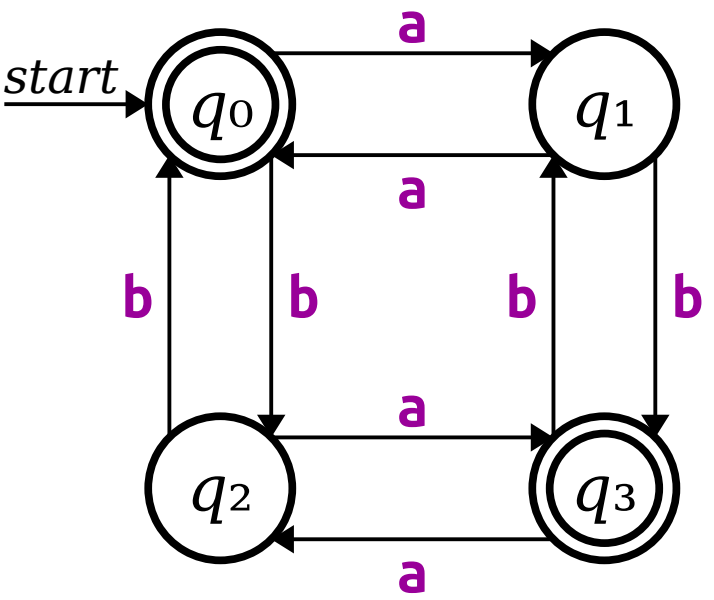
Finite Automata

Part Two

Recap from Last Time

Finite Automata

- A ***finite automaton*** is a mathematical model of a computing device.
- It consists of ***states*** linked by ***transitions***.
- Some of states may be marked as ***accepting states***. These are denoted with a double ring. The remaining states are ***rejecting states***.
- If the device ends in an accepting state after seeing all the input, ***accepts*** the input (says `YES`)
- If the device does not end in an accepting state after seeing all the input, it ***rejects*** the input (says `NO`).



Formal Language Theory

- An ***alphabet*** is a set, usually denoted Σ, consisting of elements called ***characters***.
  - $a \in \Sigma$ means "a is a single character."
- A ***string over Σ*** is a finite sequence of zero or more characters taken from Σ.
- The ***empty string*** has no characters and is denoted ε.
- A ***language over Σ*** is a set of strings over Σ.
- The language **Σ*** is the set of all strings over Σ.
  - $w \in \Sigma^*$ means "w is a string of characters from Σ."

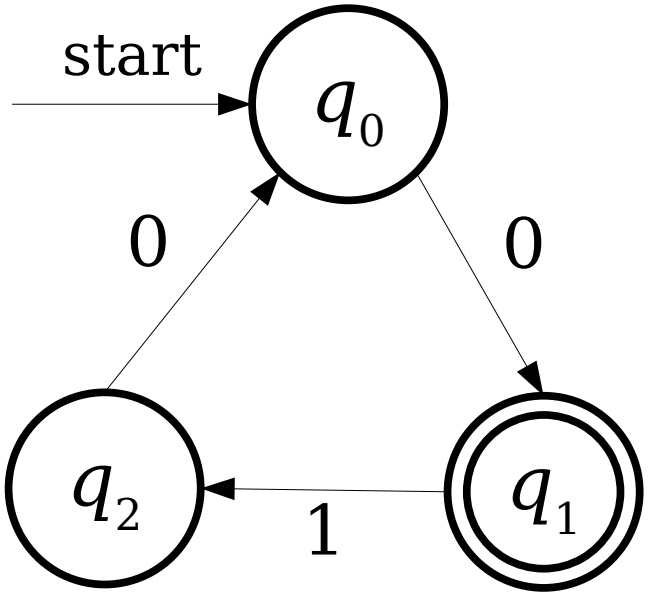
The Language of an Automaton

- If A is an automaton that processes strings over Σ, the ***language of A***, denoted $\mathcal{L}(A)$, is the set of all strings A accepts.
- Formally:

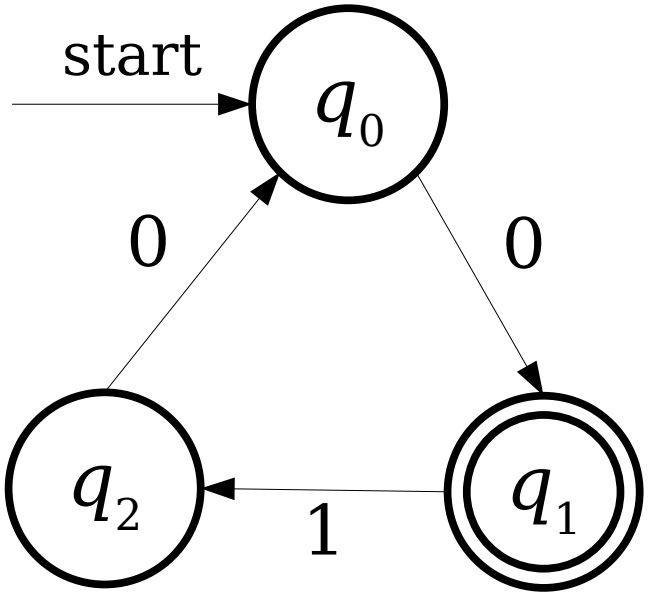
$$\mathcal{L}(A) = \{ w \in \Sigma^* \mid A \text{ accepts } w \}$$

New Stuff!

A Small Problem

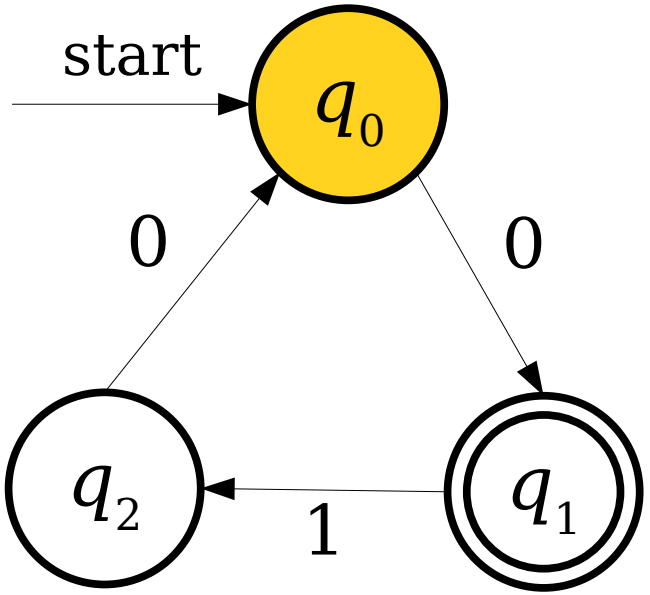


A Small Problem



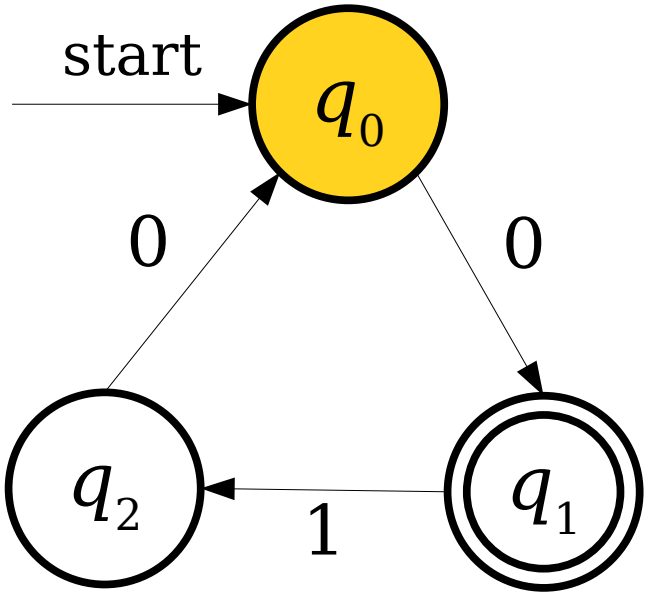
0	1	1	0

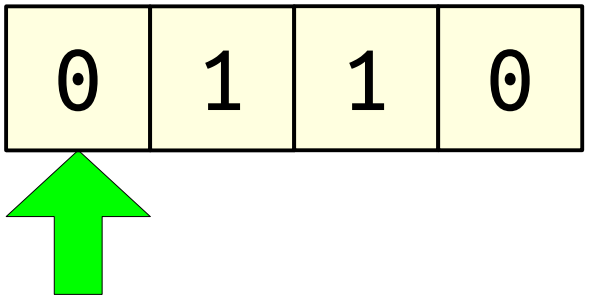
A Small Problem



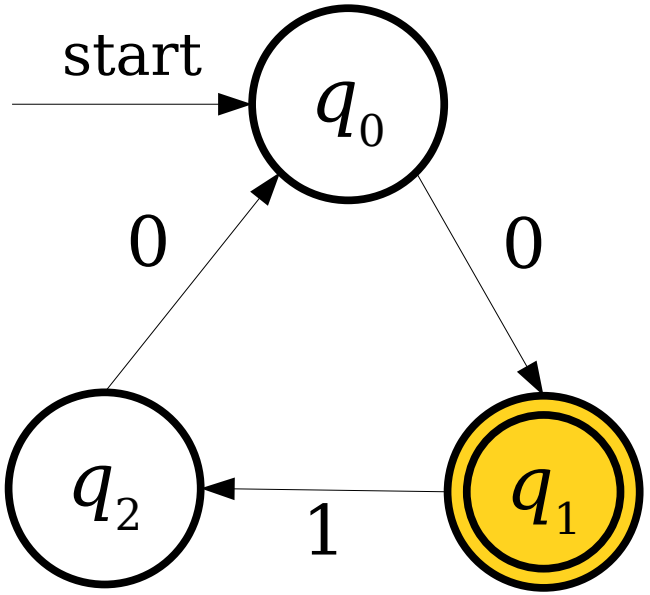
0	1	1	0

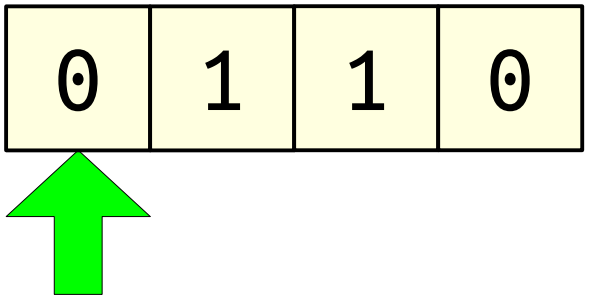
A Small Problem



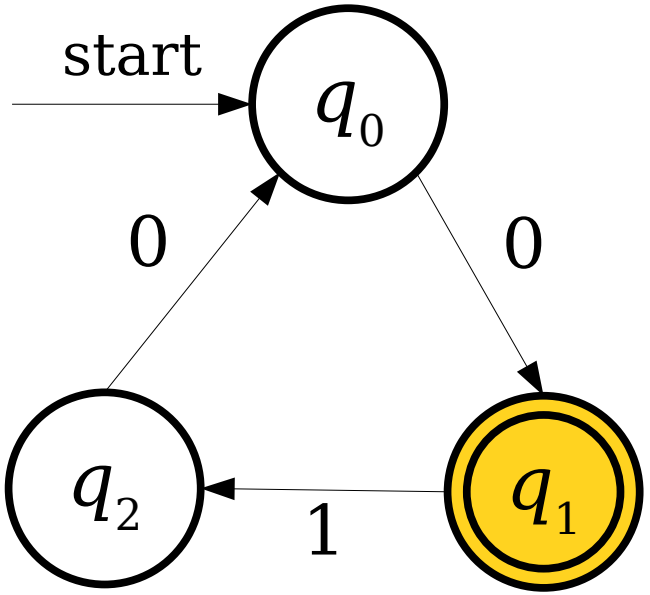


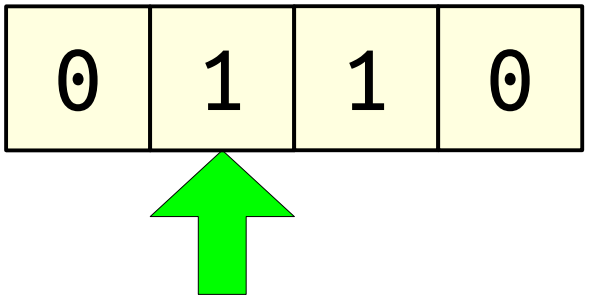
A Small Problem



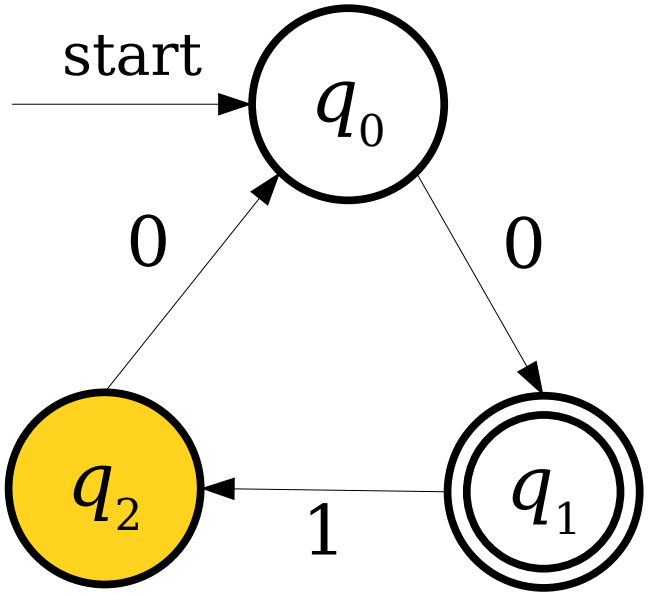


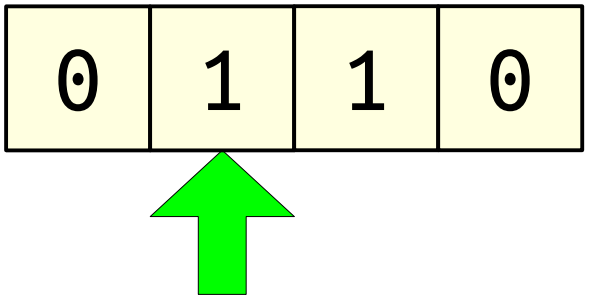
A Small Problem



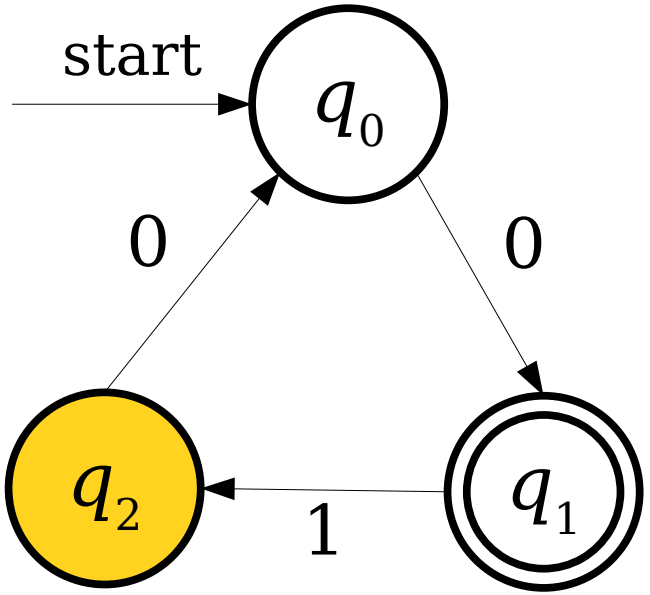


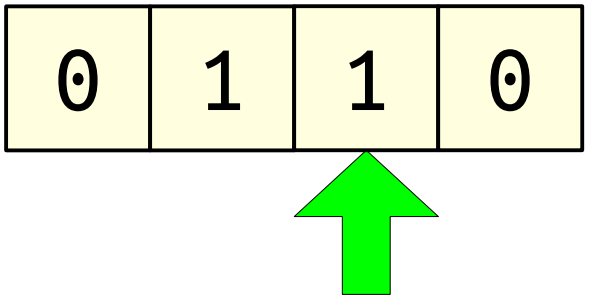
A Small Problem



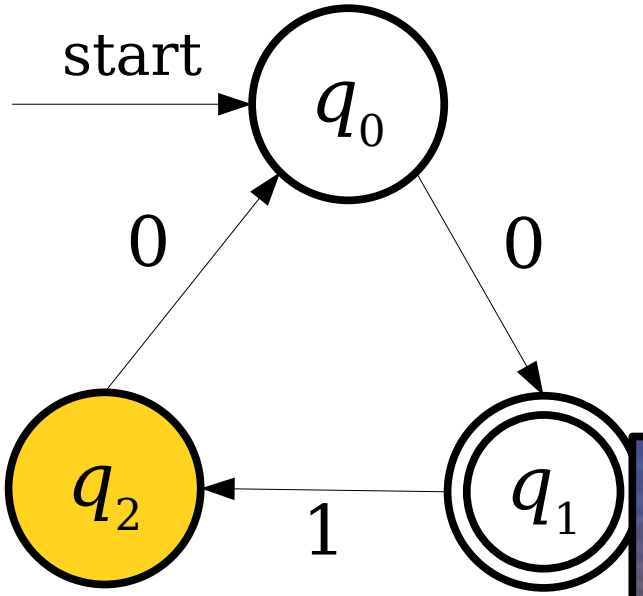


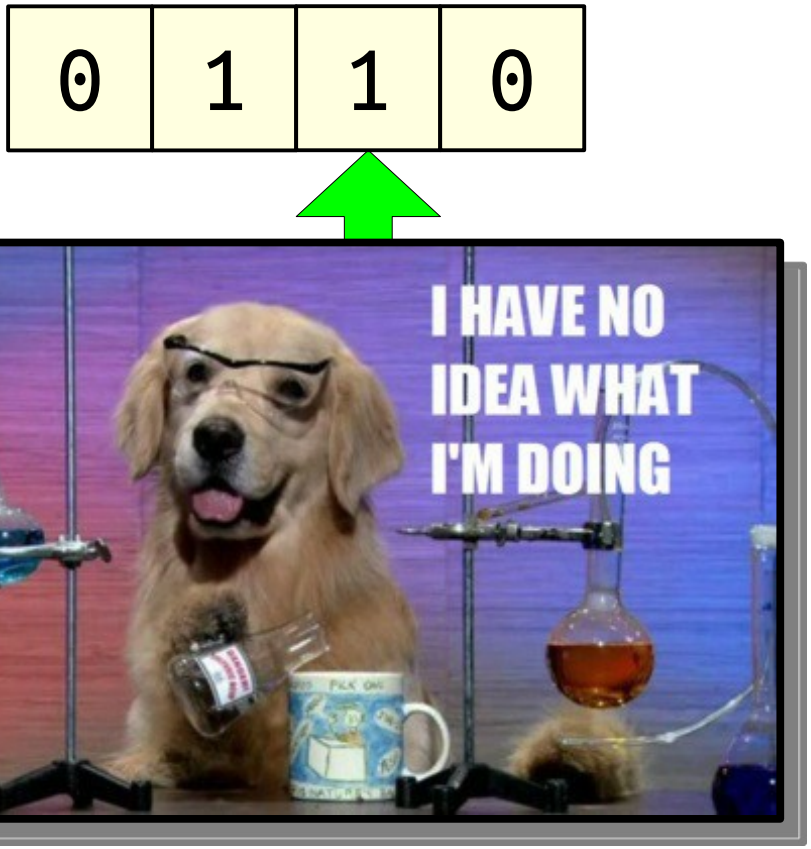
A Small Problem



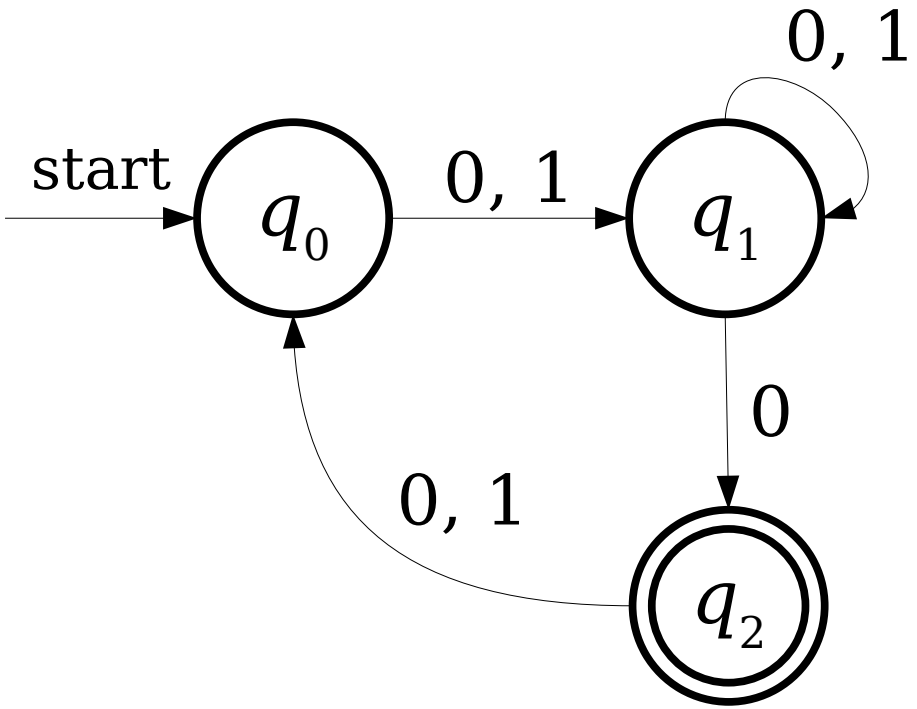


A Small Problem

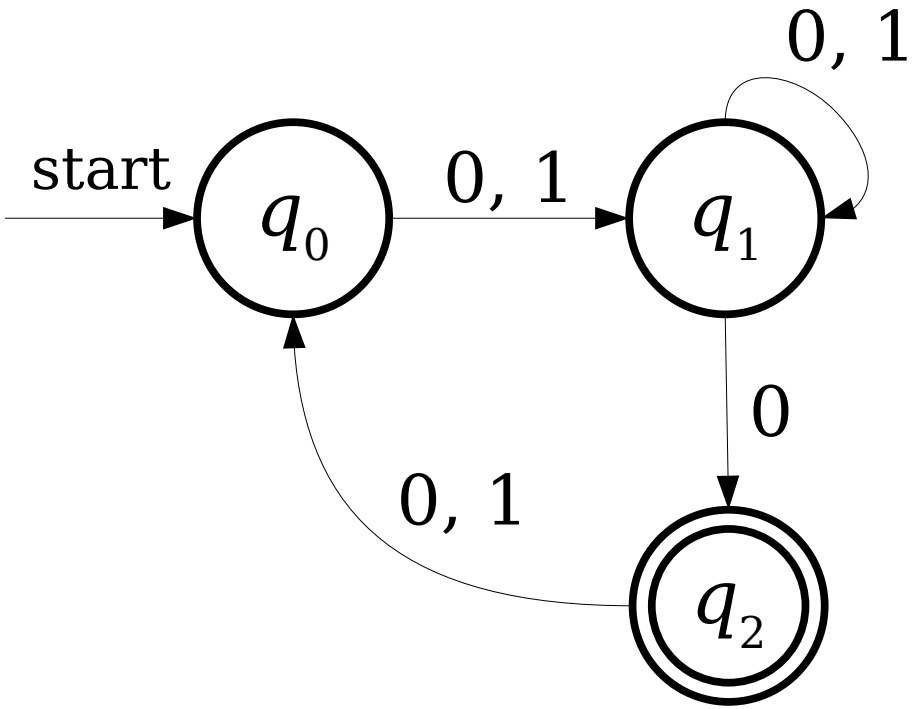


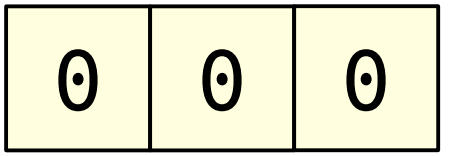


Another Small Problem

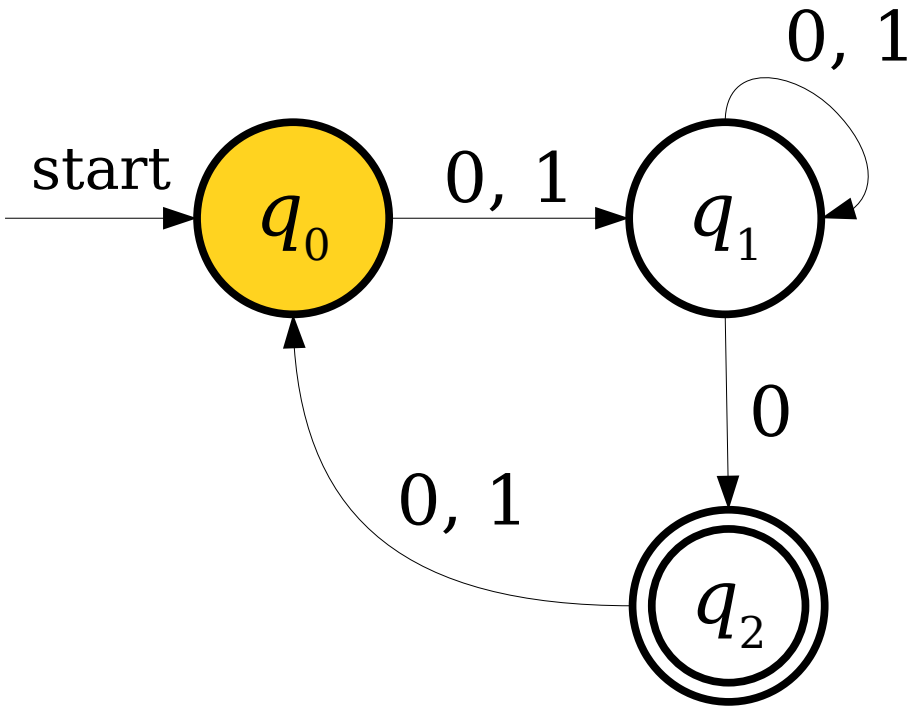


Another Small Problem



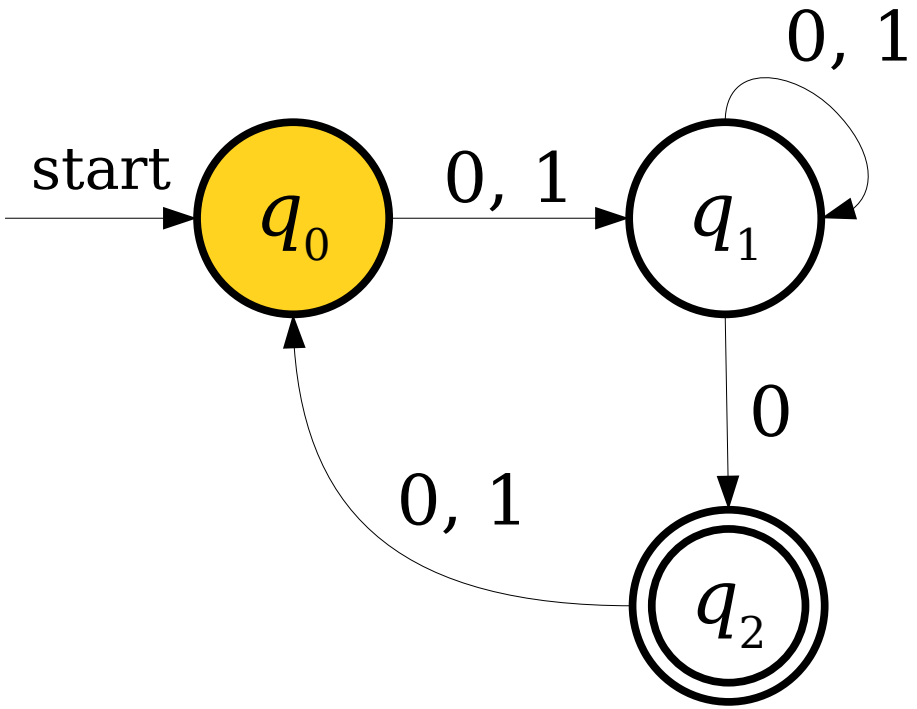


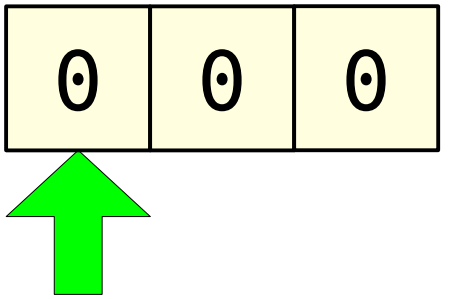
Another Small Problem



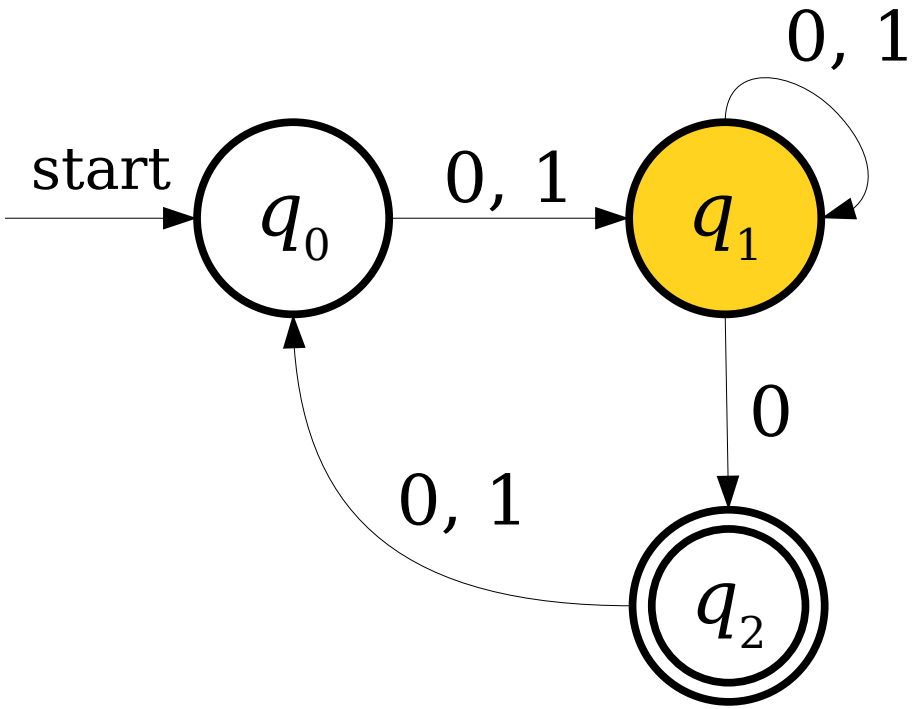
0	0	0

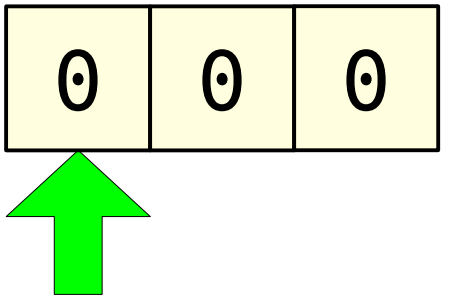
Another Small Problem



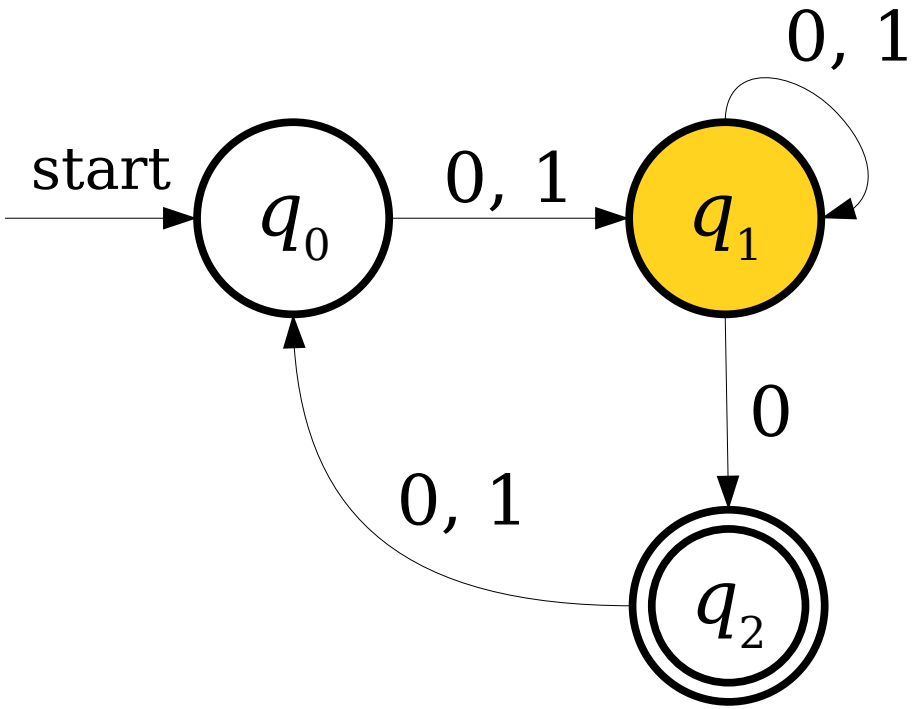


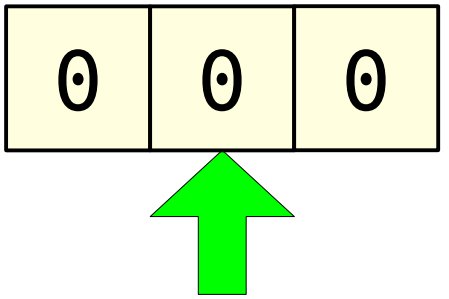
Another Small Problem



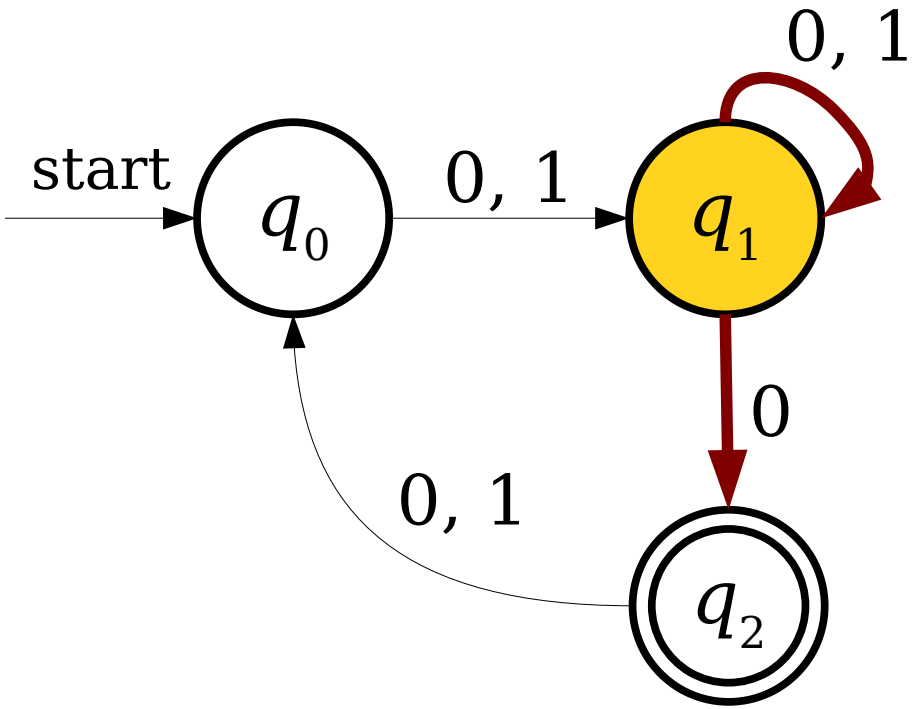


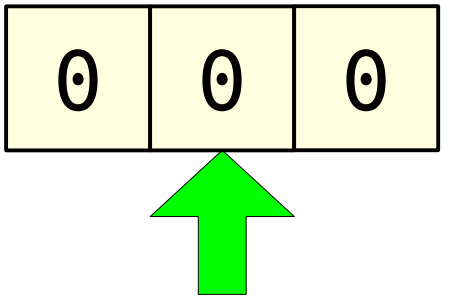
Another Small Problem



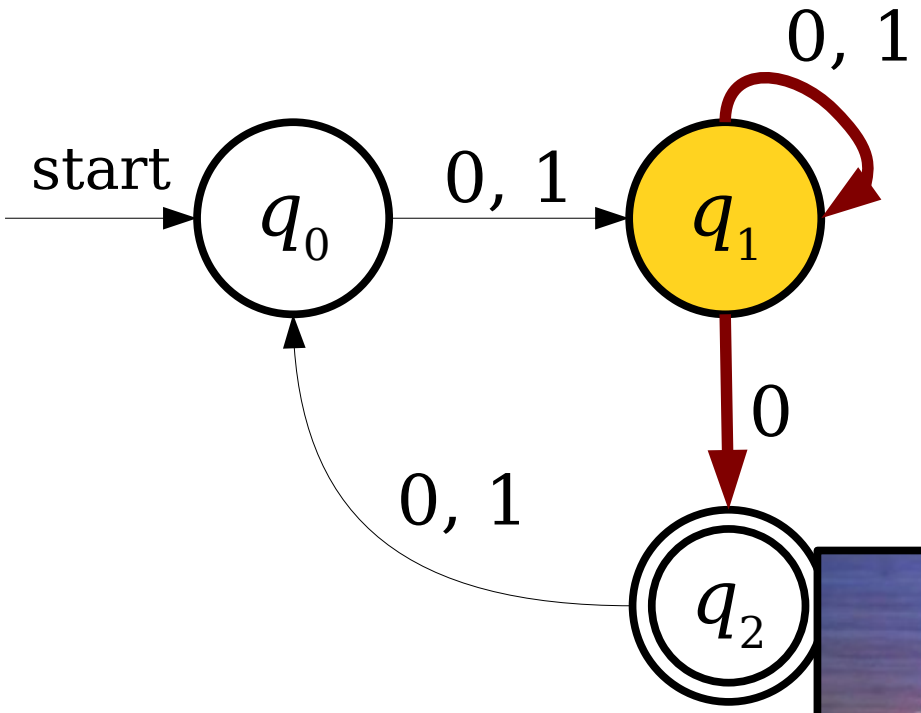


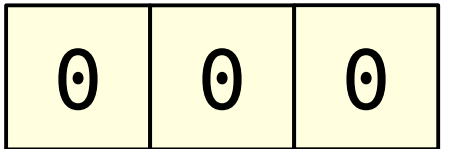
Another Small Problem

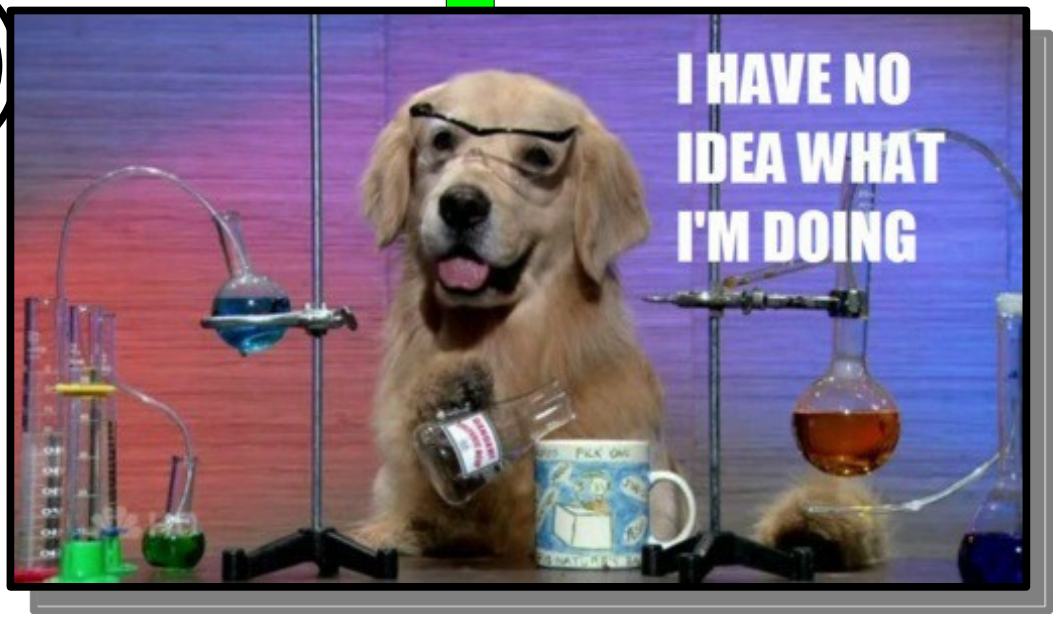




Another Small Problem







Two Solutions

- Today, we'll see two ways of addressing this issue.
- First, we'll introduce ***DFAs***, which disallow choices, and see how to design them.
- Then, we'll introduce ***NFAs***, which allow choices, and see how to interpret that power of choice.

DFAs

- A ***DFA*** is a
  - ***D***eterministic
  - ***F***inite
  - ***A***utomaton
- DFAs are the simplest type of automaton that we will see in this course.

DFAs

- A DFA is defined relative to some alphabet Σ.
- For each state in the DFA, there must be ***exactly one*** transition defined for each symbol in Σ.
  - This is the “deterministic” part of DFA.
- There is a unique start state.
- There are zero or more accepting states.

Designing DFAs

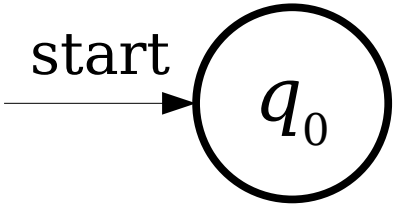
- At each point in its execution, the DFA can only remember what state it is in.
- ***DFA Design Tip:*** Build each state to correspond to some piece of information you need to remember.
  - Each state acts as a “memento” of what you're supposed to do next.
  - Only finitely many different states means only finitely many different things the machine can remember.

Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$

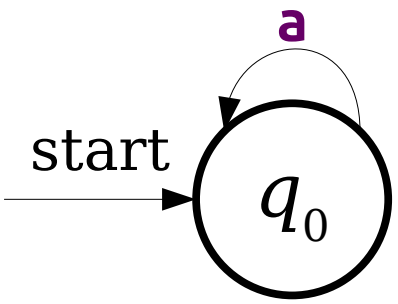
Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



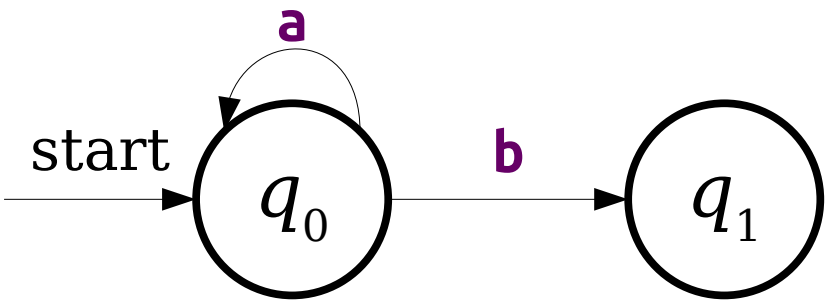
Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



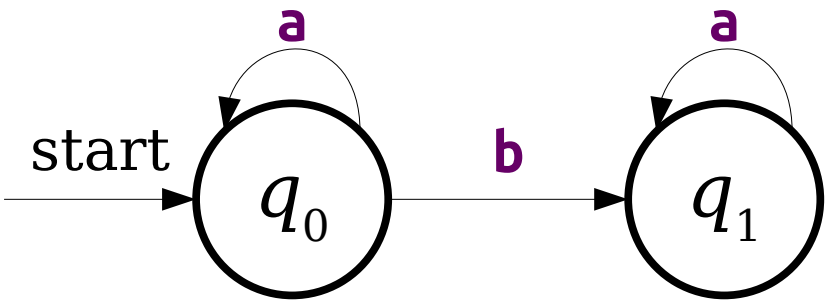
Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



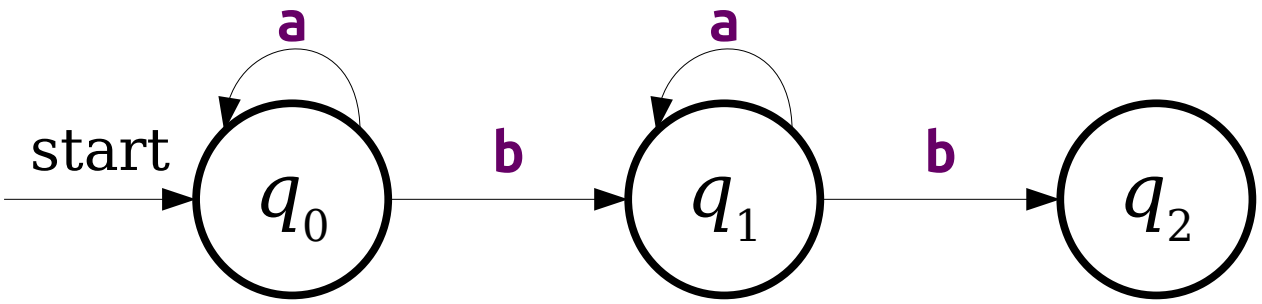
Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



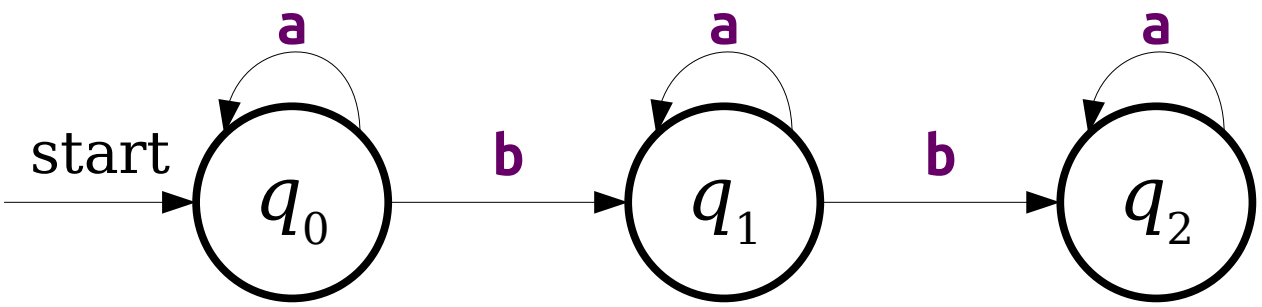
Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



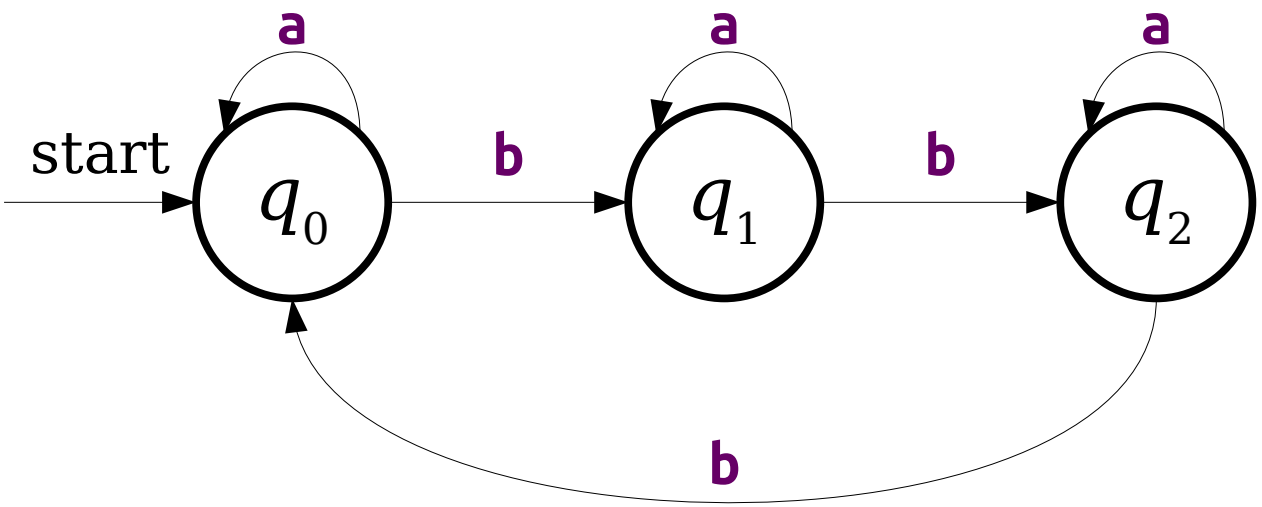
Recognizing Languages with DFAs

$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



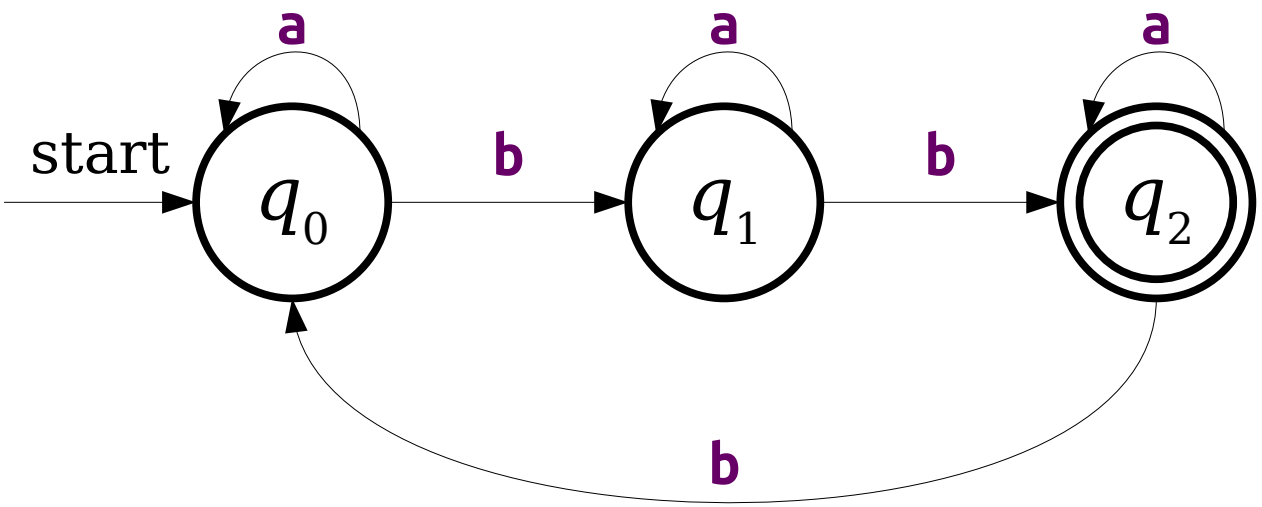
Recognizing Languages with DFAs

$L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



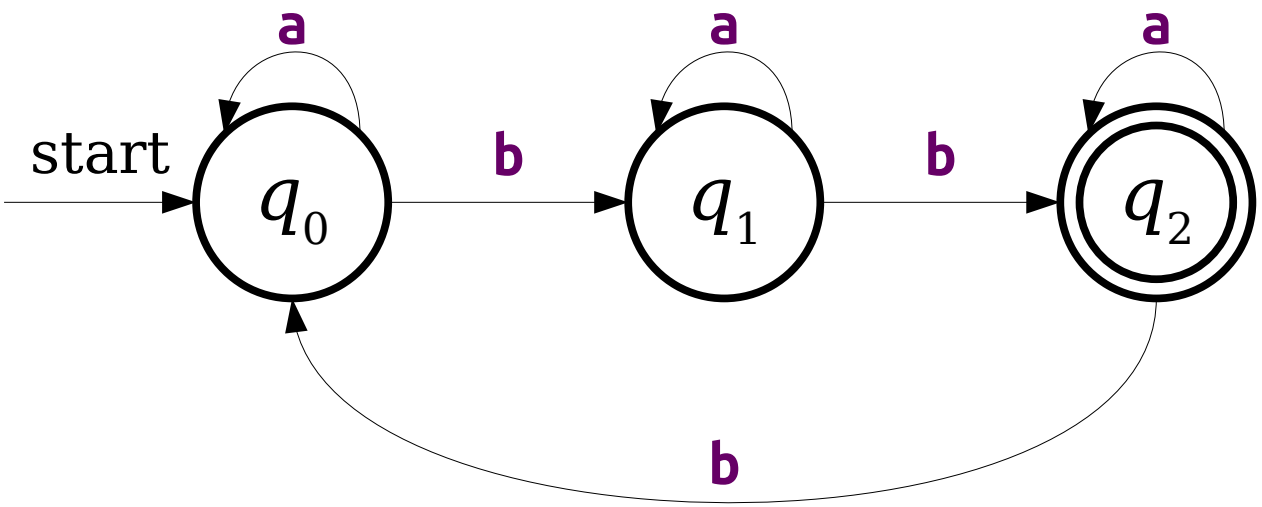
Recognizing Languages with DFAs

$L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



Recognizing Languages with DFAs

$L = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* |$ the number of **b**'s in w is congruent to two modulo three $\}$



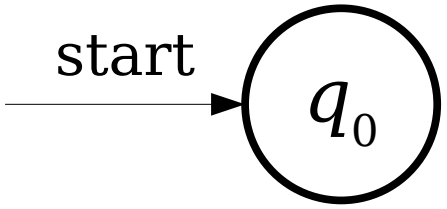
Each state remembers the remainder of the number of **b**s seen so far modulo three.

