Finite Automata

Part One

Computability Theory

What problems can we solve with a computer?

What problems can we solve with a computer?

What kind of computer?

What is a computer?

Bell Labs in Oakland, CA
(Photo by Larry Luckham, 1969)

Apple
Watch
(2018)

Two Challenges

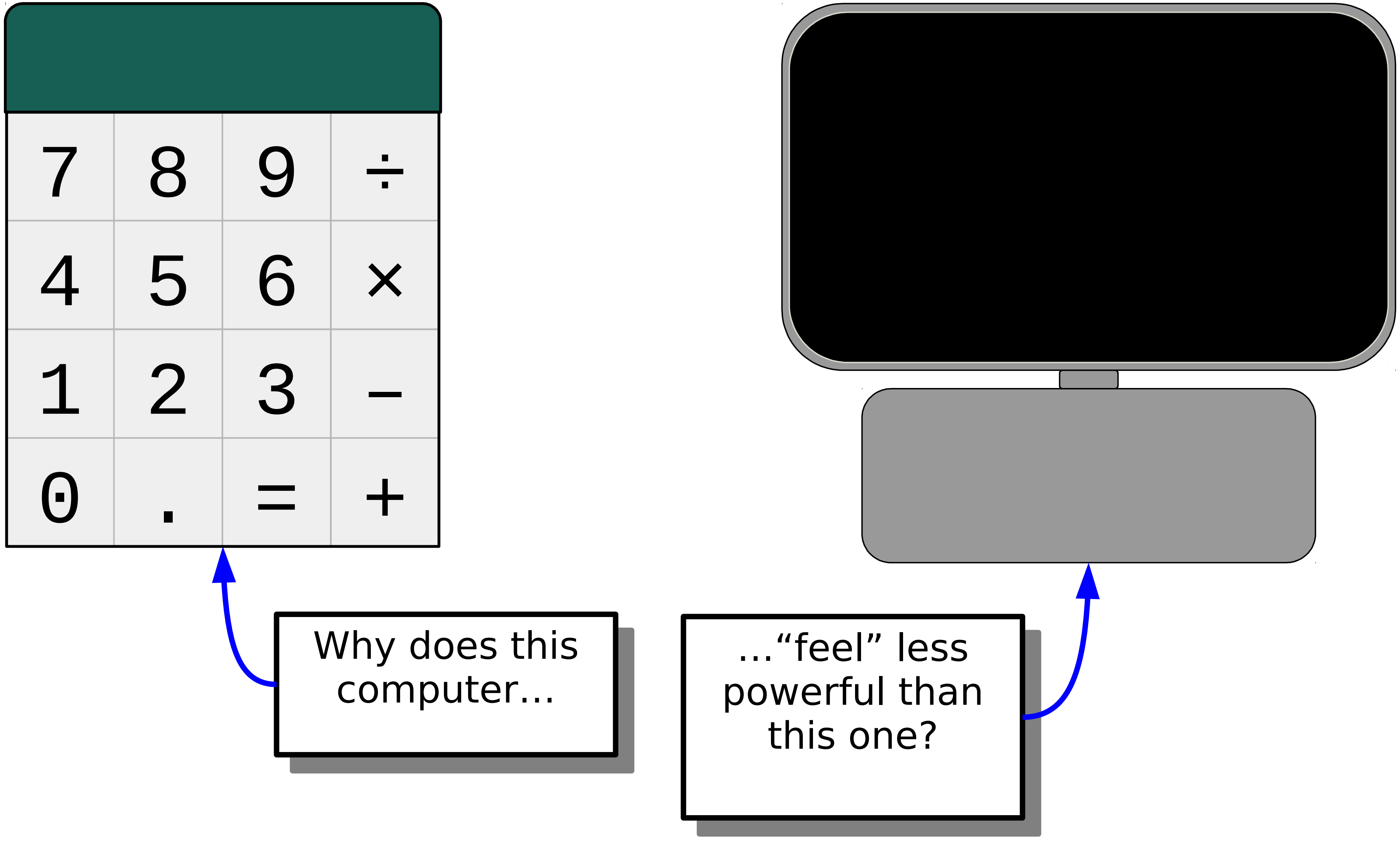
- Computers are dramatically better now than they've ever been, and that trend continues.
- Writing proofs on formal definitions is hard, and computers are *way* more complicated than sets, graphs, or functions.
- ***Key Question:*** How can we prove what computers can and can't do…
  - … so that our results are still true in 20 years?
  - … without multi-hundred page proofs?

Enter Automata

- An ***automaton*** is a mathematical model of a computing device.
- It's an ***abstraction*** of a real computer
  - Same way that graphs are abstractions of social networks, transportation grids, etc.

What do these automata look like?

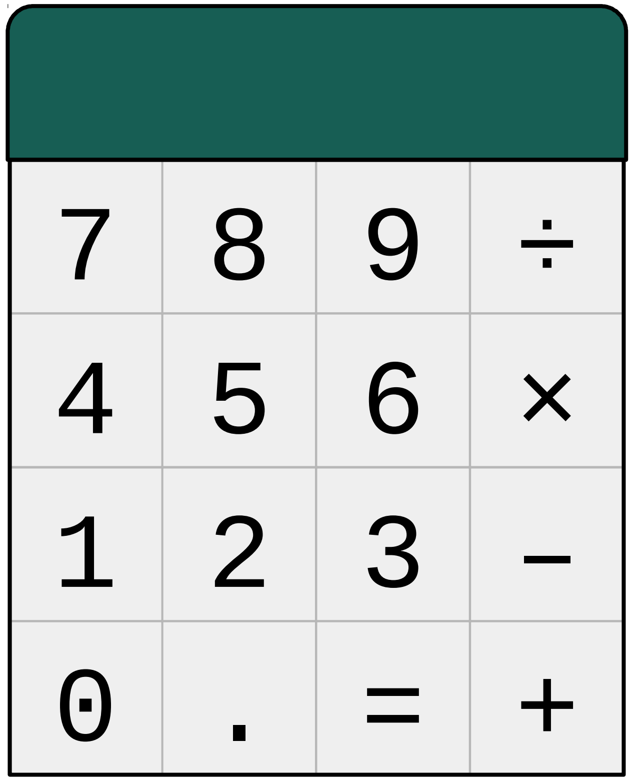
A Tale of Two Computers

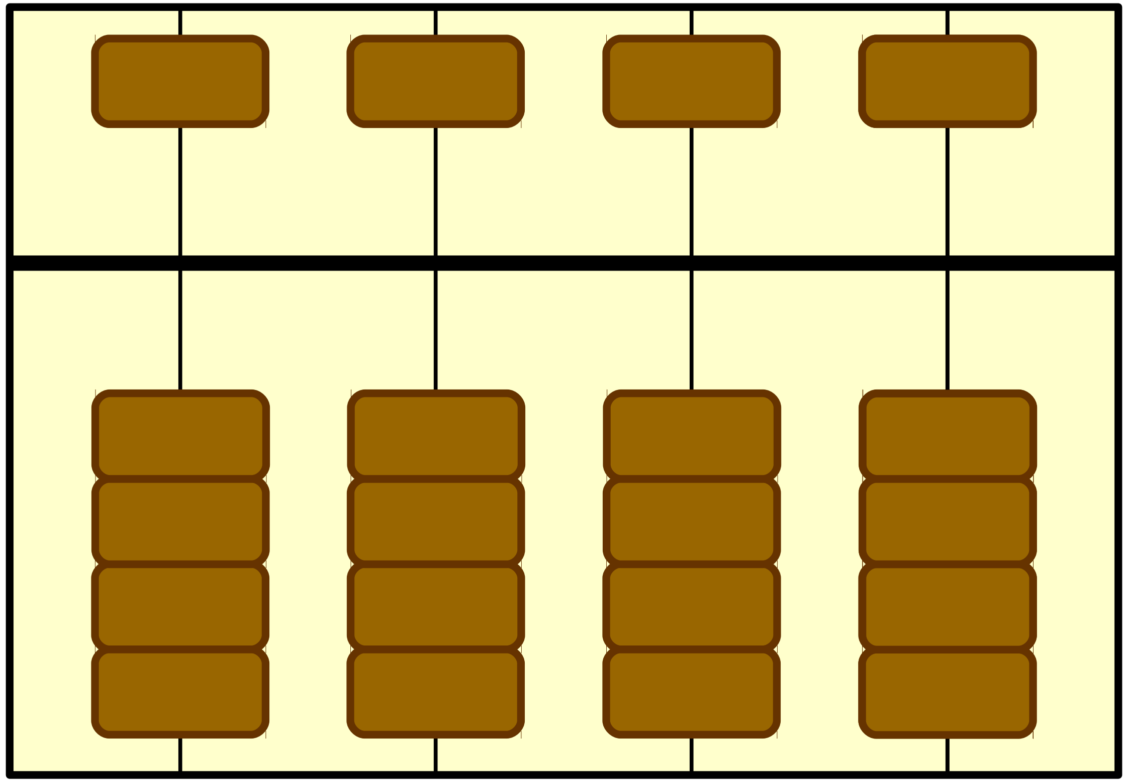

7 8 9 ÷
4 5 6 ×
1 2 3 −
0 . = +
Why does this computer…
…"feel" less powerful than this one?

Calculators vs. Desktops

- A calculator has a ***small amount of memory***. A desktop computer has a ***large amount of memory***.
- A calculator performs a ***fixed set of functions***. A desktop is ***reprogrammable*** and can run many different programs.

Computing with Finite Memory





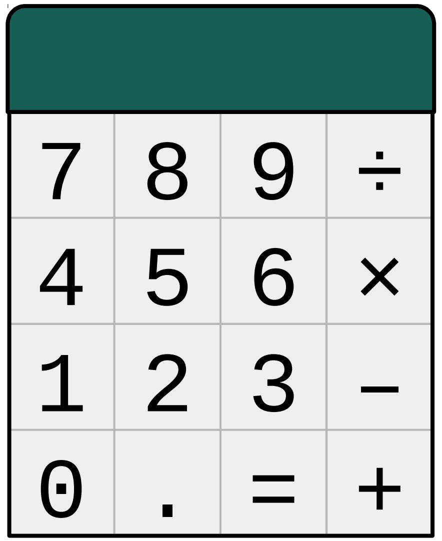
Data stored electronically.
Algorithm is in silicon.
Memory limited by display.

