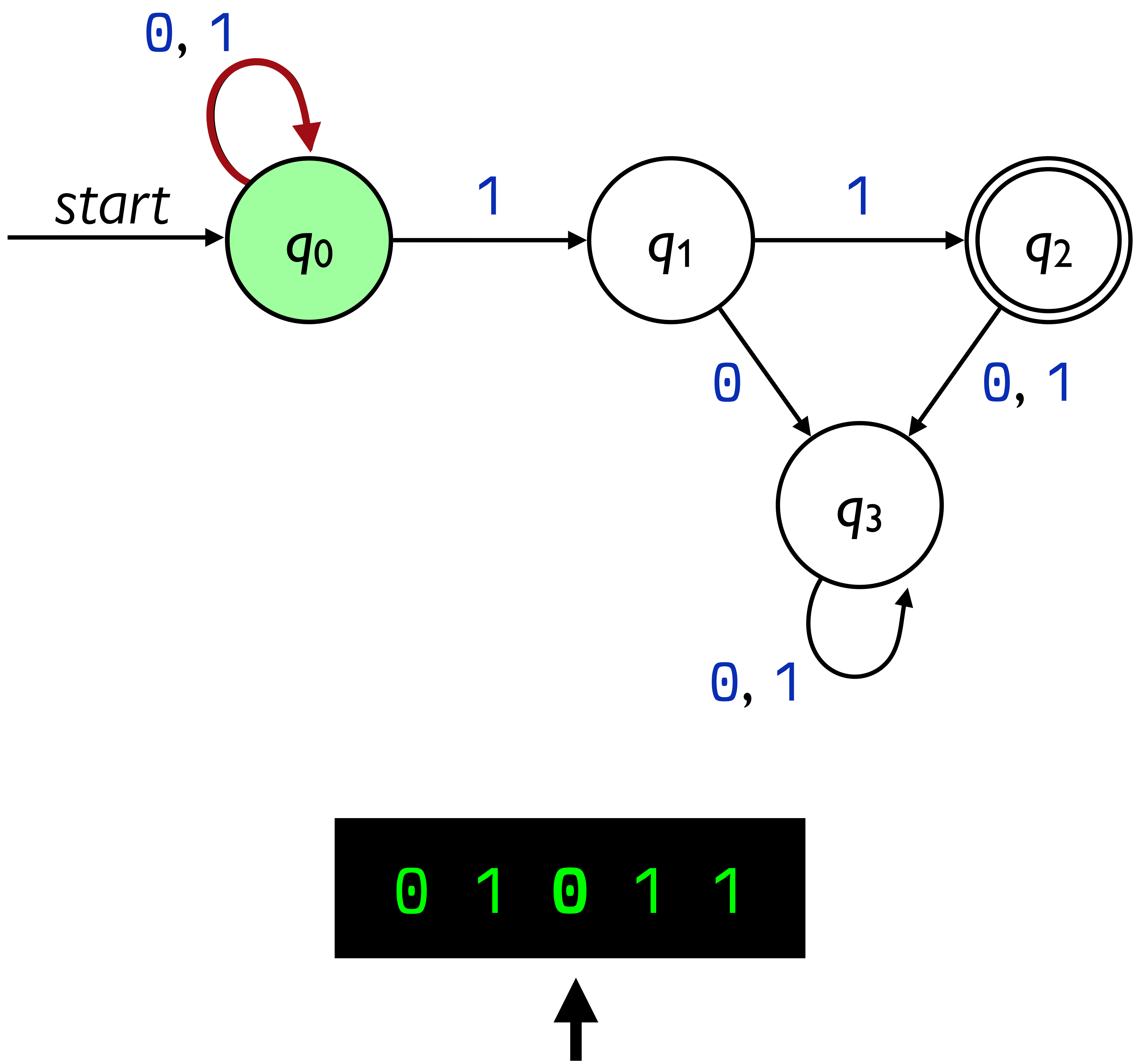

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

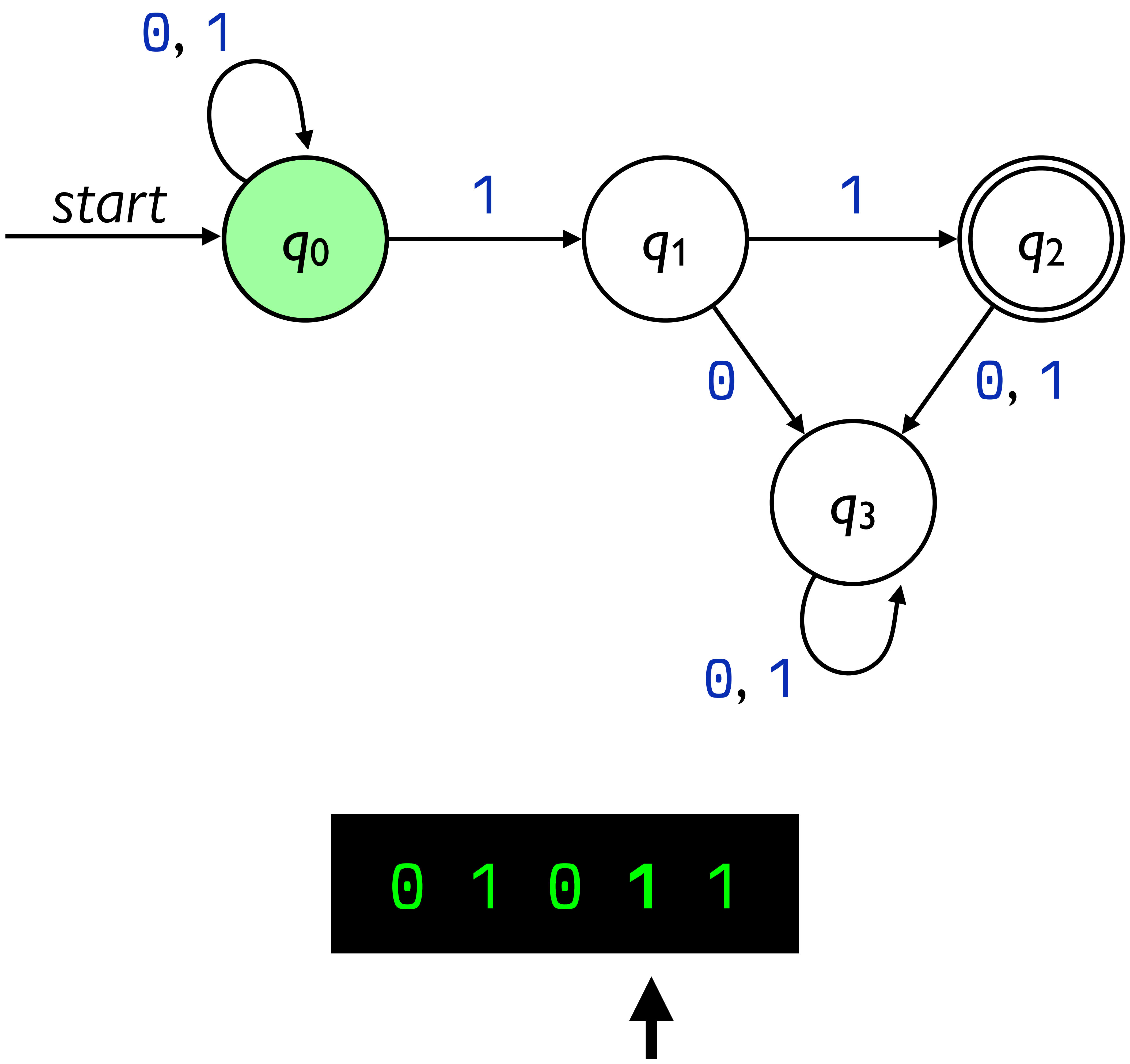


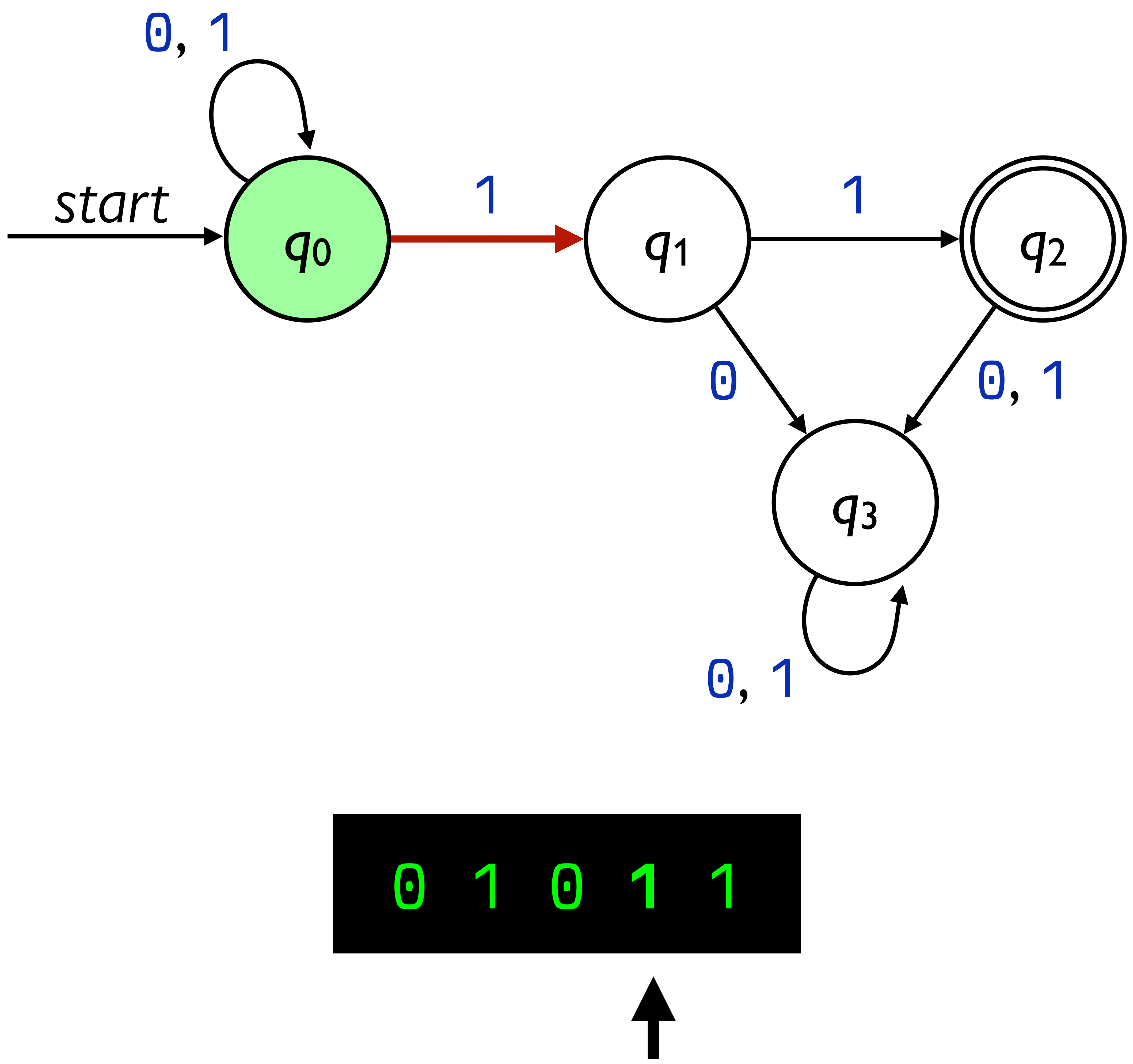
0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

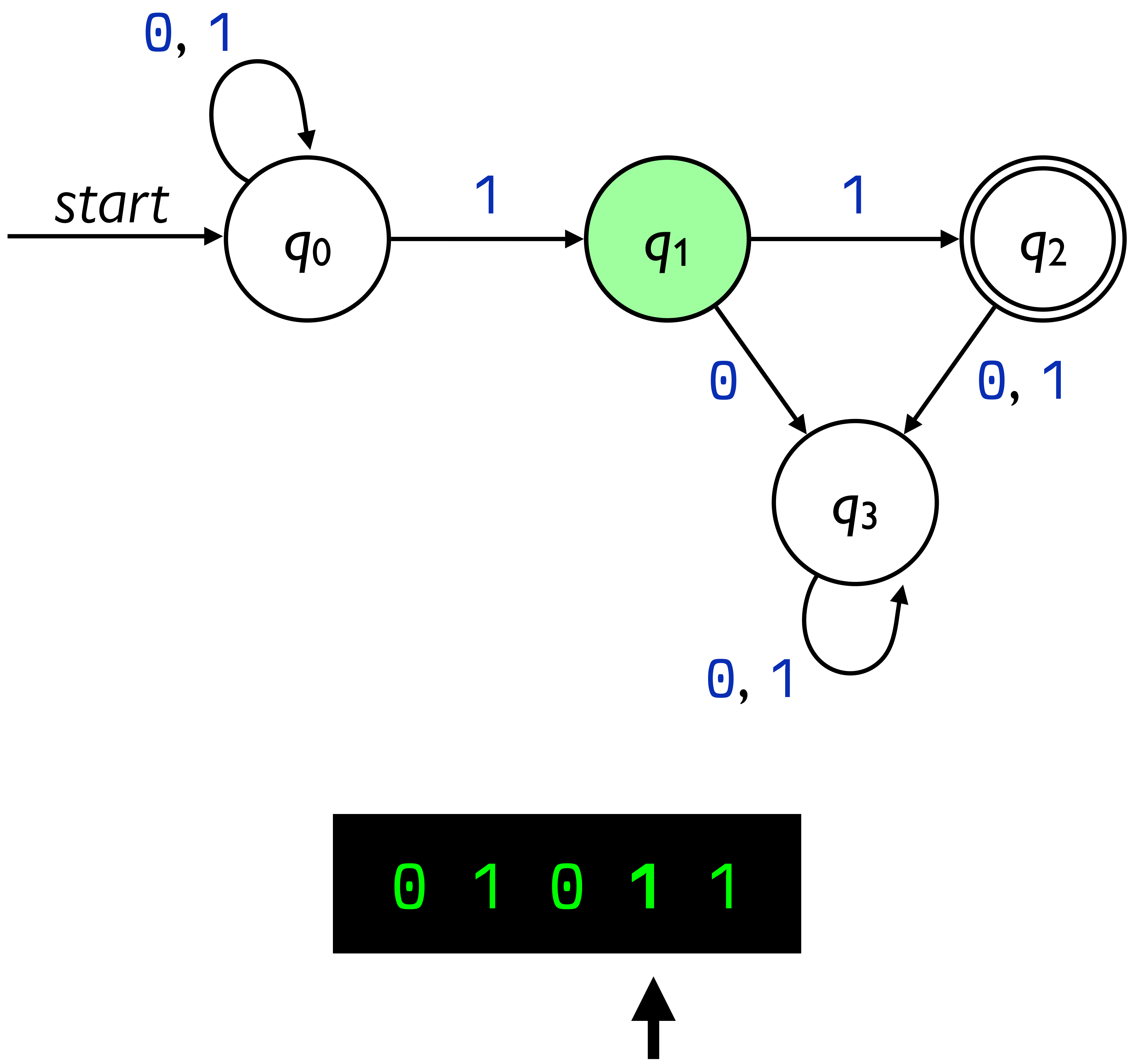


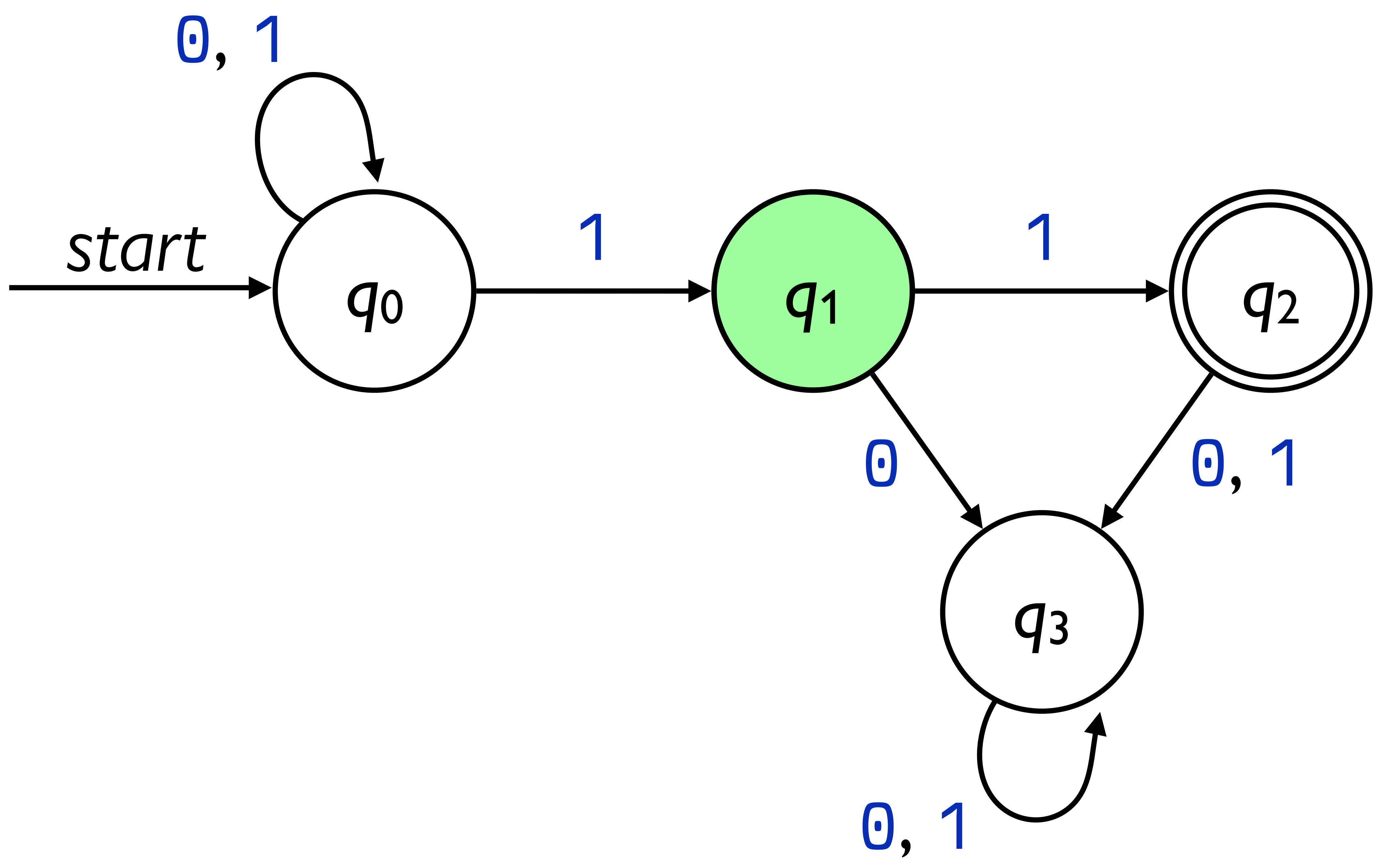
0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

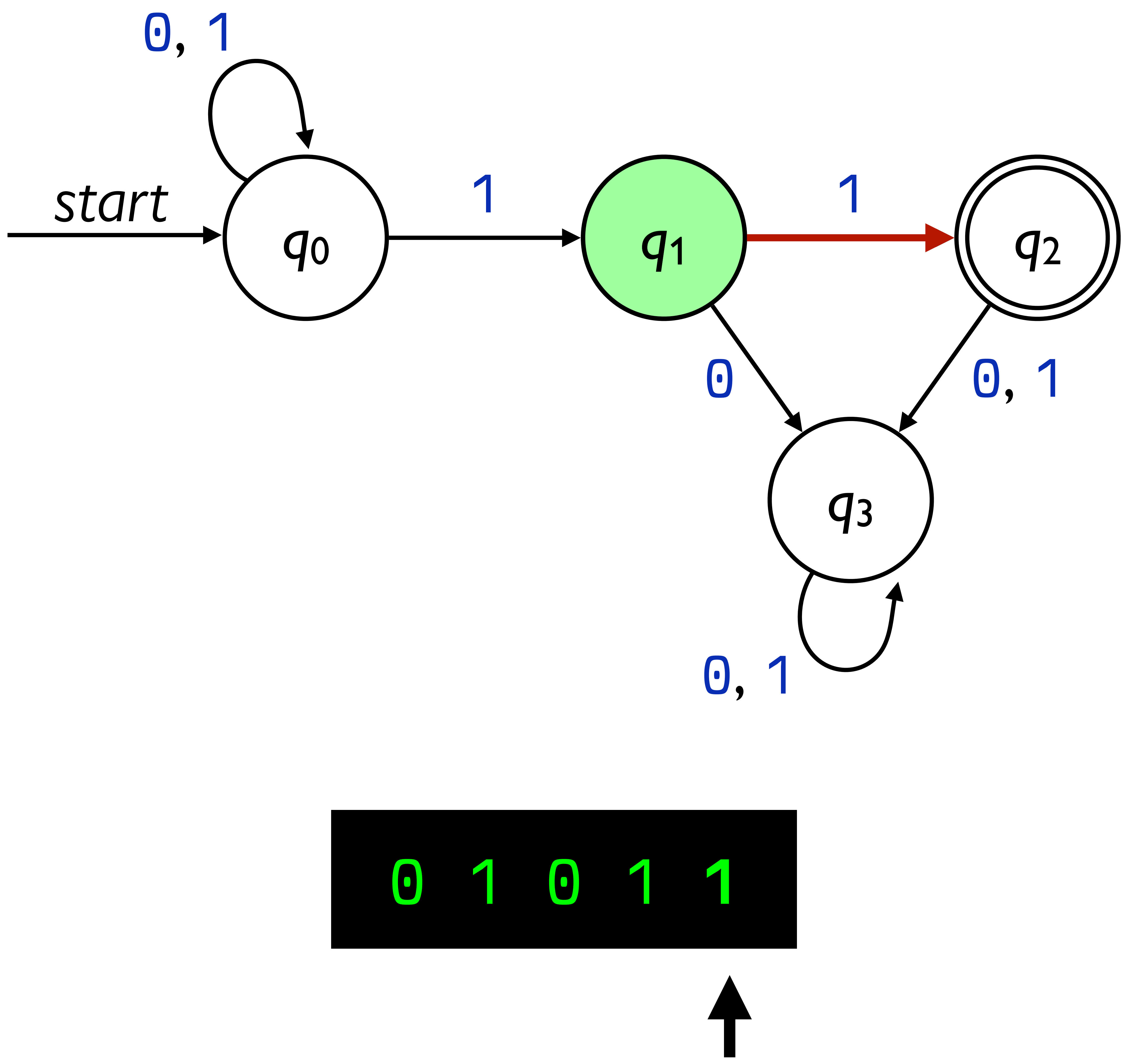

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

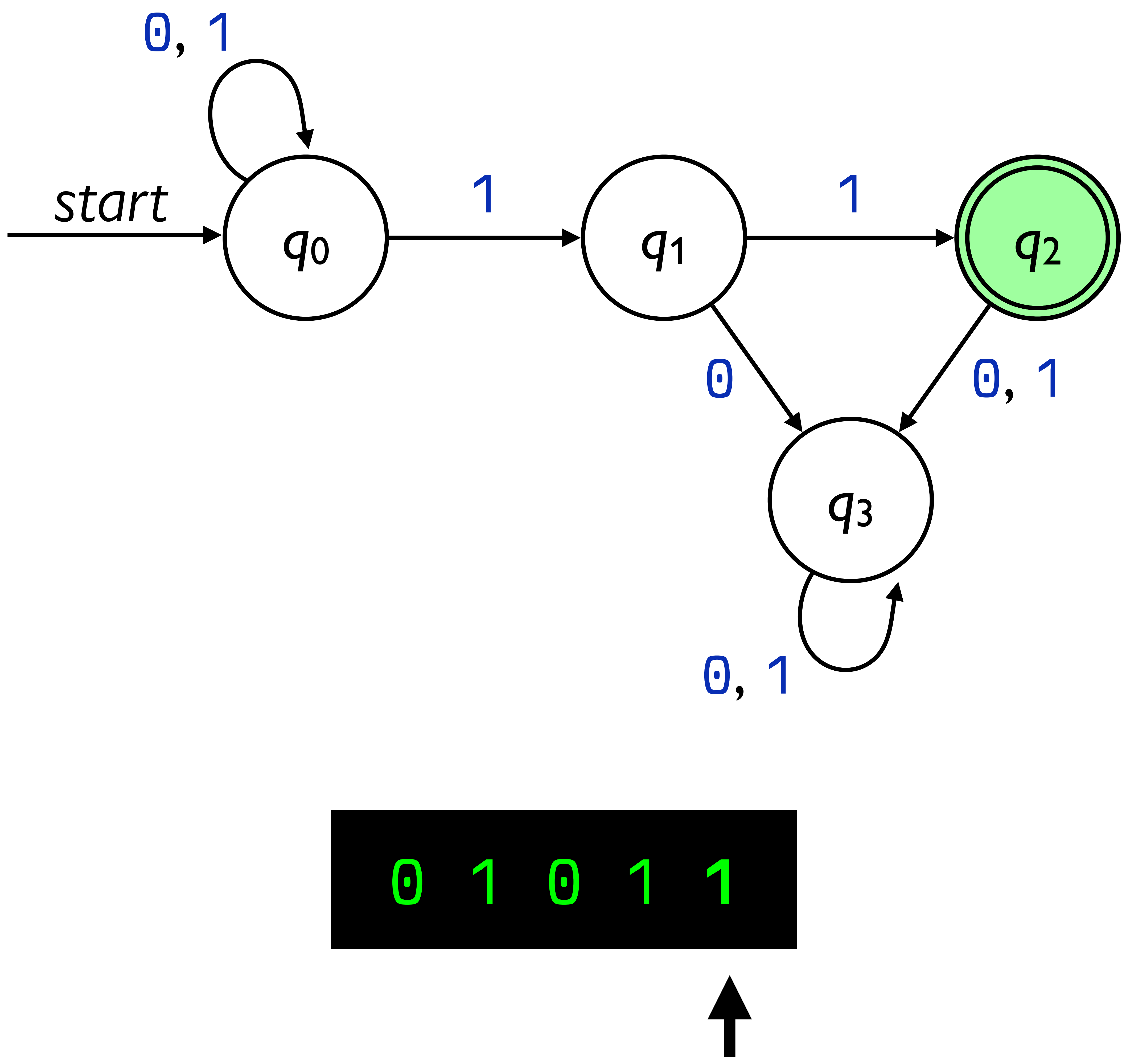

0, 1
start
q_0
1
q_1
1
q_2
0
0, 1
q_3
0, 1
0 1 0 1 1


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

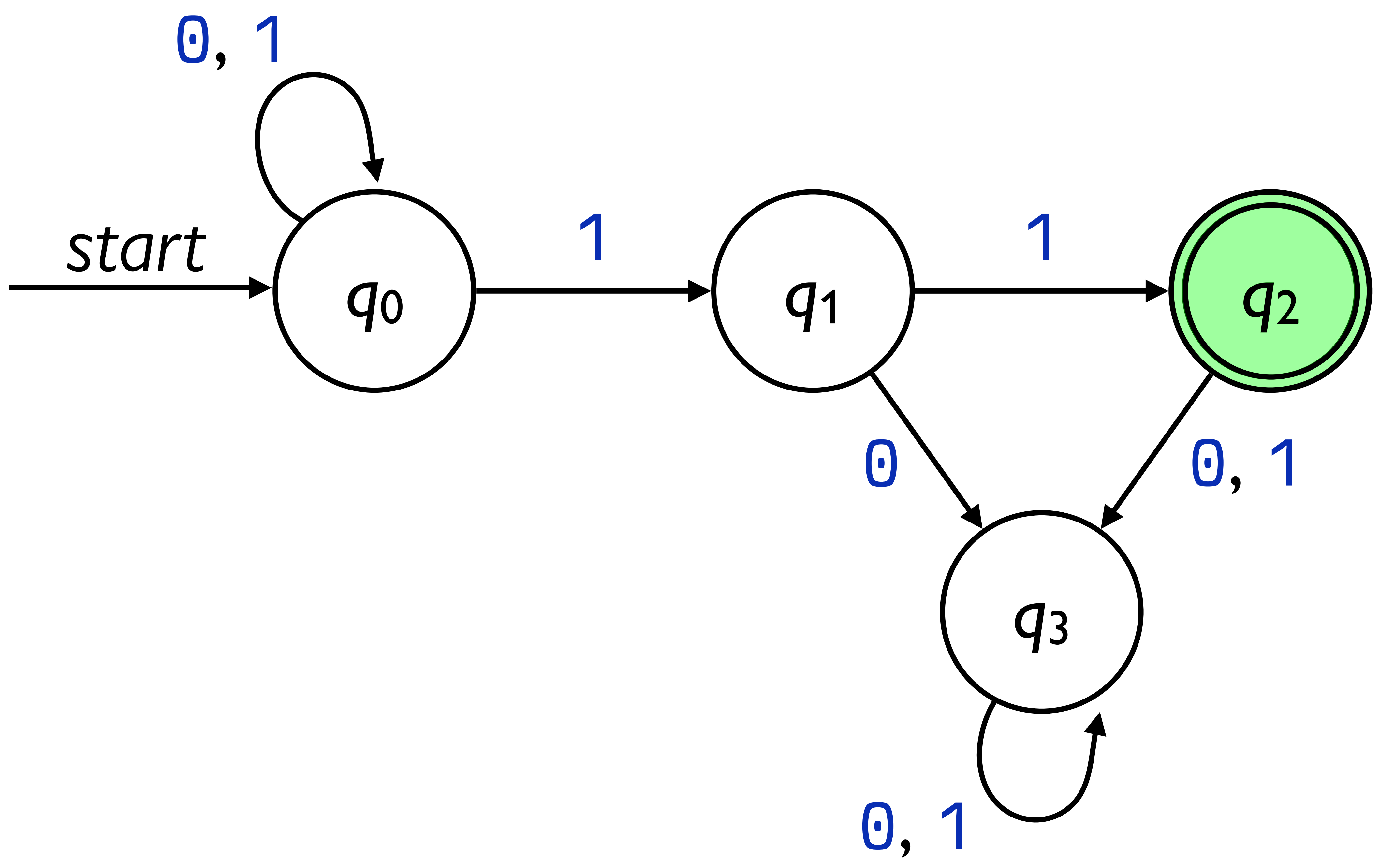

0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1


0 1 0 1 1


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

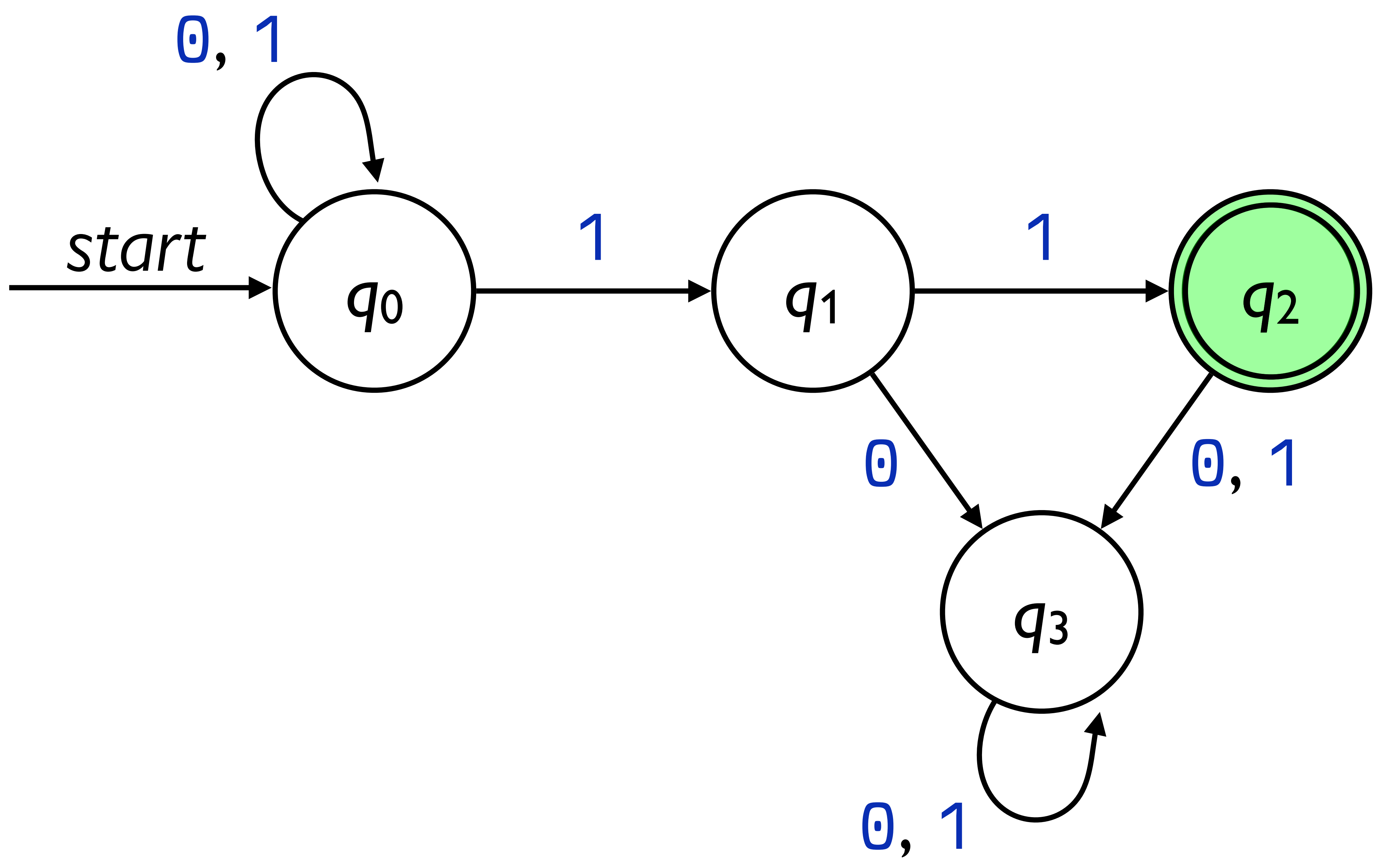


0, 1
start
q0
1
q1
1
q2
0
0, 1
q3
0, 1
0 1 0 1 1

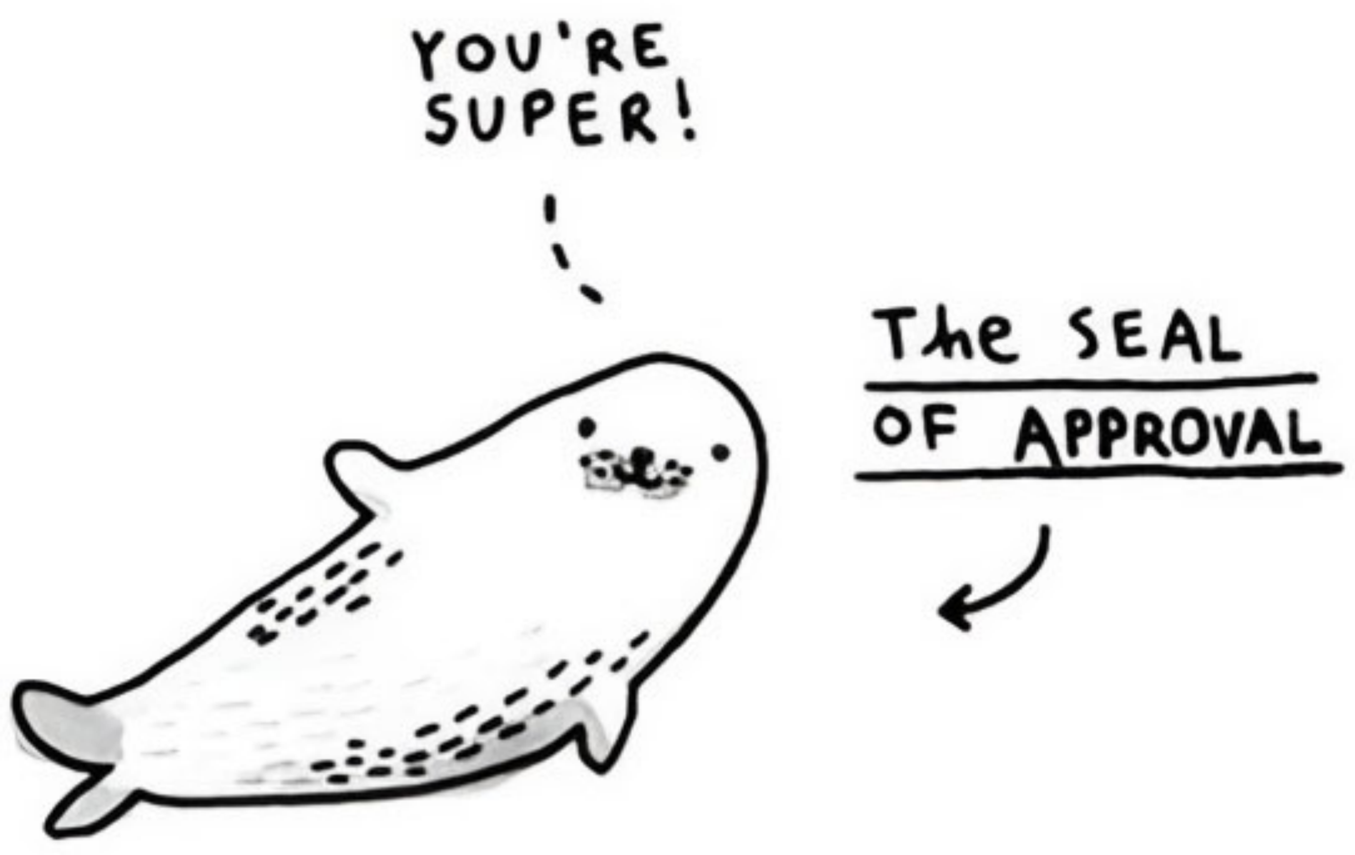


0 1 0 1 1

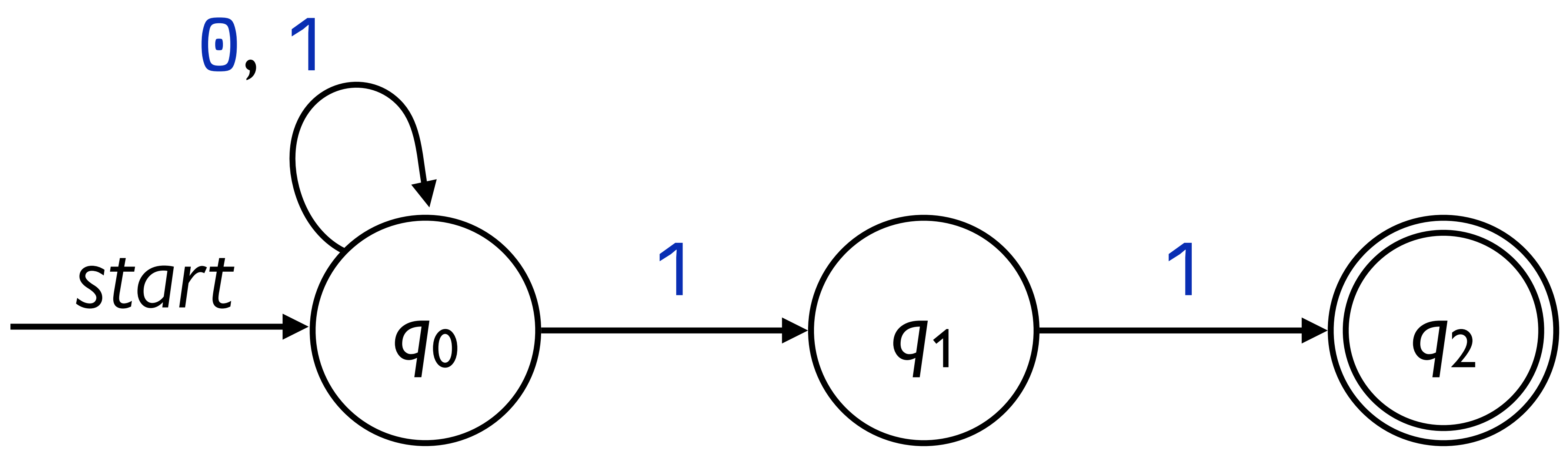
Illustration by
Gemma Correll



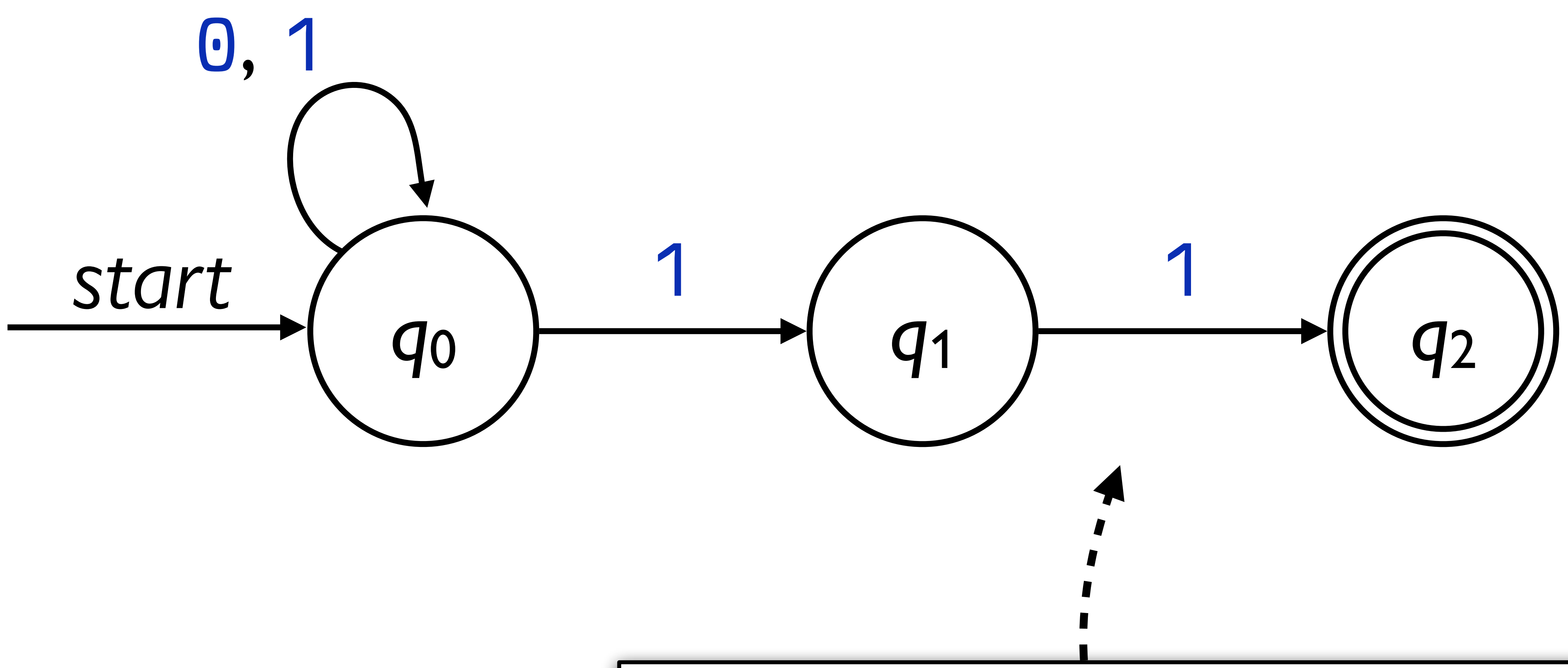
`0 1 0 1 1`



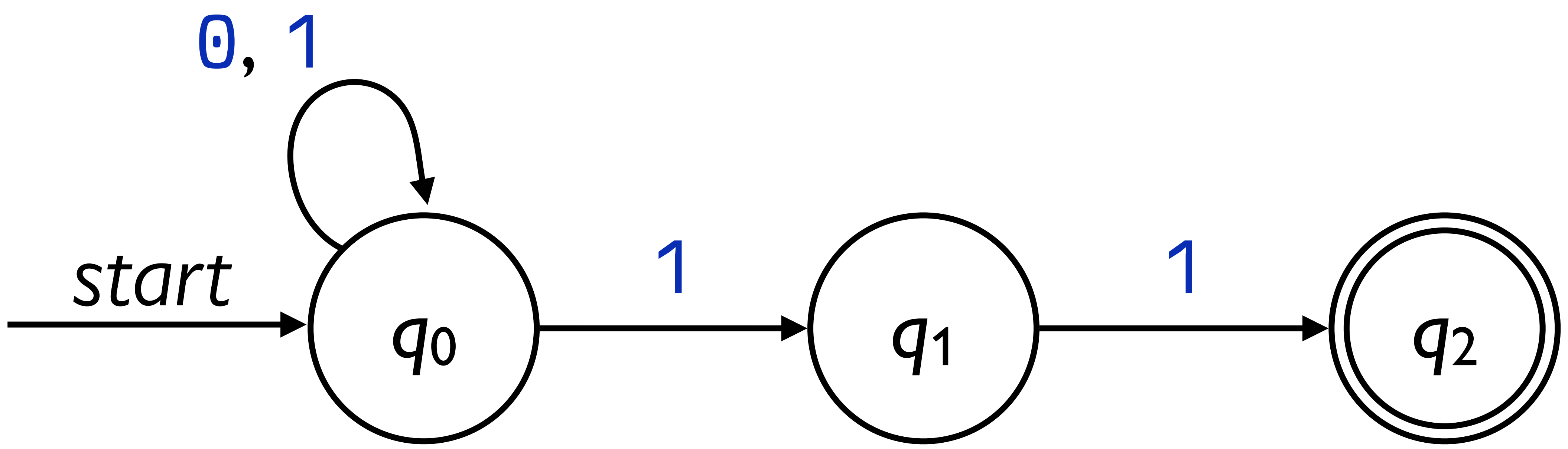
A more complex NFA



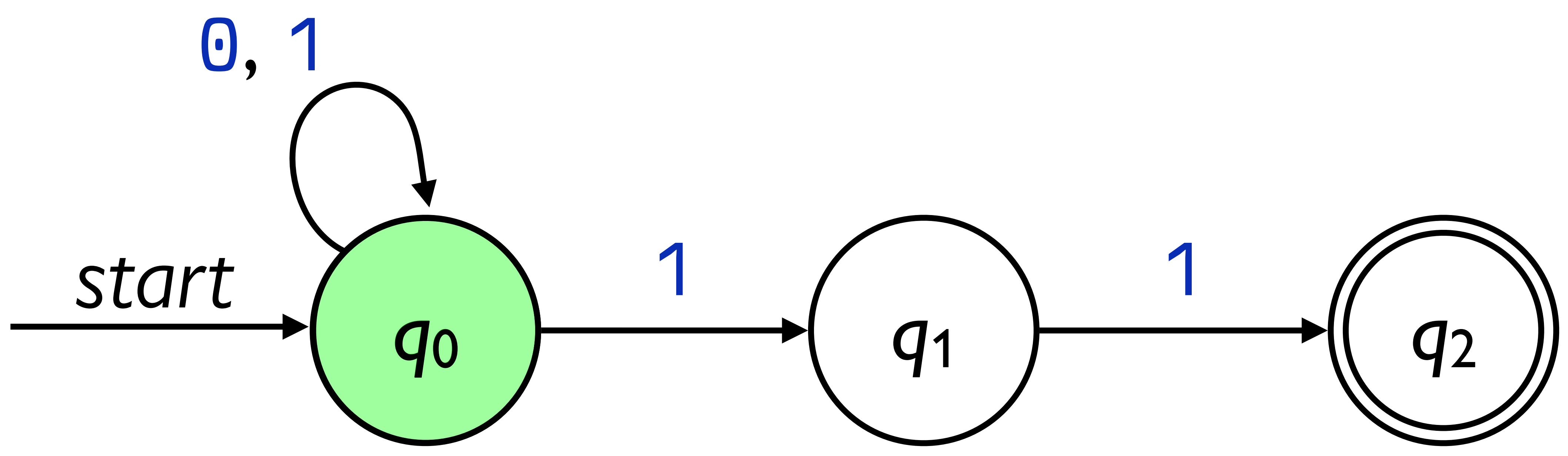
A more complex NFA



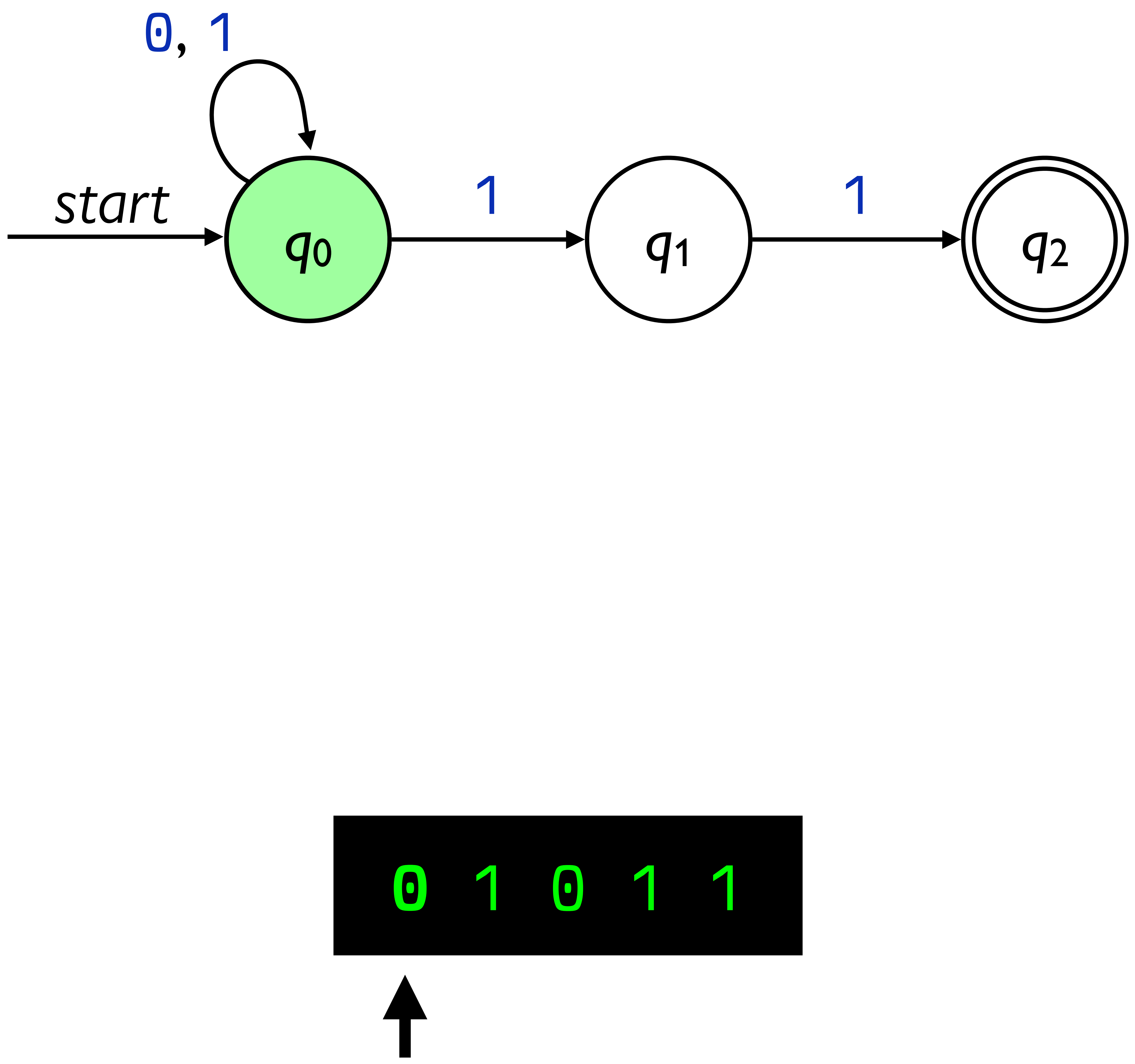
If an NFA needs to make a transition when no transition exists, the automaton dies and that particular path does not accept.