Recognizing Languages with DFAs

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

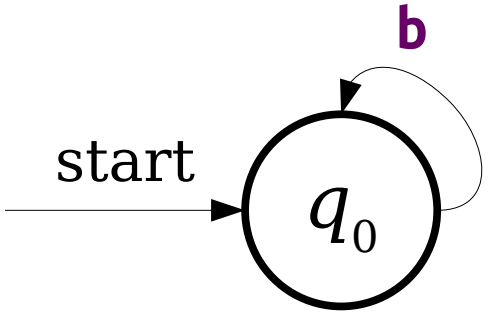
Recognizing Languages with DFAs

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



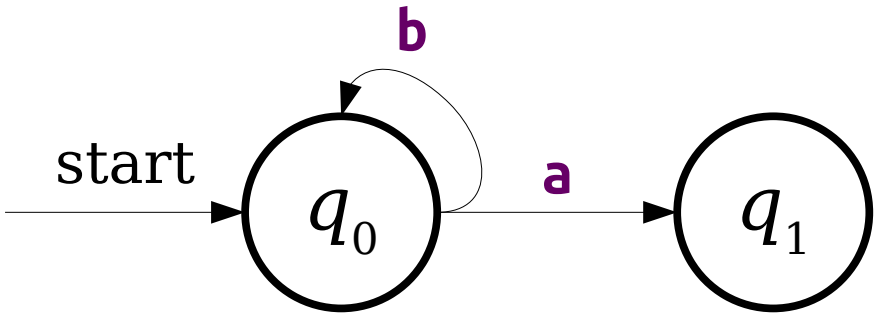
Recognizing Languages with DFAs

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



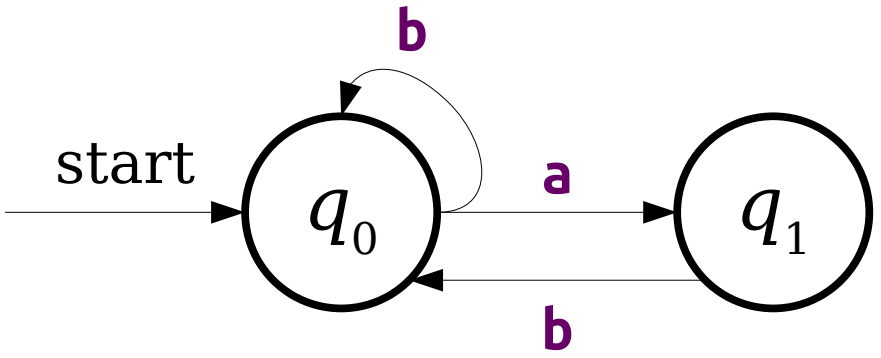
Recognizing Languages with DFAs

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



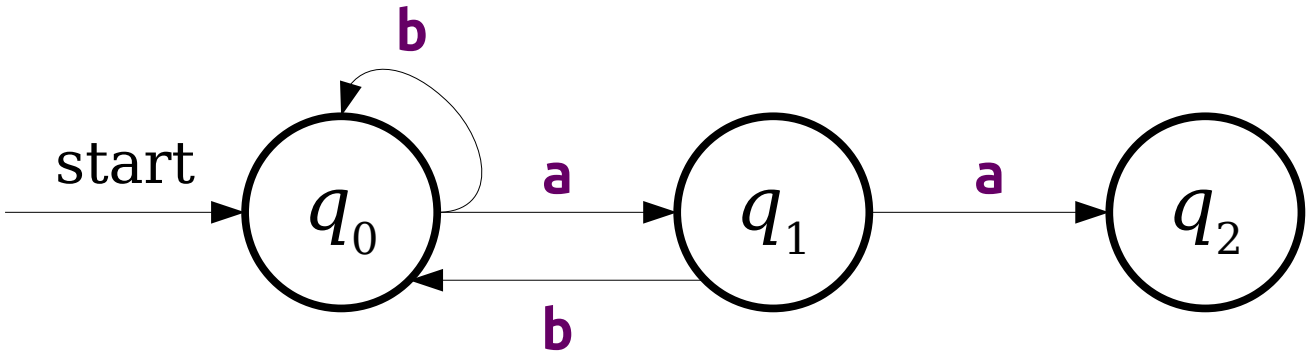
Recognizing Languages with DFAs

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



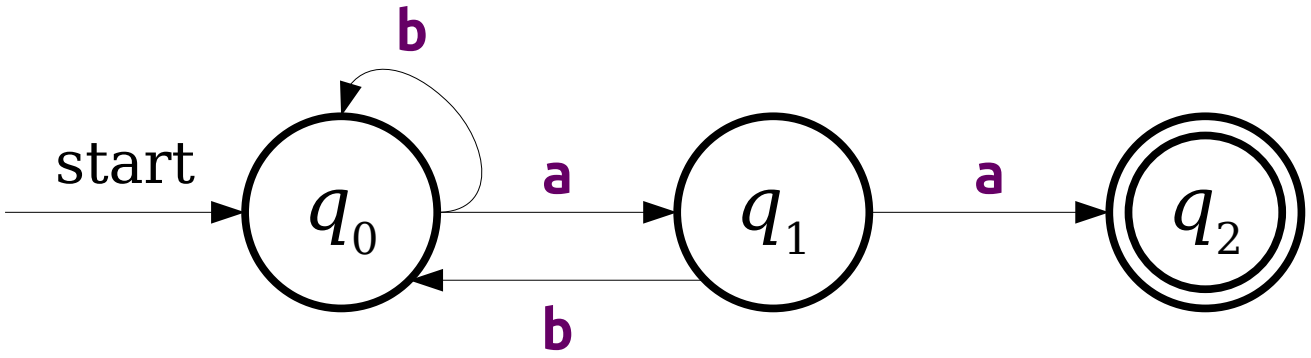
Recognizing Languages with DFAs

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



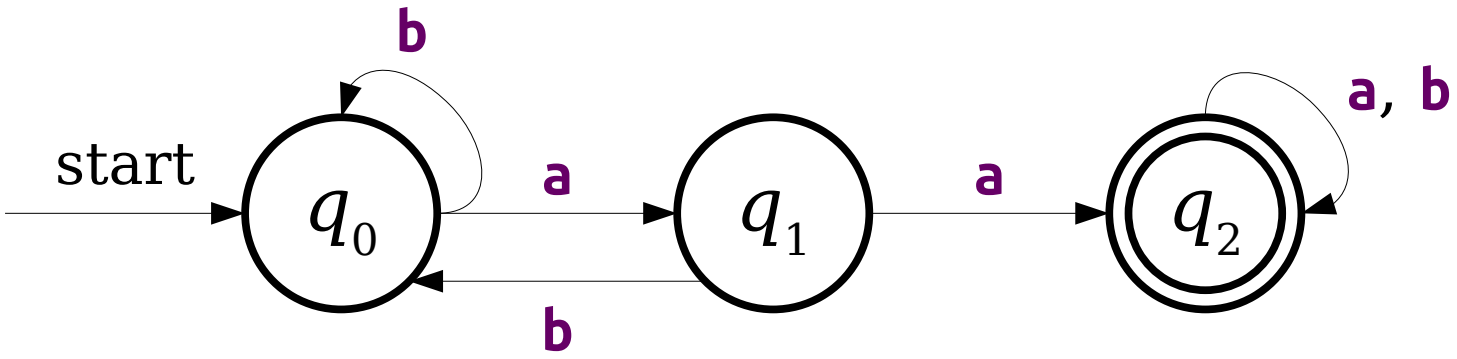
Recognizing Languages with DFAs

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



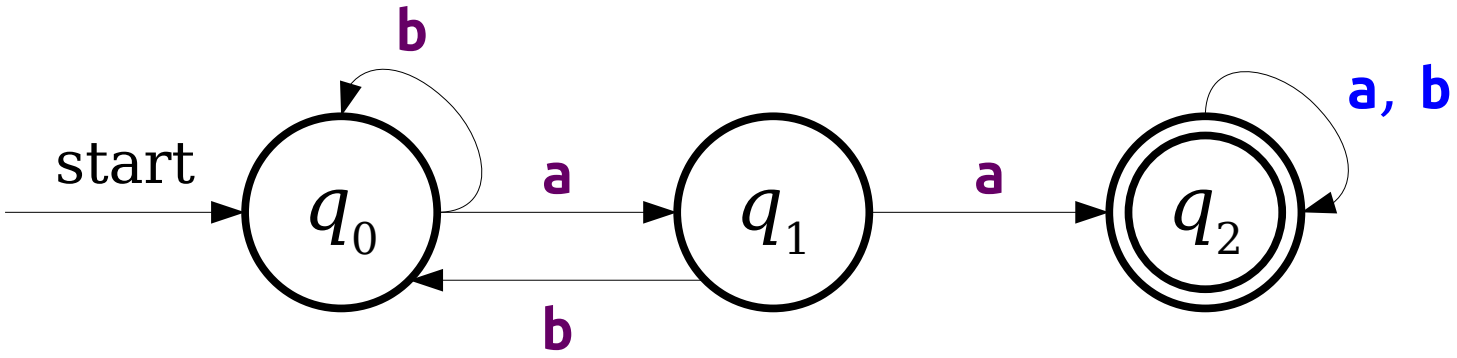
Recognizing Languages with DFAs

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



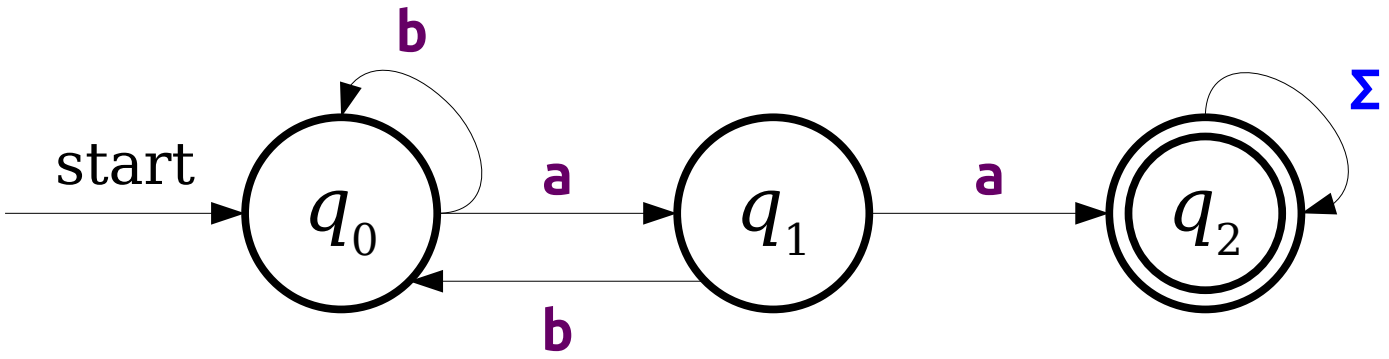
Recognizing Languages with DFAs

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



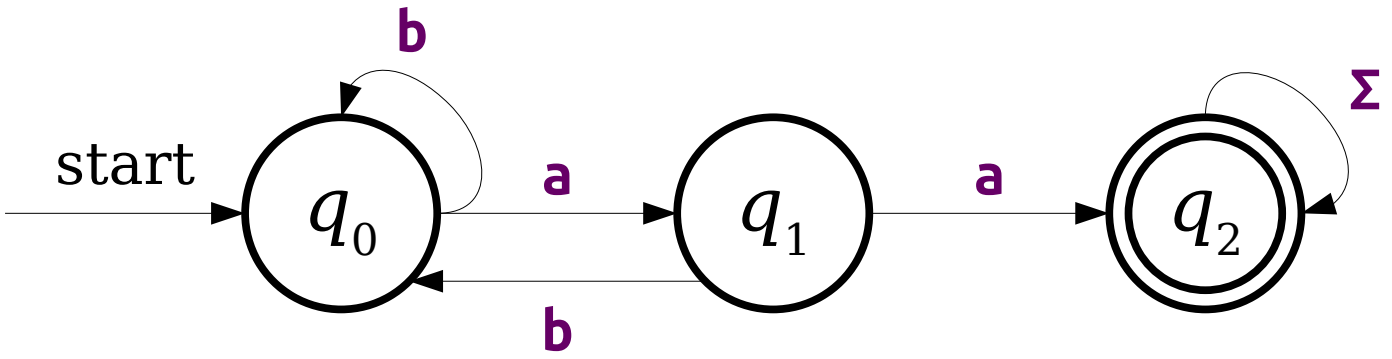
Recognizing Languages with DFAs

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



Recognizing Languages with DFAs

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



Time-Out For Announcements!

Revise-and-Resubmit

- We will begin accepting revisions to midterm problems tomorrow at 2:30PM. The deadline to submit is Monday at 2:30PM.
- Feel free to resubmit answers to any problem where you didn't earn all the points. We'll grade the new submission in place of your old one.
  - Pro tip: Gradescope supports LaTeX formatting, but you need to use `$$two dollar signs$$` to enter math mode.
- Exam statistics are up on EdStem if you're curious, though we expect a big jump once revise-and-resubmit closes.

Your Questions

“Advice on managing disappointment, re: midterm? Working this hard for grades like these can make one feel beyond helping. What do you recommend?”

For starters, I'm sorry to hear this! That feeling is not fun at all, and basically everyone encounters it at some point when learning CS. Hang in there – it's going to be okay.

I want to push back against the "beyond helping" comment. I've seen people make dramatic turnarounds in their CS103 trajectories.

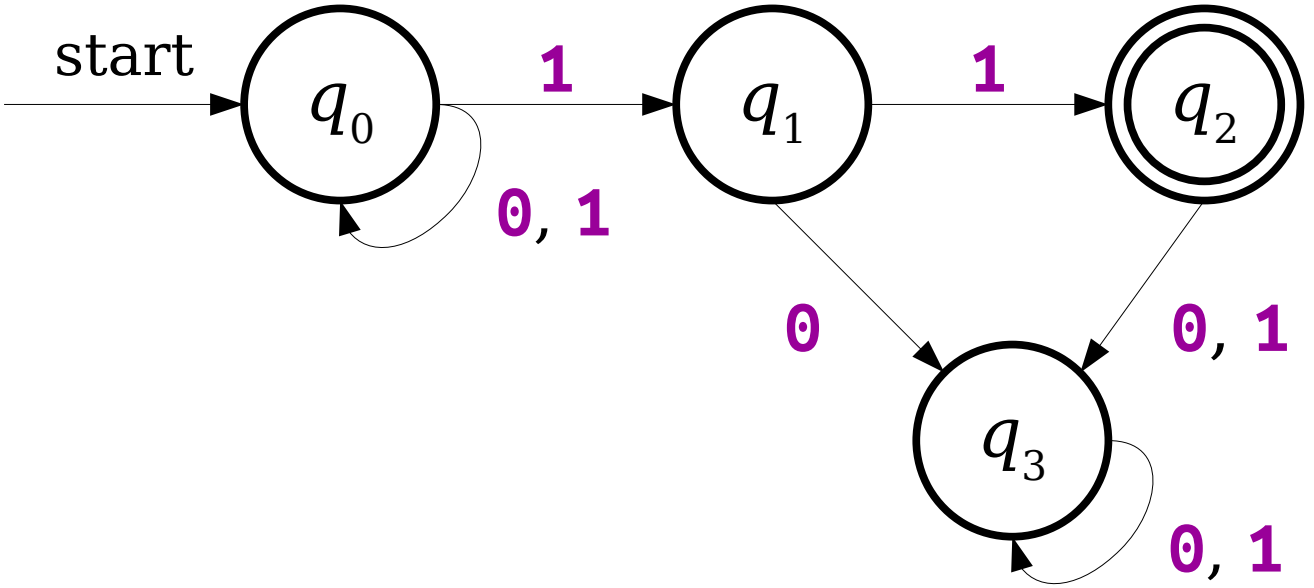
Also, remember to take stock of what you've learned so far. How would you have done on the midterm if you took it on Day 1 of the quarter? You're rapidly absorbing a totally new way of thinking about things. It can feel overwhelming at times, but you are growing and expanding your mind. Don't lose sight of that.

Finally, feel free to come talk to us about where you're spending your time in this course. We are happy to talk strategy and see if we can find a way to improve outcomes while reducing time investment.

Back to CS103!

NFAs

The Motivation



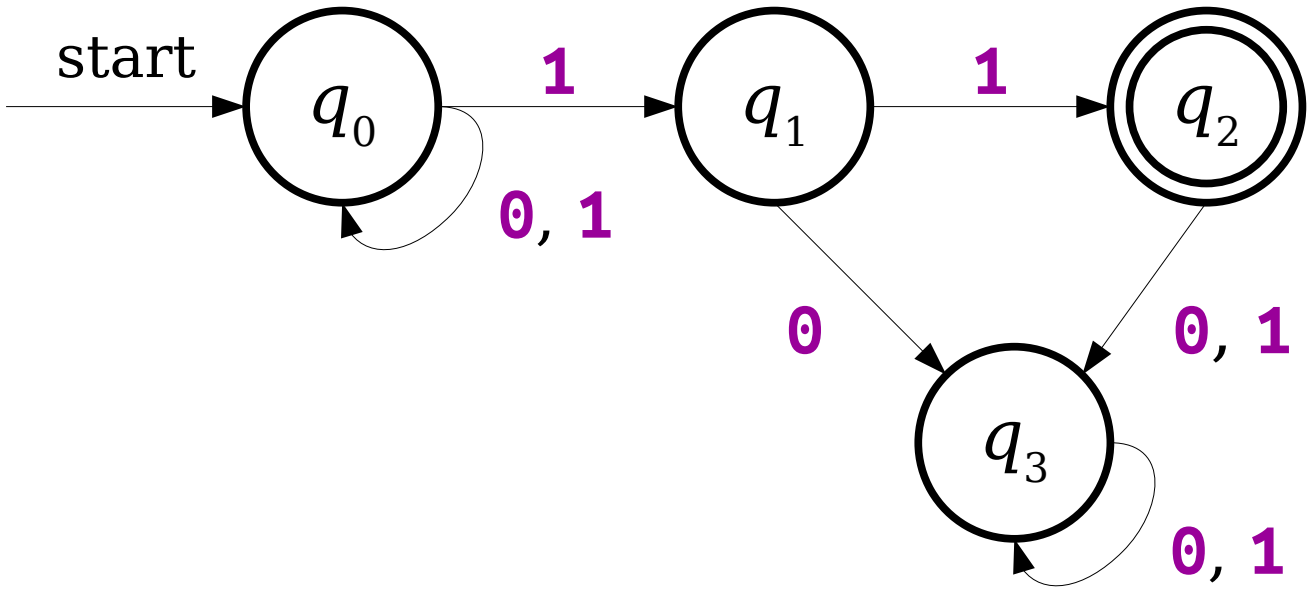
NFAs

- An ***NFA*** is a
  - ***N***ondeterministic
  - ***F***inite
  - ***A***utomaton
- Structurally similar to a DFA, but represents a fundamental shift in how we'll think about computation.

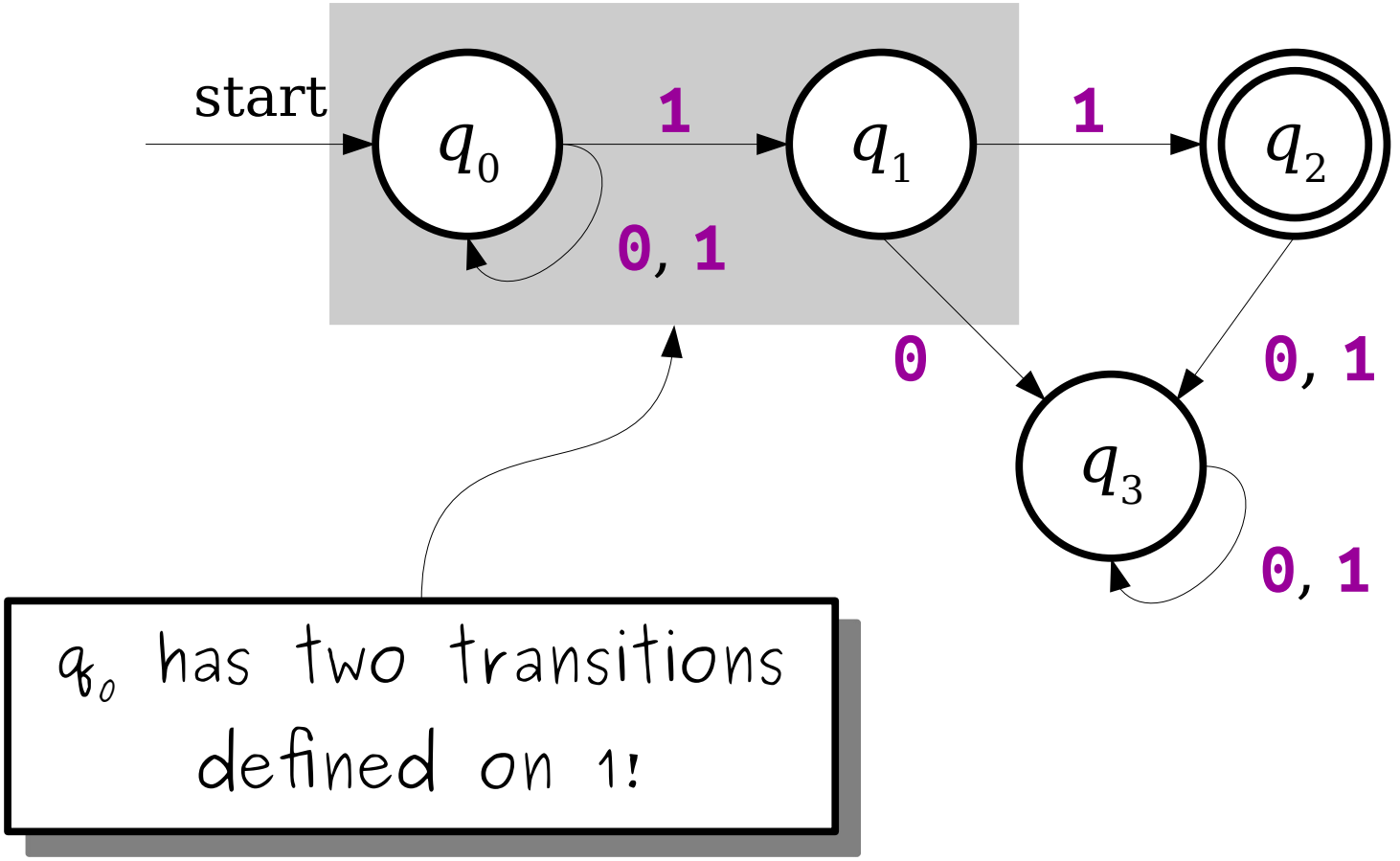
(Non)determinism

- A model of computation is ***deterministic*** if at every point in the computation, there is exactly one choice that can make.
  - The machine accepts if that series of choices leads to an accepting state.
- A model of computation is ***nondeterministic*** if the computing machine has a finite number of choices available to make at each point, possibly including zero.
- The machine accepts if ***any*** series of choices leads to an accepting state.
  - (This sort of nondeterminism is technically called ***existential nondeterminism***, the most philosophical-sounding term we'll introduce all quarter.)

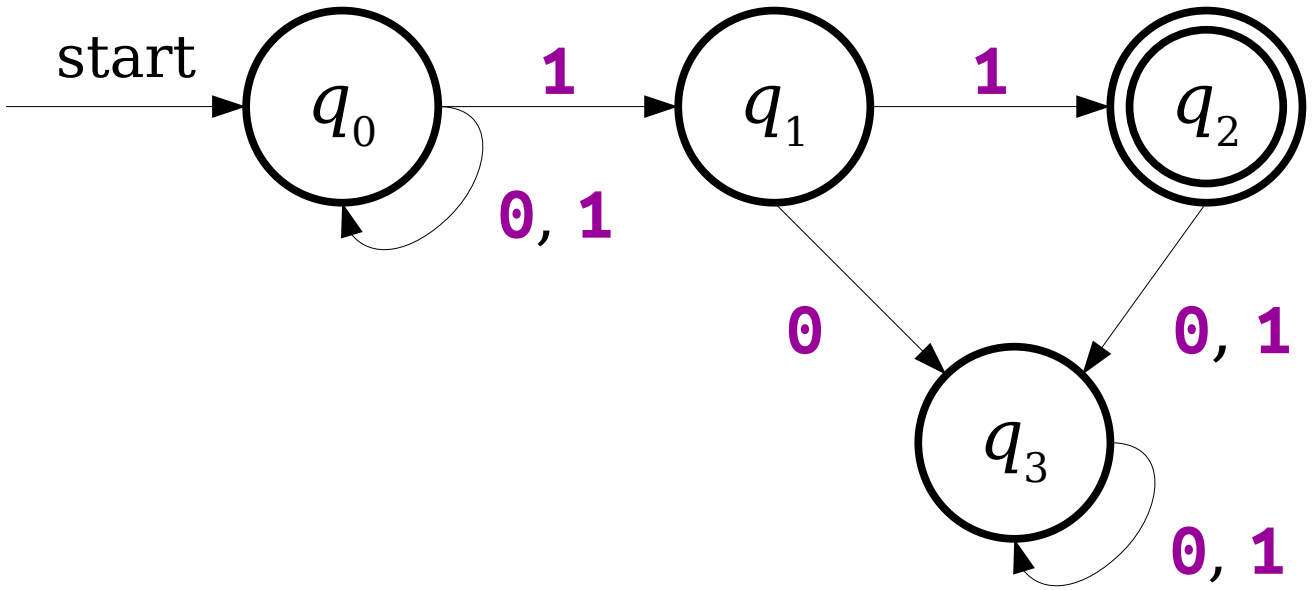
A Simple NFA



A Simple NFA

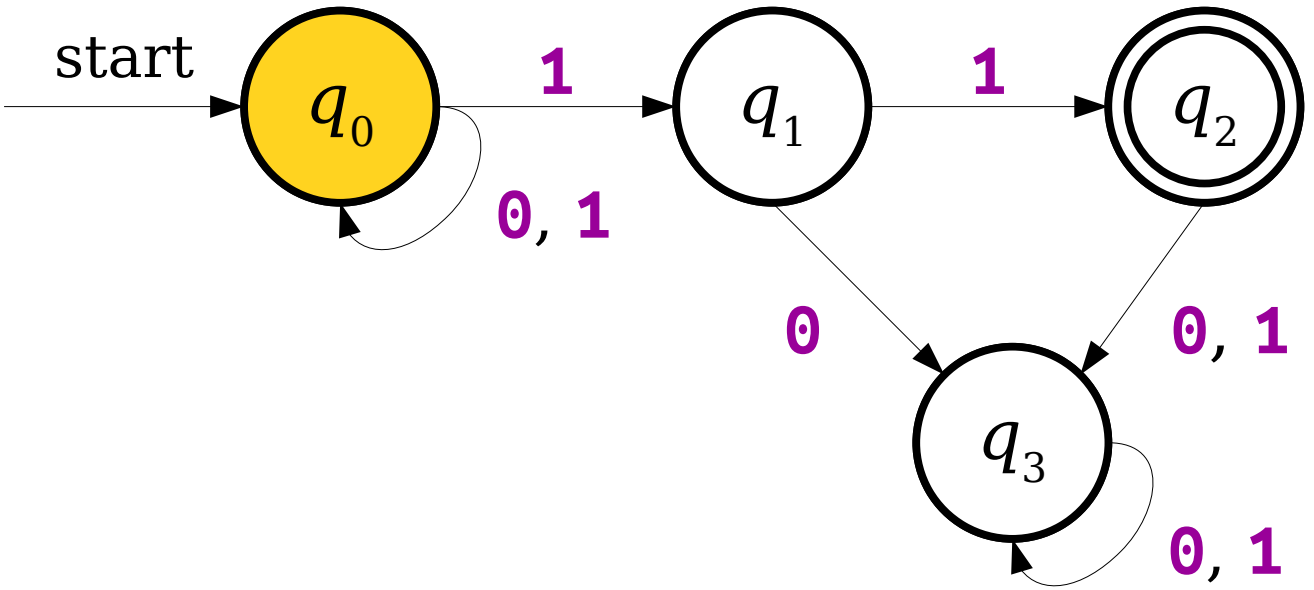

start
q_0
q_1
q_2
q_3
1
1
0, 1
0
0, 1
0, 1
q_0 has two transitions defined on 1!