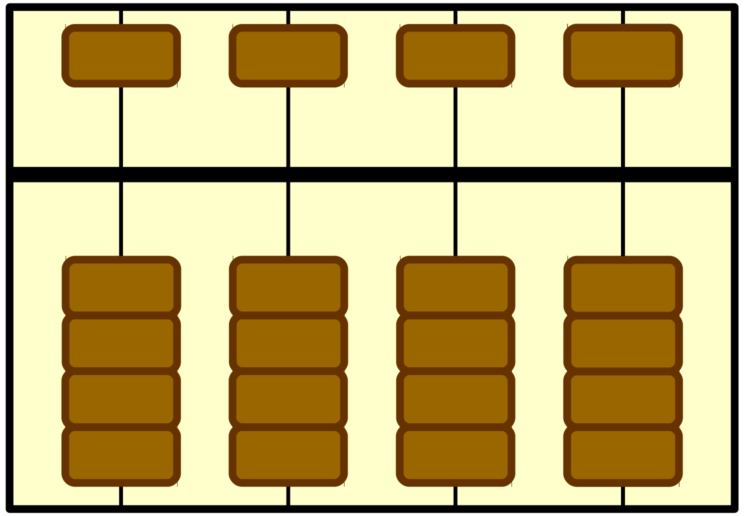
Data stored in wood.
Algorithm is in brain.
Memory limited by beads.

How do we model “memory” and ”an algorithm” when they can take on so many forms?

What's in Common?

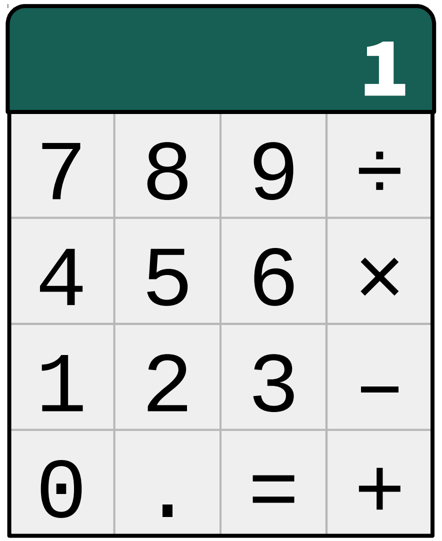
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

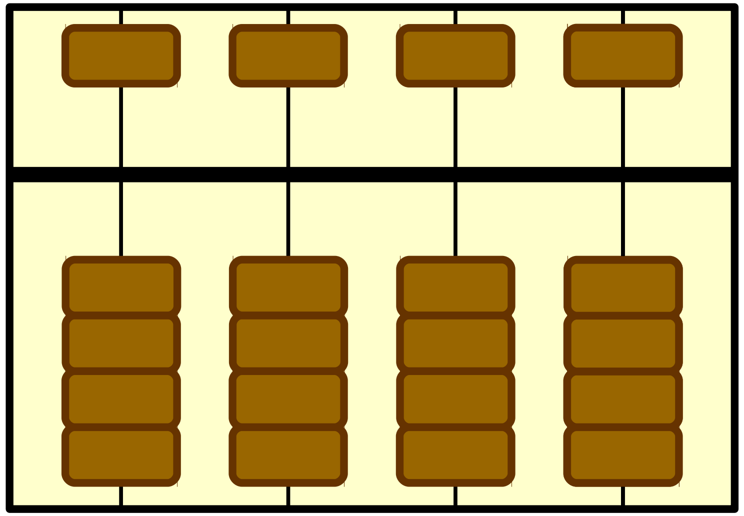




What’s in Common?

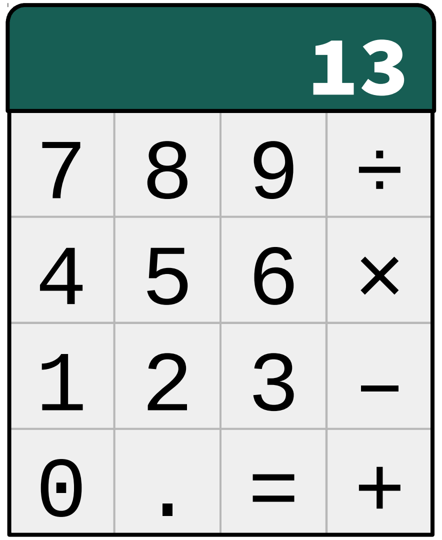
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

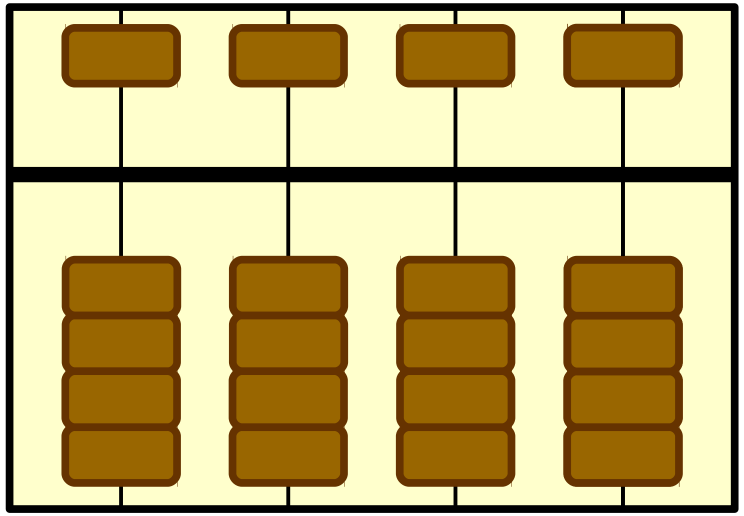




What’s in Common?

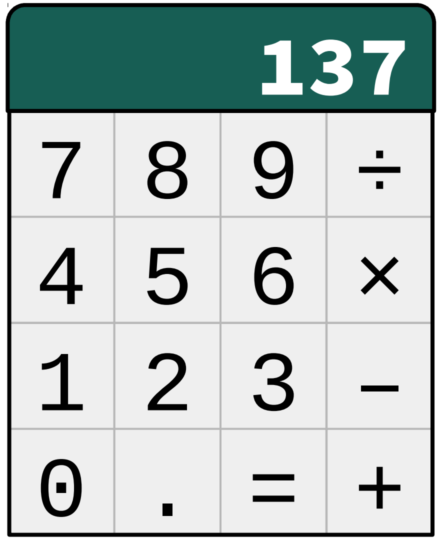
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

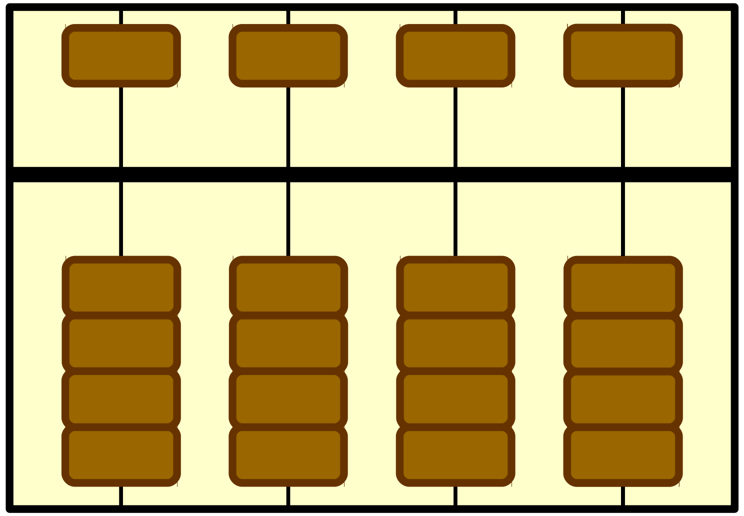




What's in Common?

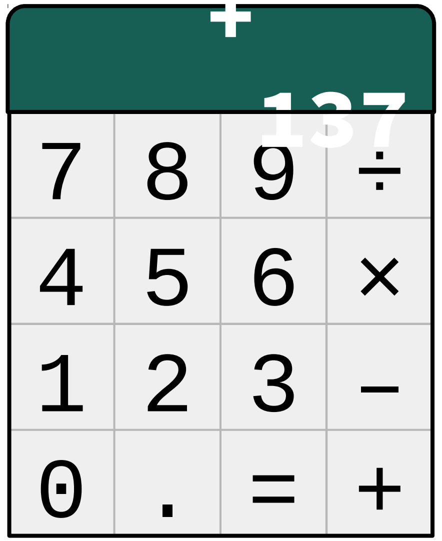
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

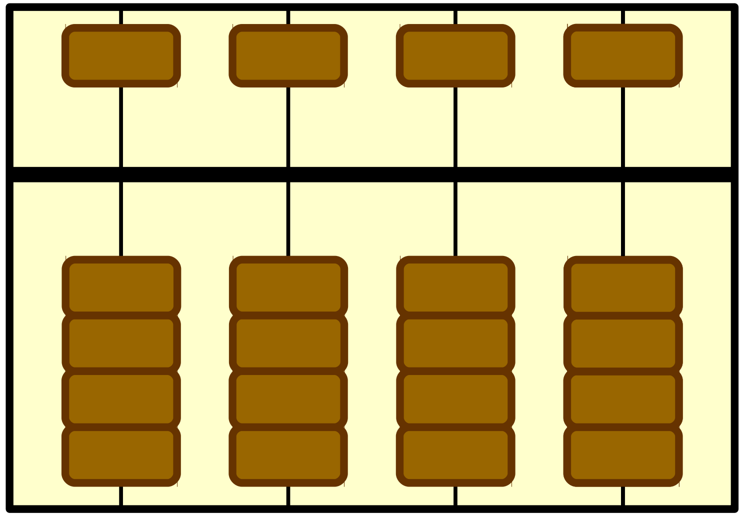




What's in Common?