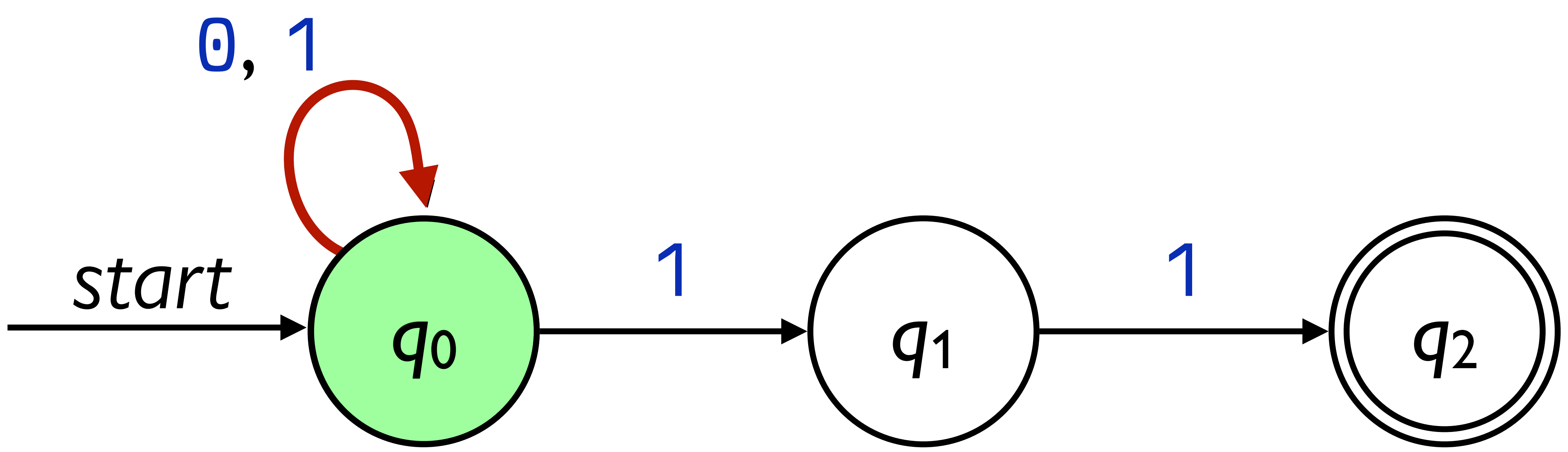


0 1 0 1 1



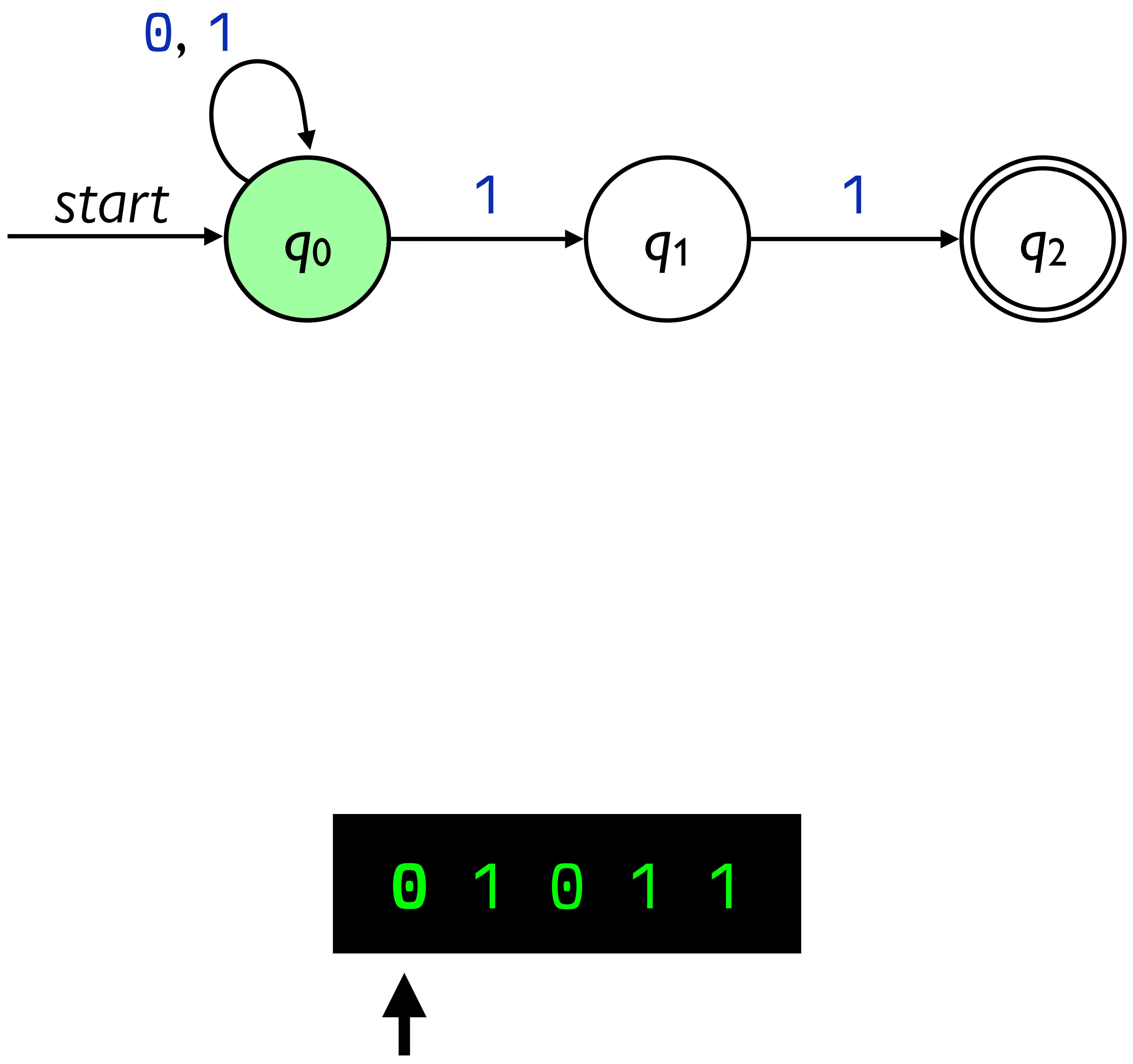
0 1 0 1 1

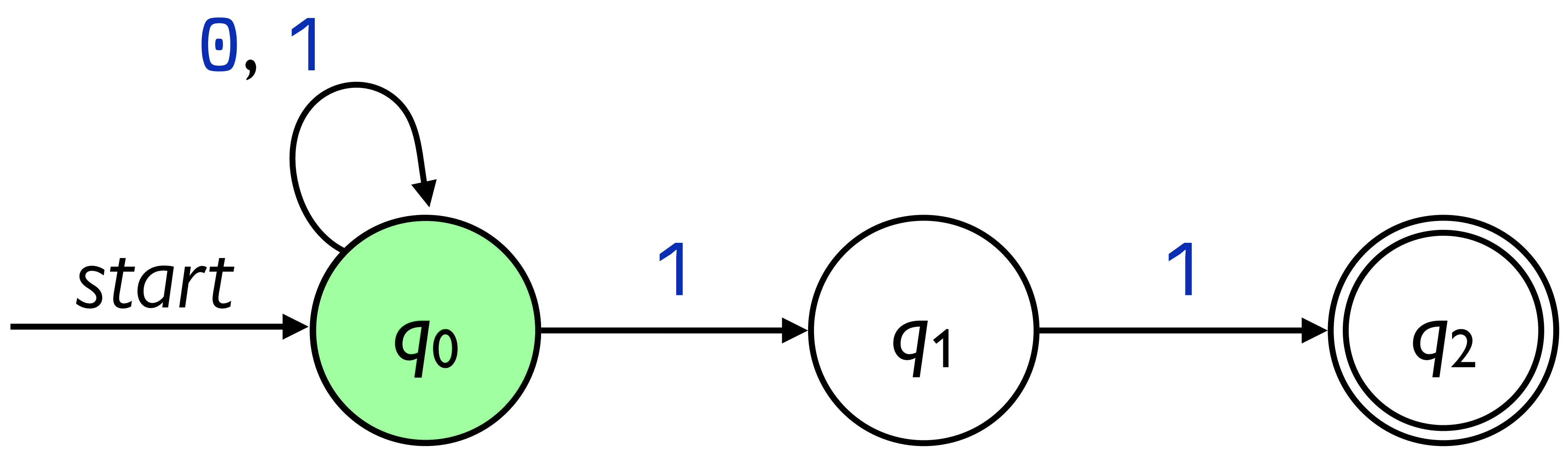

0, 1
start
q0
1
q1
1
q2
0 1 0 1 1

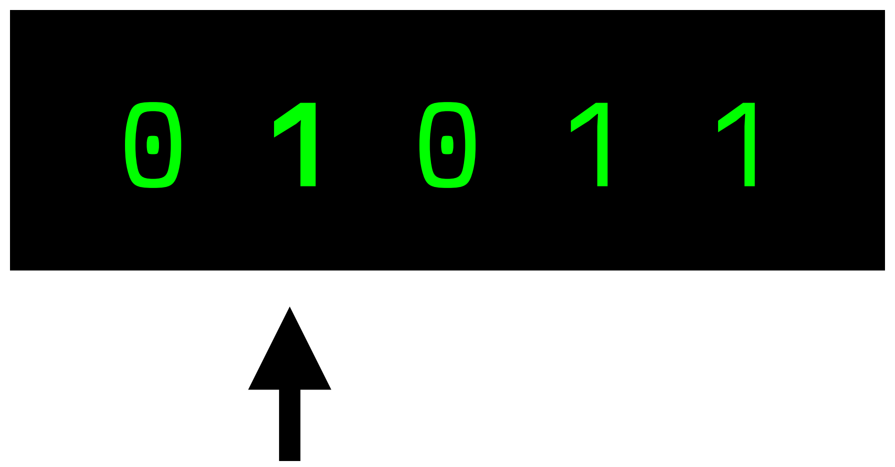


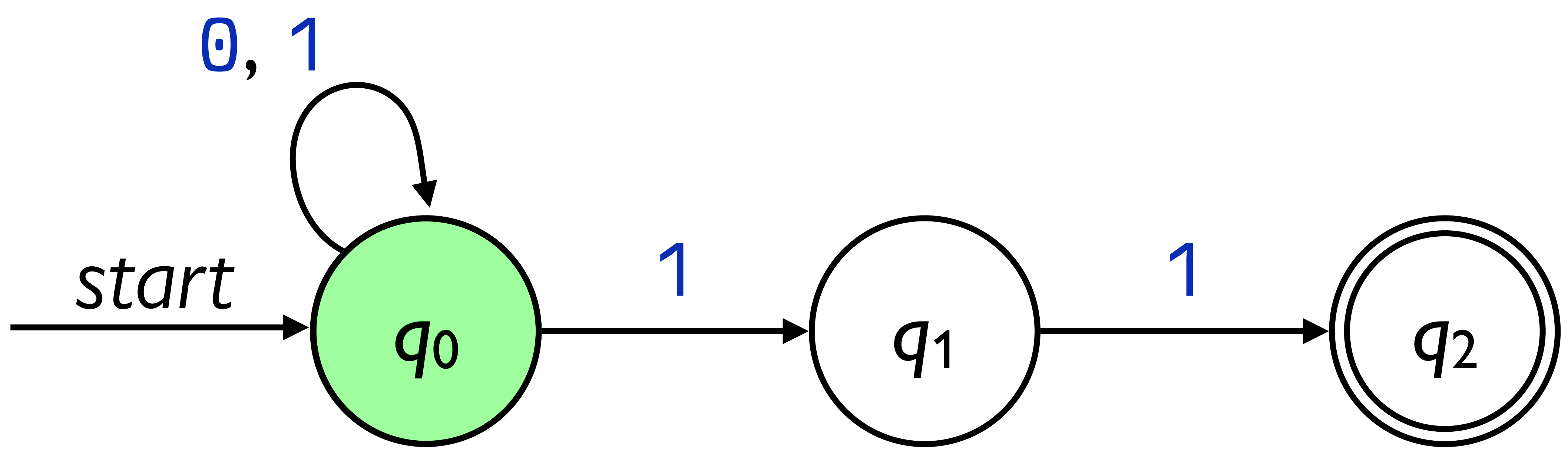
0, 1
start
q_0
1
q_1
1
q_2

0 1 0 1 1

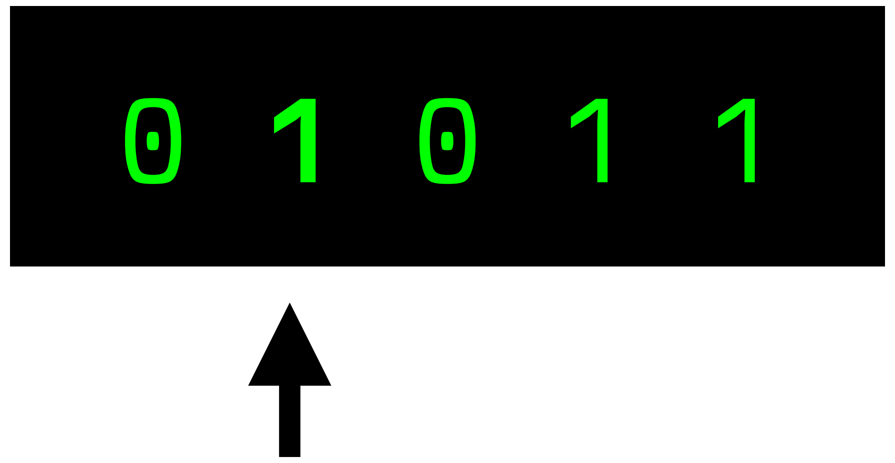

0, 1
start
q0
1
q1
1
q2
0 1 0 1 1


0, 1
start
q_0
1
q_1
1
q_2

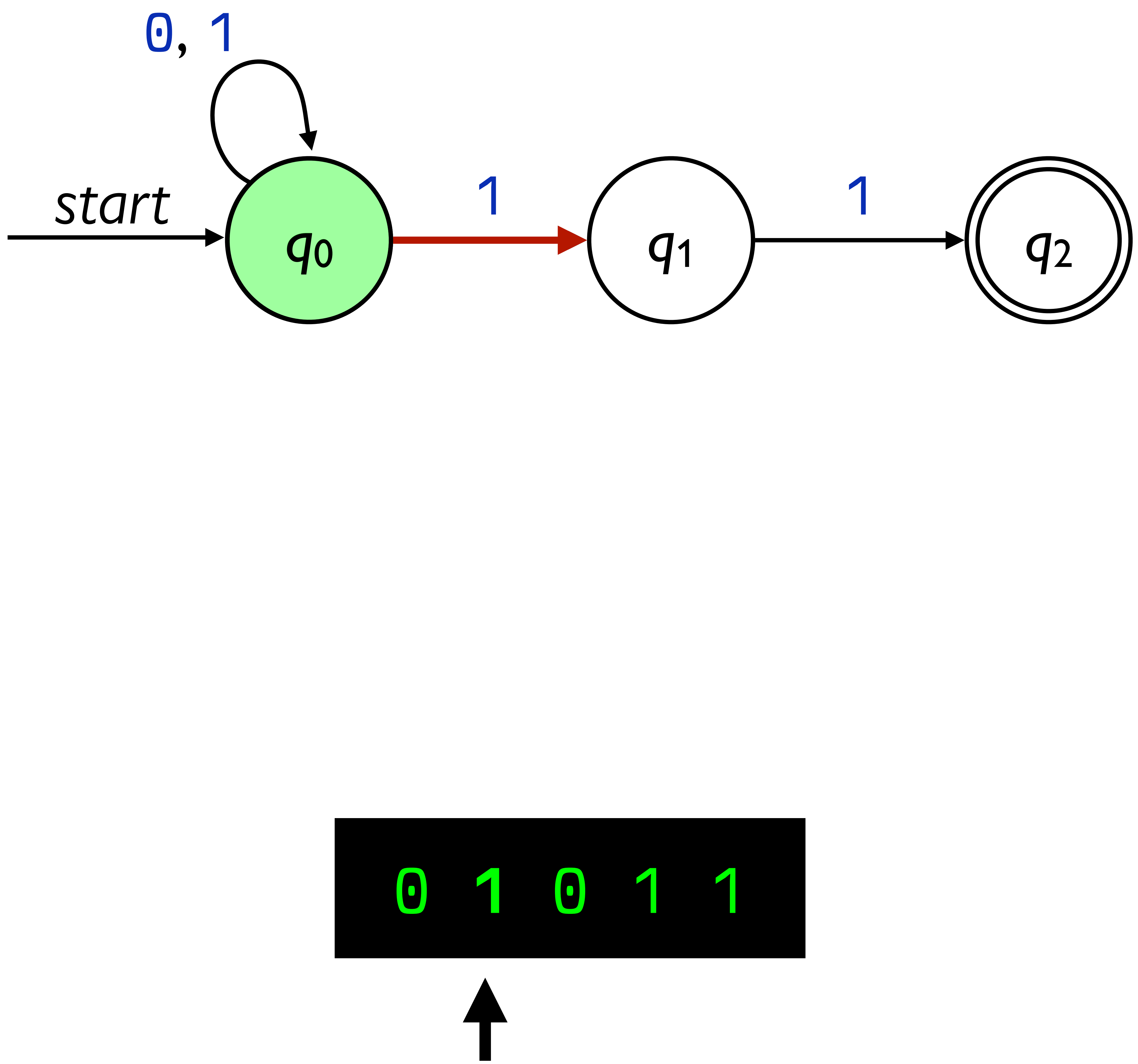

0 1 0 1 1

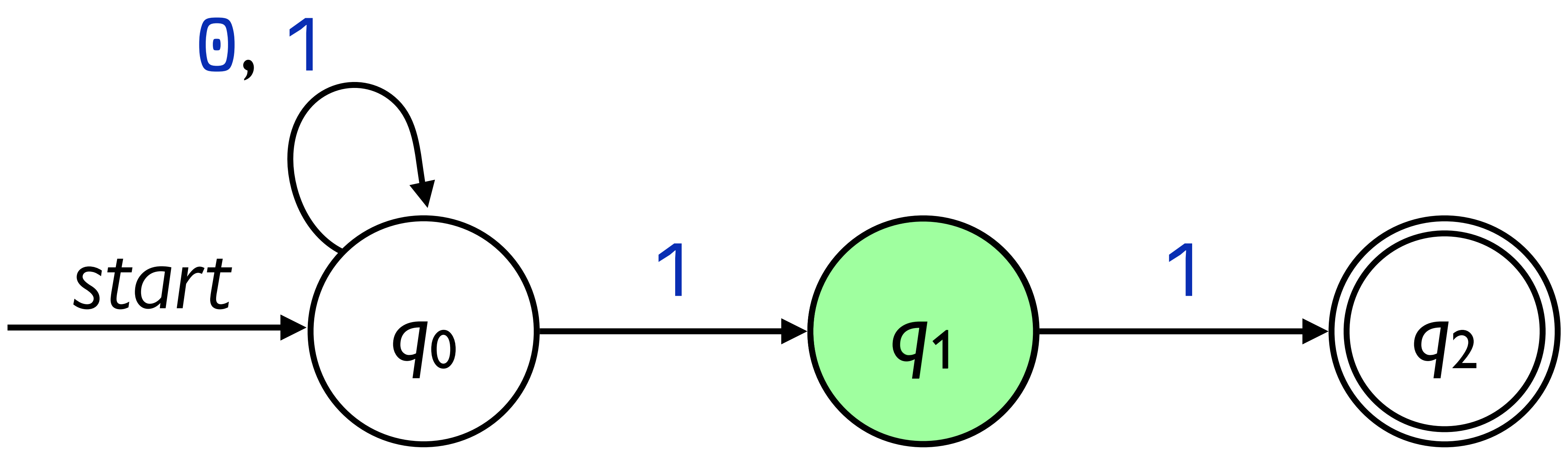


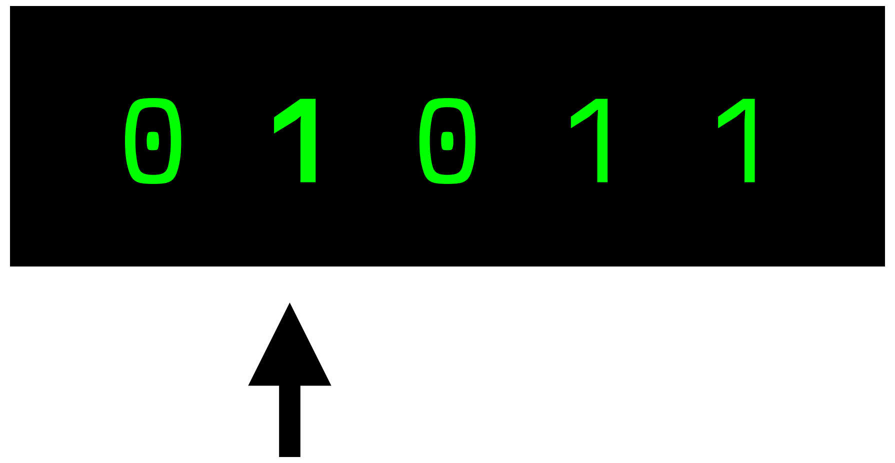
0, 1
start
q_0
1
q_1
1
q_2

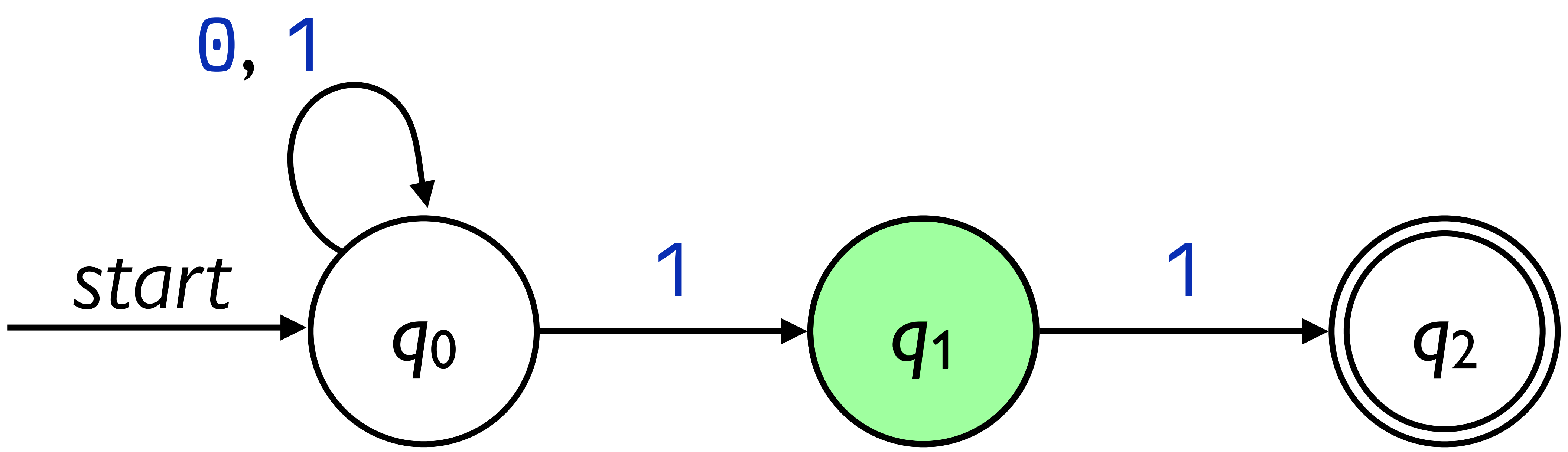


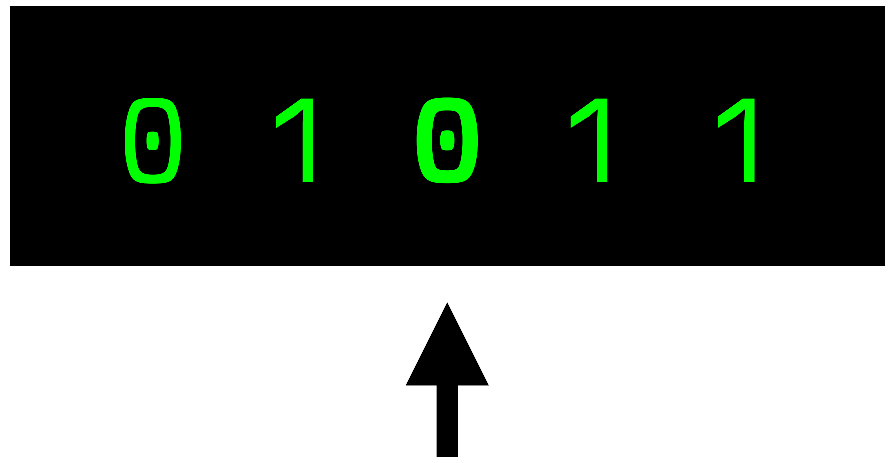
0 1 0 1 1

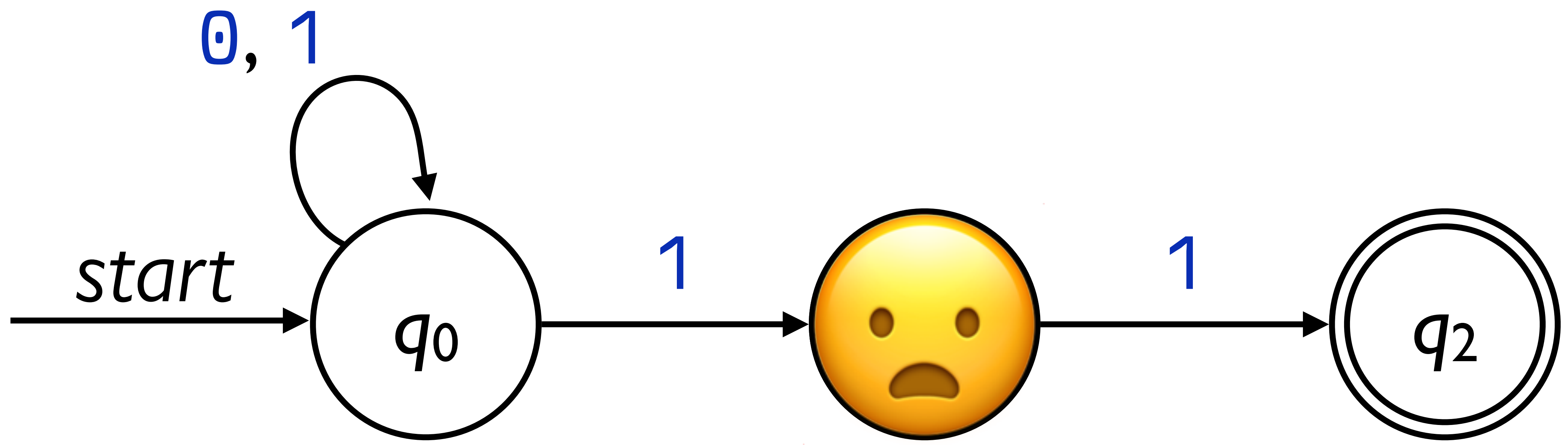

0, 1
start
q_0
1
q_1
1
q_2
0 1 0 1 1


0, 1
start
q0
1
q1
1
q2

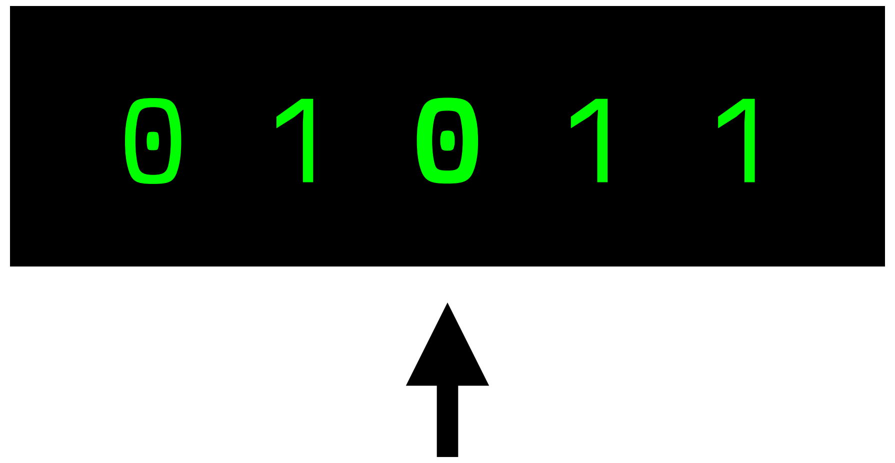

0 1 0 1 1

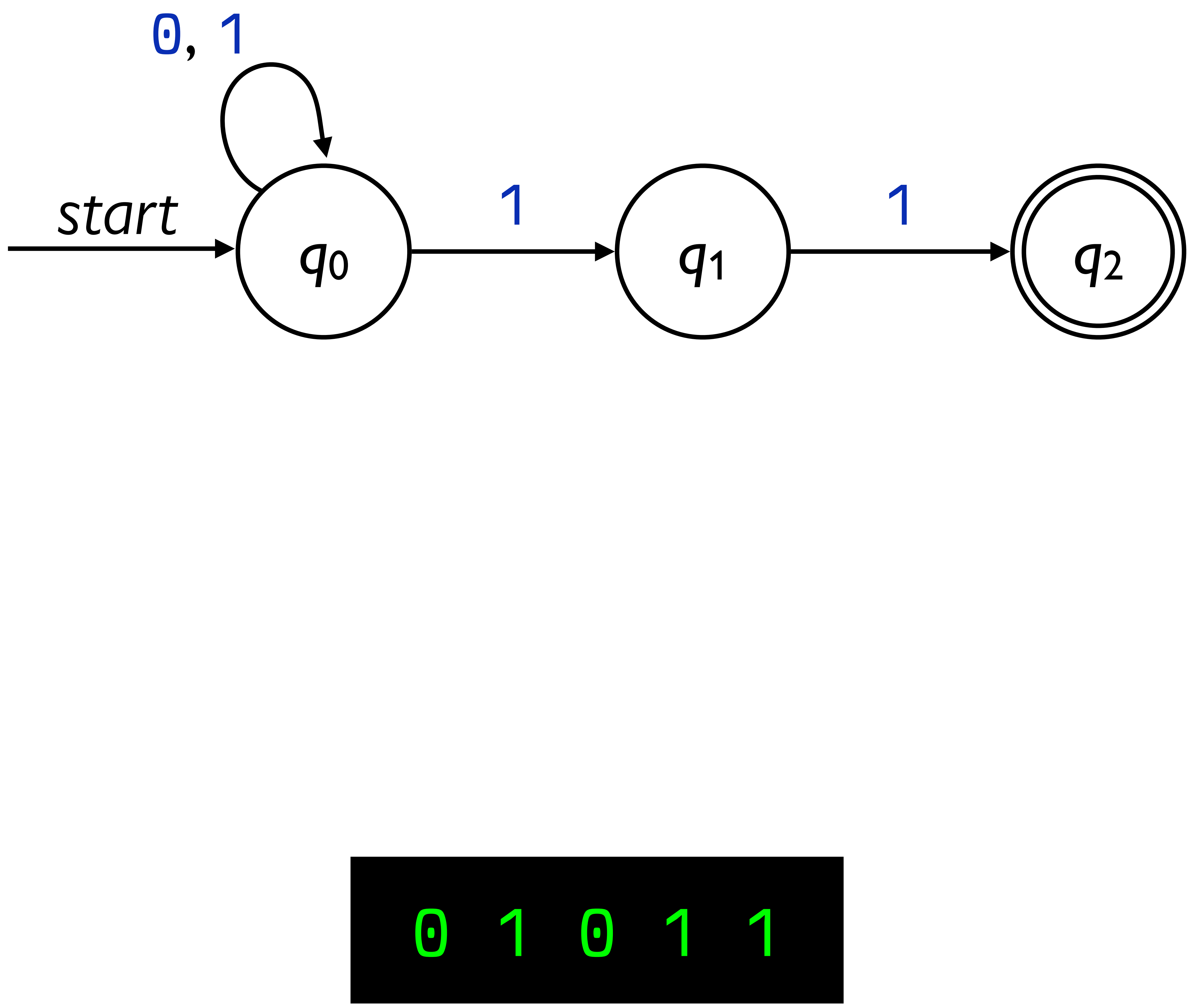

0, 1
start
q_0
1
q_1
1
q_2

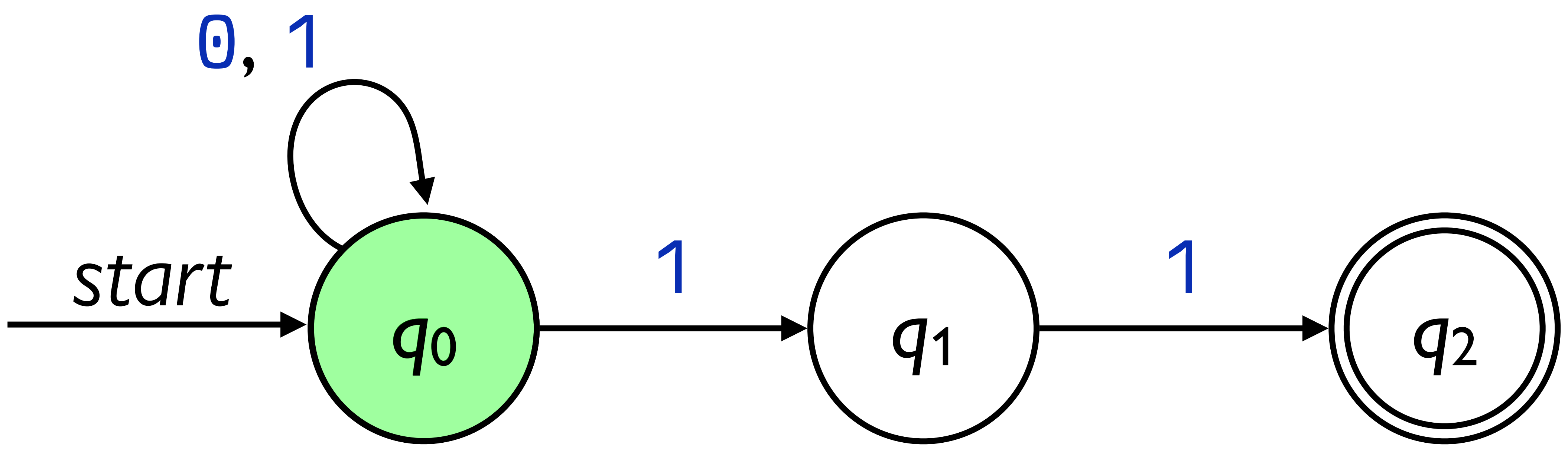

0 1 0 1 1



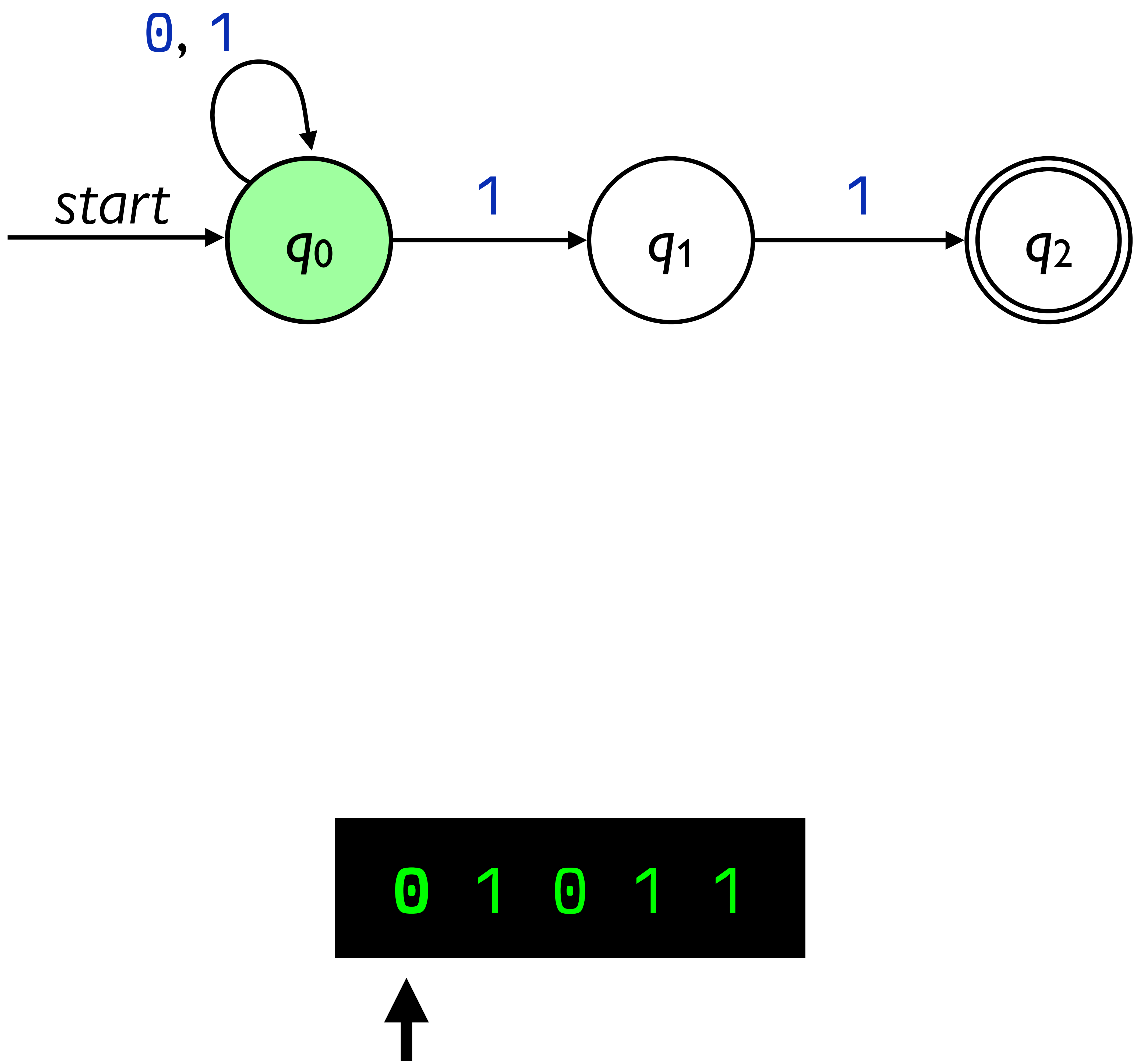
Nowhere to go!

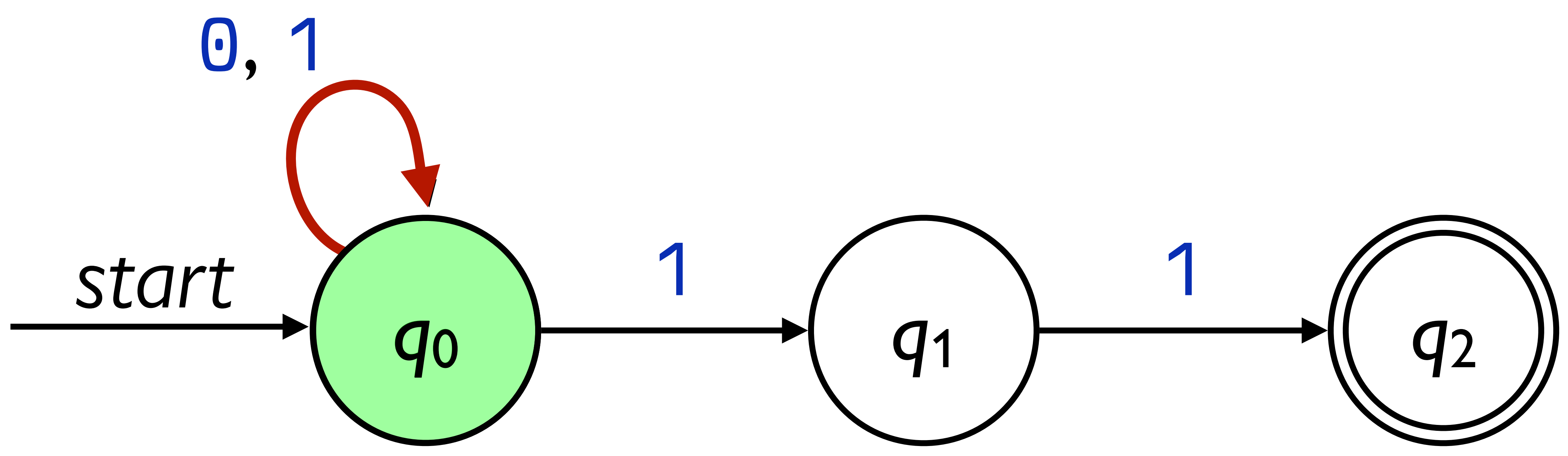



0, 1
start
q_0
1
q_1
1
q_2
0 1 0 1 1

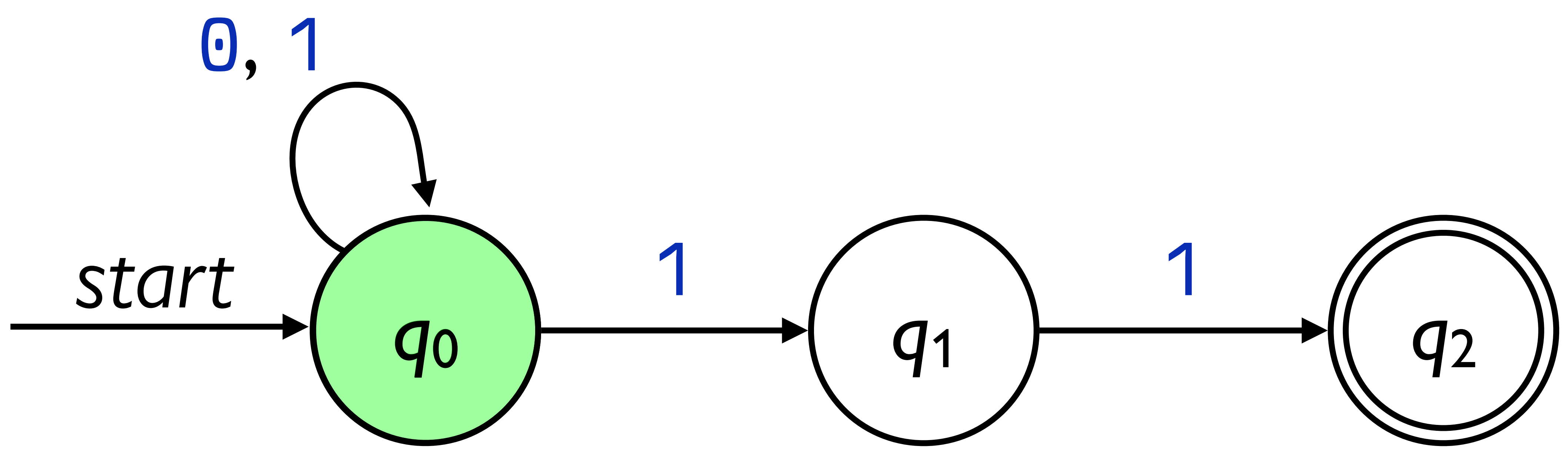


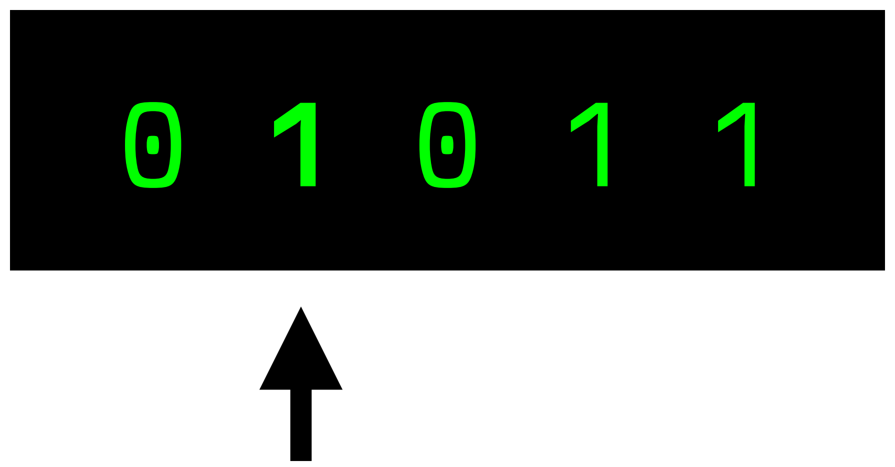
0 1 0 1 1

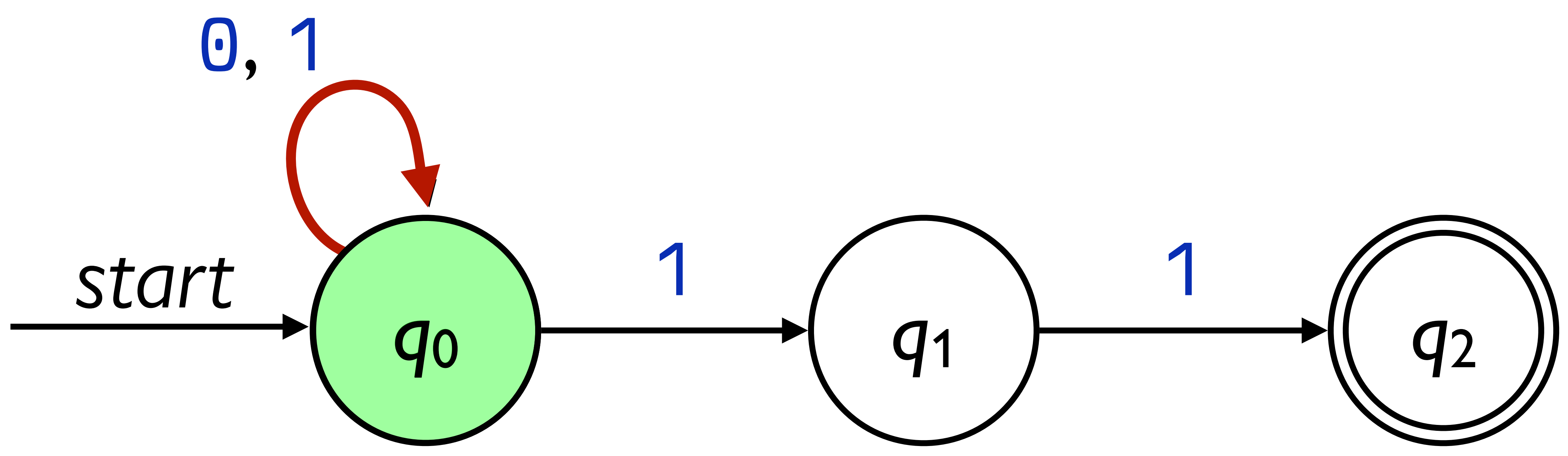

0, 1
start
q0
1
q1
1
q2
0 1 0 1 1

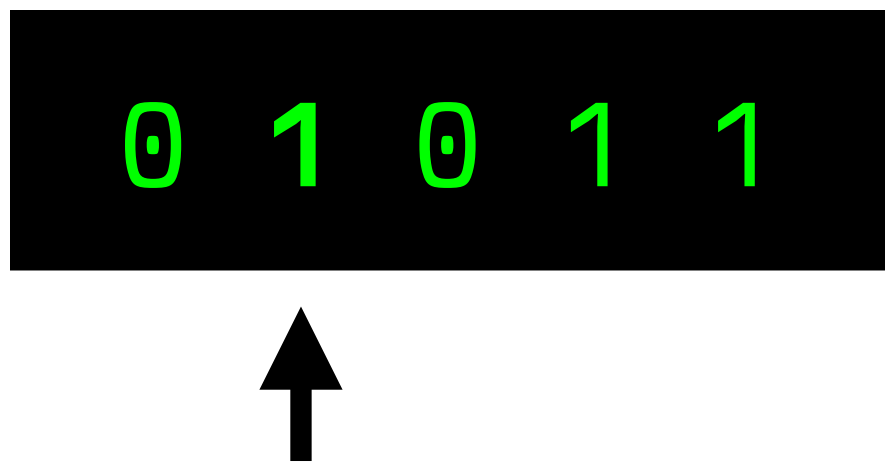

0, 1
start
q0
1
q1
1
q2

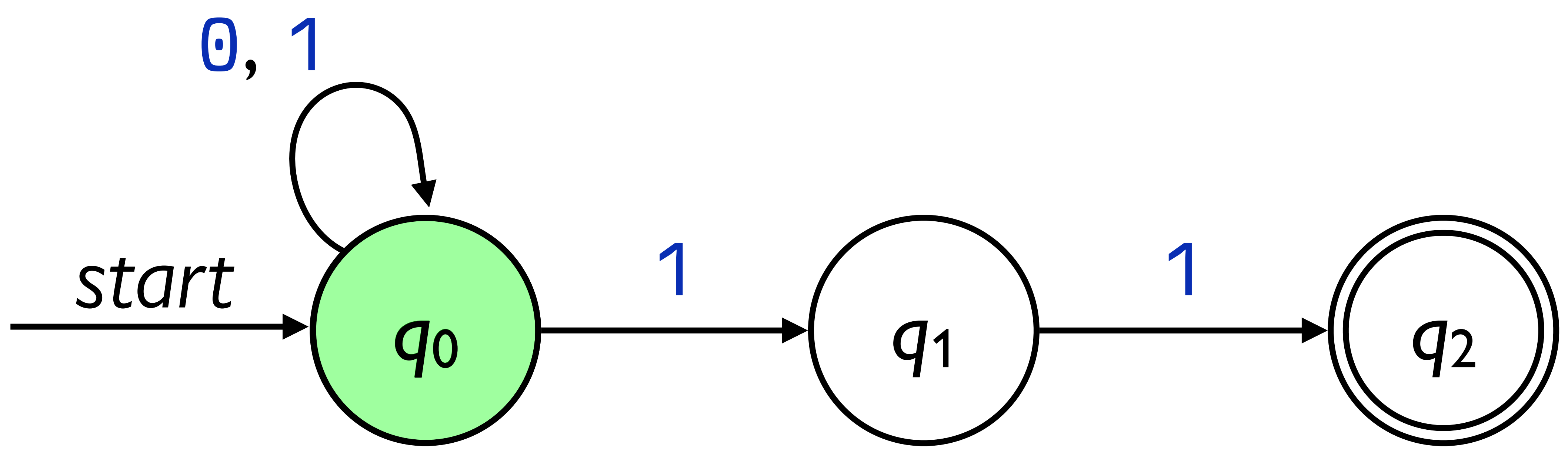

0 1 0 1 1

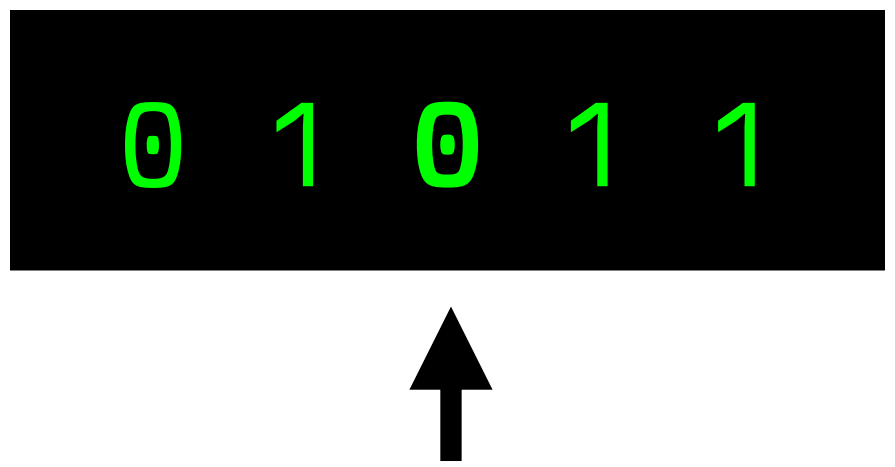

0, 1
start
q_0
1
q_1
1
q_2

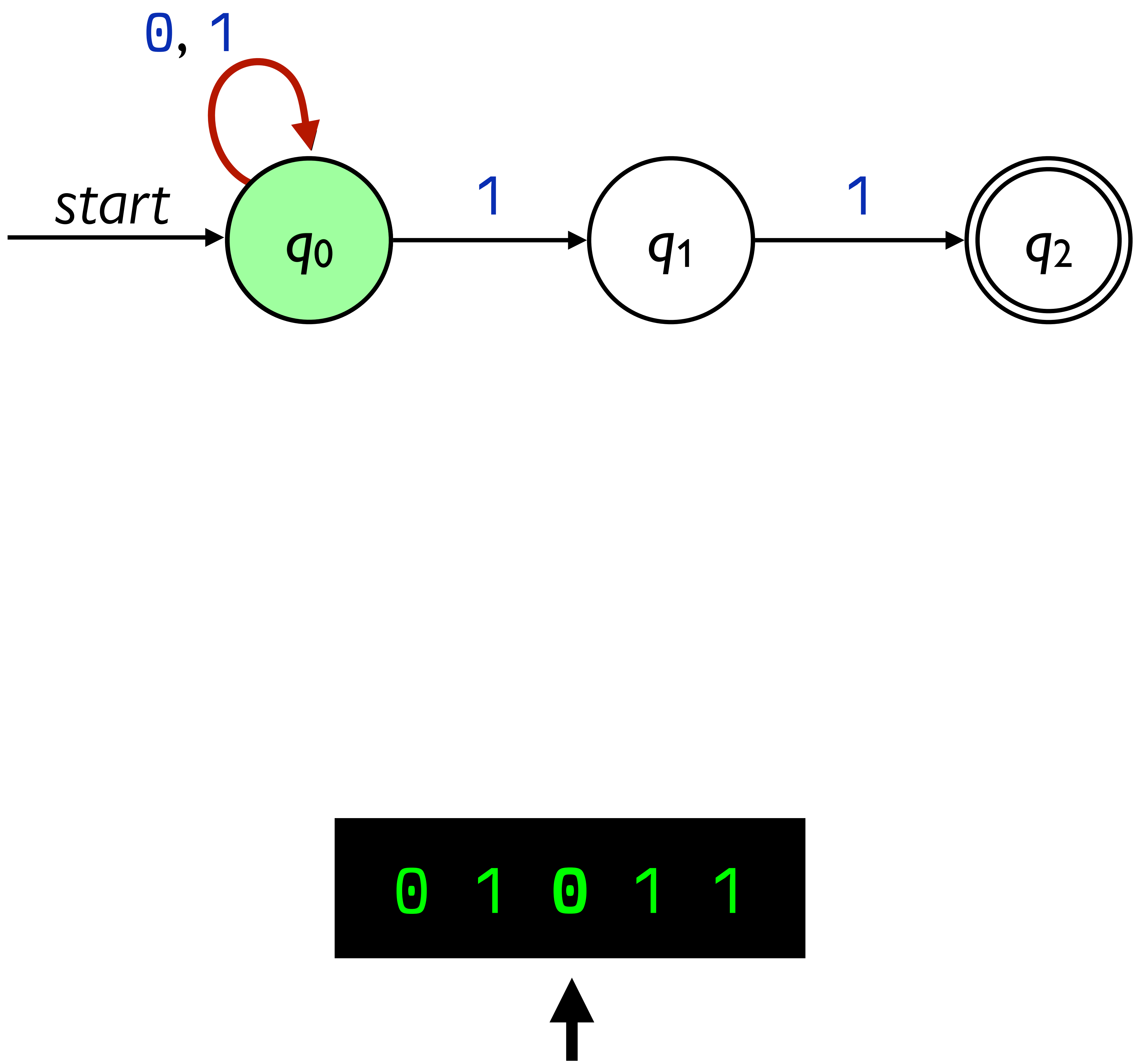

0 1 0 1 1


0, 1
start
q_0
1
q_1
1
q_2

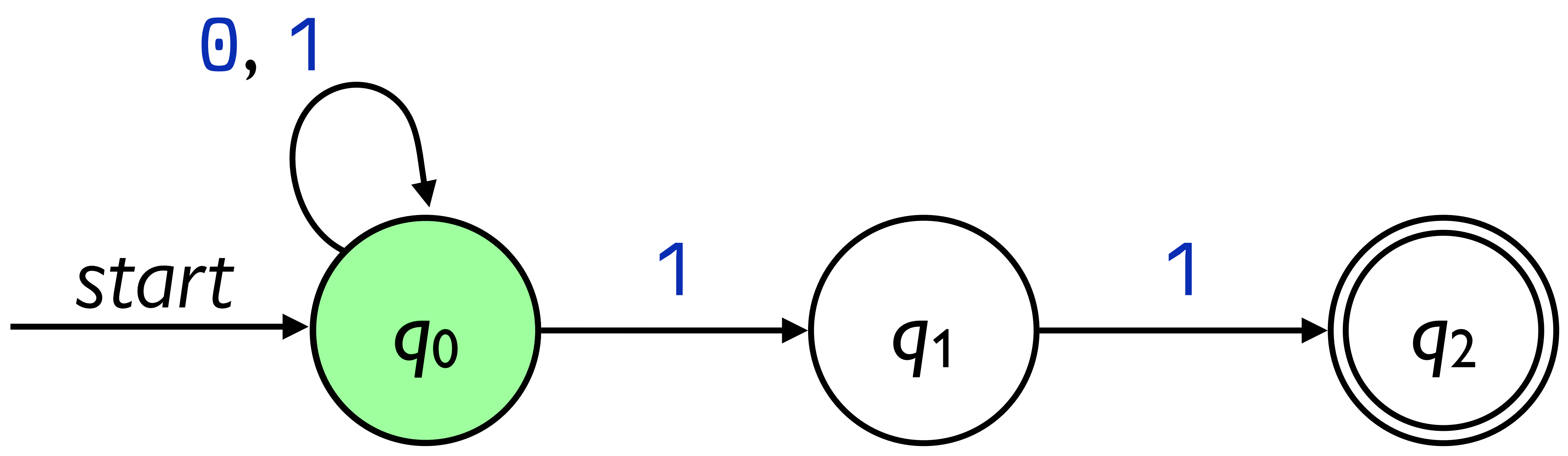

0 1 0 1 1

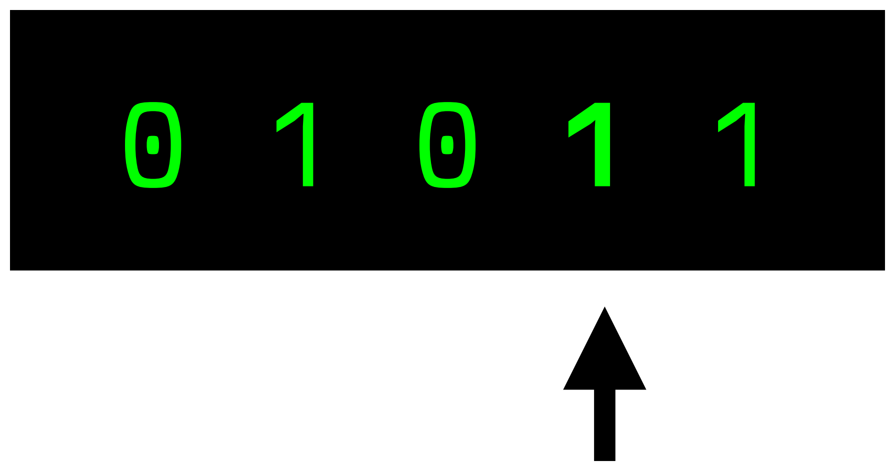

0, 1
start
q_0
1
q_1
1
q_2

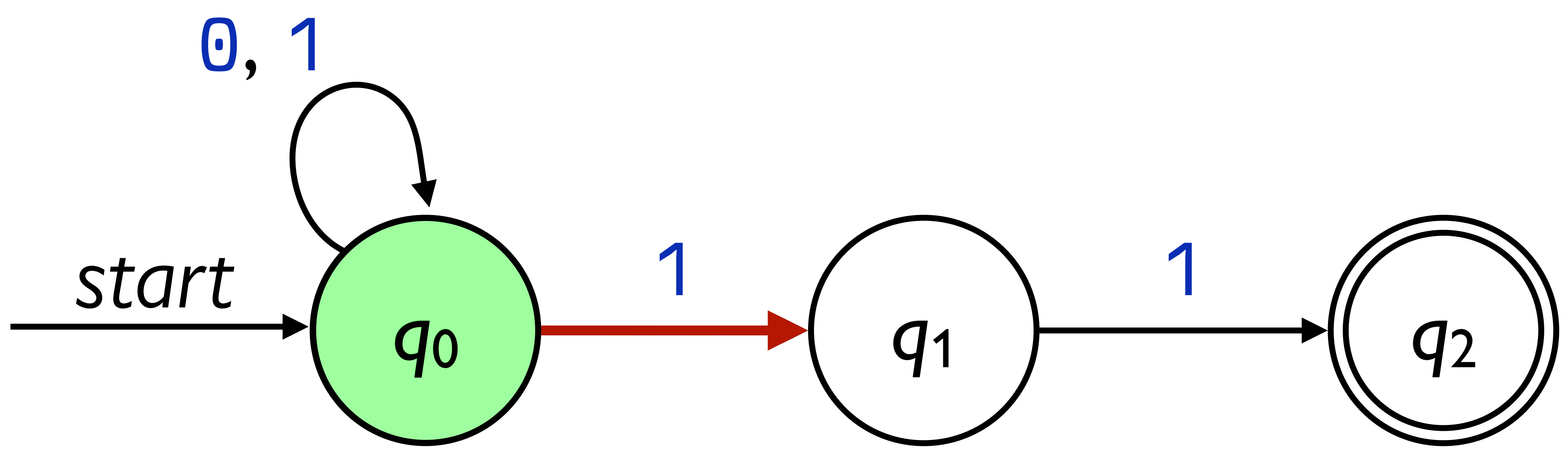
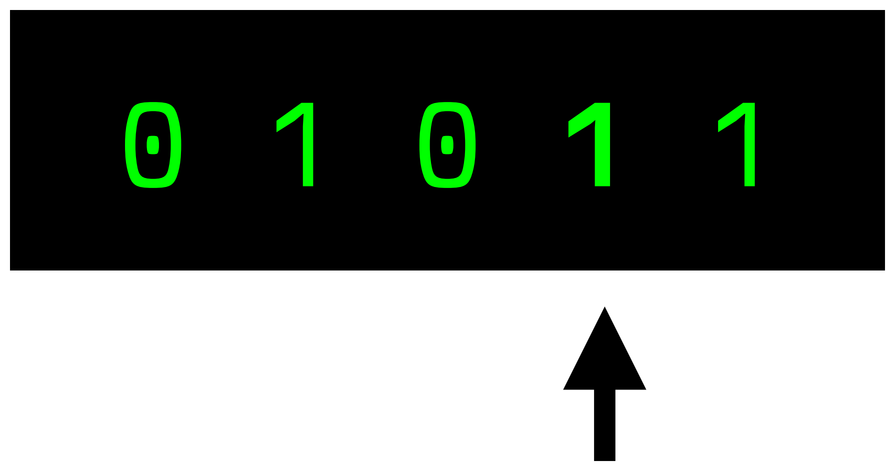