A Simple NFA



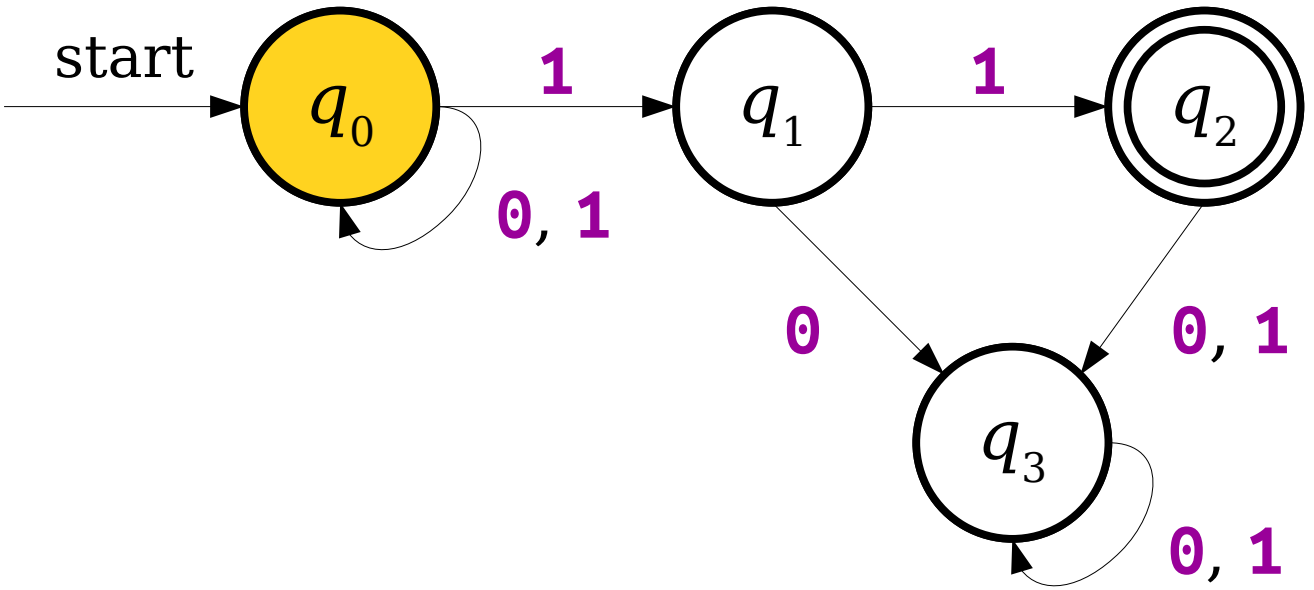
0	1	0	1	1

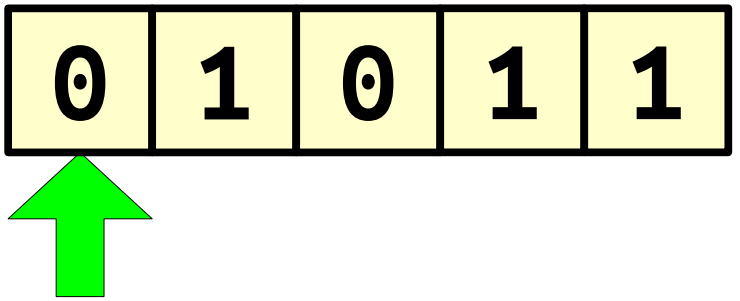
A Simple NFA



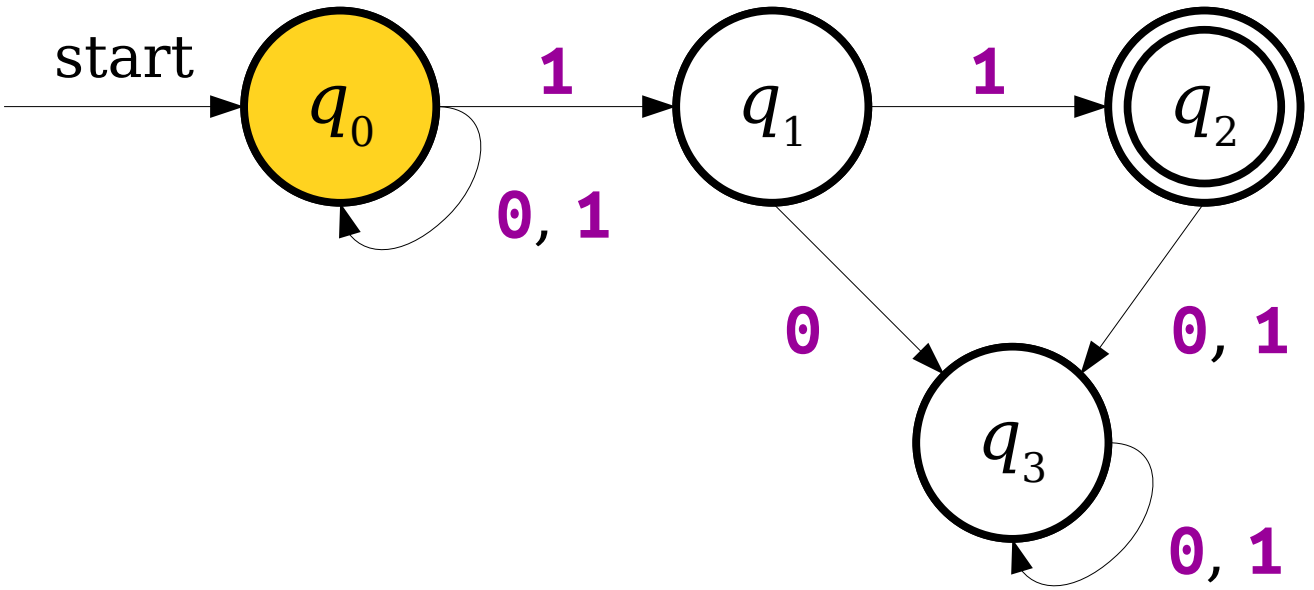
0	1	0	1	1

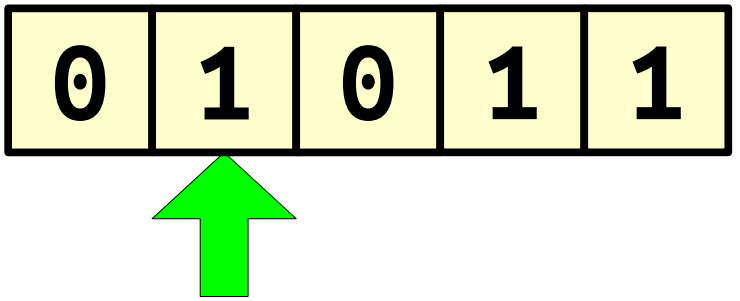
A Simple NFA



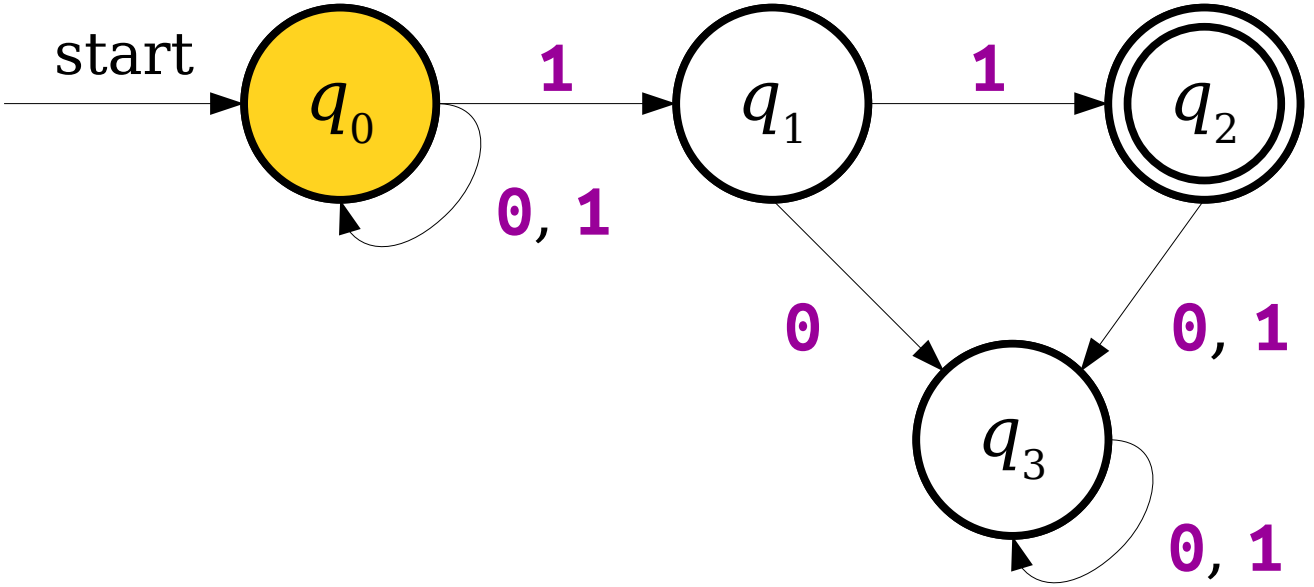


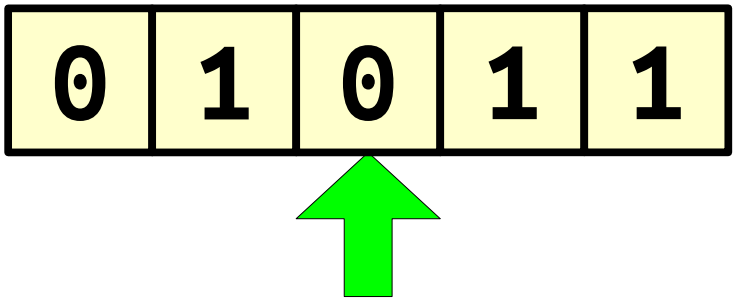
A Simple NFA



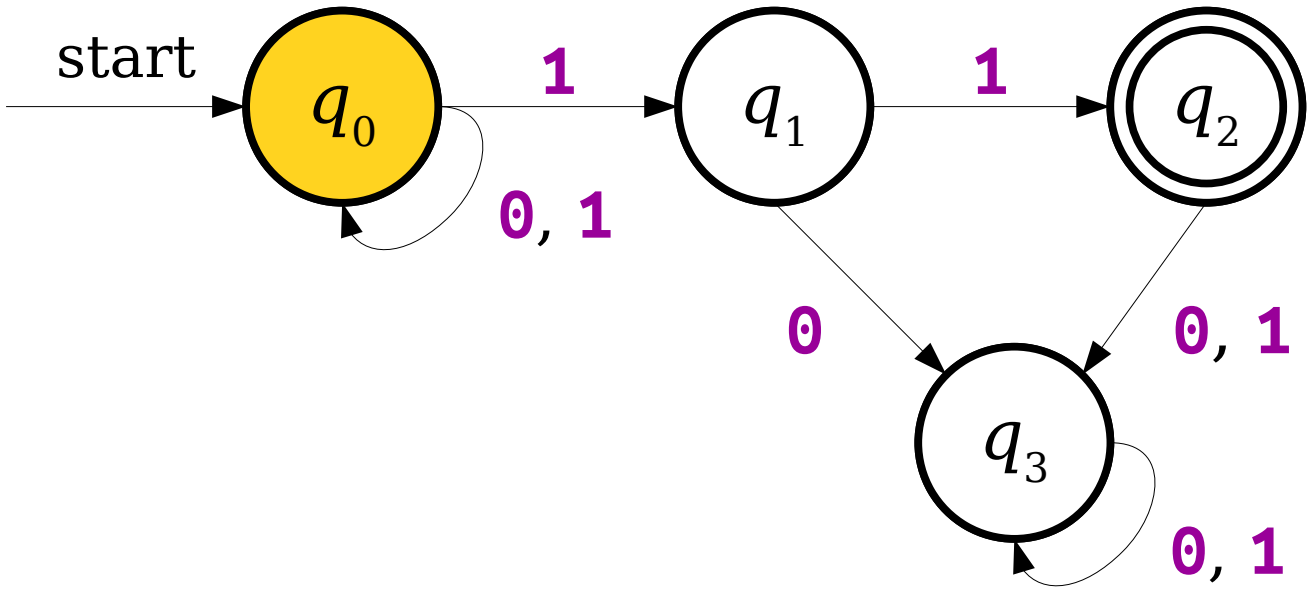


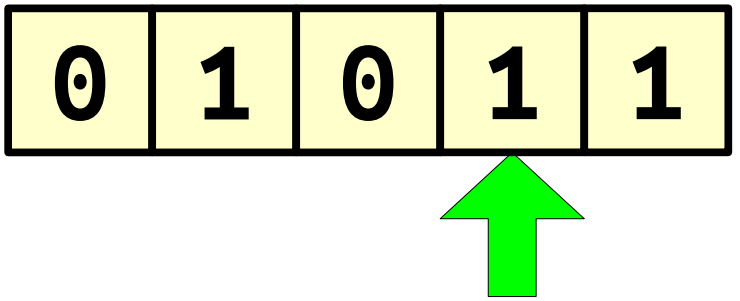
A Simple NFA



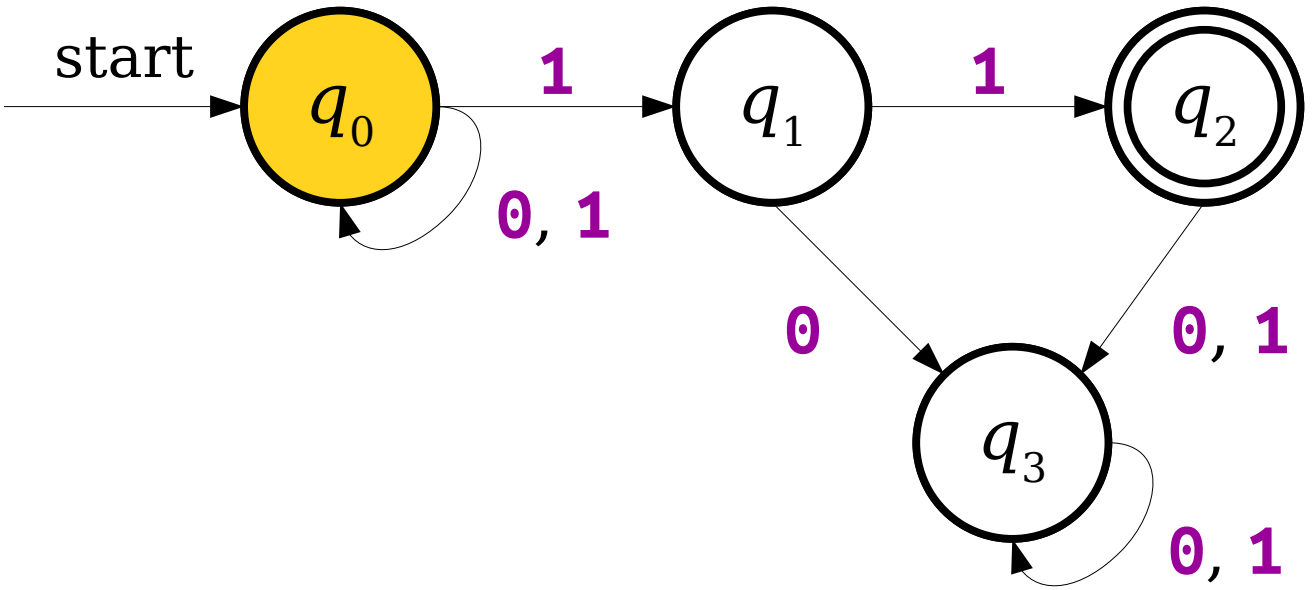


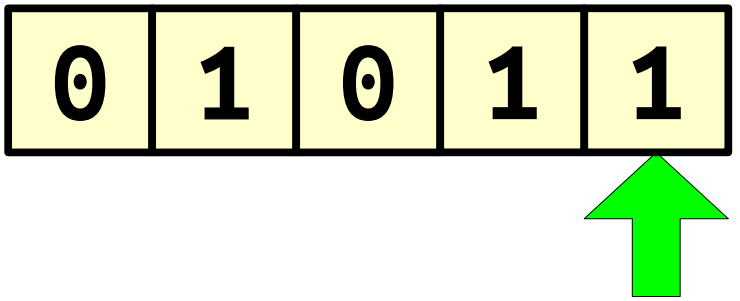
A Simple NFA



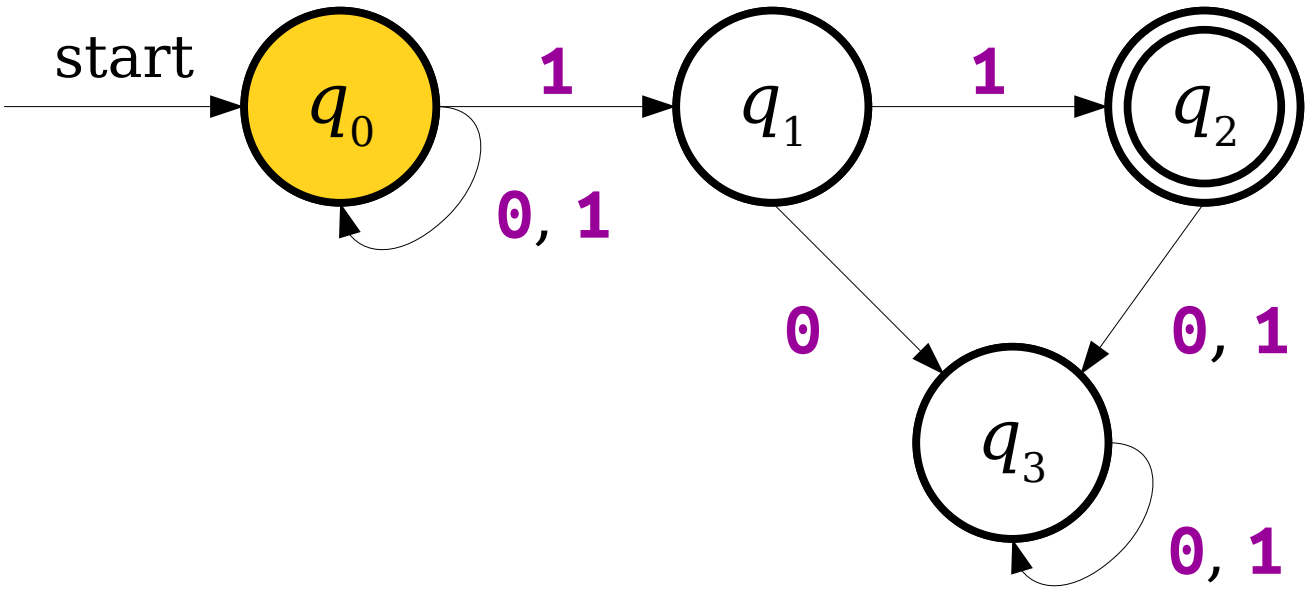


A Simple NFA



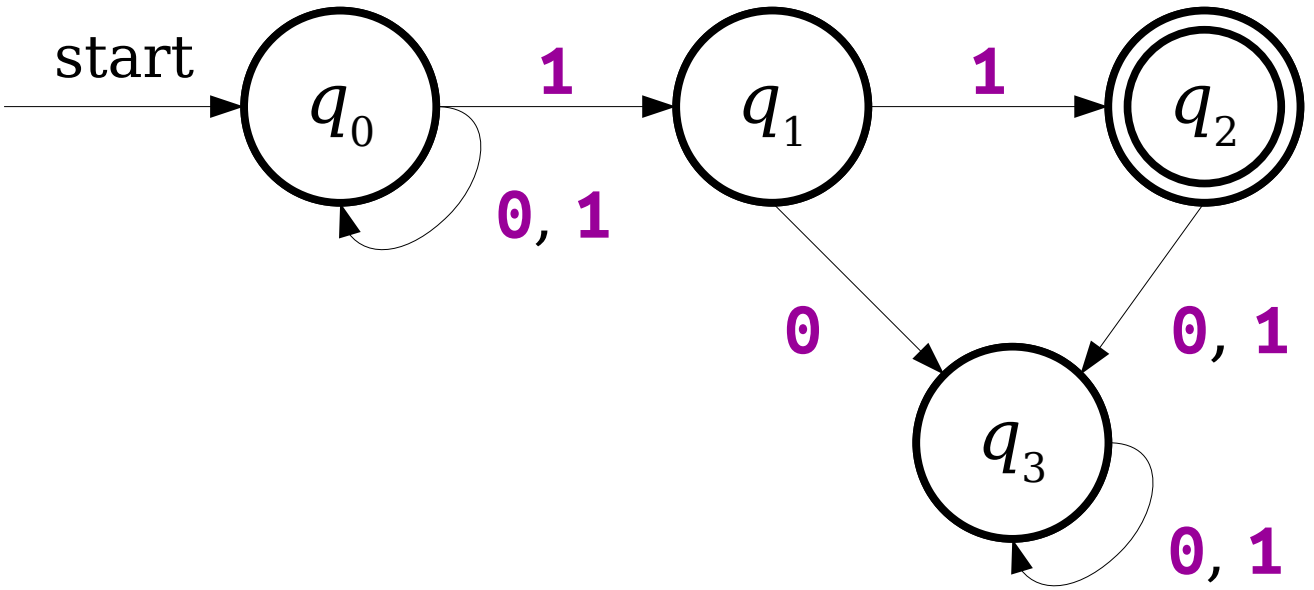


A Simple NFA



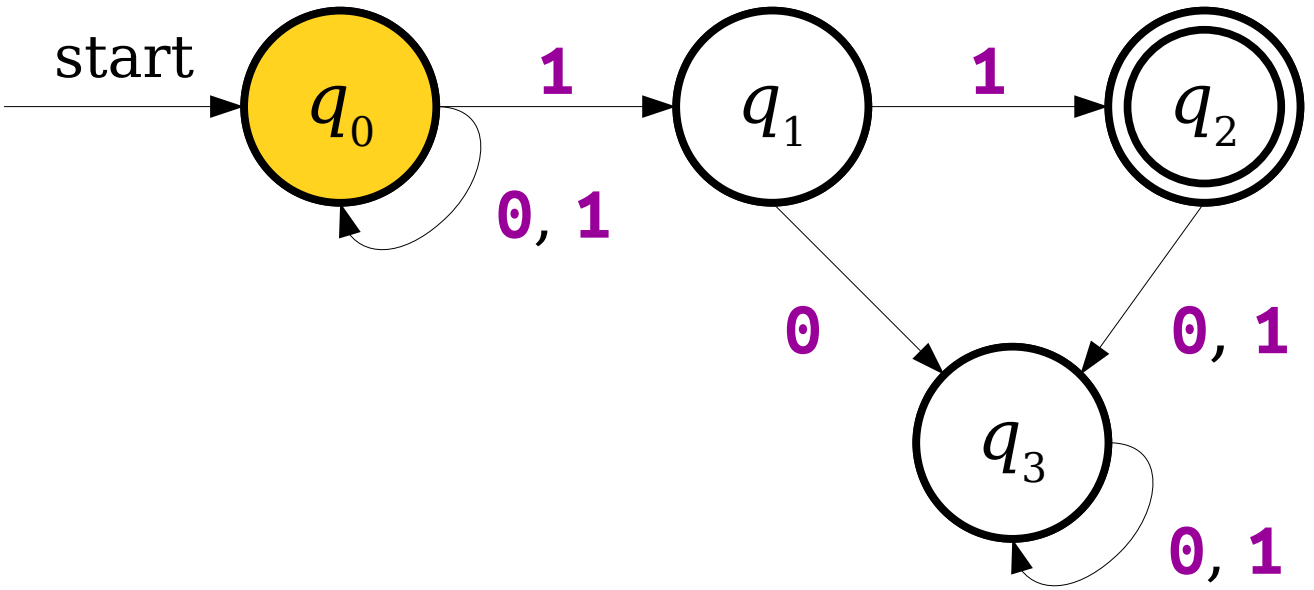
0	1	0	1	1

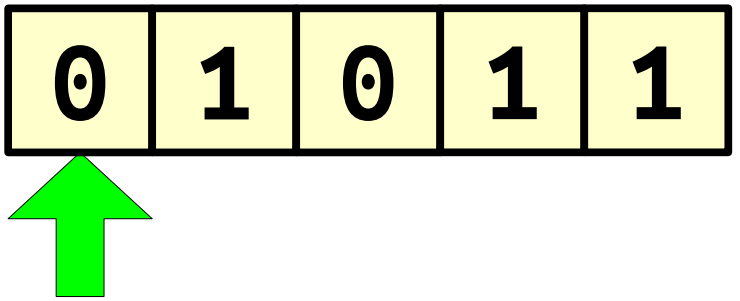
A Simple NFA



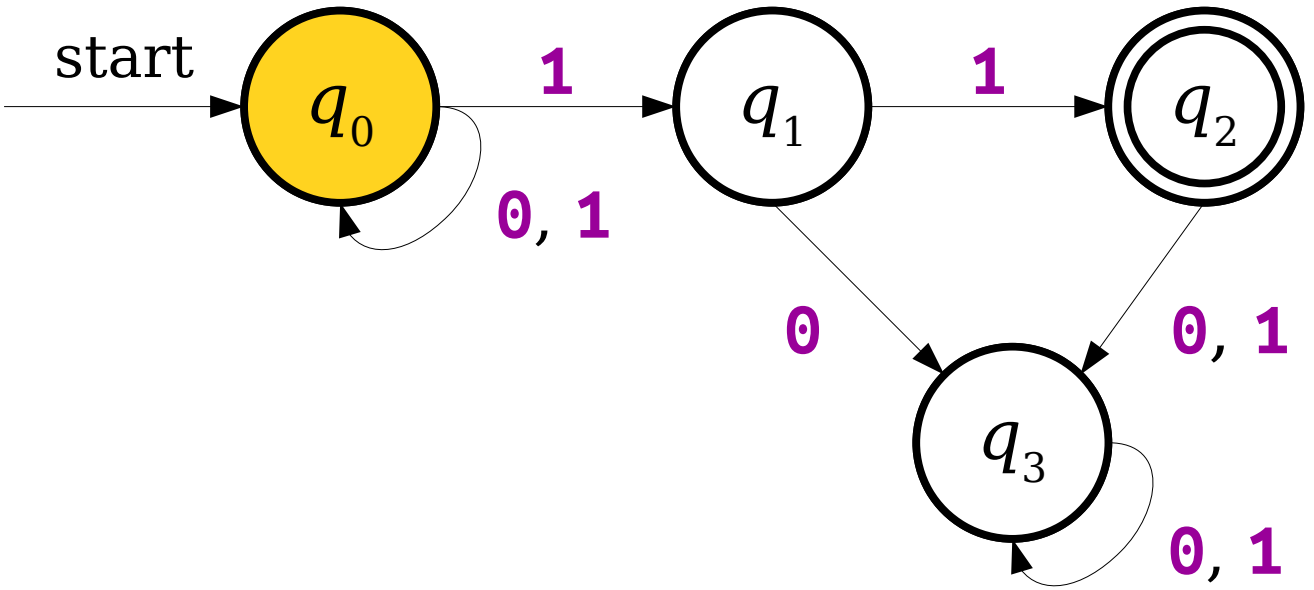
0	1	0	1	1

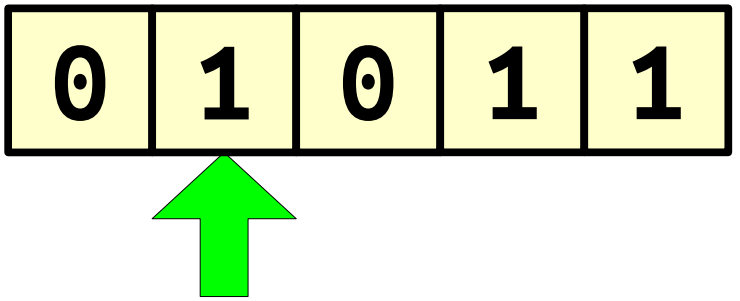
A Simple NFA



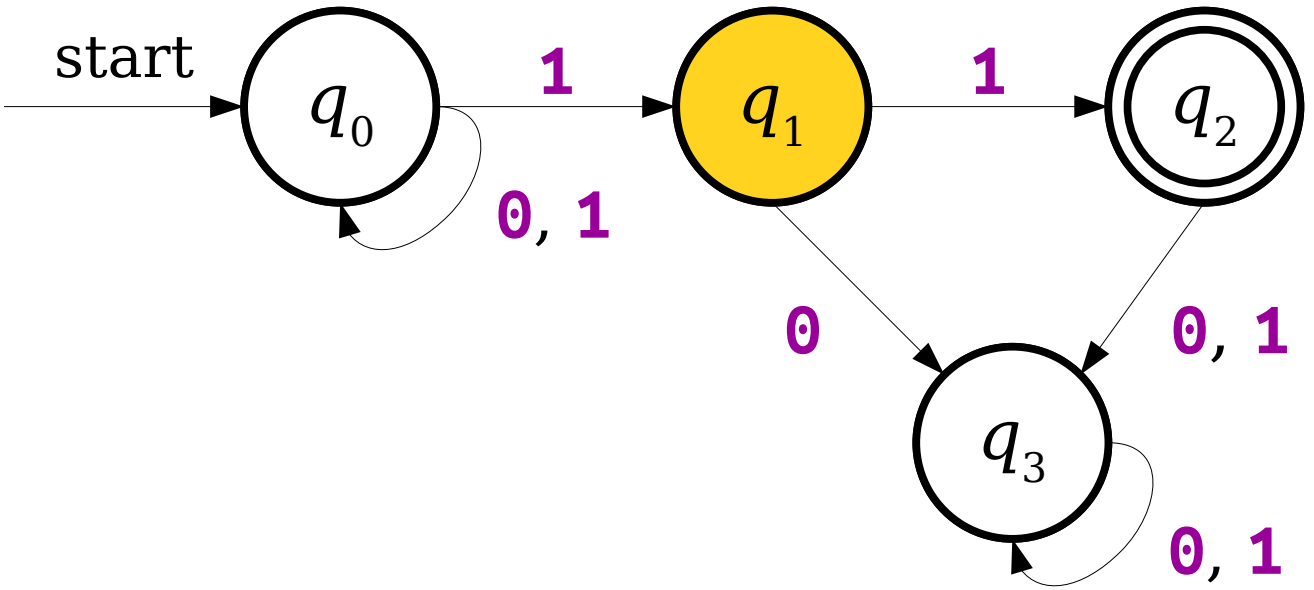


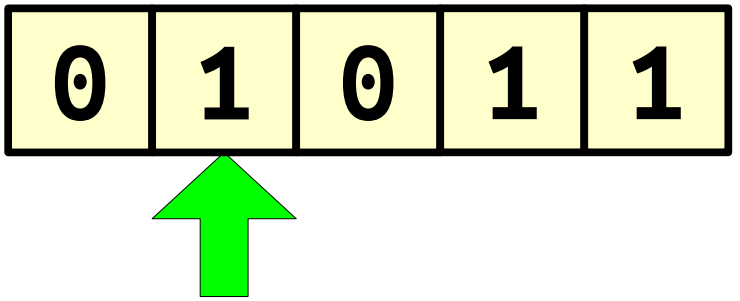
A Simple NFA



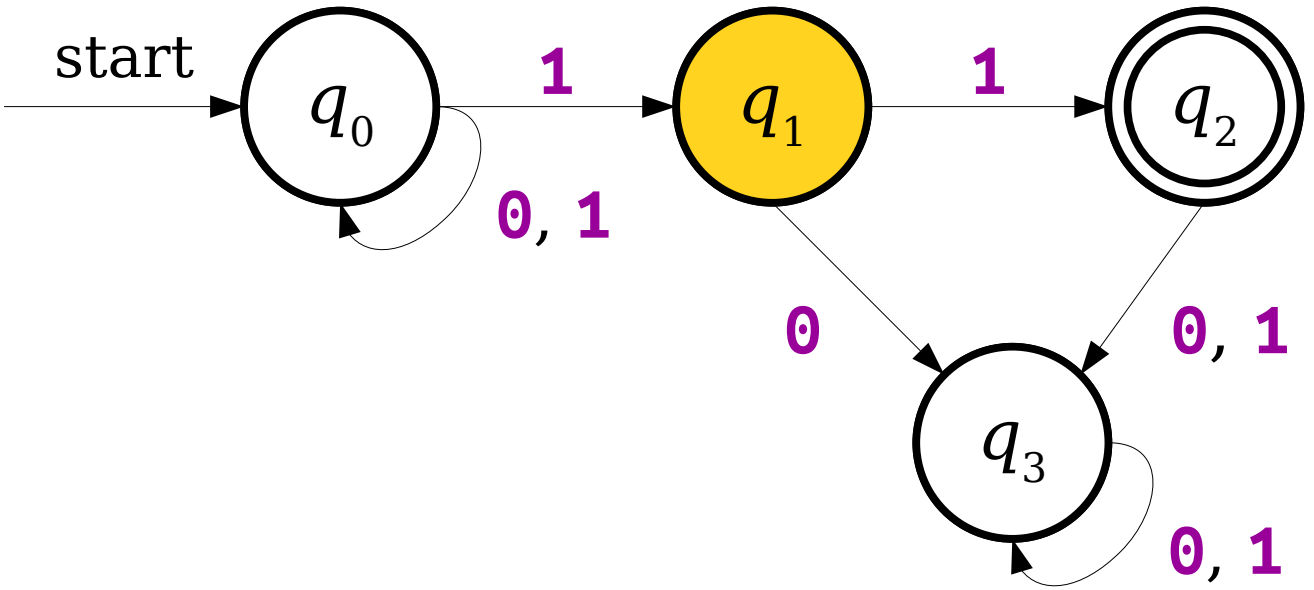


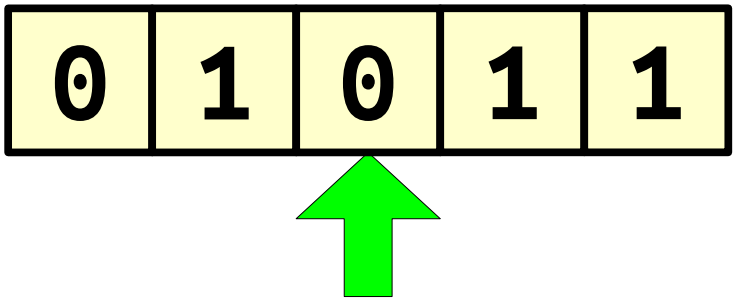
A Simple NFA



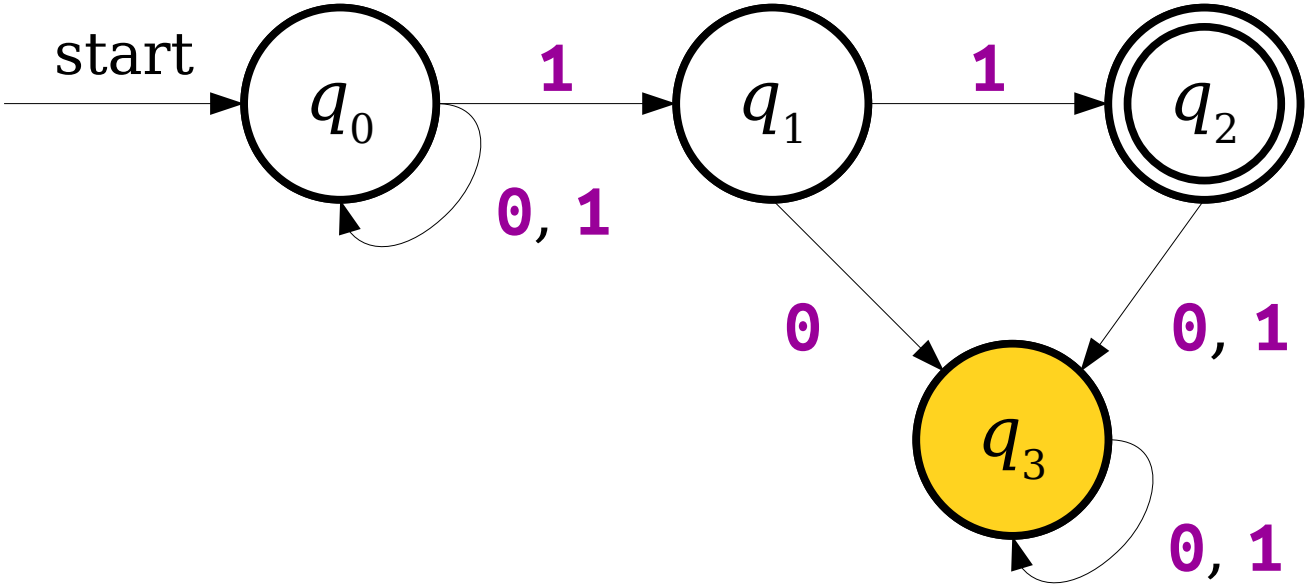


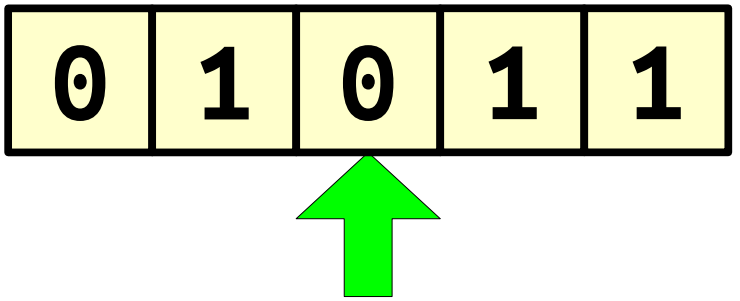
A Simple NFA



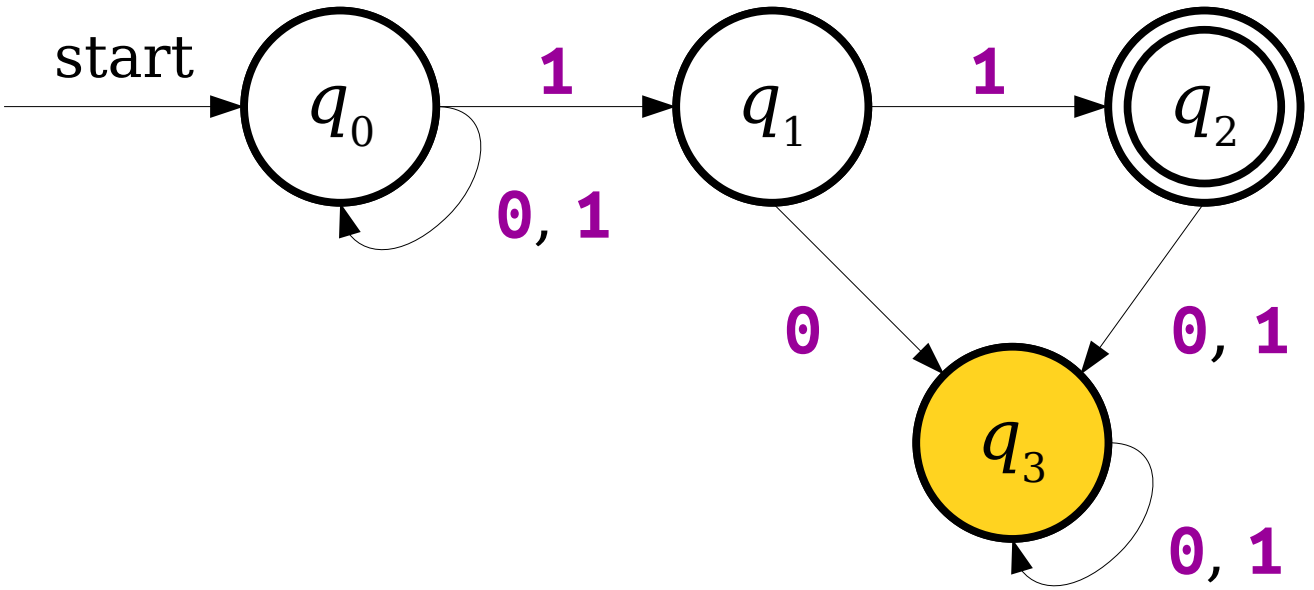


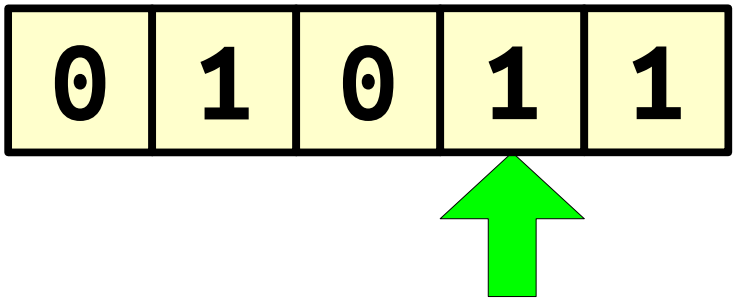
A Simple NFA



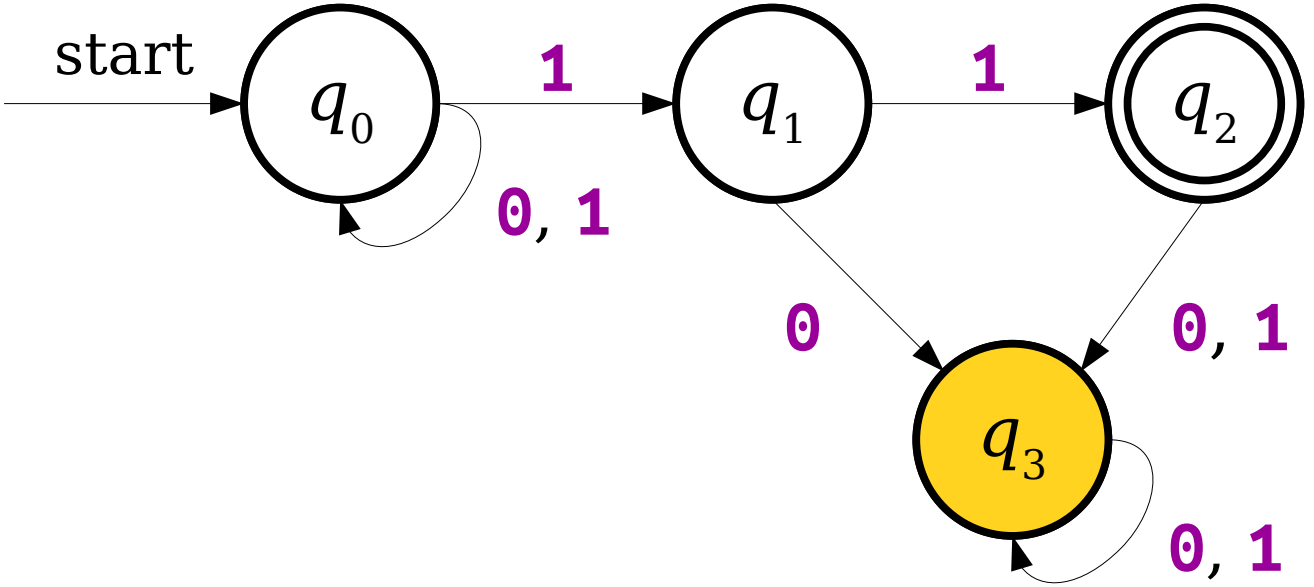


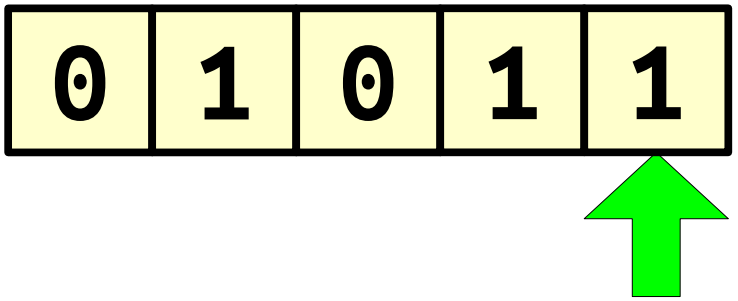
A Simple NFA



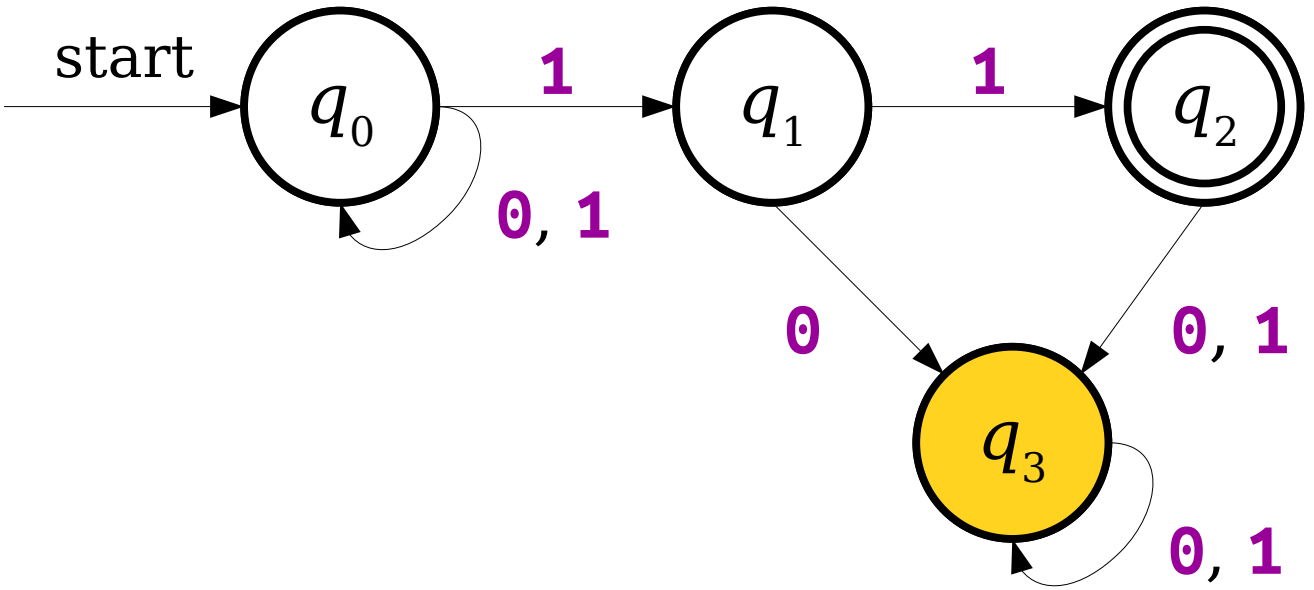


A Simple NFA



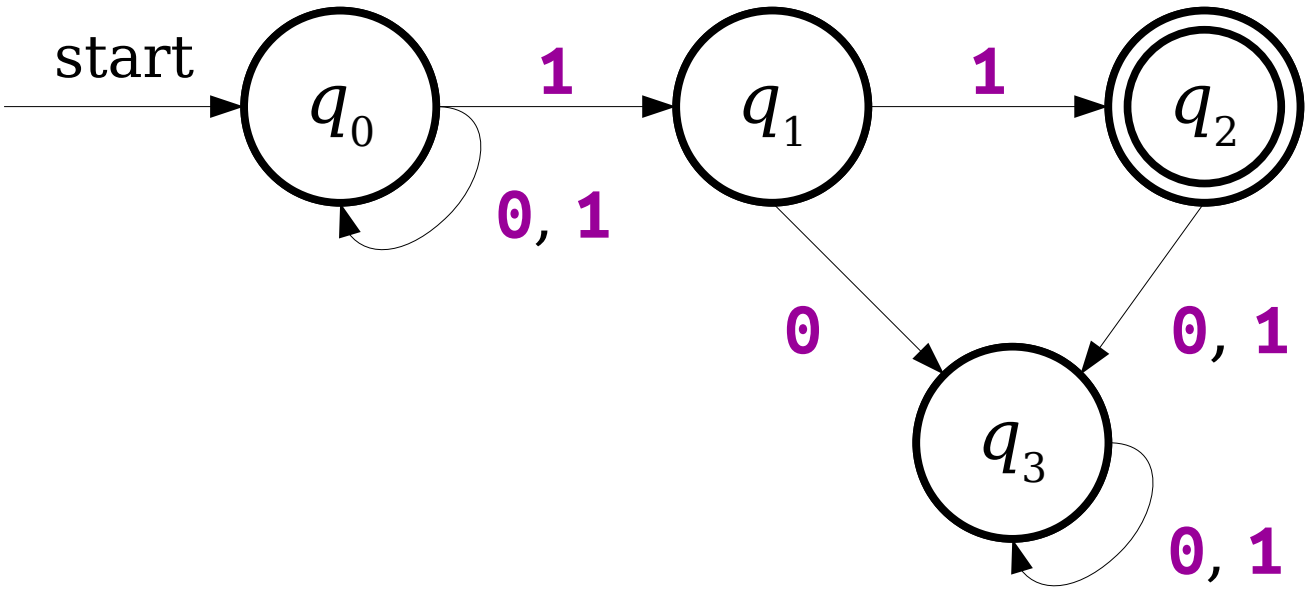


A Simple NFA



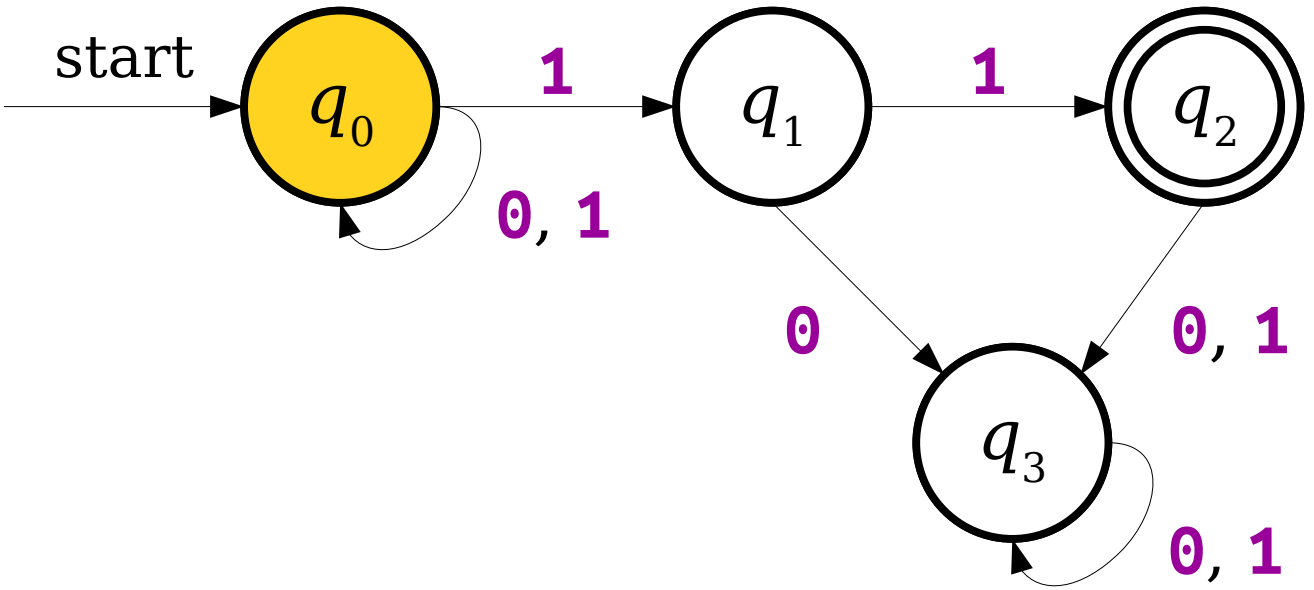
0	1	0	1	1

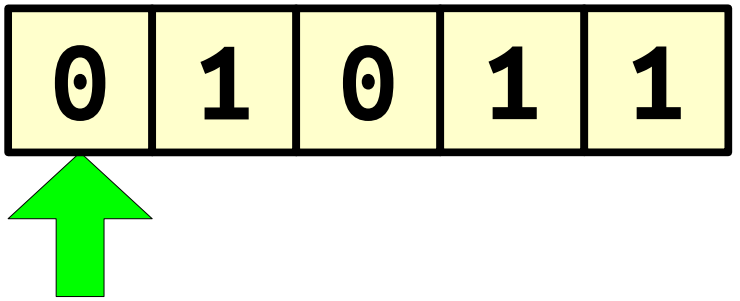
A Simple NFA



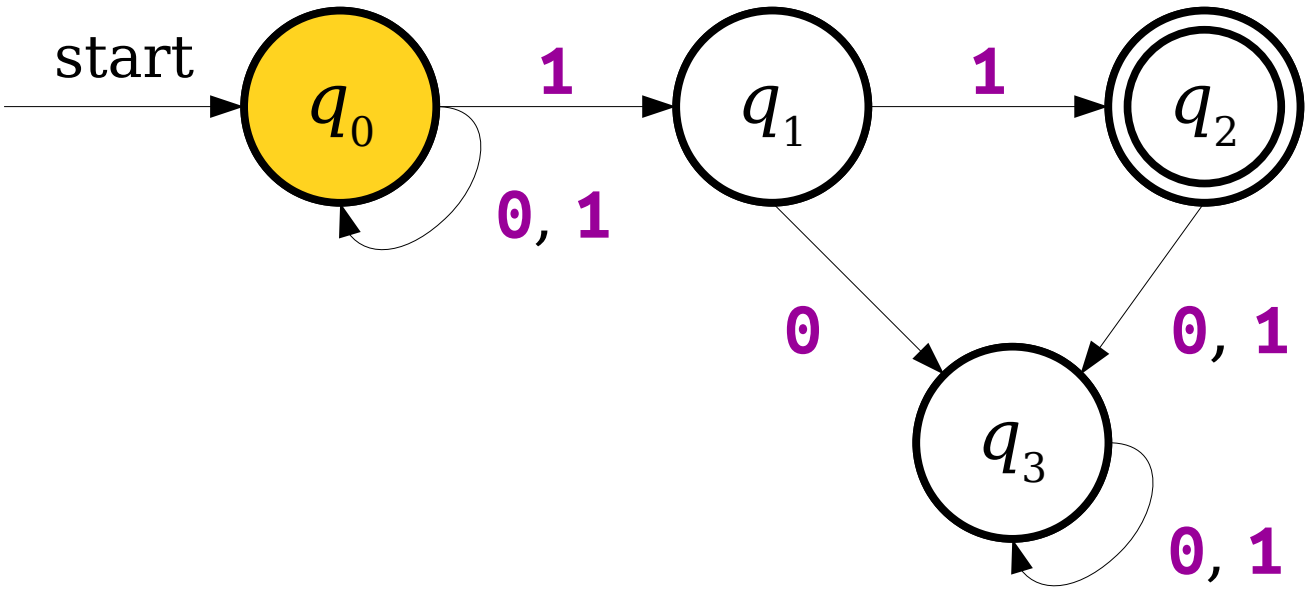
0	1	0	1	1

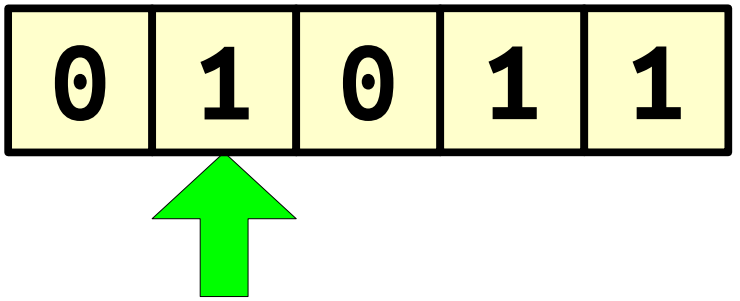
A Simple NFA



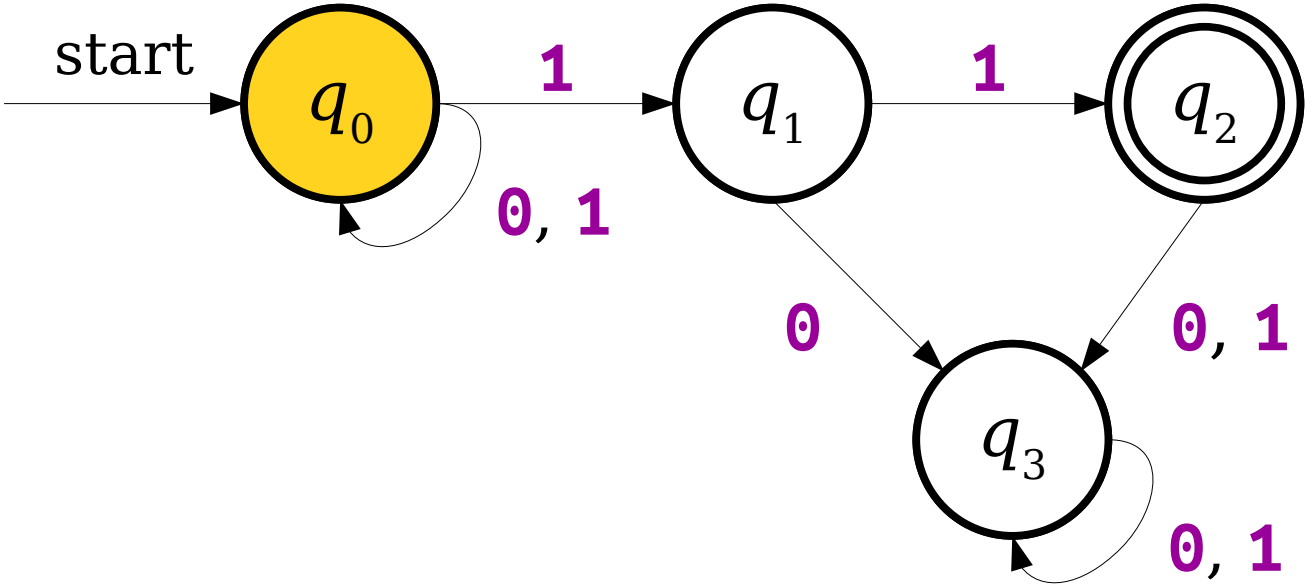


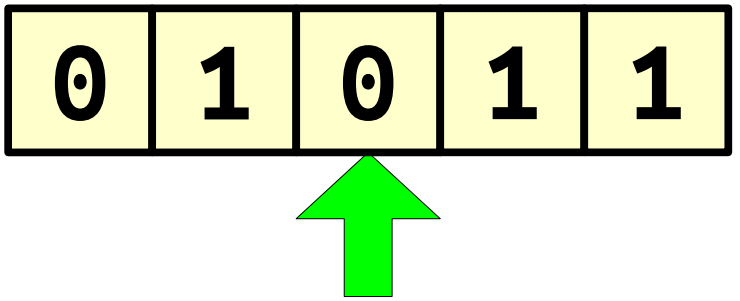
A Simple NFA



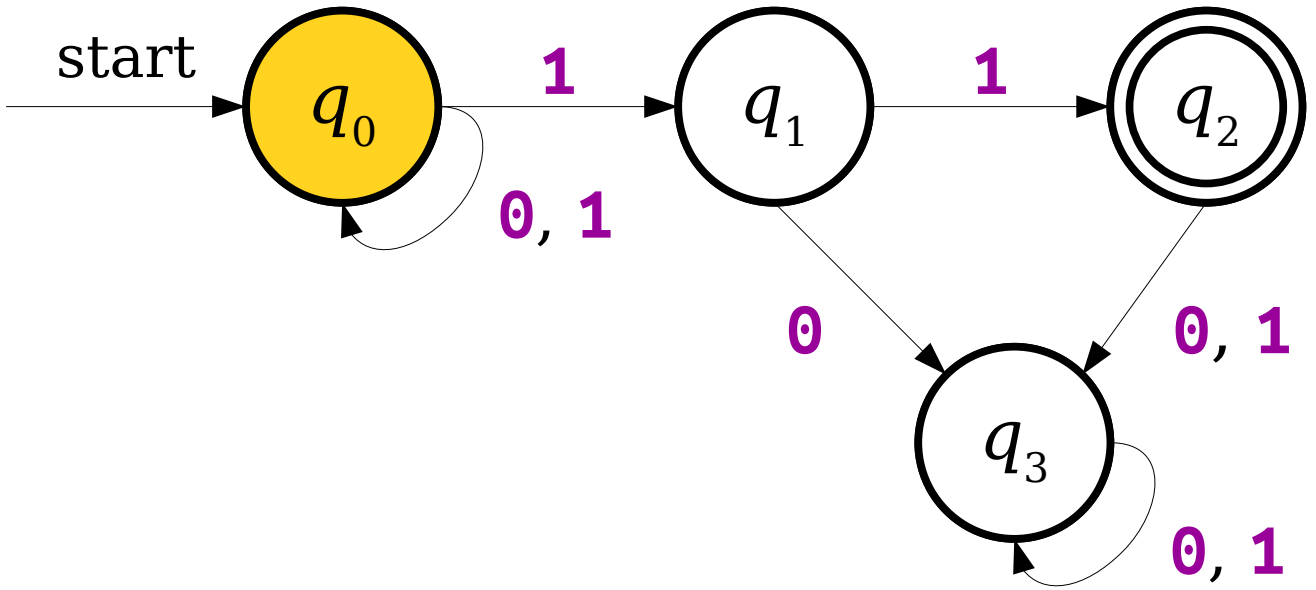


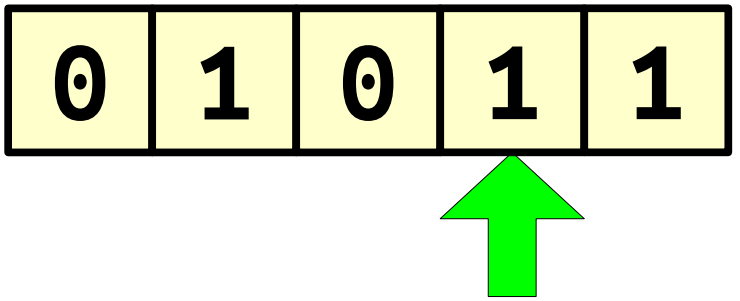
A Simple NFA



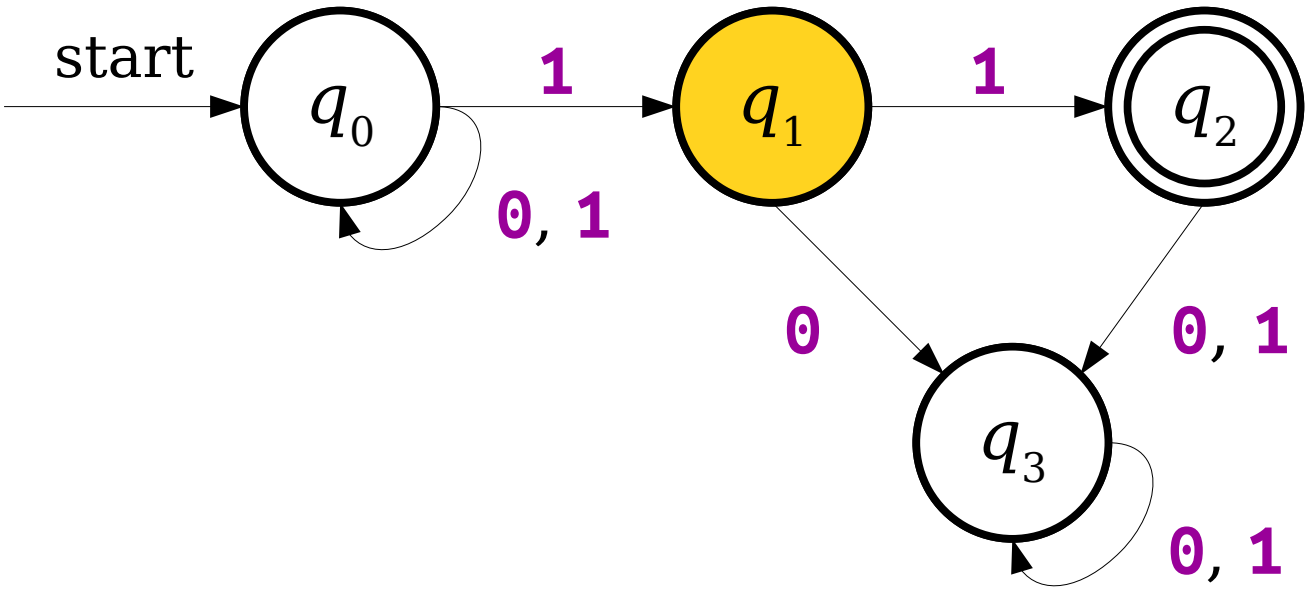


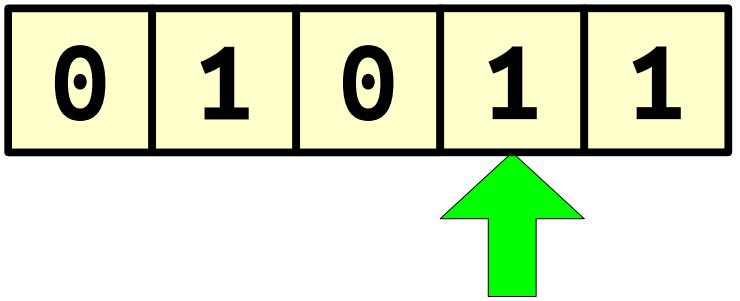
A Simple NFA



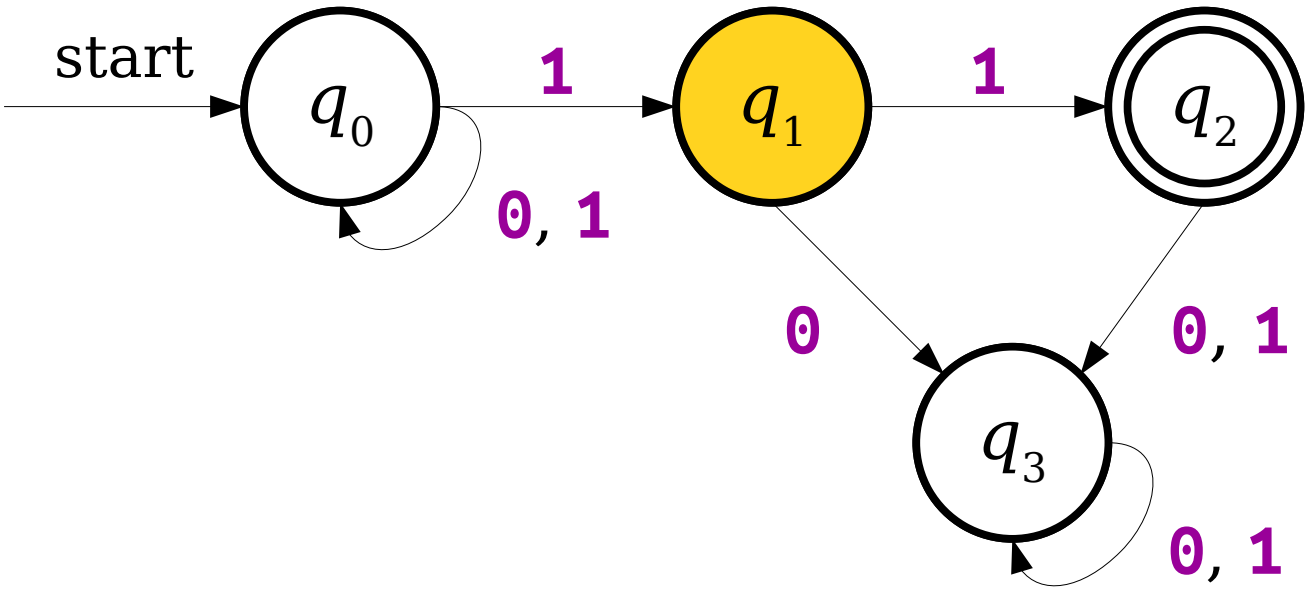


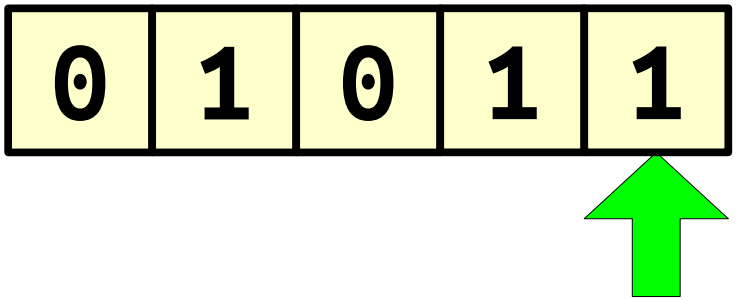
A Simple NFA



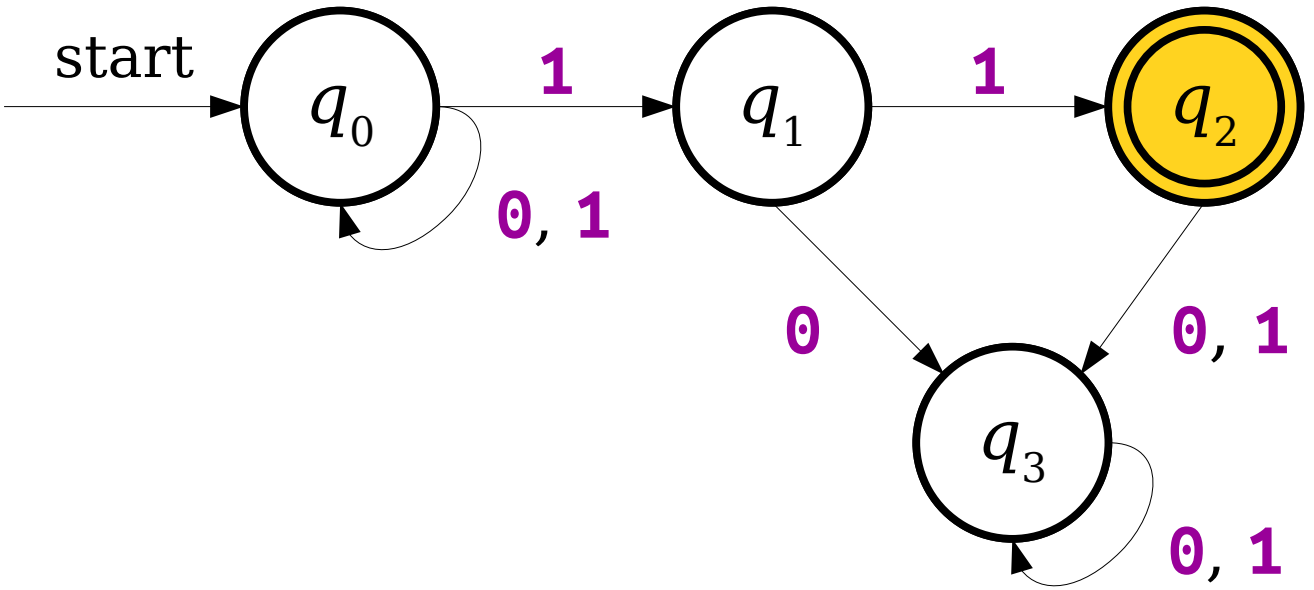


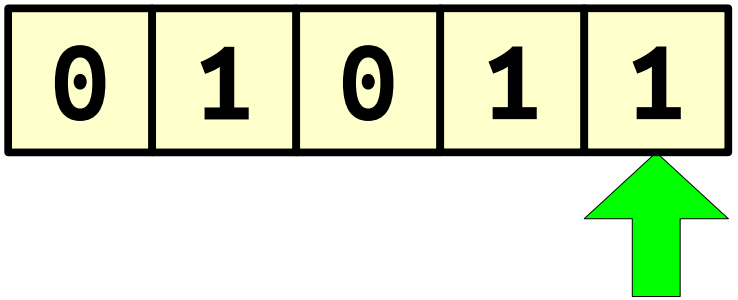
A Simple NFA



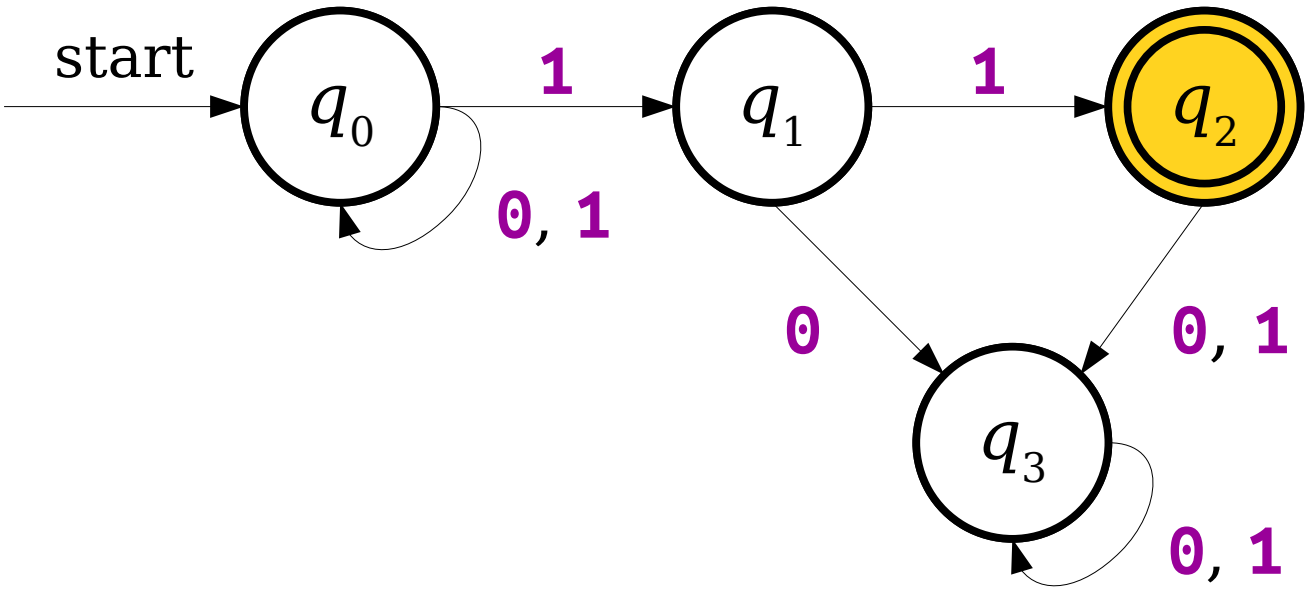


A Simple NFA



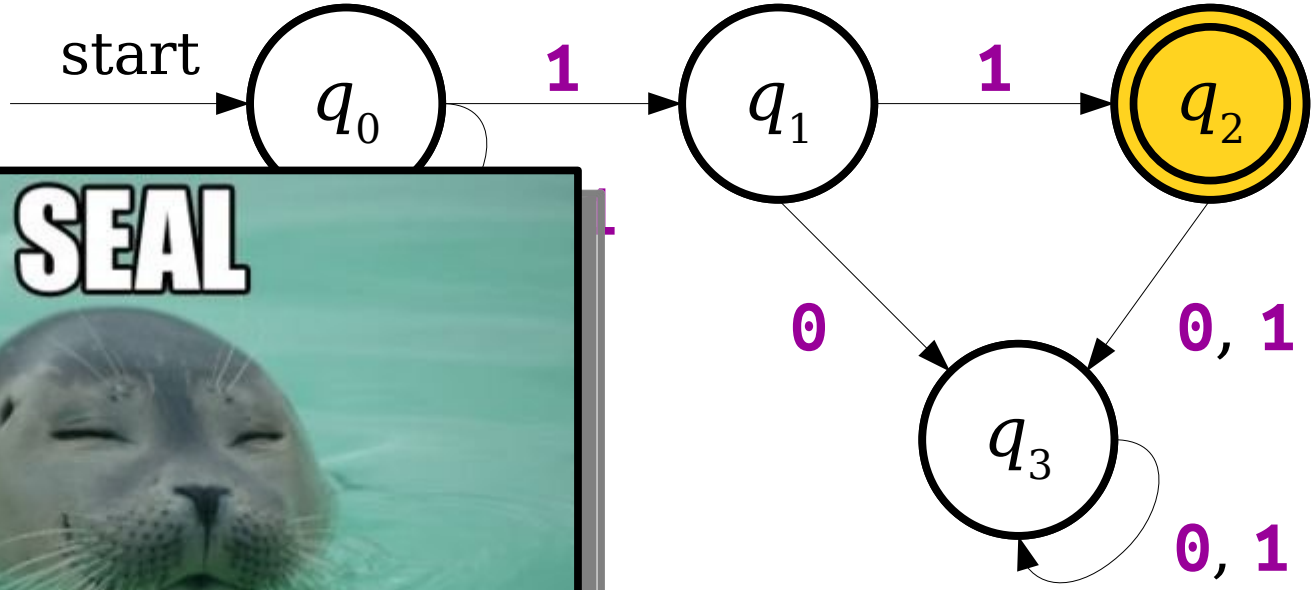


A Simple NFA



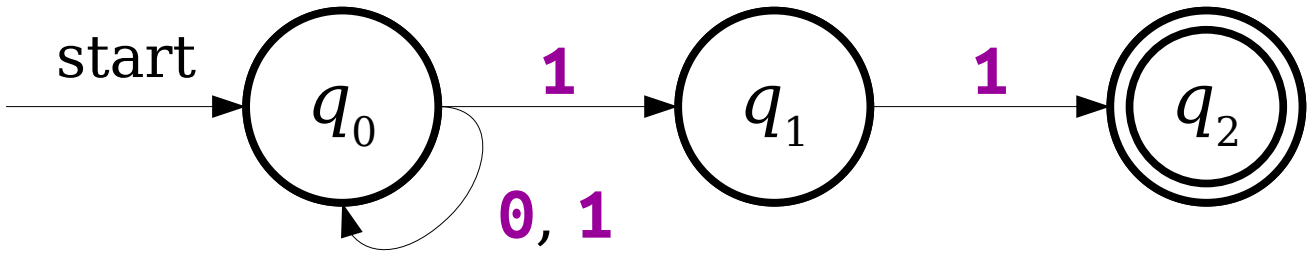
0	1	0	1	1

A Simple NFA

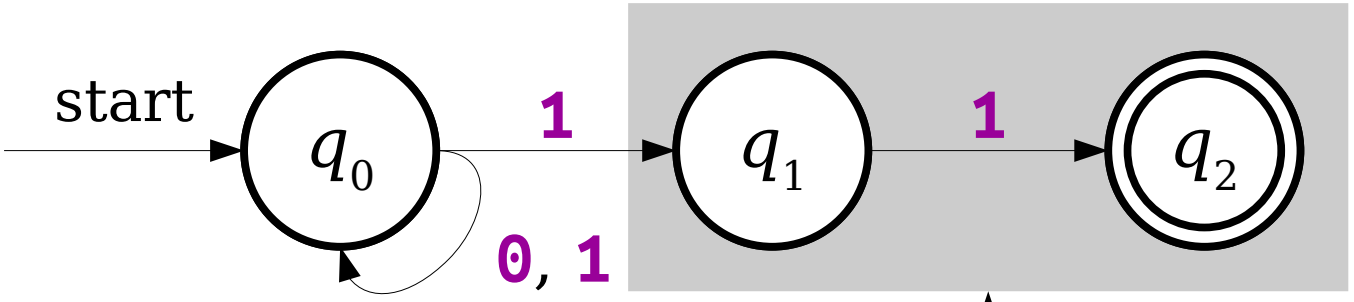


0	1	0	1	1

A More Complex NFA

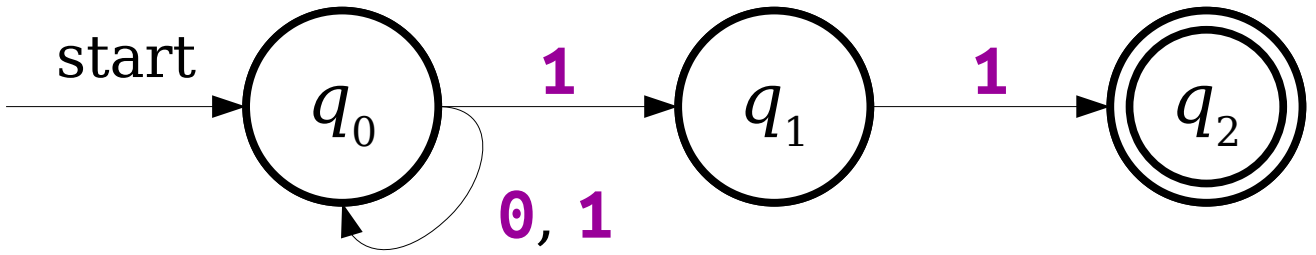


A More Complex NFA



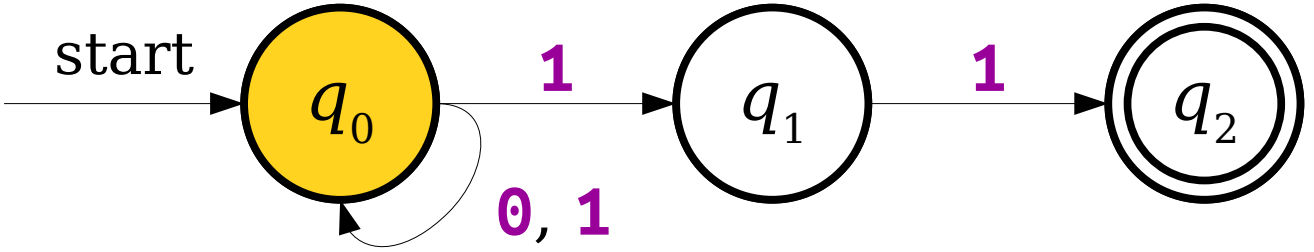
If a NFA needs to make a transition when no transition exists, the automaton dies and that particular path does not accept.

A More Complex NFA



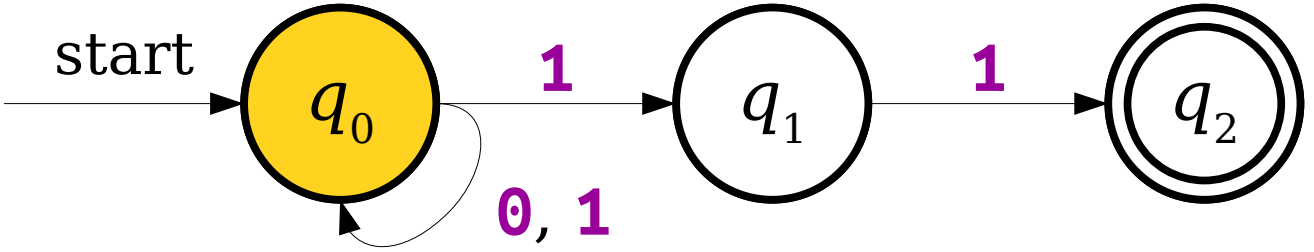
0	1	0	1	1

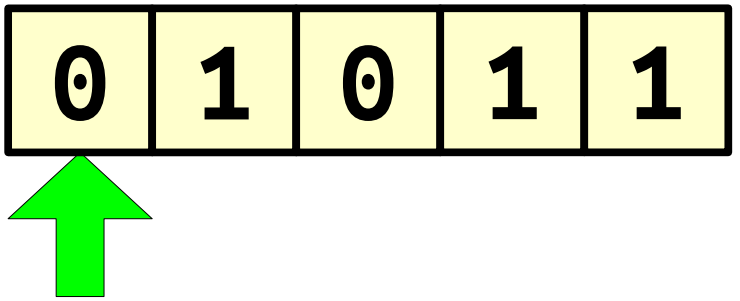
A More Complex NFA



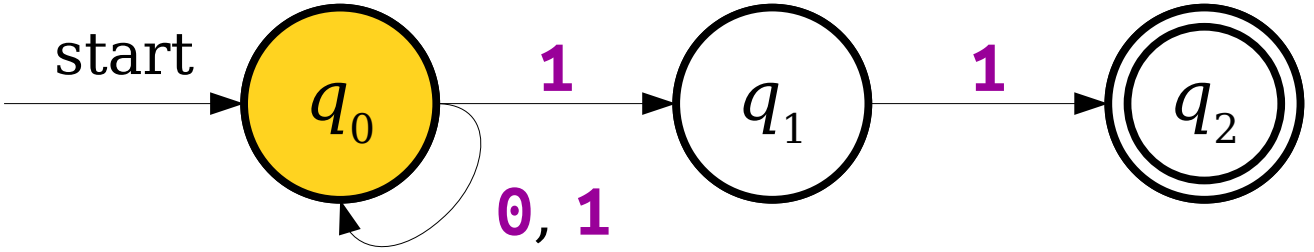
0	1	0	1	1

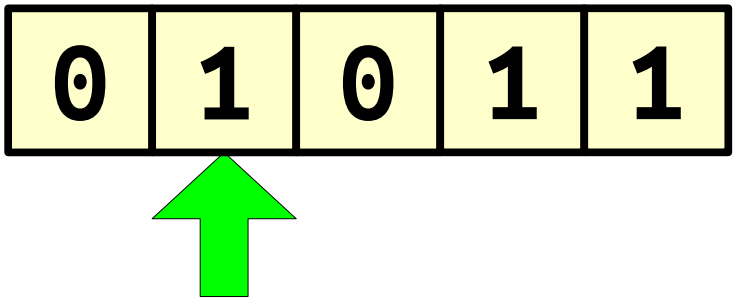
A More Complex NFA



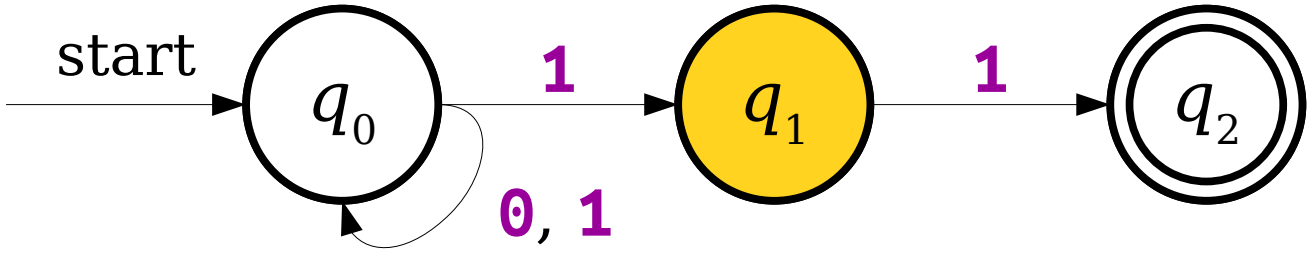


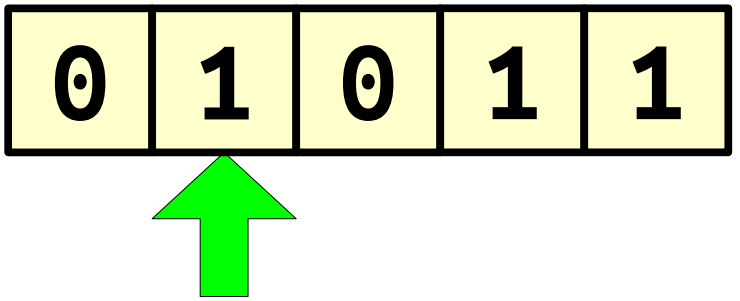