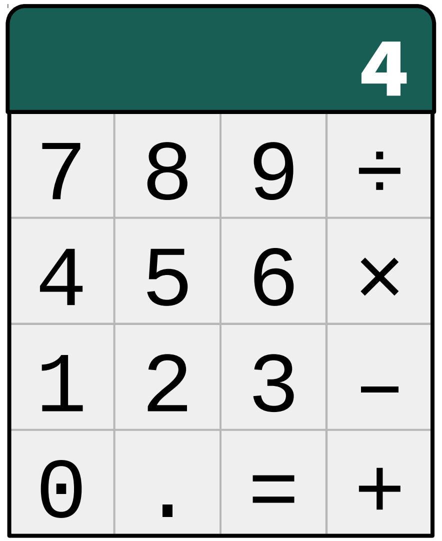
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

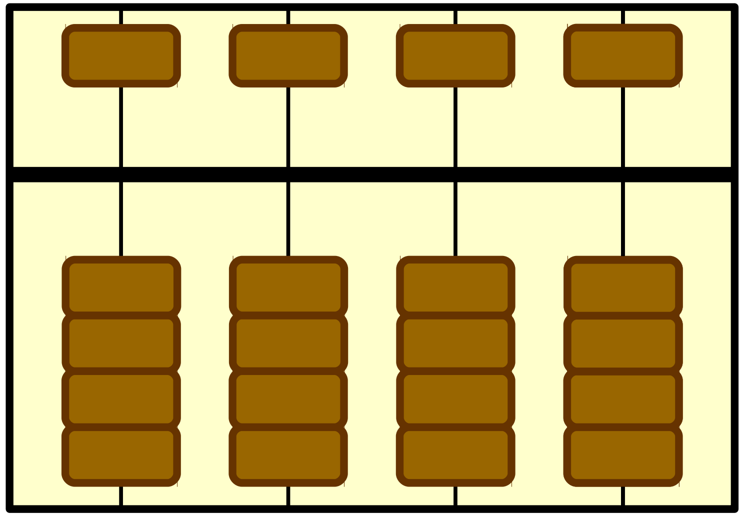




What's in Common?

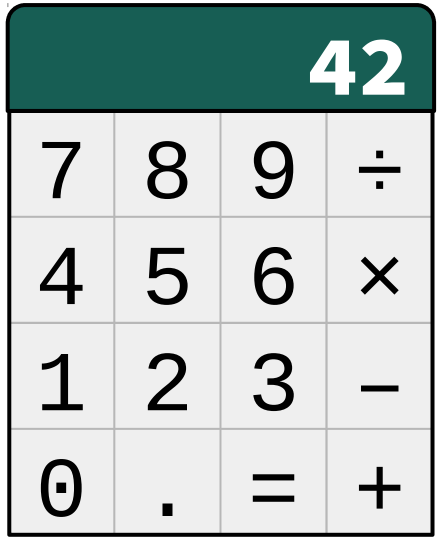
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

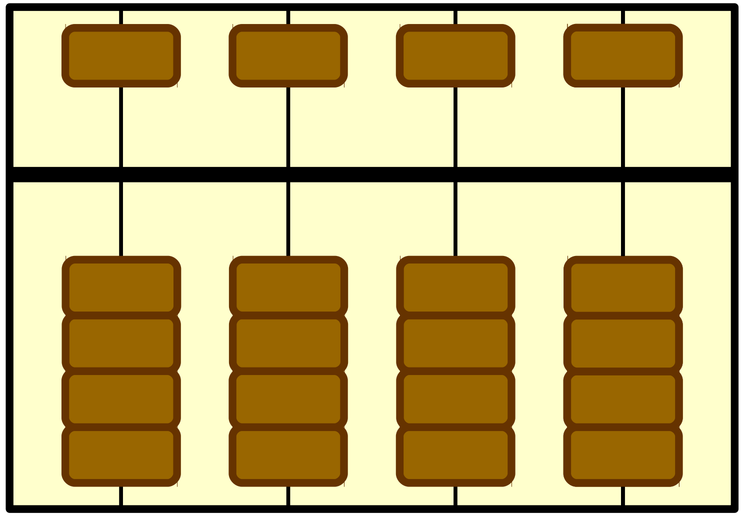




What's in Common?

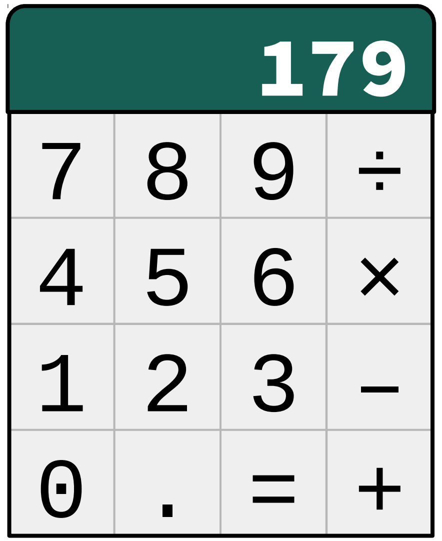
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

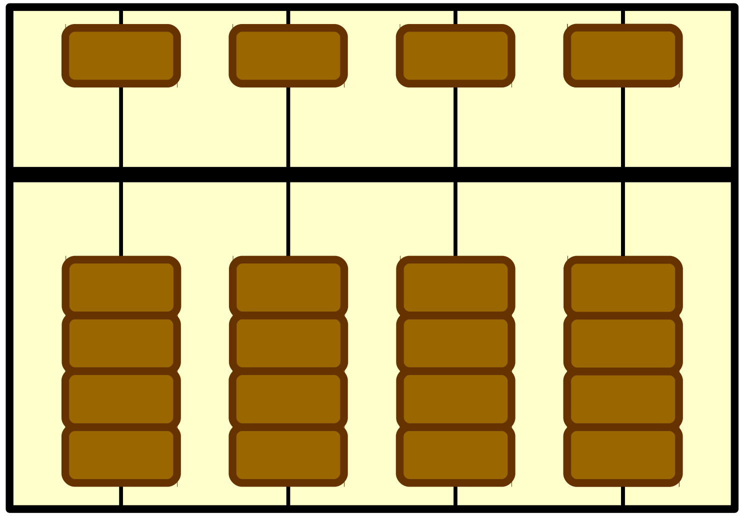




What's in Common?

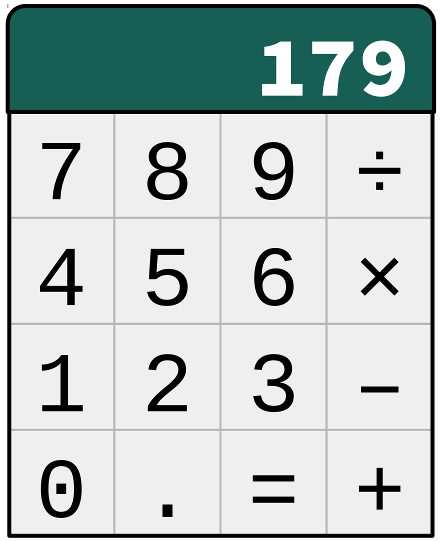
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

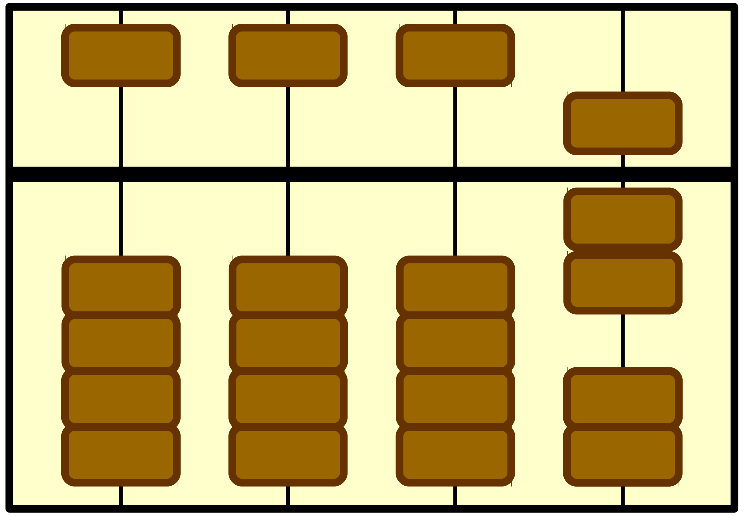




What's in Common?

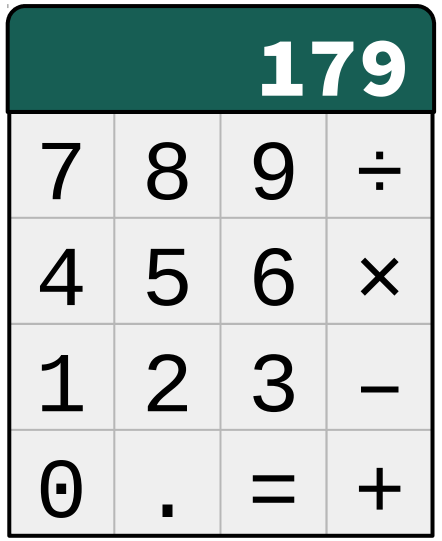
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

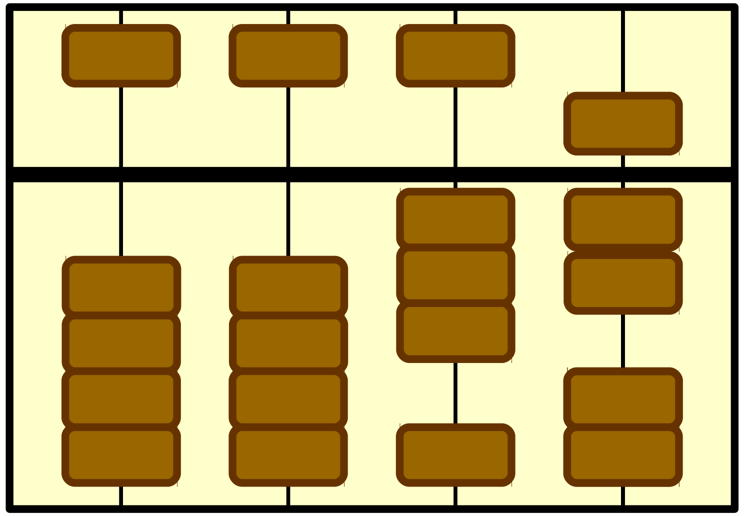




What's in Common?