0 1 0 1 1

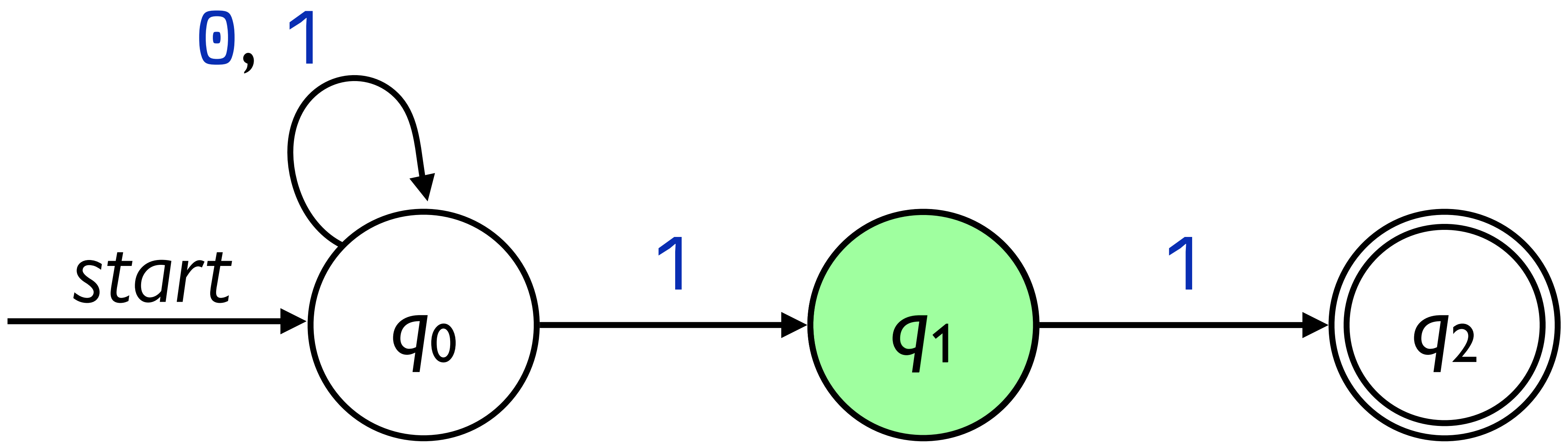


0, 1
start
q0
1
q1
1
q2
0 1 0 1 1

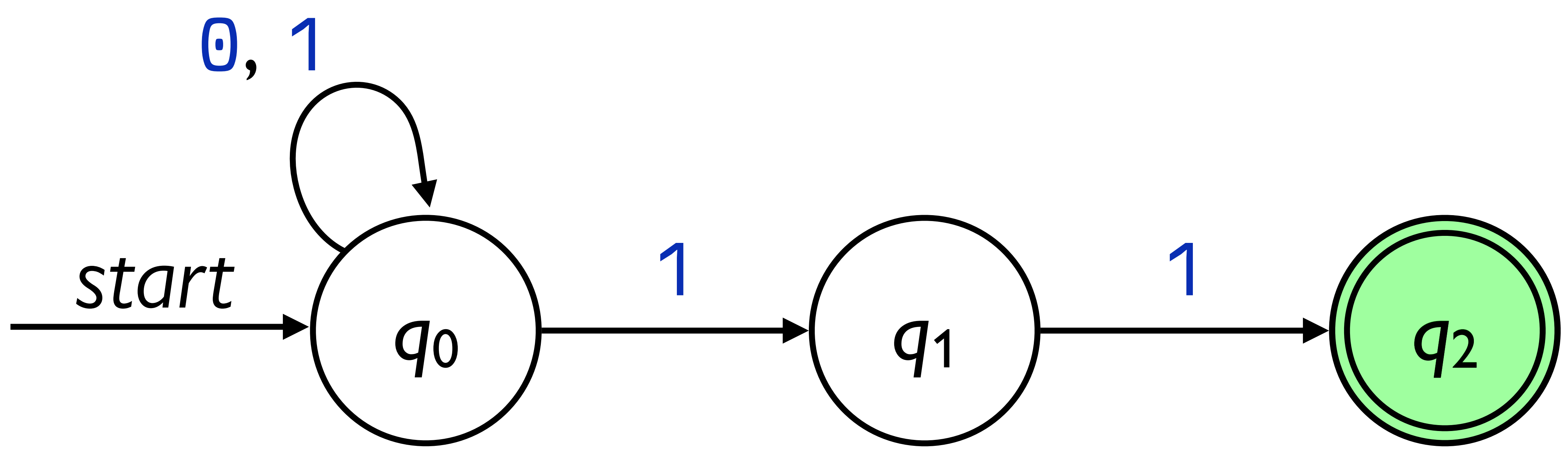

0, 1
start
q_0
1
q_1
1
q_2


0 1 0 1 1

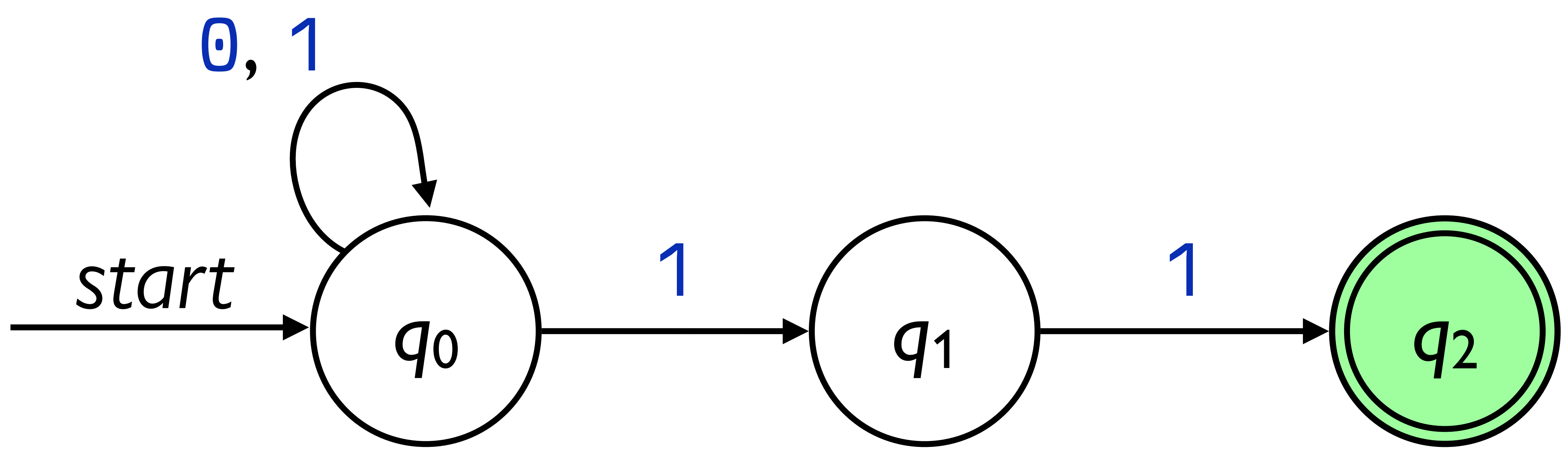

0, 1
start
q0
1
q1
1
q2

0 1 0 1 1


0, 1
start
q_0
1
q_1
1
q_2


0 1 0 1 1

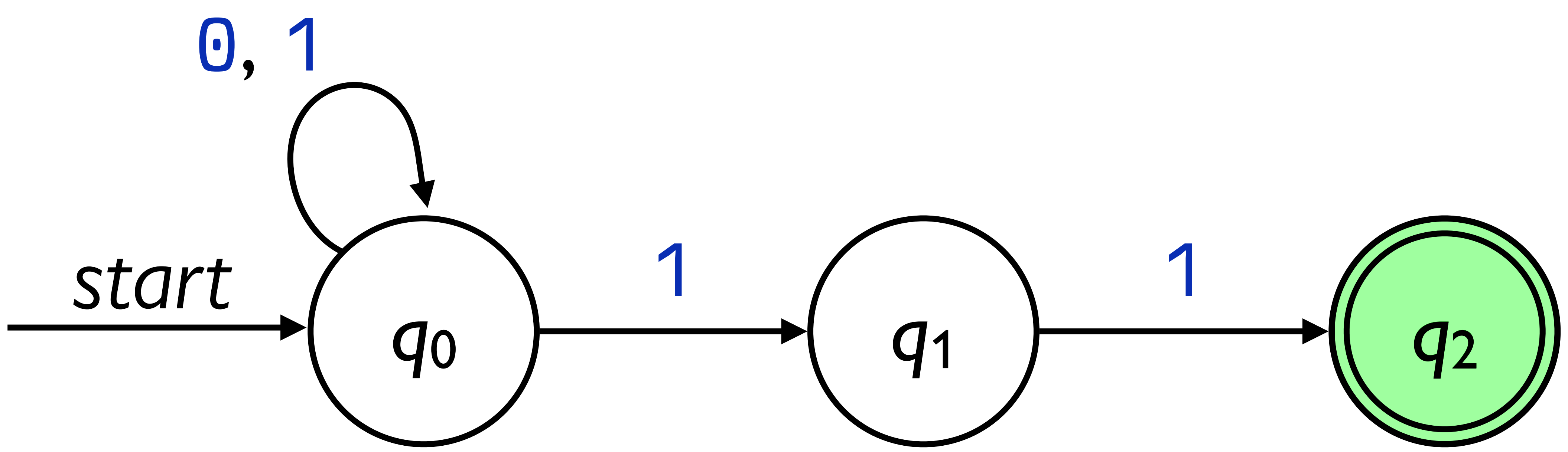

0, 1
start
q0
1
q1
1
q2


0 1 0 1 1

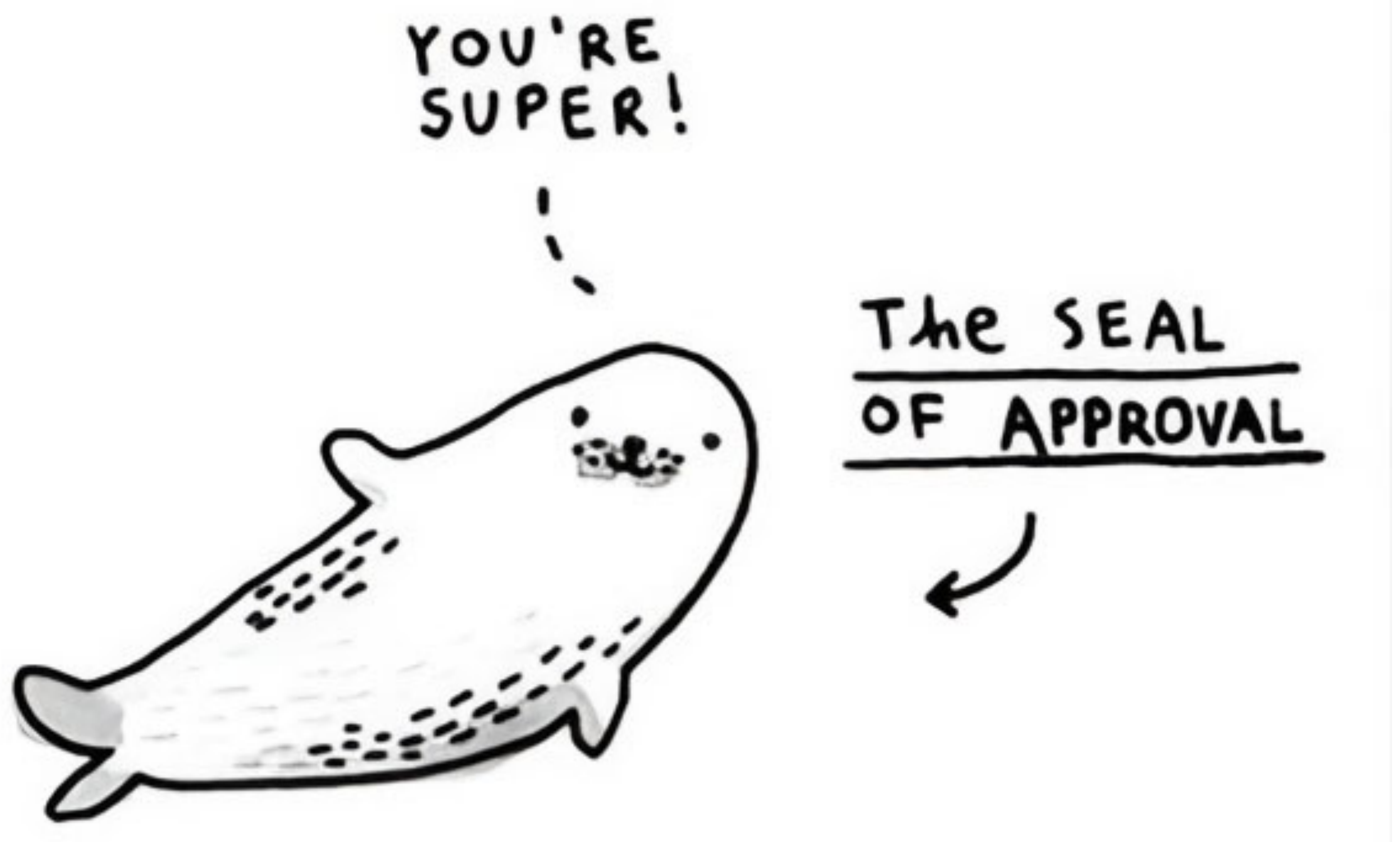


0 1 0 1 1

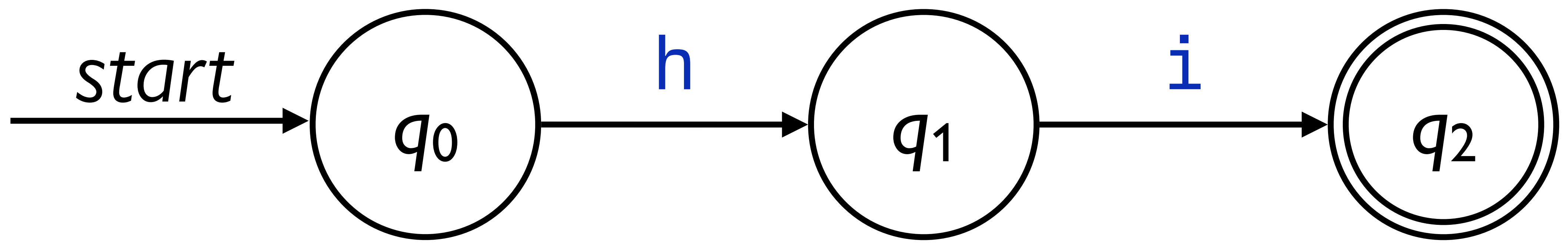
Illustration by Gemma Correll



0 1 0 1 1

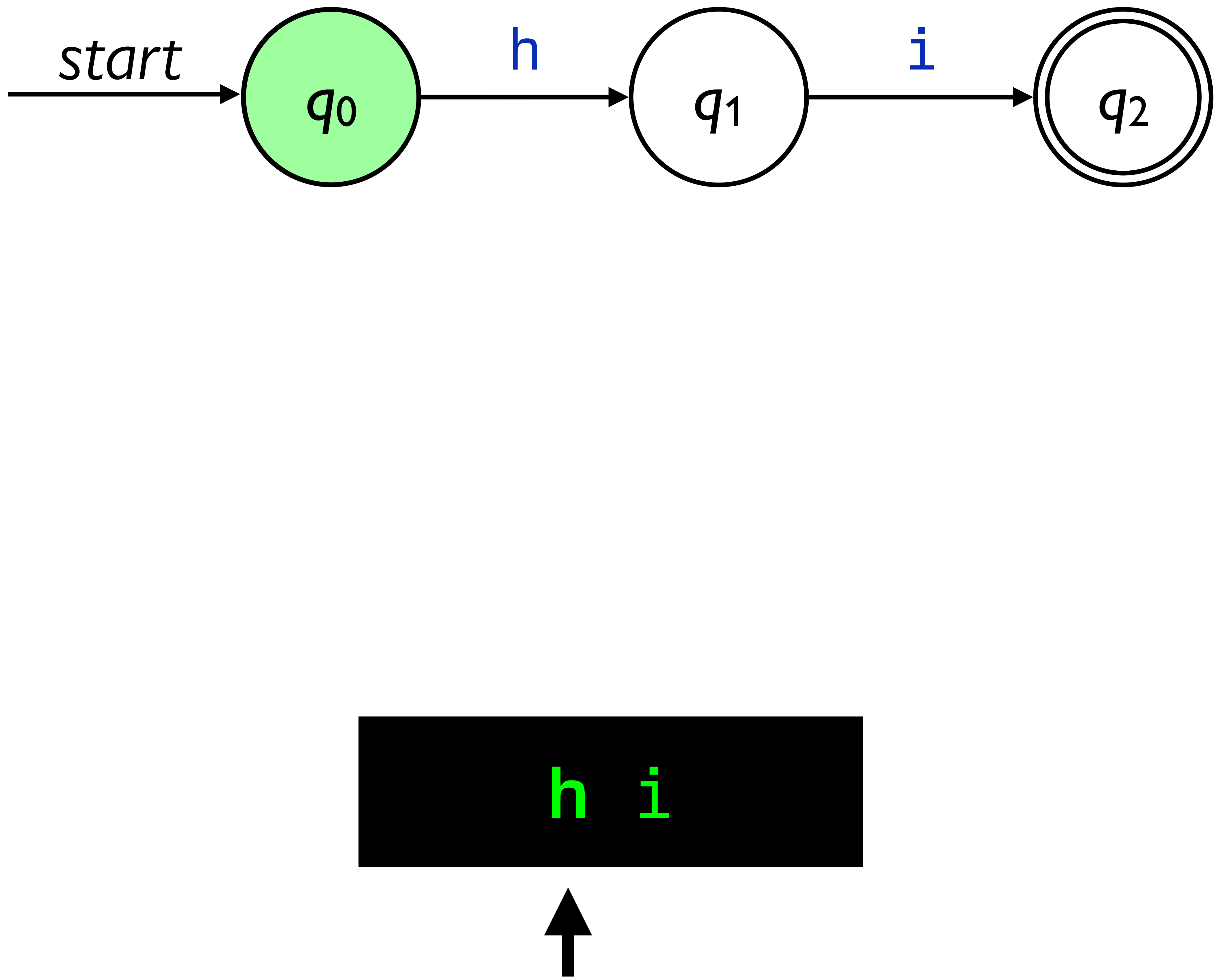


Hello, NFA!

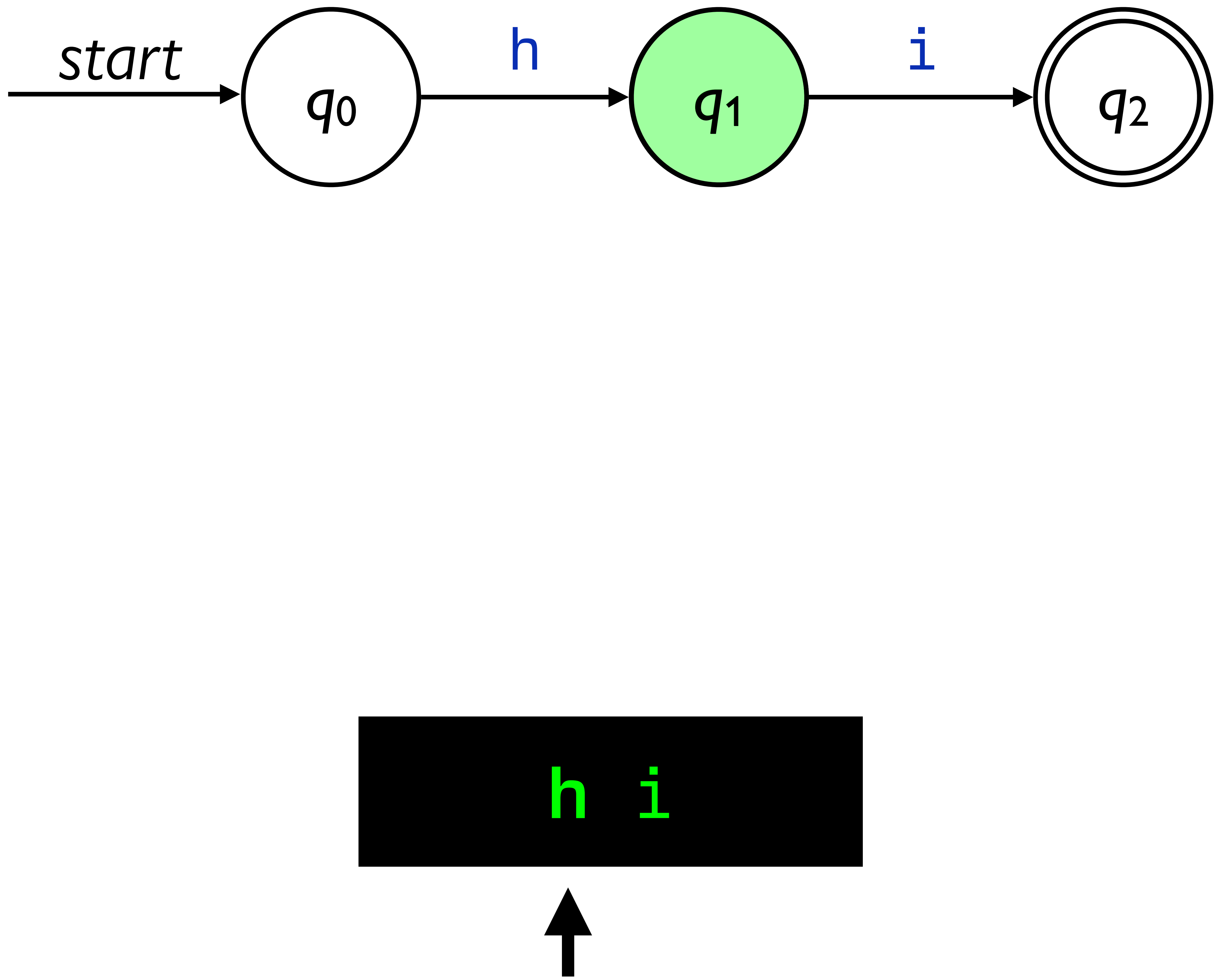



```
h i
```

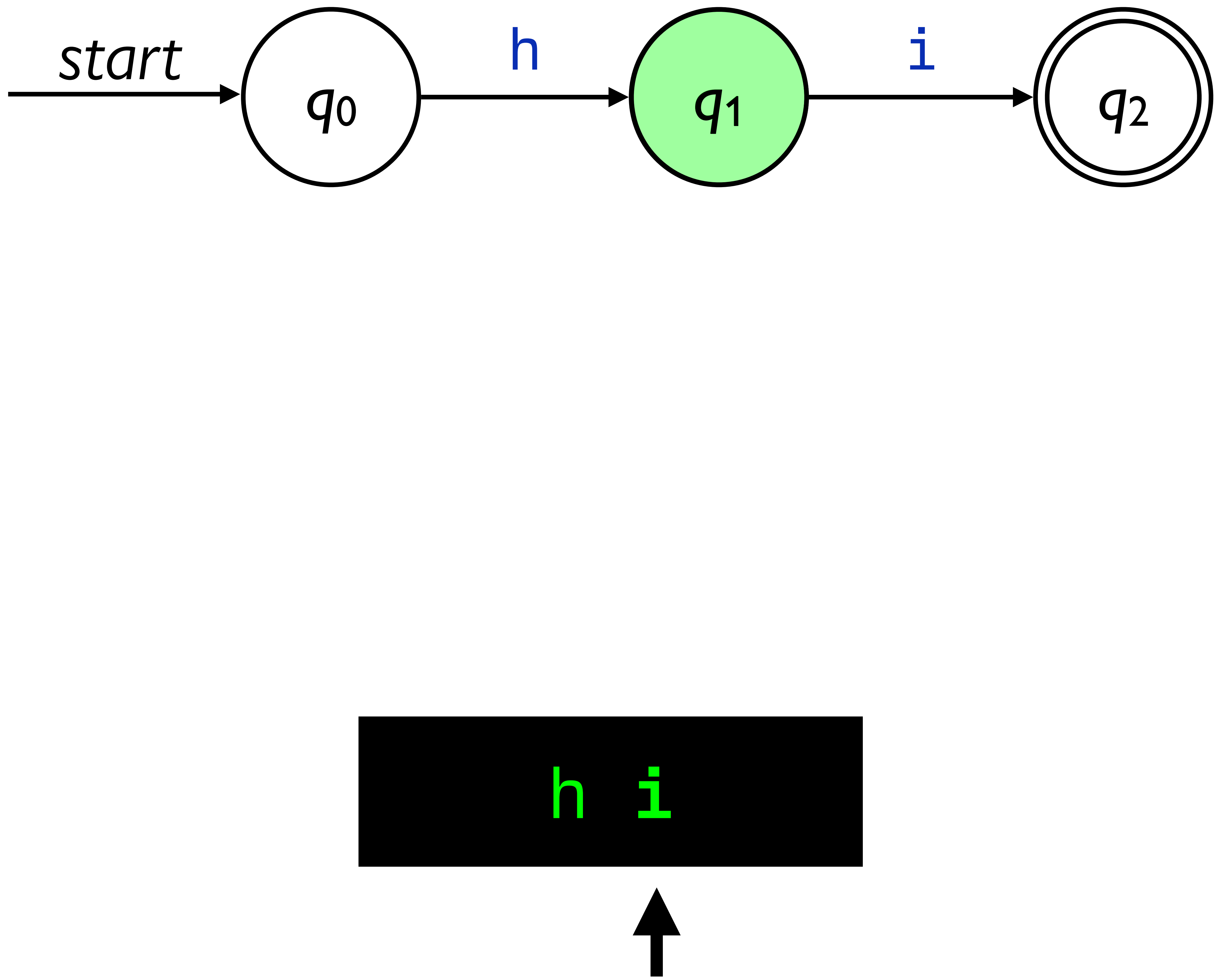
Hello, NFA!



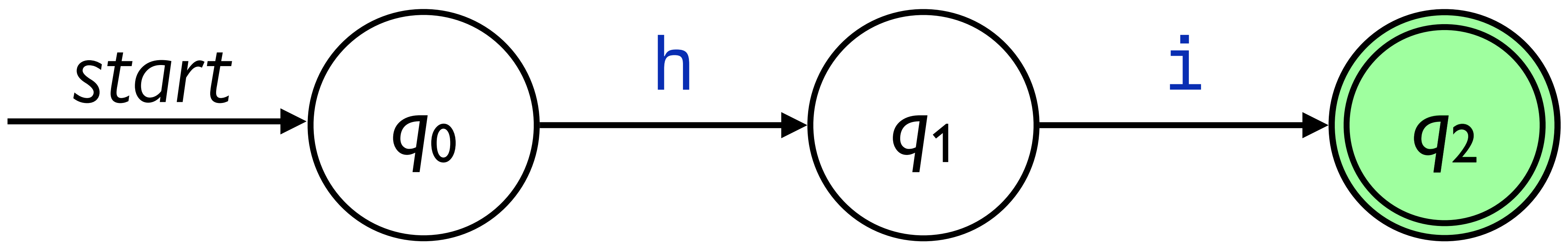
Hello, NFA!

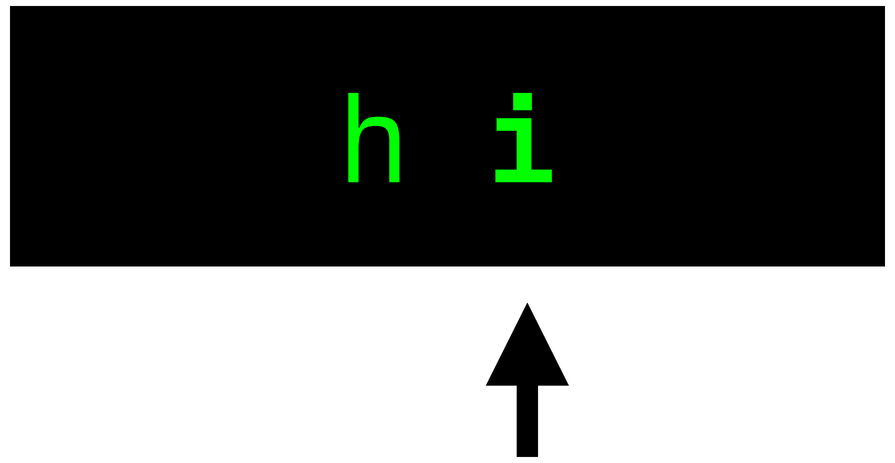


Hello, NFA!

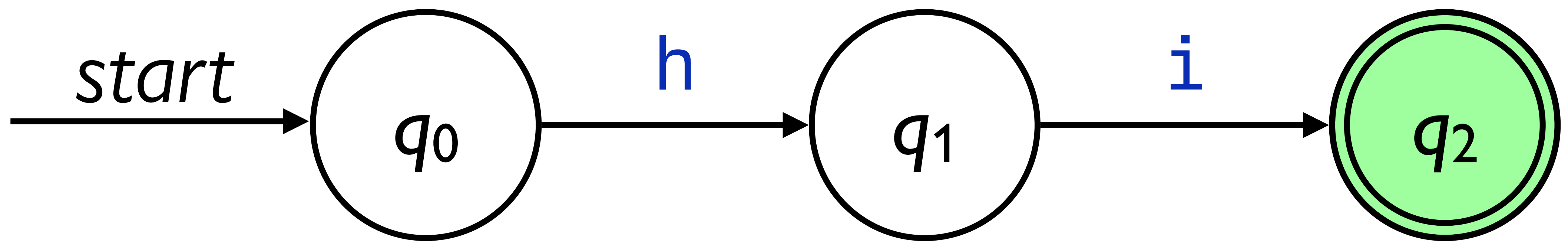


Hello, NFA!





Hello, NFA!



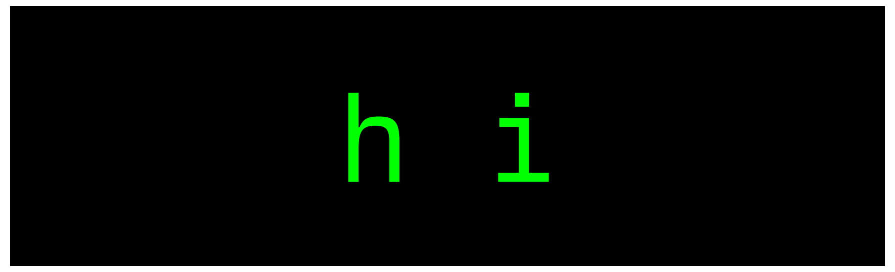
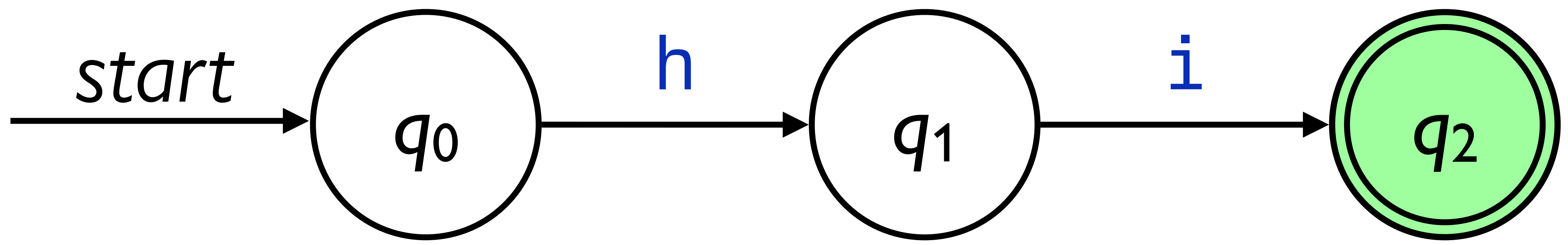
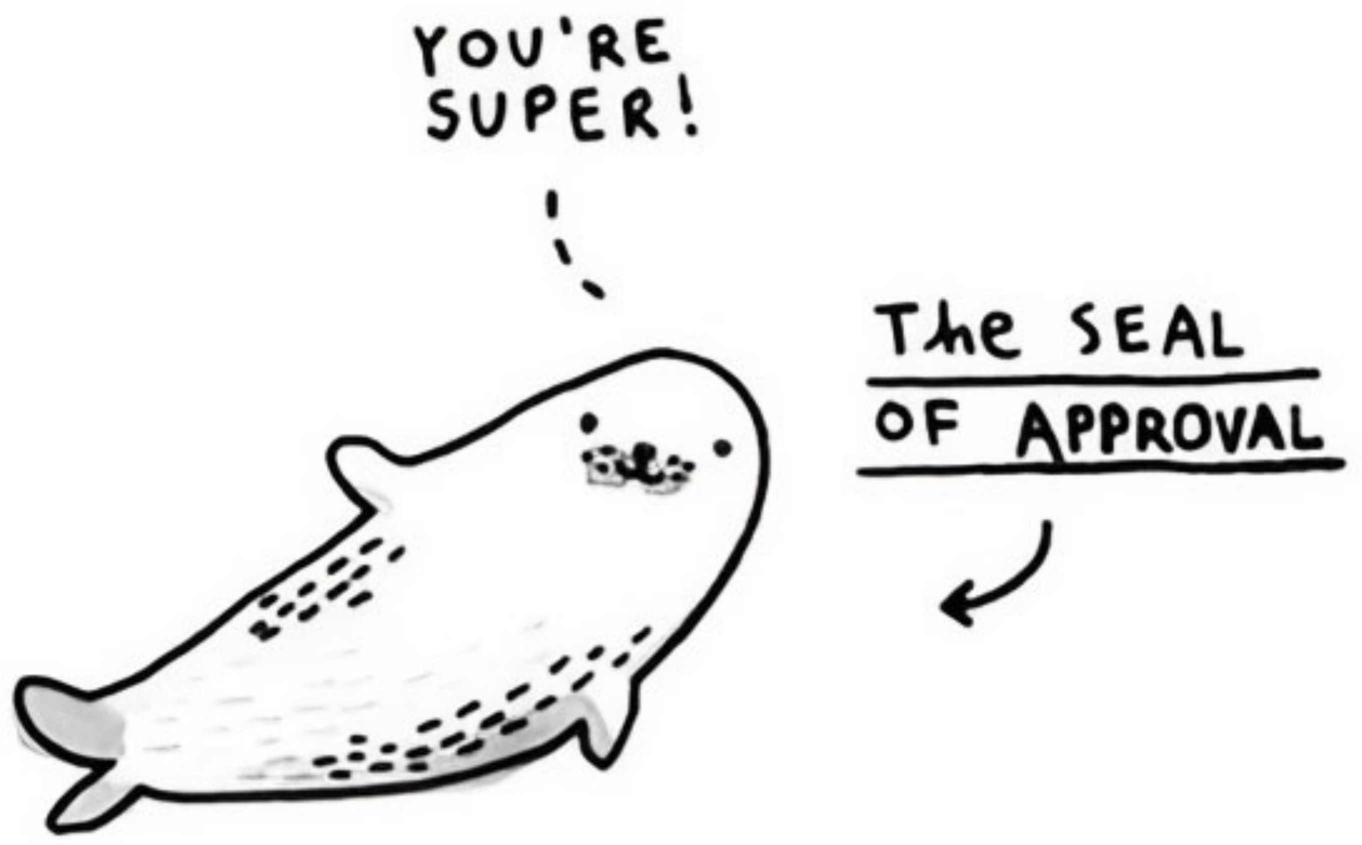


Illustration by Gemma Correll

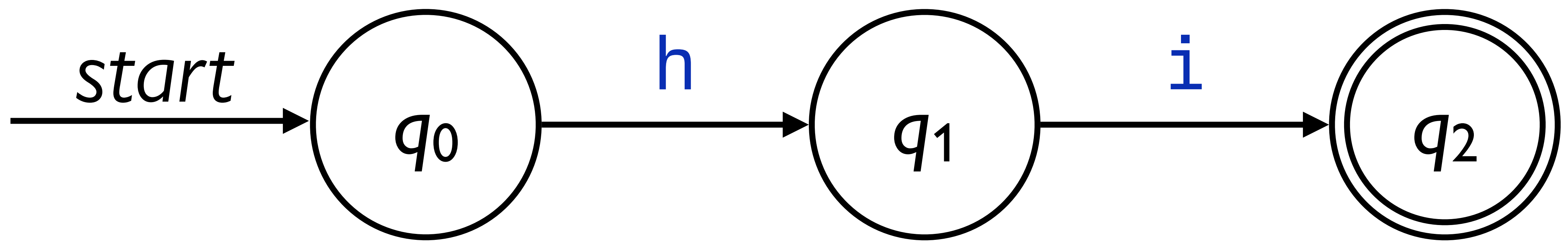
Hello, NFA!



`h i`

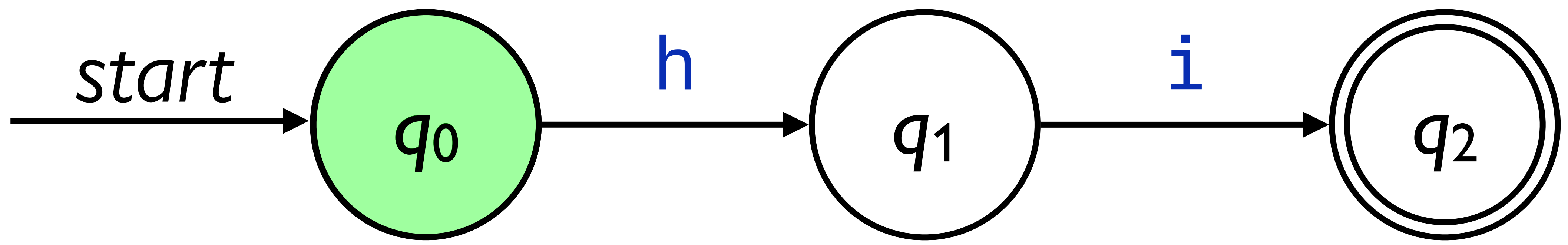


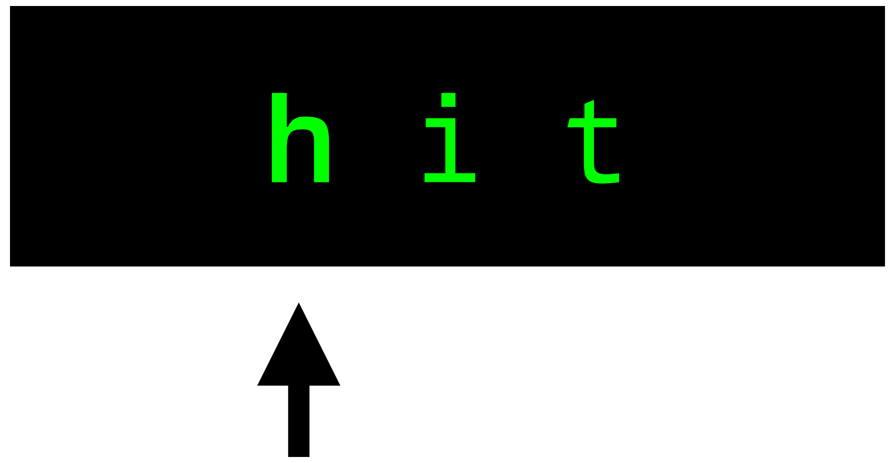
Tragedy in paradise



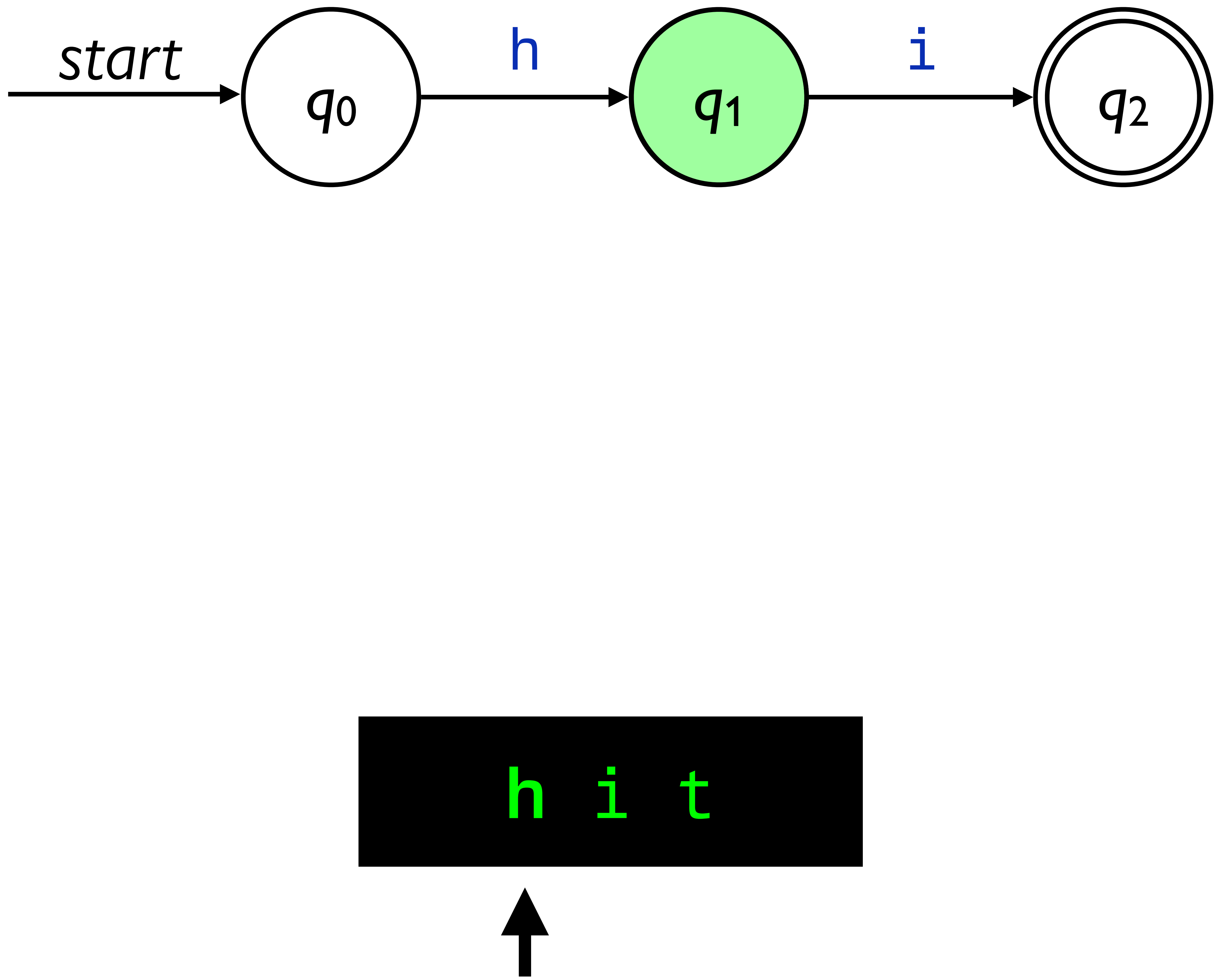
h i t

Tragedy in paradise

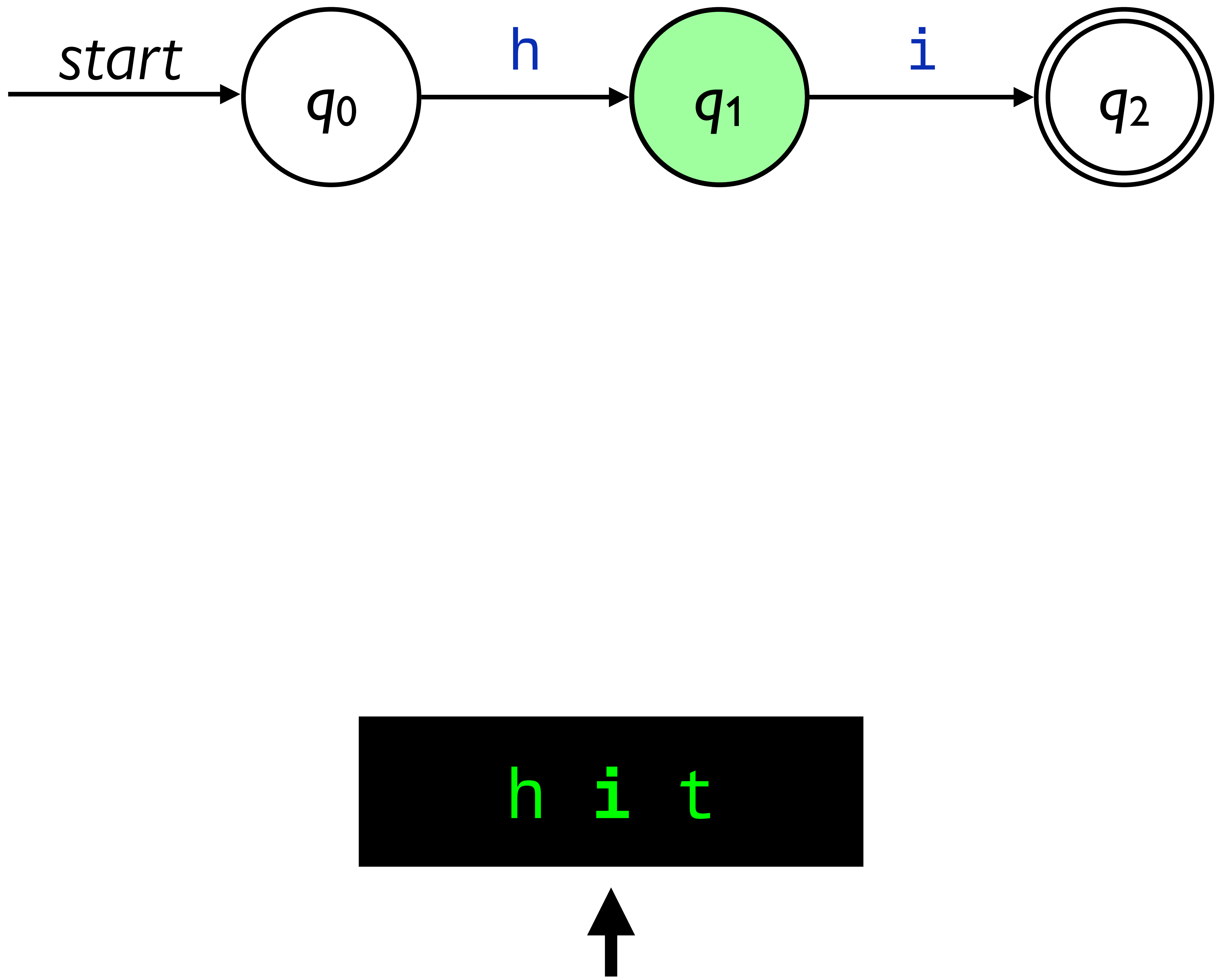




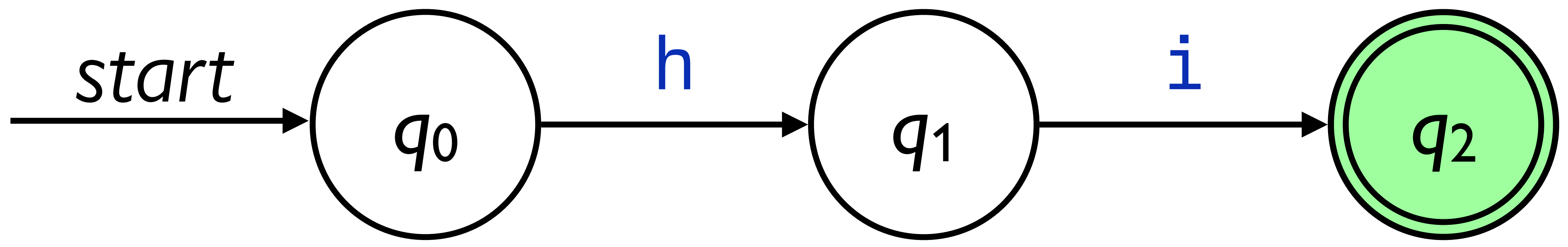
Tragedy in paradise

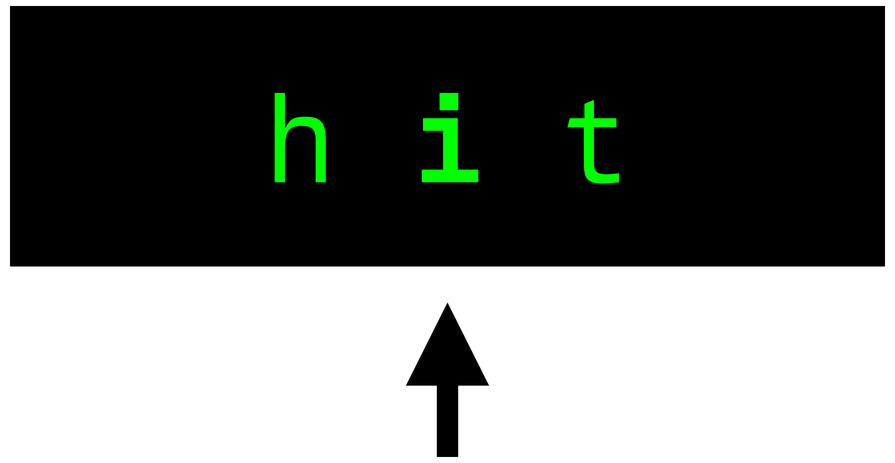


Tragedy in paradise

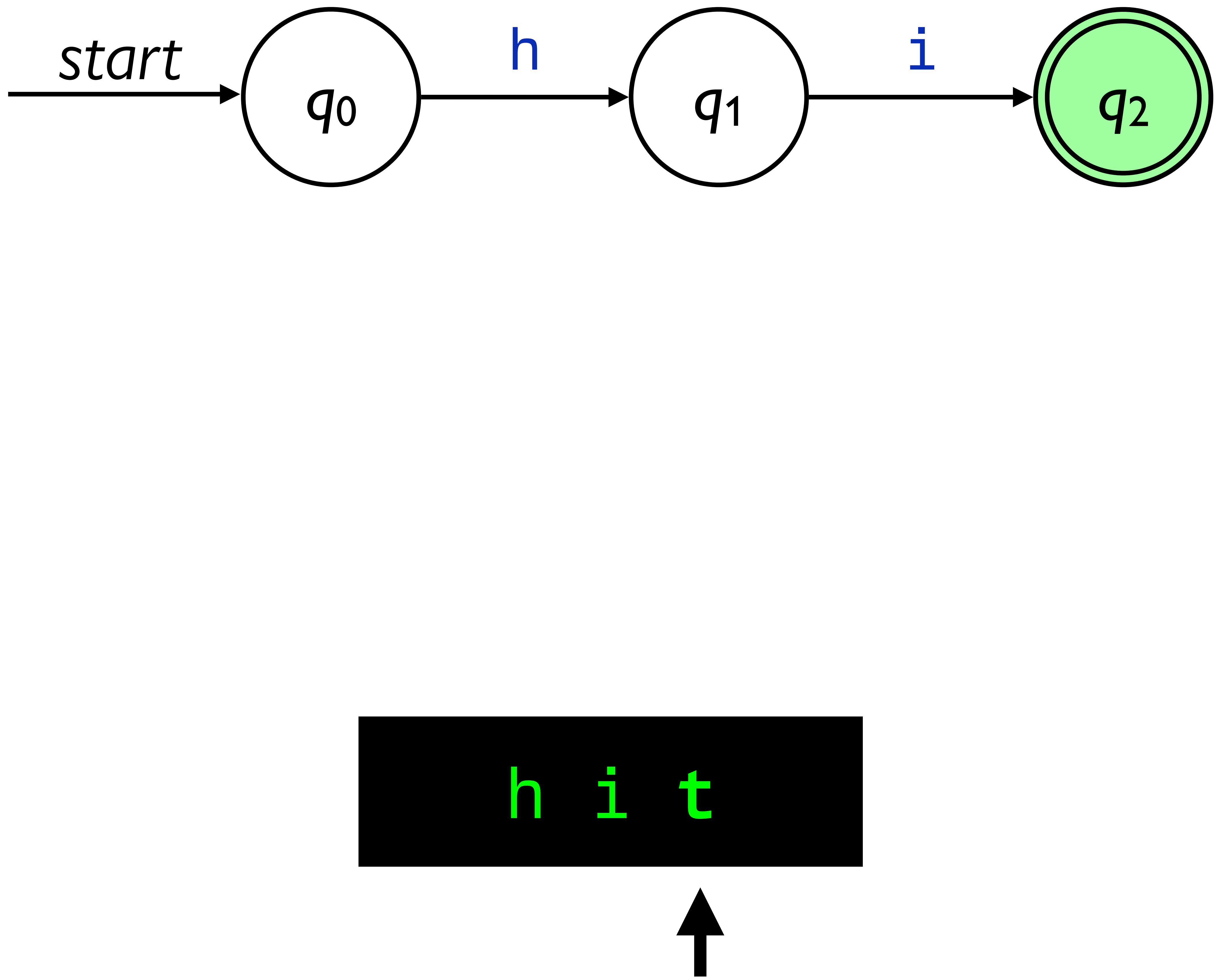


Tragedy in paradise

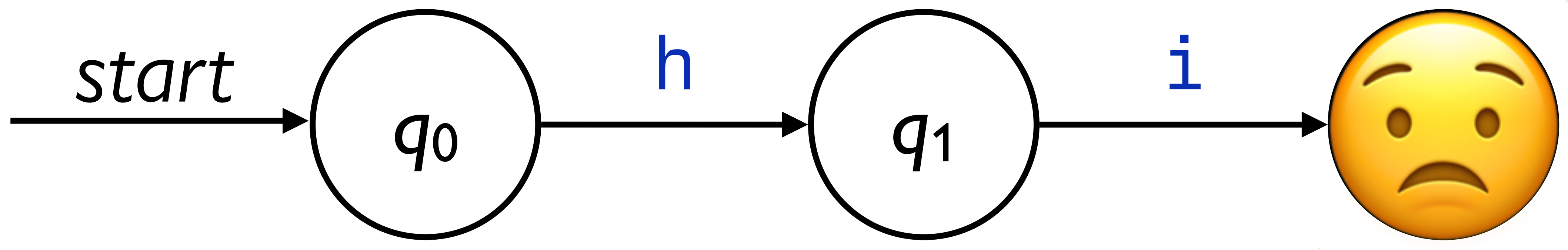




Tragedy in paradise



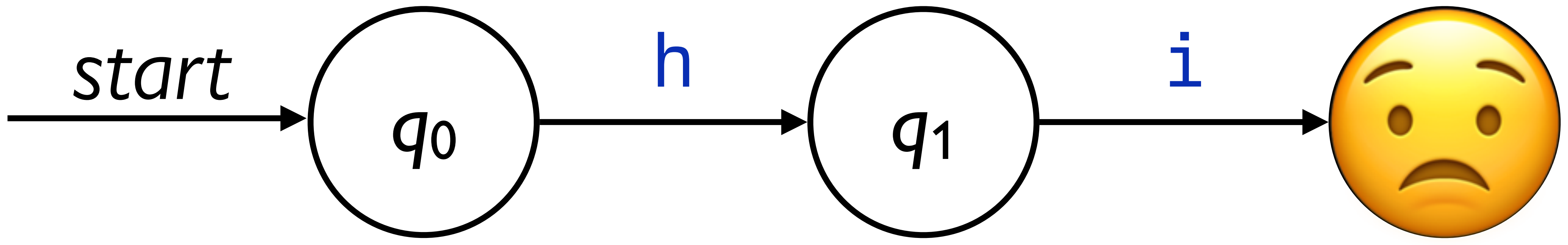
Tragedy in paradise



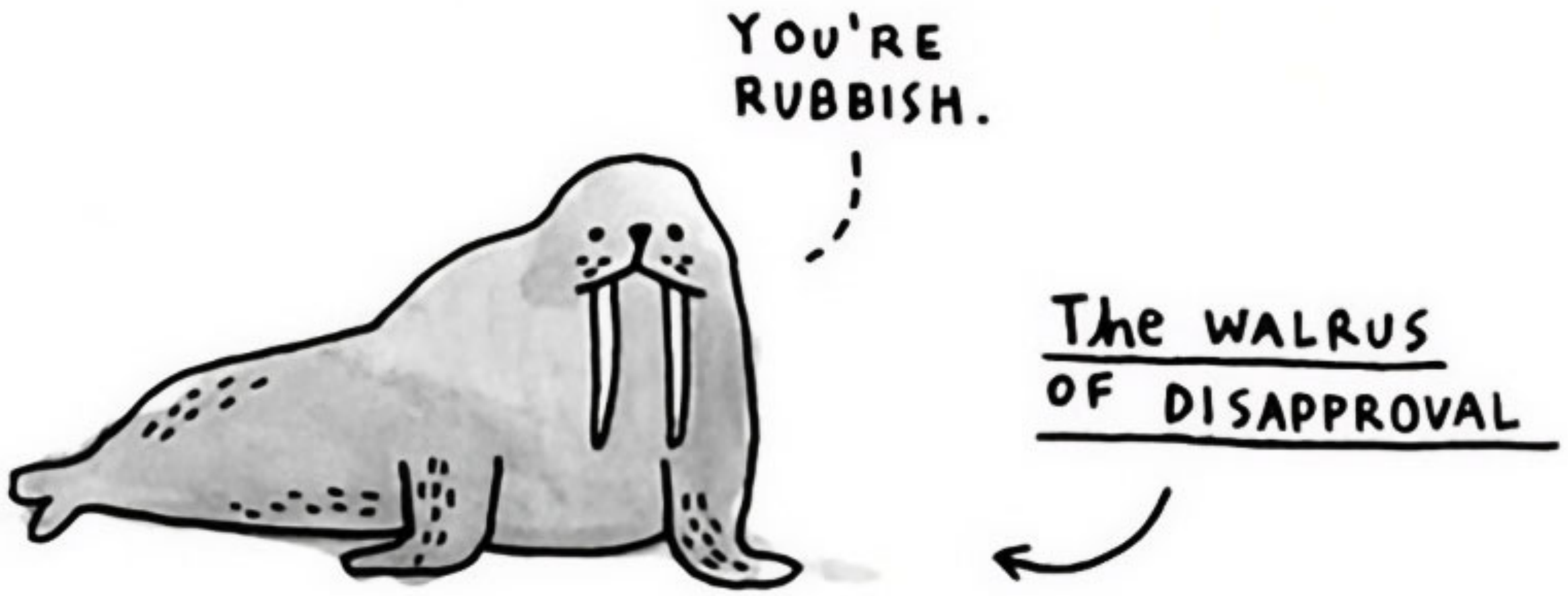

```
h i t
```

Illustration by
Gemma Correll

Tragedy in paradise



`h i t`



Formally, an NFA is defined like a DFA:

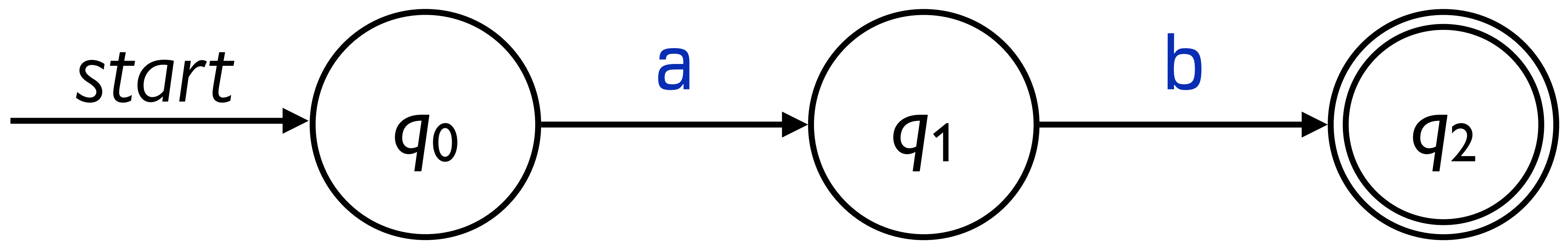
$$N = (Q, \Sigma, \delta, q_0, F)$$

Except now the output of the transition function δ – e.g., $\delta(q_0, \mathrm{a})$ – isn't a single state but a ***set of states***:

$$\delta: Q \times \Sigma \rightarrow \wp(Q)$$

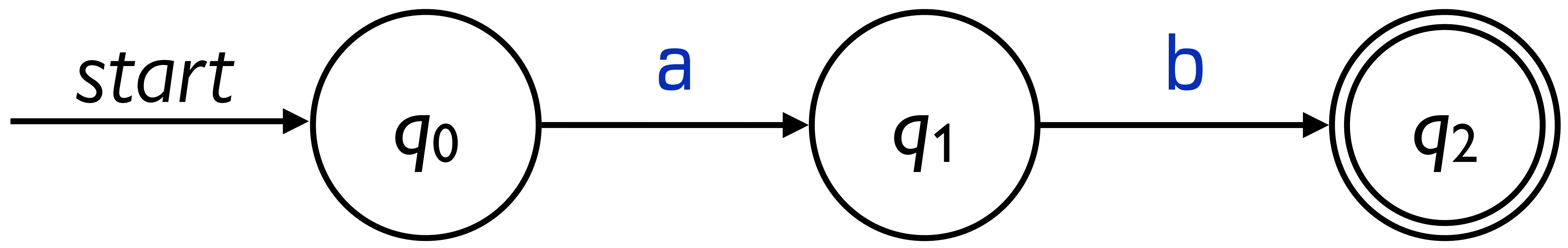
The language of an NFA is

$$L(N) = \{w \in \Sigma^* \mid N \text{ accepts } w\}$$



Let Σ = {a, b}.