A More Complex NFA



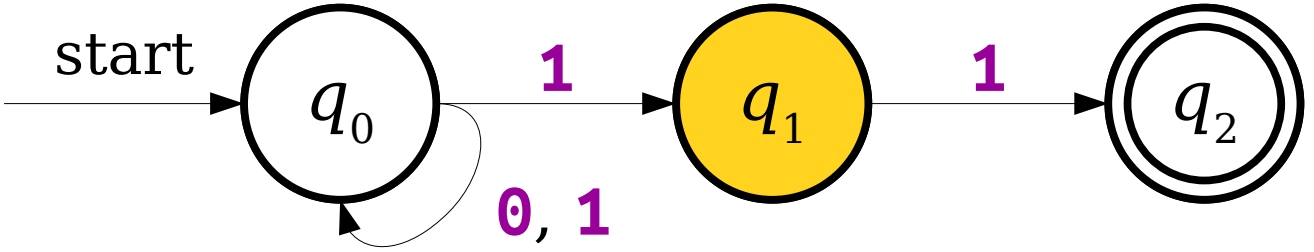


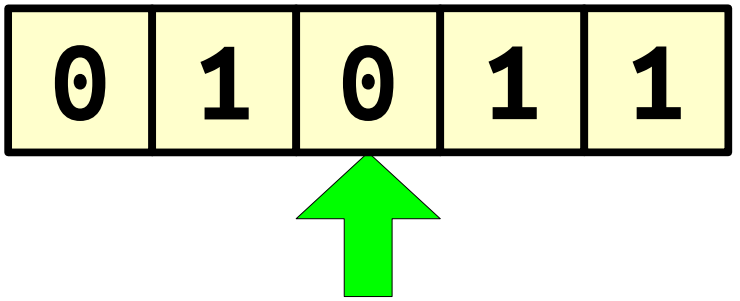
A More Complex NFA



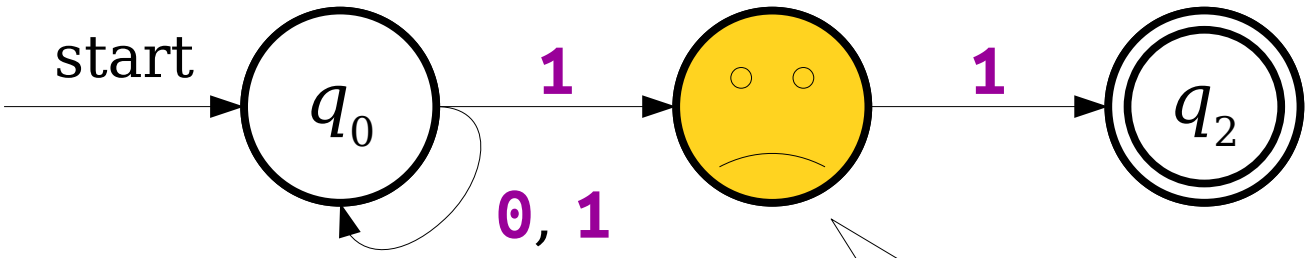


A More Complex NFA

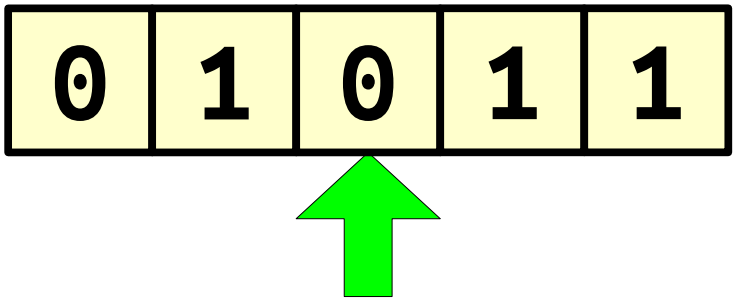




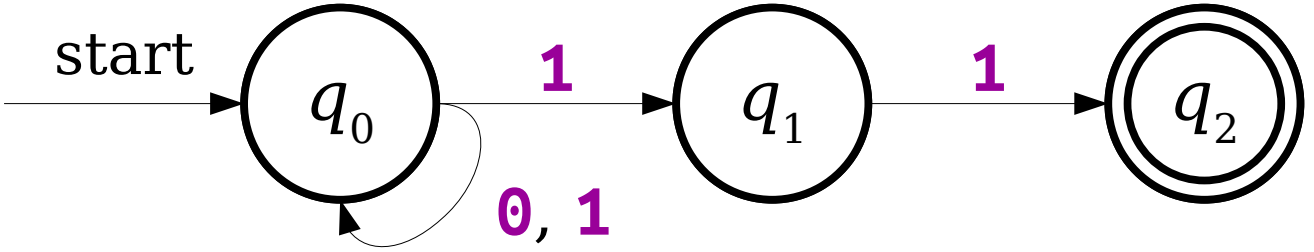
A More Complex NFA

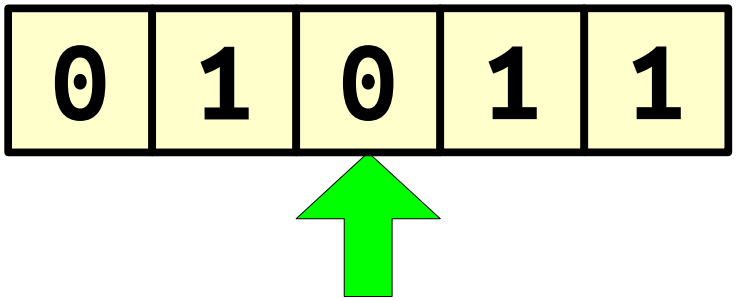


Oh no! There's no transition defined!

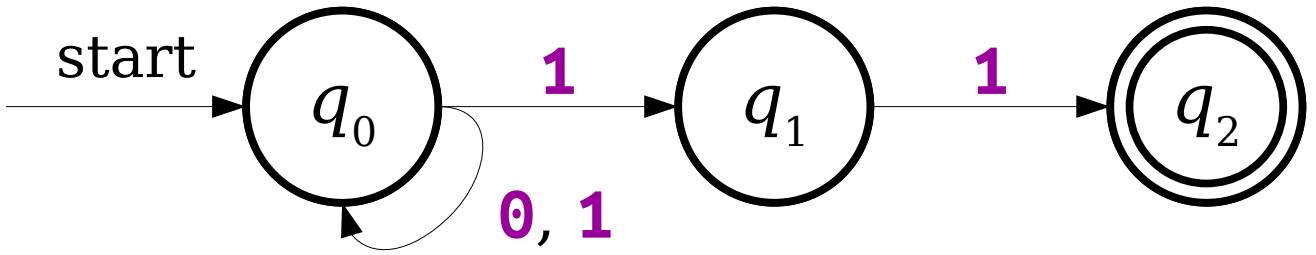


A More Complex NFA



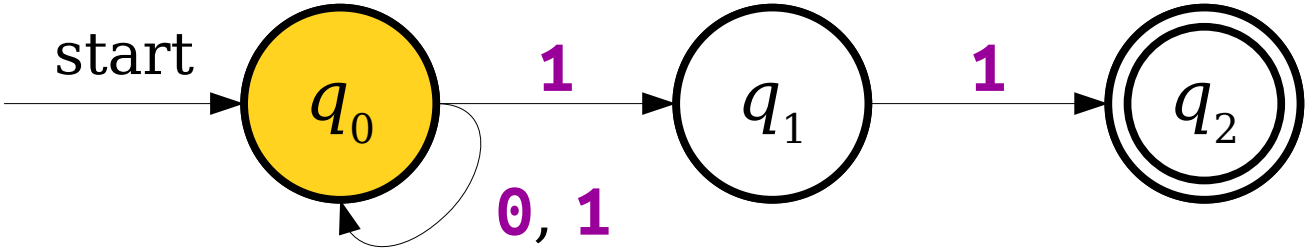


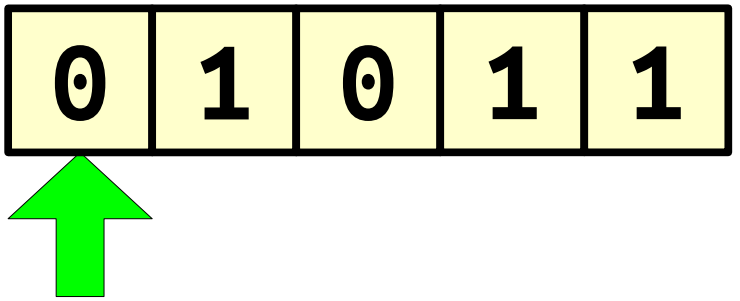
A More Complex NFA



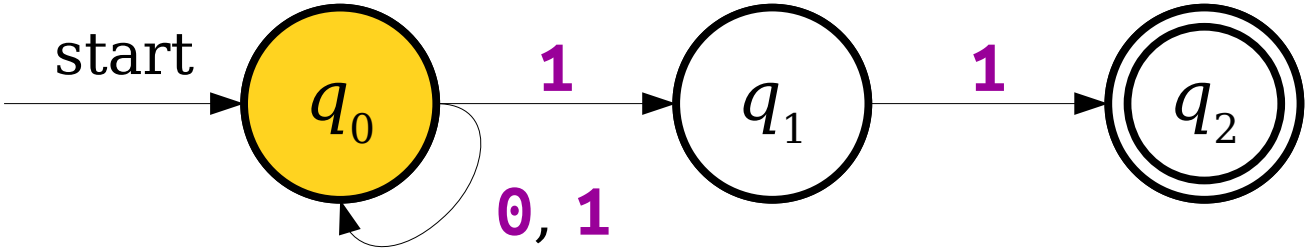
0	1	0	1	1

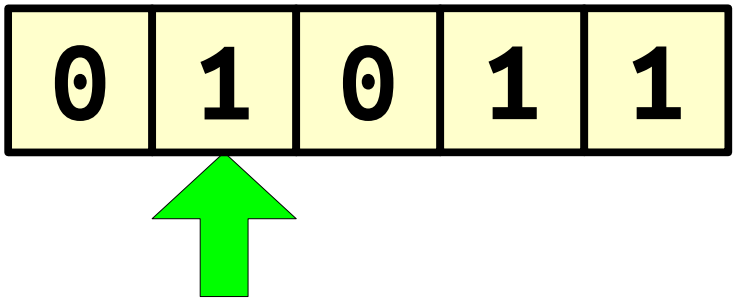
A More Complex NFA



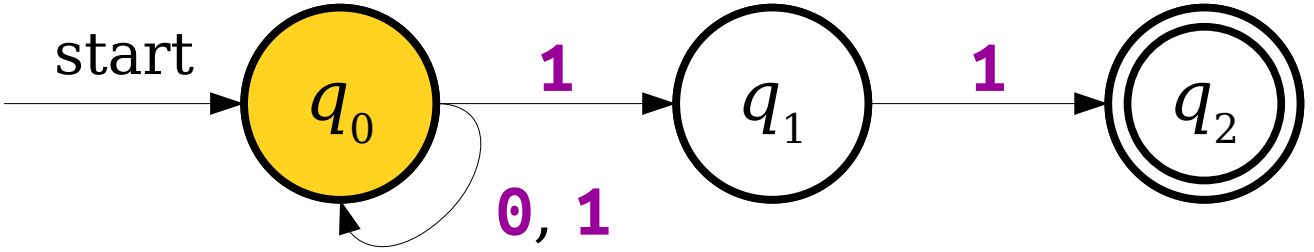


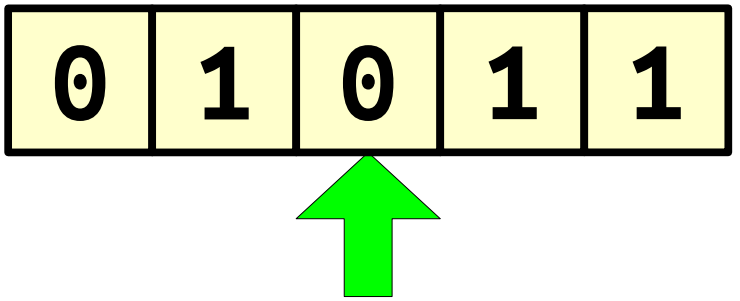
A More Complex NFA



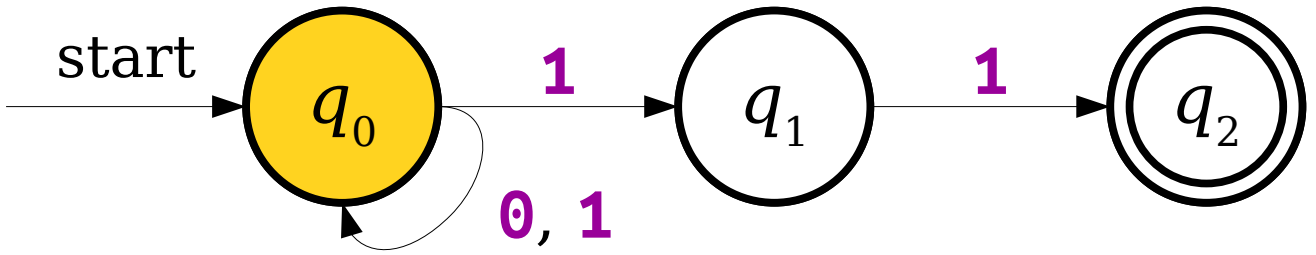


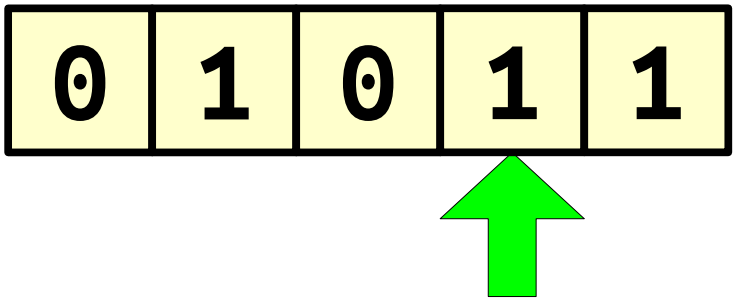
A More Complex NFA



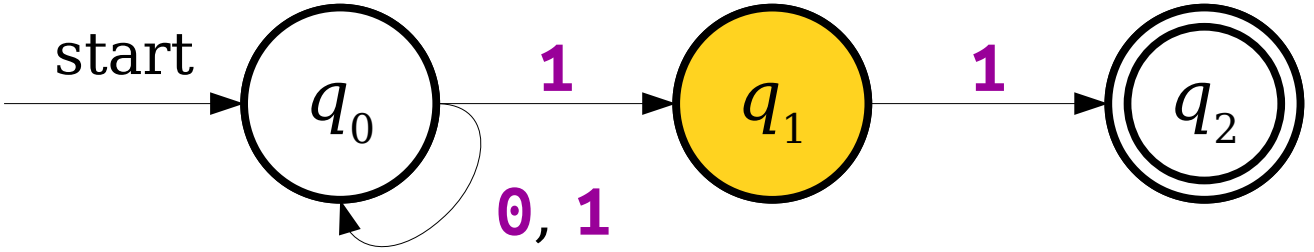


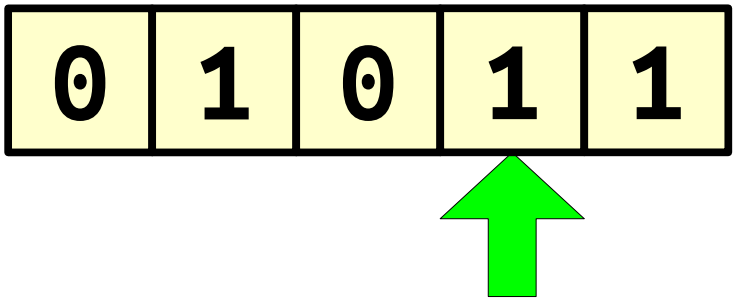
A More Complex NFA



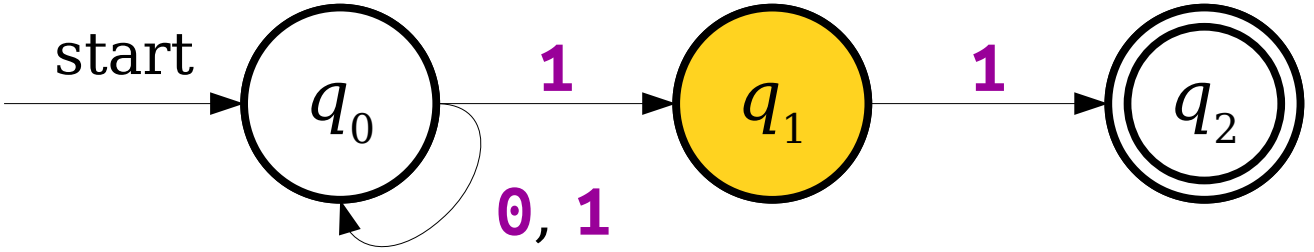


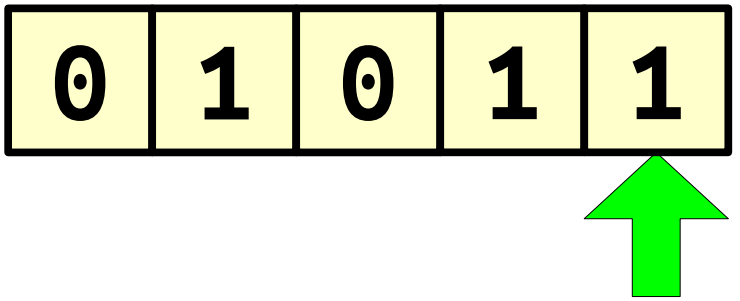
A More Complex NFA



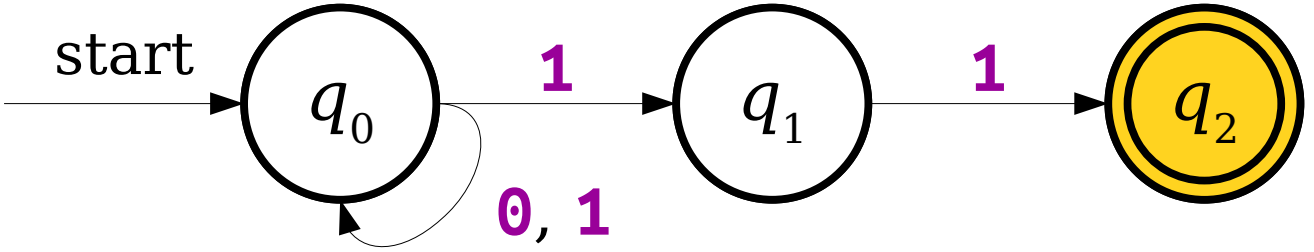


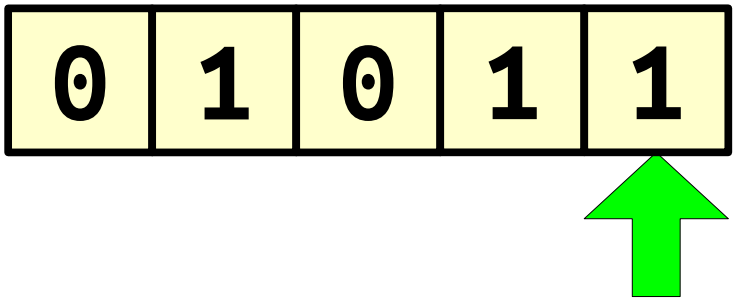
A More Complex NFA



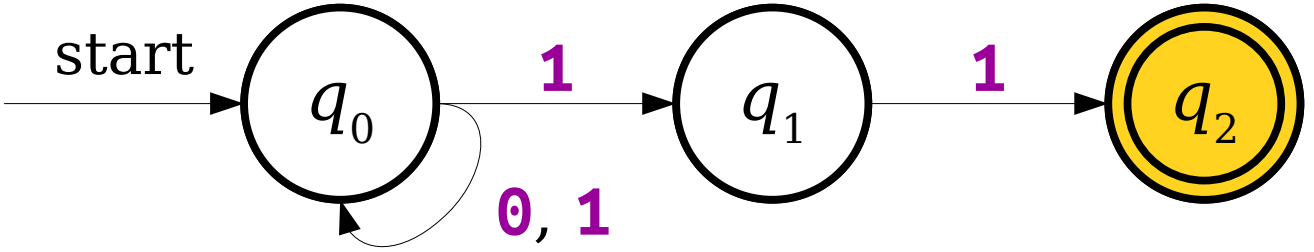


A More Complex NFA



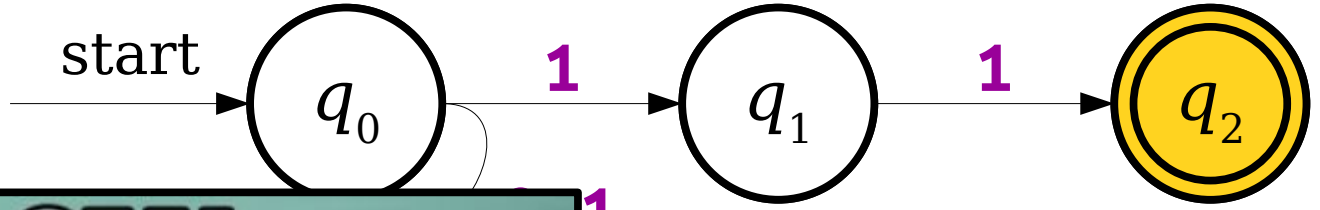


A More Complex NFA



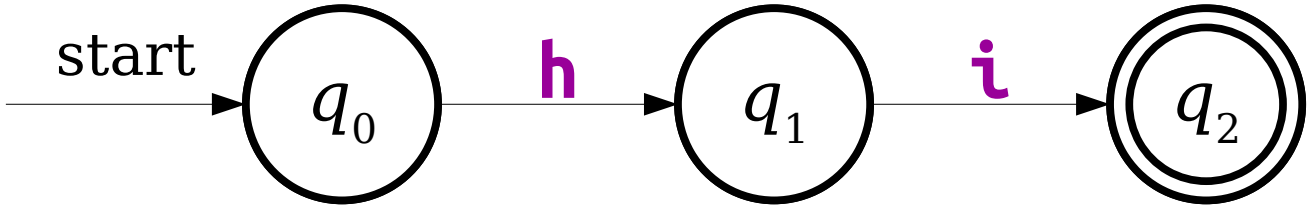
0	1	0	1	1

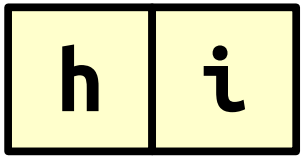
A More Complex NFA



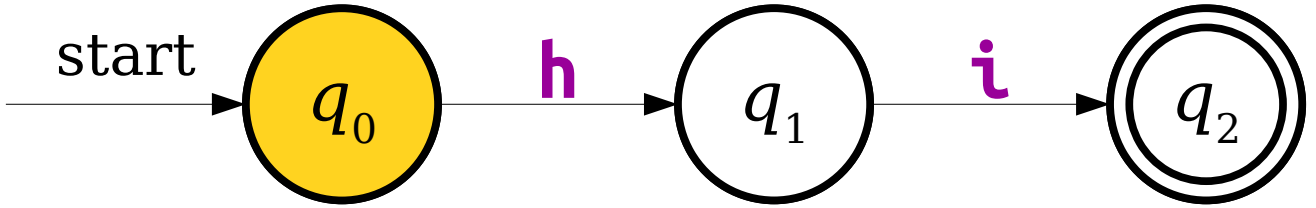
		0	1	1

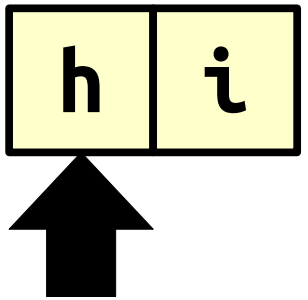
Hello, NFA!



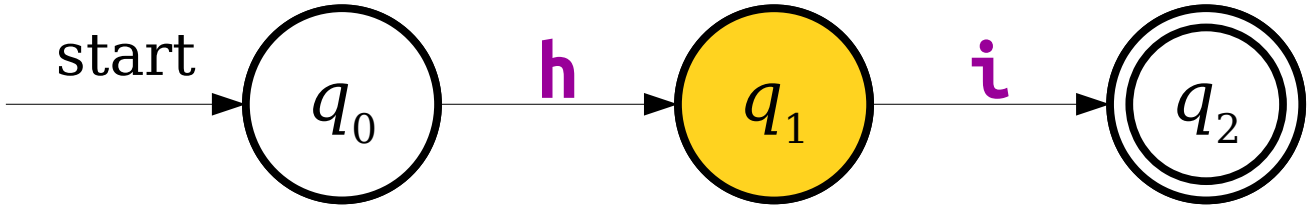


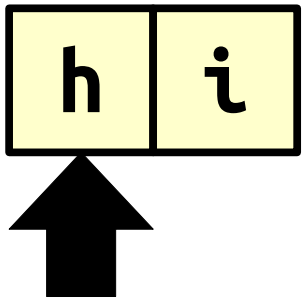
Hello, NFA!



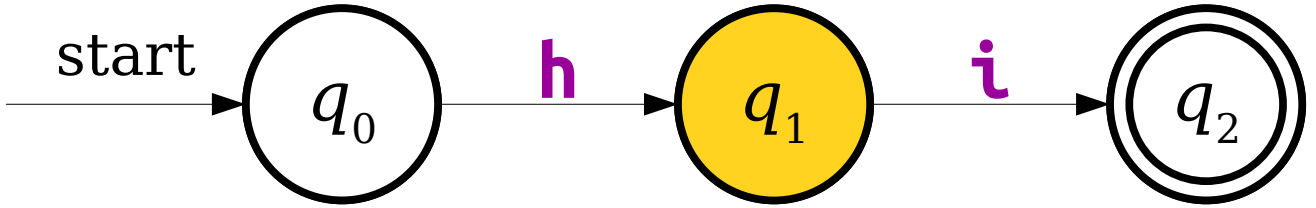


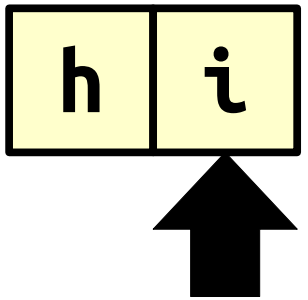
Hello, NFA!



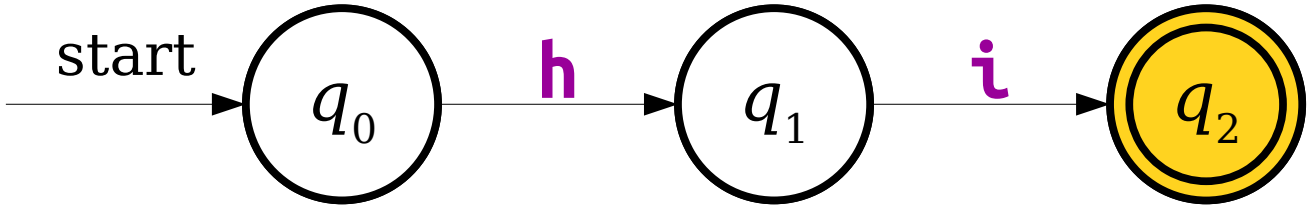


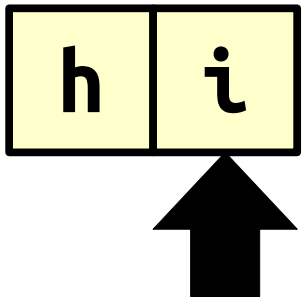
Hello, NFA!



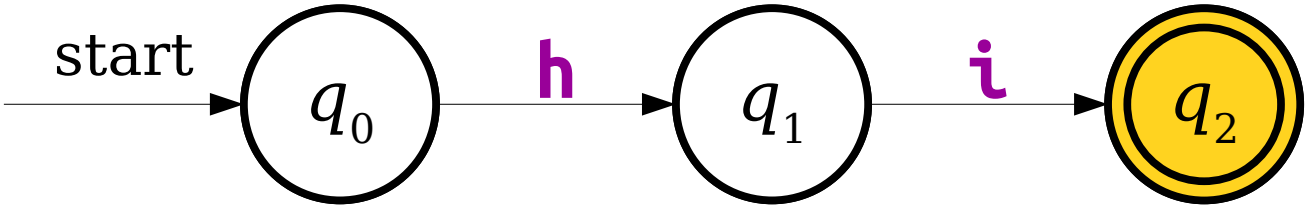


Hello, NFA!



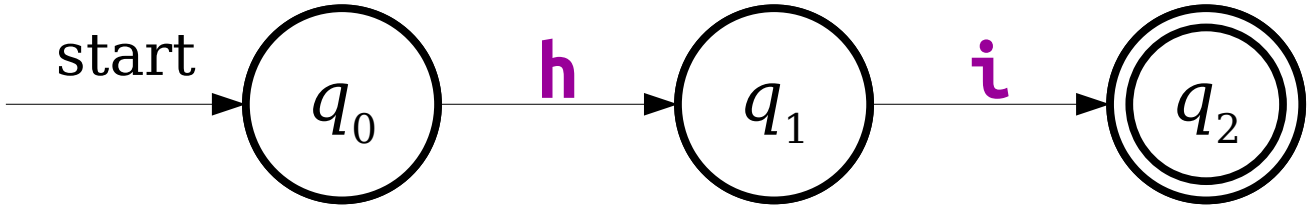


Hello, NFA!



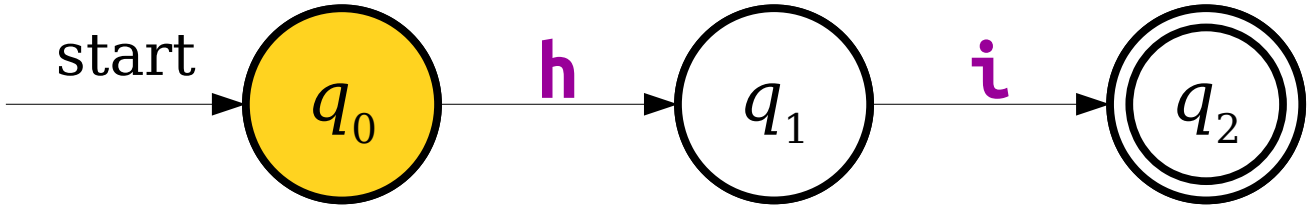
h	i

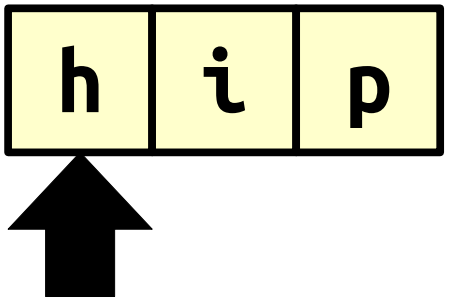
Tragedy in Paradise



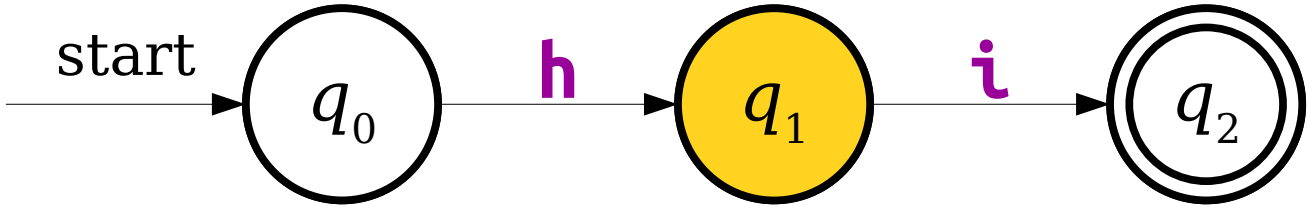
h	i	p

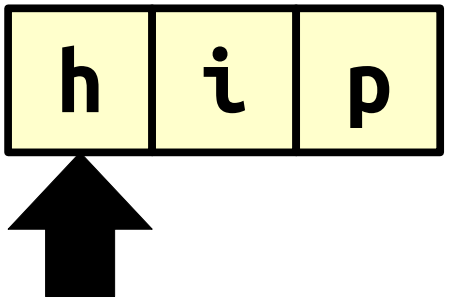
Tragedy in Paradise



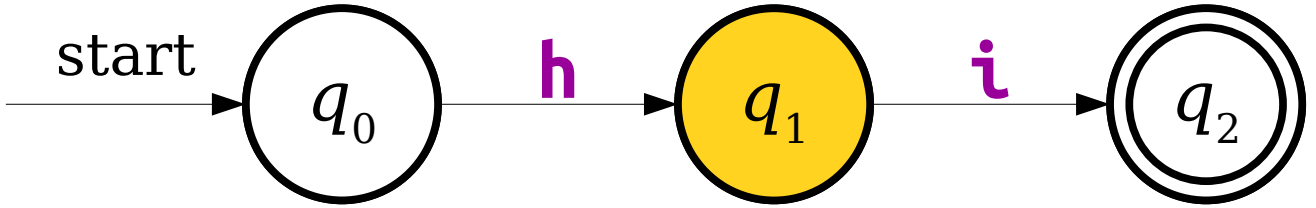


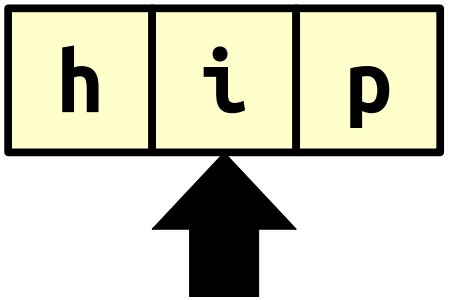
Tragedy in Paradise



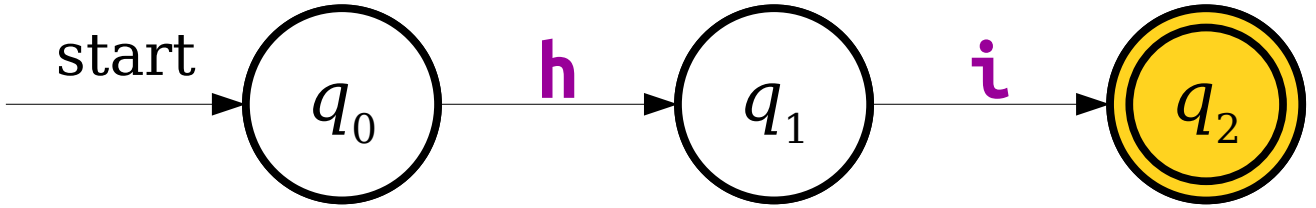


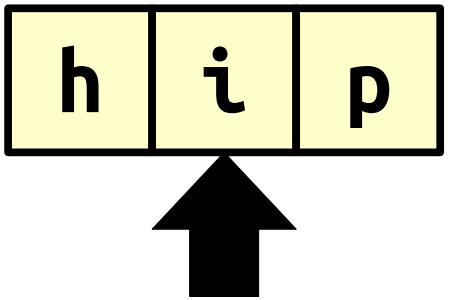
Tragedy in Paradise



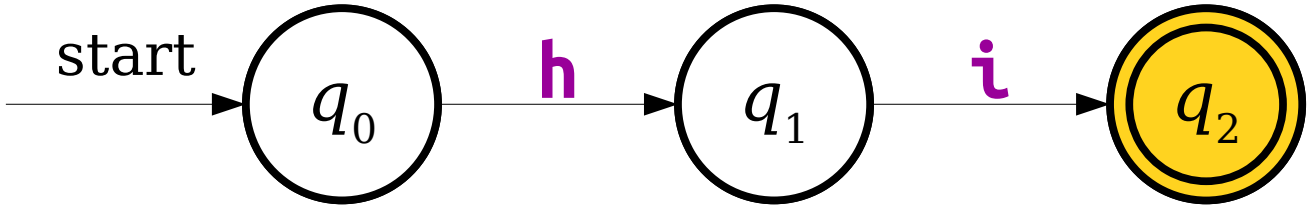


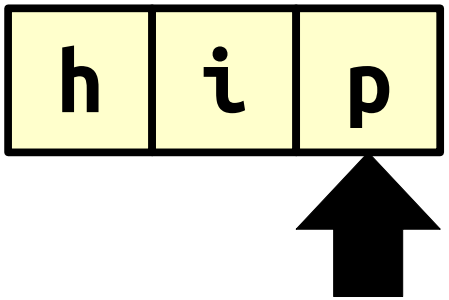
Tragedy in Paradise



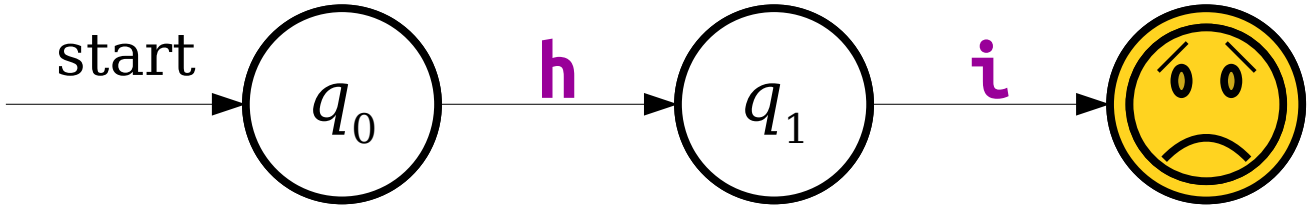


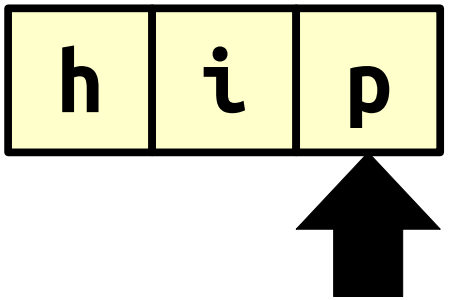
Tragedy in Paradise



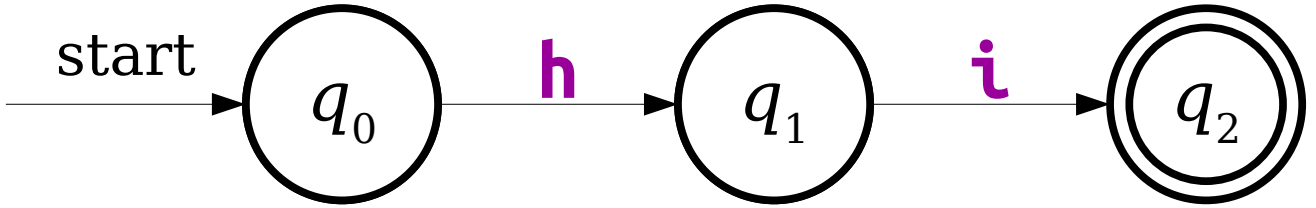


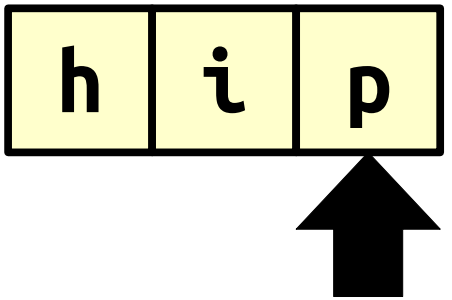
Tragedy in Paradise

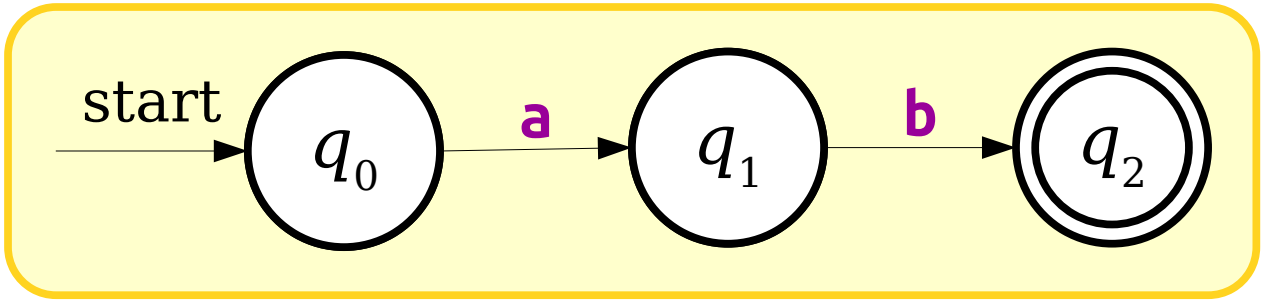




Tragedy in Paradise

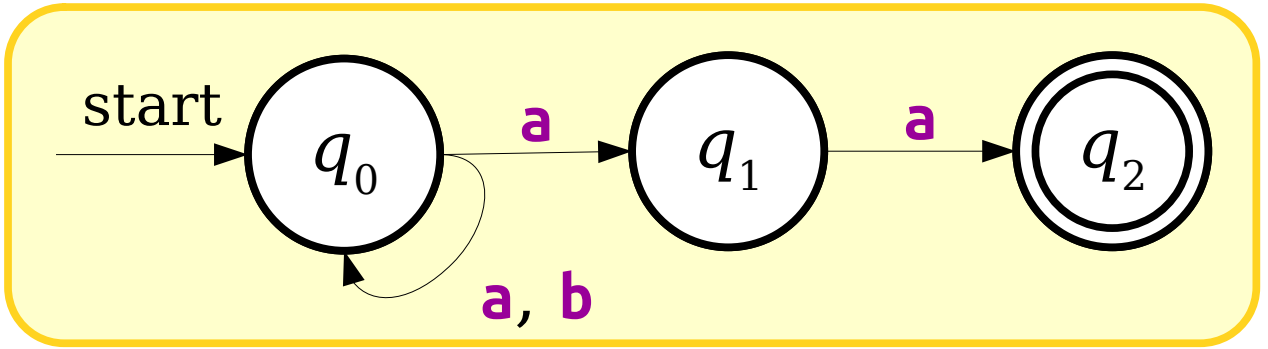




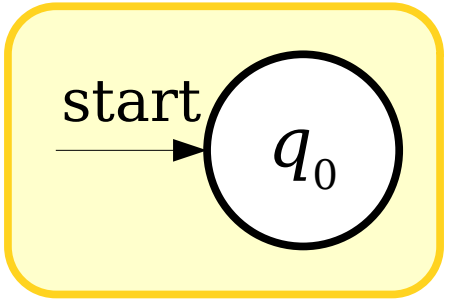


{ab}

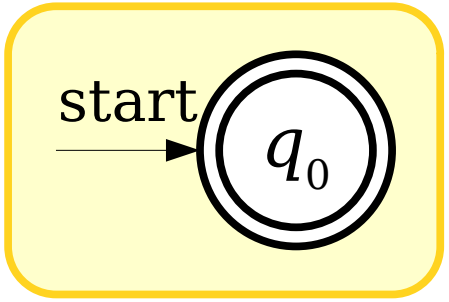
Question to ponder:
Why is the answer
{ $w \in \Sigma^*$ | w ends in aaa }
not correct?