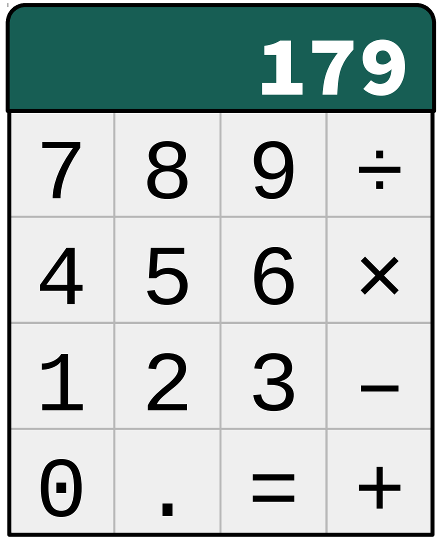
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

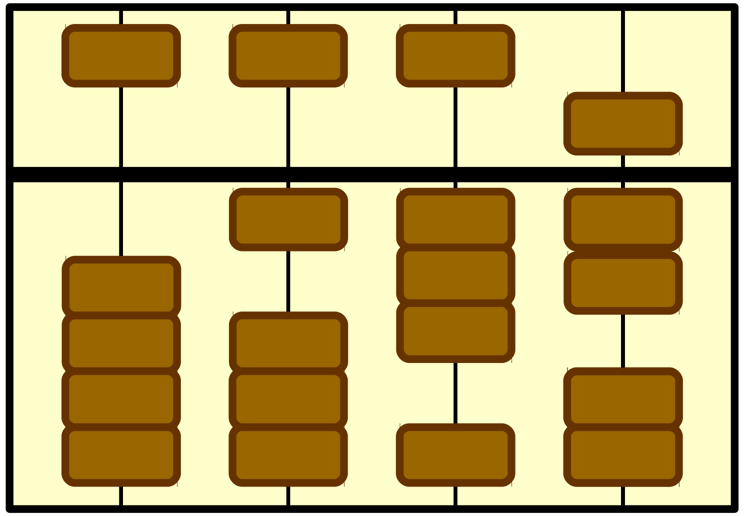




What's in Common?

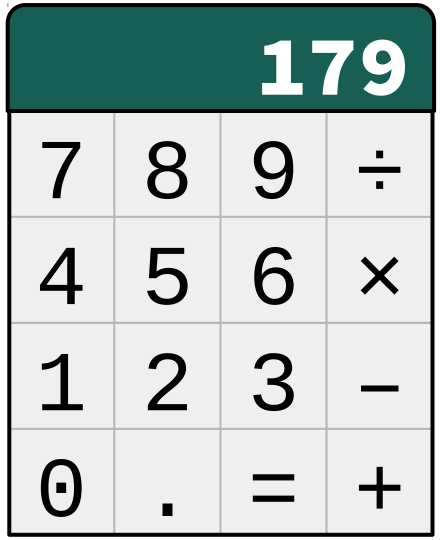
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

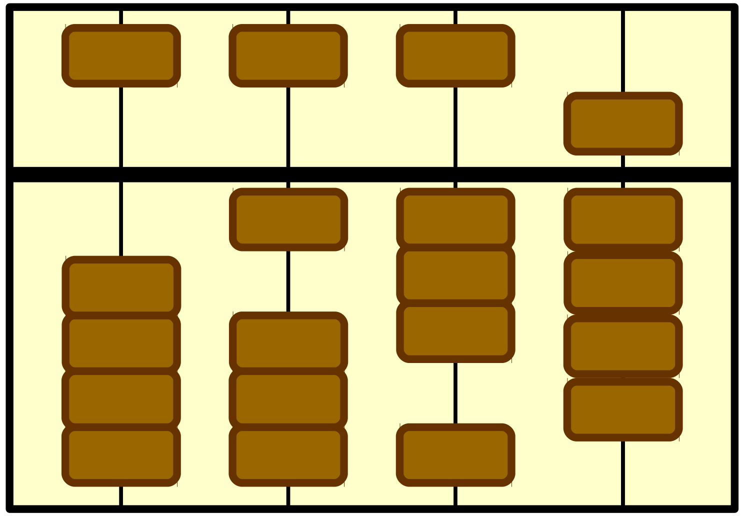




What's in Common?

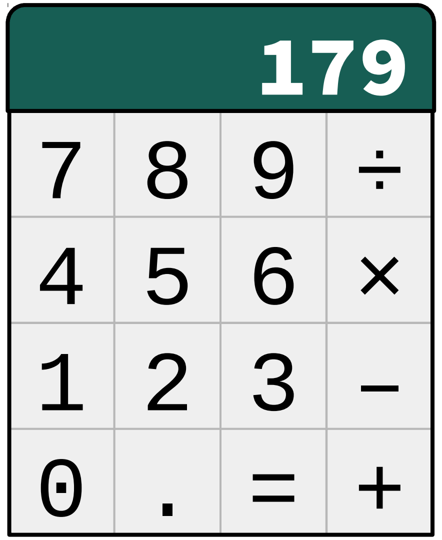
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

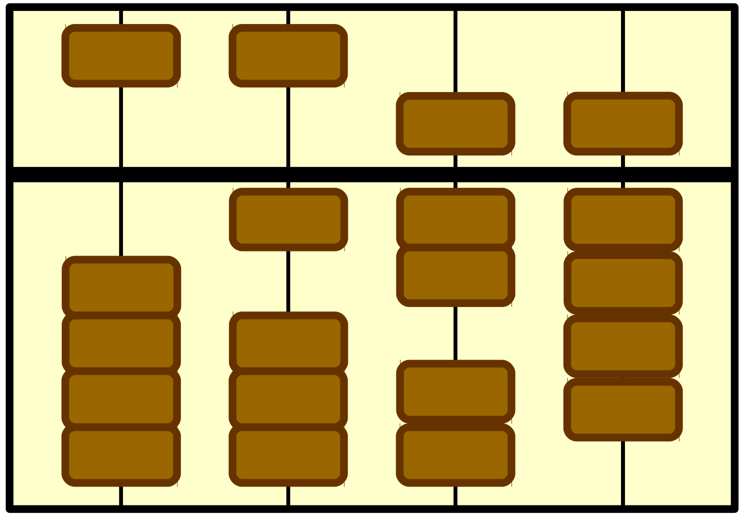




What’s in Common?

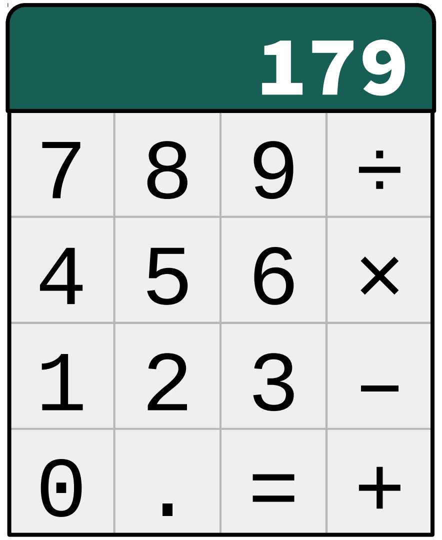
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.

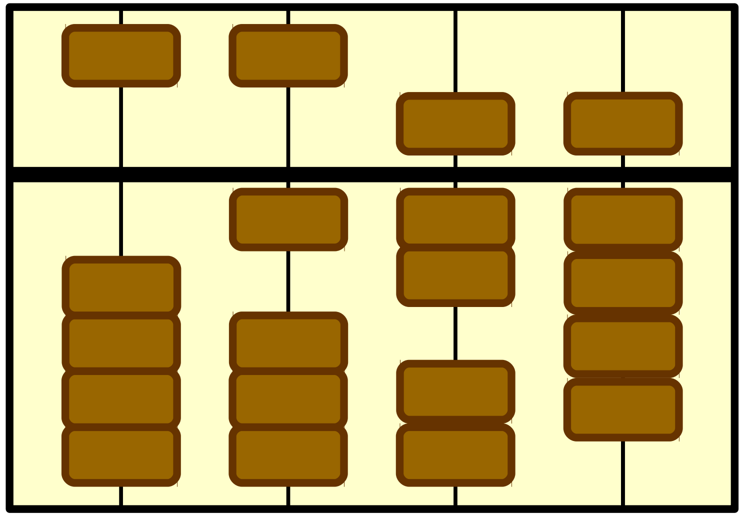




What’s in Common?

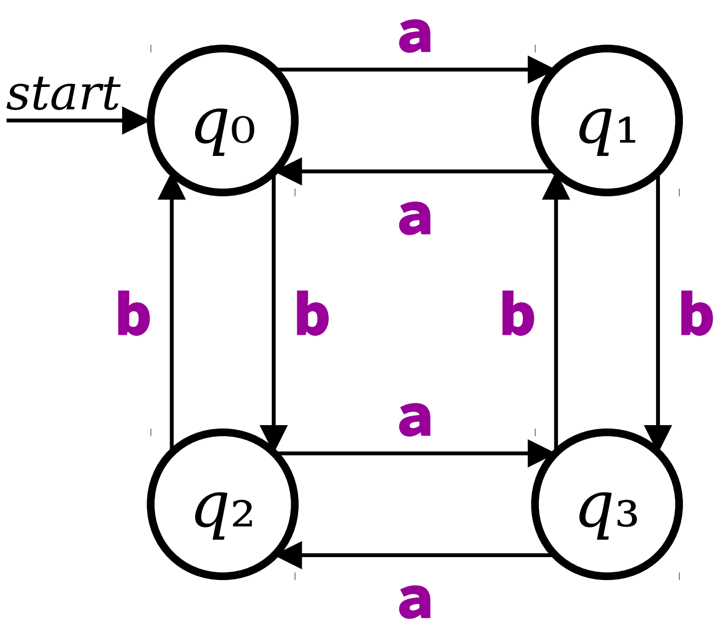
- These machines ***receive input*** from an external source.
- That input is provided ***sequentially***, one discrete unit at a time.
- Each input causes the device to ***change configuration***. This change, big or small, is where the computation happens.
- Once all input is provided, we can ***read off an answer*** based on the configuration of the device.





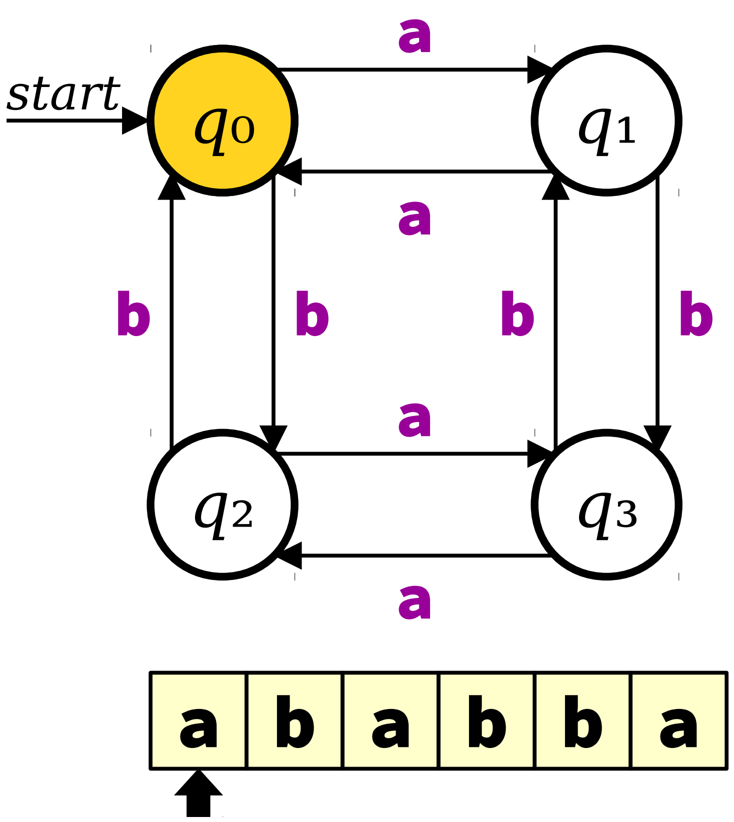
Modeling Finite Computation

- We will model a finite-memory computer as a collection of ***states*** linked by ***transitions***.
- Each state corresponds to one possible configuration of the device's memory.
- Each transition indicates how memory changes in response to inputs.
- Some state is designated as the ***start state***. The computation begins in that state.