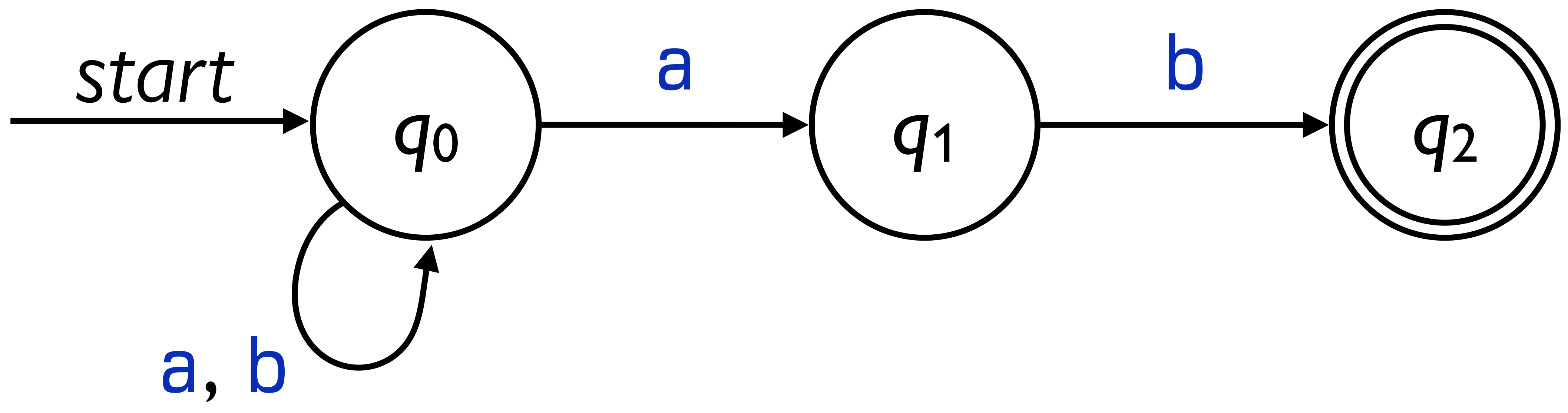
What's the language of this NFA?



Let Σ = {a, b}.

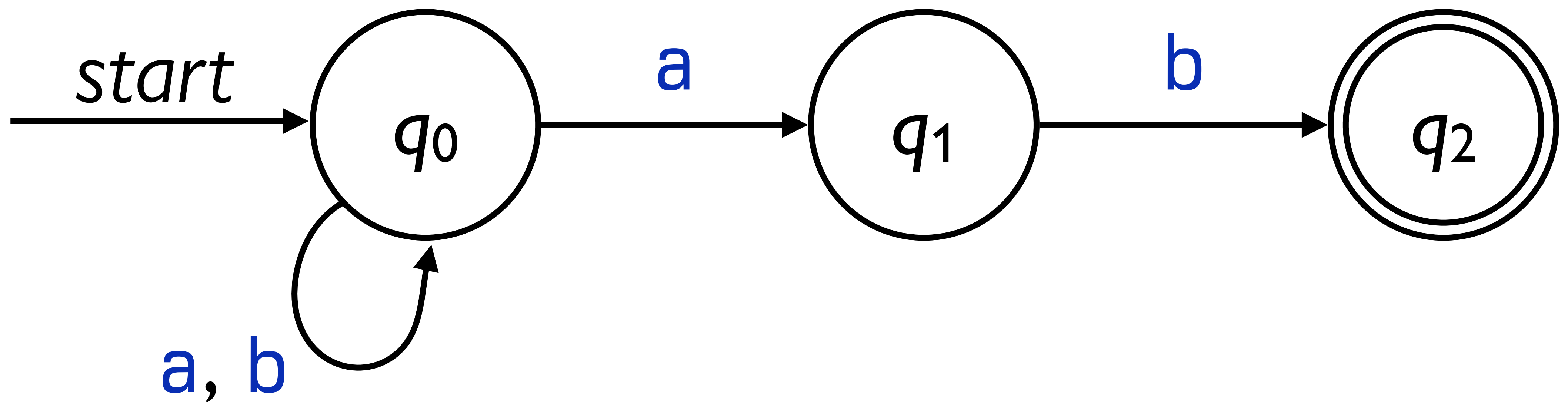
What's the language of this NFA?

L = {ab}



Let $\Sigma = \{a, b\}$.

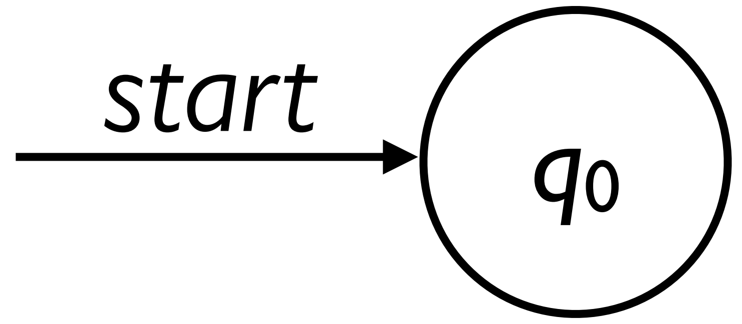
What's the language of this NFA?



Let Σ = {a, b}.

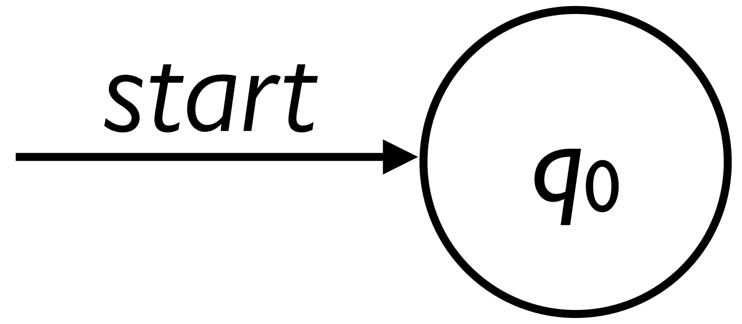
What's the language of this NFA?

$L = \{w \in \Sigma^* \mid$ ab is a suffix of $w\}$



Let Σ = {a, b}.

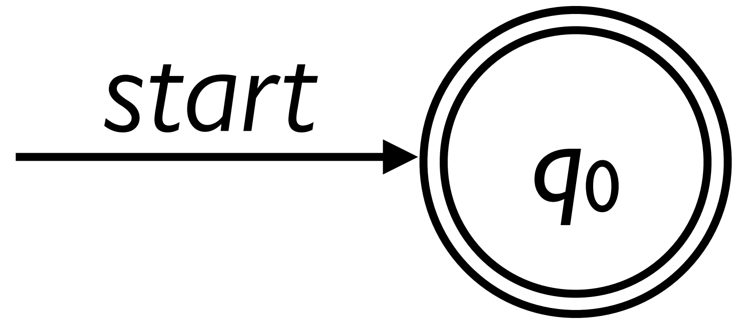
What's the language of this NFA?



Let Σ = {a, b}.

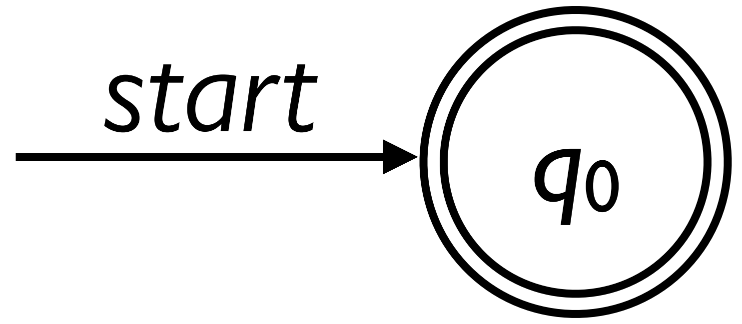
What's the language of this NFA?

$$L = \varnothing$$



Let Σ = {a, b}.

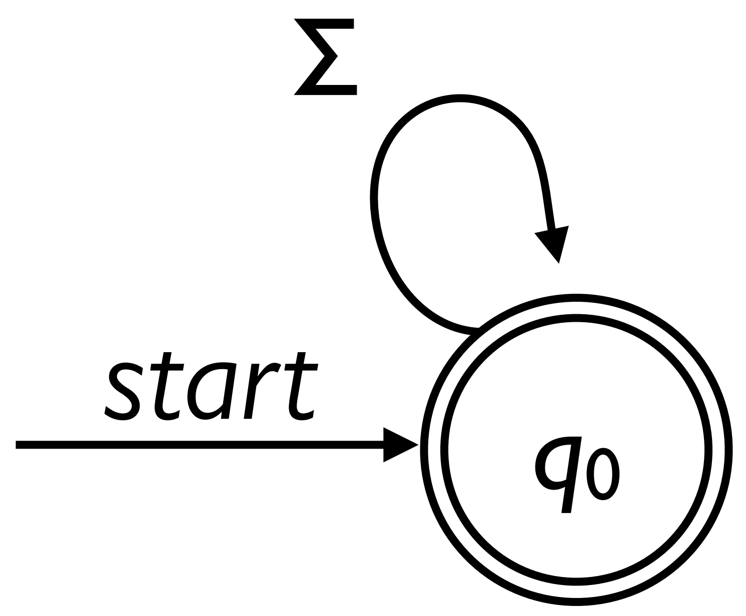
What's the language of this NFA?



Let $\Sigma = \{a, b\}$.

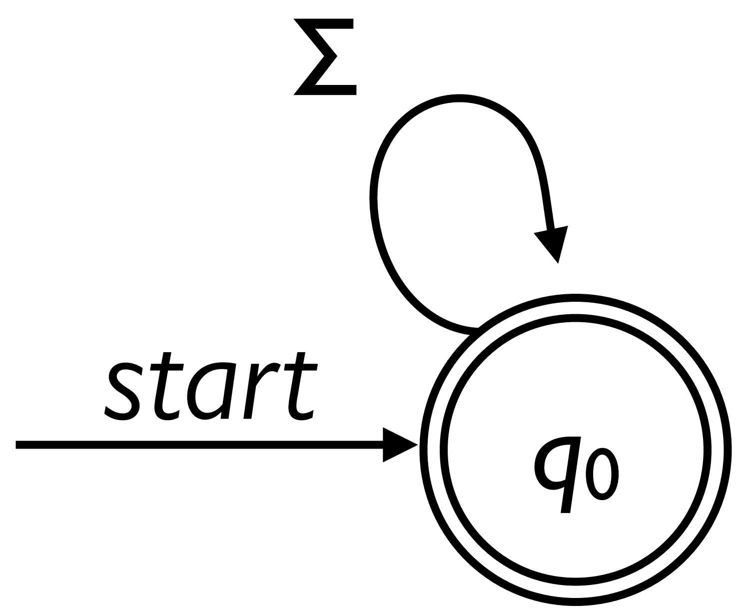
What's the language of this NFA?

$L = \{\varepsilon\}$



Let $\Sigma = \{a, b\}$.

What's the language of this NFA?



Let $\Sigma = \{a, b\}$.

What's the language of this NFA?

$$L = \Sigma^*$$

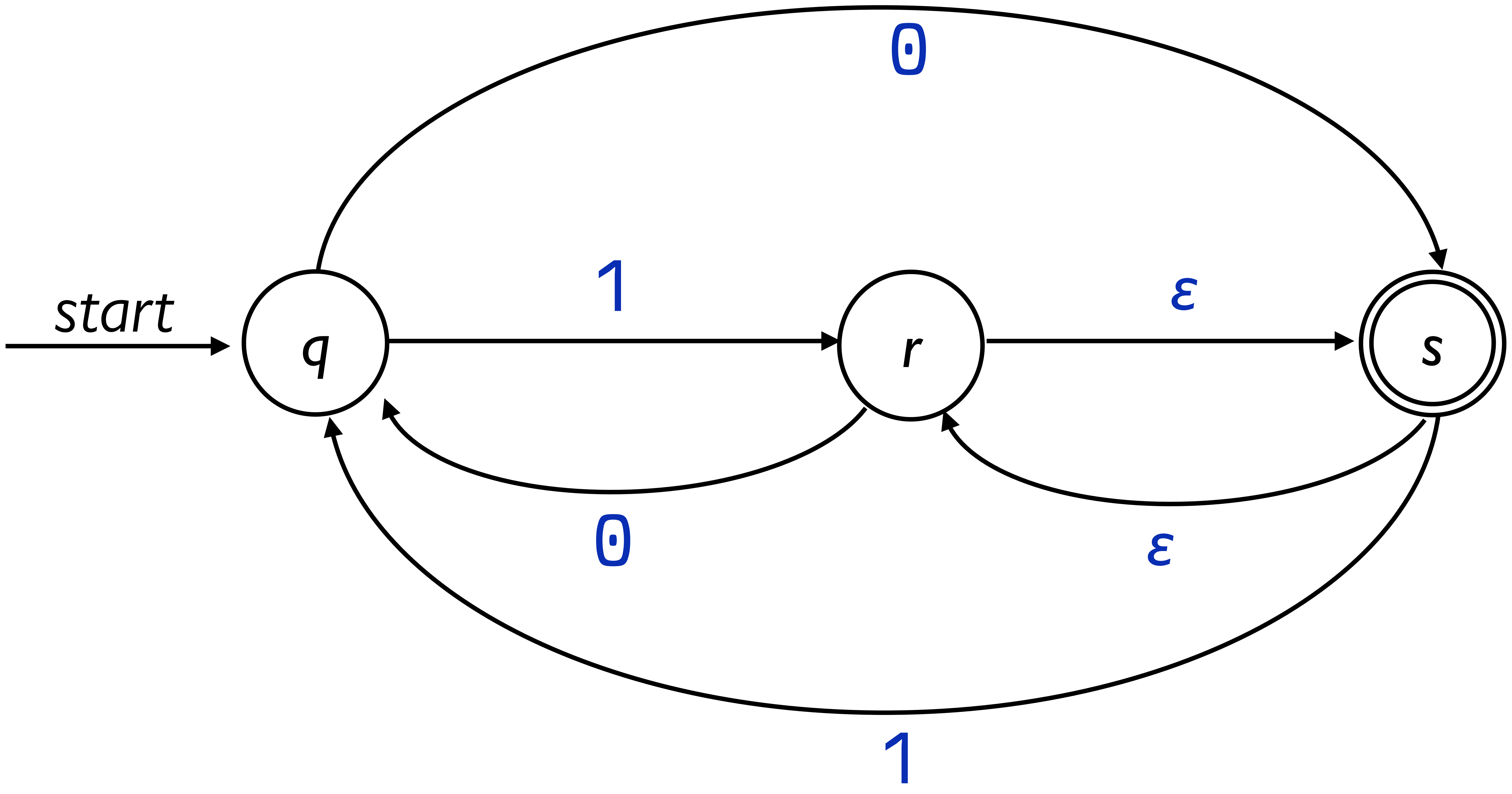
For DFAs, you must read a symbol in order for the machine to make a move.

However, NFAs can move without consuming an input symbol – an ***ε-transition***.

An NFA can follow any number of ε-transitions at any time without consuming any input.

Example

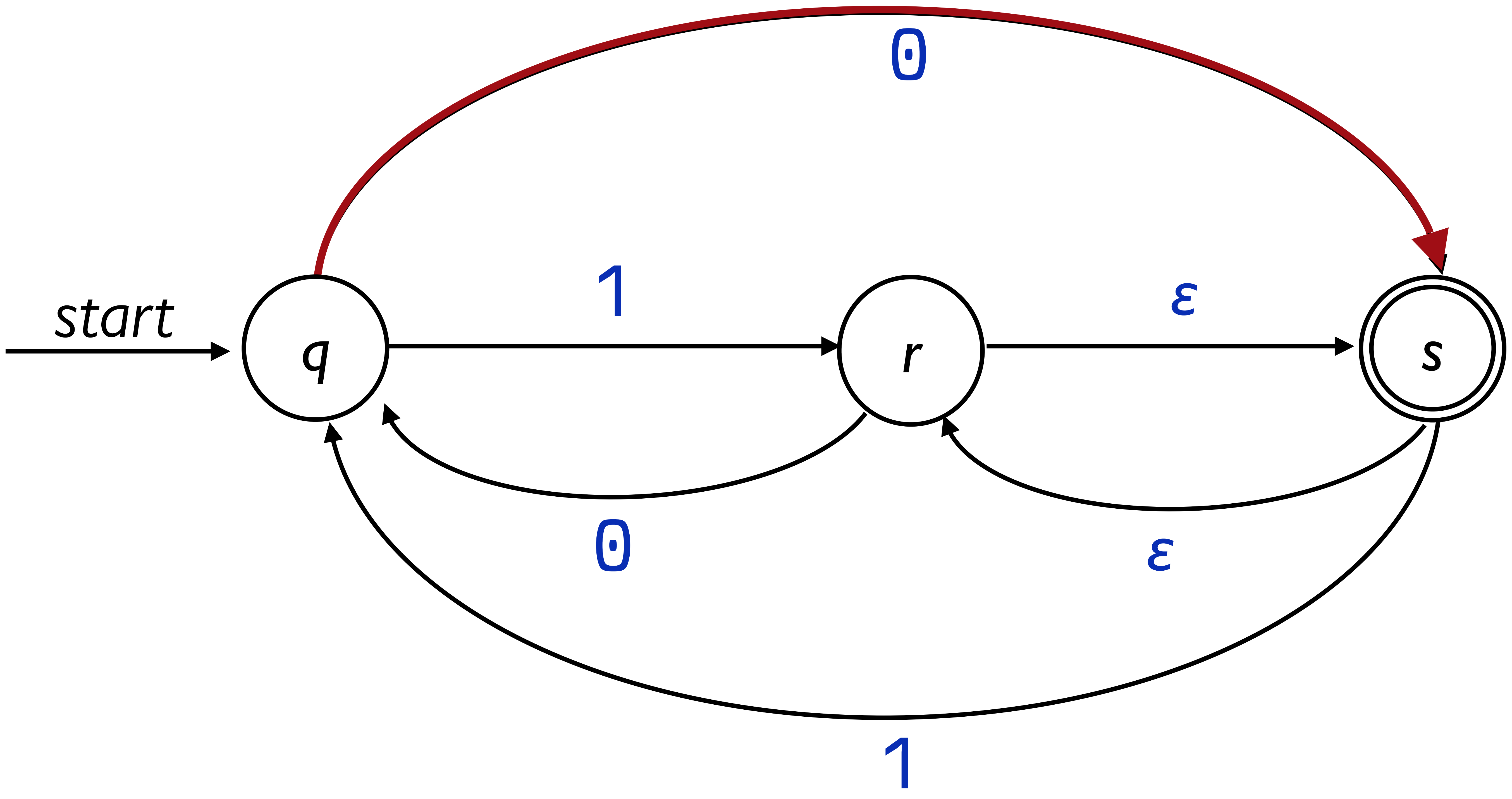
0 0 1



Example

0 0 1

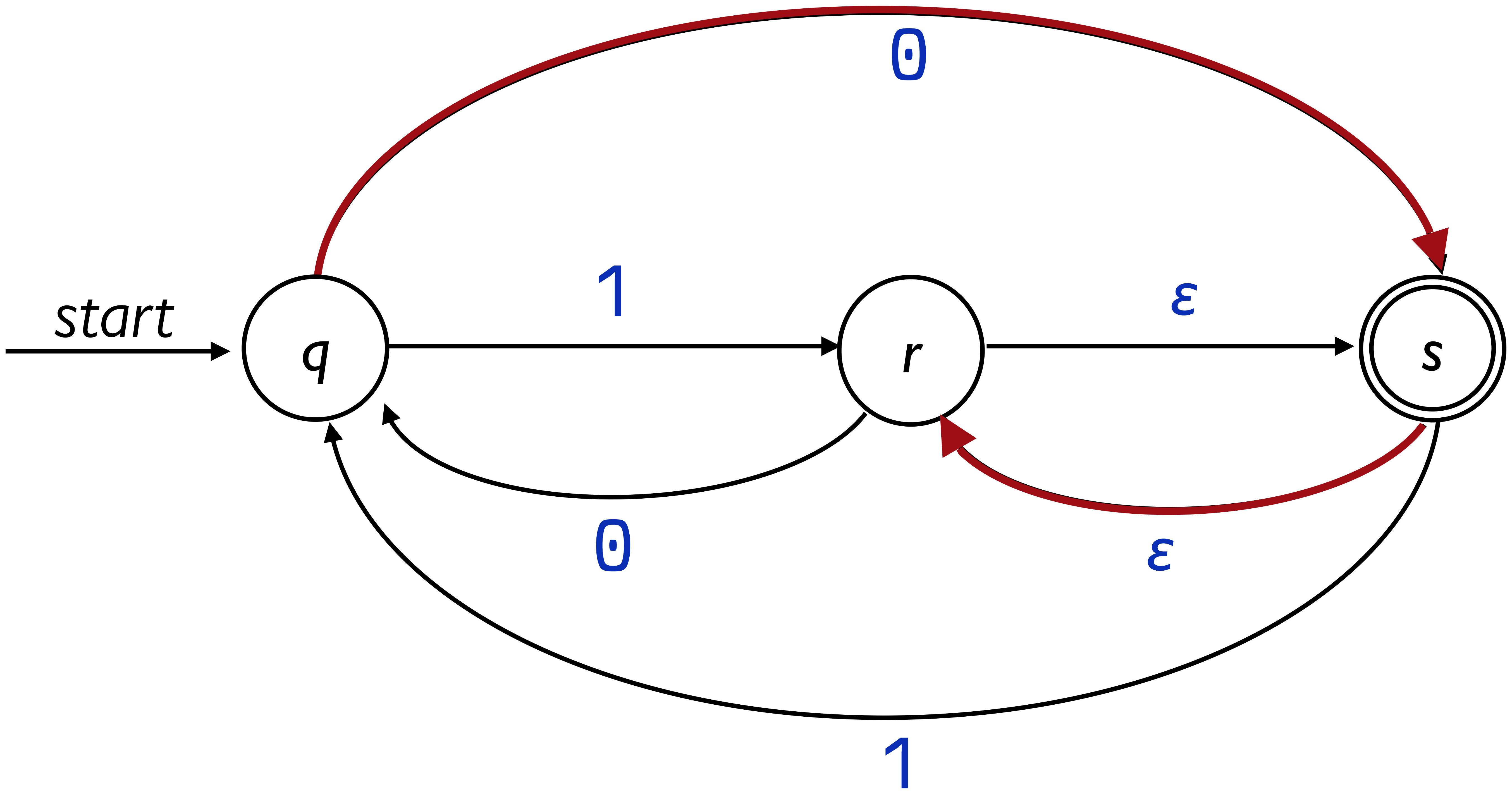
0



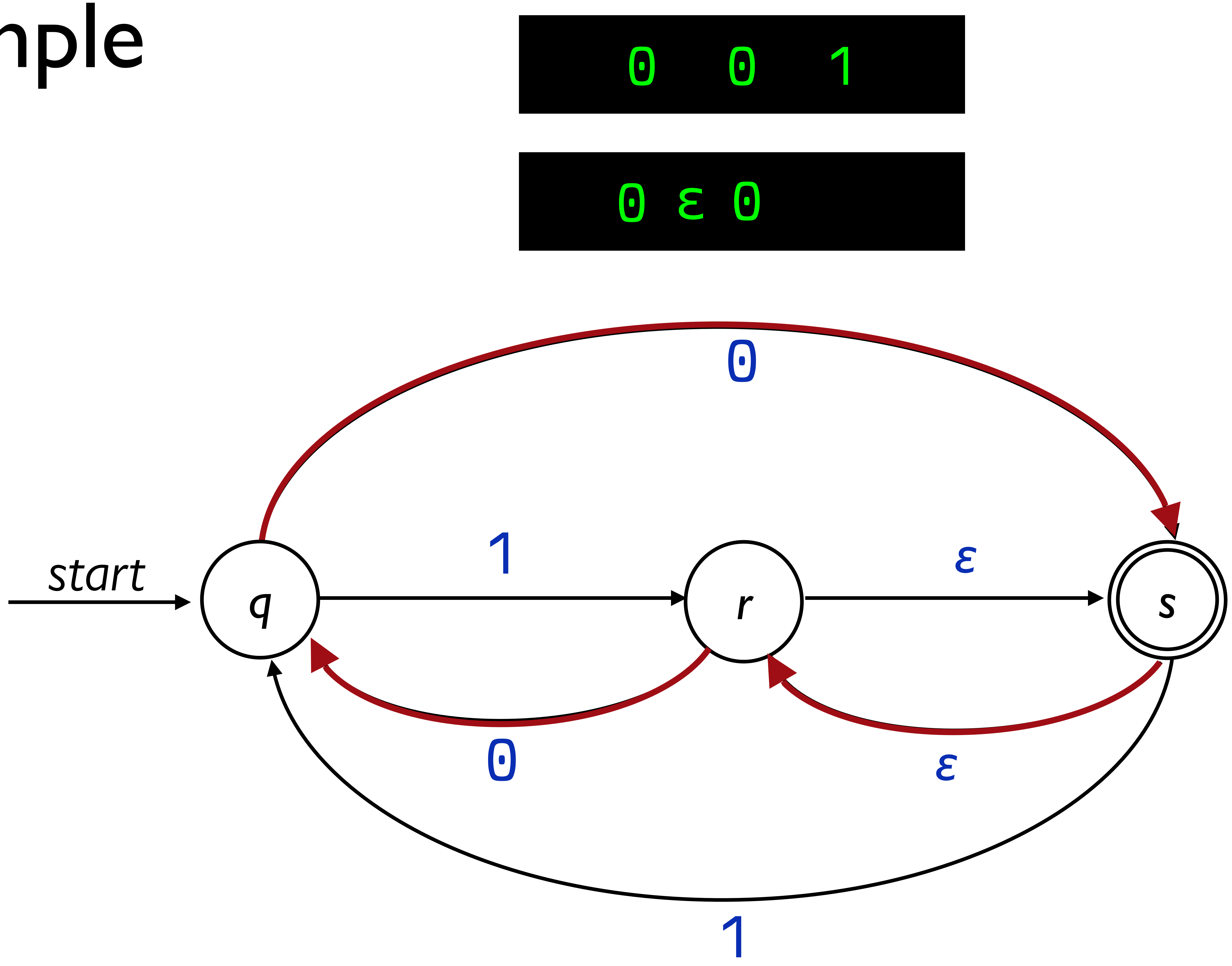
Example

0 0 1

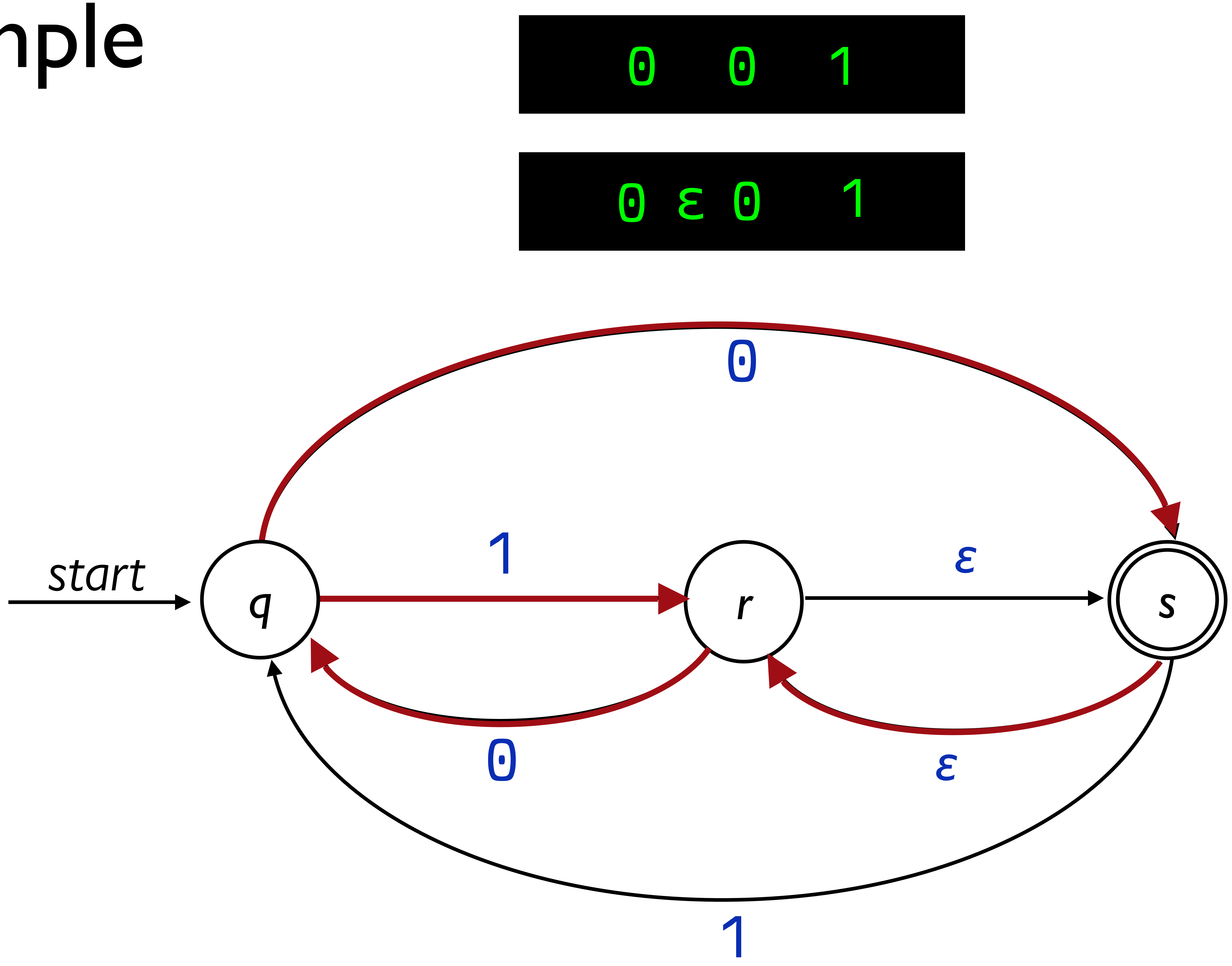
0 ε



Example



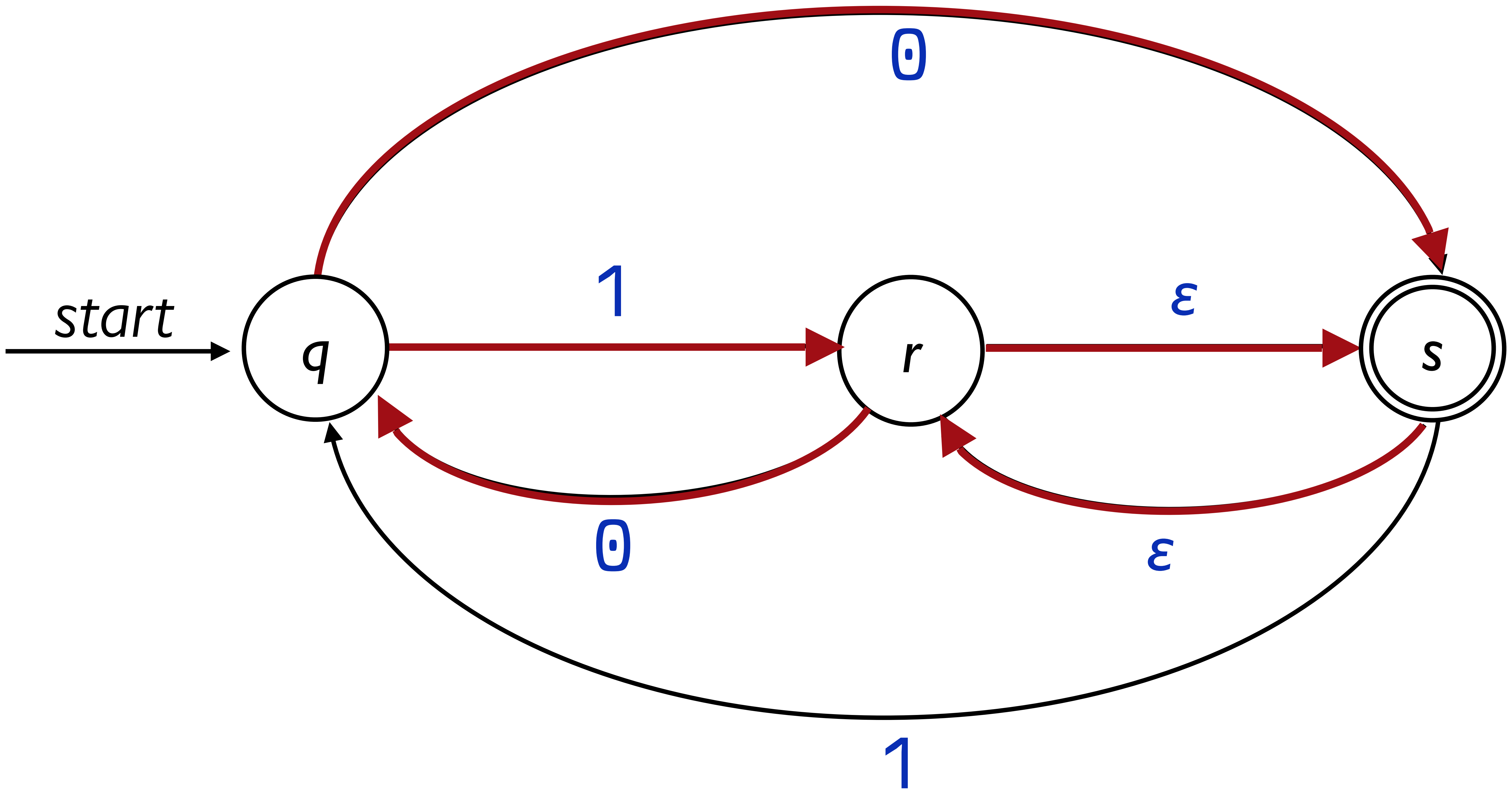
Example



Example

```
0  0  1
```

```
0 ε 0  1 ε
```



NFAs are not *required* to follow ε-transitions; they're just another choice of path for the computation.

Allowing ε-transitions requires one more update to our formal definition:

Now instead of

$$\delta: Q \times \Sigma \to \wp(Q)$$

we have

$$\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to \wp(Q).$$

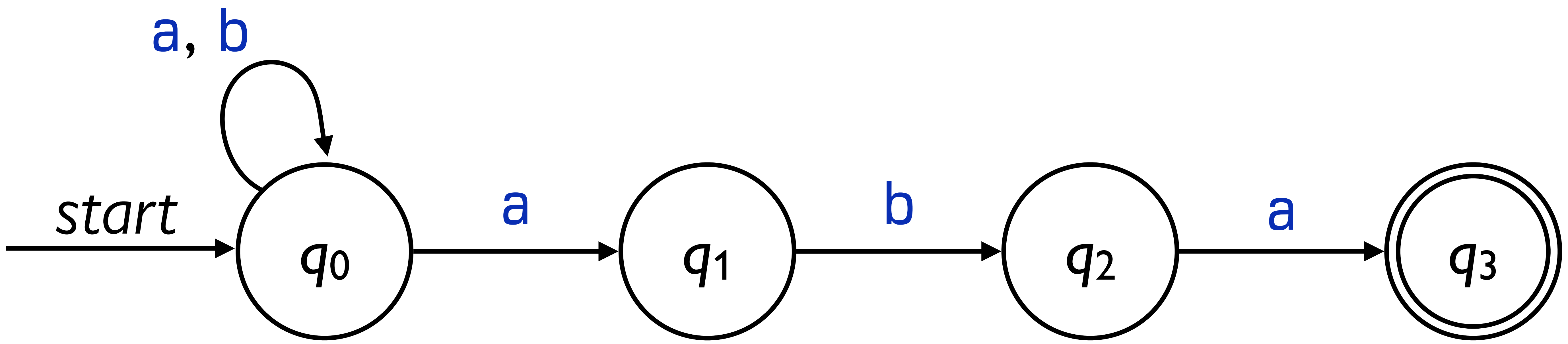
Thinking about NFAs

Nondeterministic machines are a serious departure from physical computers.

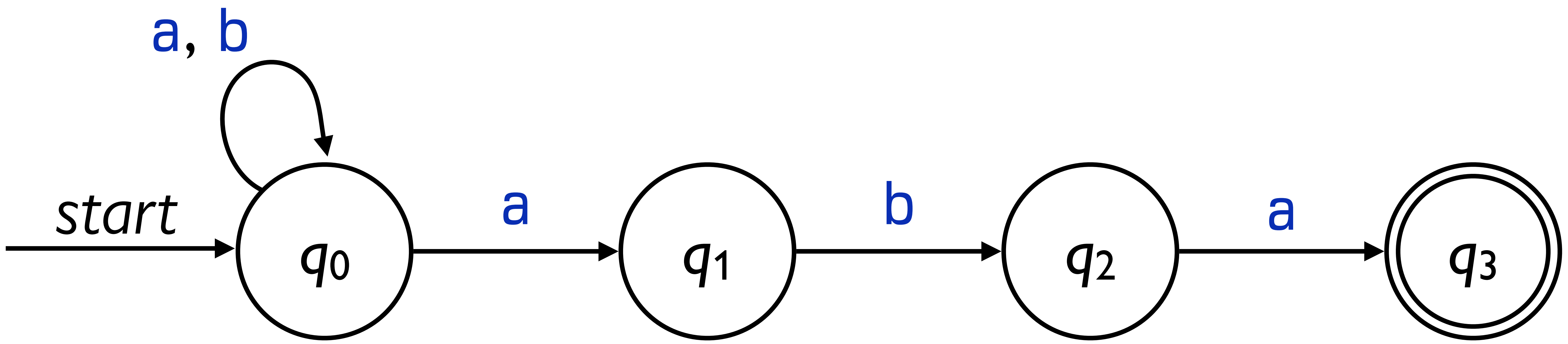
There are two helpful ways to think about nondeterministic computation:

- Perfect positive guessing
- Massive parallelism

Perfect positive guessing

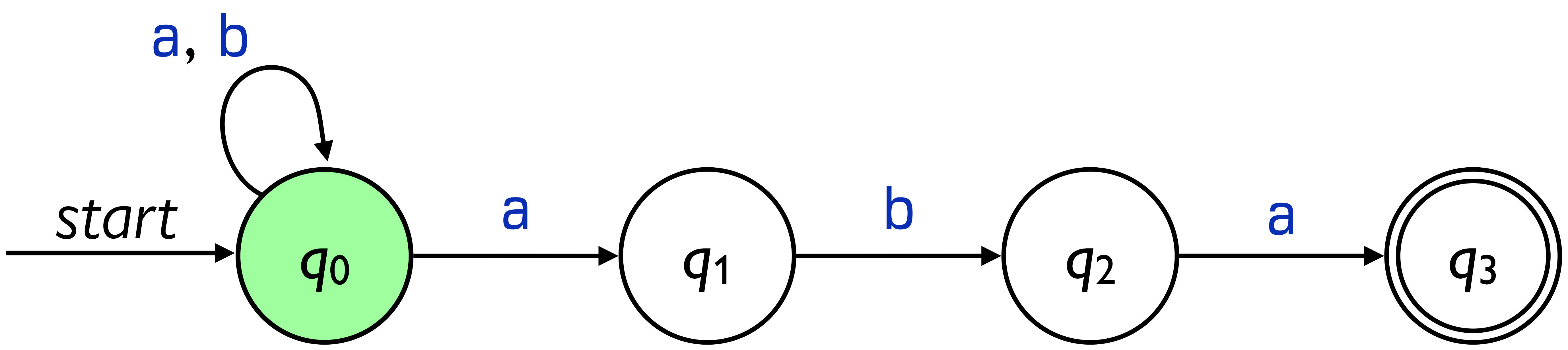


Perfect positive guessing



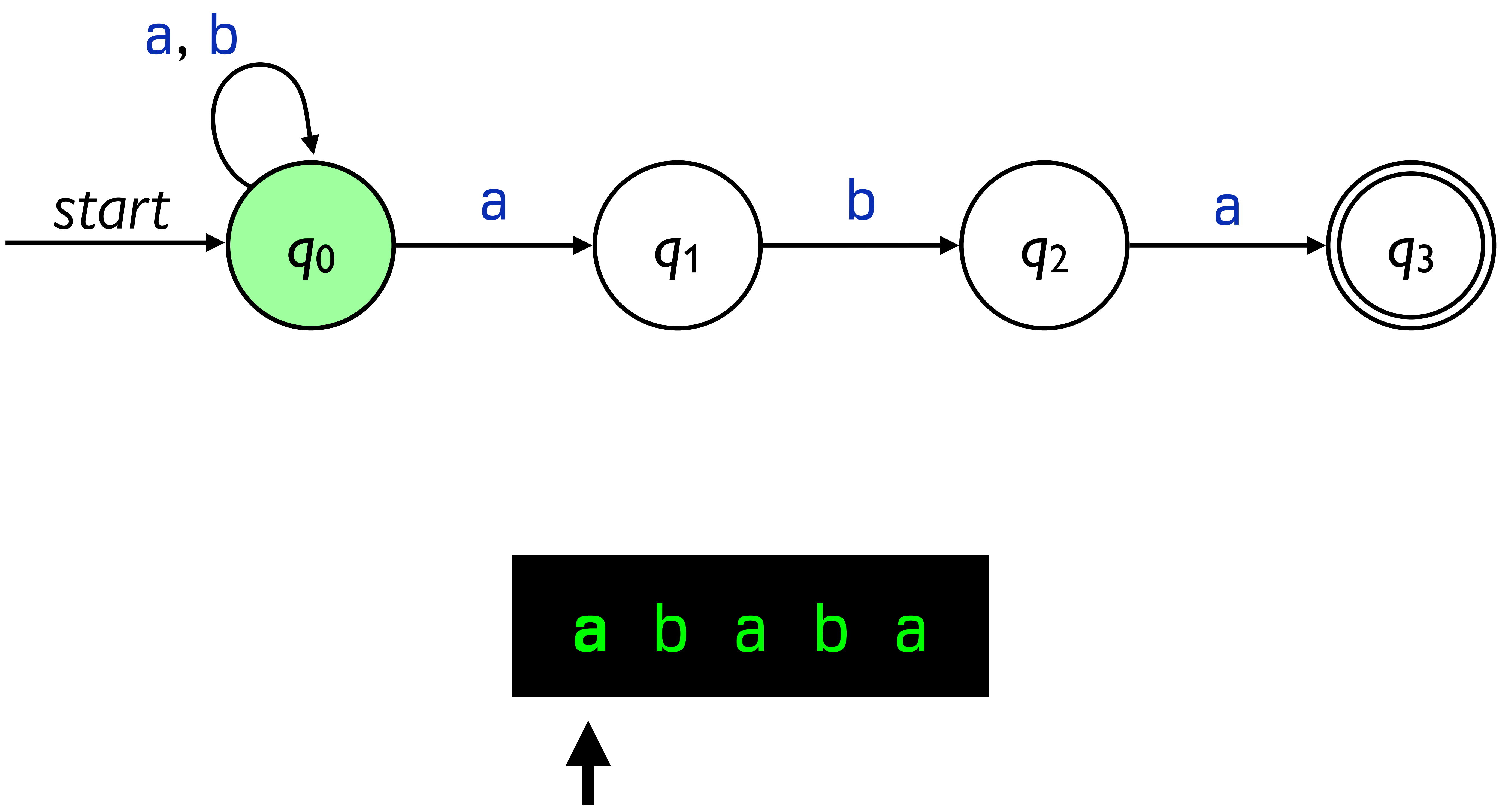
a b a b a

Perfect positive guessing

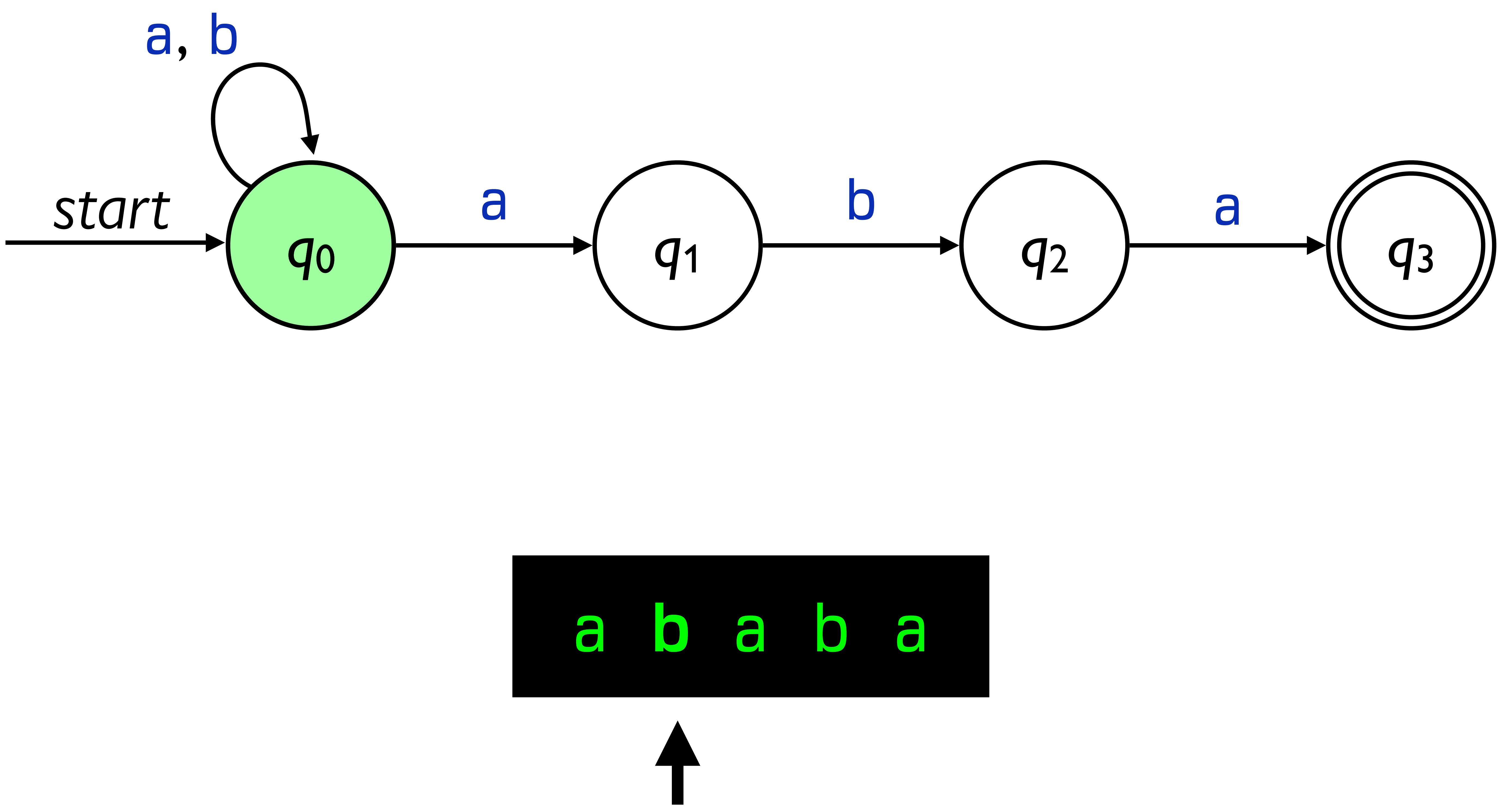


a b a b a

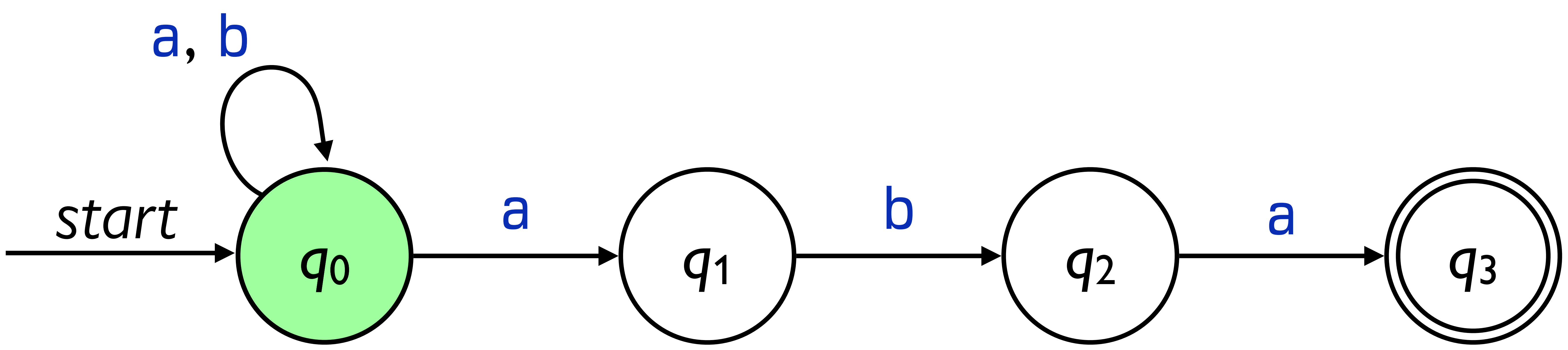
Perfect positive guessing

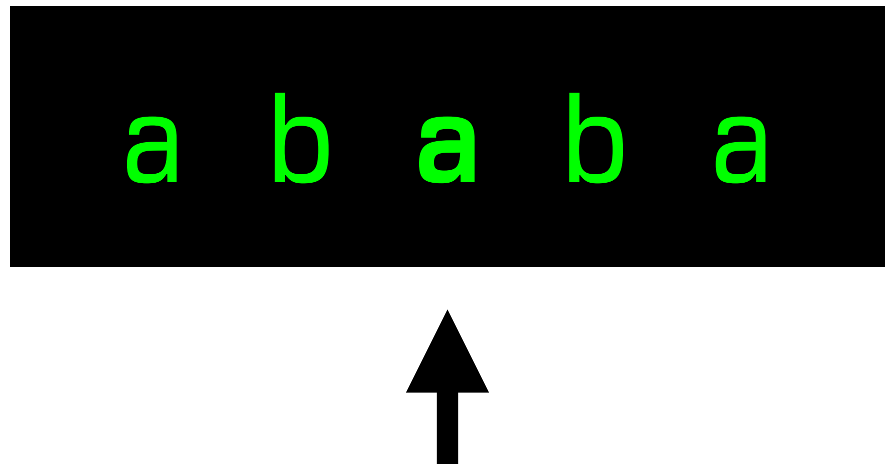


Perfect positive guessing

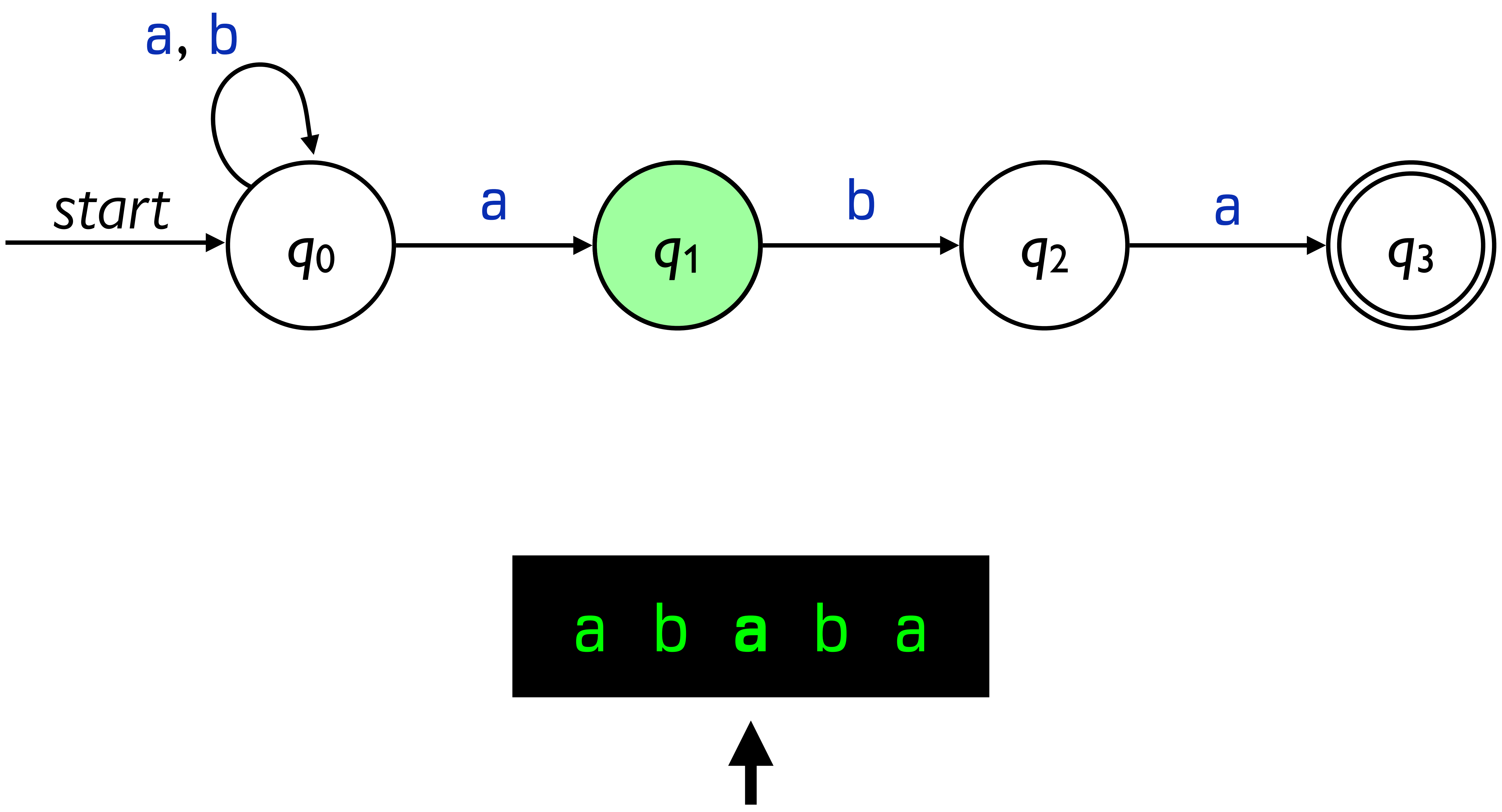


Perfect positive guessing

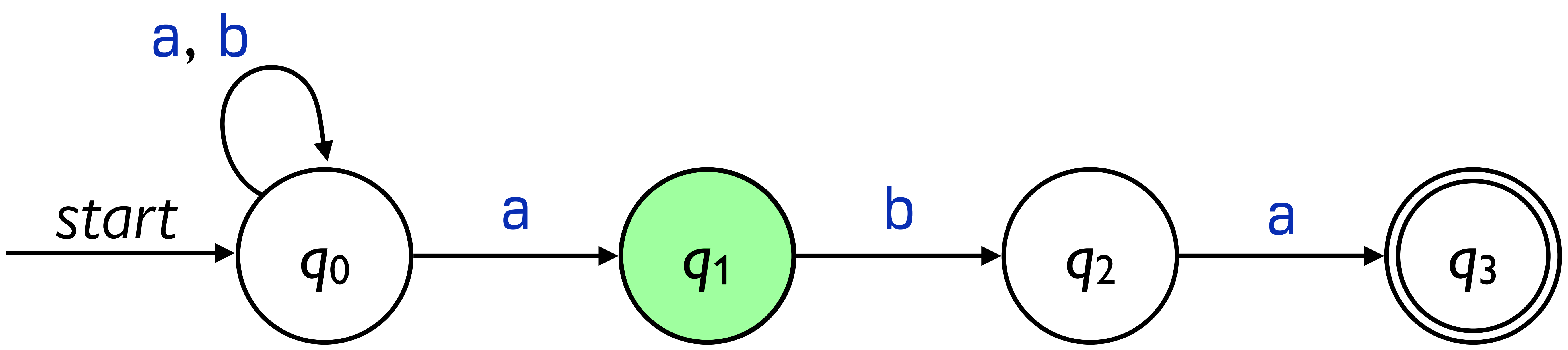


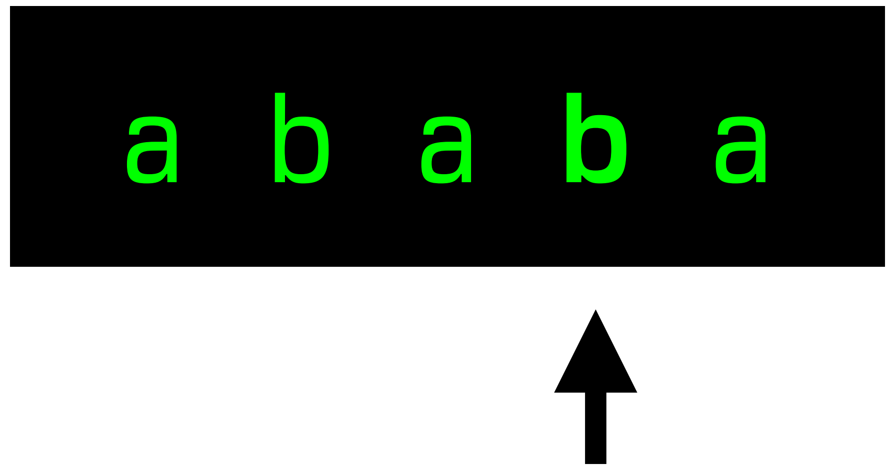


Perfect positive guessing

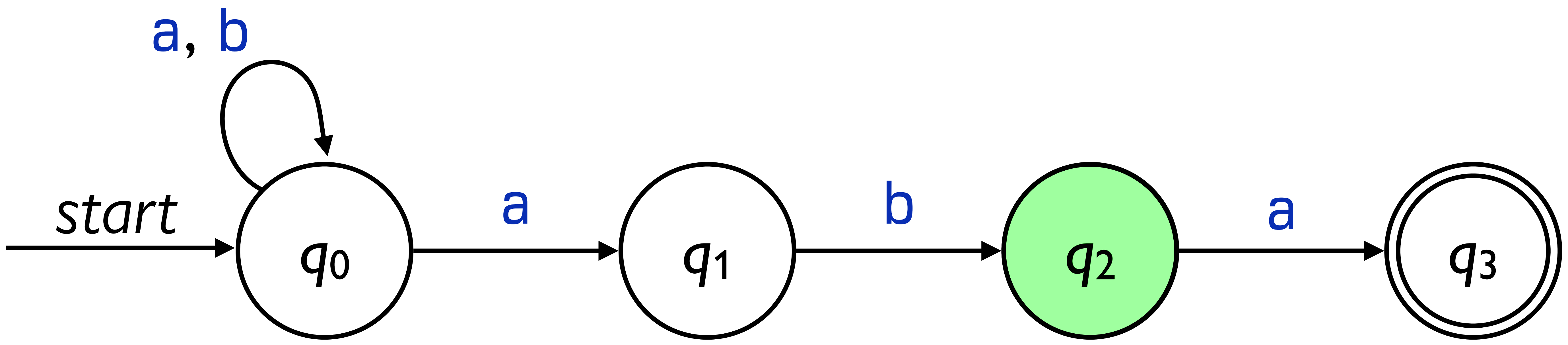


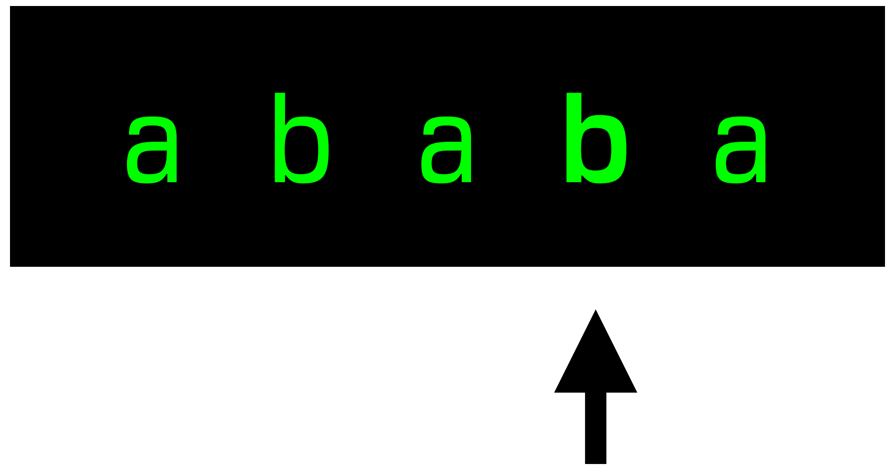
Perfect positive guessing



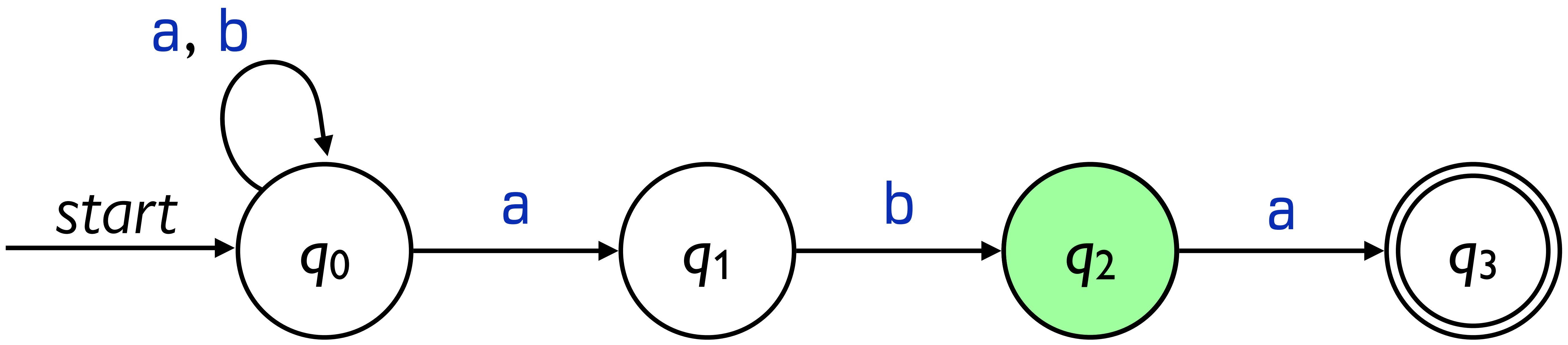


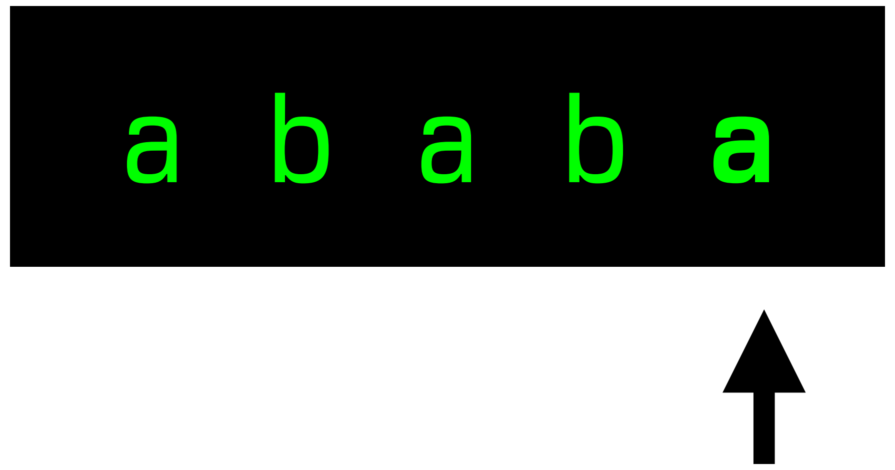
Perfect positive guessing



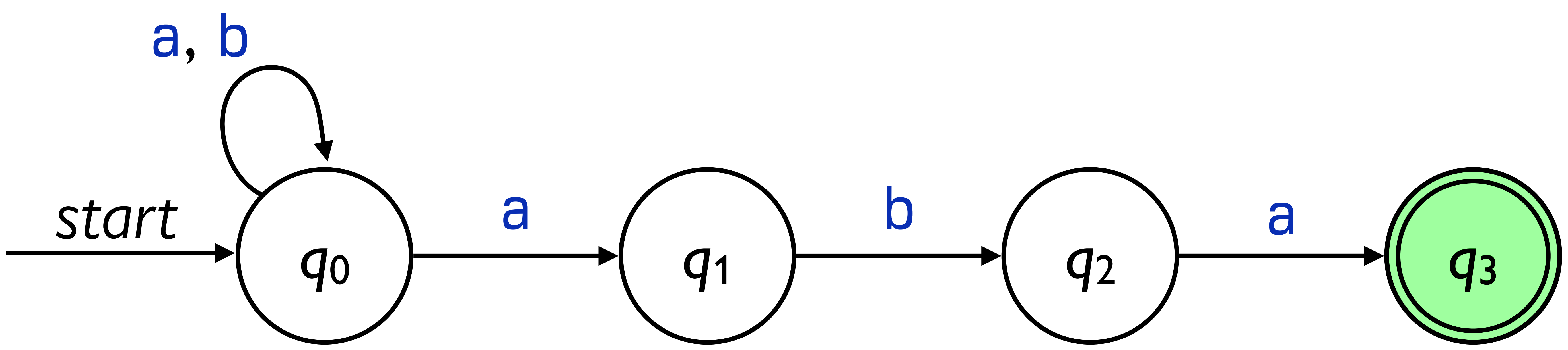


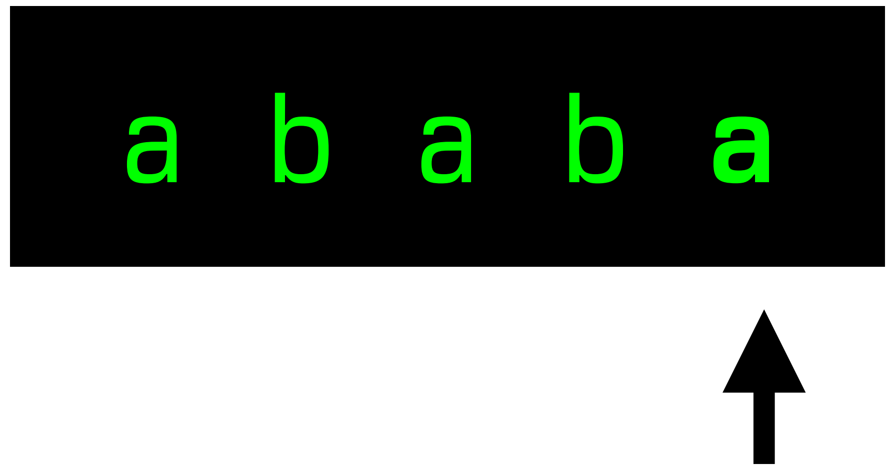
Perfect positive guessing



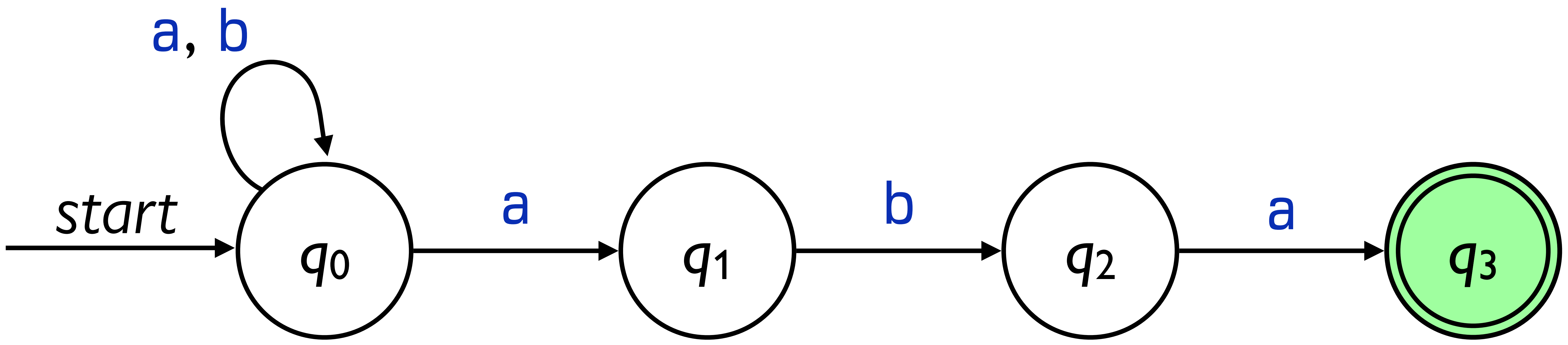


Perfect positive guessing





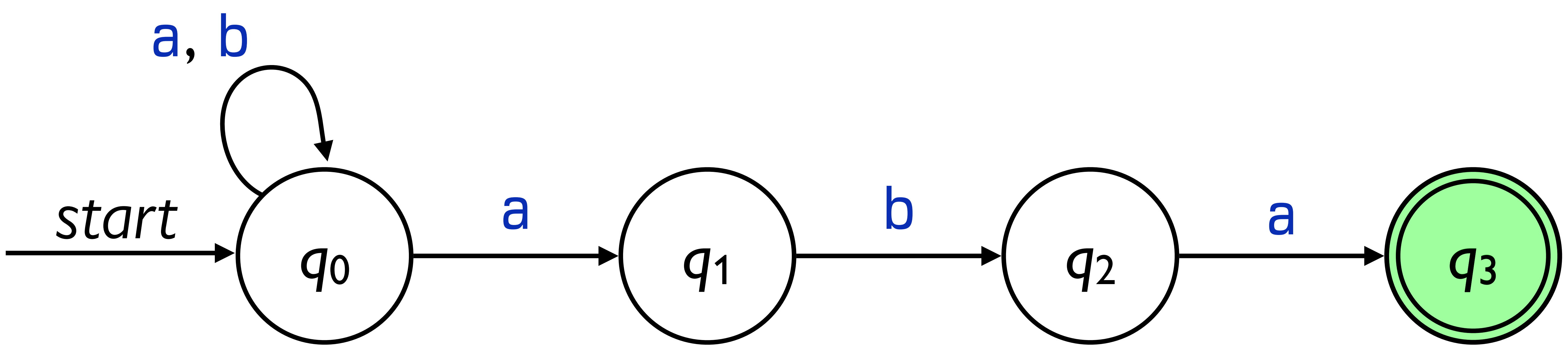
Perfect positive guessing



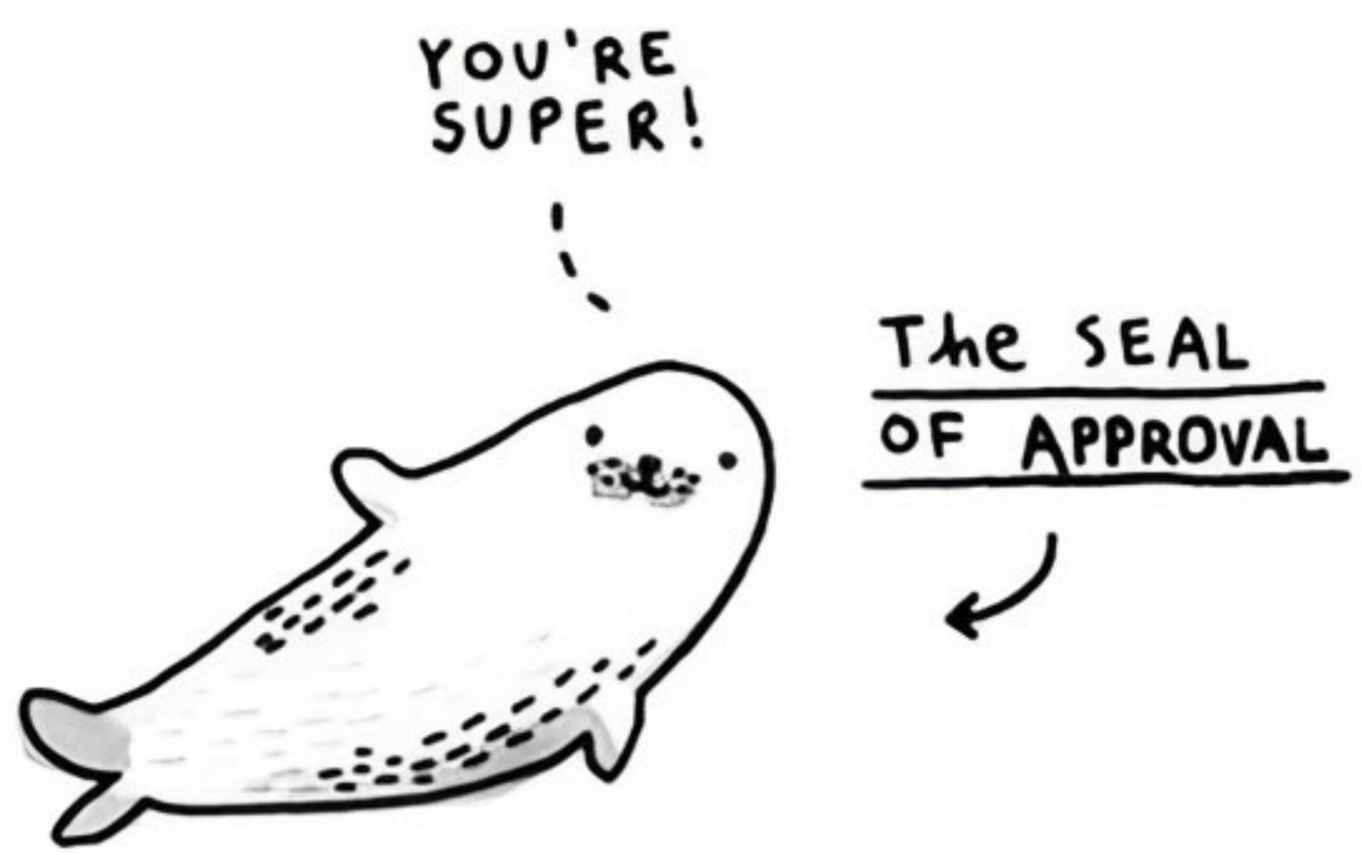
a b a b a

Illustration by Gemma Correll

Perfect positive guessing



a b a b a



NFAs have a "Liquid Luck" potion

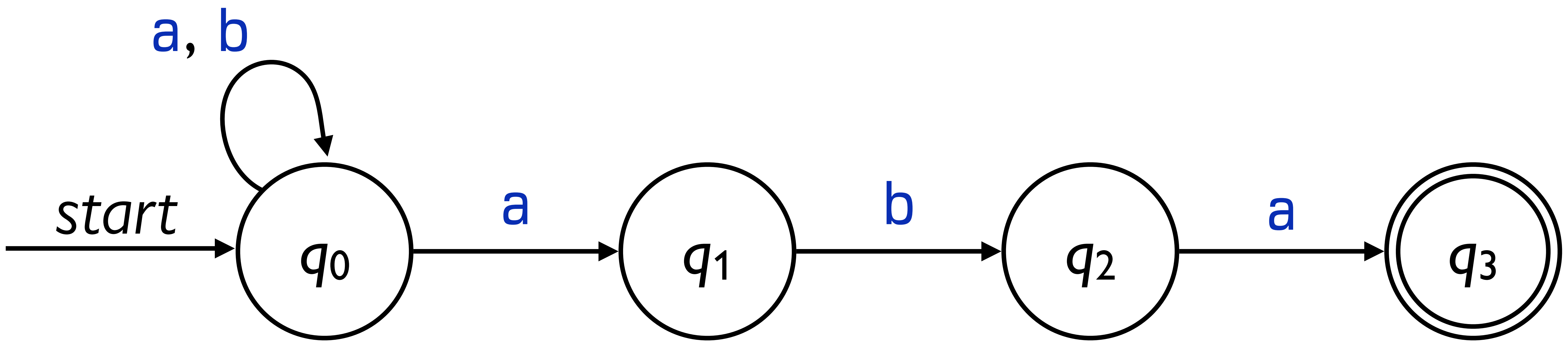
Perfect positive guessing

We can think of nondeterministic machines as having *magic powers* that enable them to guess the correct choice of moves to make.

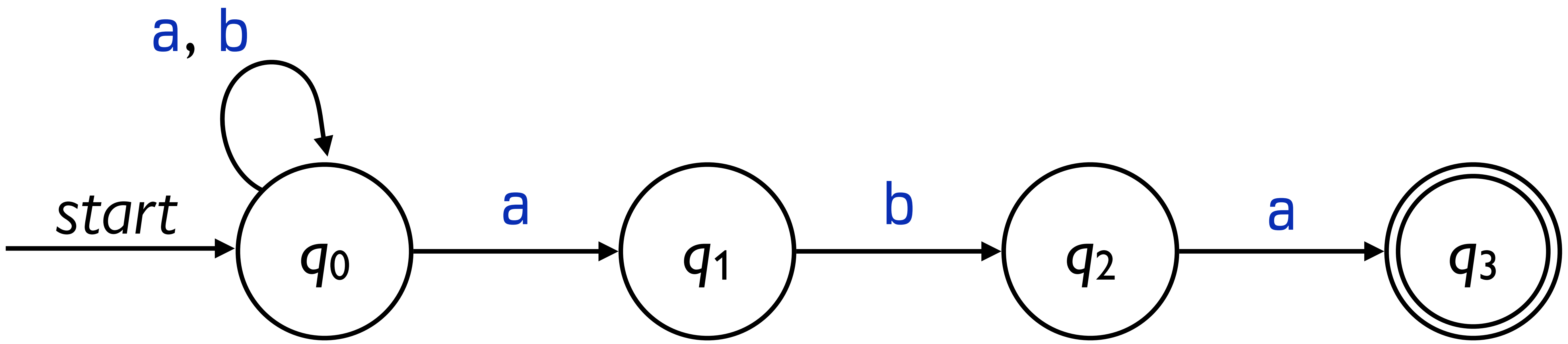
- If there is at least one choice leading to an accept state for the input, the machine will guess it.
- If there are no choices, the machine guesses any one of the wrong answers.

There's no known way to physically model this intuition for nondeterminism; we have left reality.