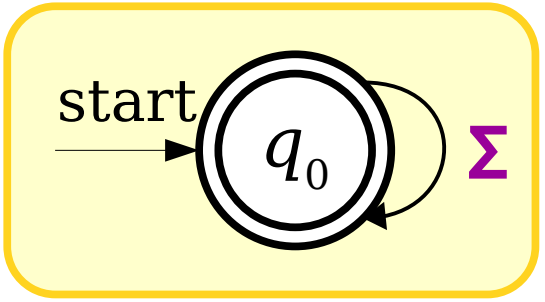
{ $w \in \Sigma^*$ | w ends in aa }



Ø



{ε}



Σ^*

The ***language of an NFA*** is

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

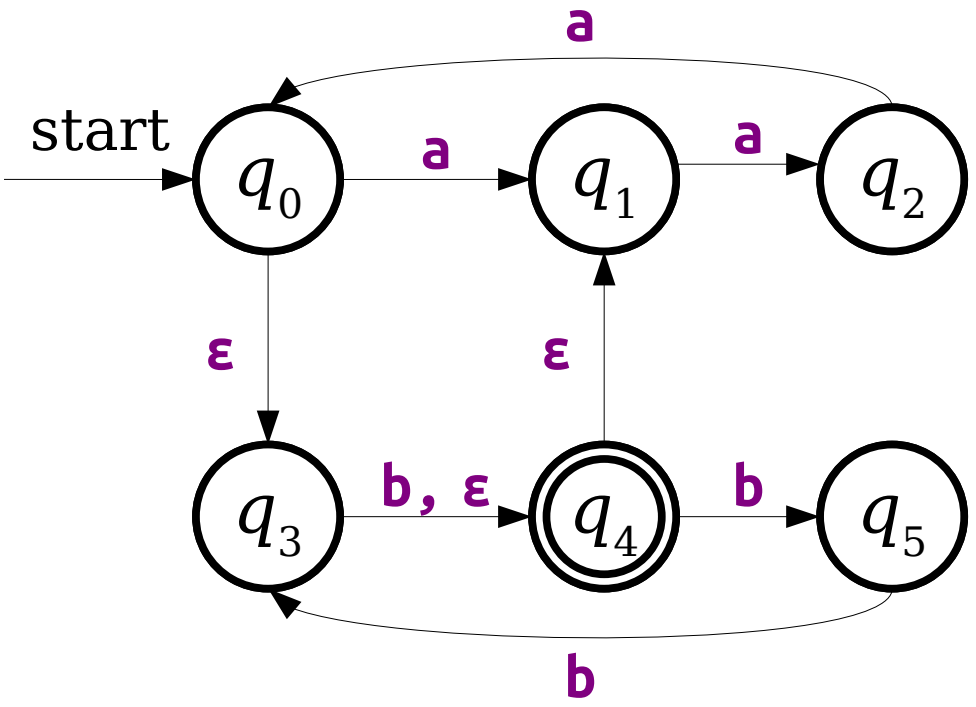
What is the language of each NFA?
(Assume $\Sigma = \{a, b\}$.)

ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

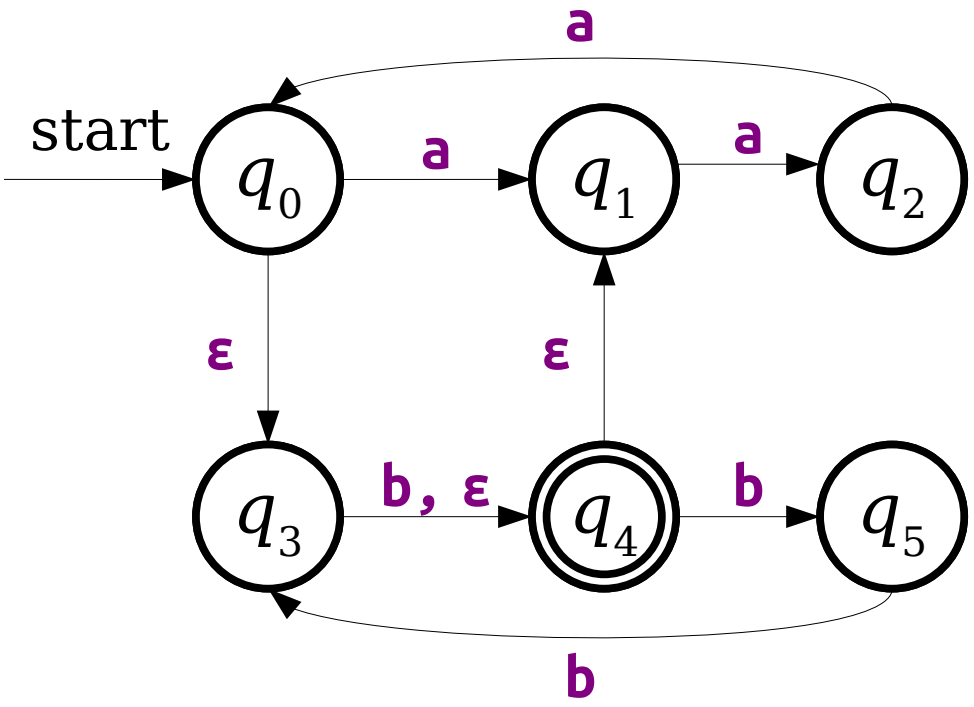
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



ε-Transitions

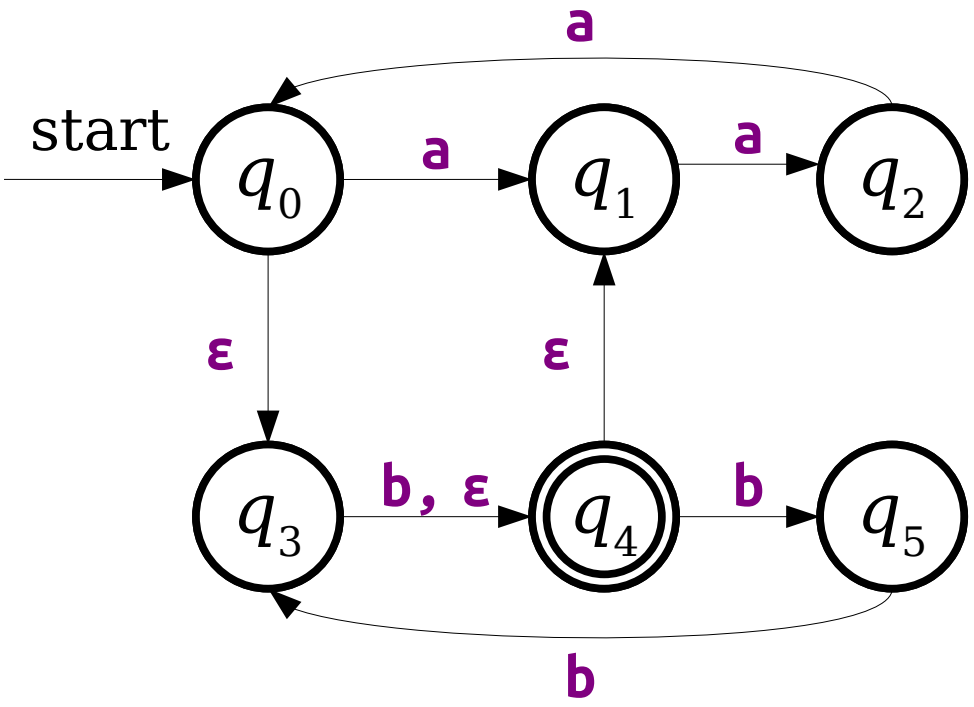
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

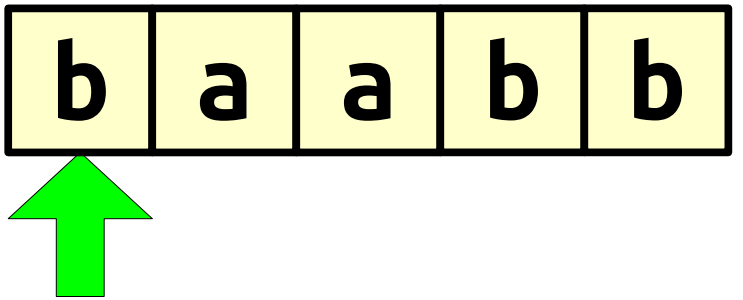


b	a	a	b	b

ε-Transitions

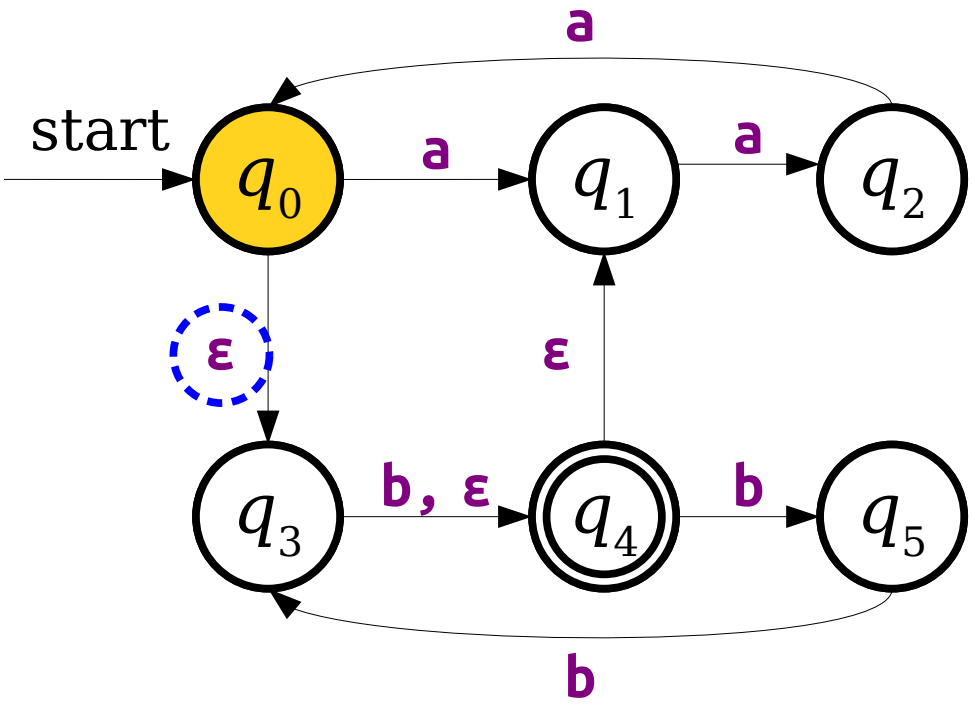
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

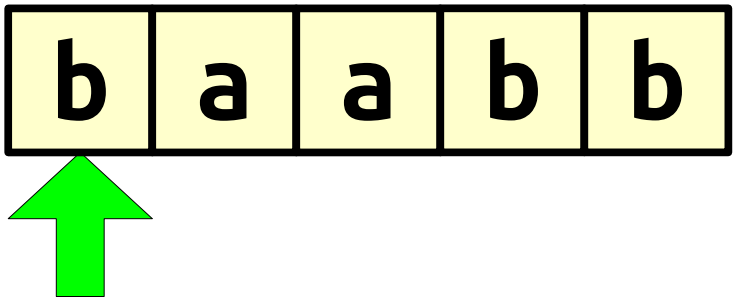




ε-Transitions

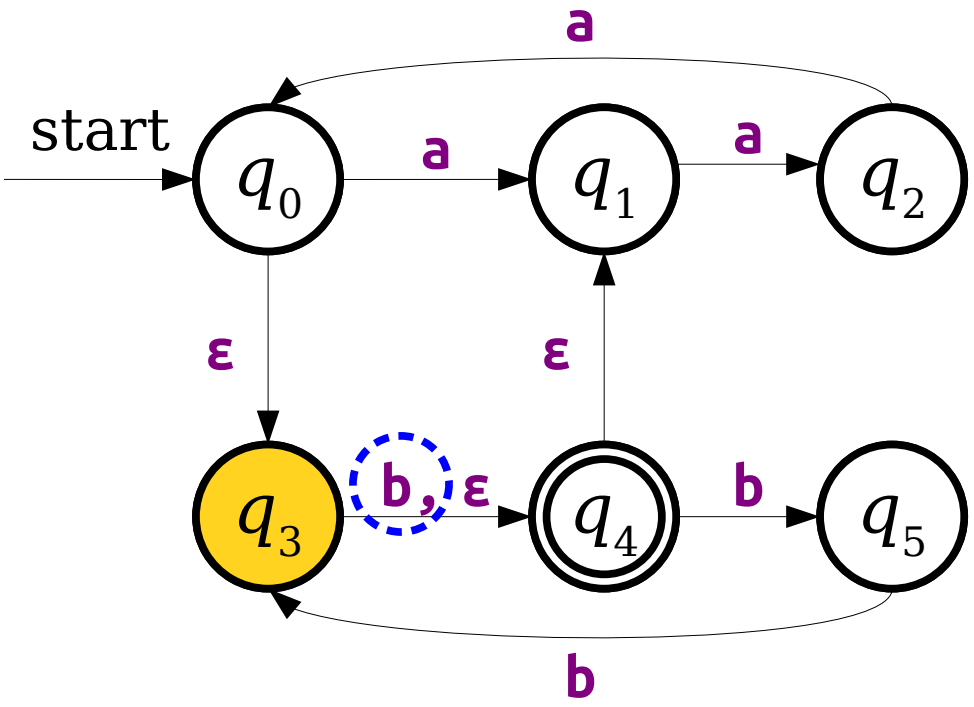
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

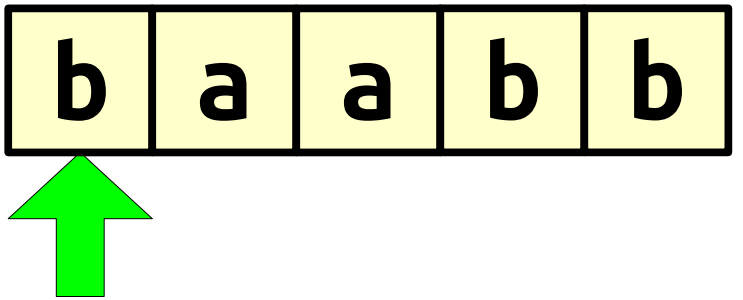




ε-Transitions

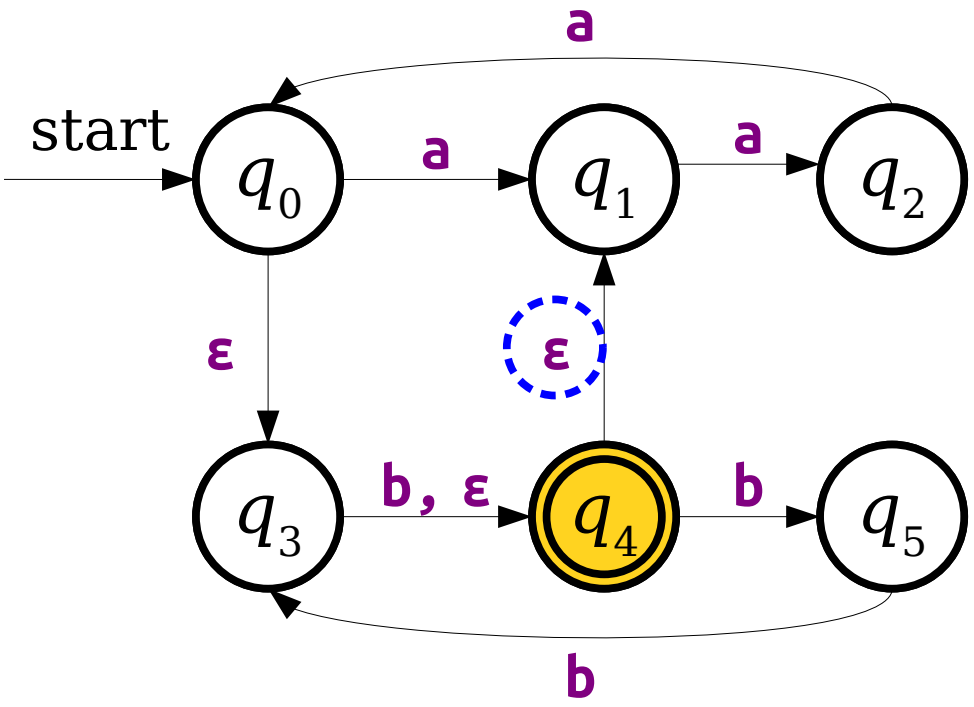
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

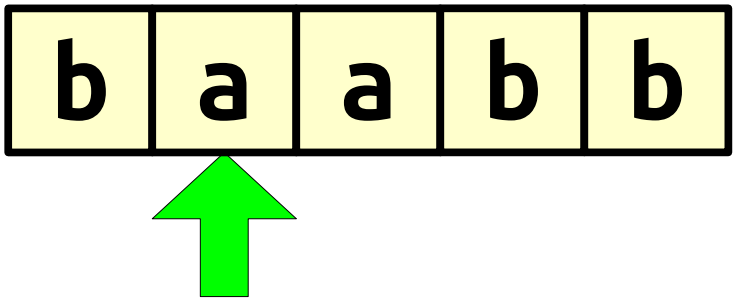




ε-Transitions

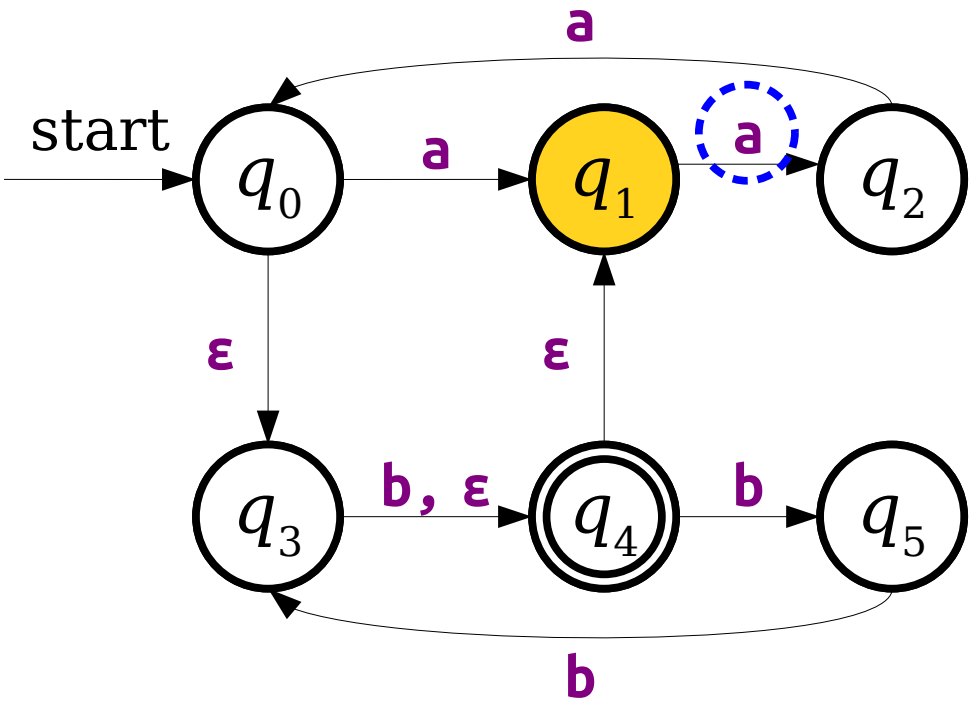
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

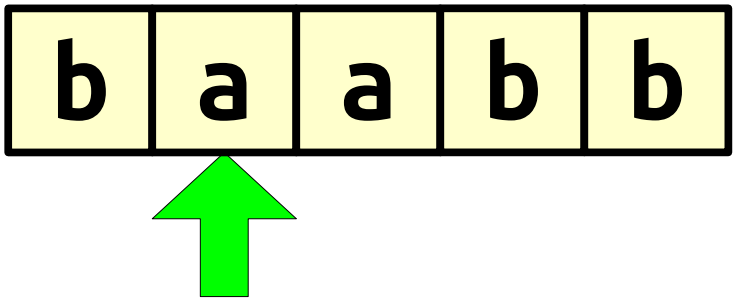




ε-Transitions

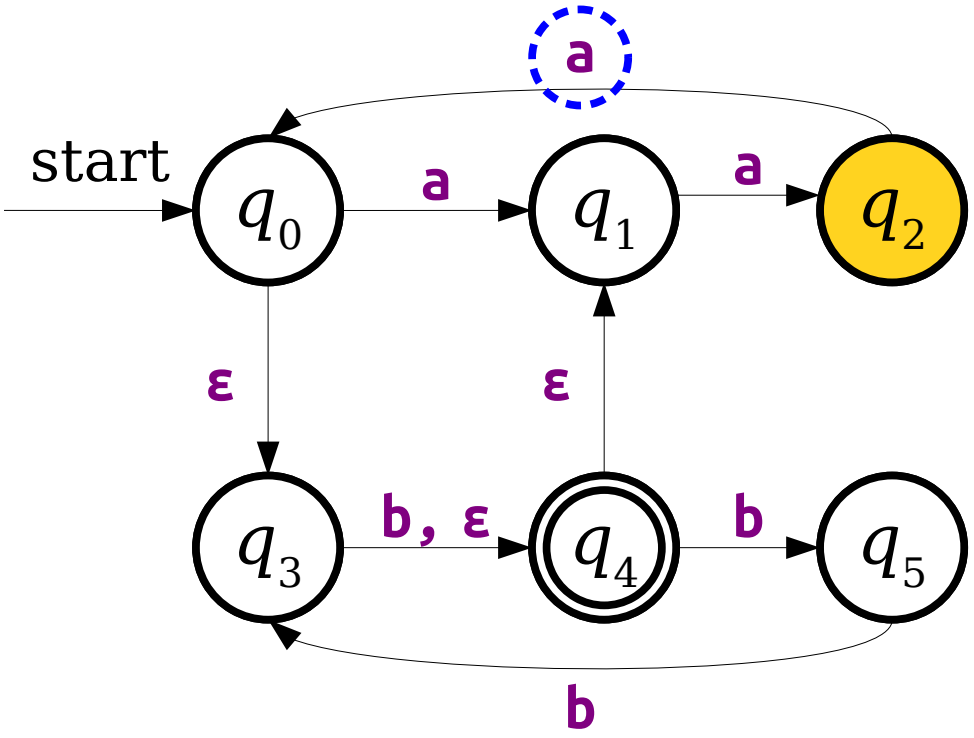
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

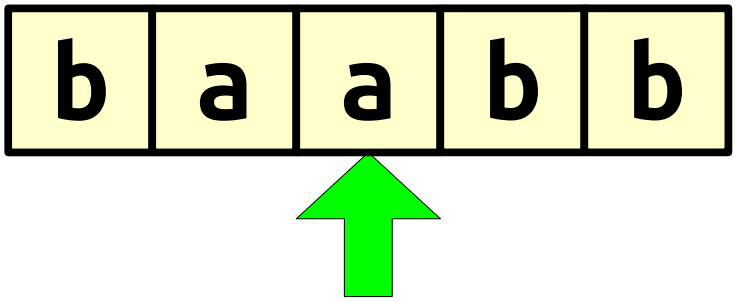




ε-Transitions

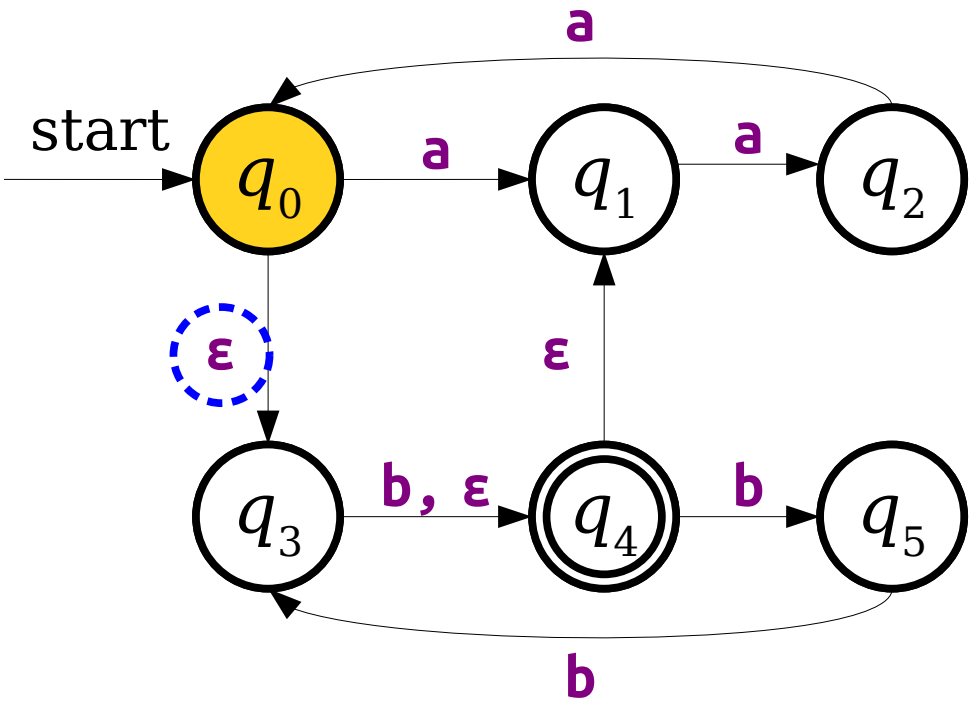
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

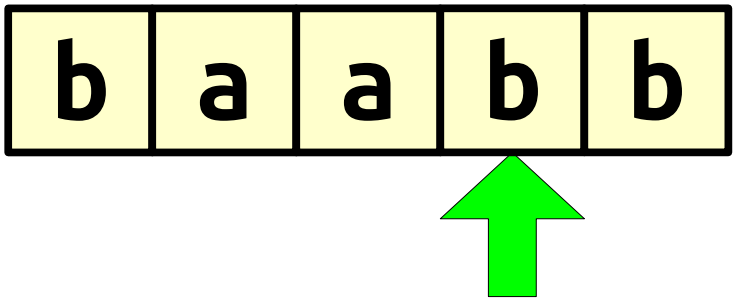




ε-Transitions

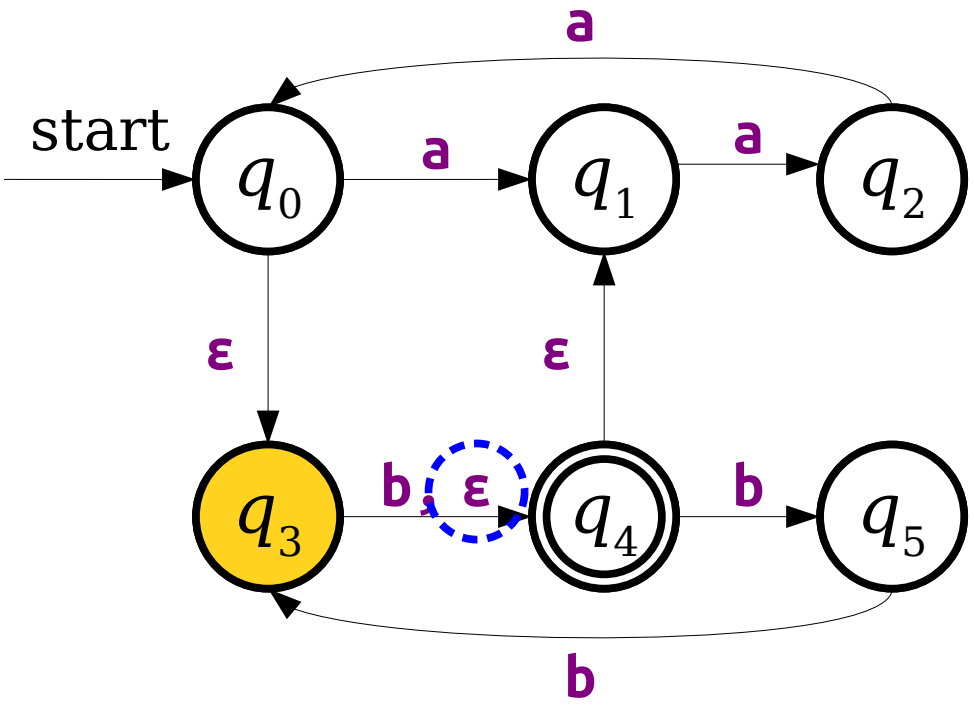
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

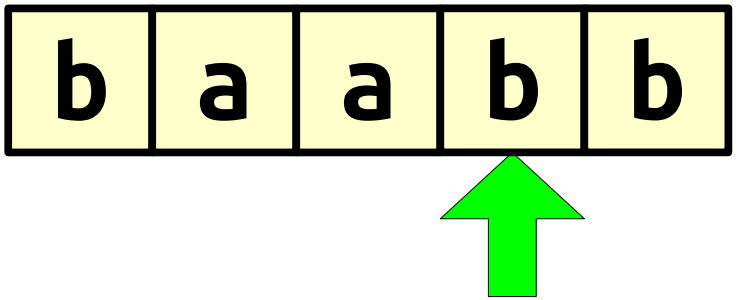




ε-Transitions

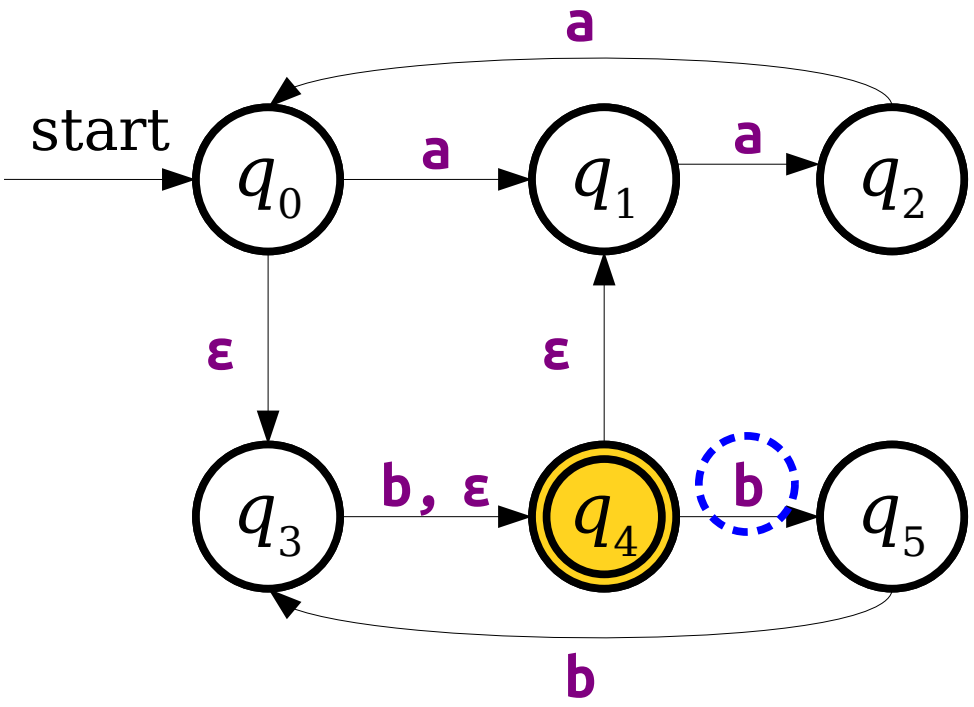
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

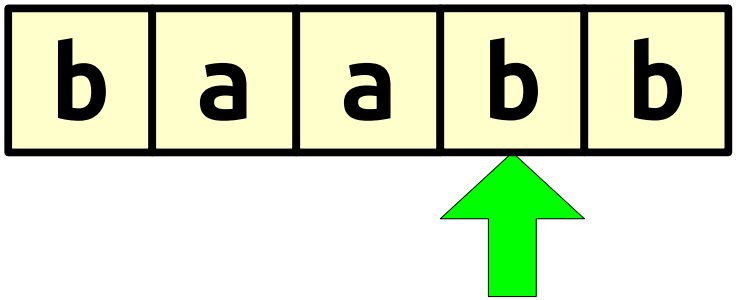




ε-Transitions

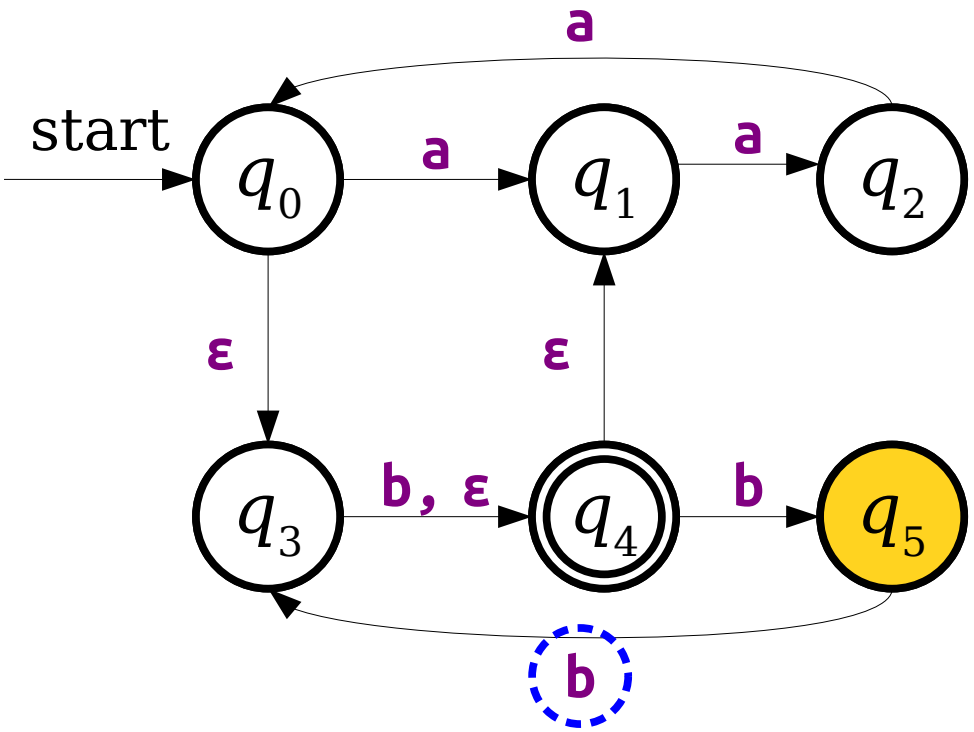
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

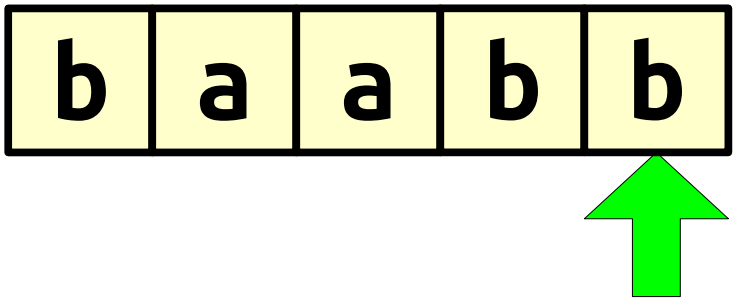




ε-Transitions

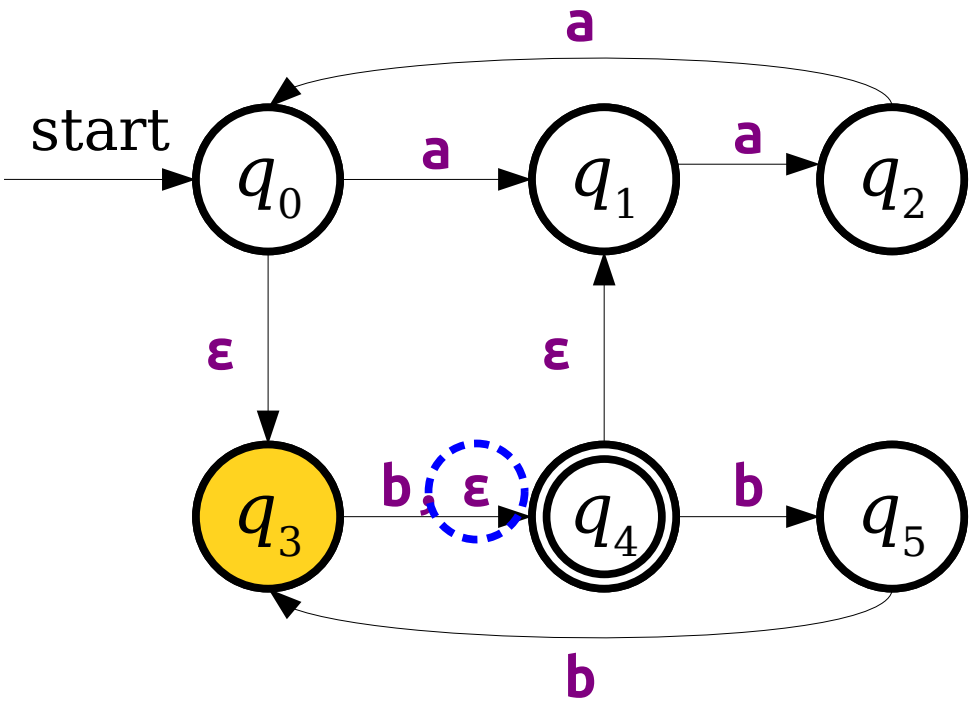
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.





ε-Transitions

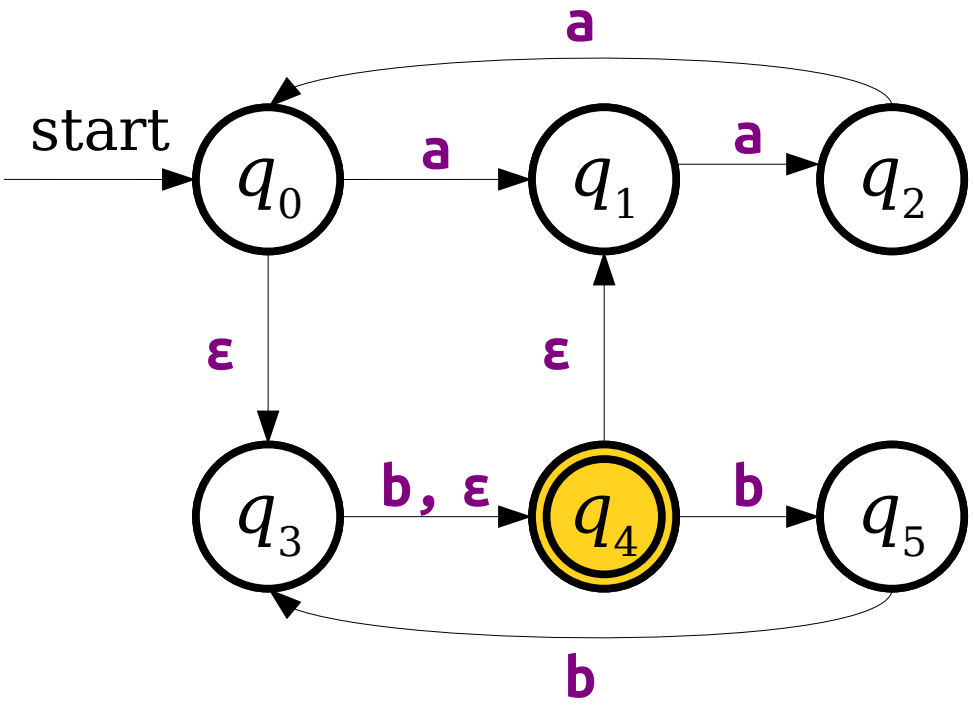
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



b	a	a	b	b

ε-Transitions

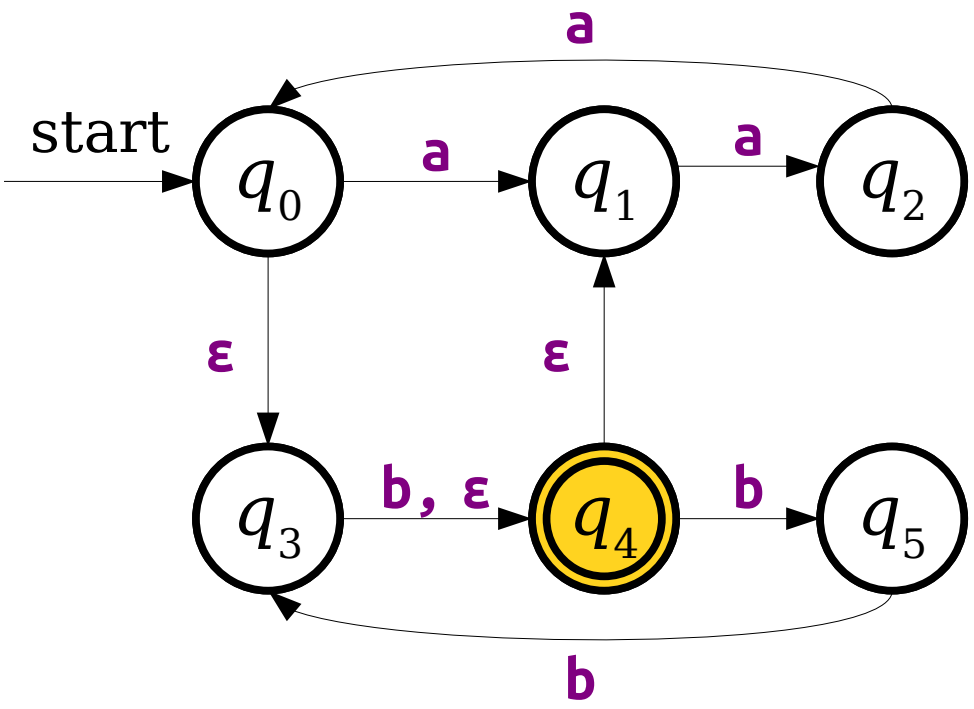
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.

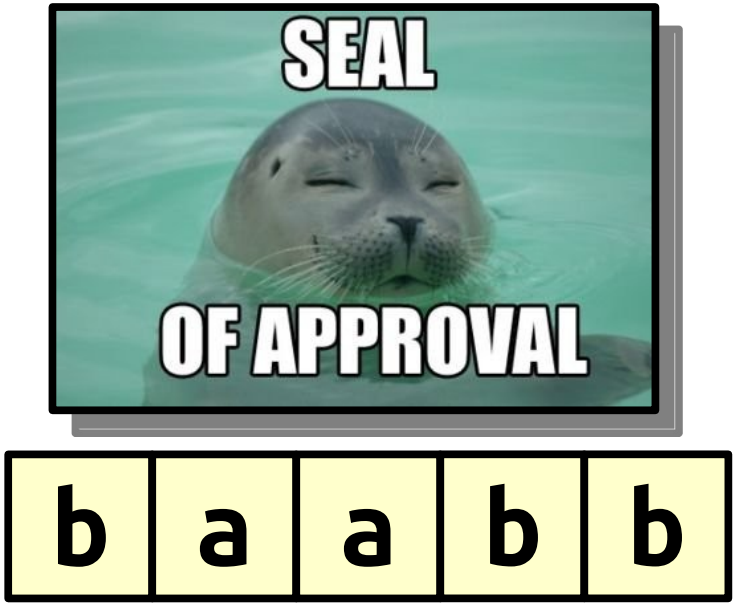


b	a	a	b	b

ε-Transitions

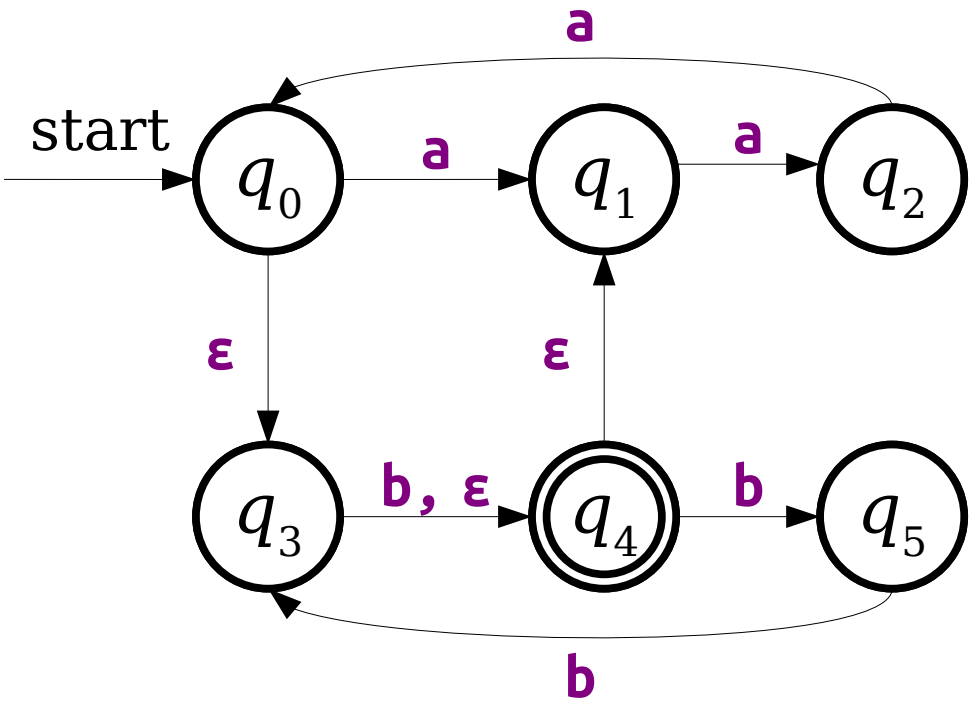
- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.





ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.