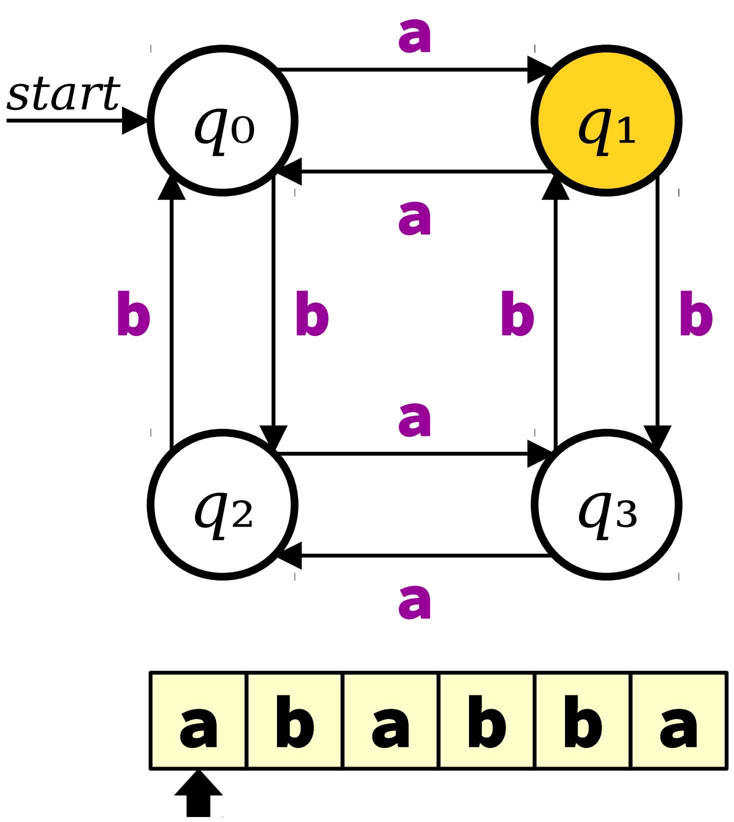
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



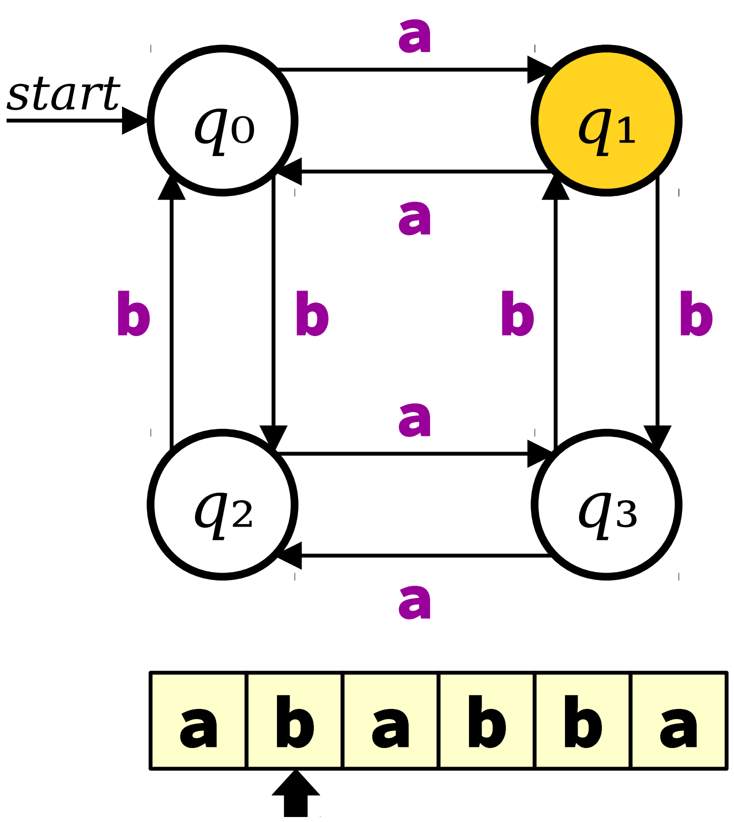
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



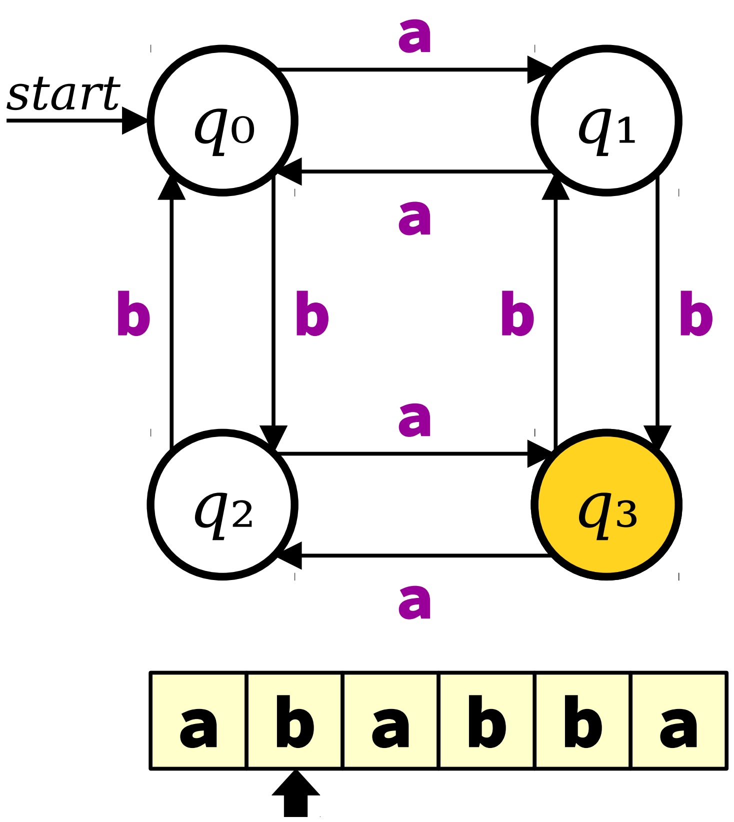
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



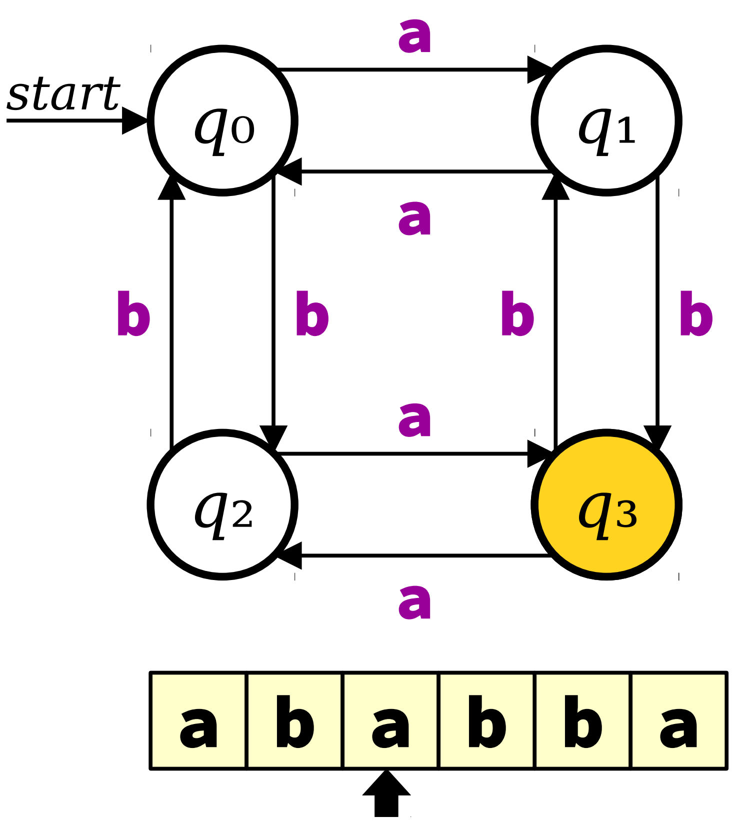
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



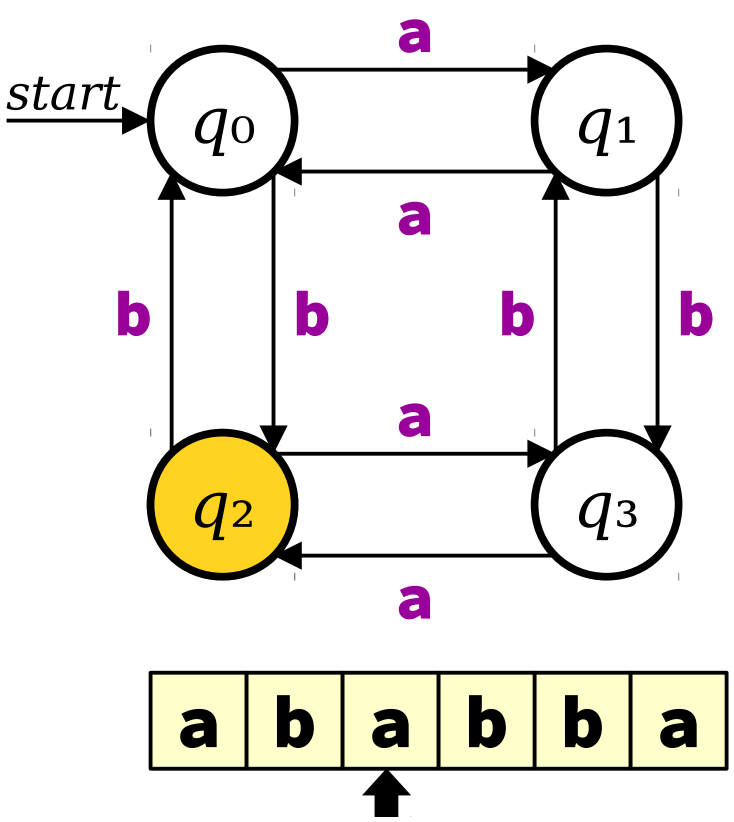
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