Massive parallelism

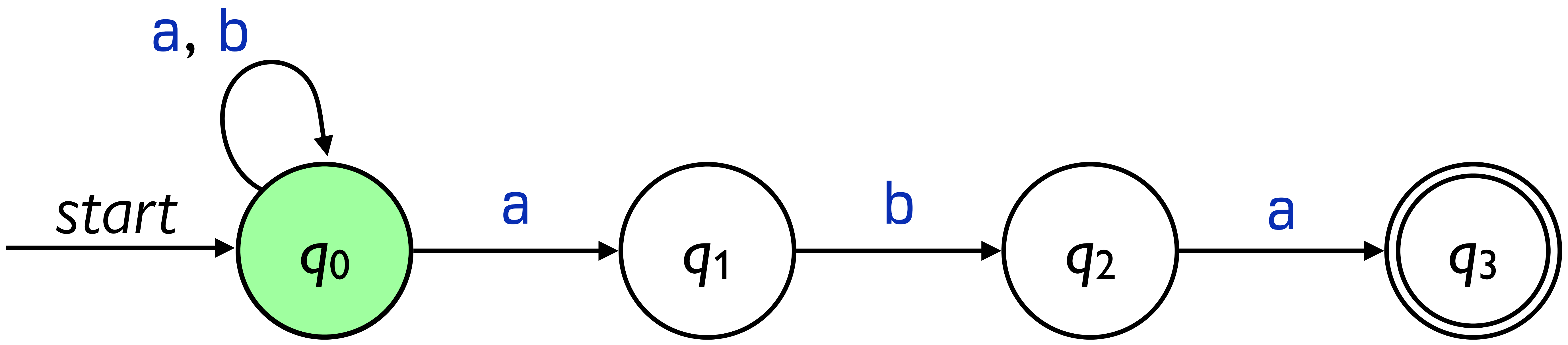


Massive parallelism



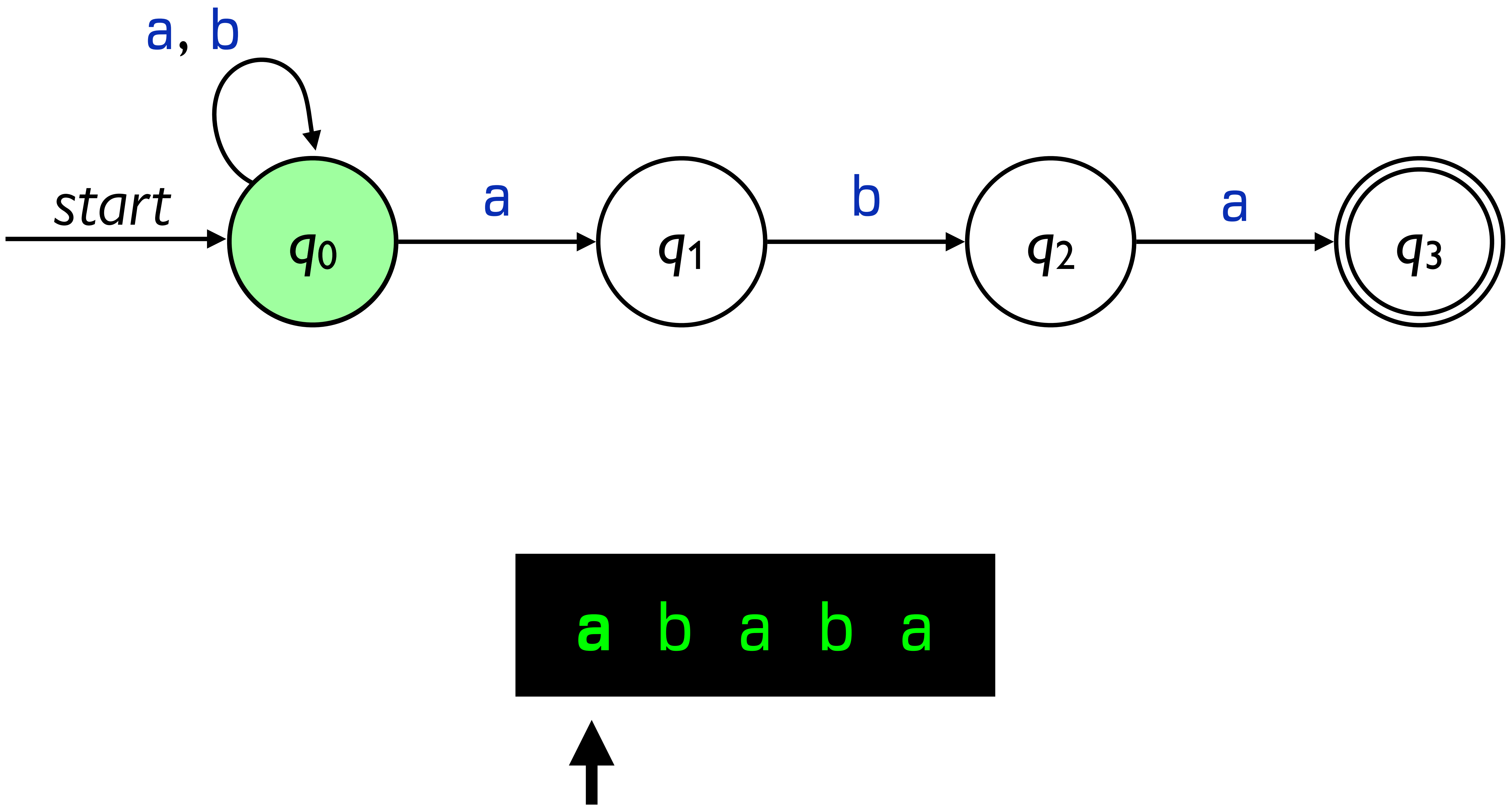
a b a b a

Massive parallelism

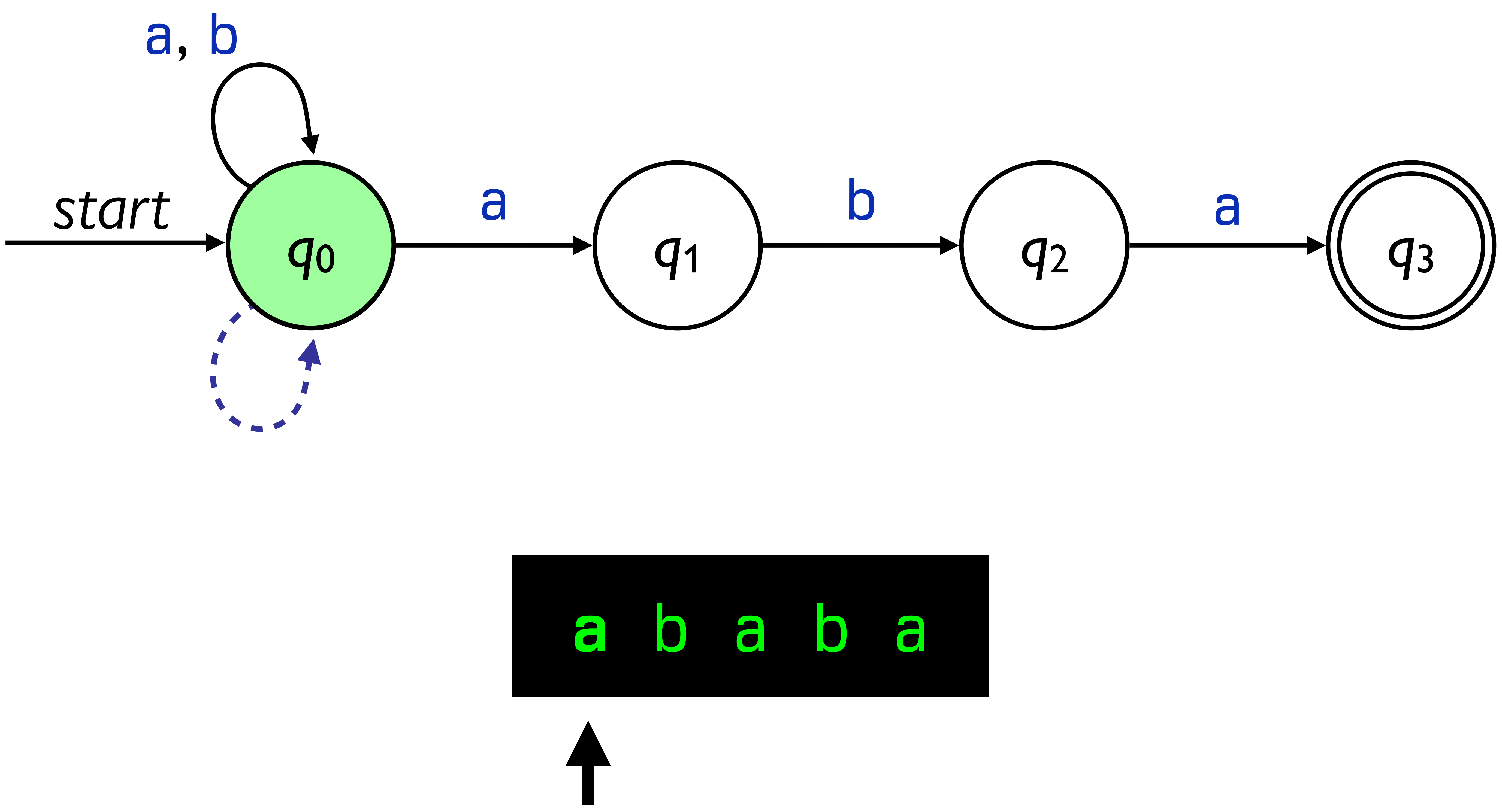


a b a b a

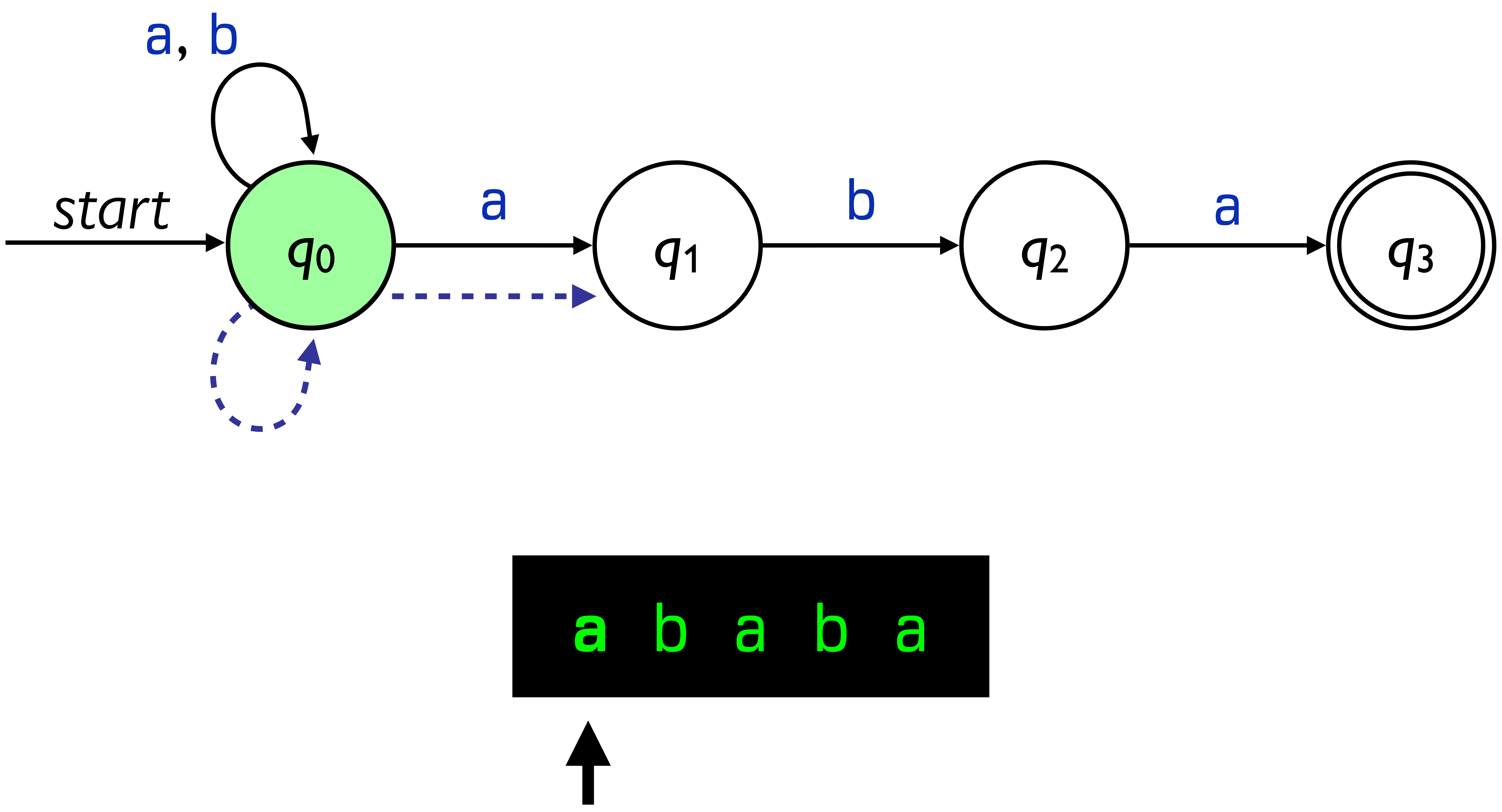
Massive parallelism



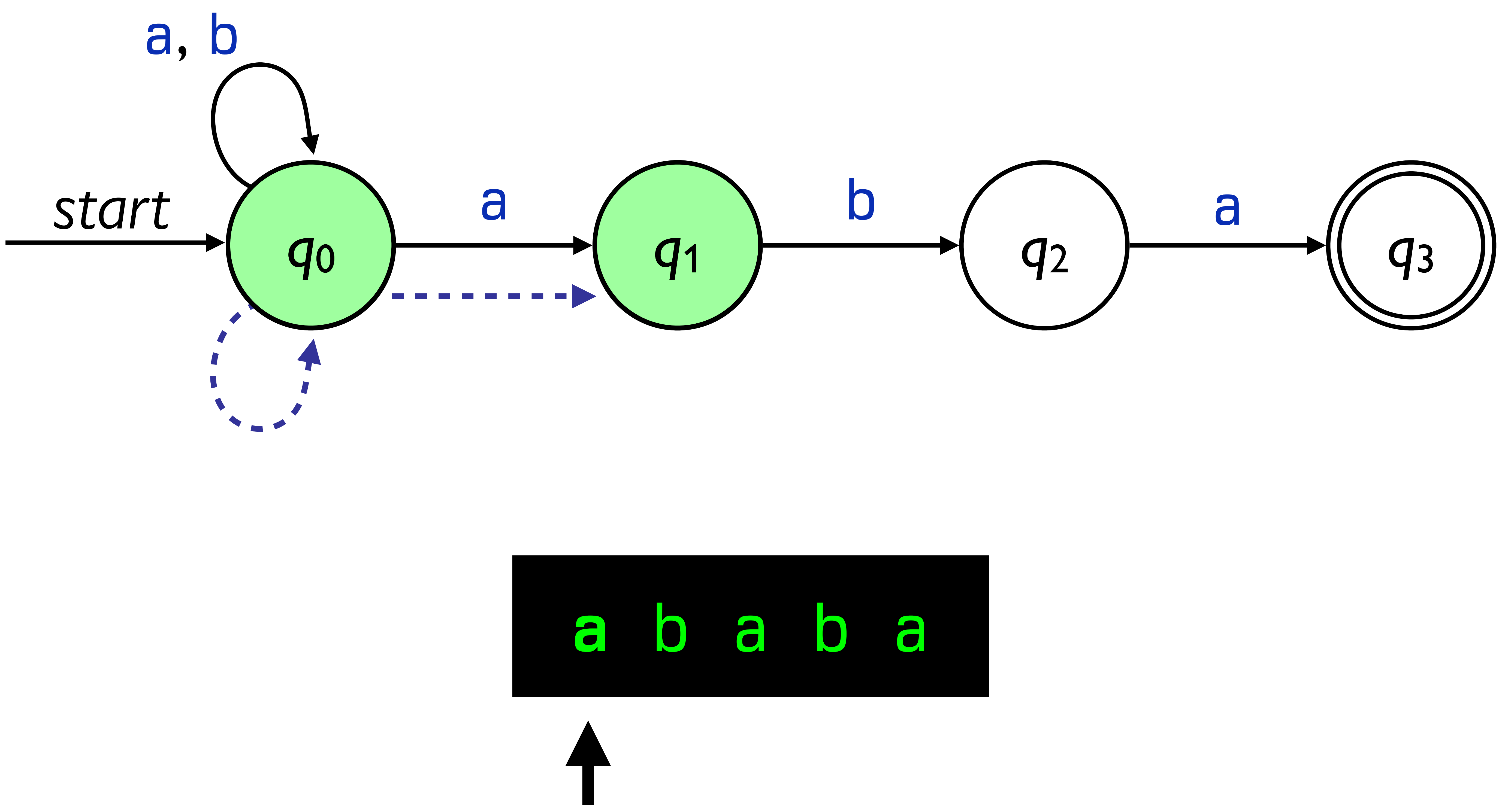
Massive parallelism



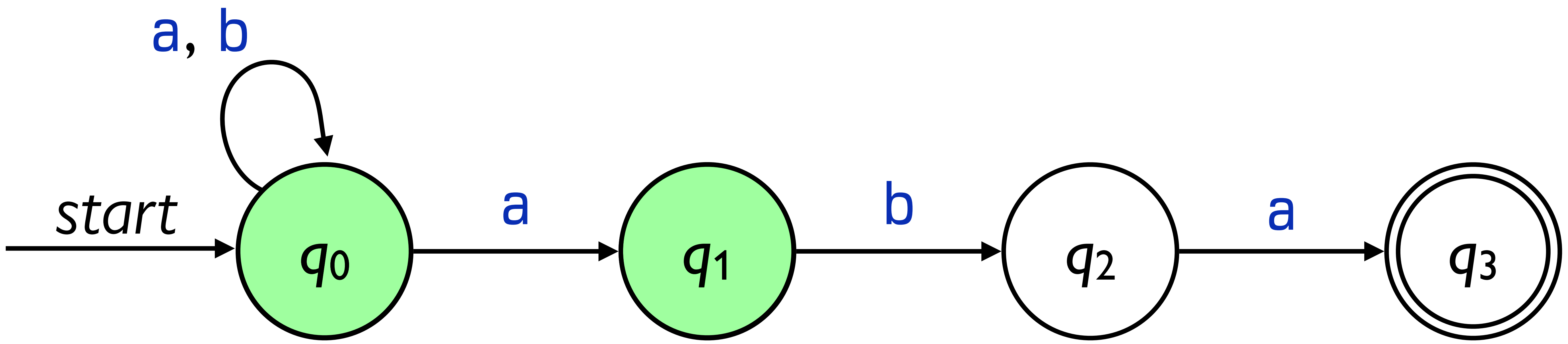
Massive parallelism

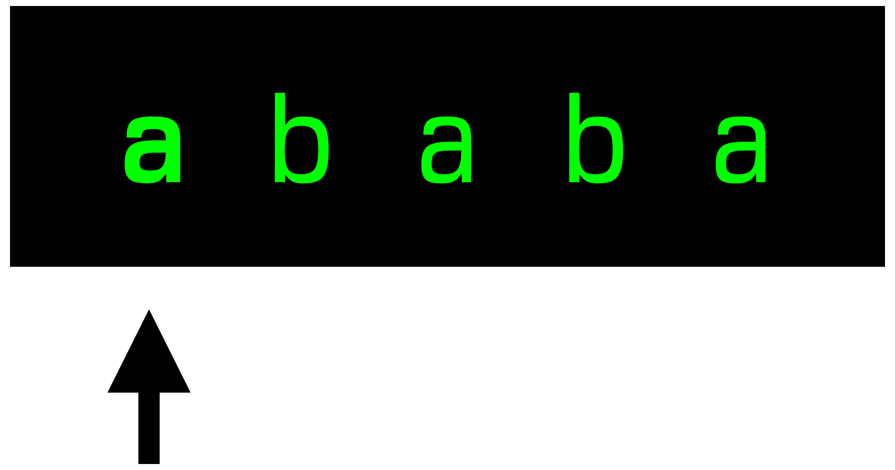


Massive parallelism

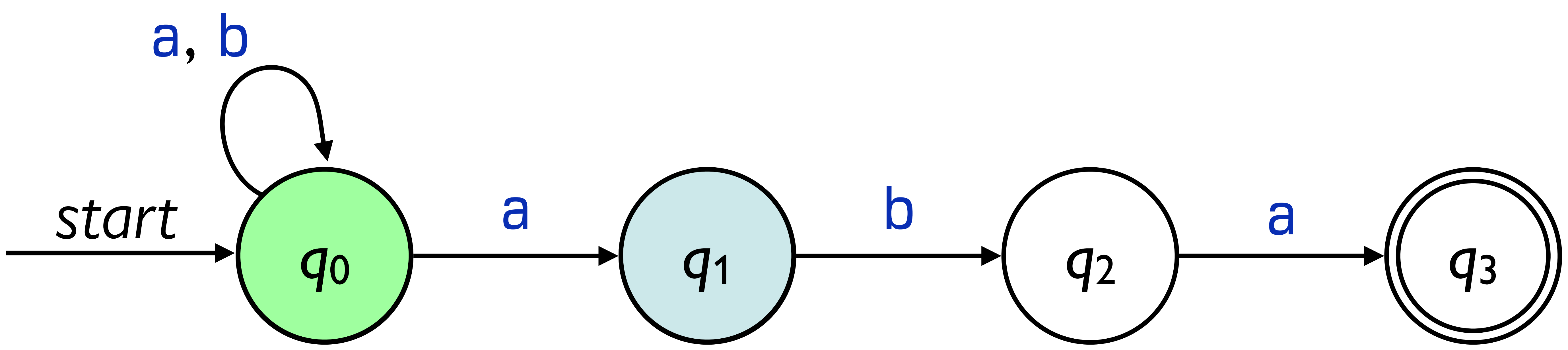


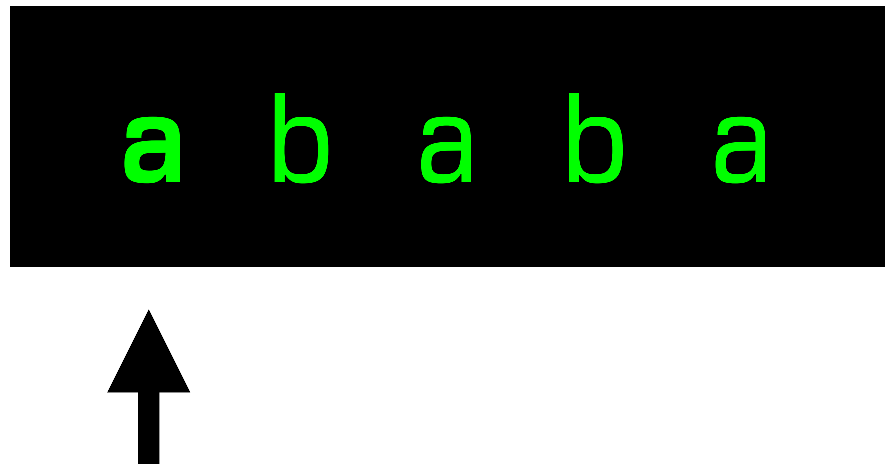
Massive parallelism



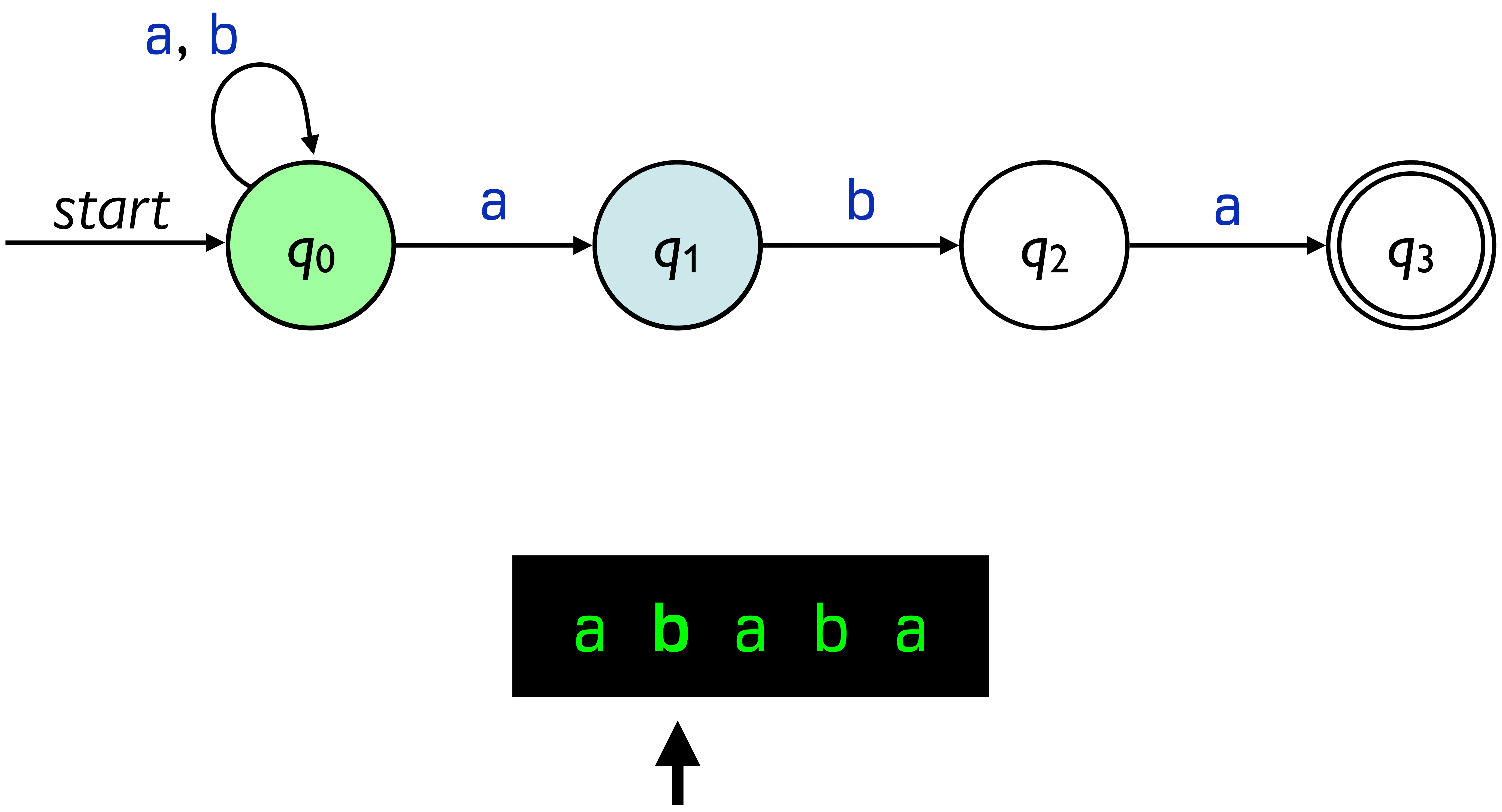


Massive parallelism

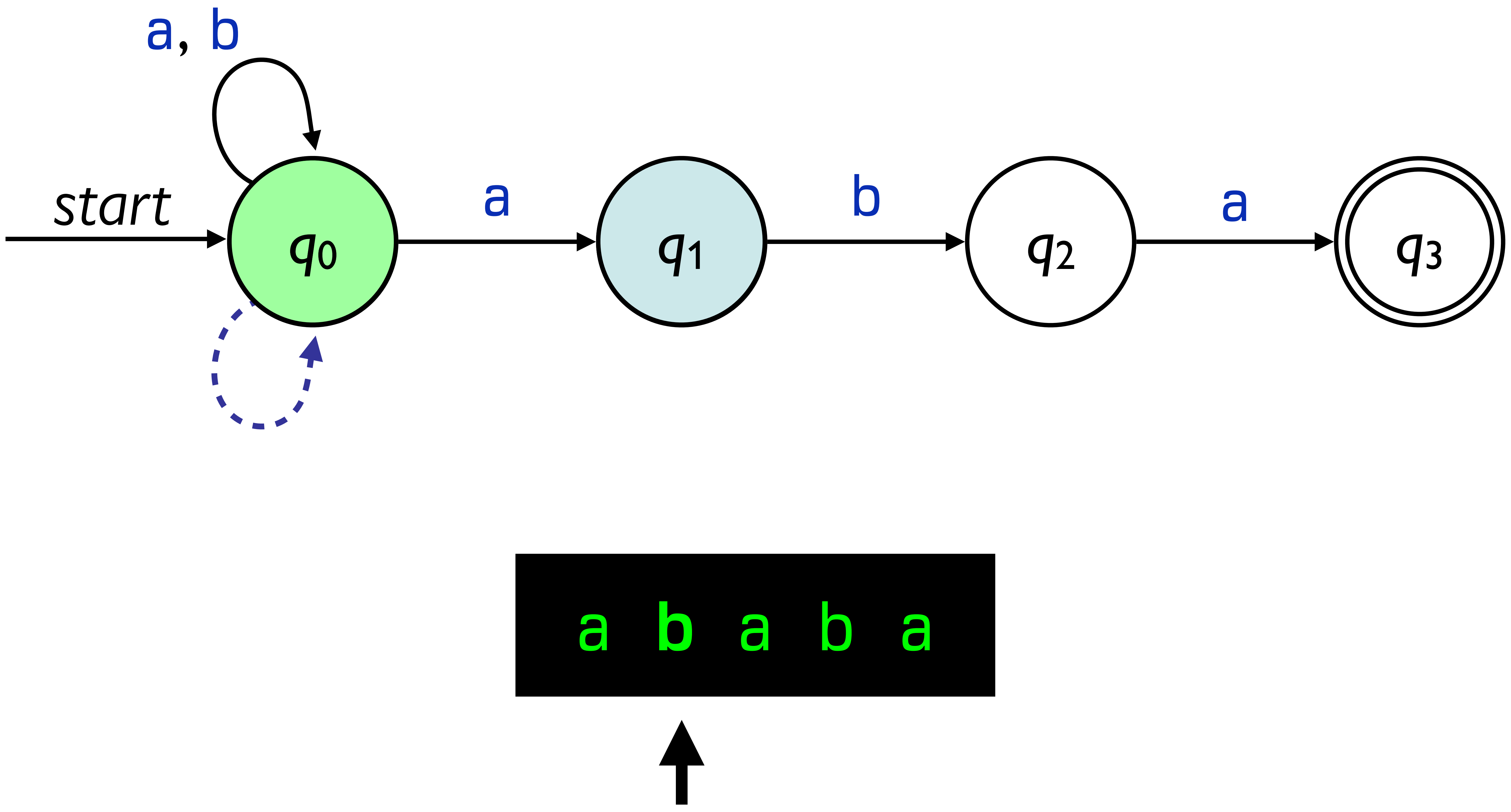




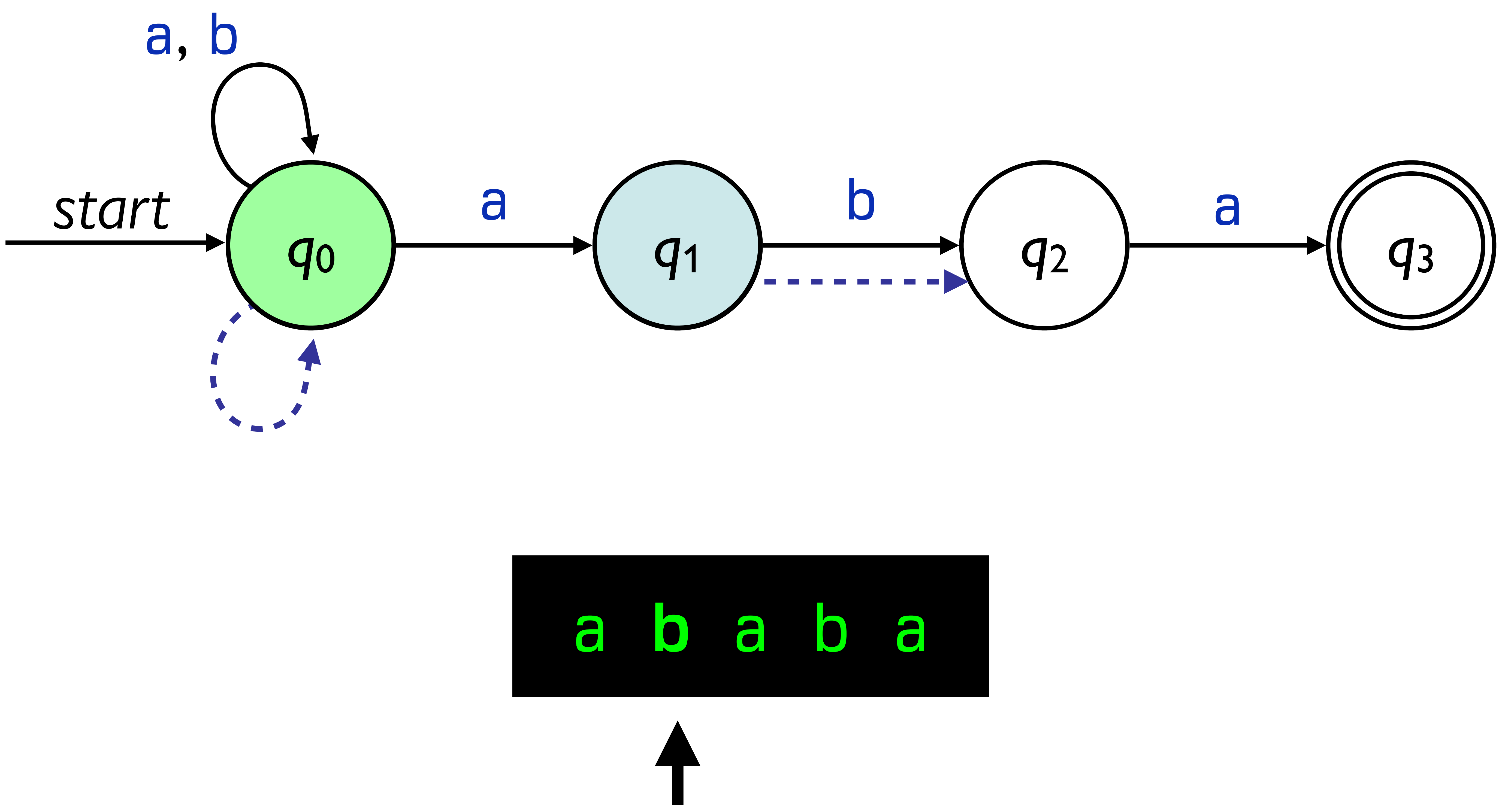
Massive parallelism



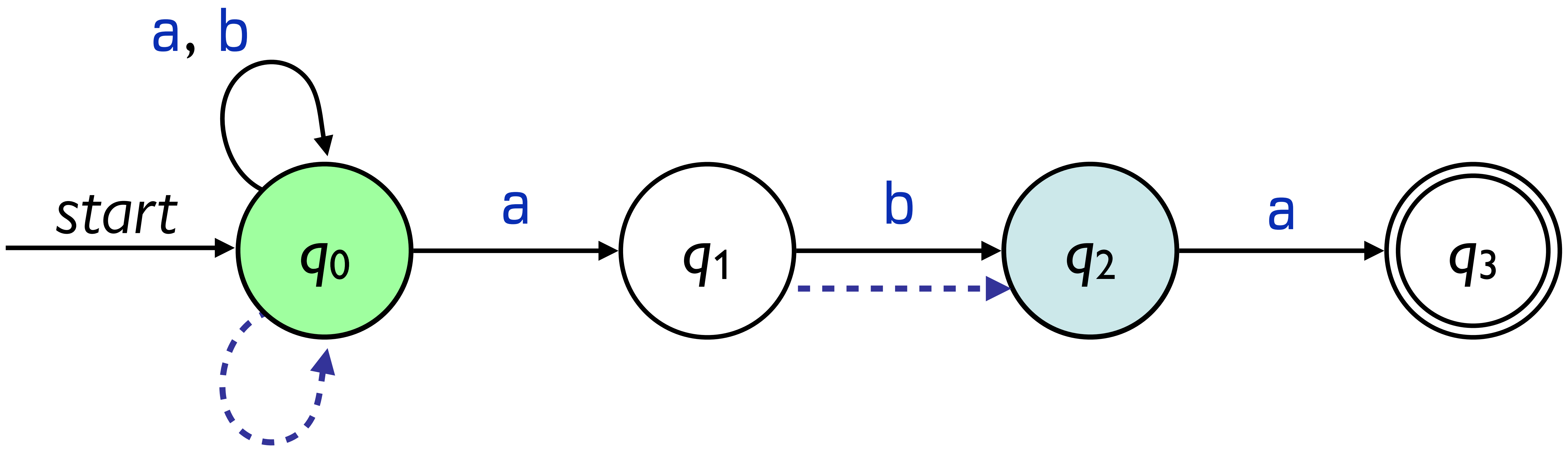
Massive parallelism

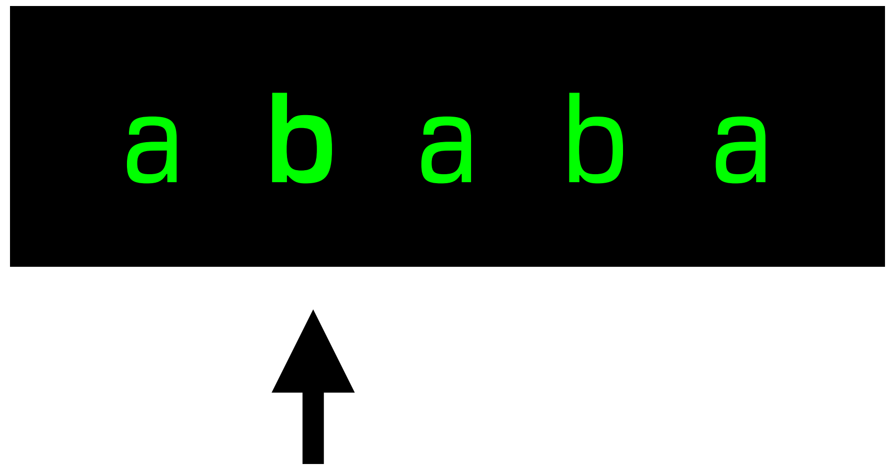


Massive parallelism

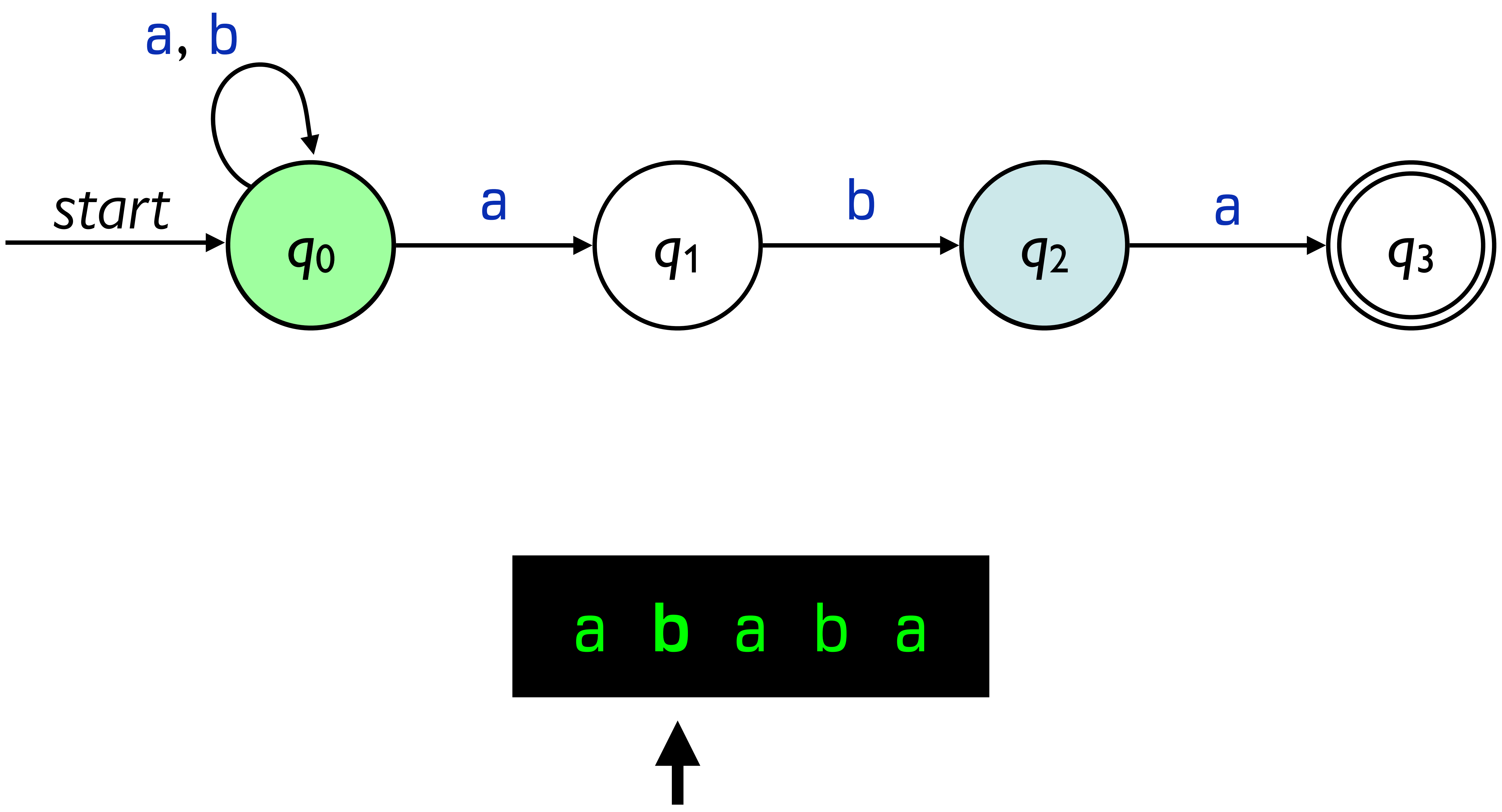


Massive parallelism

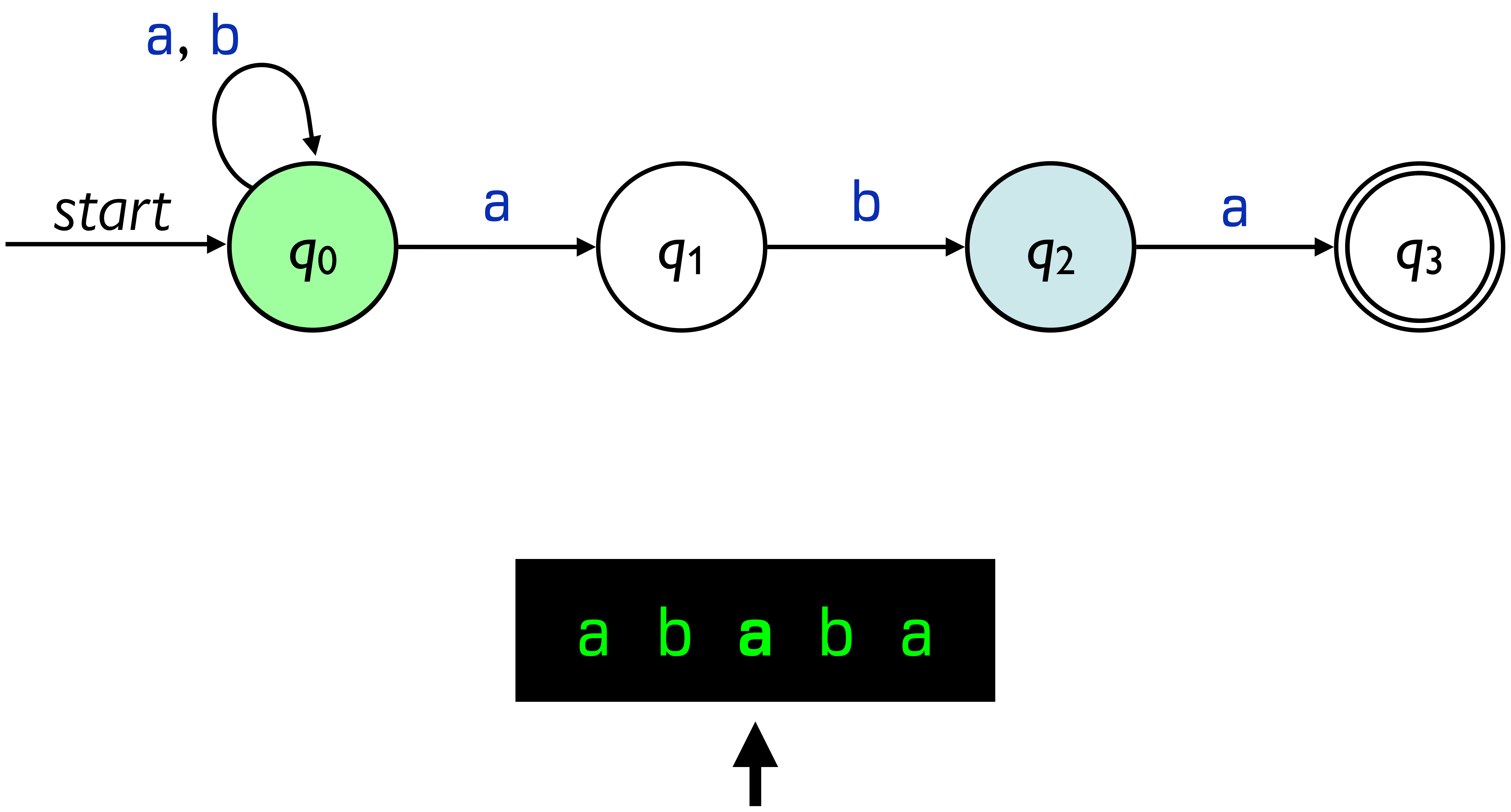




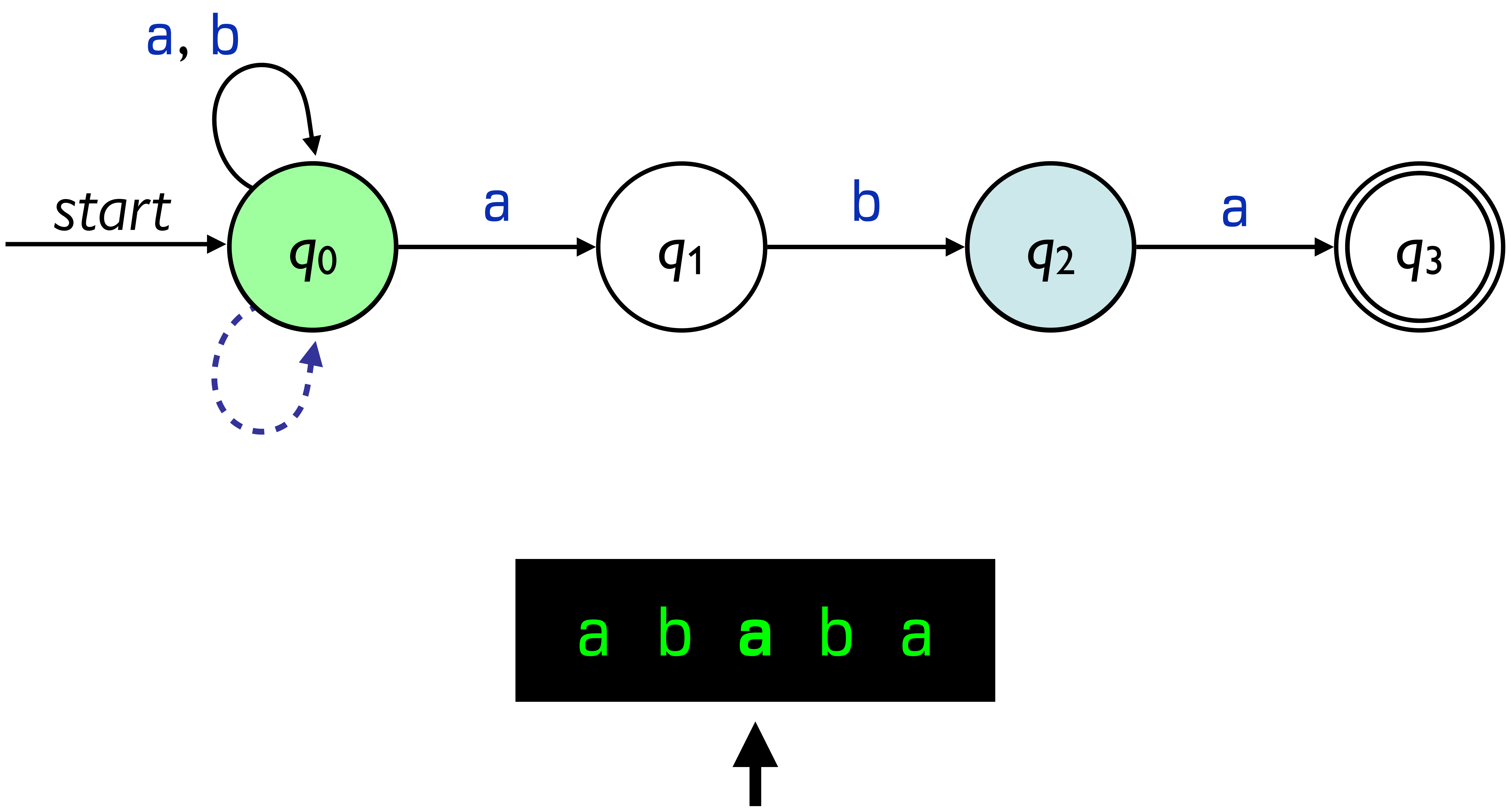
Massive parallelism



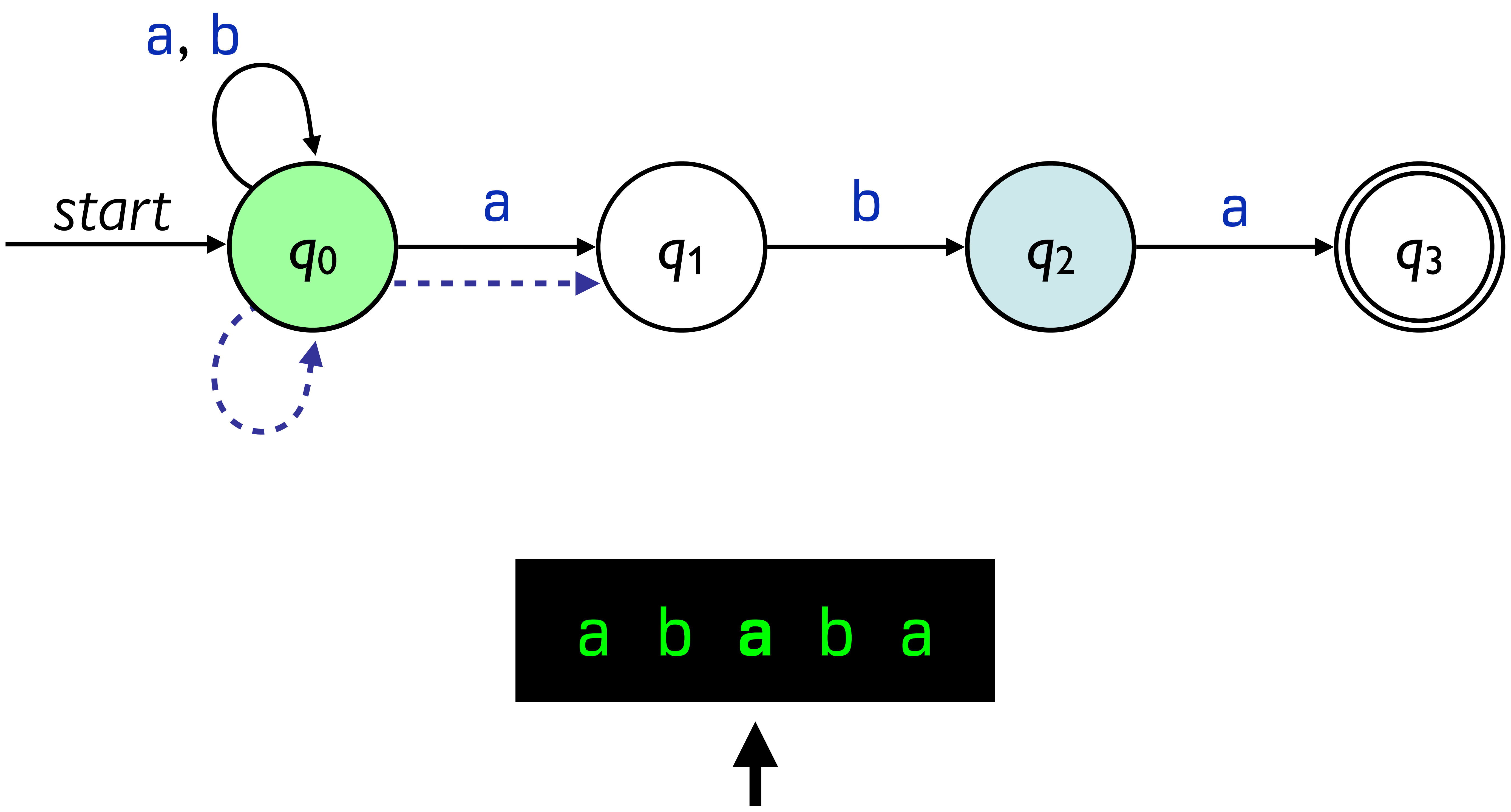
Massive parallelism



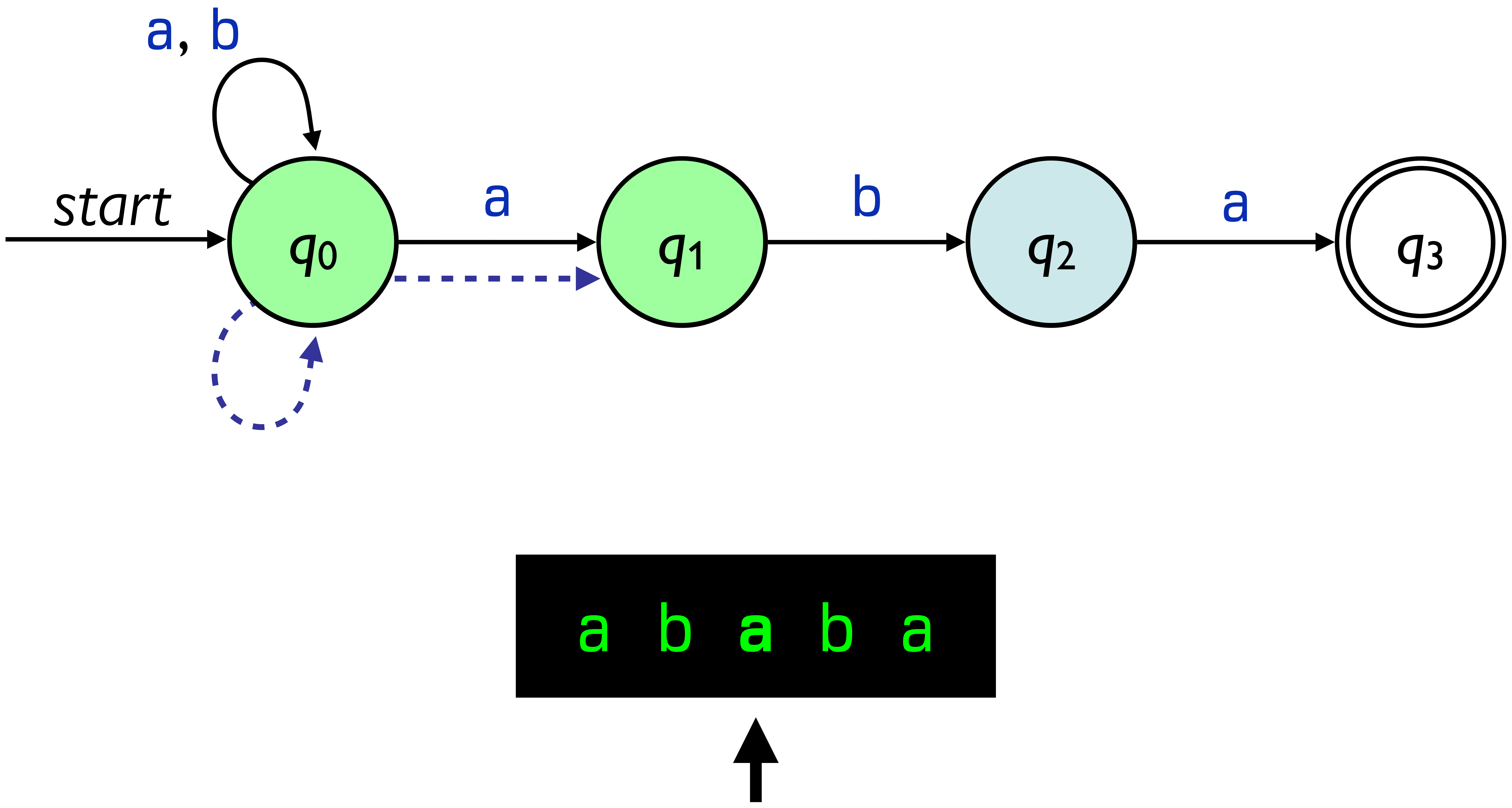
Massive parallelism



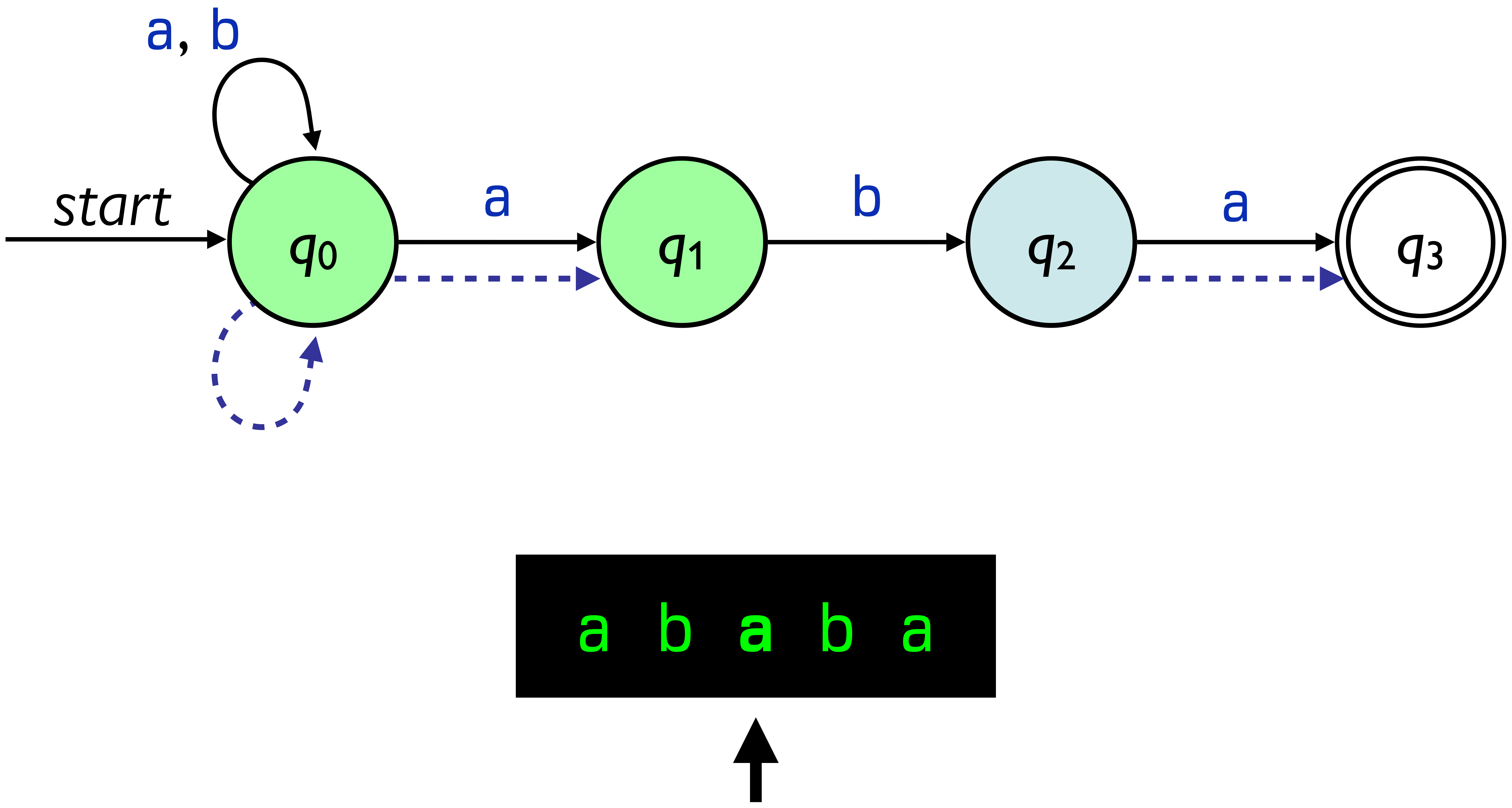
Massive parallelism



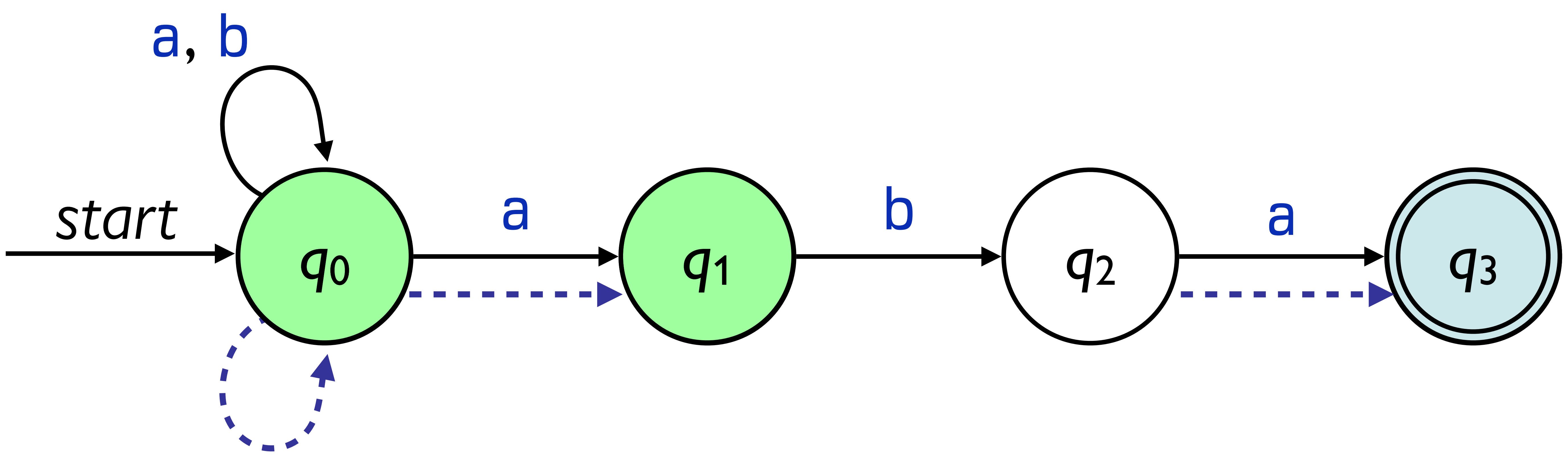
Massive parallelism

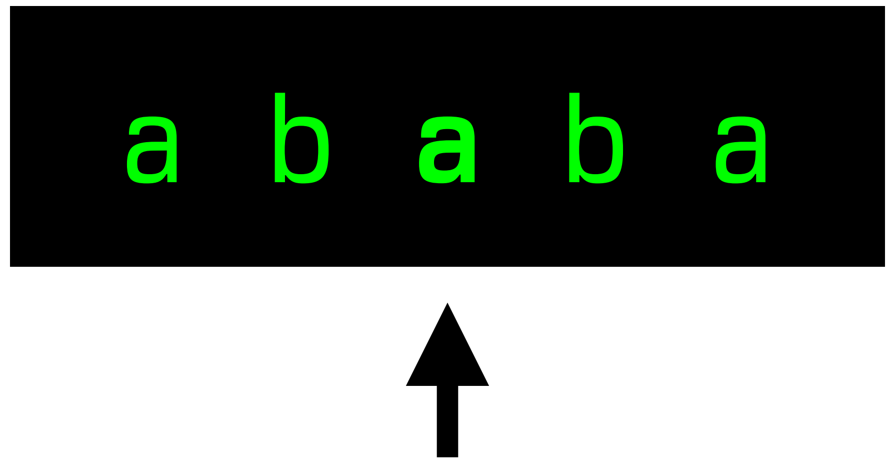


Massive parallelism

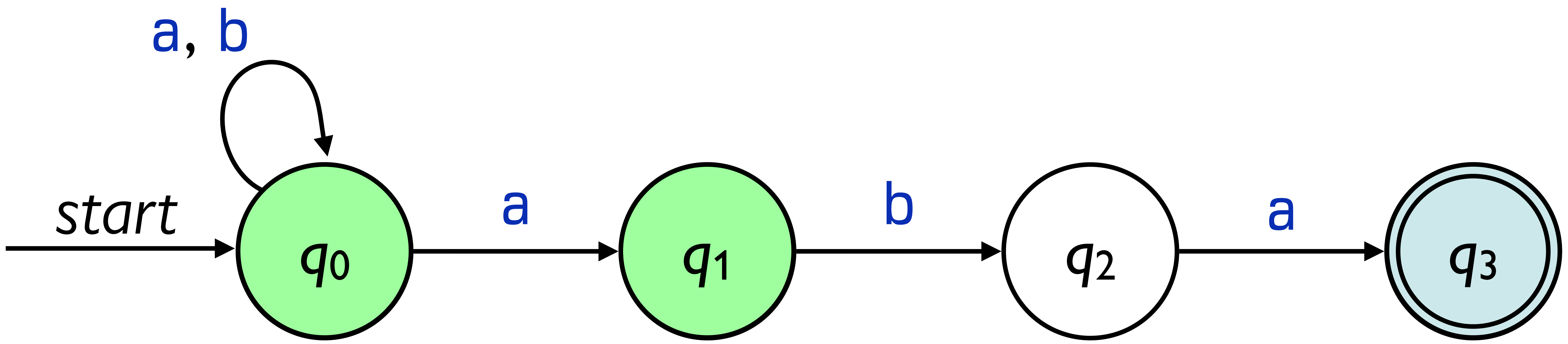


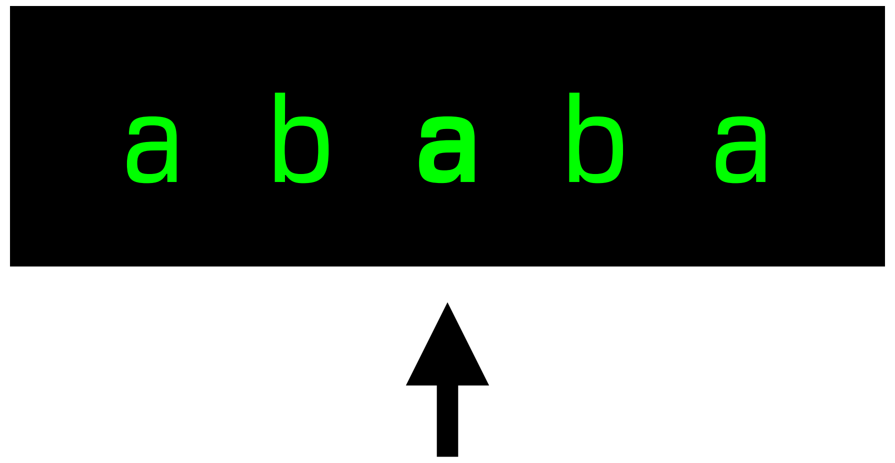
Massive parallelism



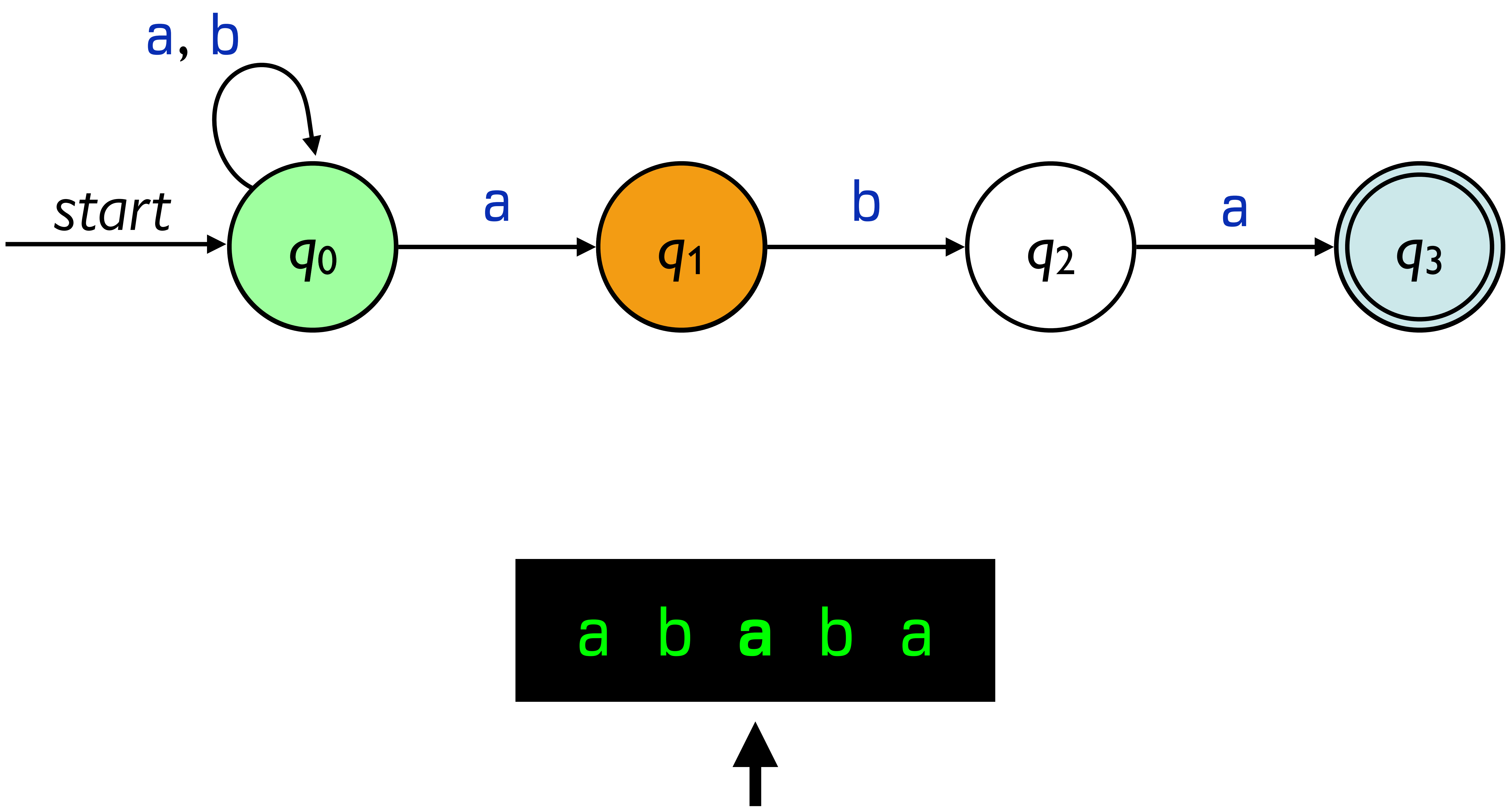


Massive parallelism

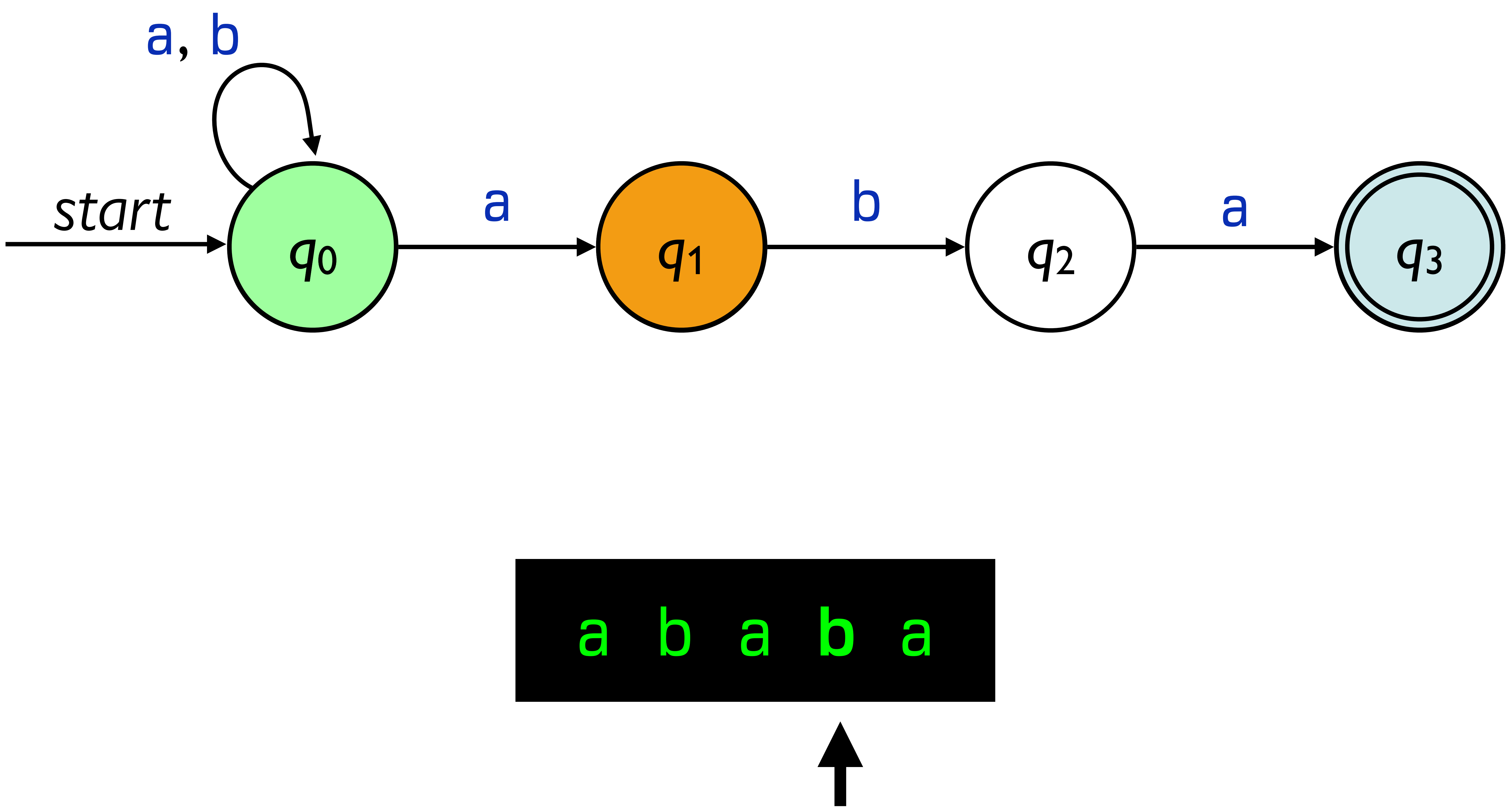




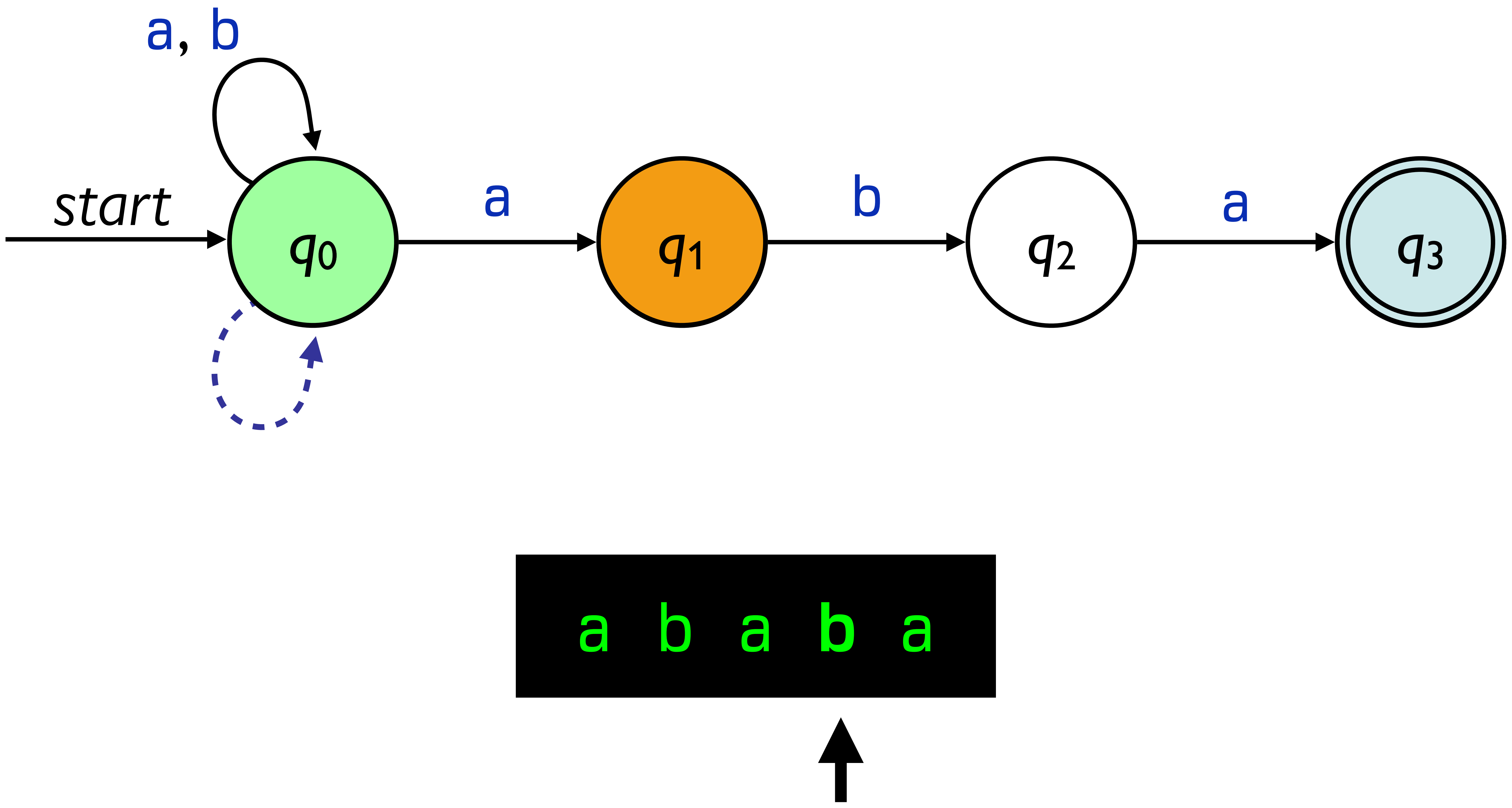
Massive parallelism



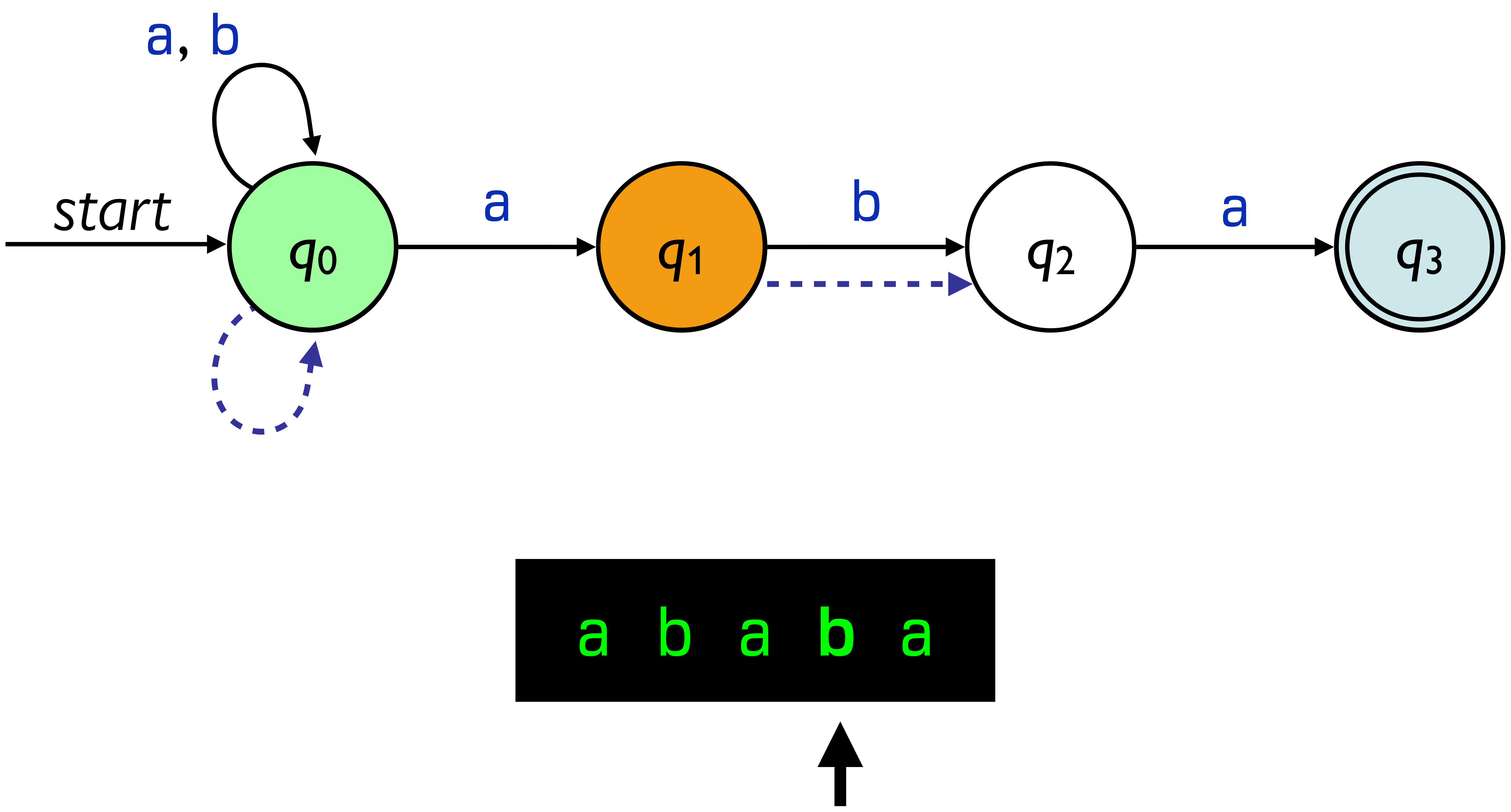
Massive parallelism



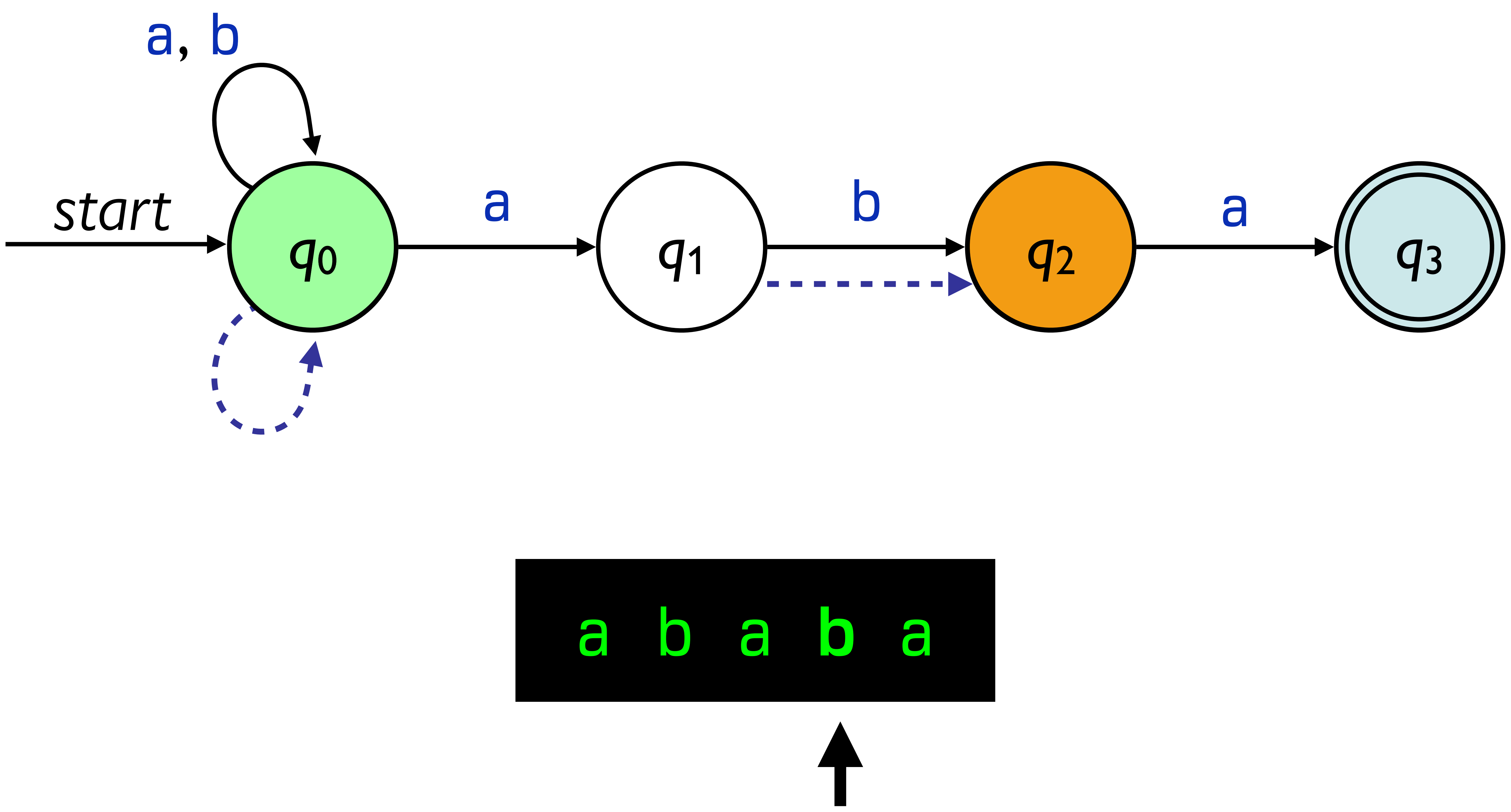
Massive parallelism



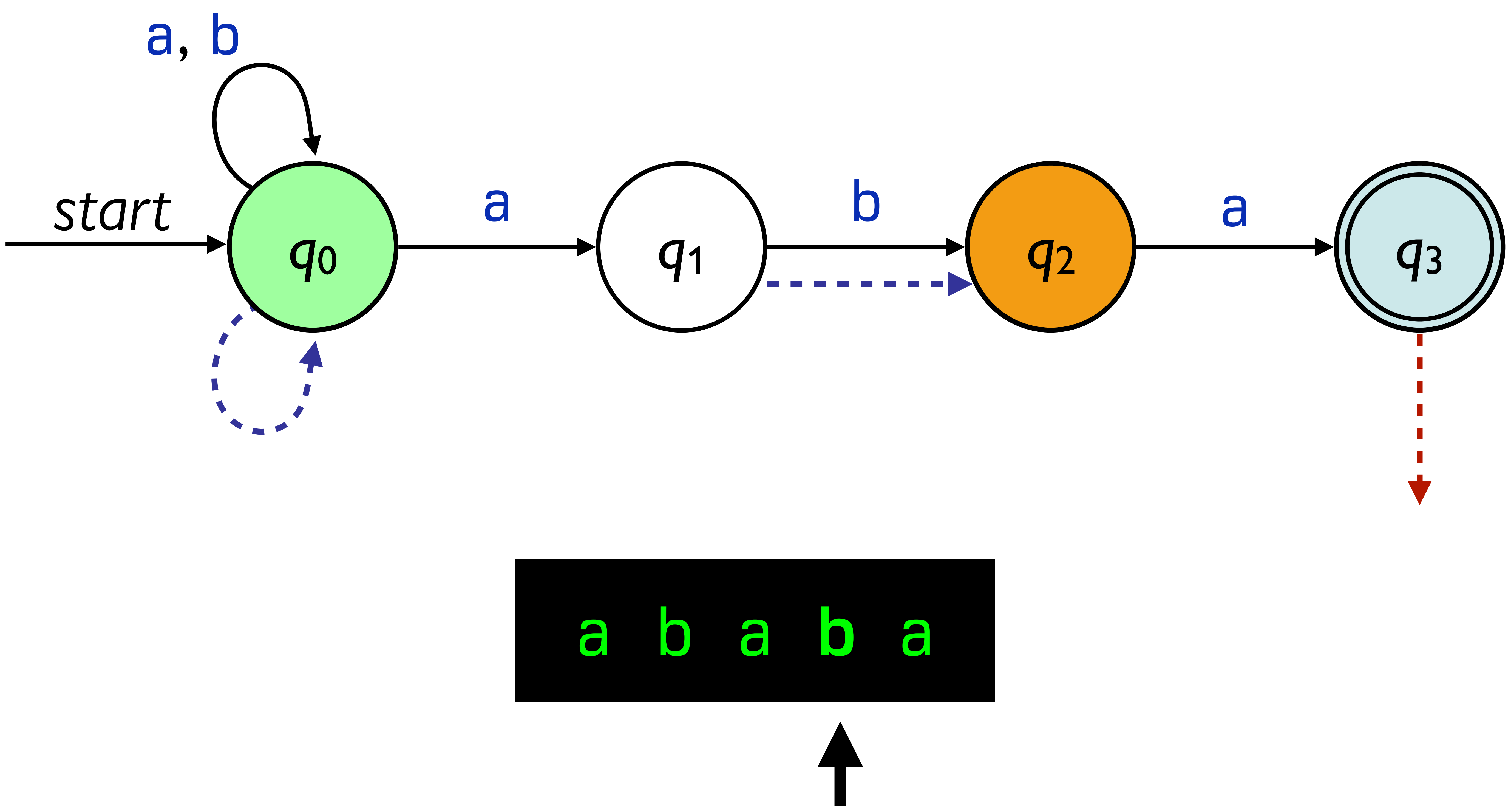
Massive parallelism



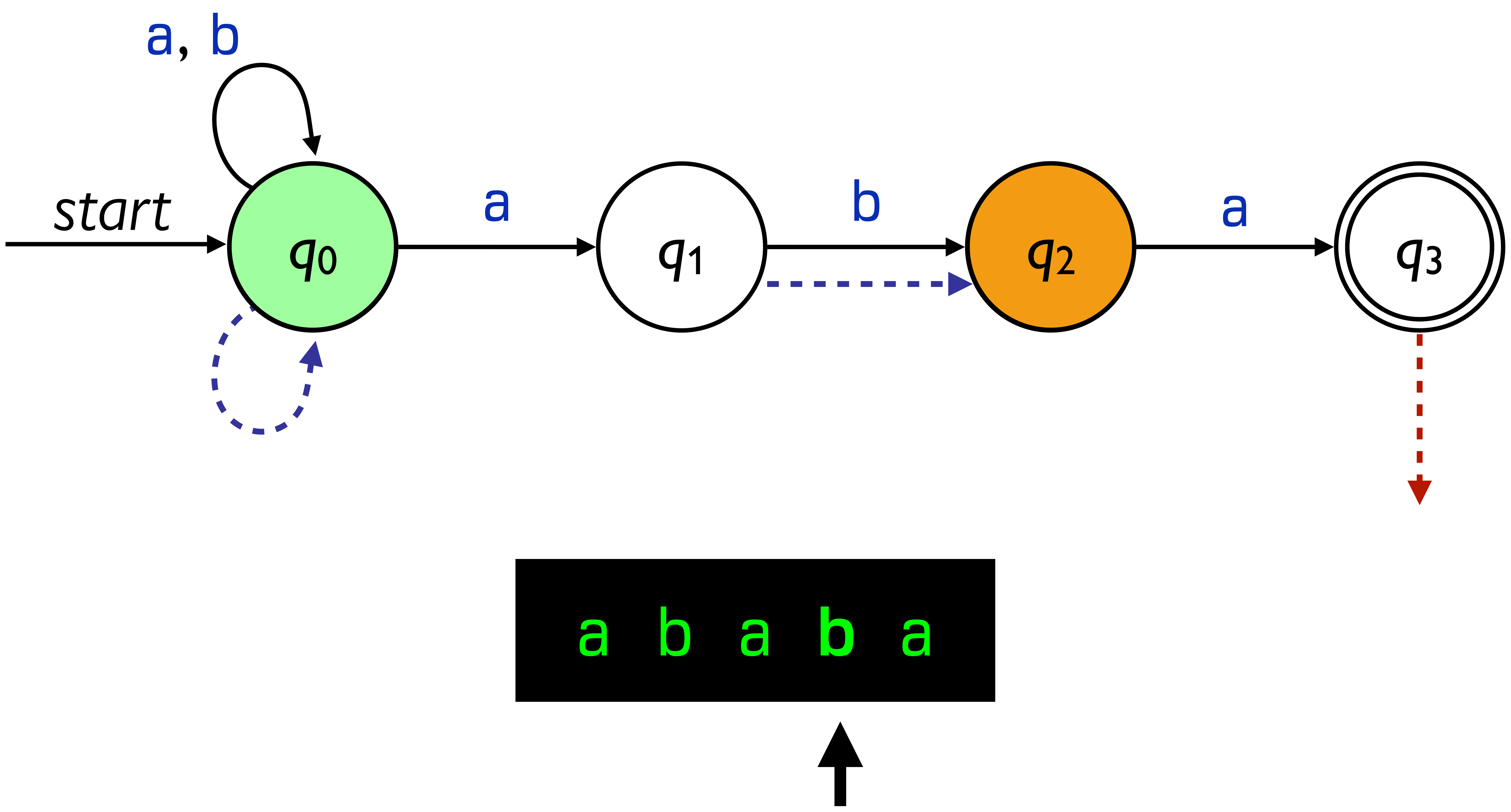
Massive parallelism



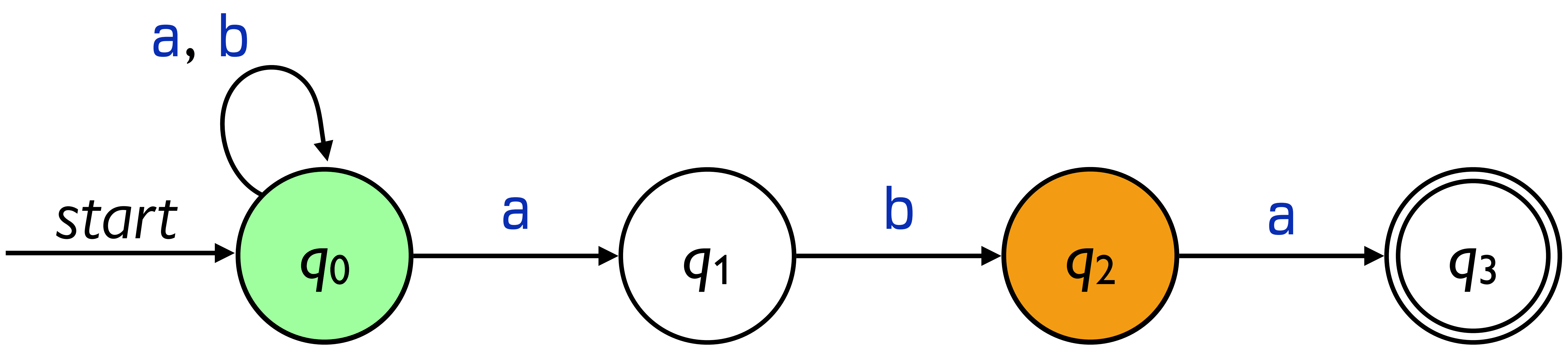
Massive parallelism

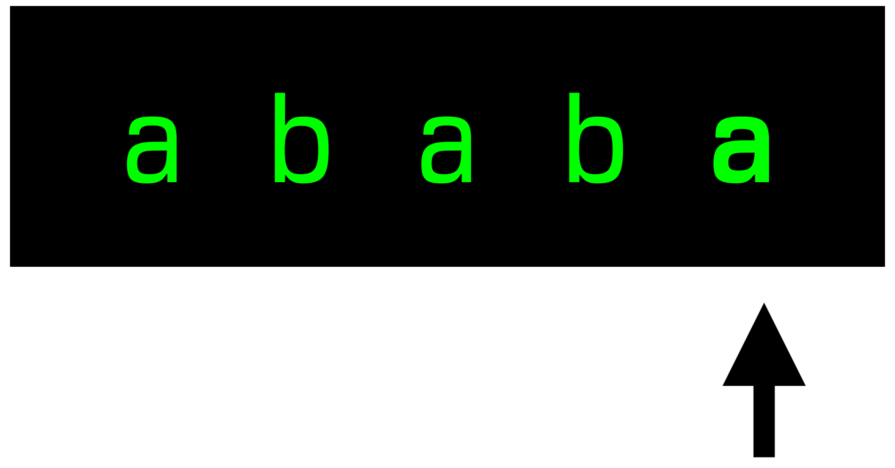


Massive parallelism

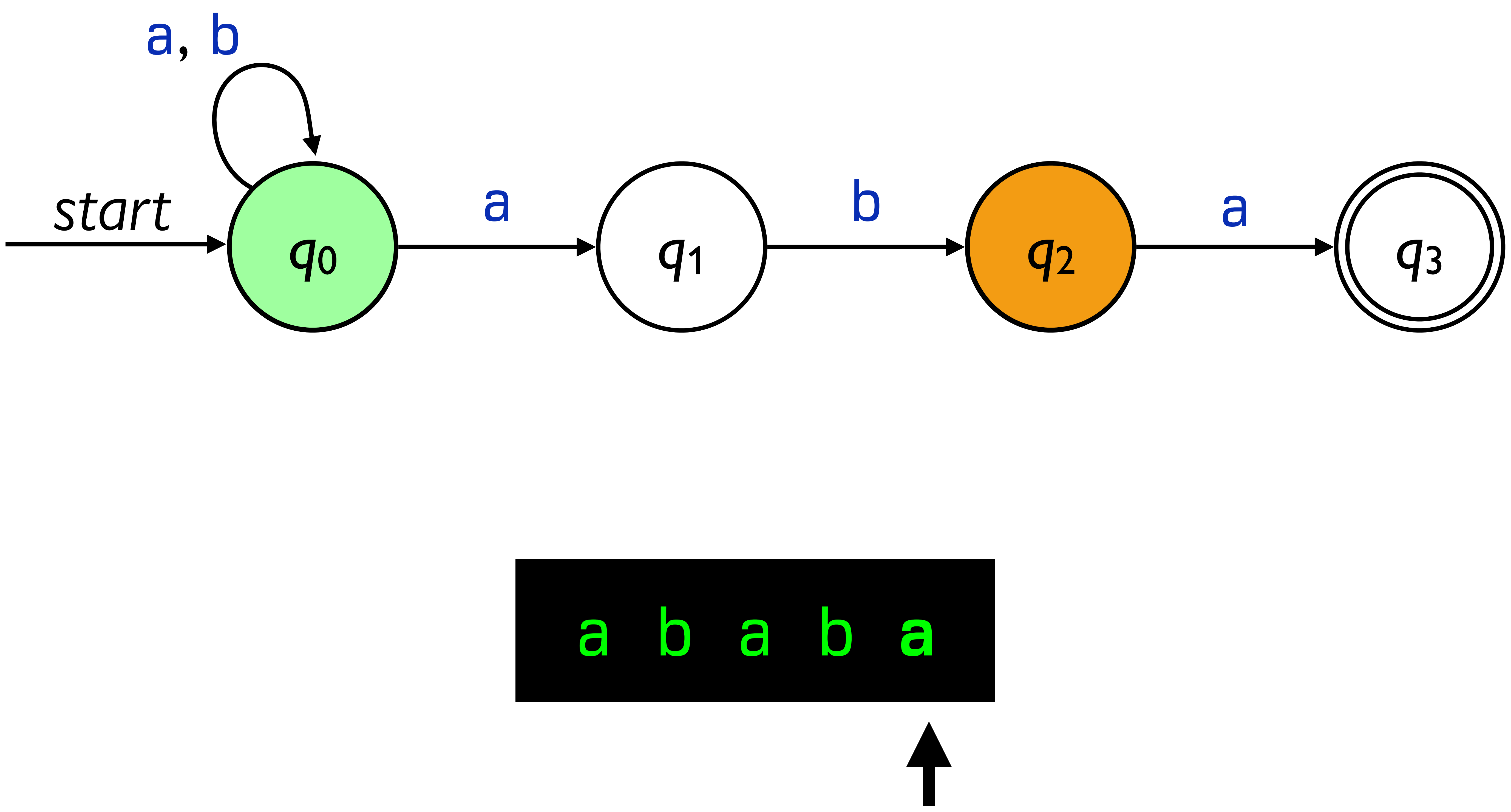


Massive parallelism

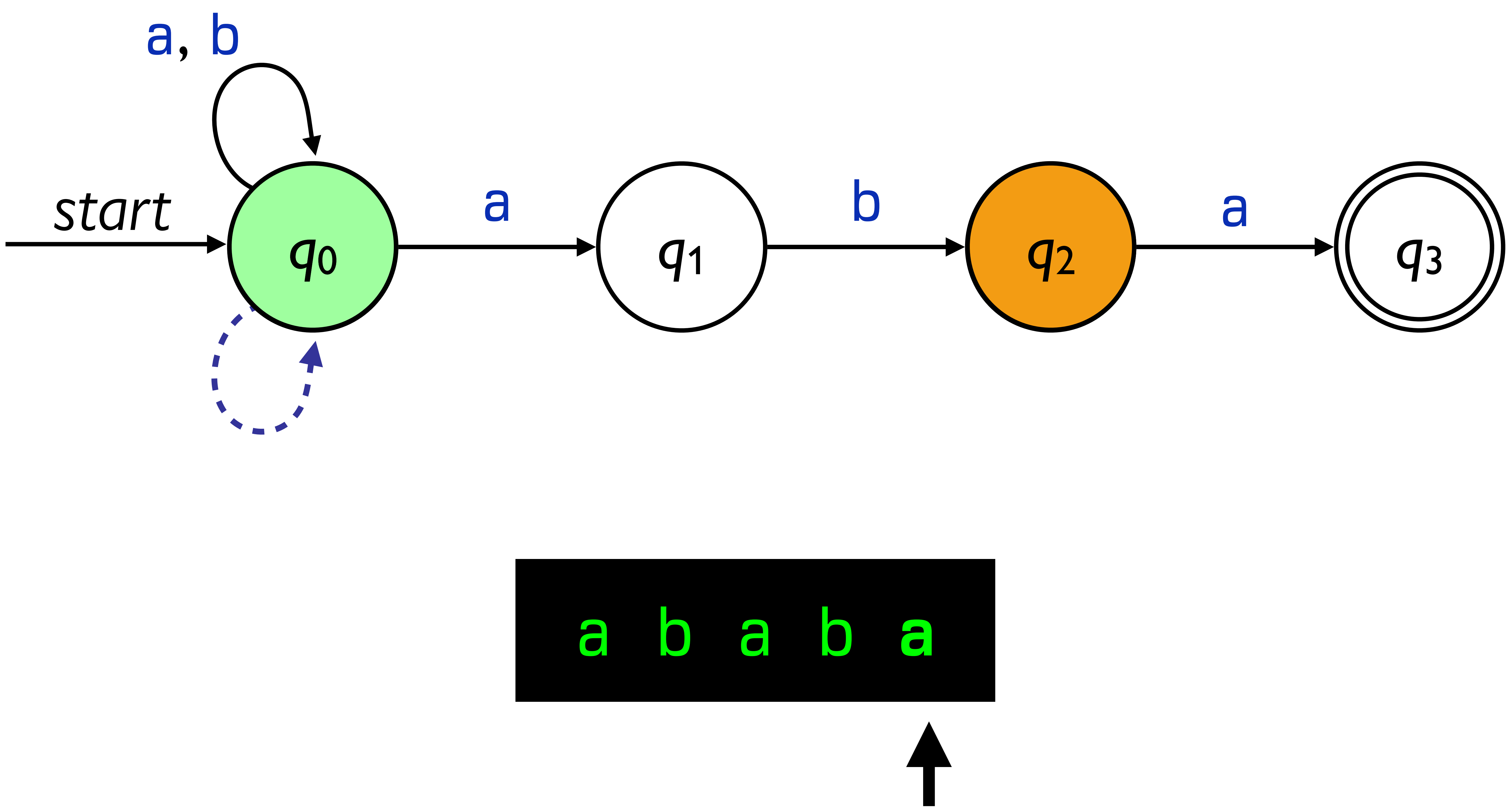




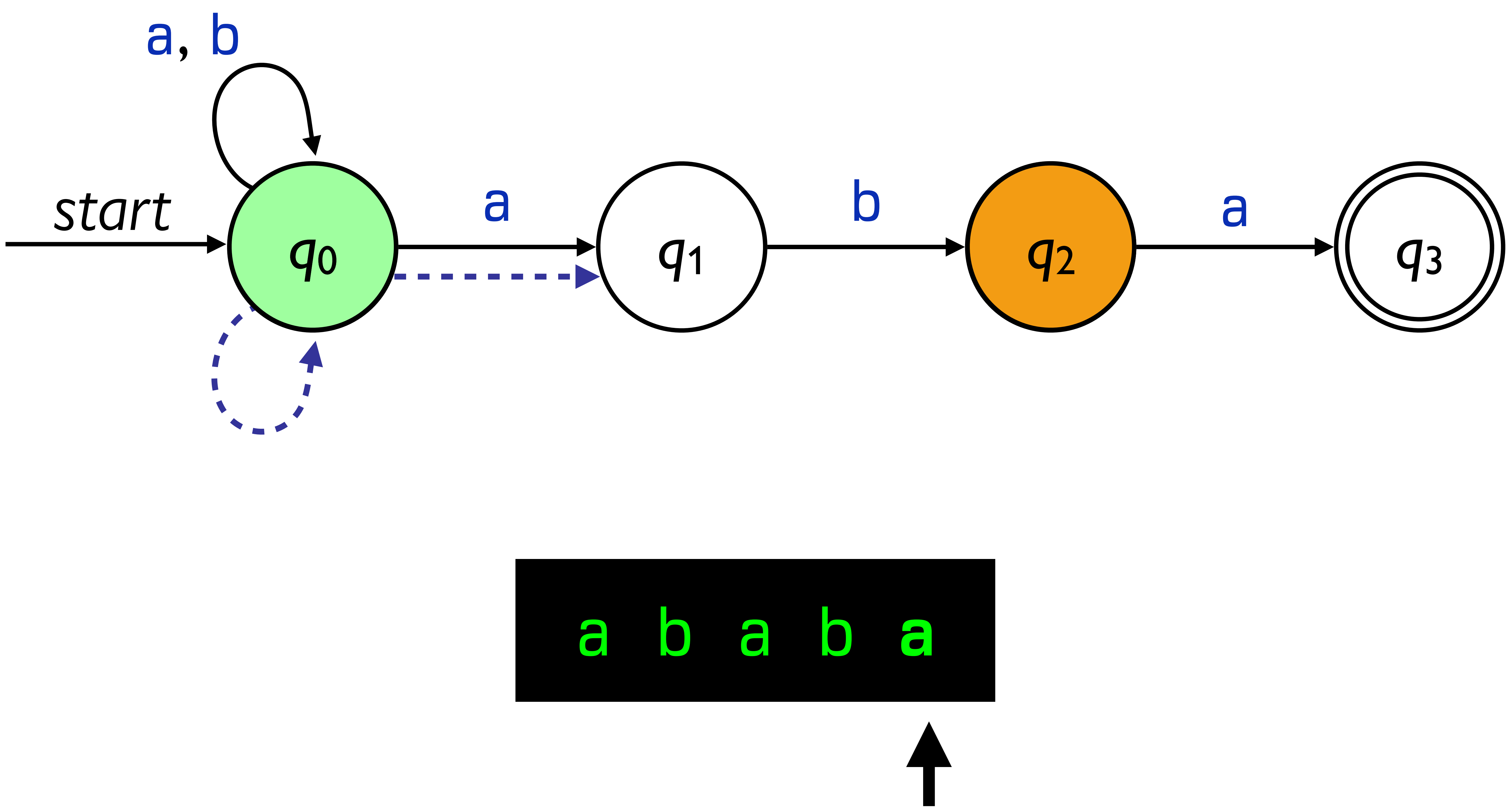
Massive parallelism



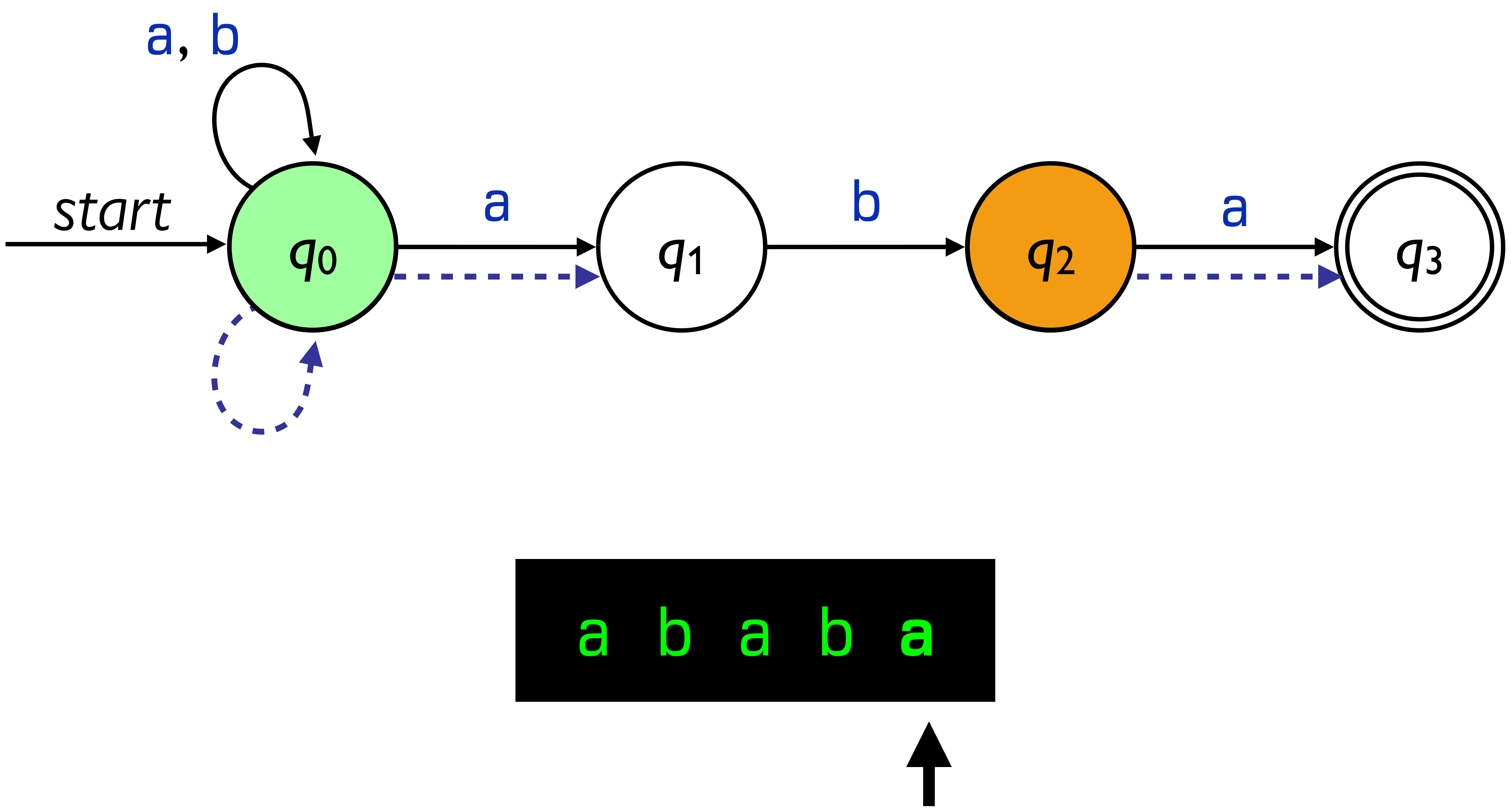
Massive parallelism



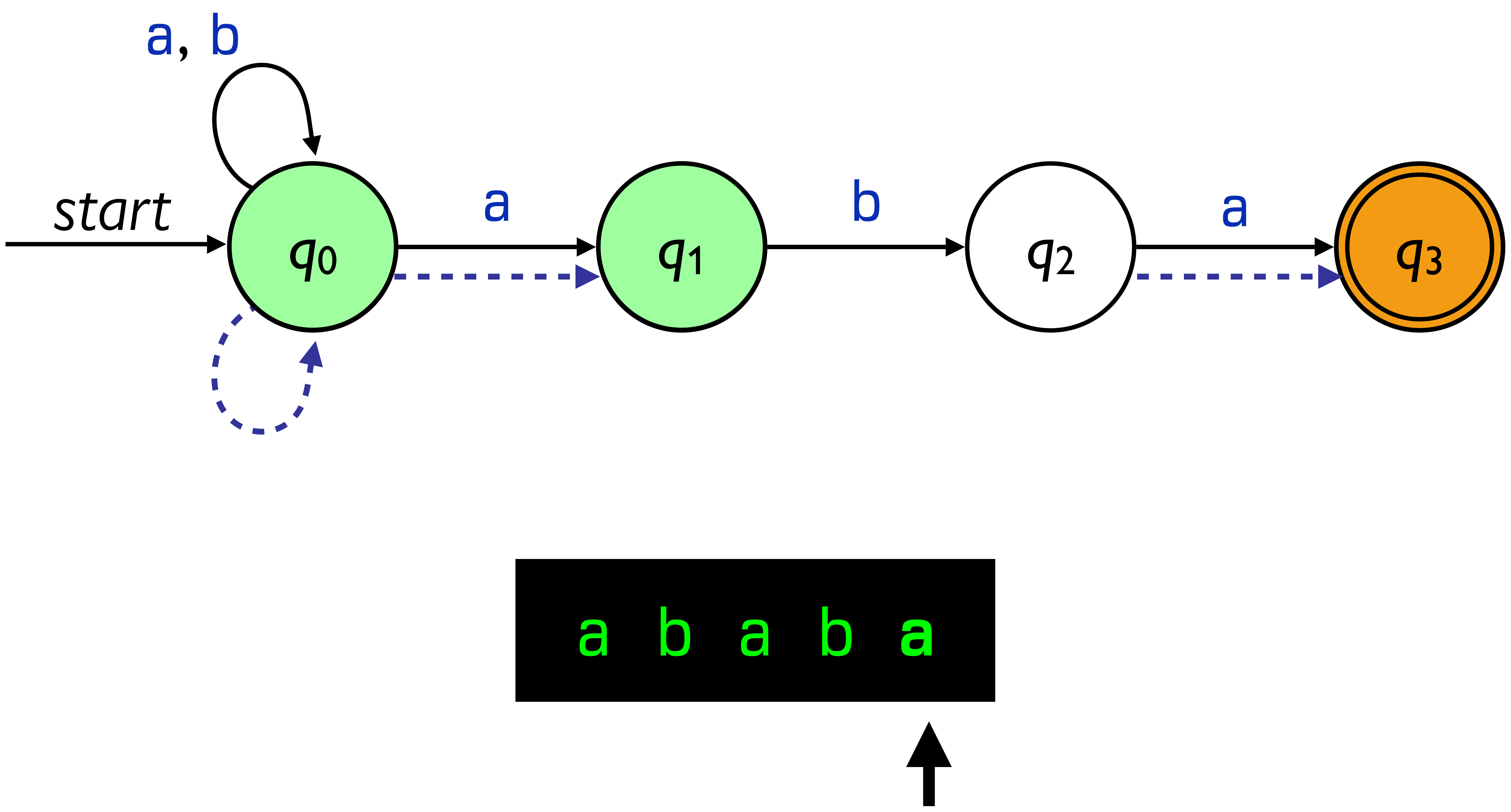
Massive parallelism



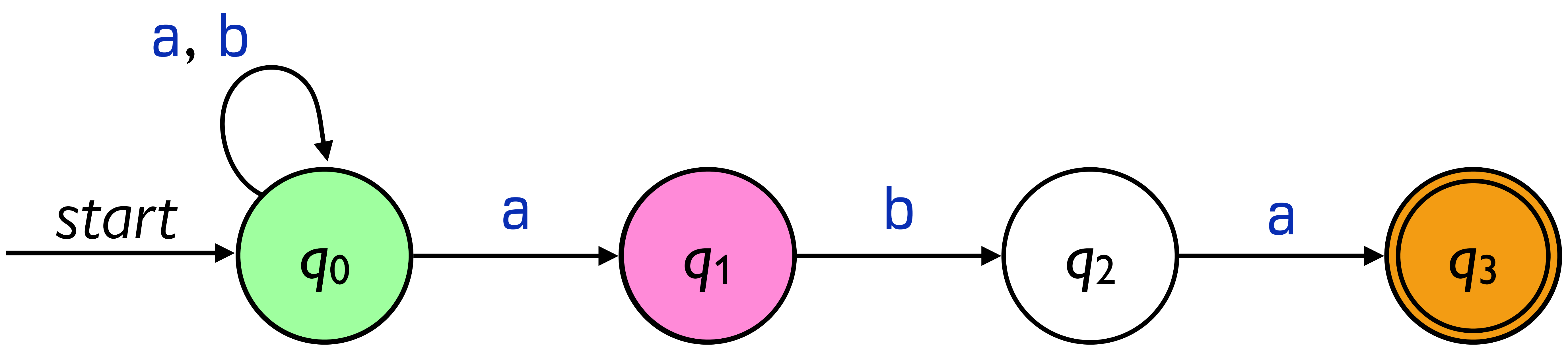
Massive parallelism



Massive parallelism

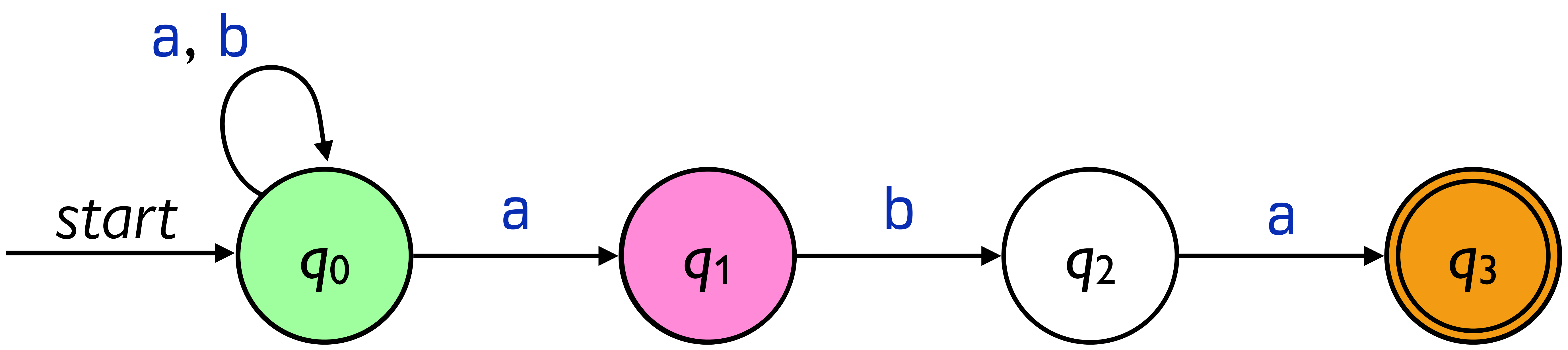


Massive parallelism



a b a b a

Massive parallelism