Not at all fun or rewarding exercise: what is the language of this NFA?

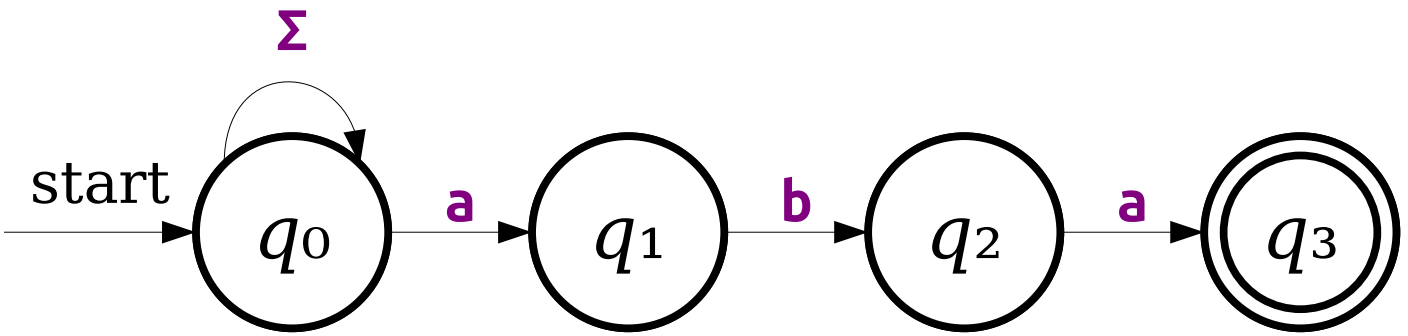
ε-Transitions

- NFAs have a special type of transition called the **ε-transition**.
- An NFA may follow any number of ε-transitions at any time without consuming any input.
- NFAs are not *required* to follow ε-transitions. It's simply another option at the machine's disposal.

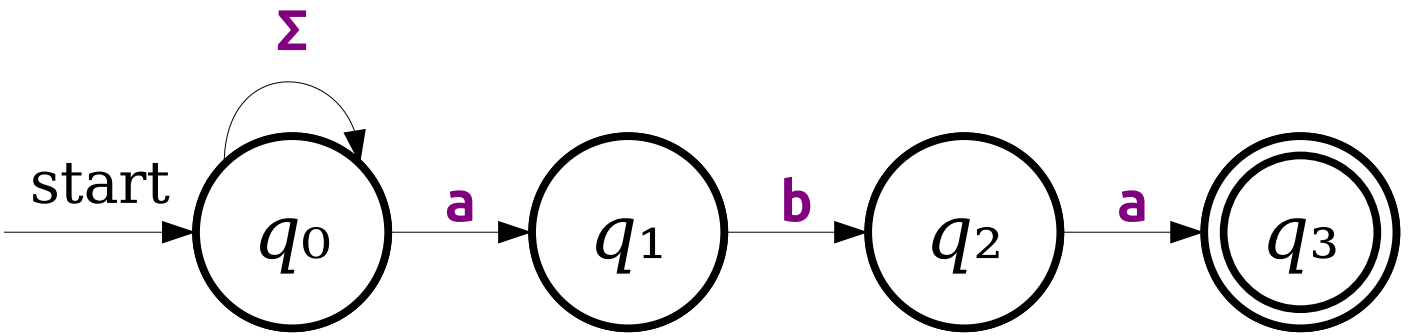
Intuiting Nondeterminism

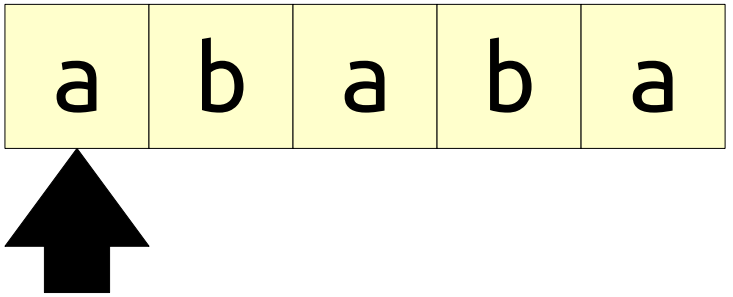
- Nondeterministic machines are a serious departure from physical computers. How can we build up an intuition for them?
- There are two particularly useful frameworks for interpreting nondeterminism:
  - ***Perfect positive guessing***
  - ***Massive parallelism***

Perfect Positive Guessing

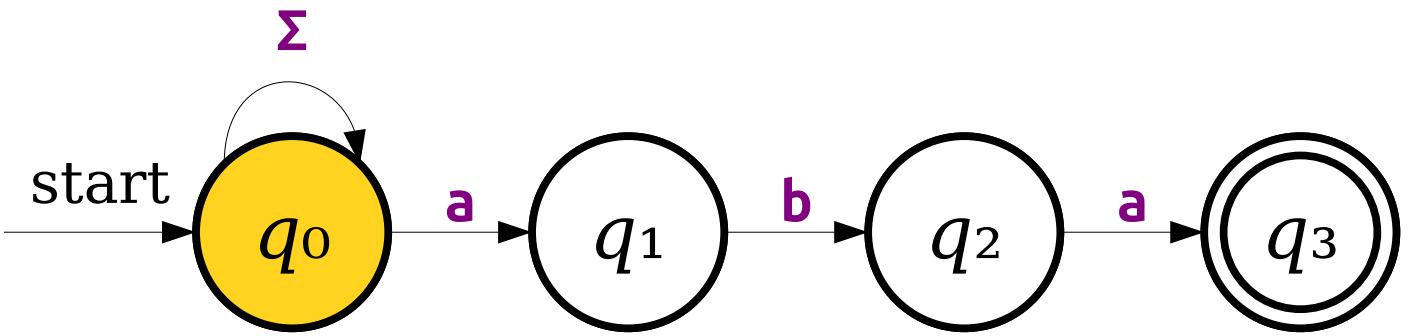


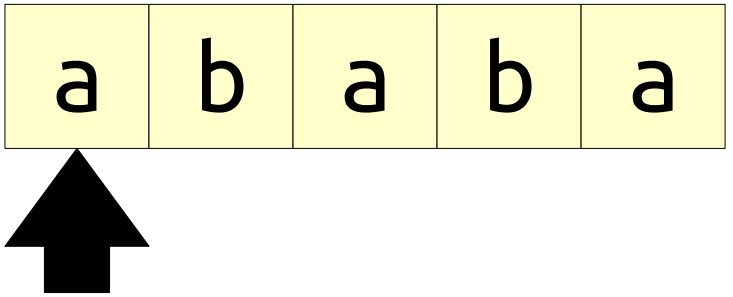
Perfect Positive Guessing



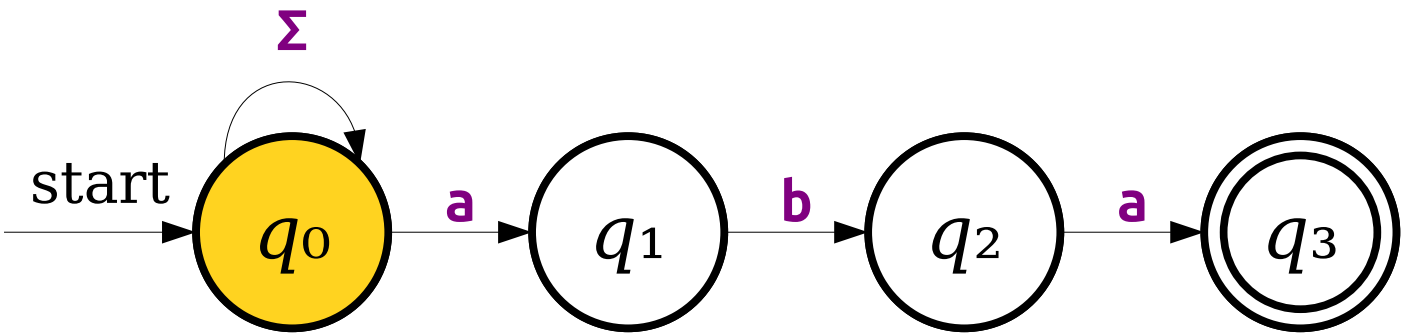


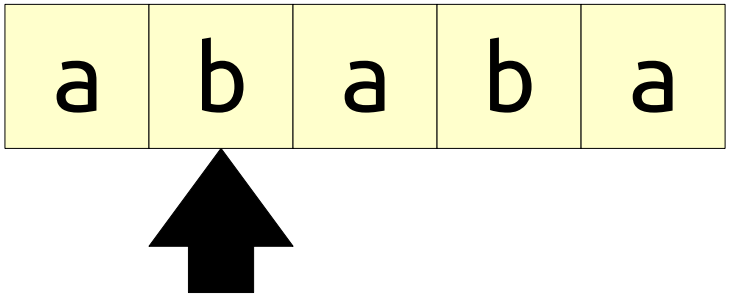
Perfect Positive Guessing



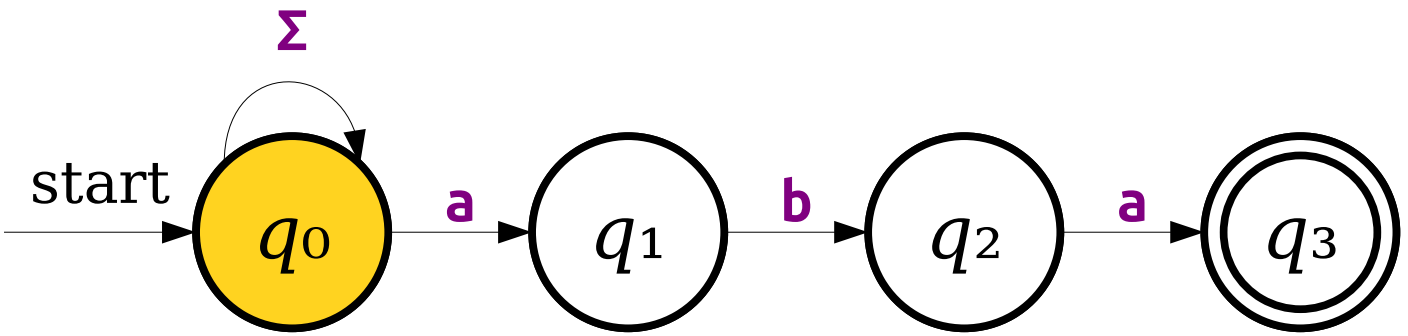


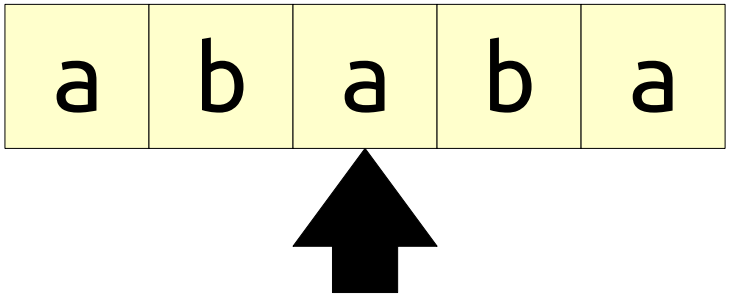
Perfect Positive Guessing



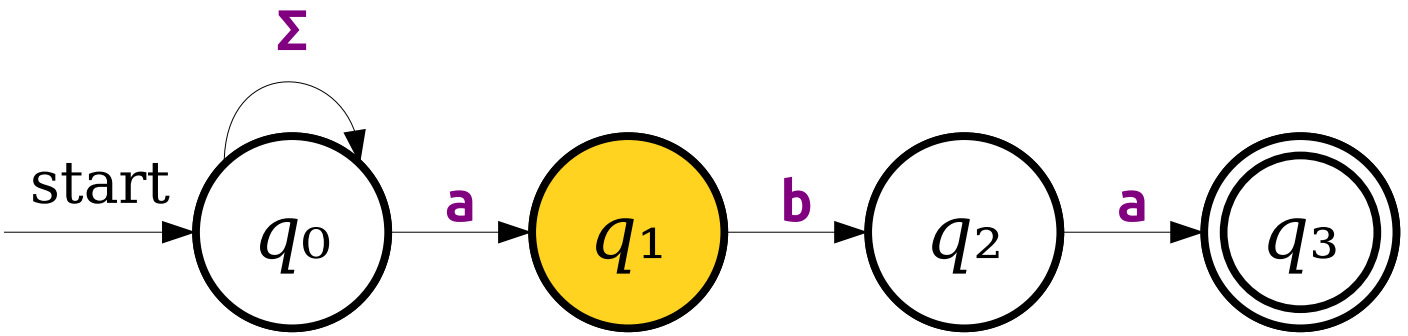


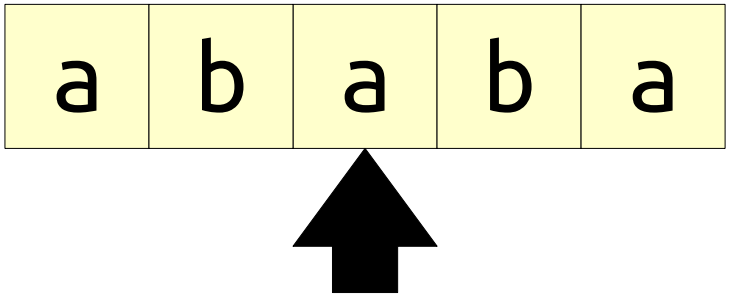
Perfect Positive Guessing



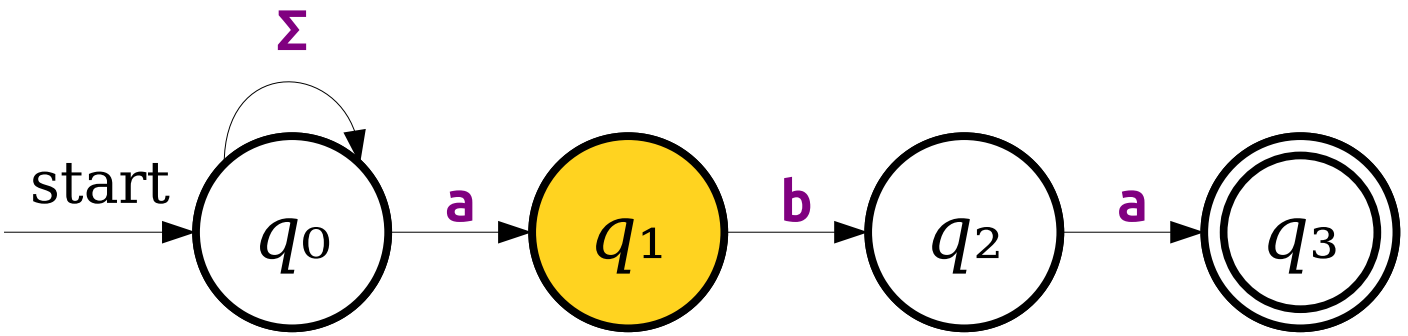


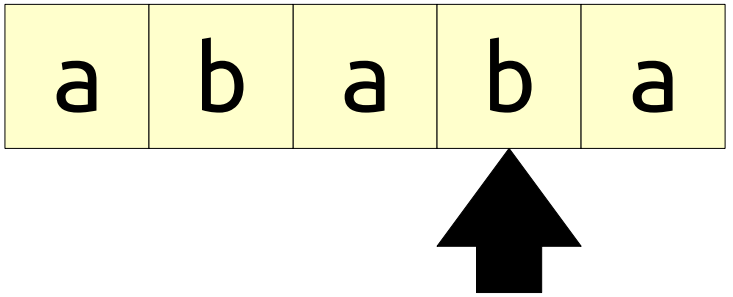
Perfect Positive Guessing



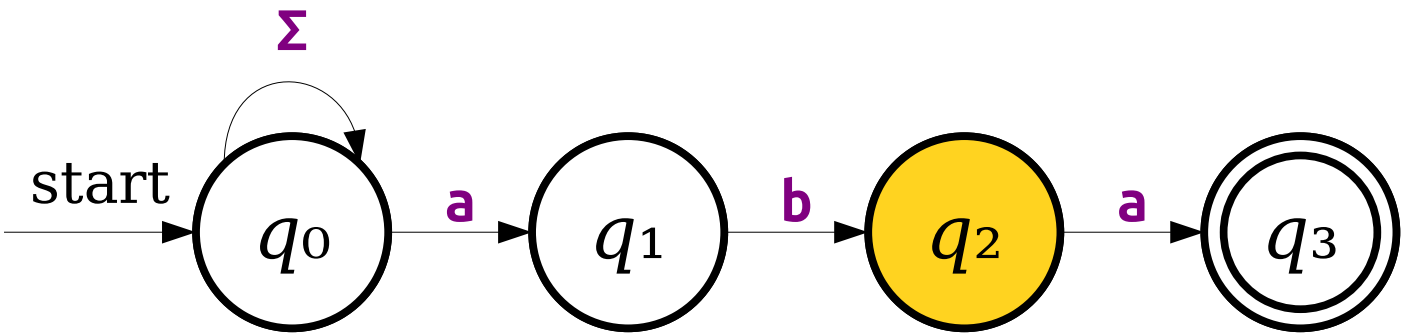


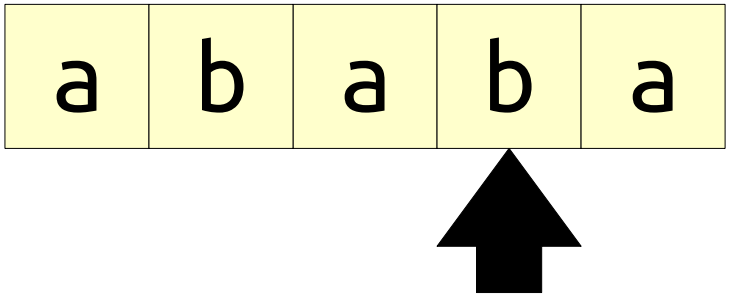
Perfect Positive Guessing



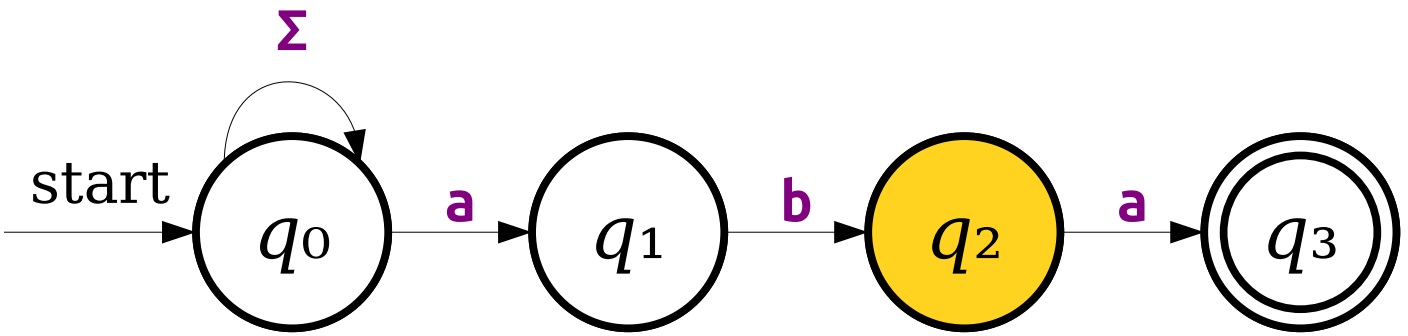


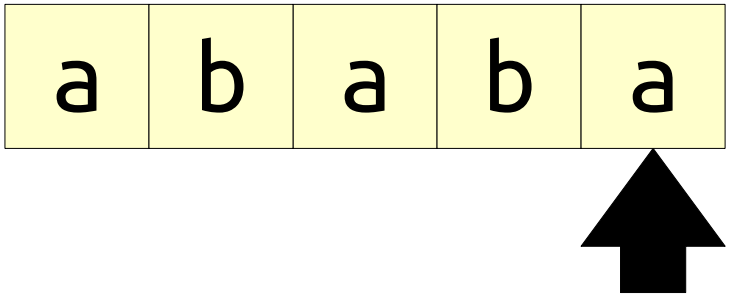
Perfect Positive Guessing



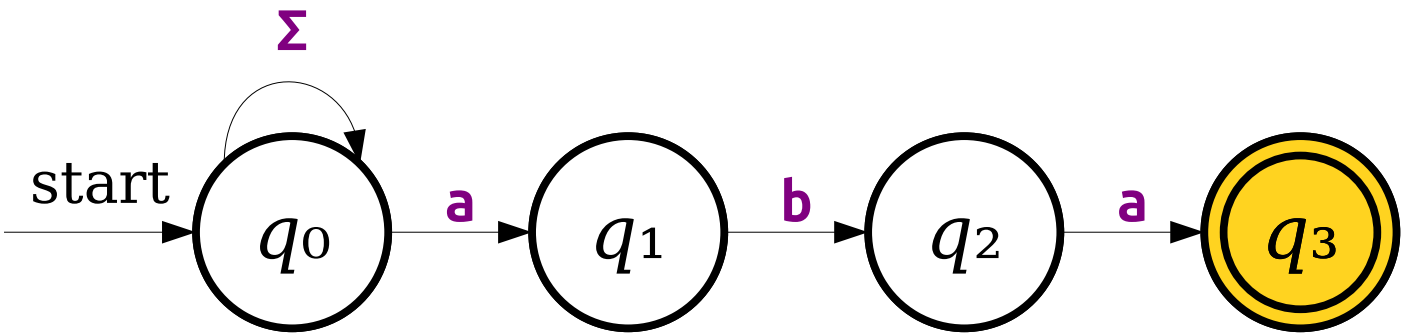


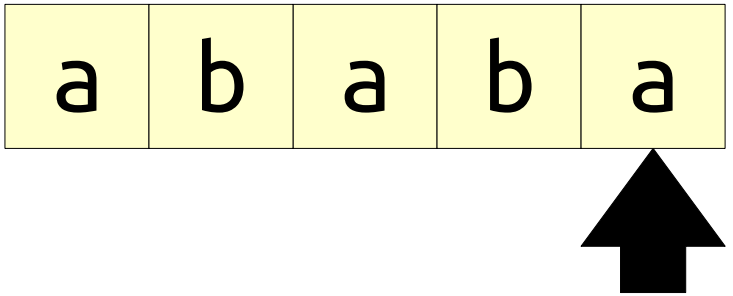
Perfect Positive Guessing



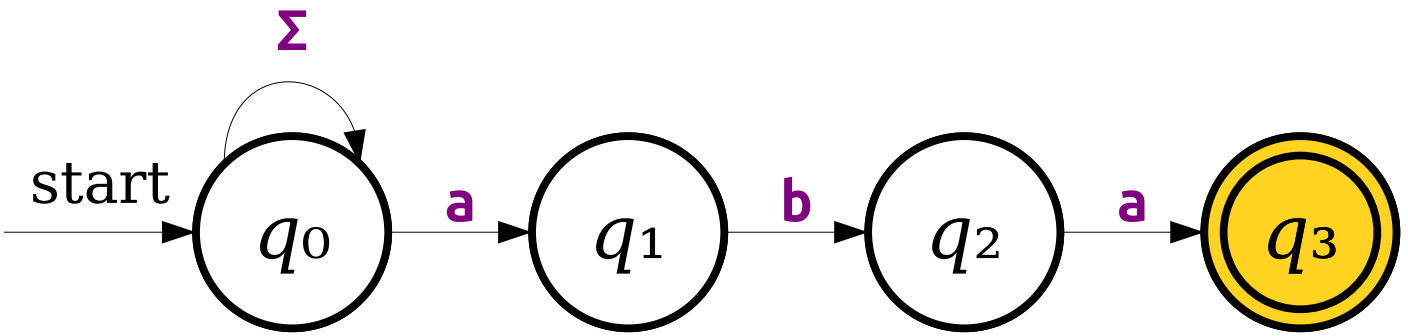


Perfect Positive Guessing





Perfect Positive Guessing

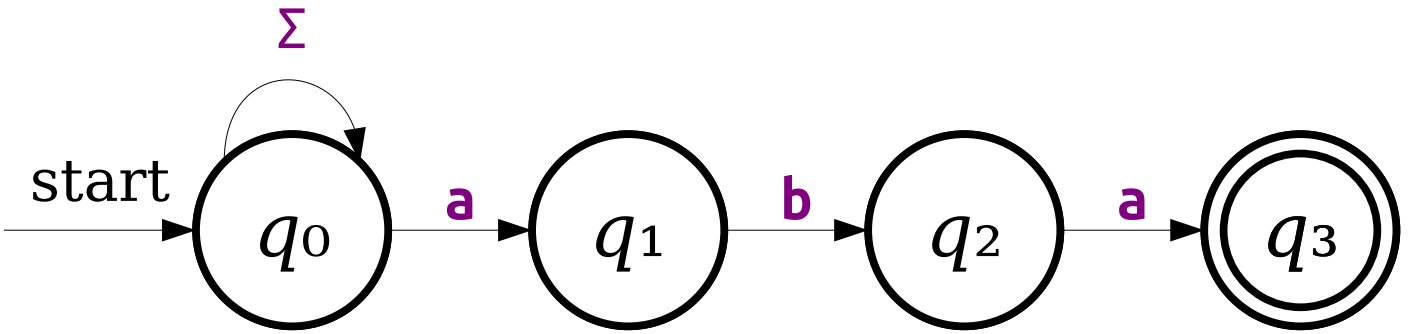


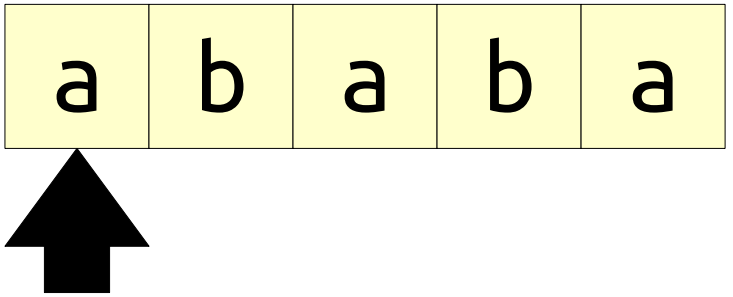
a	b	a	b	a

Perfect Positive Guessing

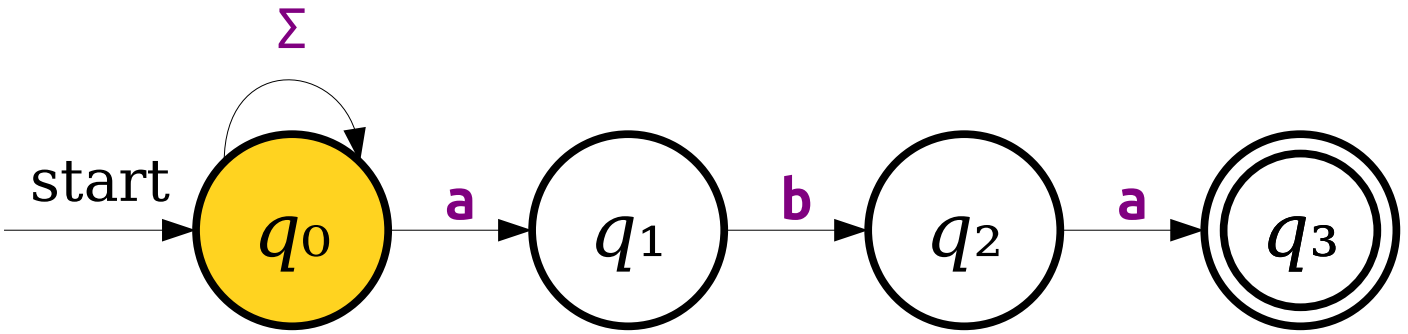
- We can view nondeterministic machines as having ***Magic Superpowers*** that enable them to guess choices that lead to an accepting state.
  - If there is at least one choice that leads to an accepting state, the machine will guess it.
  - If there are no choices, the machine guesses any one of the wrong guesses.
- There is no known way to physically model this intuition of nondeterminism – this is quite a departure from reality!