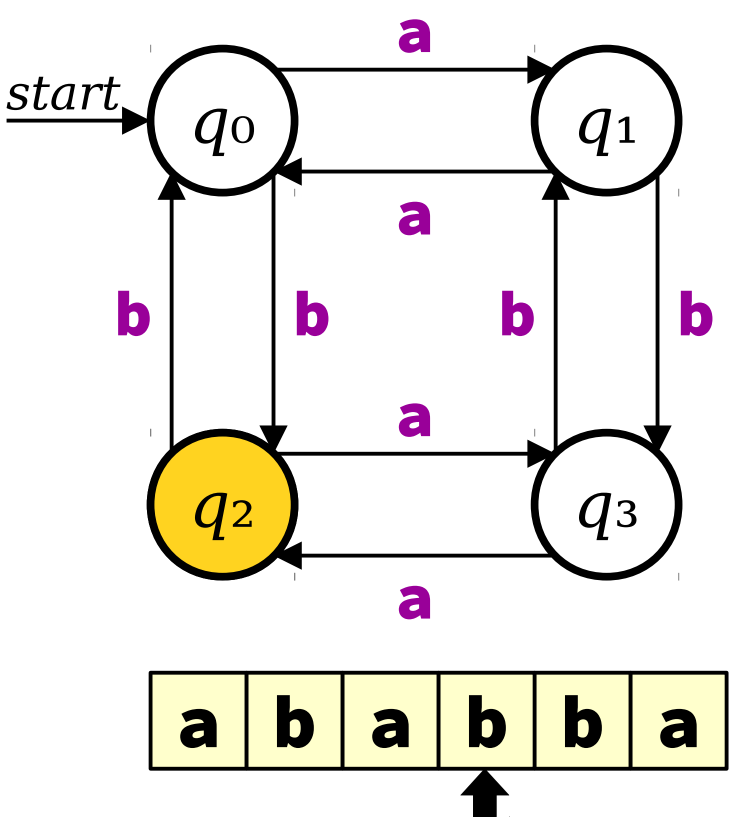
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



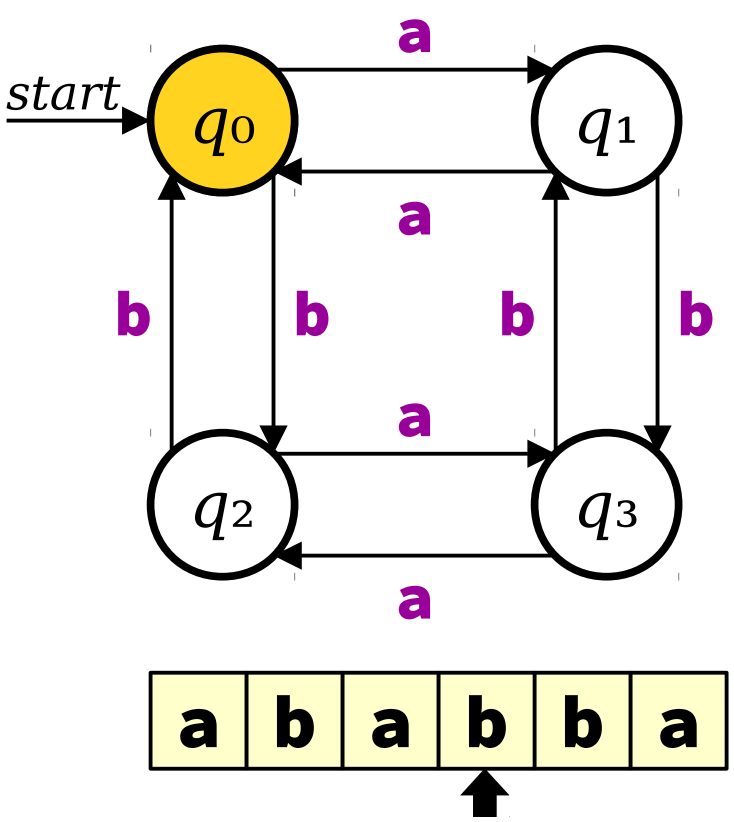
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



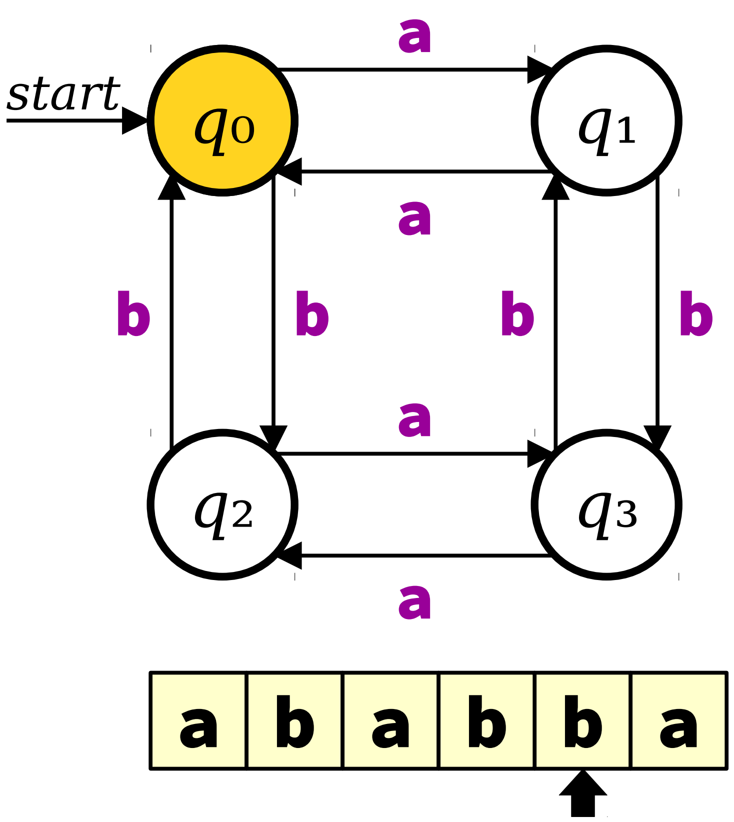
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



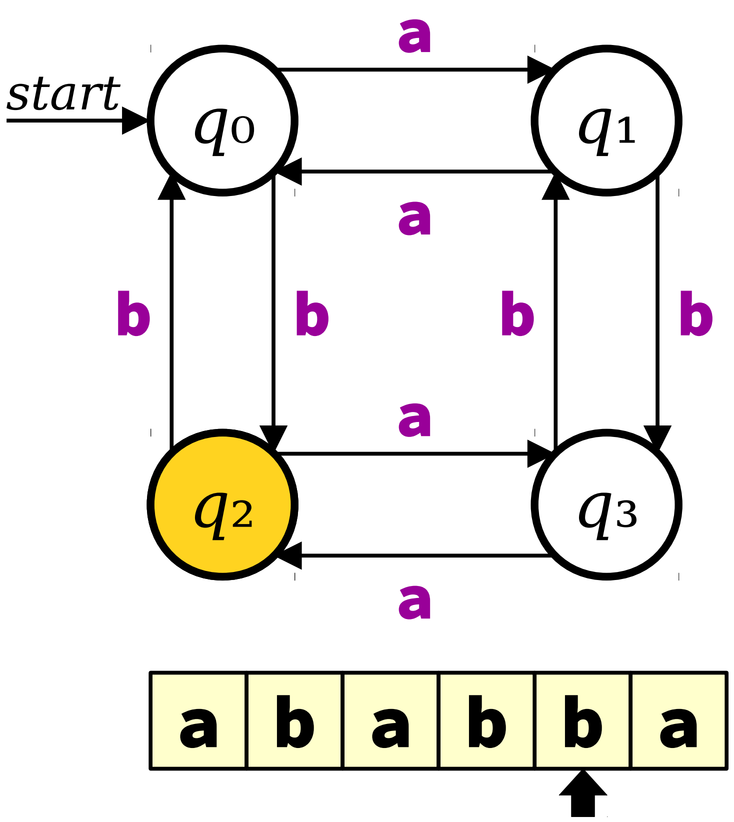
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



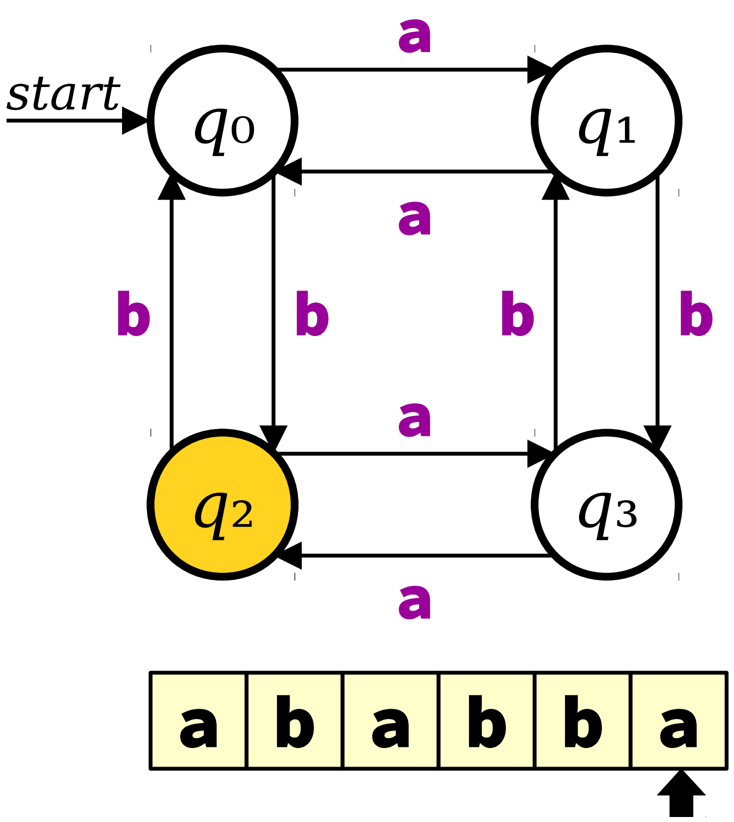
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