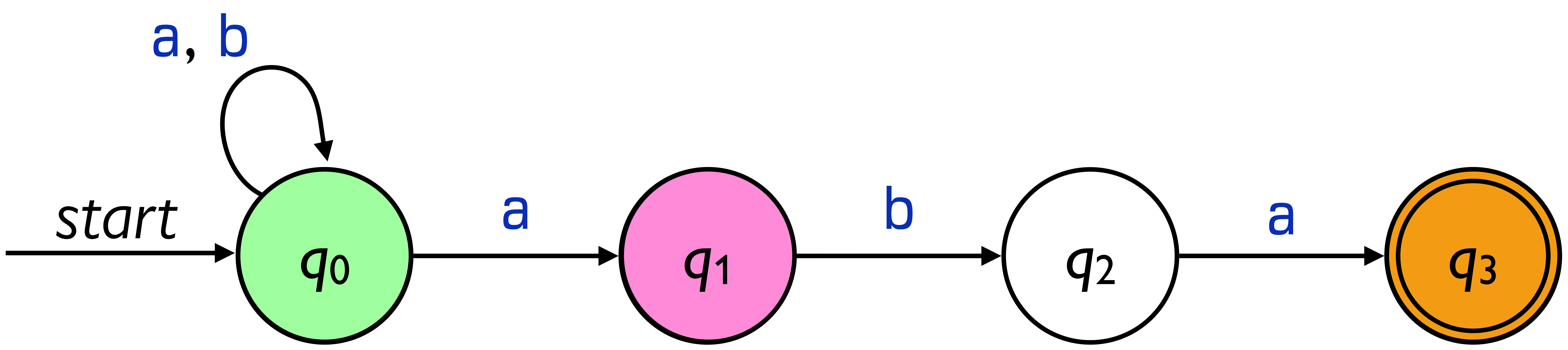


One of the states we're in is an accept state, so there is a path where the NFA accepts the input string.

a b a b a

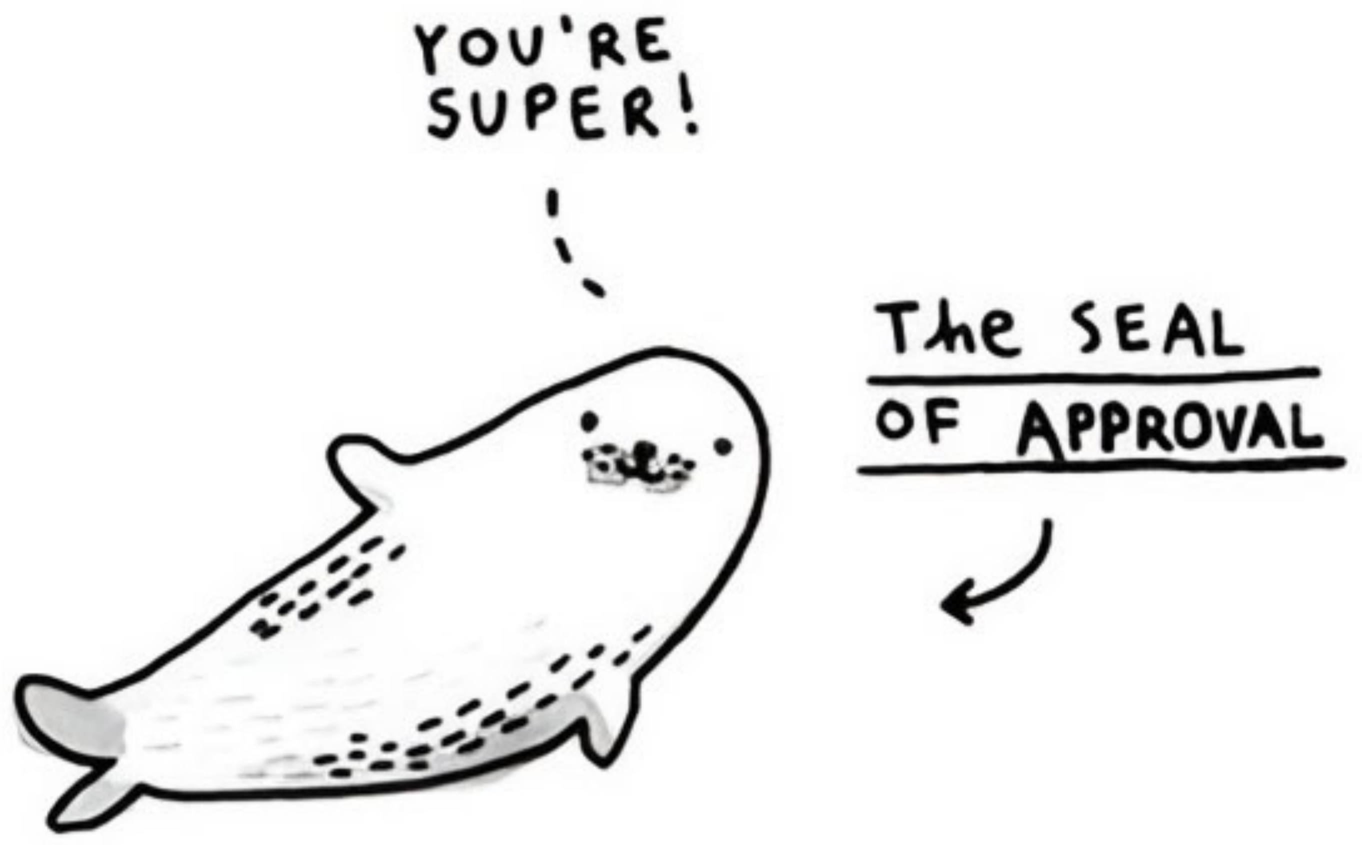
Illustration by
Gemma Correll

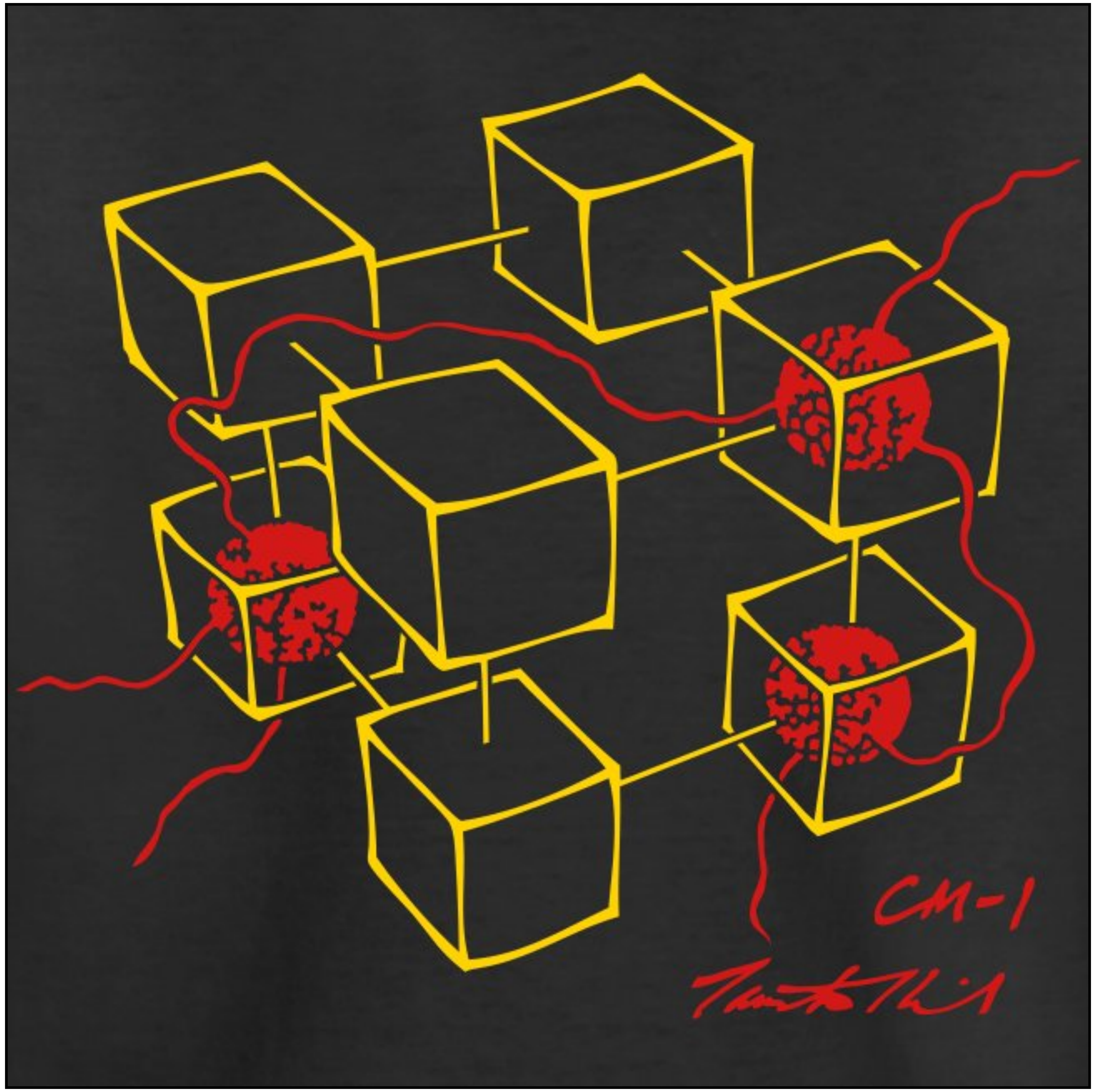
Massive parallelism



One of the states we're in is an accept state, so there is a path where the NFA accepts the input string.

a b a b a





Connection machine CM-1 schematic art by Tamiko Thiel, 1983

The future was and is massive parallelism.

Massive parallelism

An NFA can also be thought of as a DFA that can be in many states at once.

Each symbol read causes a transition on every active state into each potential state that could be visited.

Nondeterministic machines can be thought of as machines that can try any number of options in parallel.

Two roads diverged in a wood, and I –

both of them, at the same time, like a boss

I took ~~the one less traveled by,~~

And that has made all the difference.

Robert Frost

Perfect guessing is a helpful way to think about how to design a machine to recognize a language.

Massive parallelism is a great way to test machines, and it has nice theoretical implications.

Language of an NFA:

An NFA accepts an input string w if ***any*** path from the start state to an accept state is labeled w.

Designing NFAs

Embrace the nondeterminism.

A good approach is *guess-and-check*:

Is there some information you'd like to have?

Have the machine *nondeterministically guess* that information!

Then have it *deterministically check* that the choice was right, i.e., filter out the bad guesses.

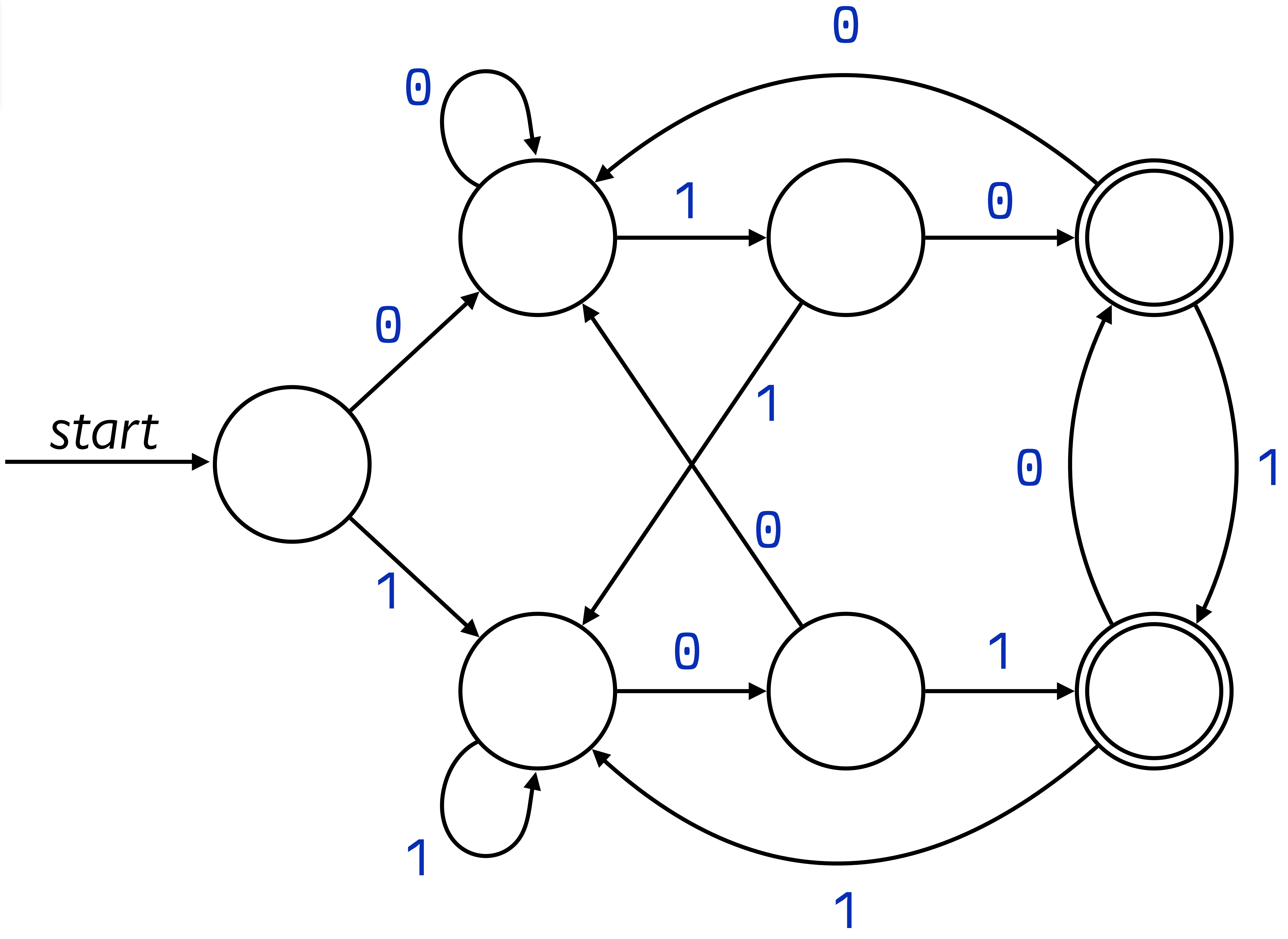
The *guess* phase corresponds to trying lots of different options.

The *check* phase corresponds to filtering out bad guesses or wrong options.

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

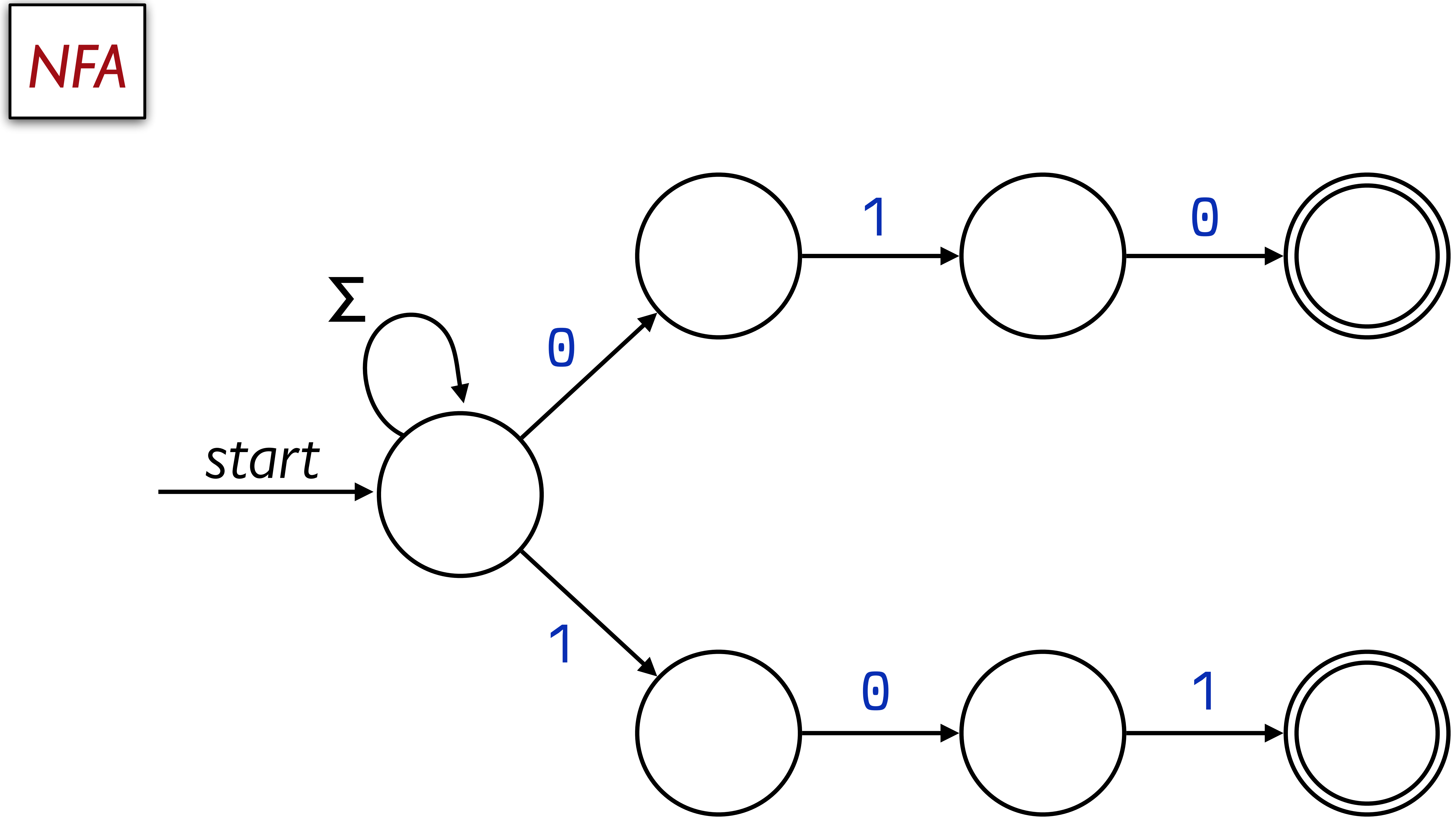
$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$

DFA

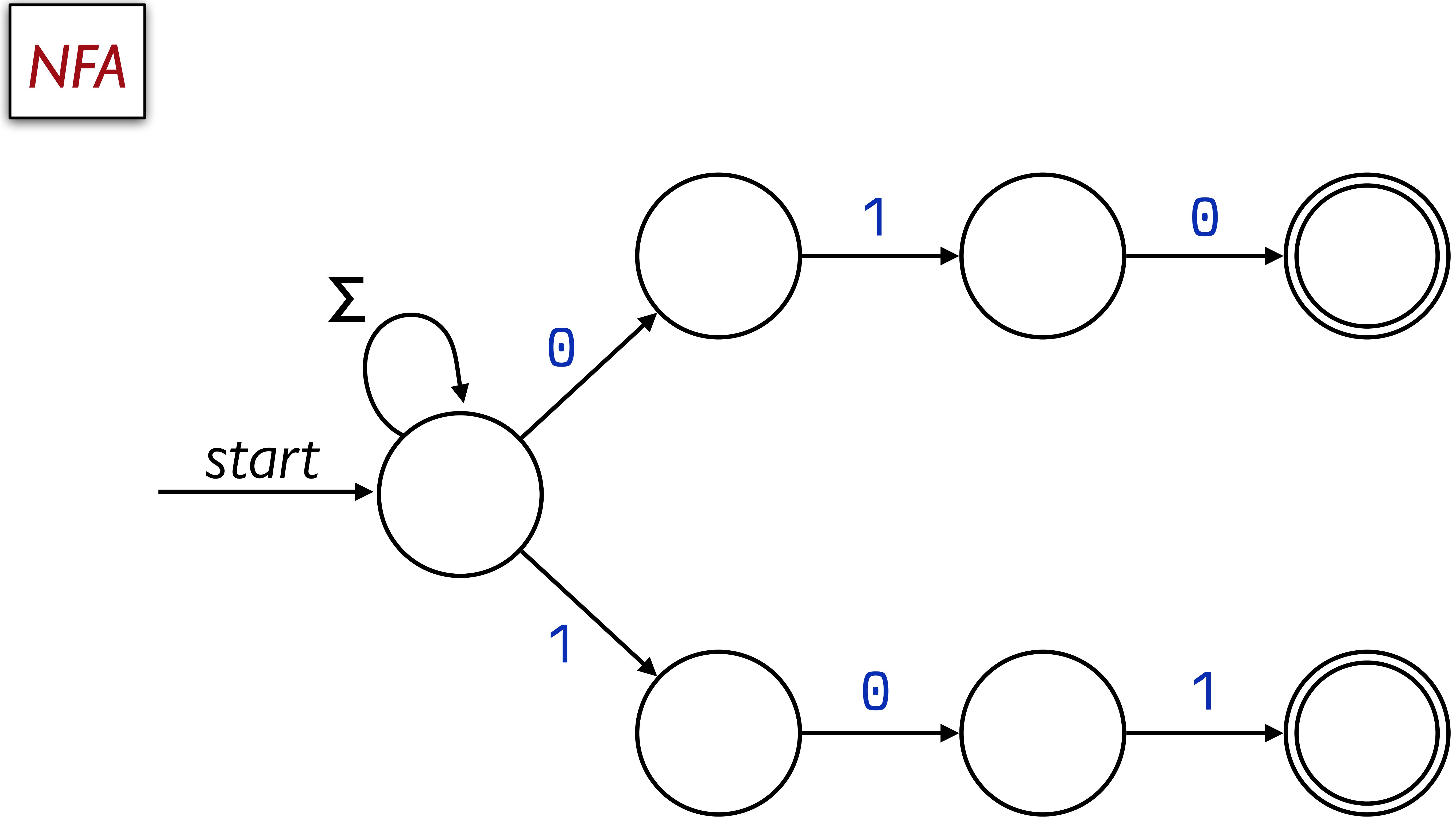


$L = \{w \in \{\texttt{0}, \texttt{1}\}^* \mid w \text{ ends in } \texttt{010} \text{ or } \texttt{101}\}$

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



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

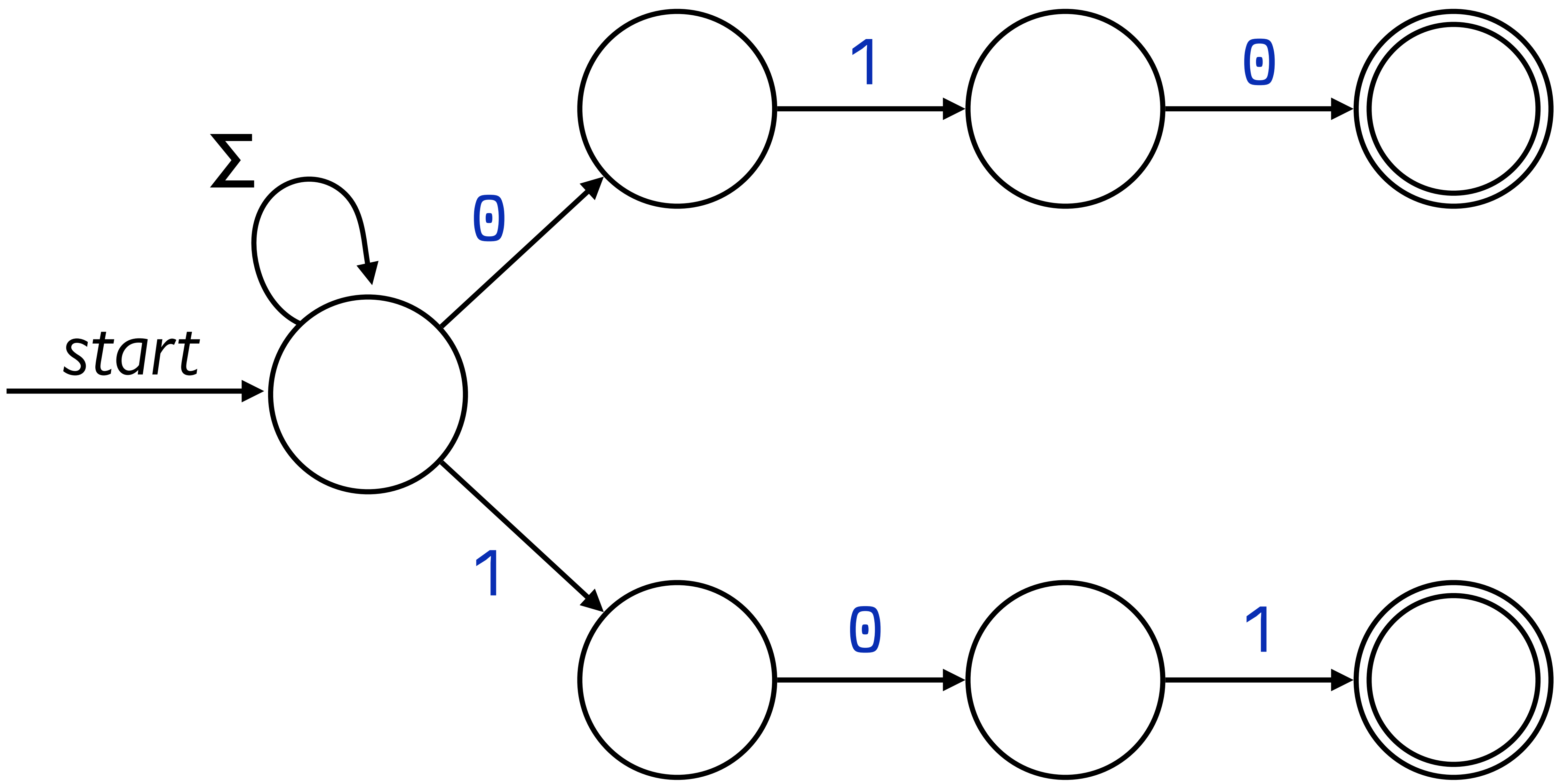


*Nondeterministically **guess** when the end of the string is coming up.*

*Deterministically **check** whether you were correct.*

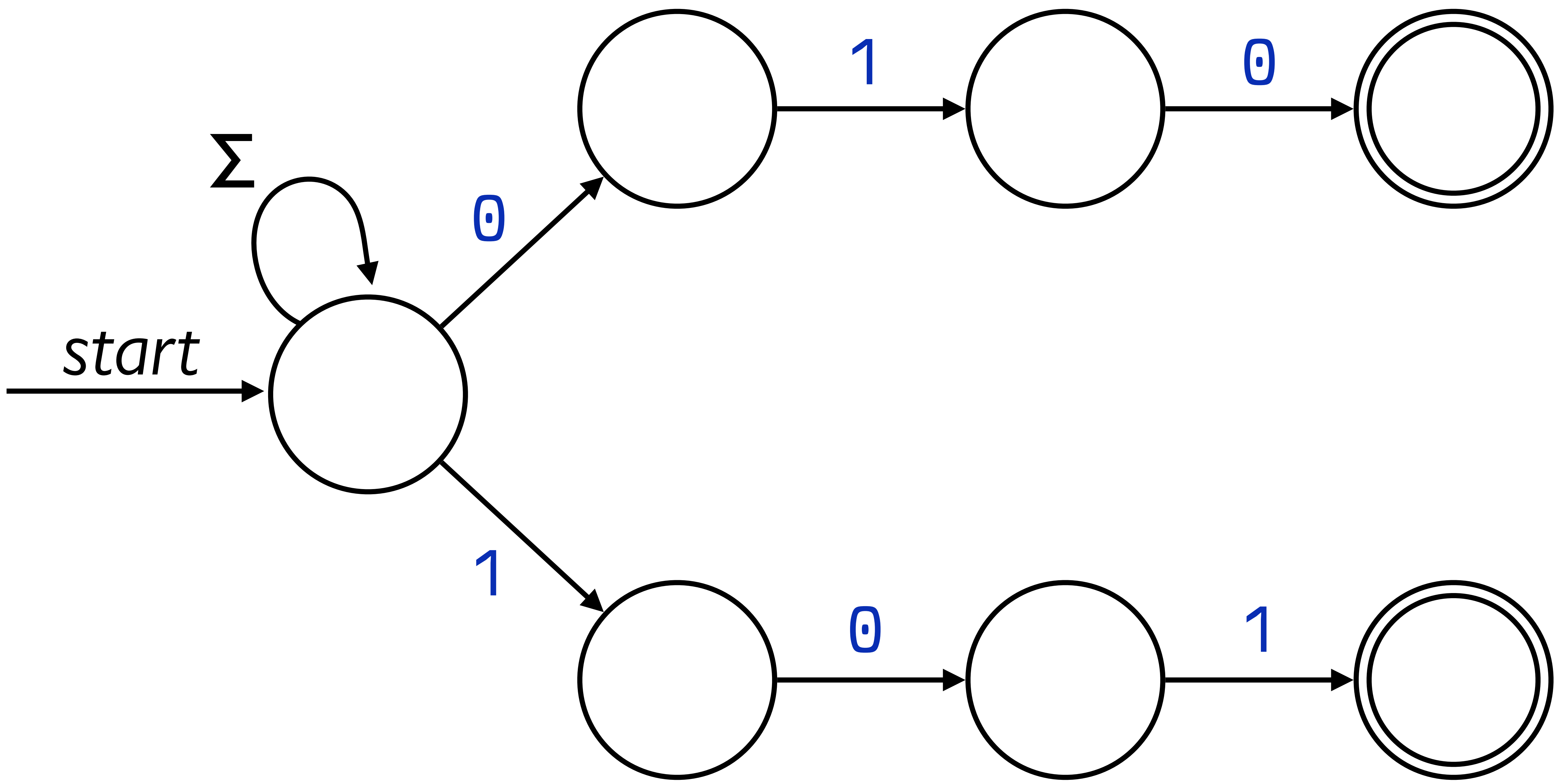
$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$