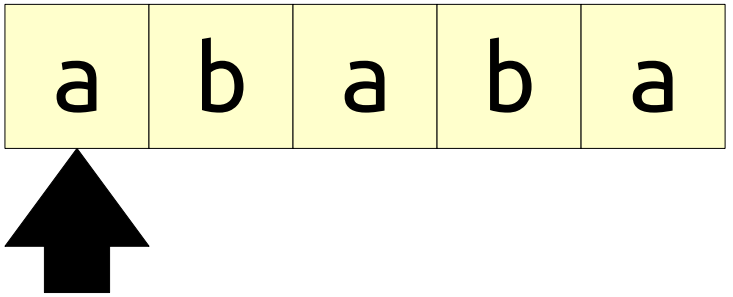
Massive Parallelism



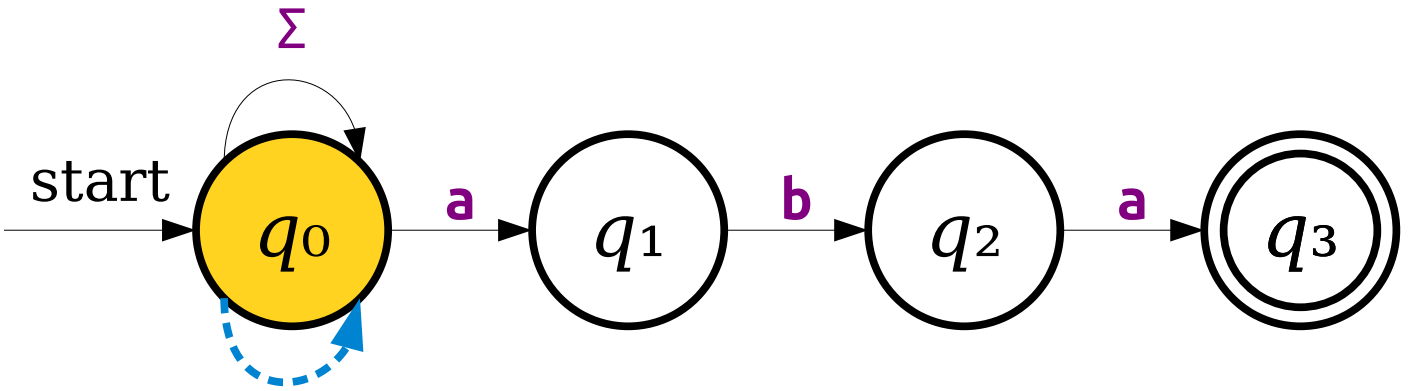


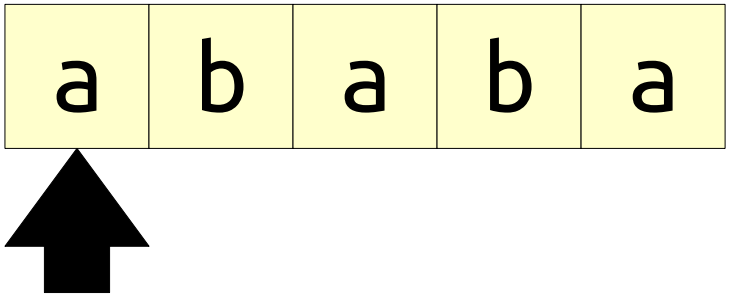
Massive Parallelism



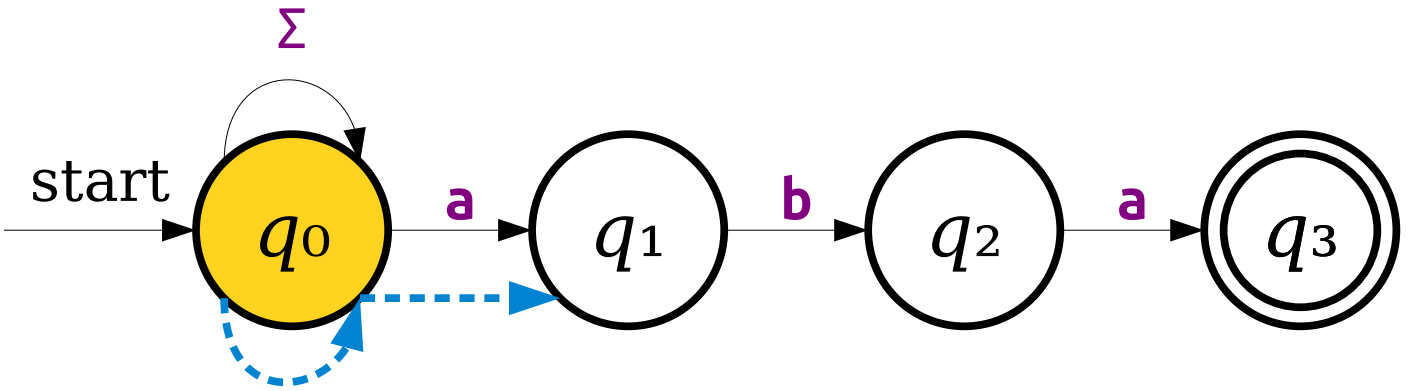


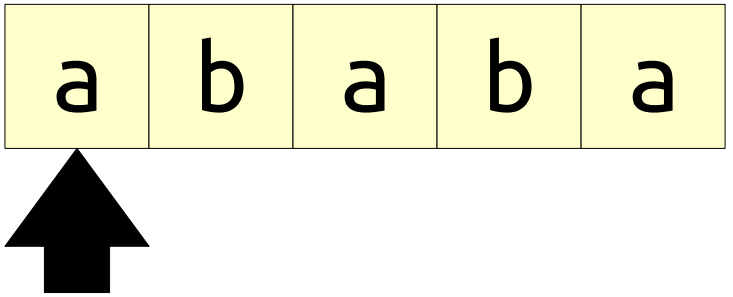
Massive Parallelism



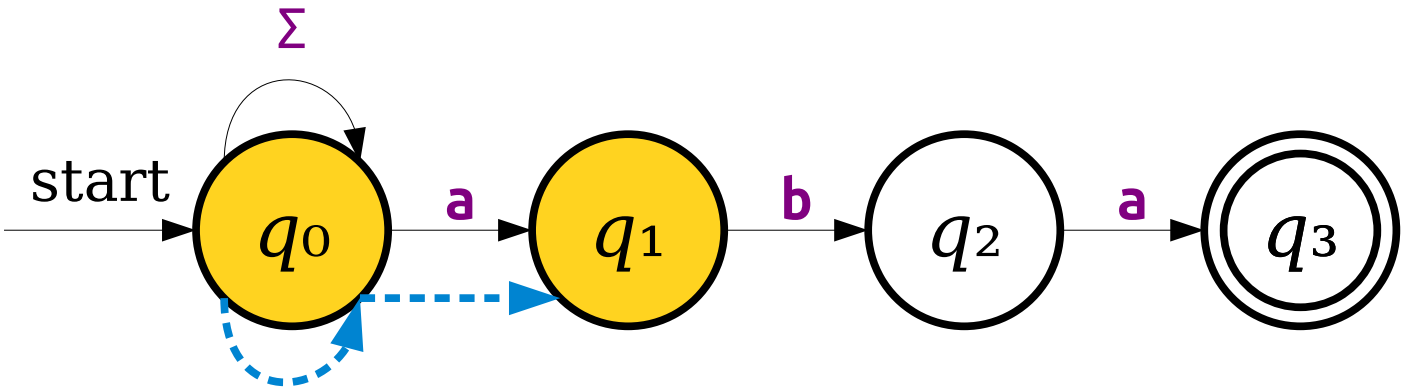


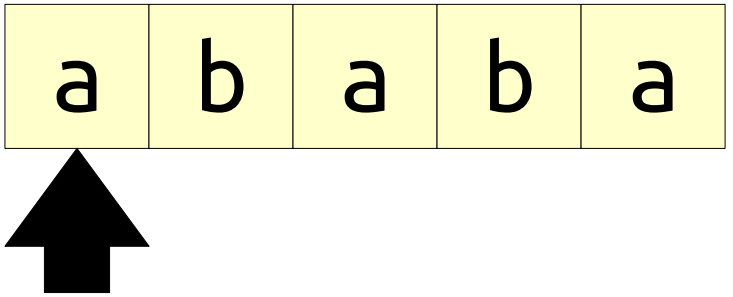
Massive Parallelism



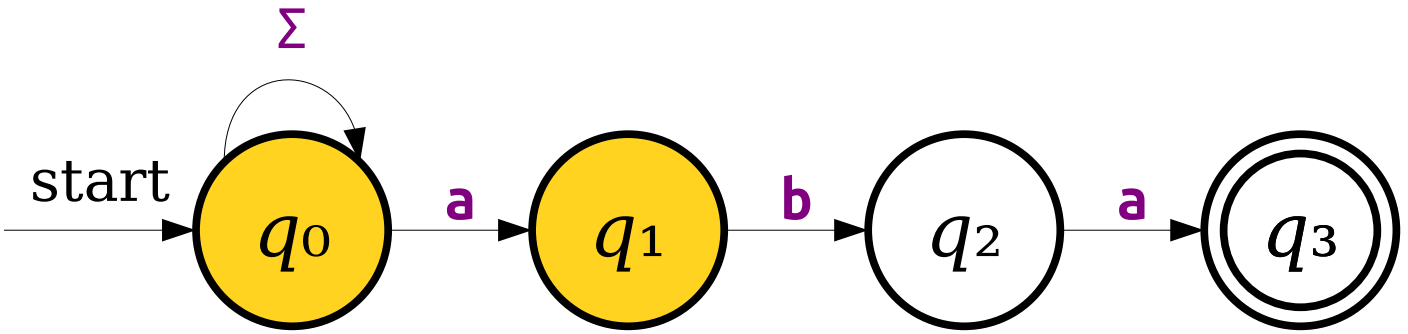


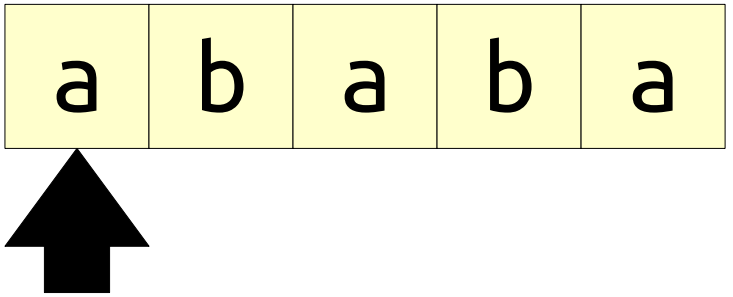
Massive Parallelism



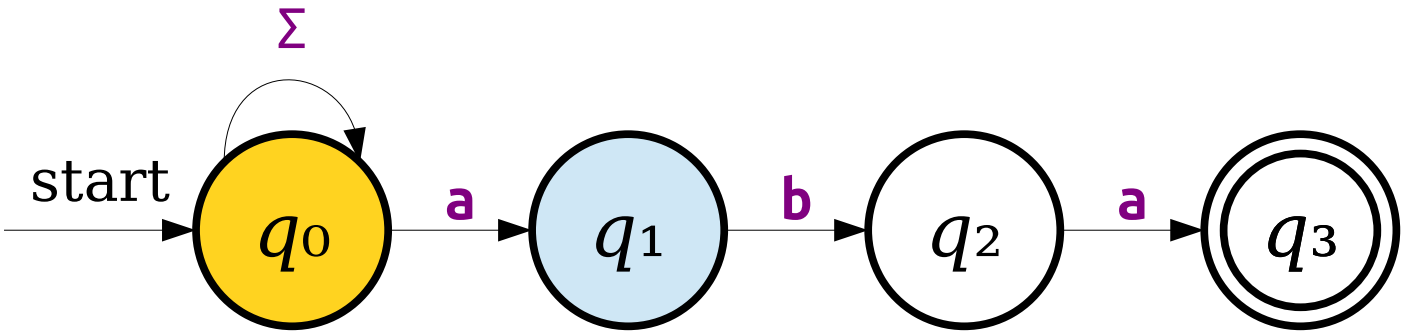


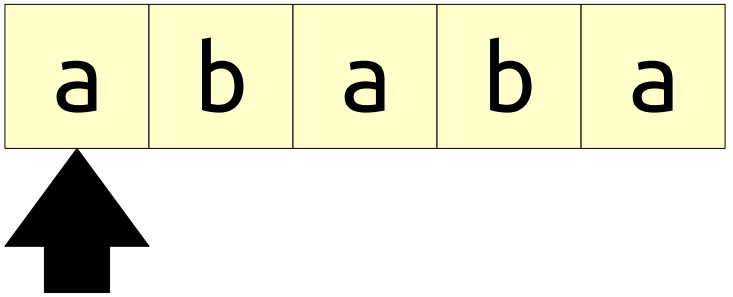
Massive Parallelism



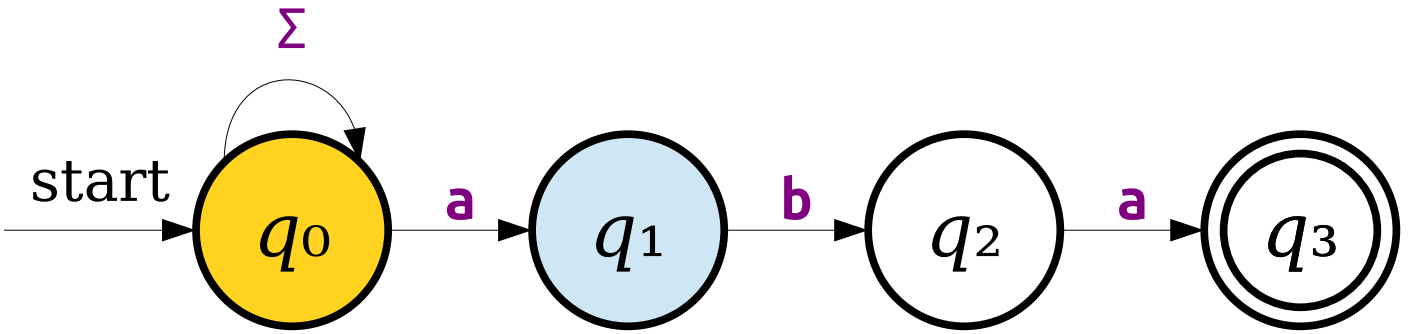


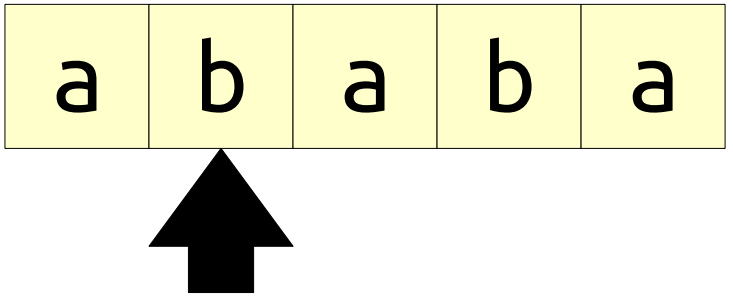
Massive Parallelism



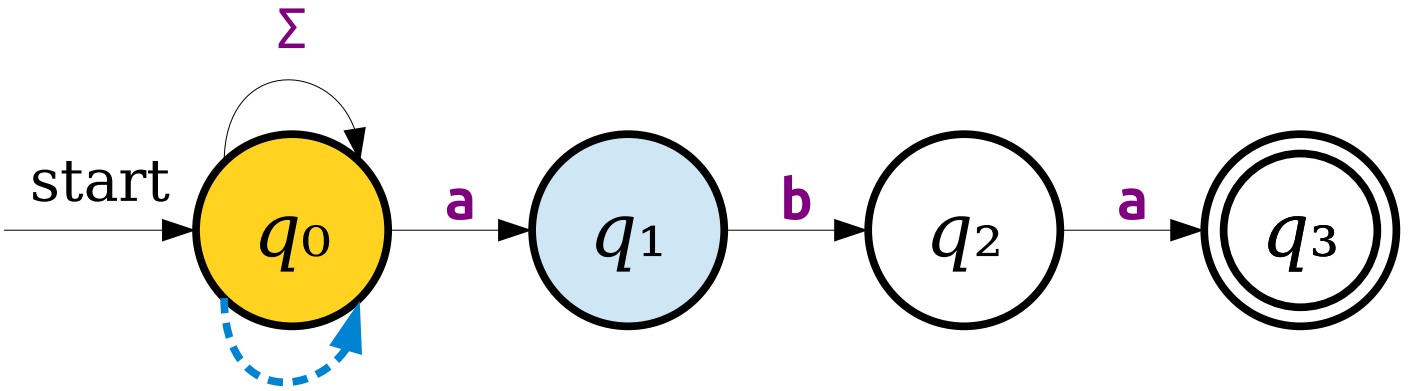


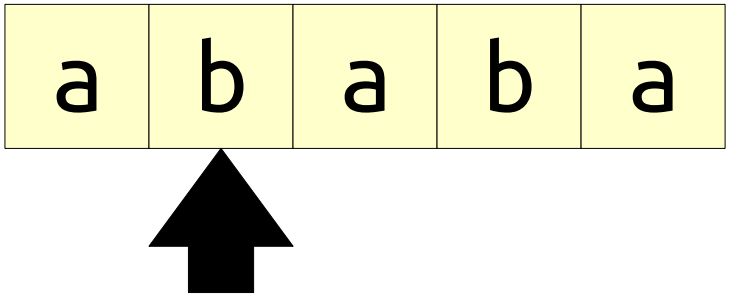
Massive Parallelism



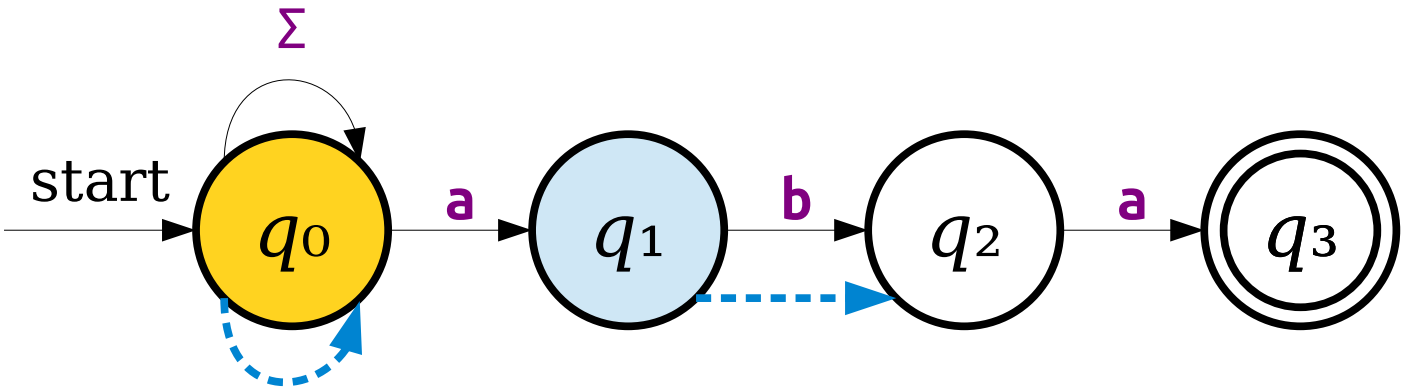


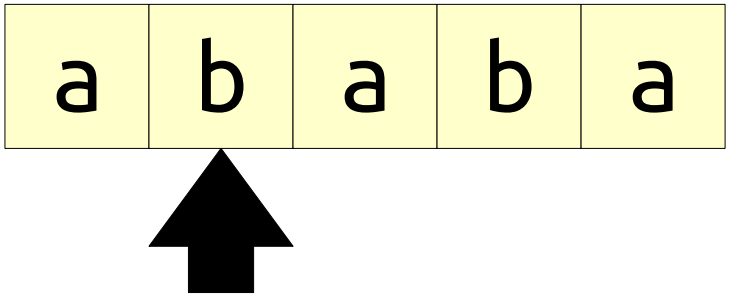
Massive Parallelism



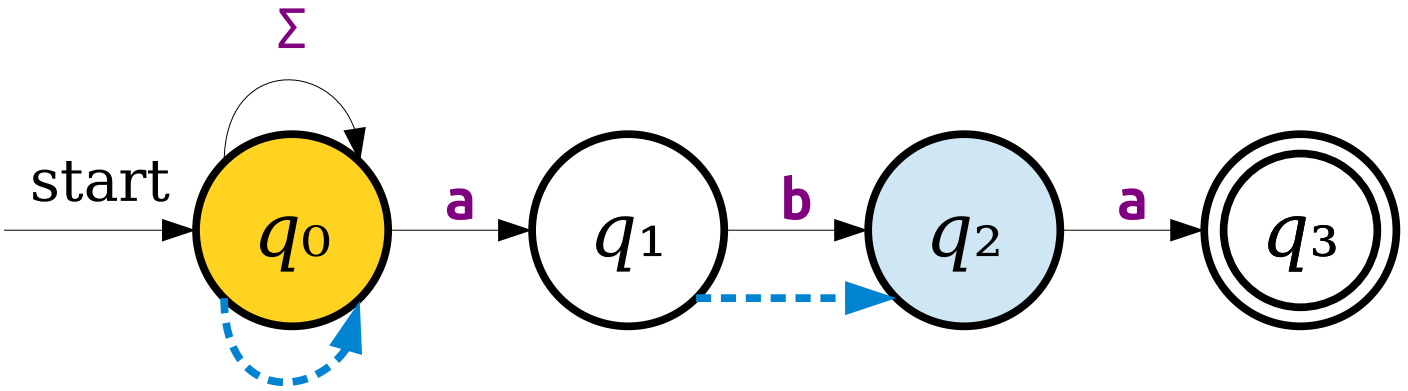


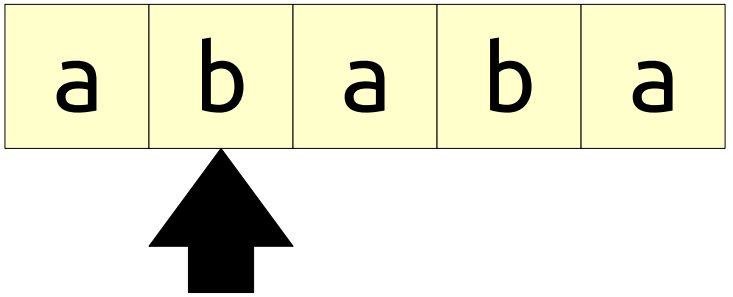
Massive Parallelism



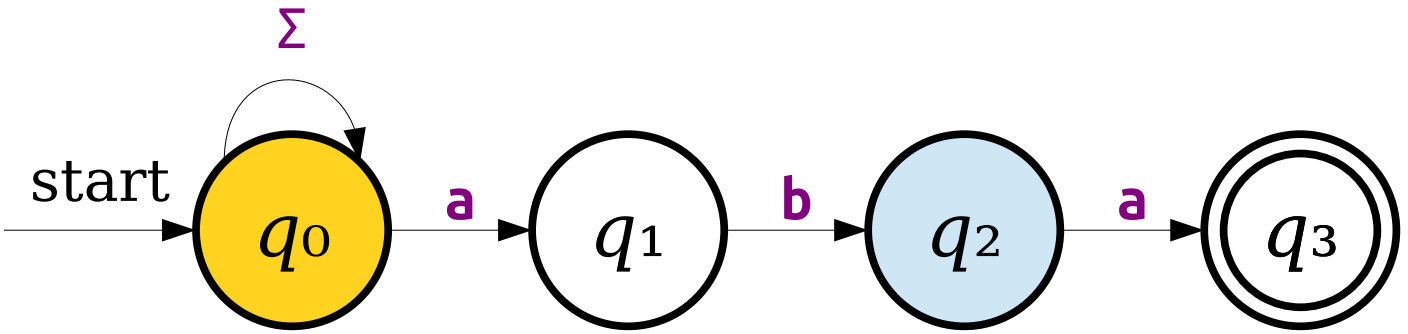


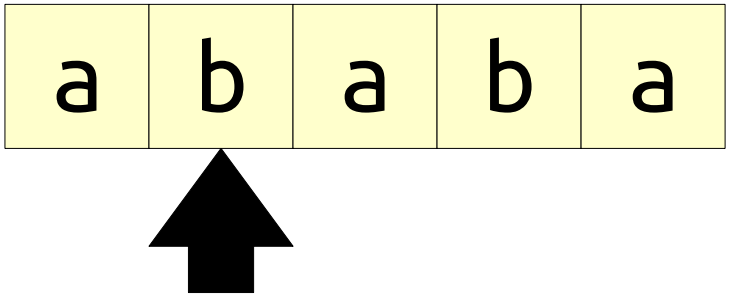
Massive Parallelism



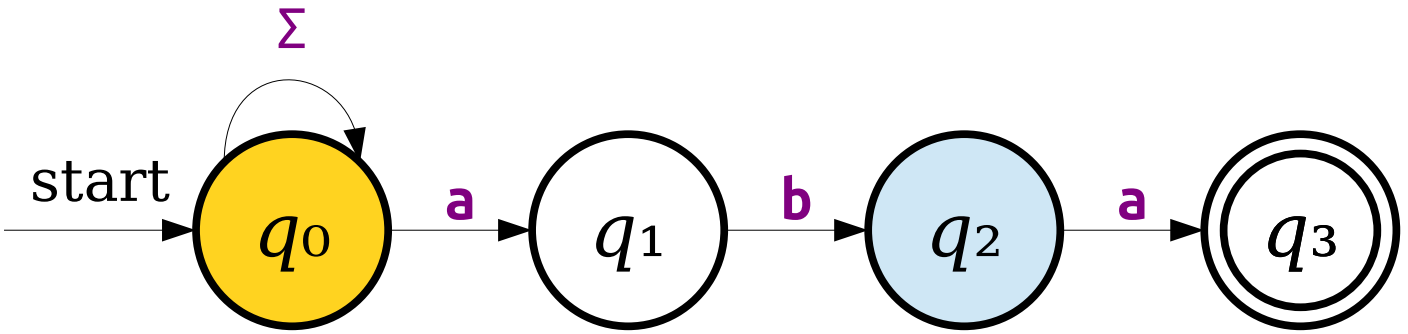


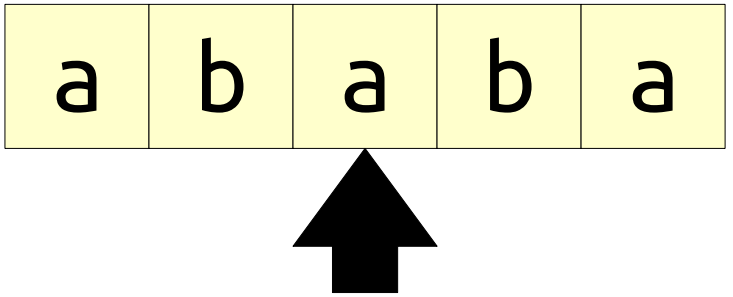
Massive Parallelism



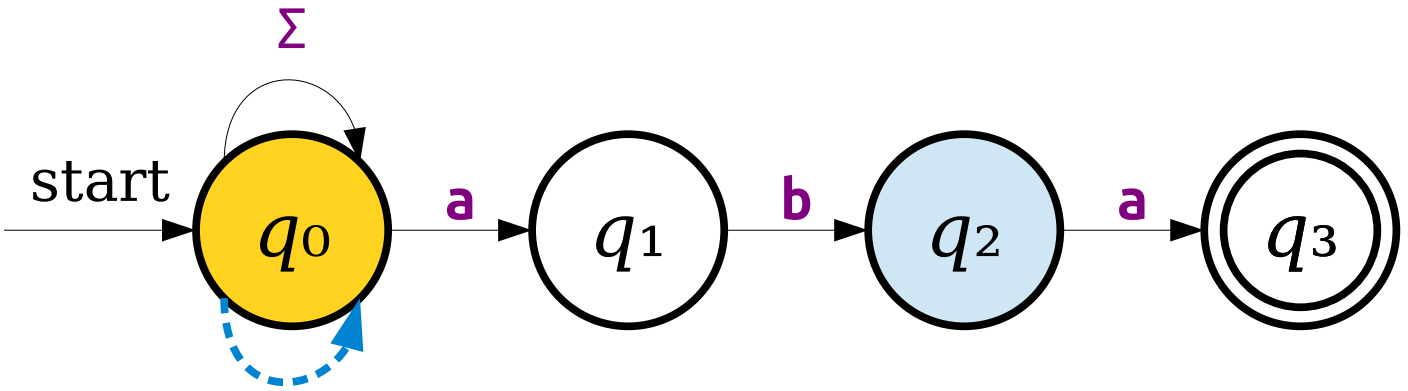


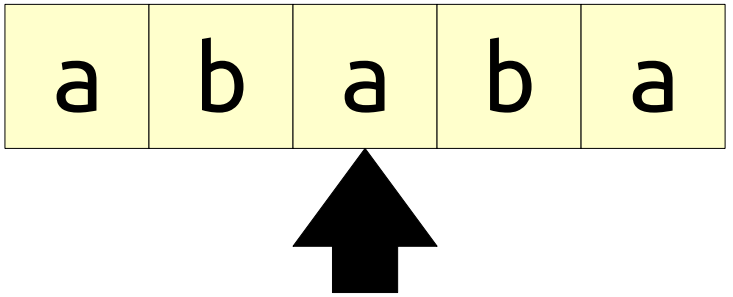
Massive Parallelism



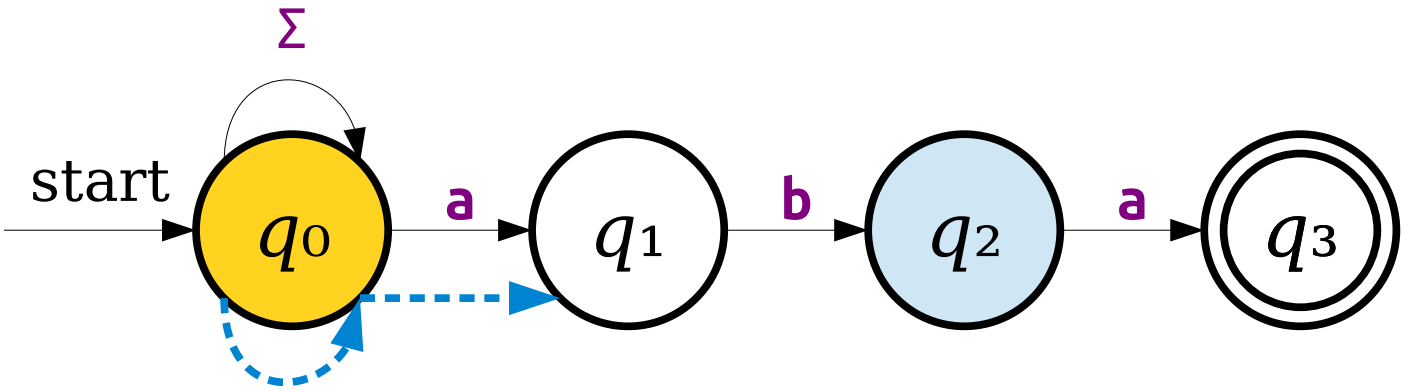


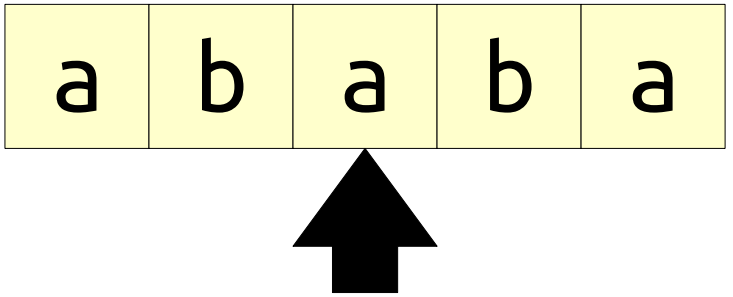
Massive Parallelism



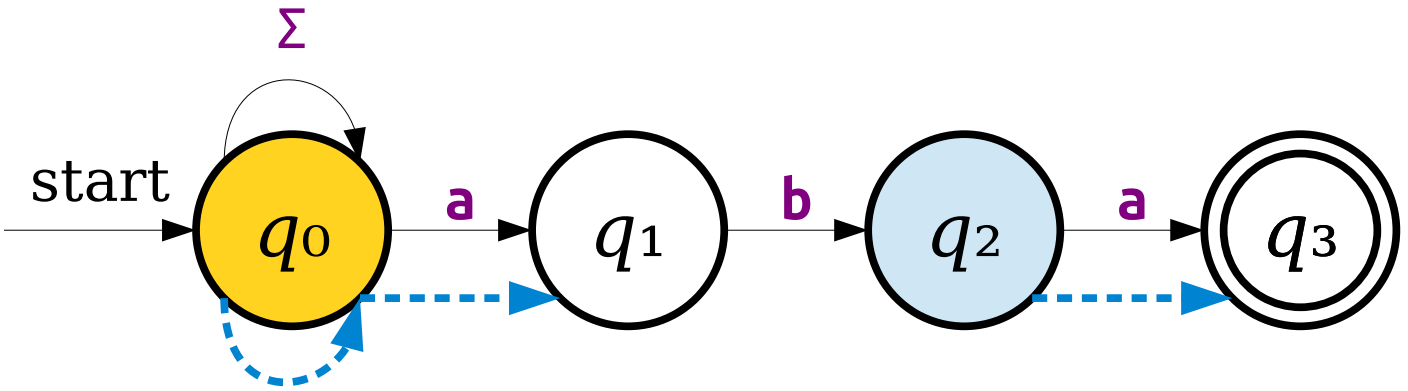


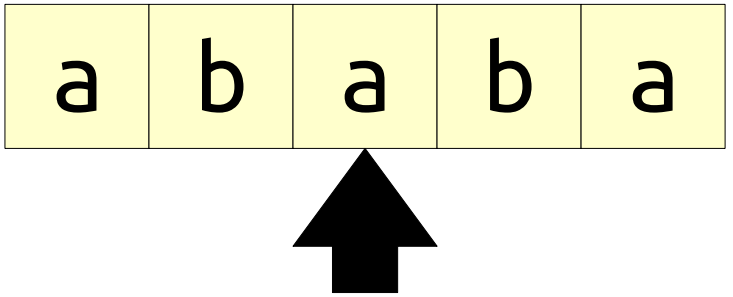
Massive Parallelism



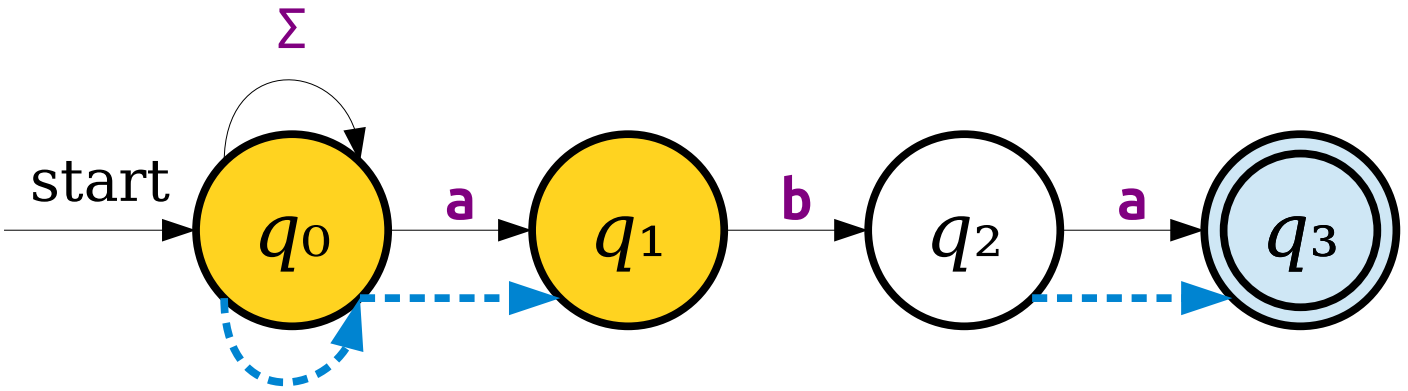


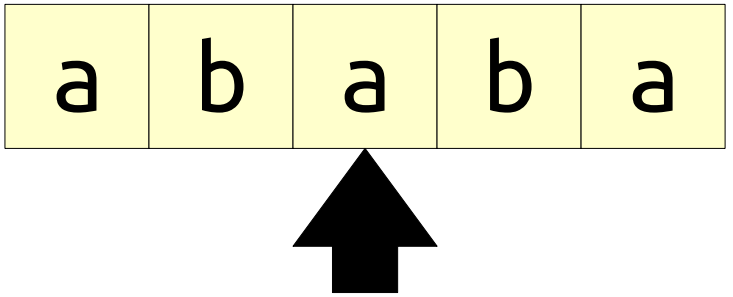
Massive Parallelism



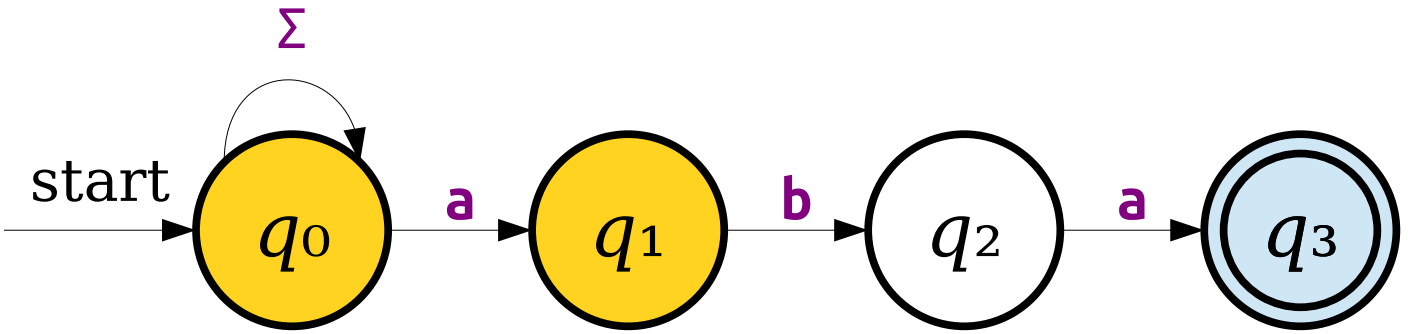


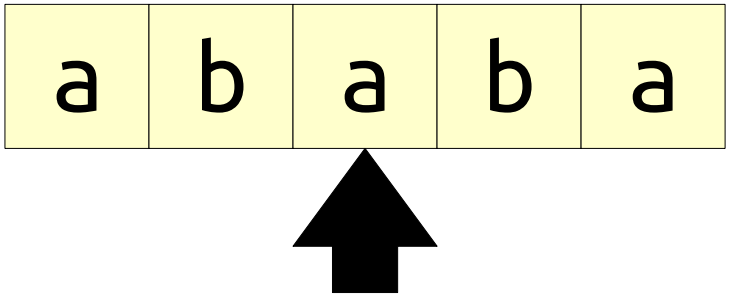
Massive Parallelism



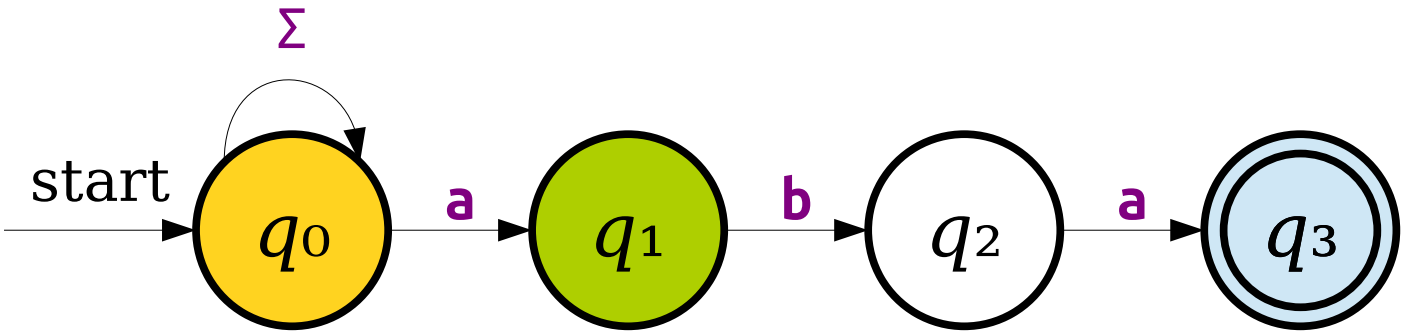


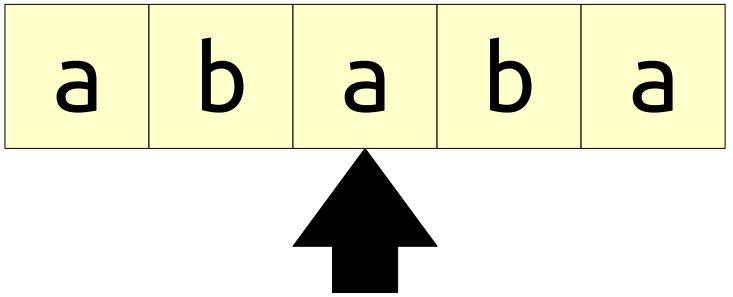
Massive Parallelism



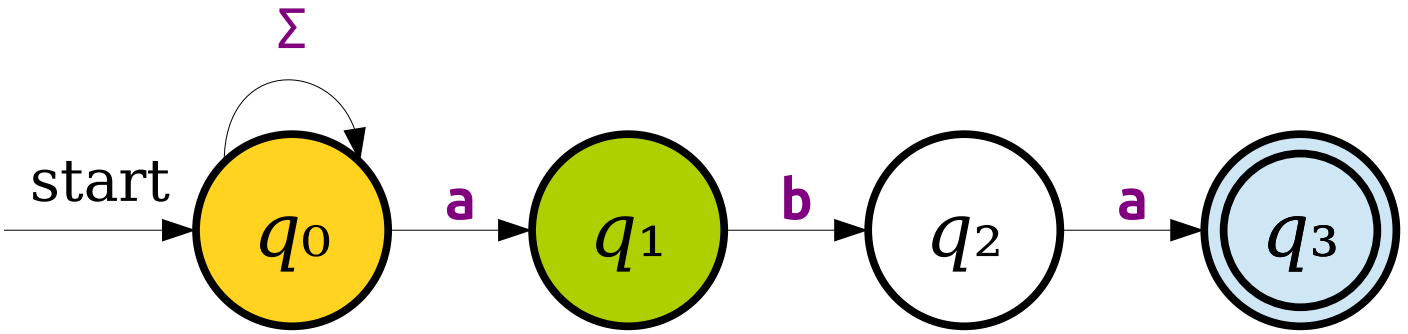


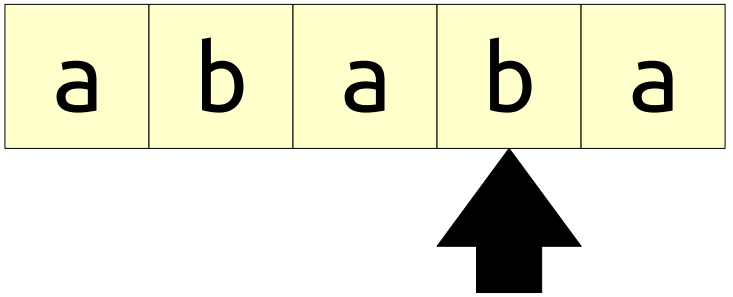
Massive Parallelism



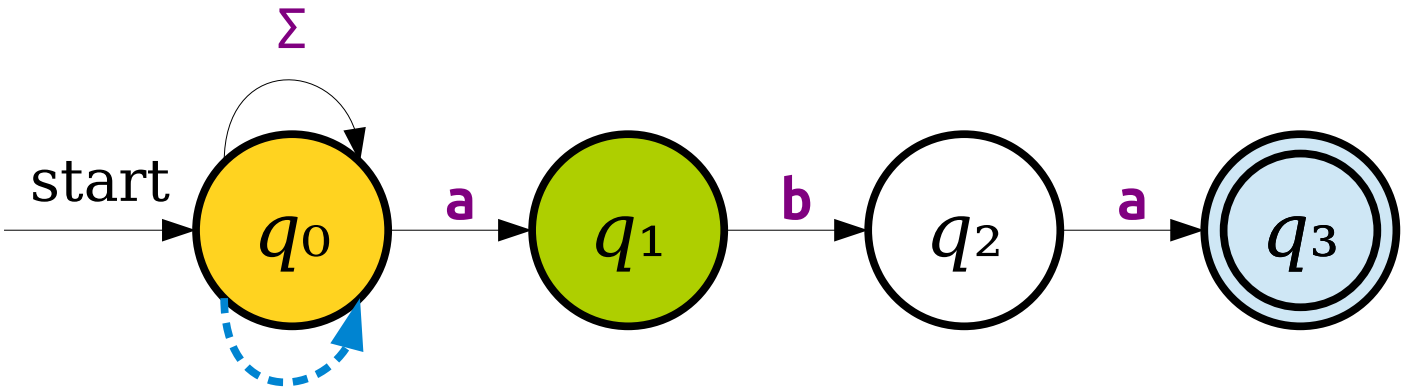


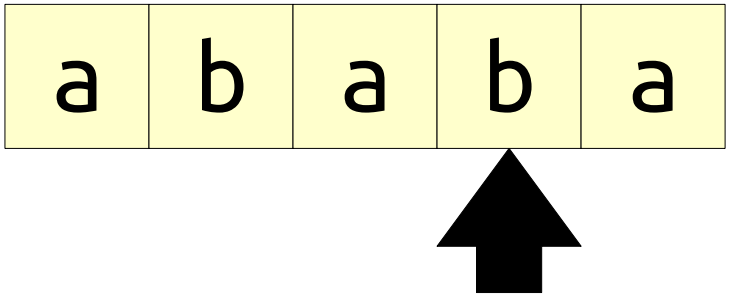
Massive Parallelism



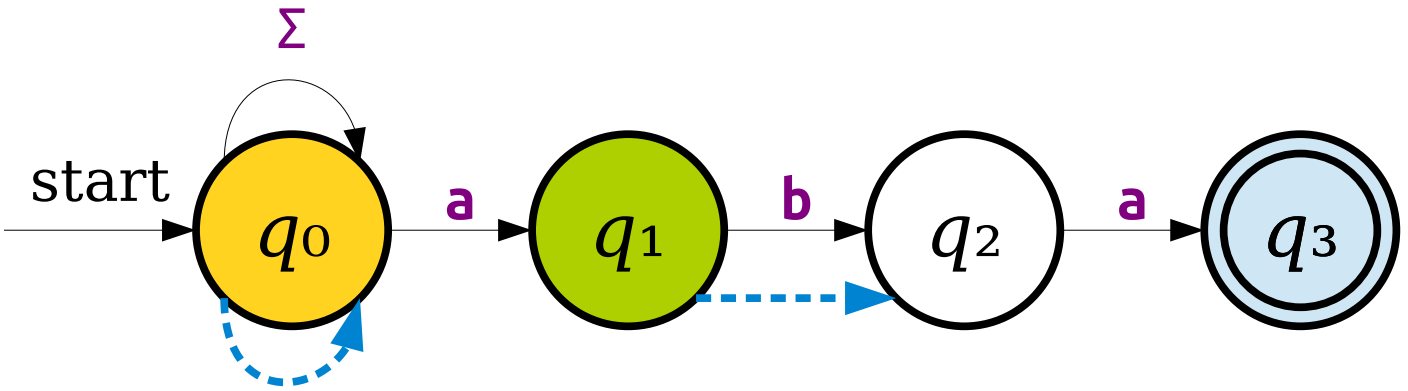


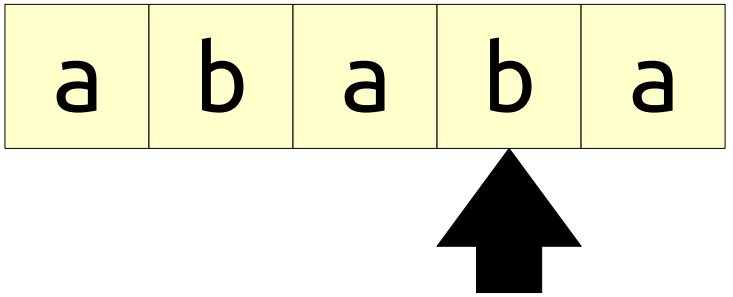
Massive Parallelism



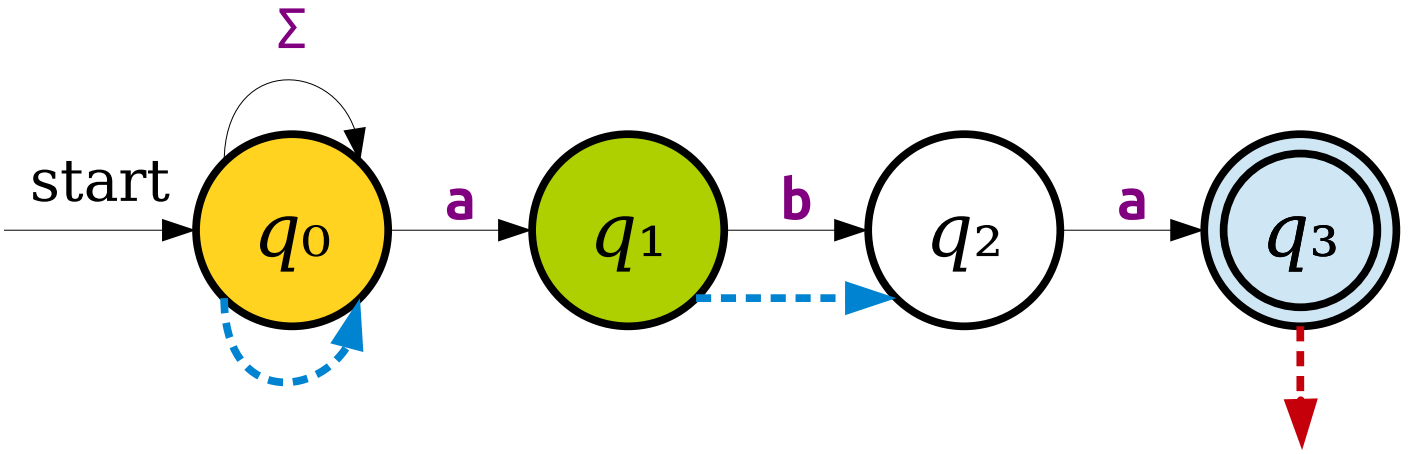


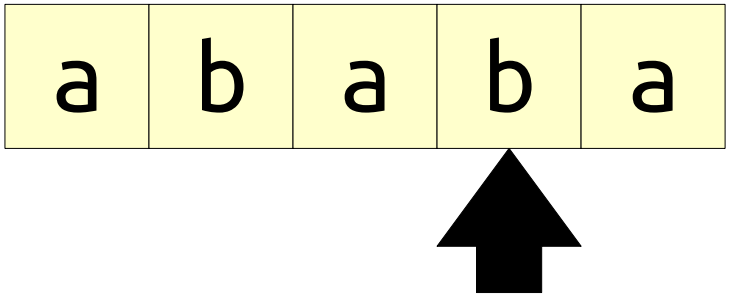
Massive Parallelism



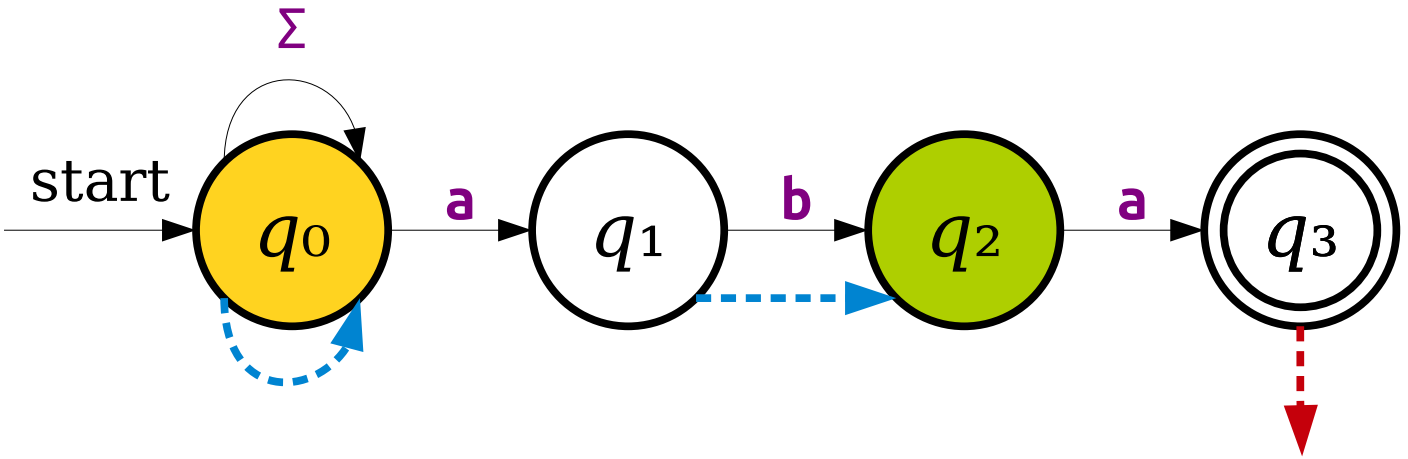


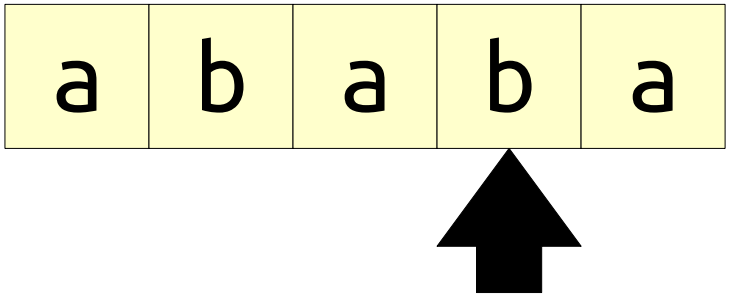
Massive Parallelism



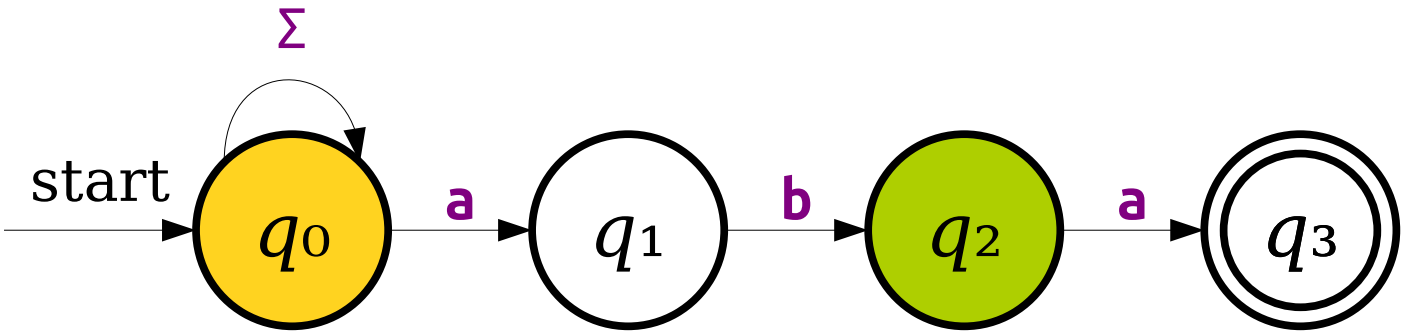


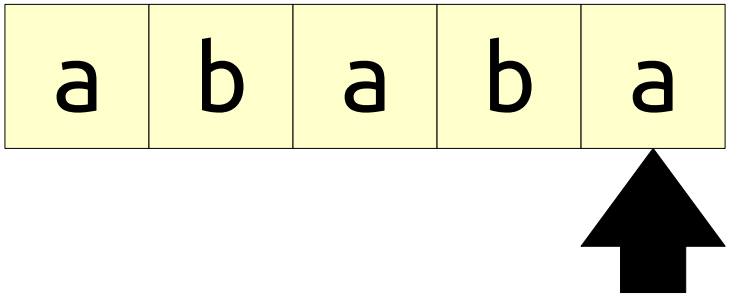
Massive Parallelism



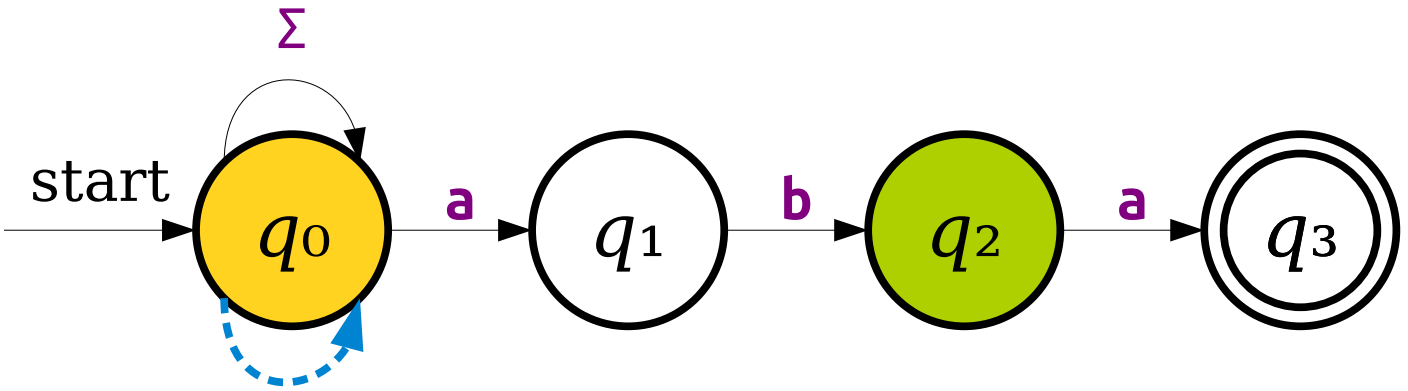


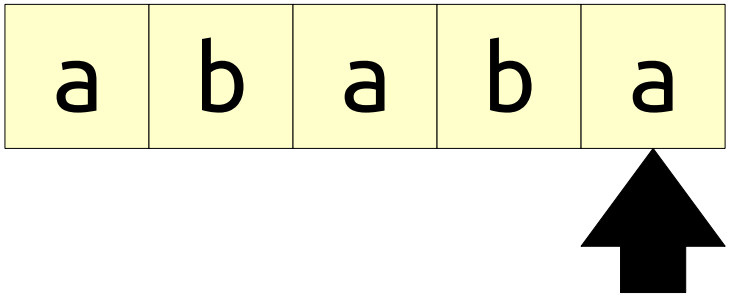
Massive Parallelism



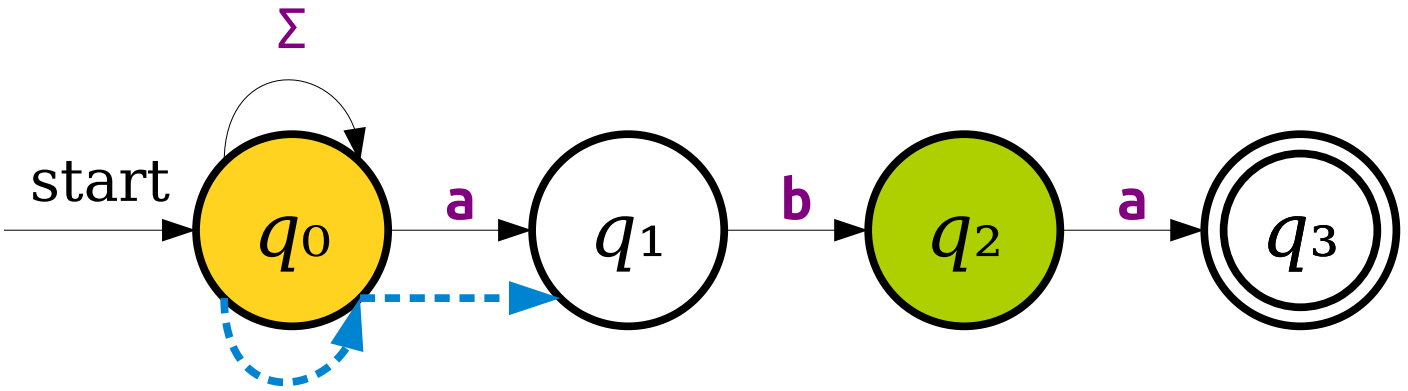


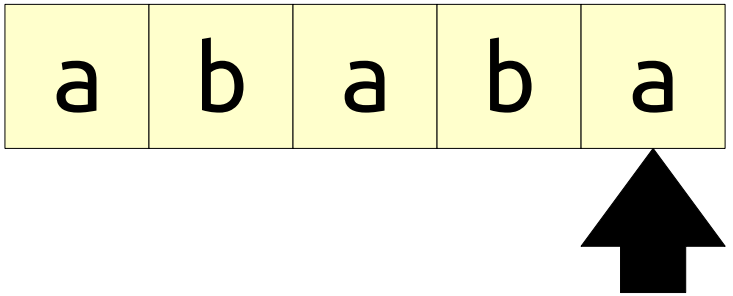
Massive Parallelism



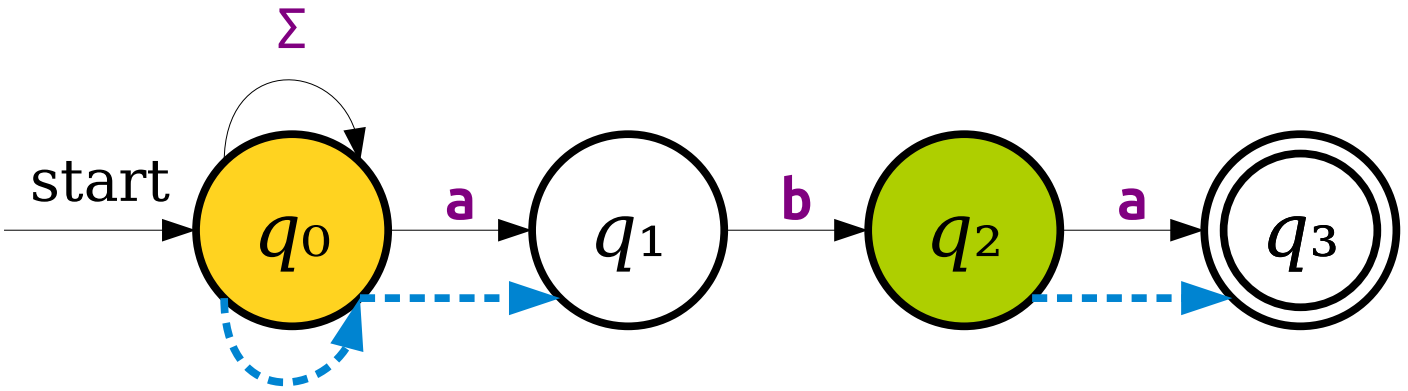


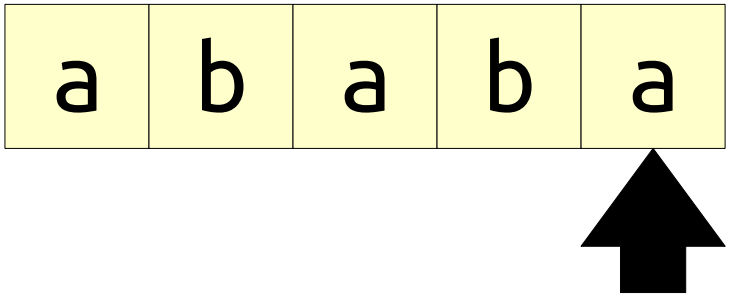
Massive Parallelism



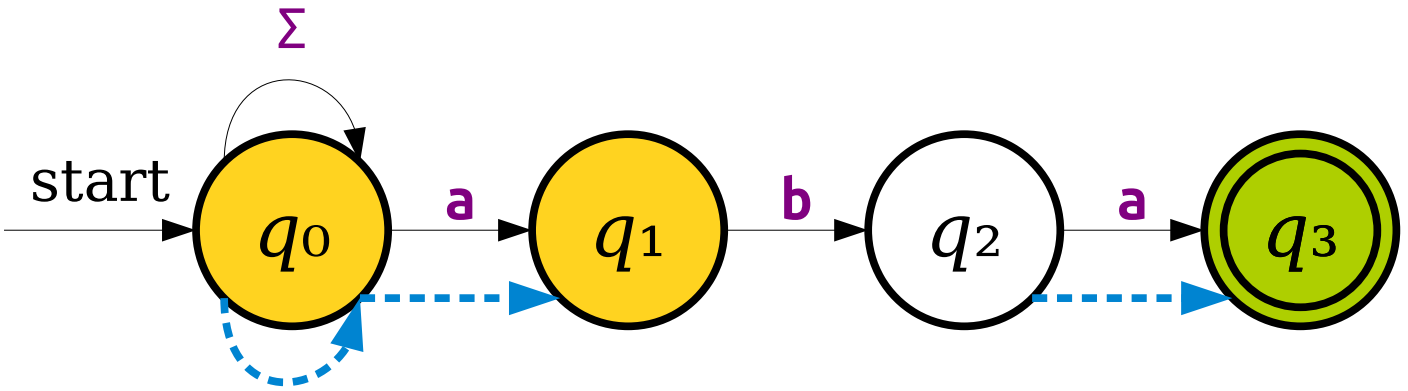


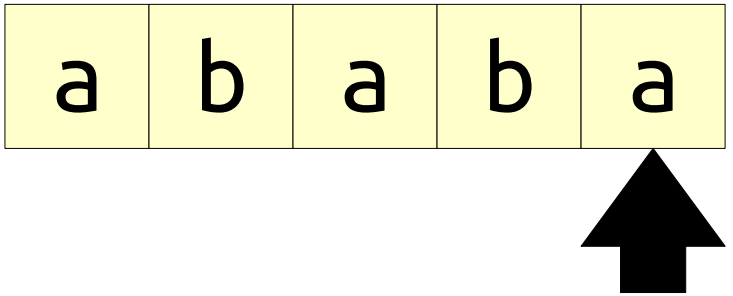
Massive Parallelism



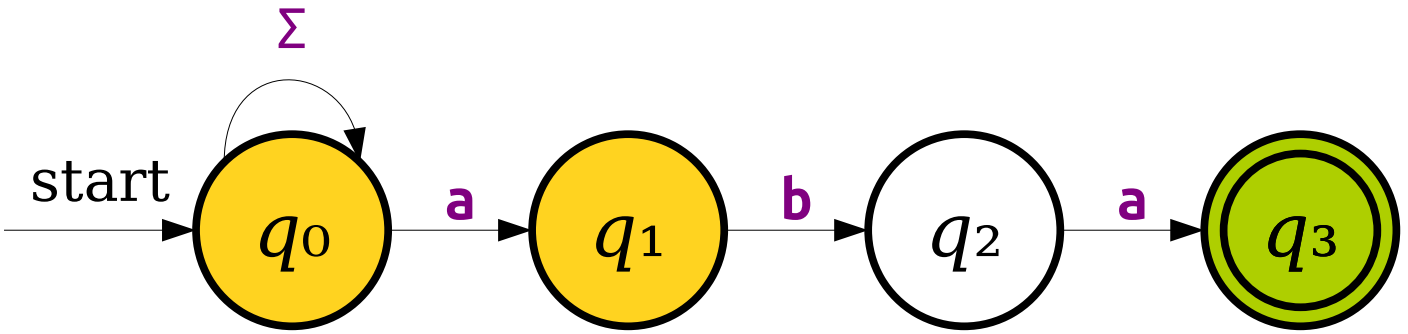


Massive Parallelism



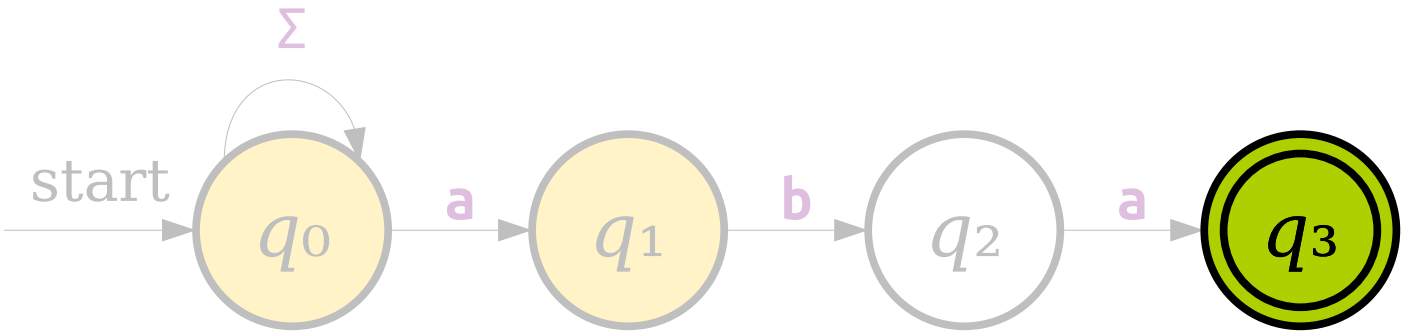


Massive Parallelism



a	b	a	b	a

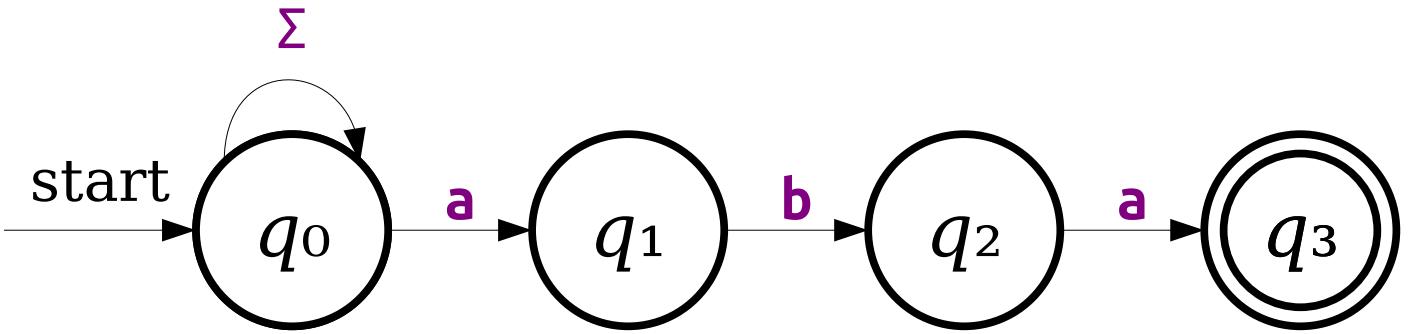
Massive Parallelism

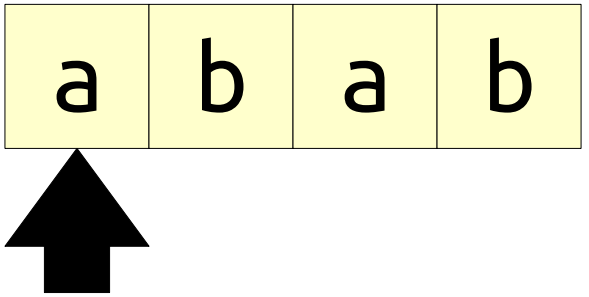


We're in at least one accepting state, so there's some path that gets us to an accepting state.