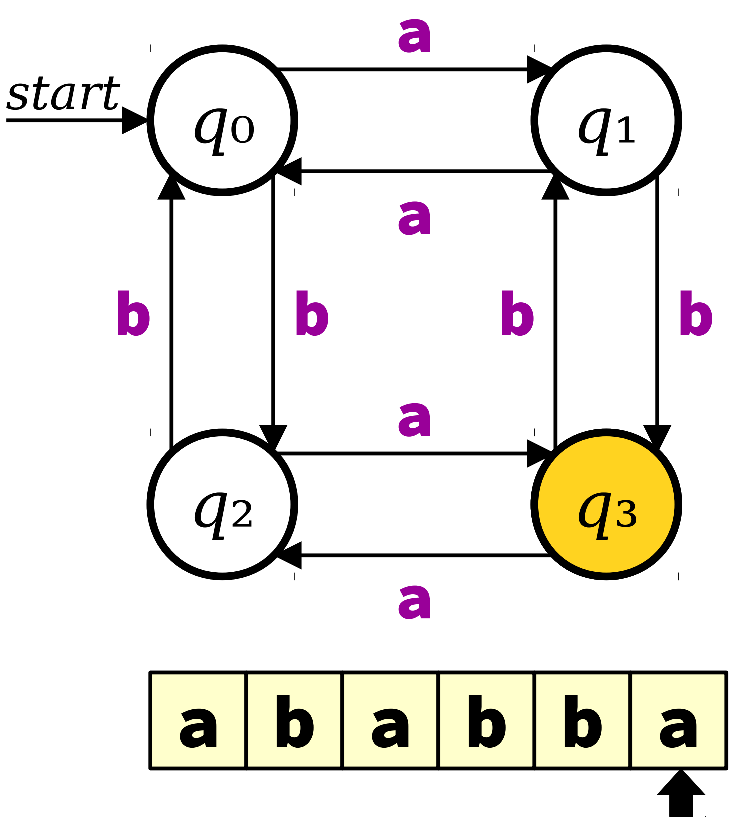
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



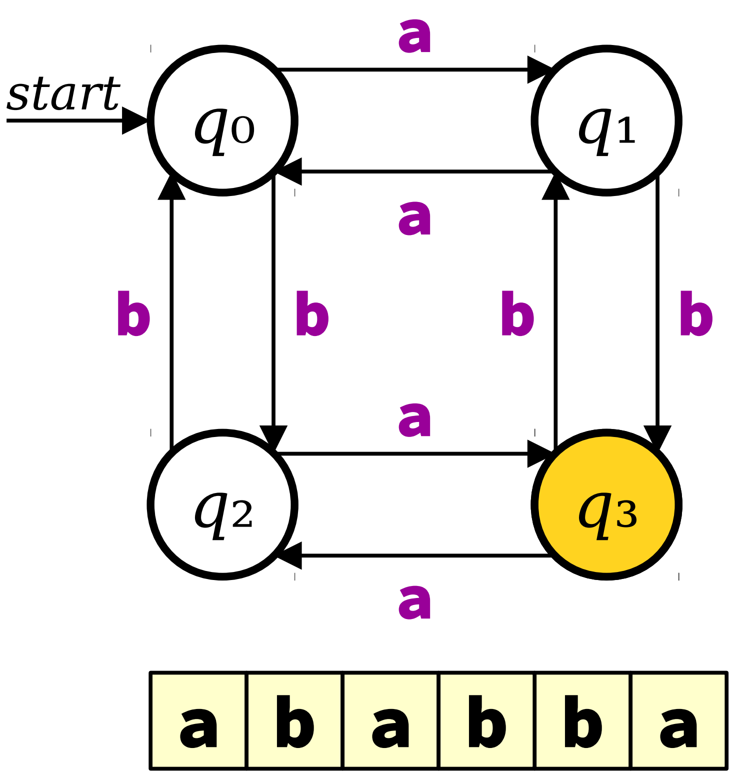
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



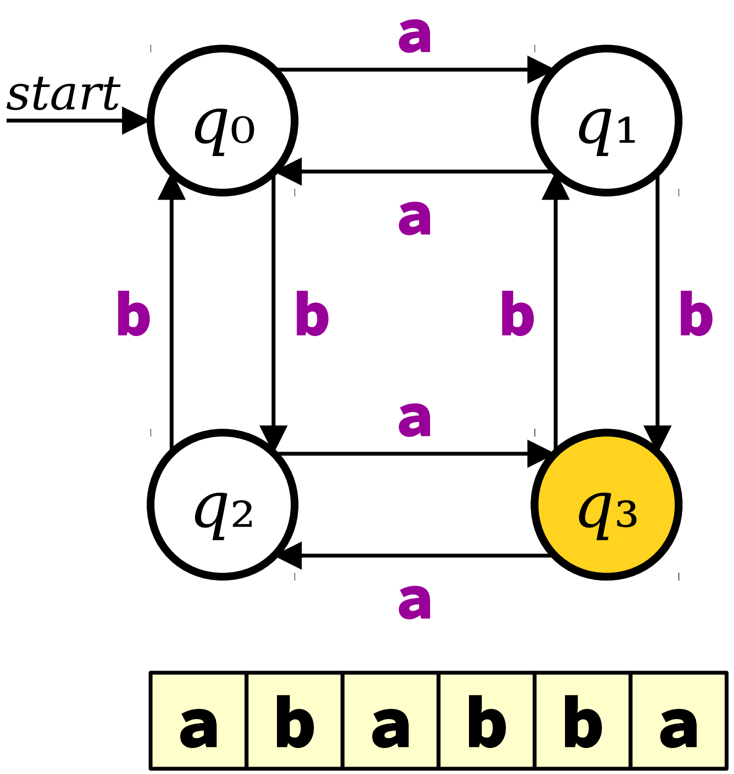
Modeling Finite Computation

- This device processes ***strings*** made of ***characters***.
  - Each character represents some external input to the device.
  - The string represents the full sequence of inputs to the device.
- To run this device, we begin in our start state and scan the input from left to right.
- Each time the machine sees a character, it ***changes state*** by following the transition labeled with that character.



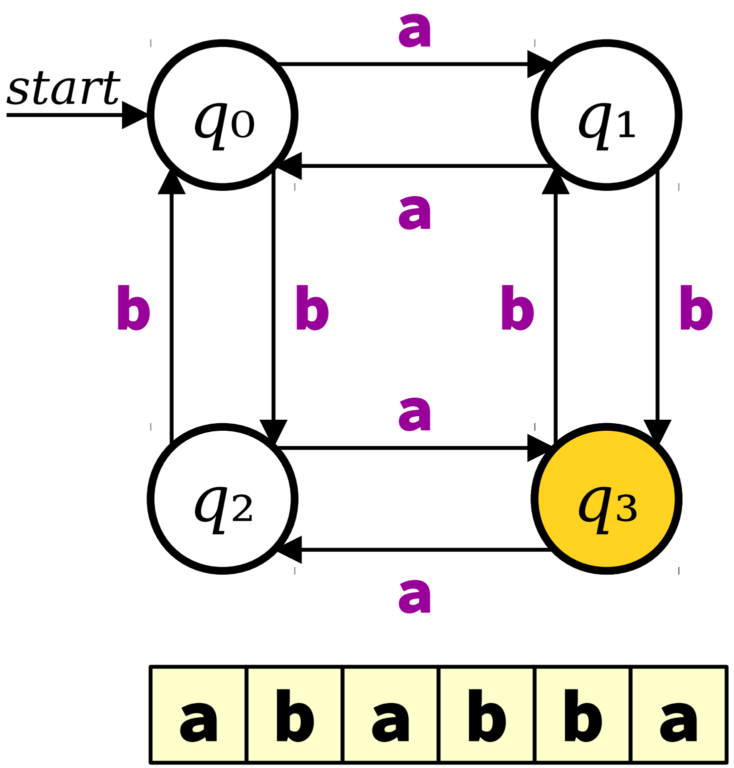
Modeling Finite Computation

- Once we've finished entering all the characters of our input, we need to obtain the result of the computation.
- As a simplifying assumption, we'll assume *(for now)* that we just need to get a single bit of output. That is, our machines will just say YES or NO.



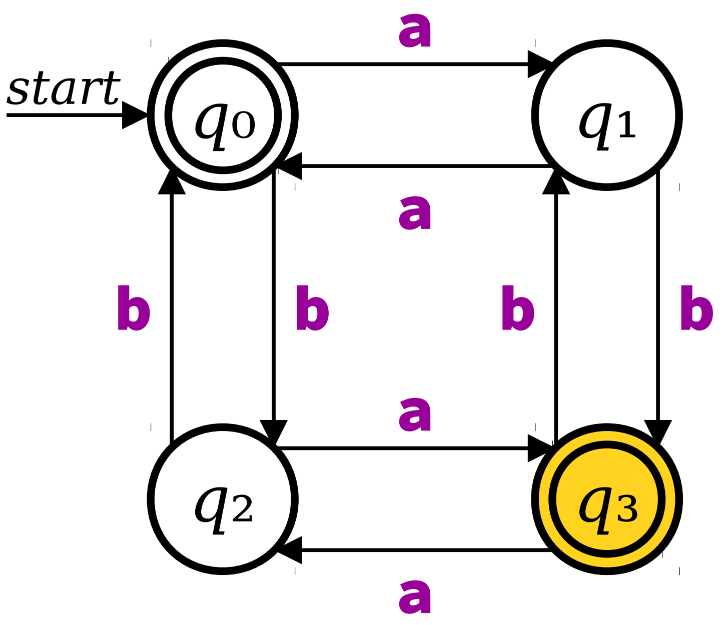
Modeling Finite Computation

- Some of the states in our computational device will be marked as ***accepting states***. These are denoted with a double ring.



Modeling Finite Computation

- Some of the states in our computational device will be marked as ***accepting states***. These are denoted with a double ring.
- 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).