NFA



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

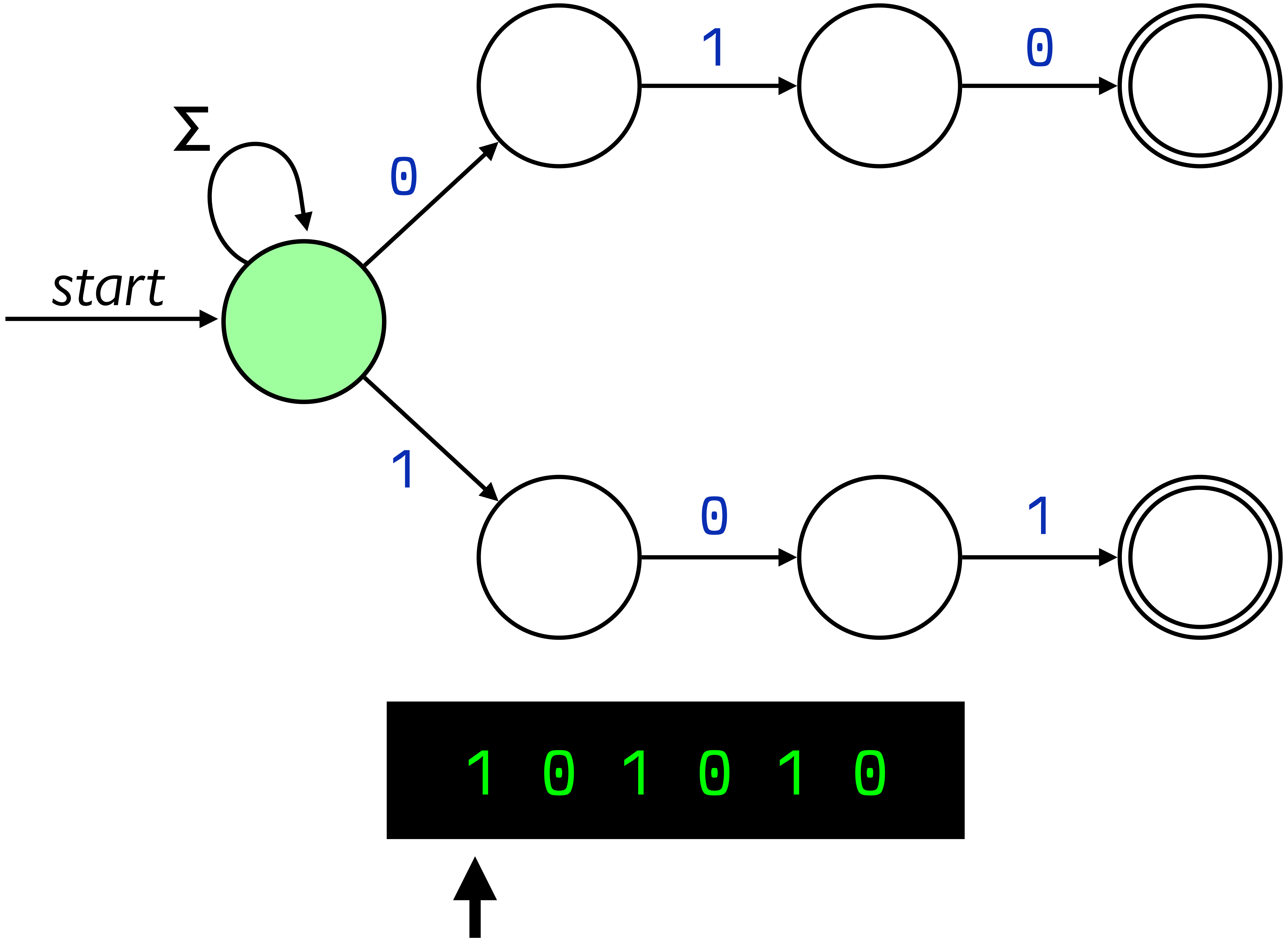
NFA



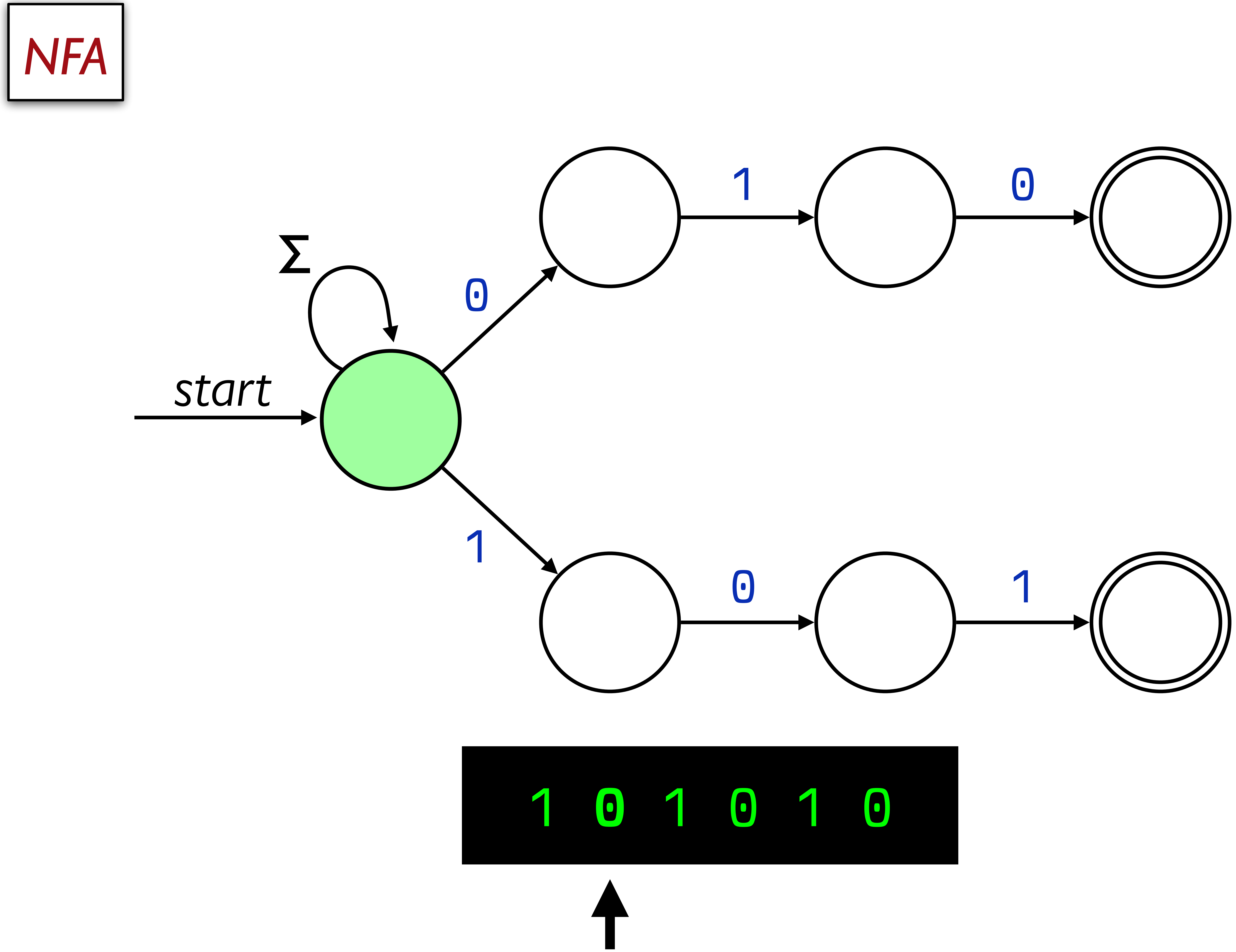
1 0 1 0 1 0

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

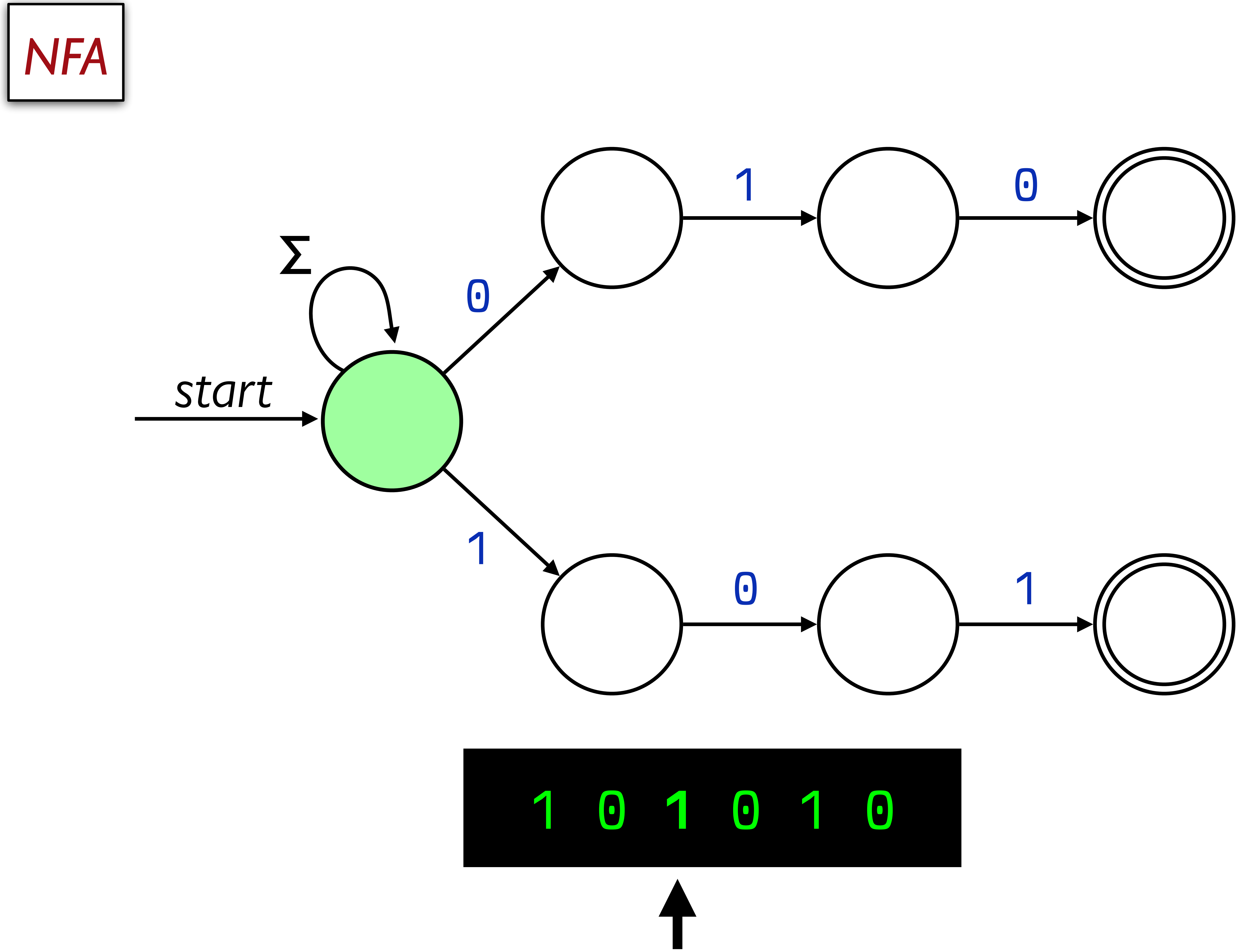
NFA



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

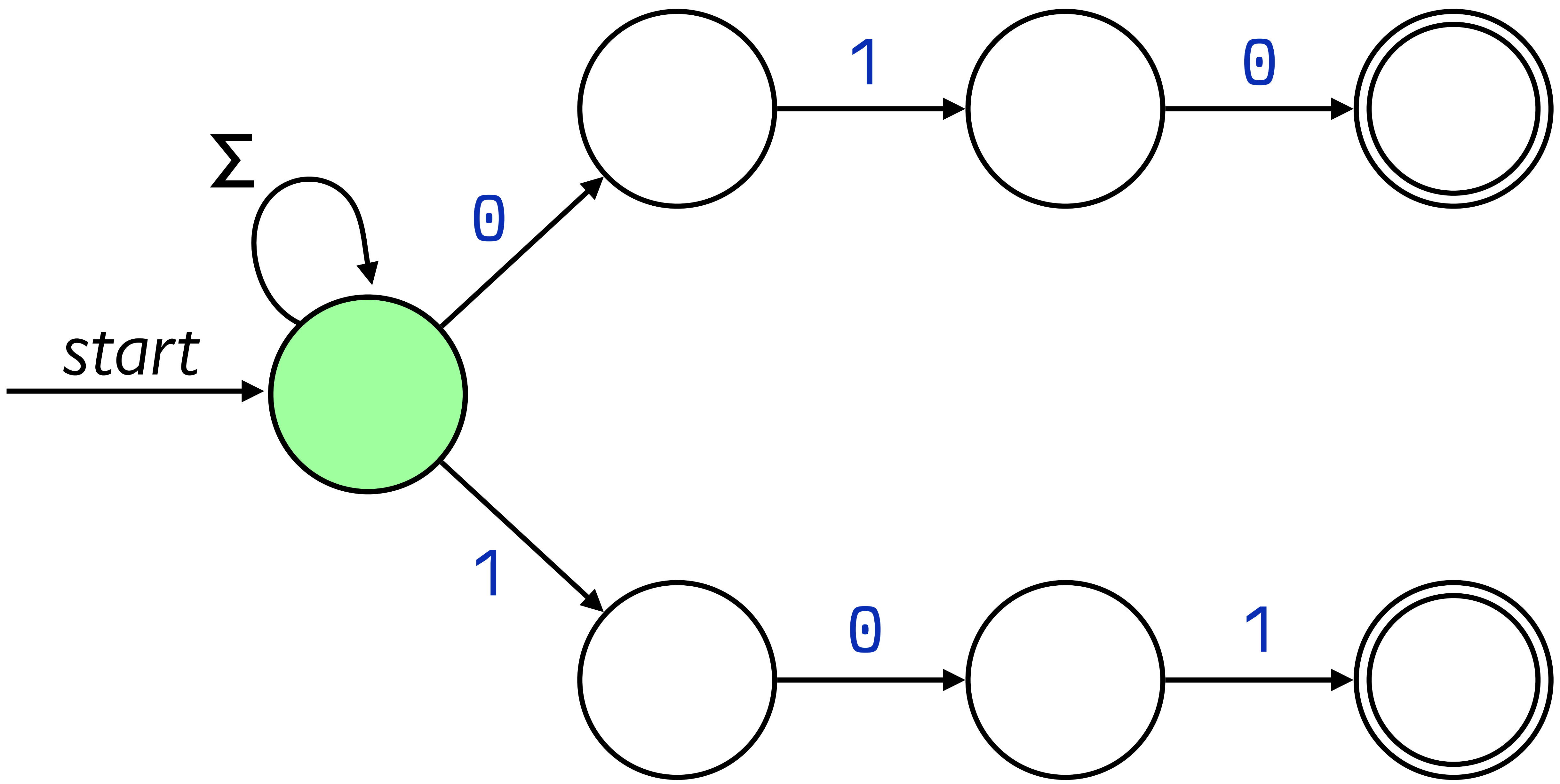


$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$



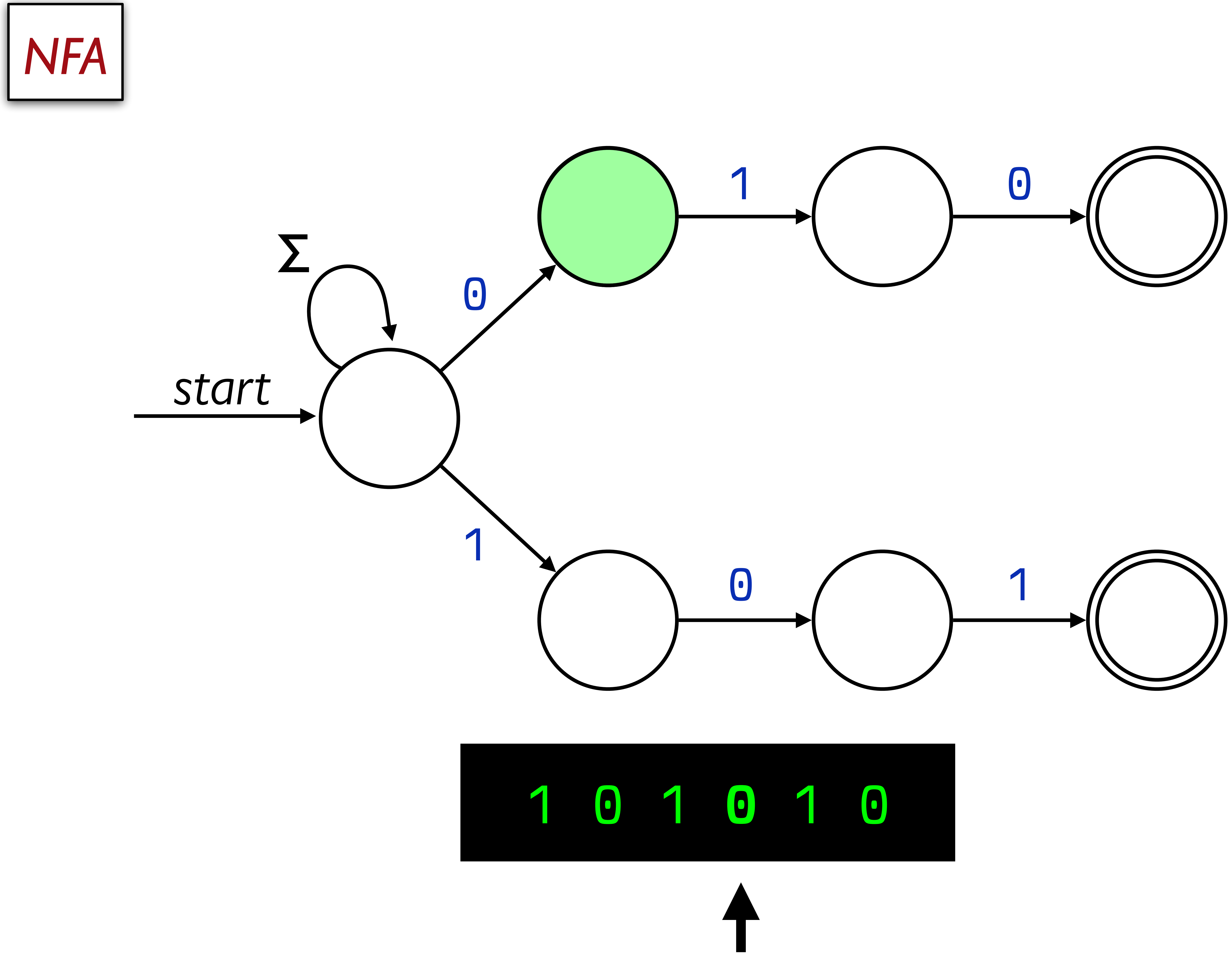
$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$

NFA

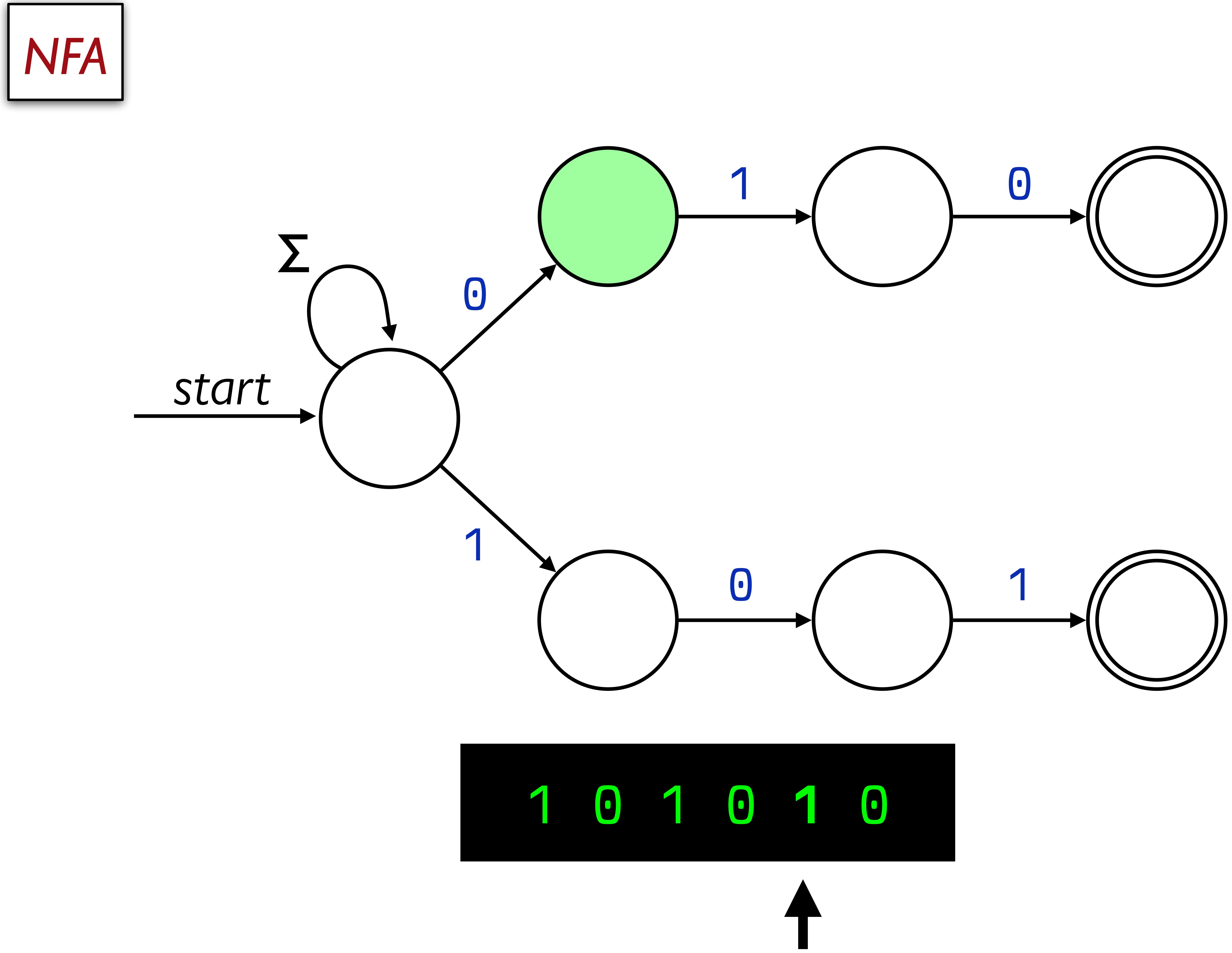


1 0 1 0 1 0

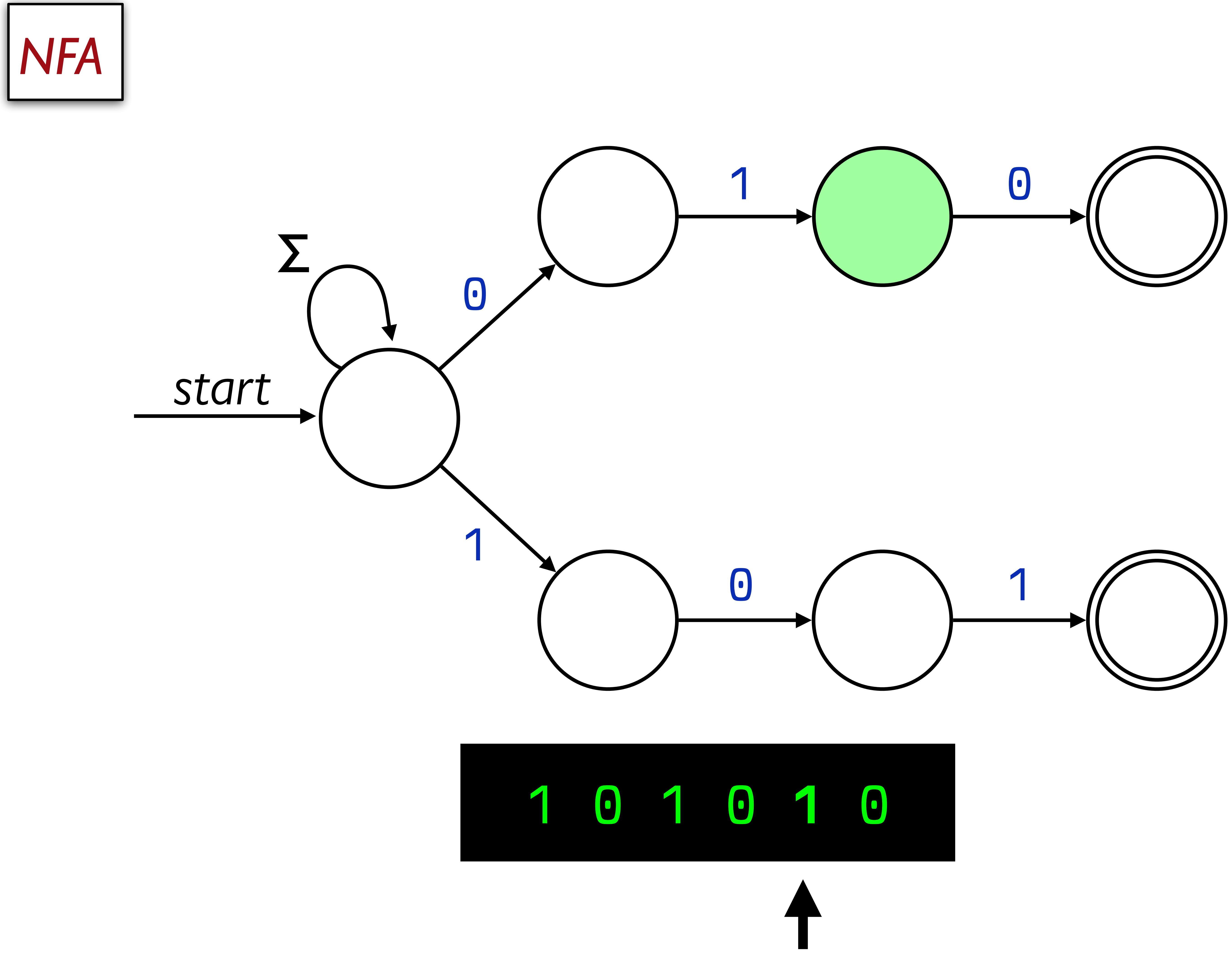
$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$



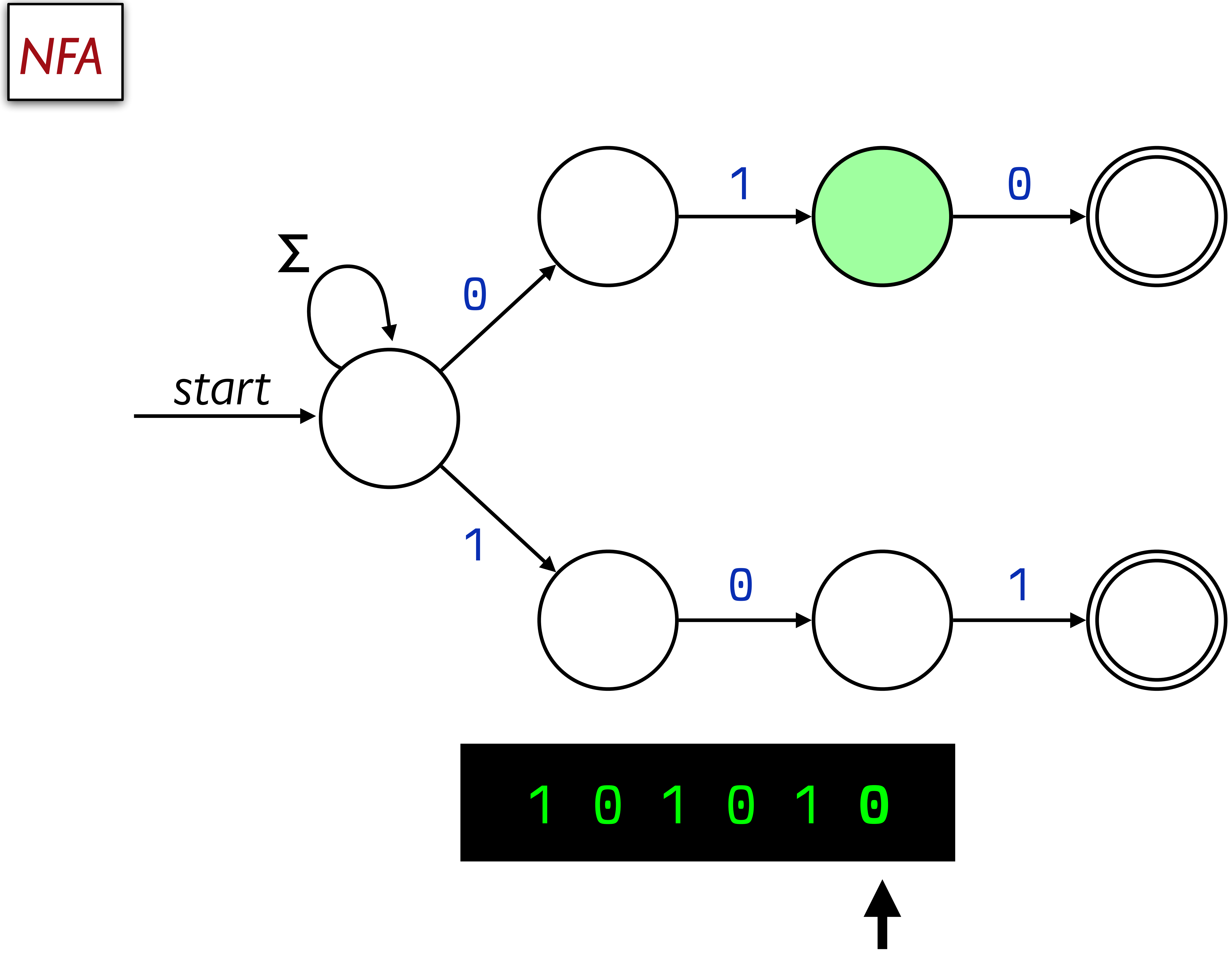
$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$



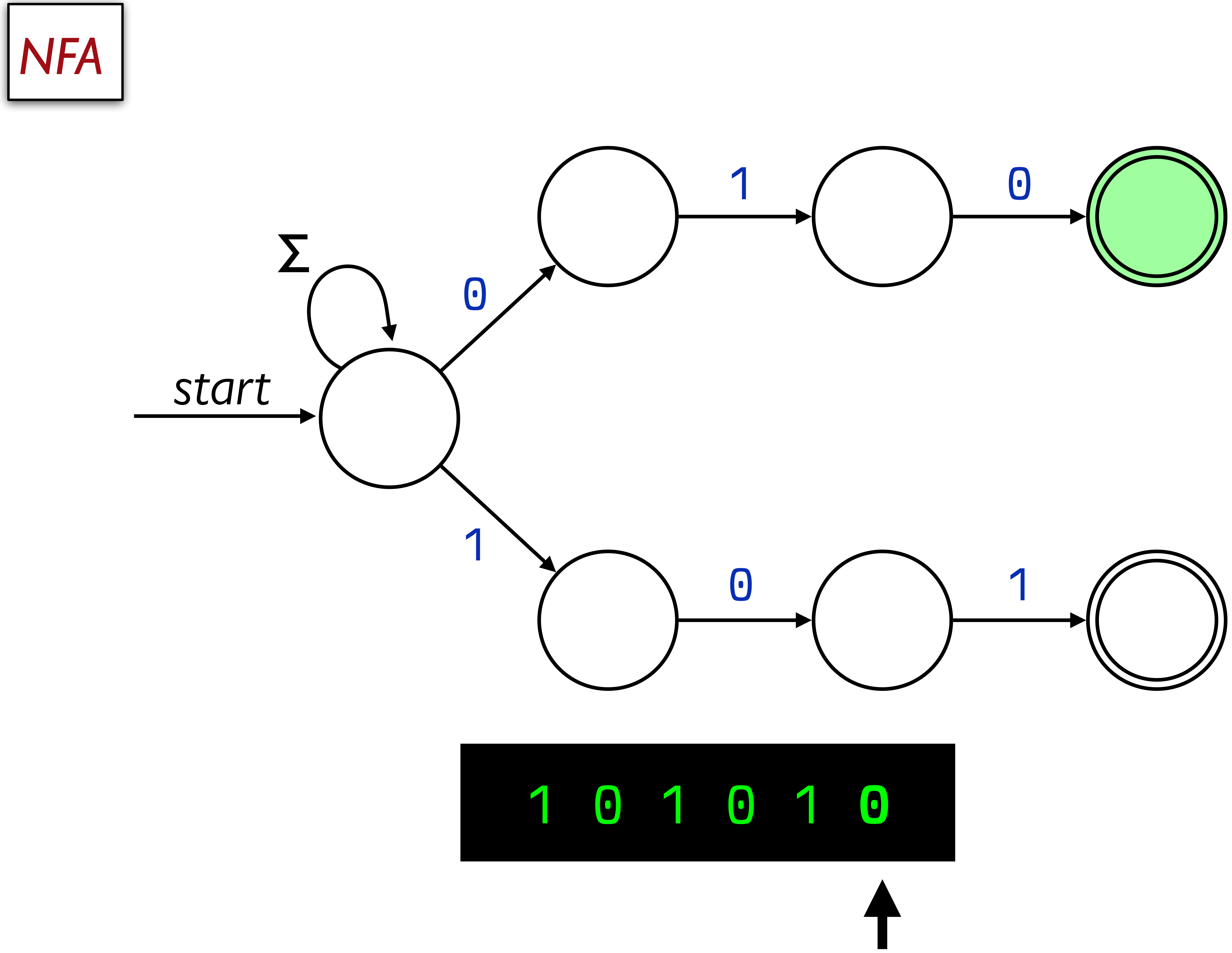
$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$



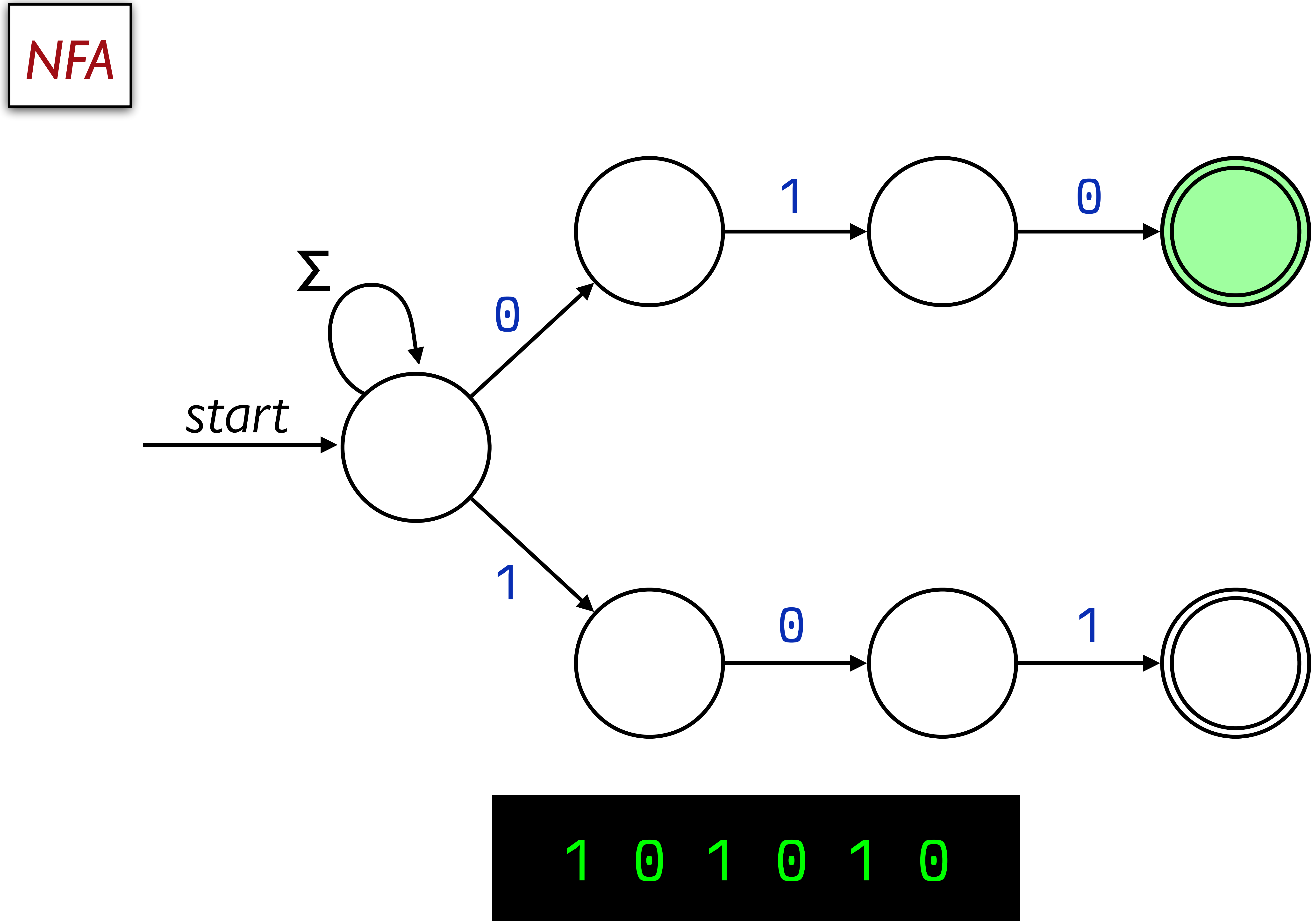
$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$



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

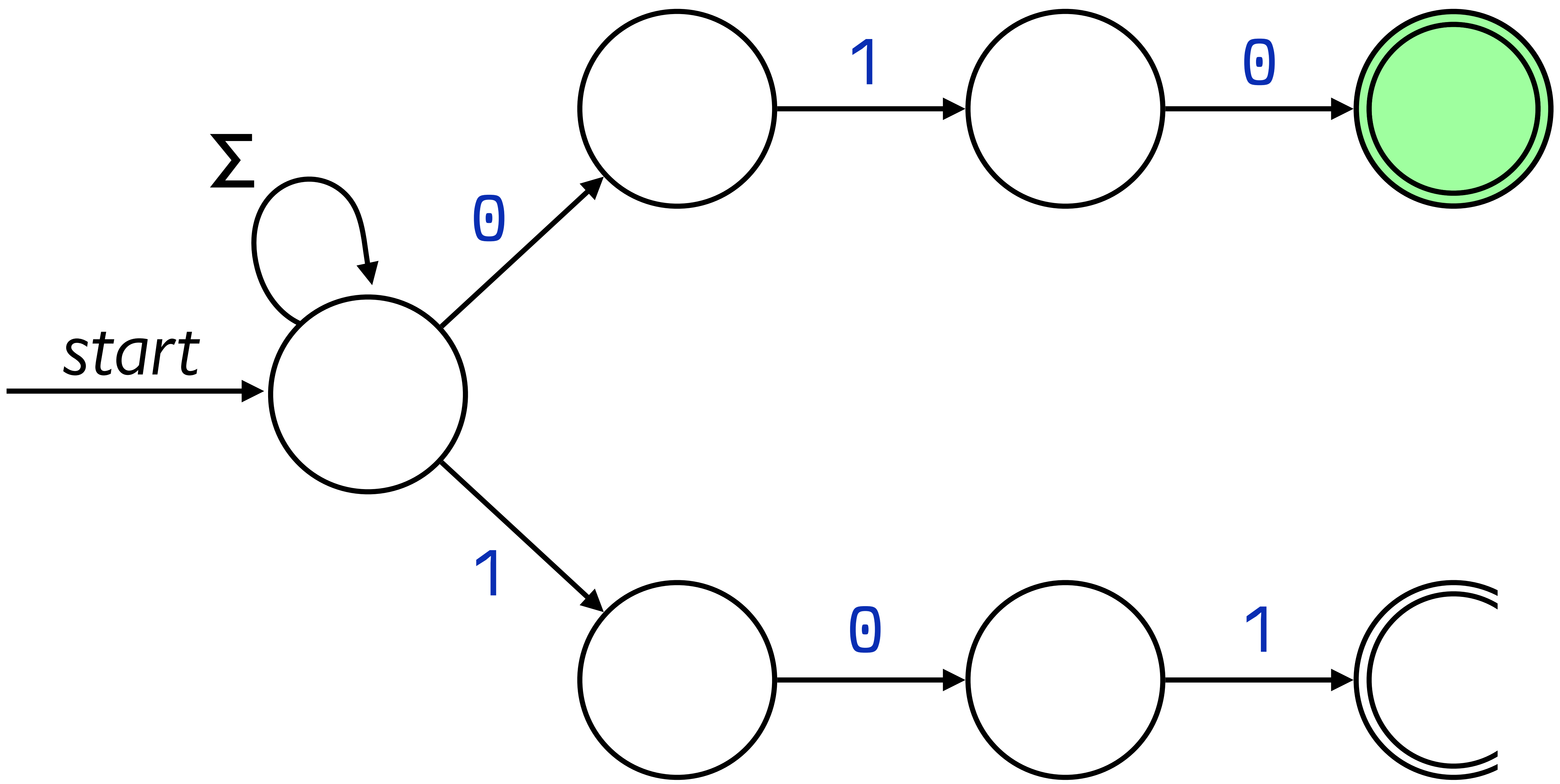


$L = \{w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101\}$



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

NFA



1 0 1 0 1 0

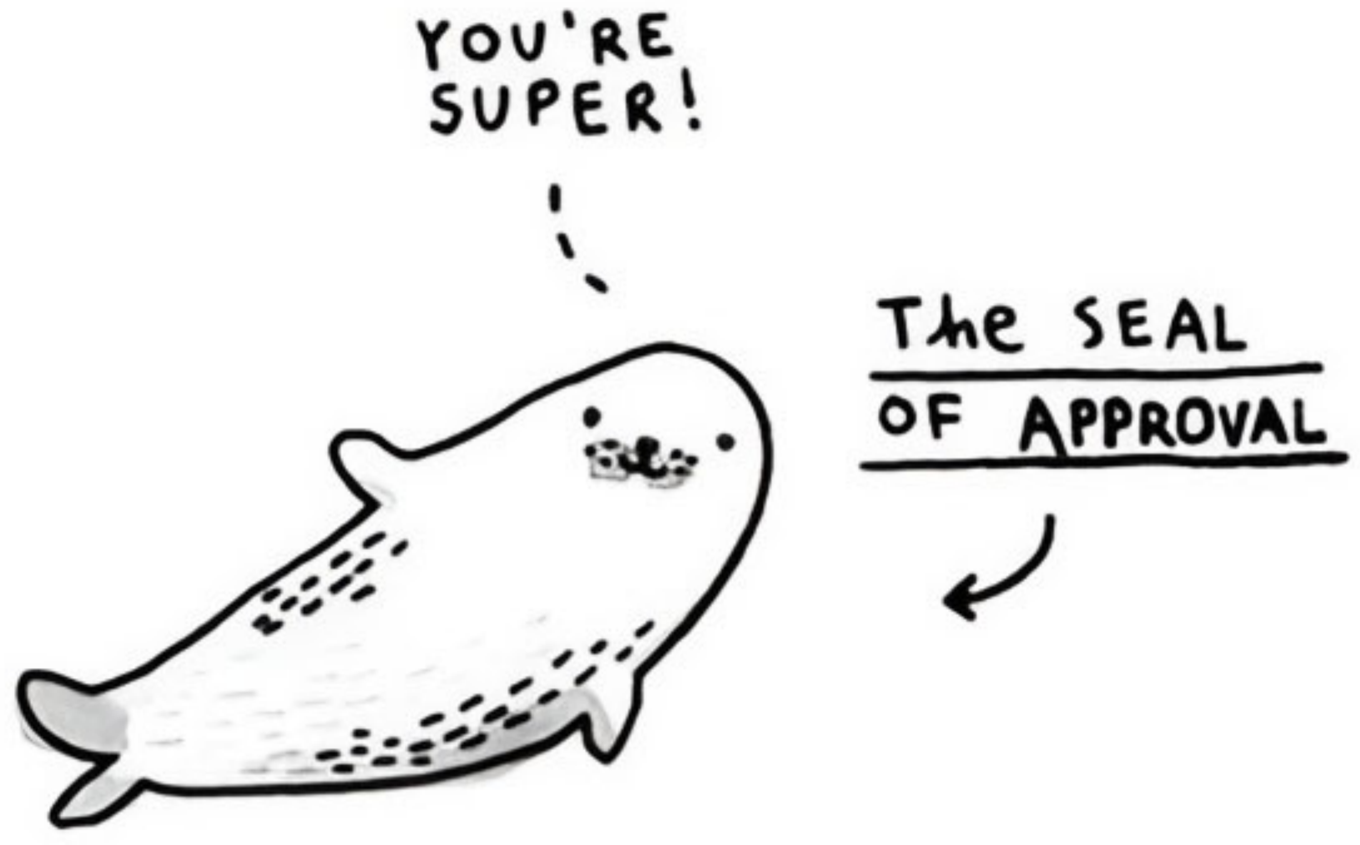
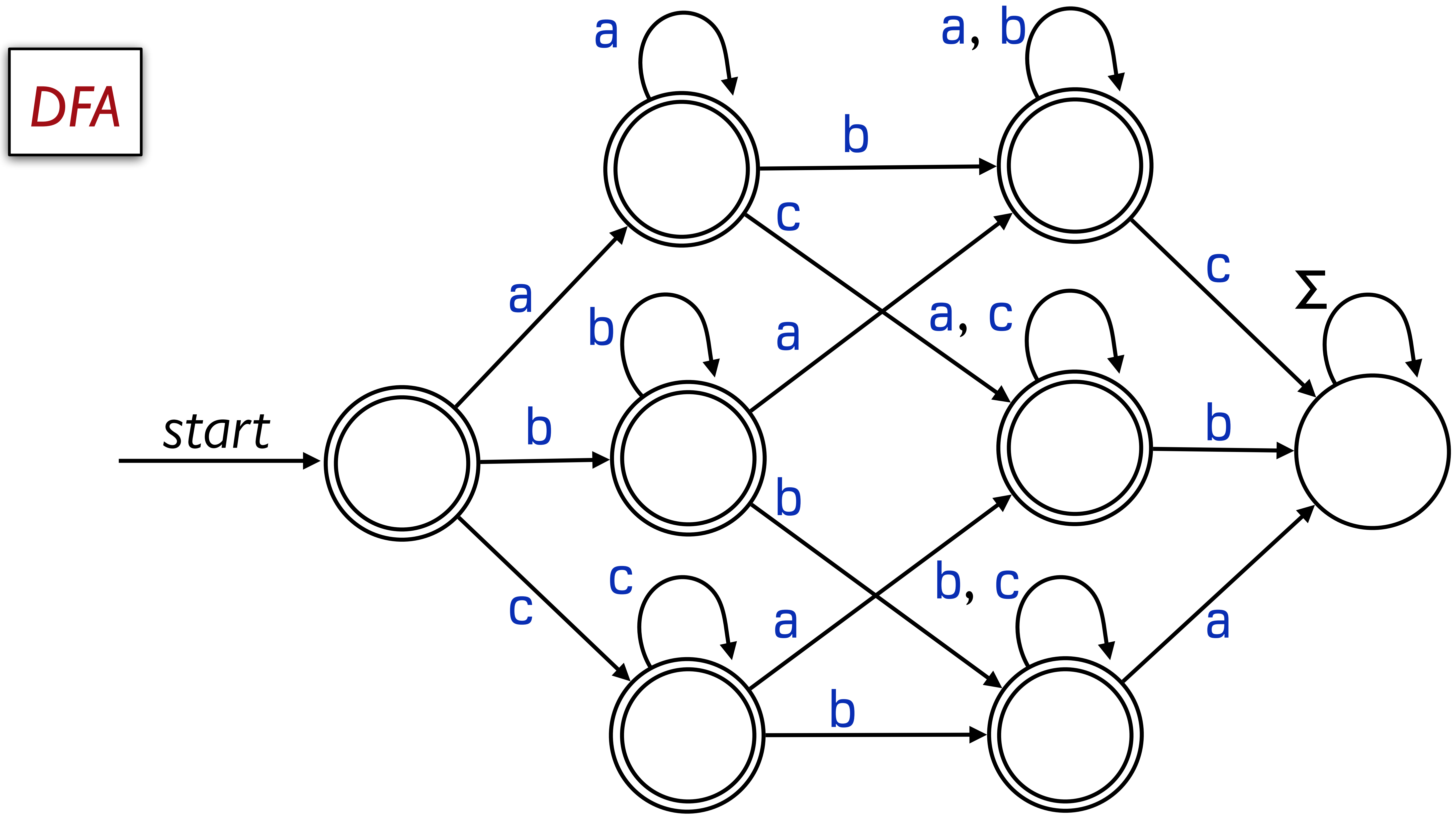


Illustration by Gemma Correll

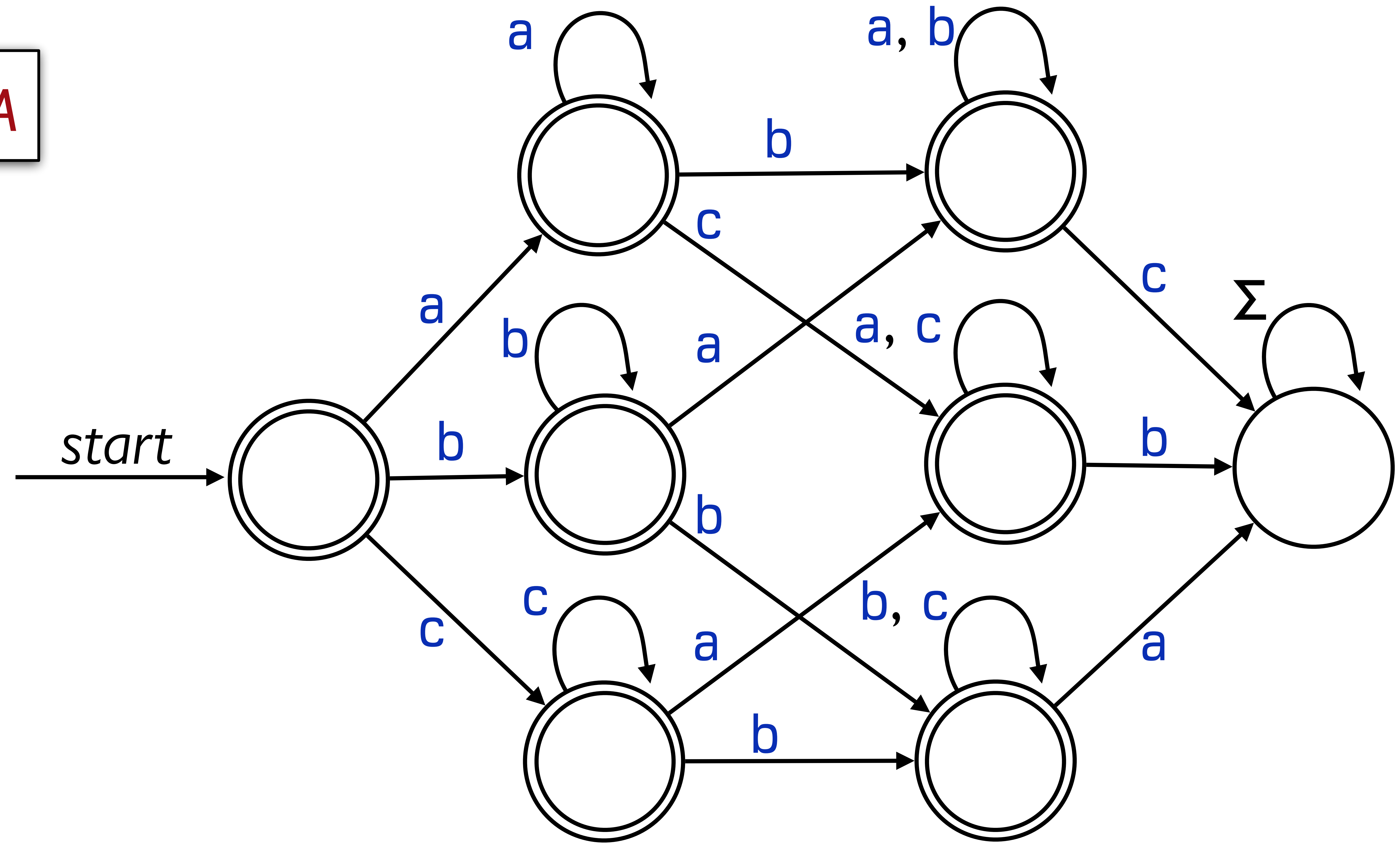
$L = \{w \in \{a, b, c\}^* \mid$ at least one of a, b, or c is not in $w\}$

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



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

DFA

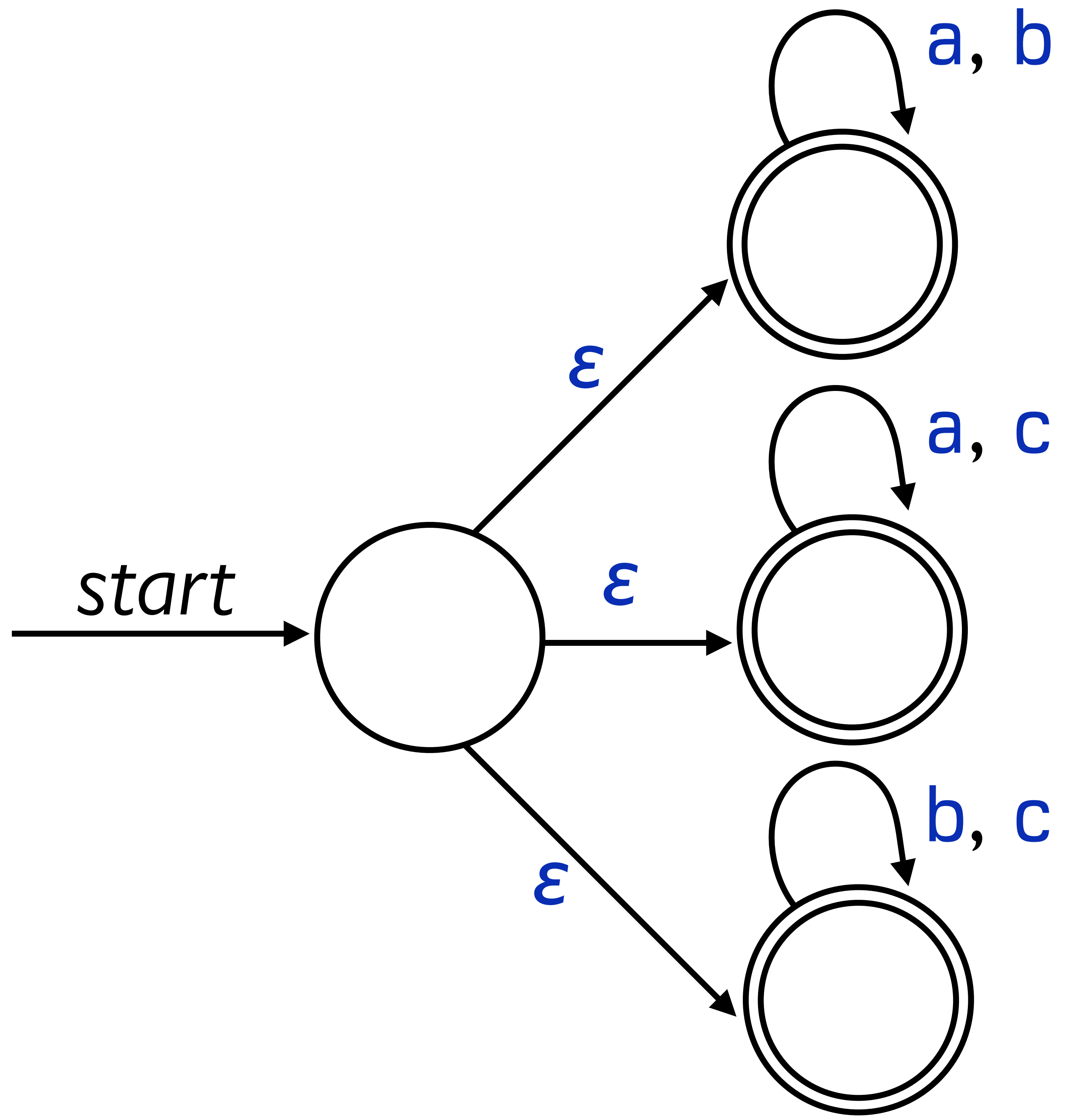


$L = \{w \in \{a, b, c\}^* \mid \text{at least one of a, b, or c is not in } w\}$

NFA

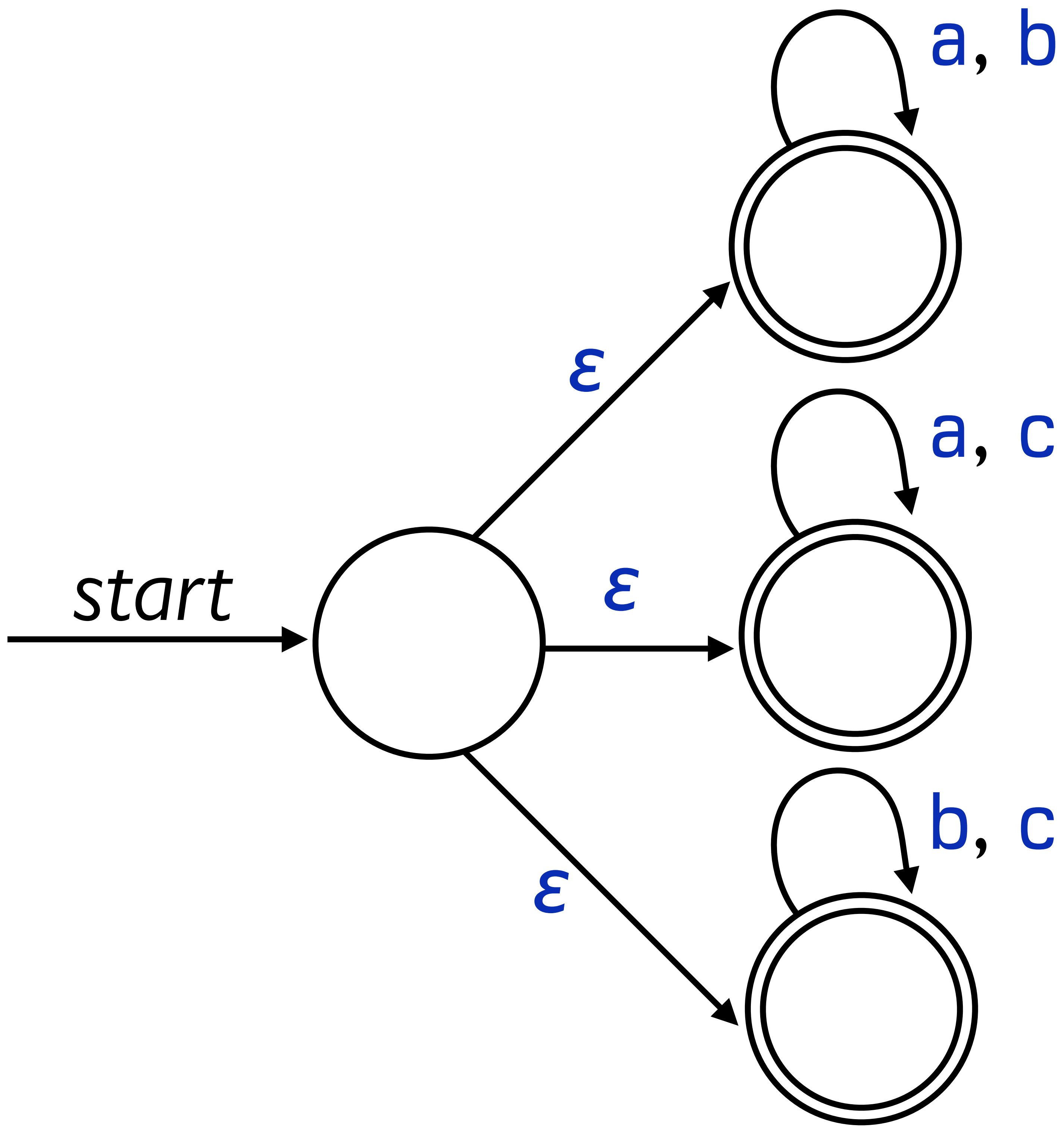
$L = \{w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w\}$

NFA



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

NFA



Nondeterministically ***guess*** *which character is missing.*

Deterministically ***check*** *whether that character is indeed missing.*

Next time

Has nondeterminism made our finite automata more powerful? How do NFAs compare to DFAs?

Acknowledgments

This lecture incorporates material from:

Nancy Ide, Vassar College

Keith Schwarz, Stanford University

Michael Sipser, *Introduction to the Theory of Computation*