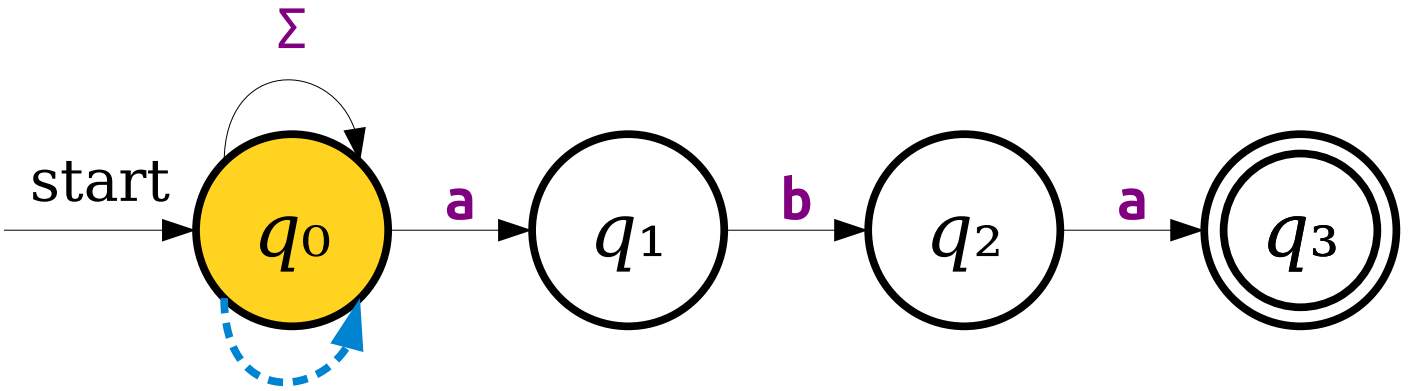
a	b	a	b	a

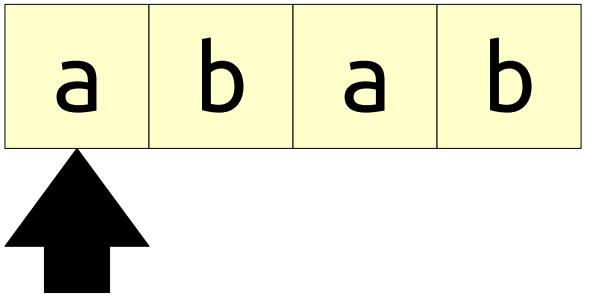
Massive Parallelism



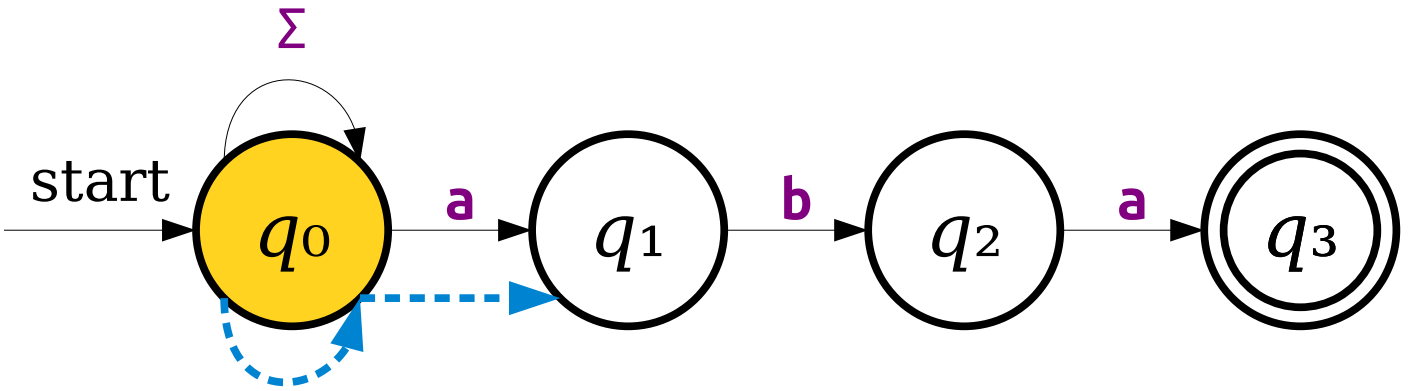


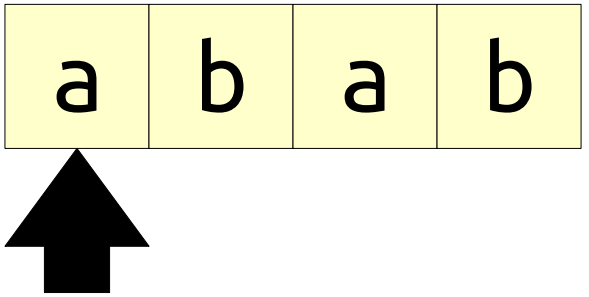
Massive Parallelism



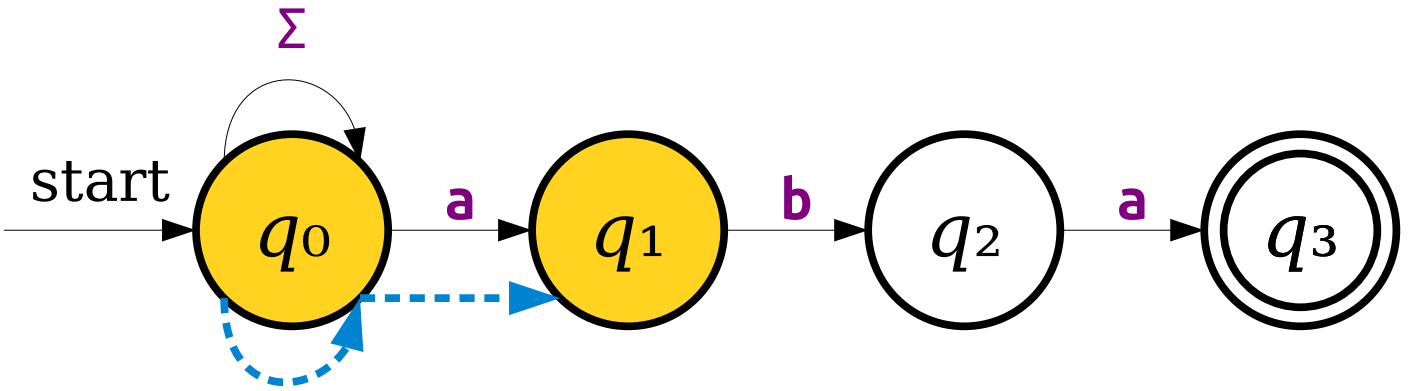


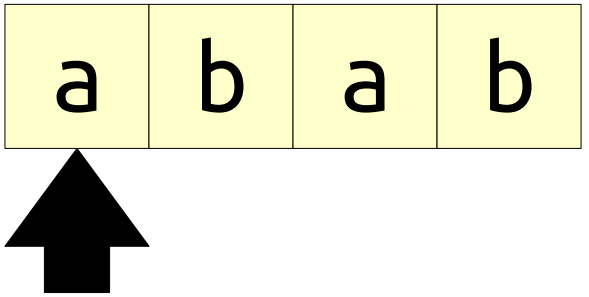
Massive Parallelism



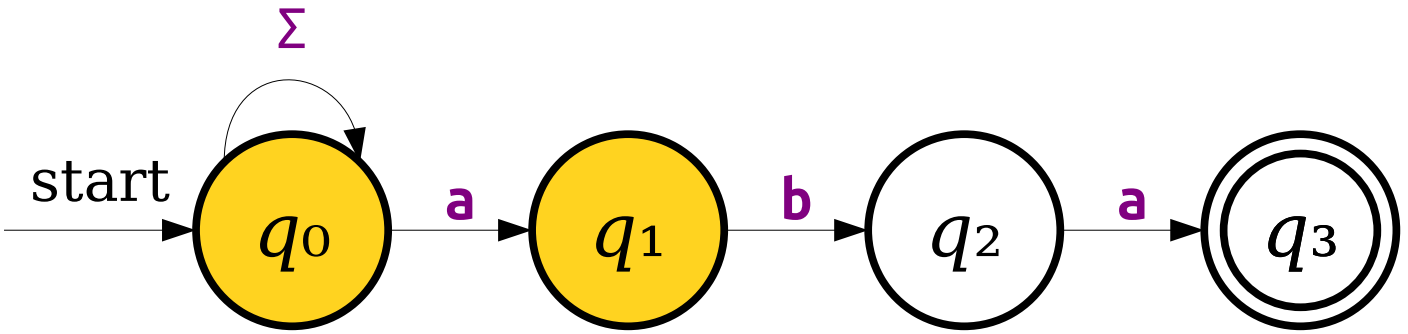


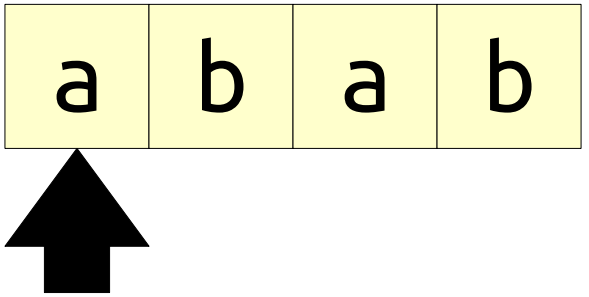
Massive Parallelism



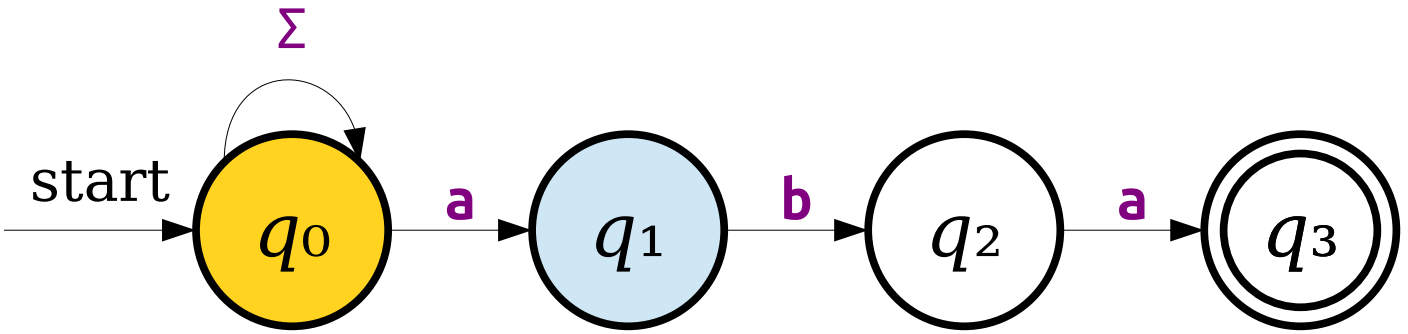


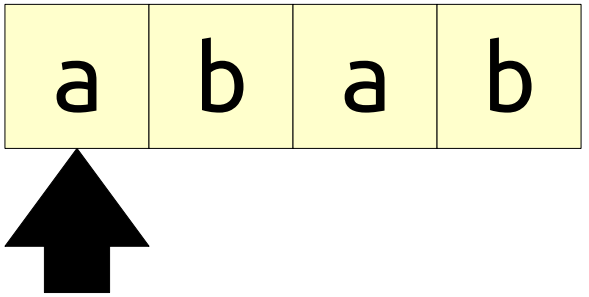
Massive Parallelism



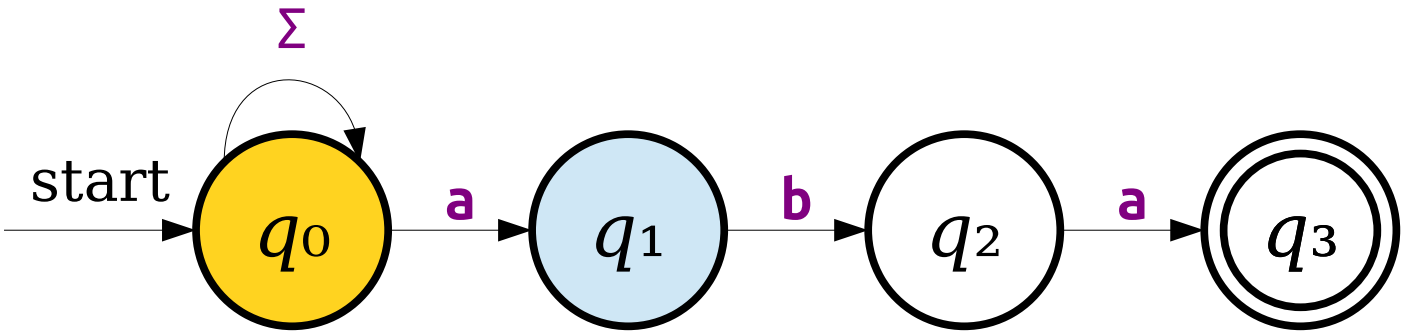


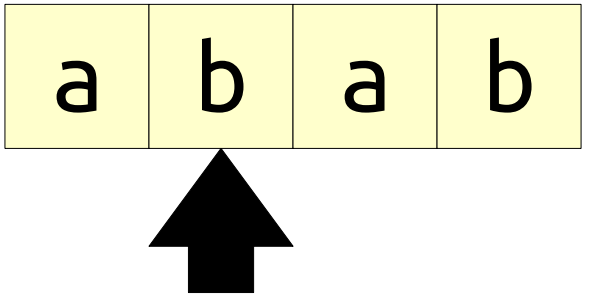
Massive Parallelism



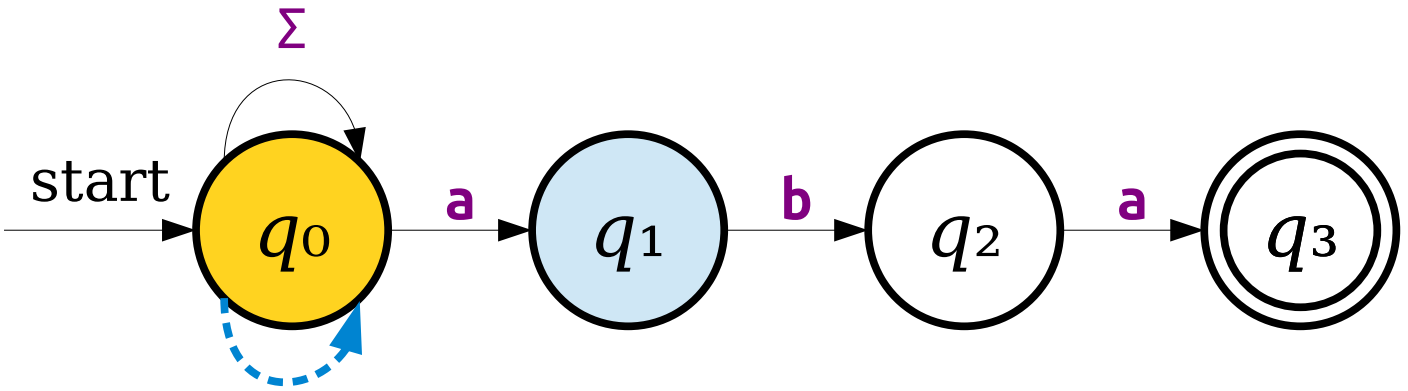


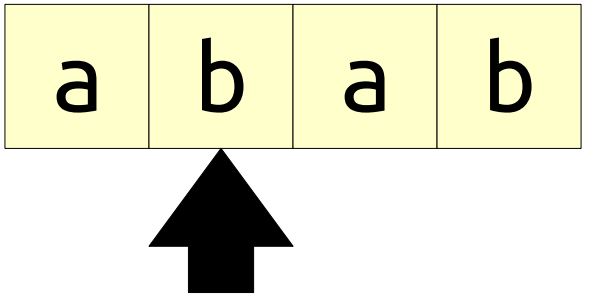
Massive Parallelism



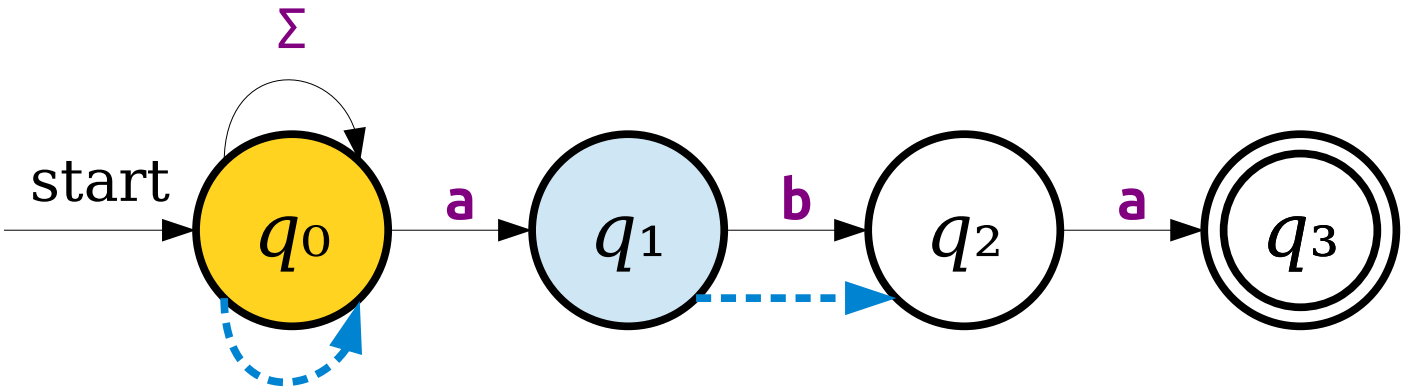


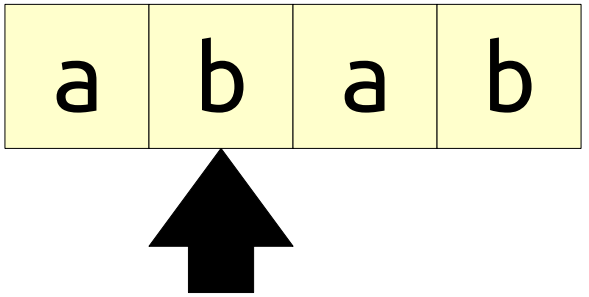
Massive Parallelism



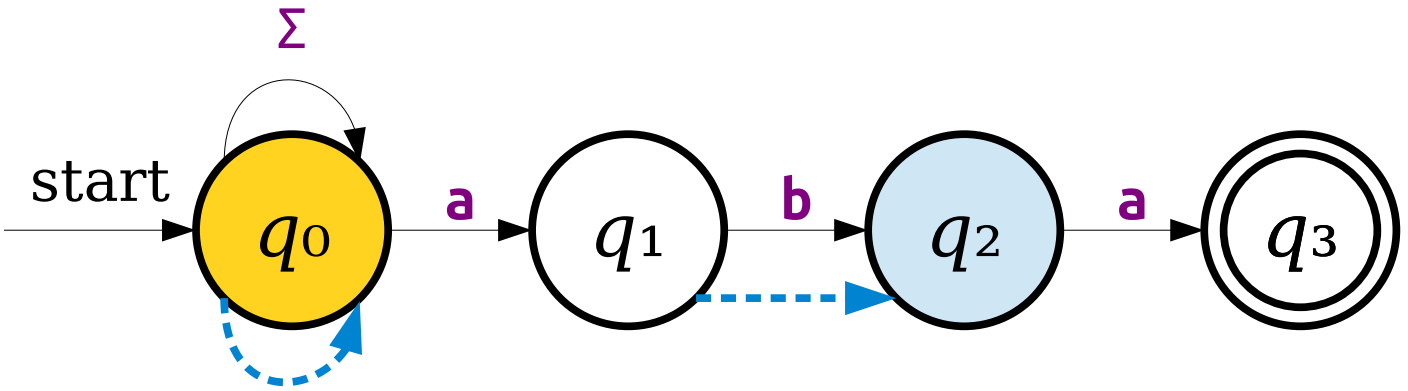


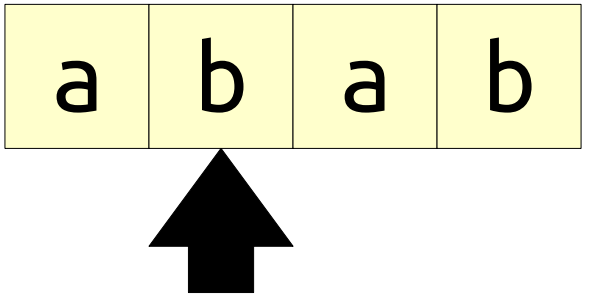
Massive Parallelism



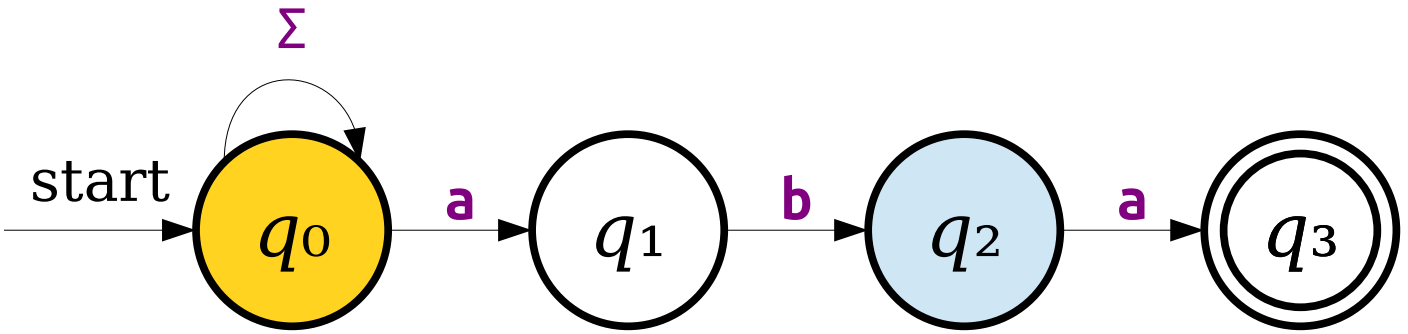


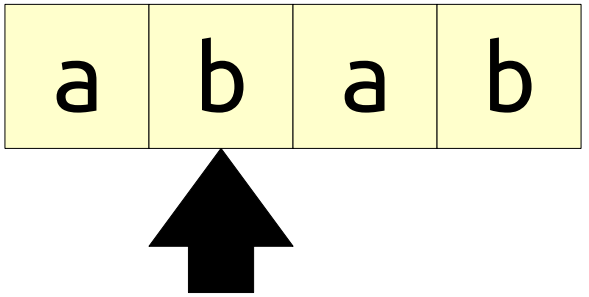
Massive Parallelism



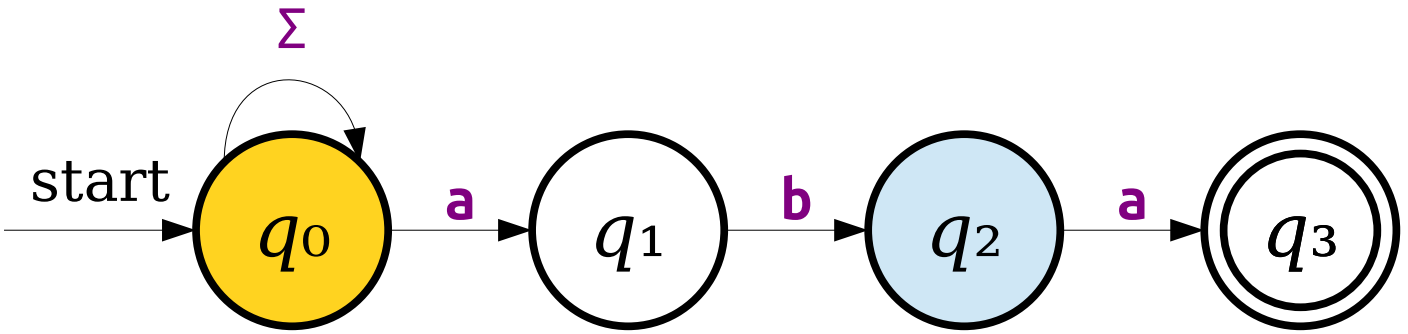


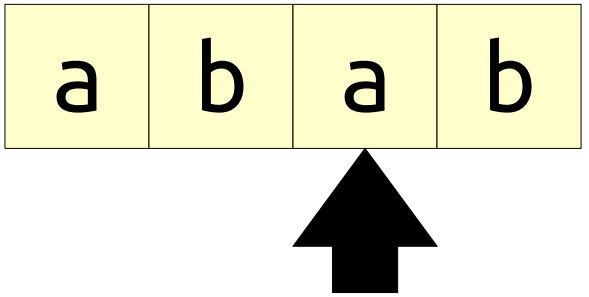
Massive Parallelism



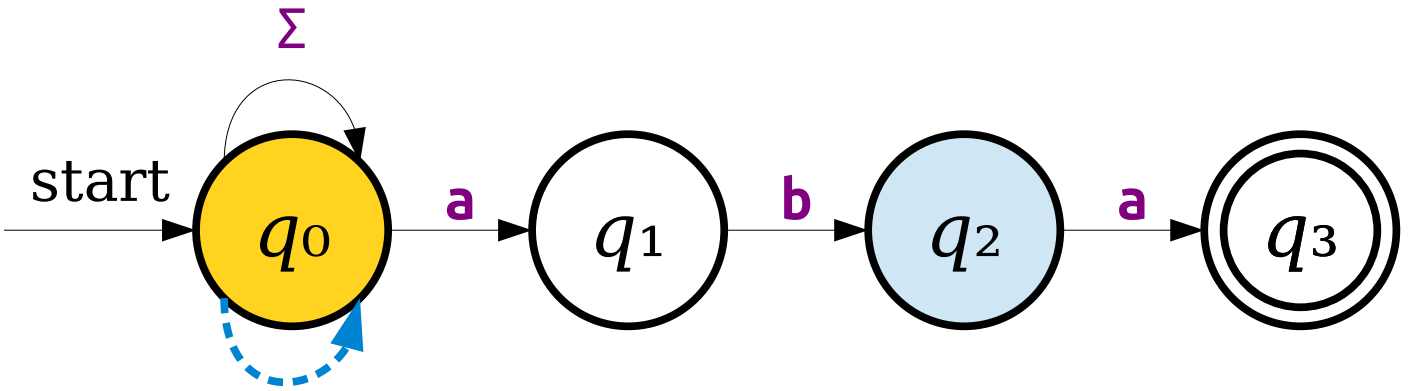


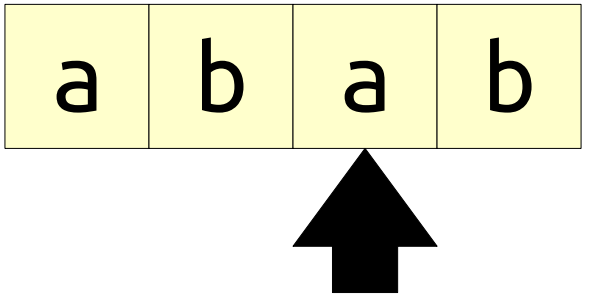
Massive Parallelism



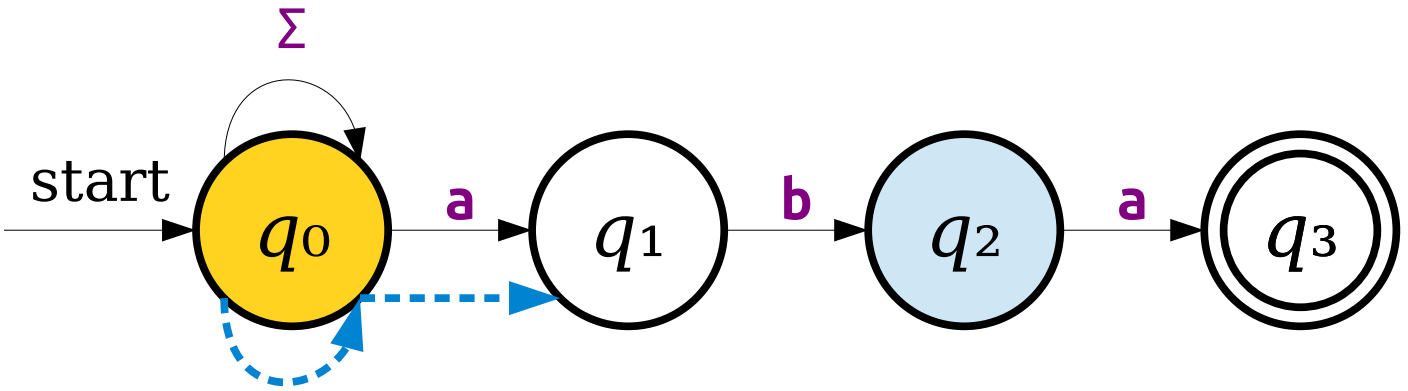


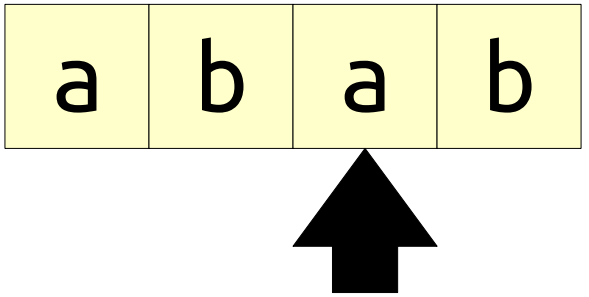
Massive Parallelism



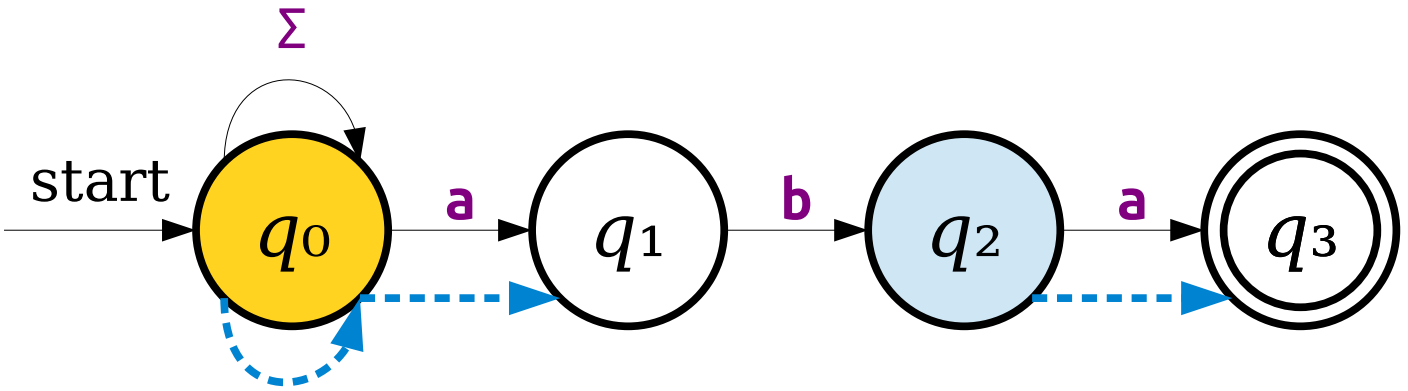


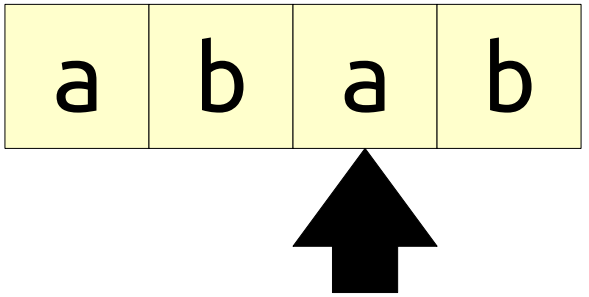
Massive Parallelism



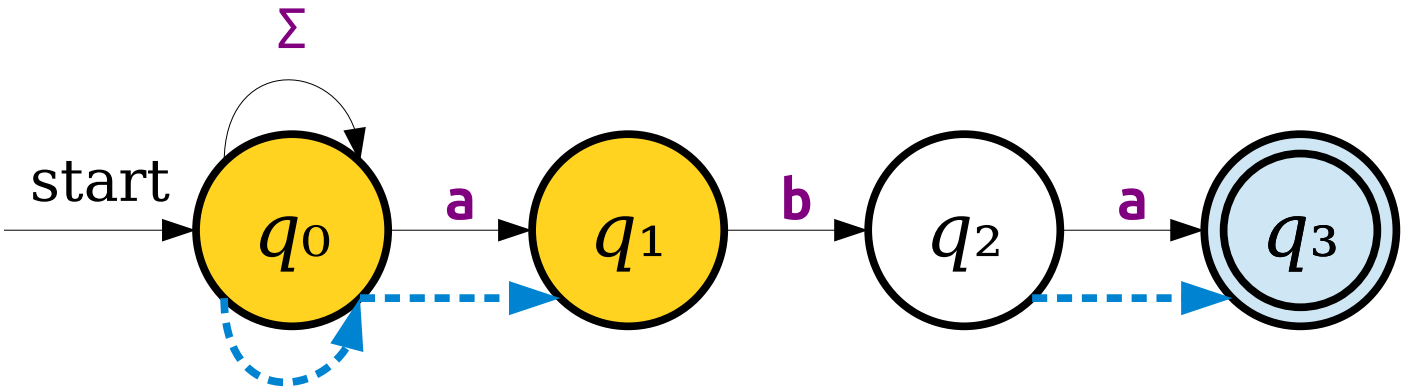


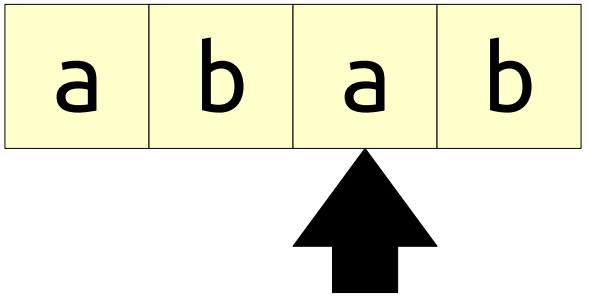
Massive Parallelism



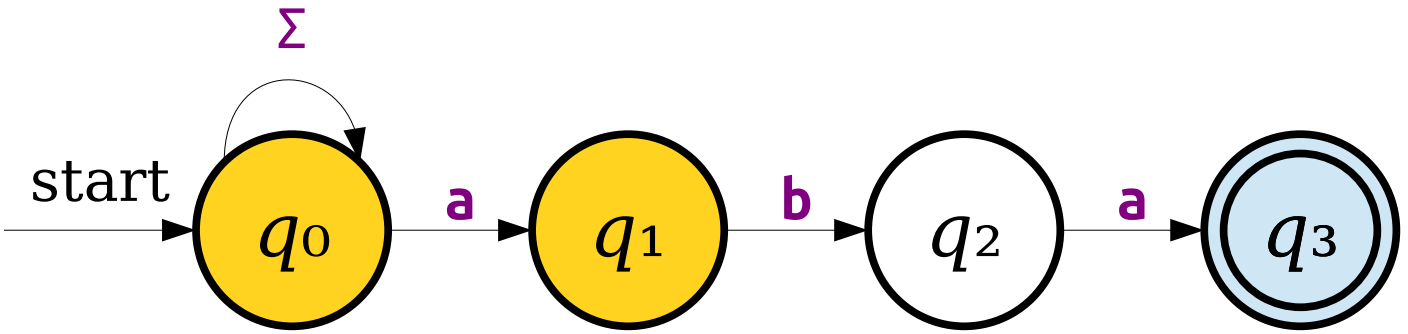


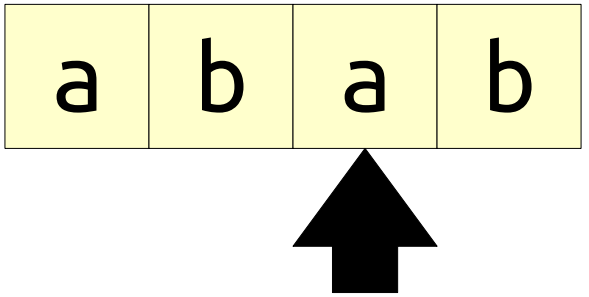
Massive Parallelism



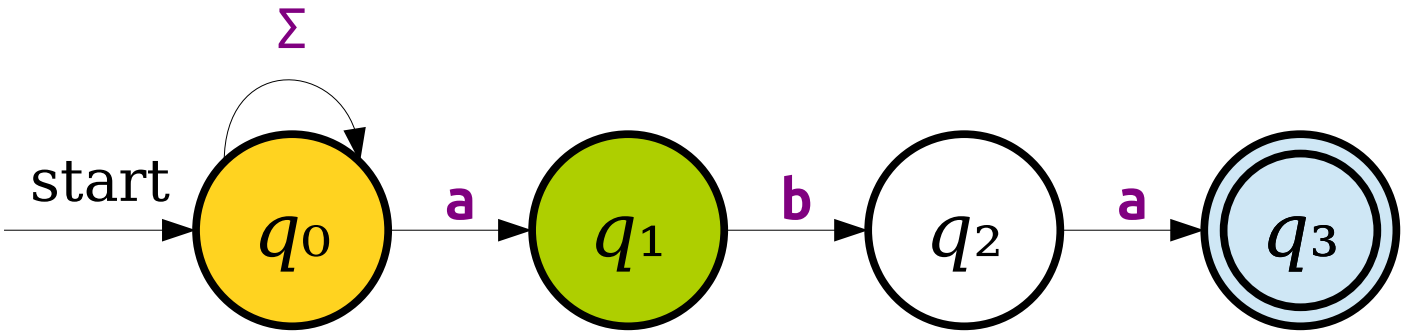


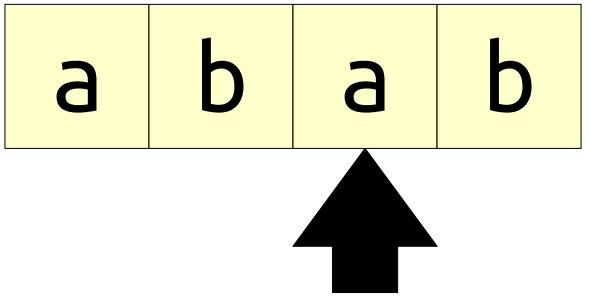
Massive Parallelism



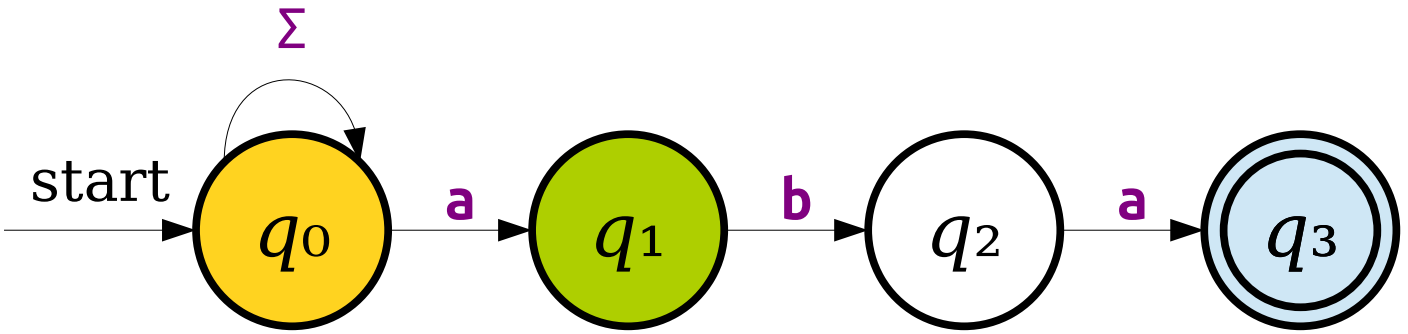


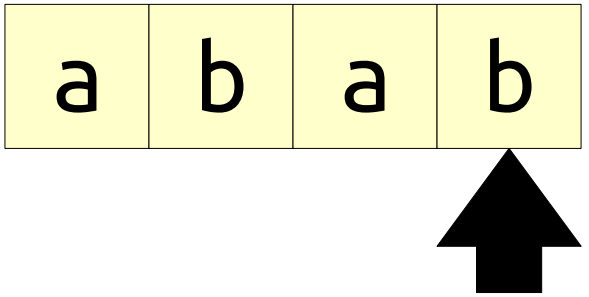
Massive Parallelism



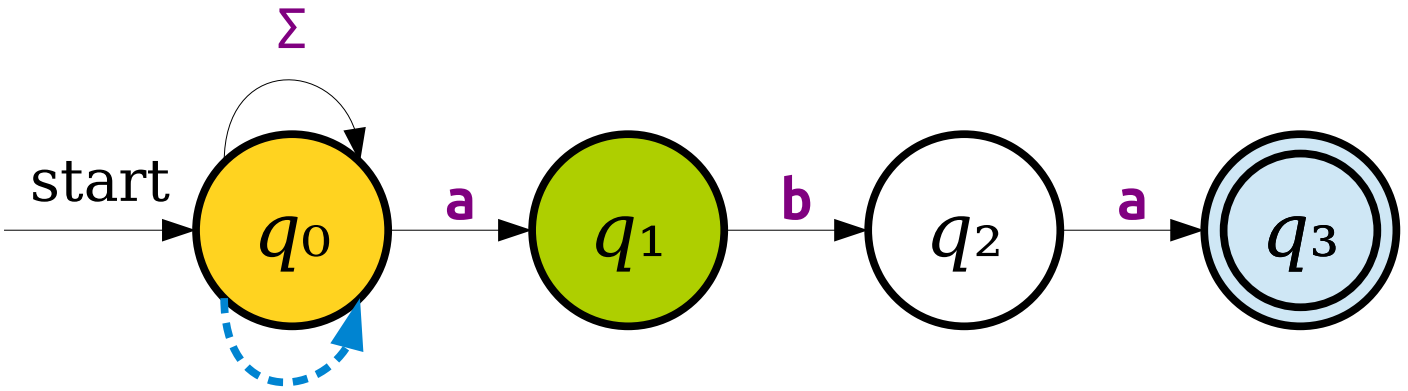


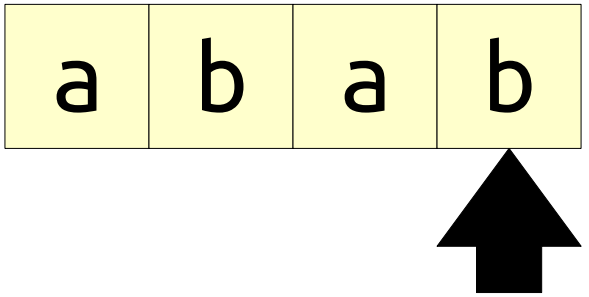
Massive Parallelism



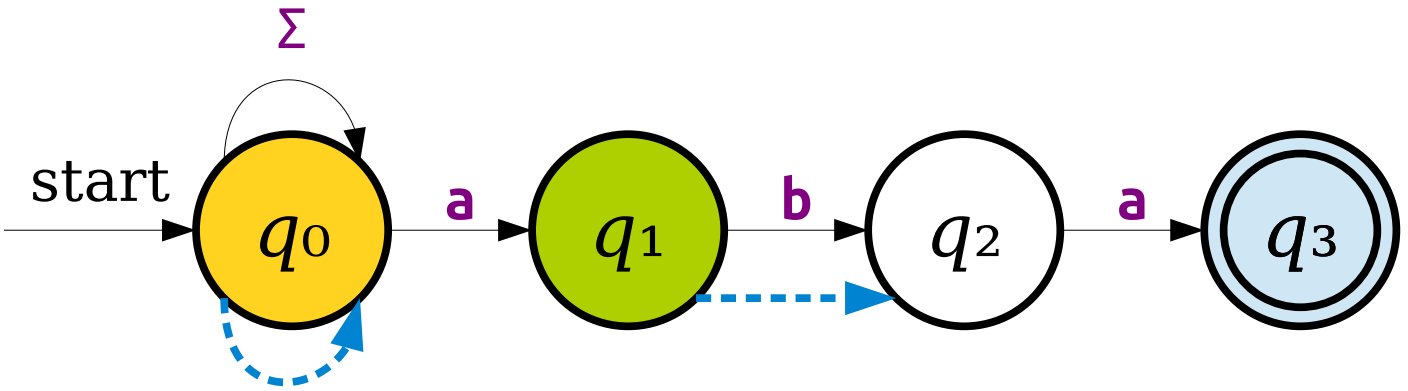


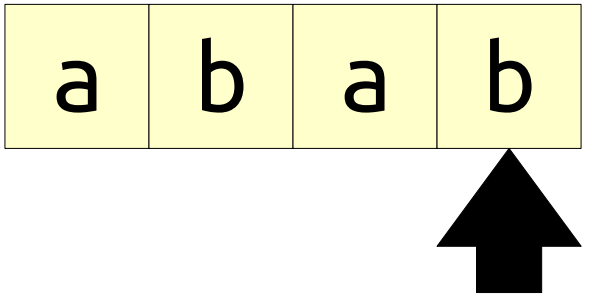
Massive Parallelism



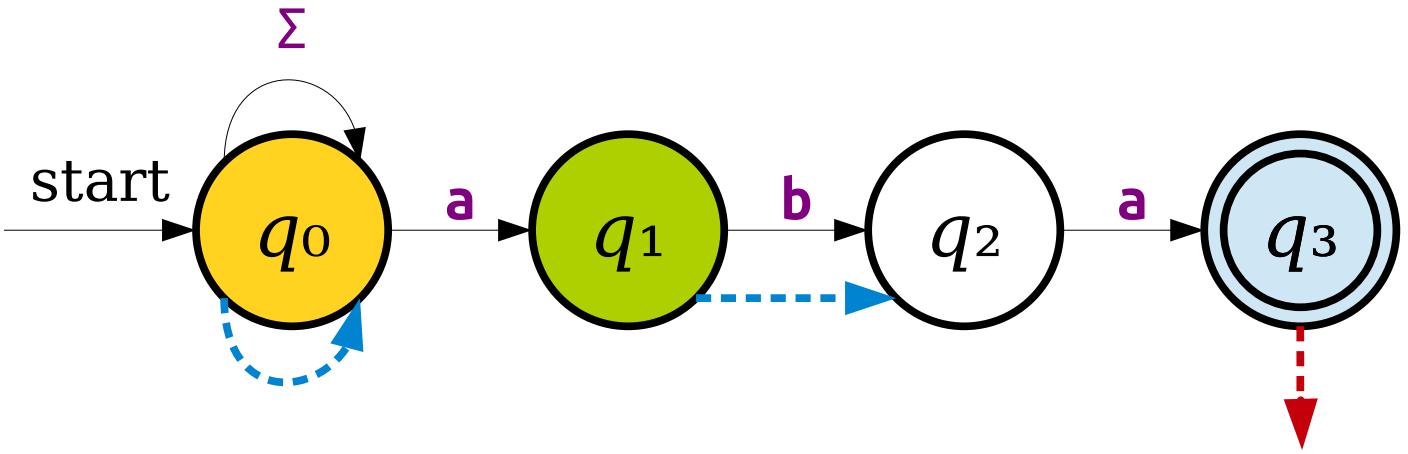


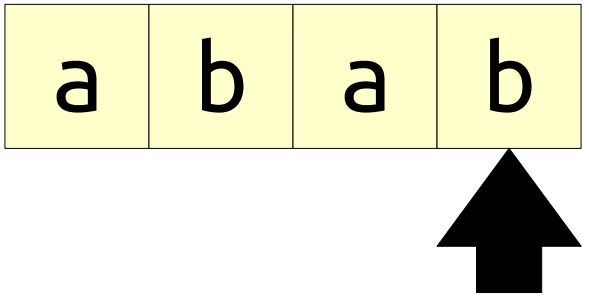
Massive Parallelism



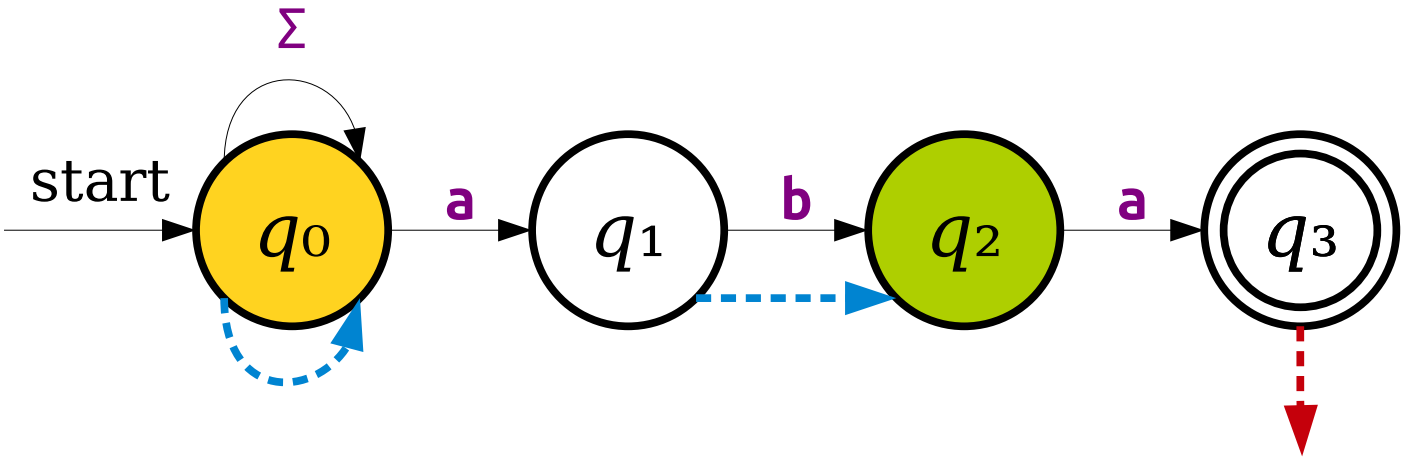


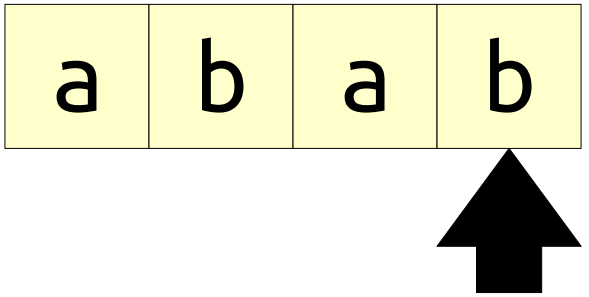
Massive Parallelism



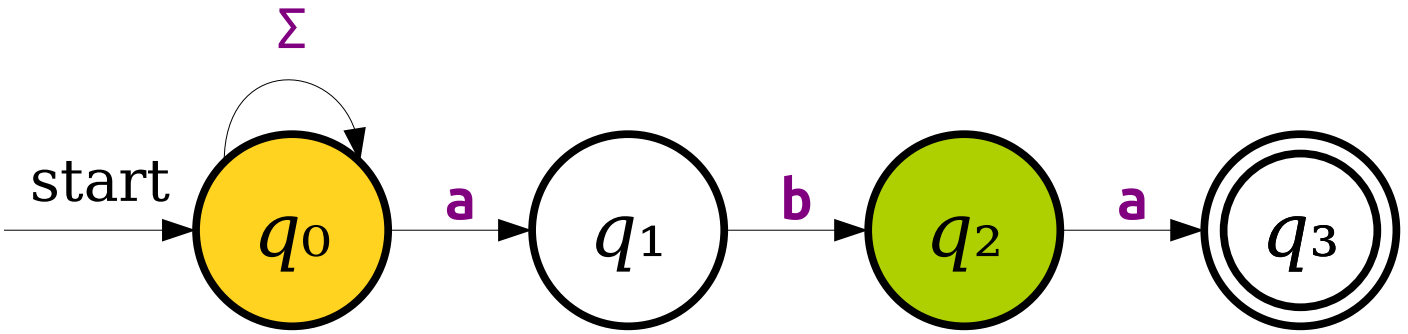


Massive Parallelism



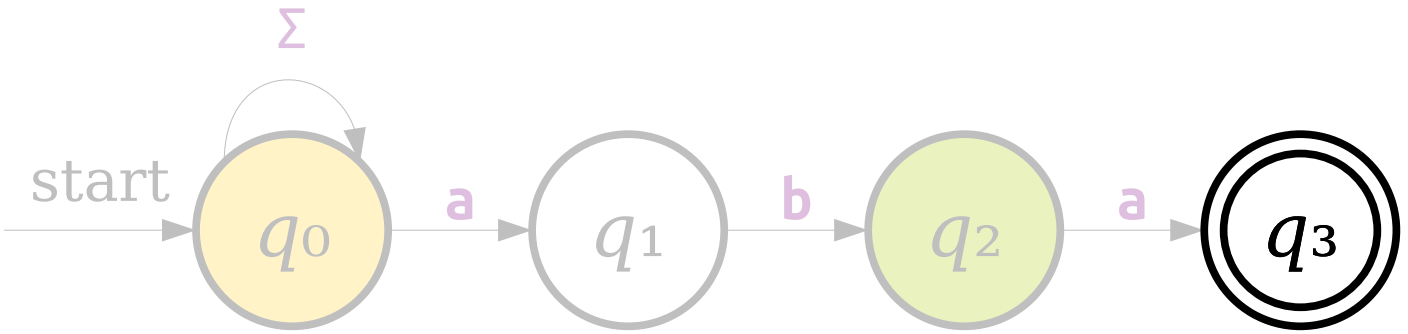


Massive Parallelism



a	b	a	b

Massive Parallelism



We're not in any accepting state, so no possible path accepts.

a	b	a	b

Massive Parallelism

- An NFA can be thought of as a DFA that can be in many states at once.
- At each point in time, when the NFA needs to follow a transition, it tries all the options at the same time.
- (Here's a rigorous explanation about how this works; read this on your own time).
  - Start off in the set of all states formed by taking the start state and including each state that can be reached by zero or more ε-transitions.
  - When you read a symbol **a** in a set of states *S*:
    - Form the set *S'* of states that can be reached by following a single **a** transition from some state in *S*.
    - Your new set of states is the set of states in *S'*, plus the states reachable from *S'* by following zero or more ε-transitions.

Next Time

- ***The Subset Construction***
  - So beautiful. So elegant. So cool!
- ***Closure Properties of Regular Languages***
  - Transforming languages by transforming machines.
- ***The Kleene Closure***
  - What's the deal with the notation Σ^*?

Appendix: More DFA Designs

More Elaborate DFAs

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

Let's have the **a** symbol be a placeholder for "some character that isn't a star or slash."

Let's design a DFA for C-style comments. Those are the ones that start with /* and end with */.

Accepted:

```
/*a*/
/**/
/***/
/*aaa*aaa*/
/*a/a*/
```

Rejected:

```
/**
/**/a/*aa*/
aaa/**/aa
/*/
/**a/
//aaaa
```

More Elaborate DFAs

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