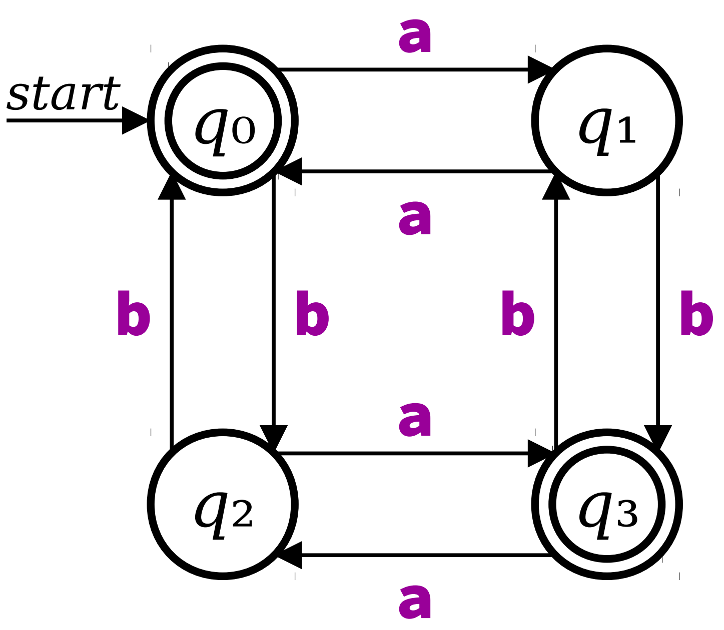
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

aabb

abbababba



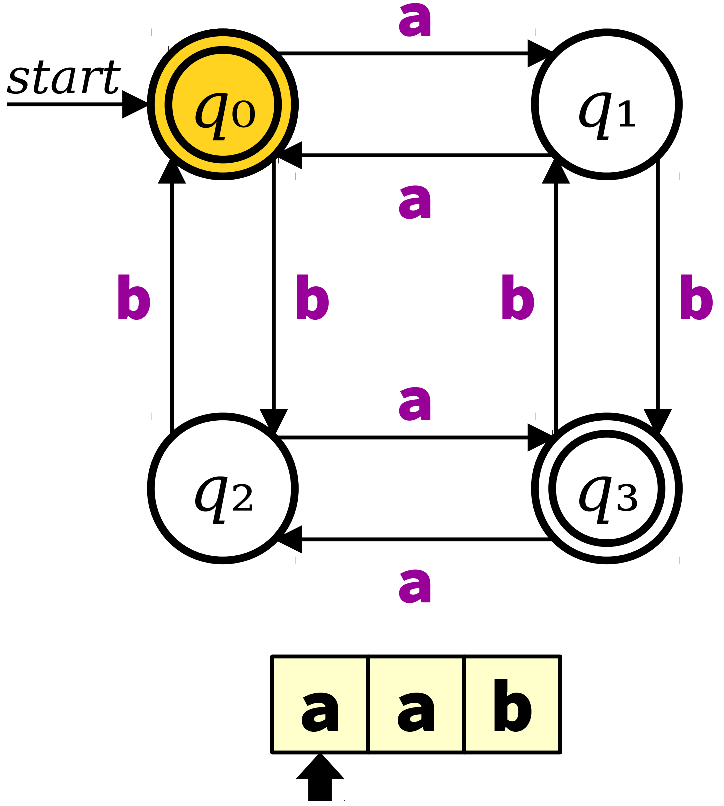
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

aabb

abbababba



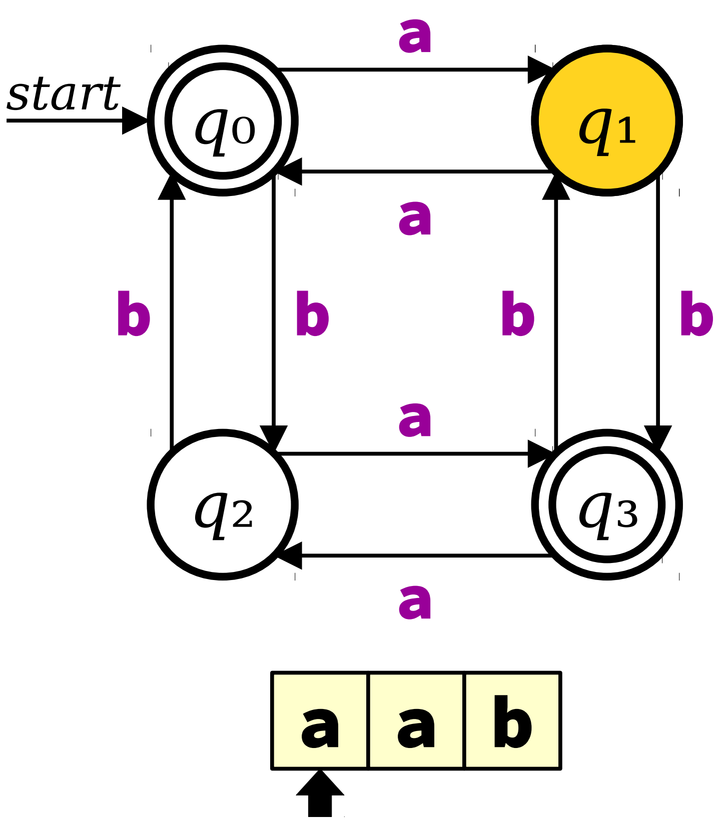
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

aabb

abbababba



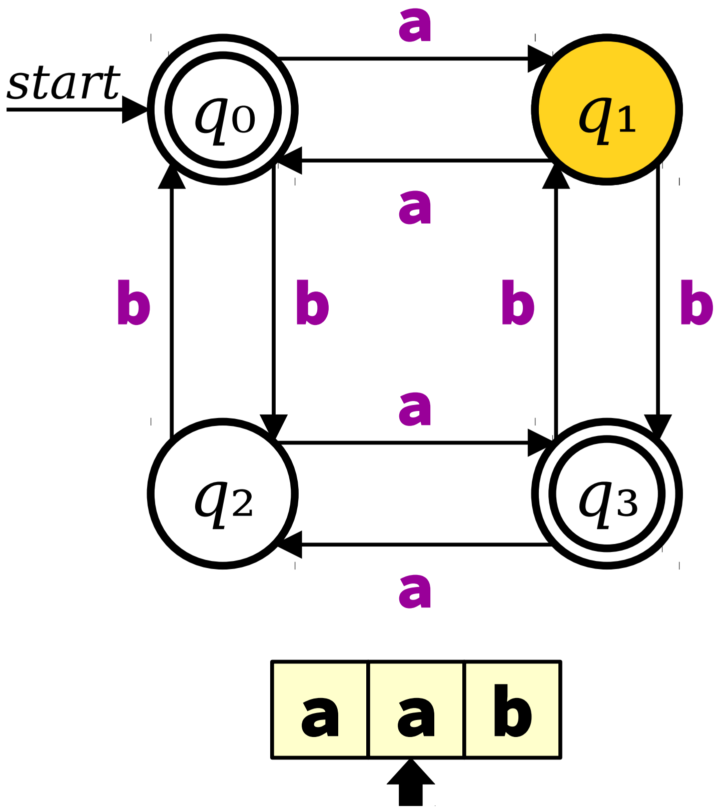
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

aabb

abbababba



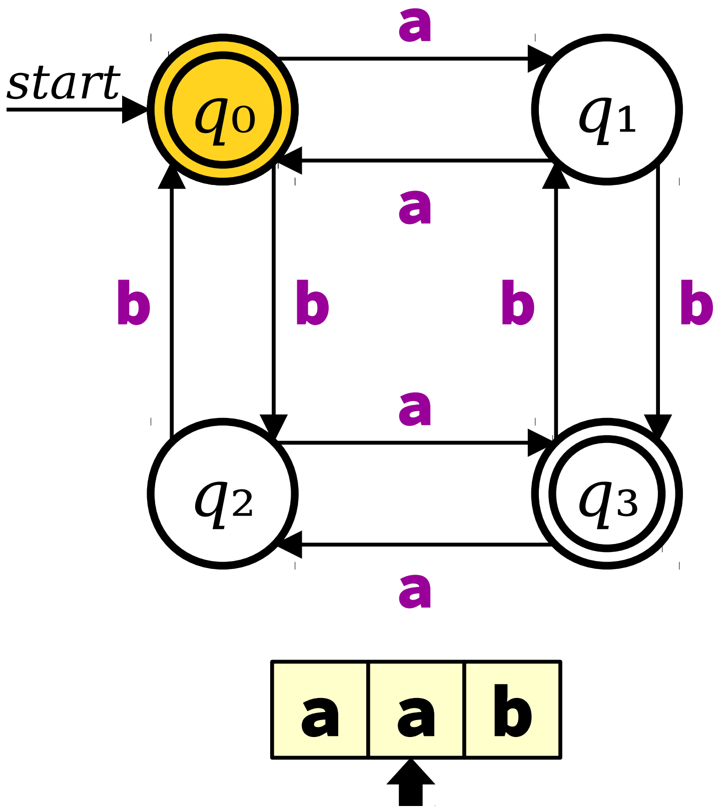
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

aabb

abbababba



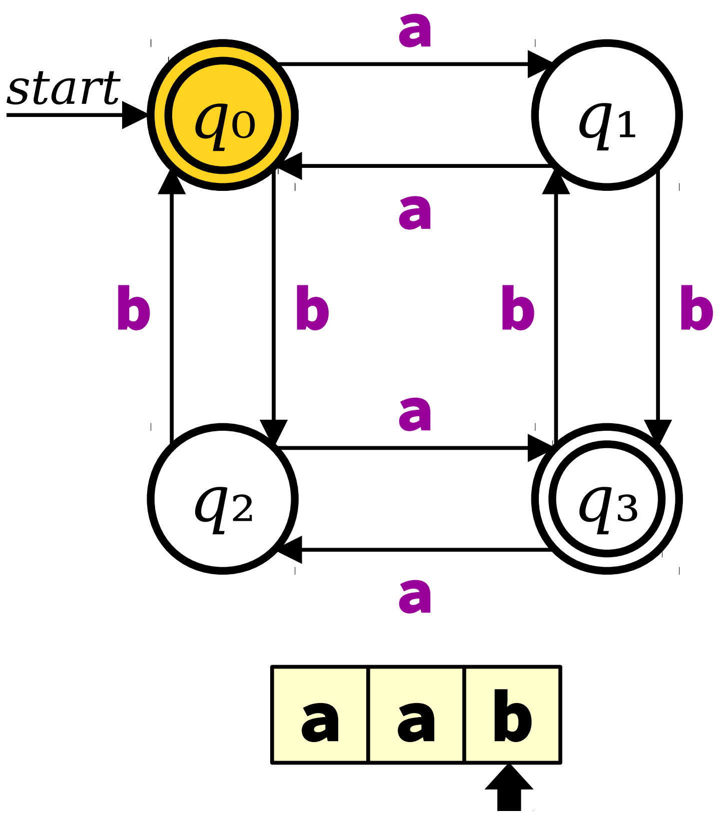
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

aabb

abbababba



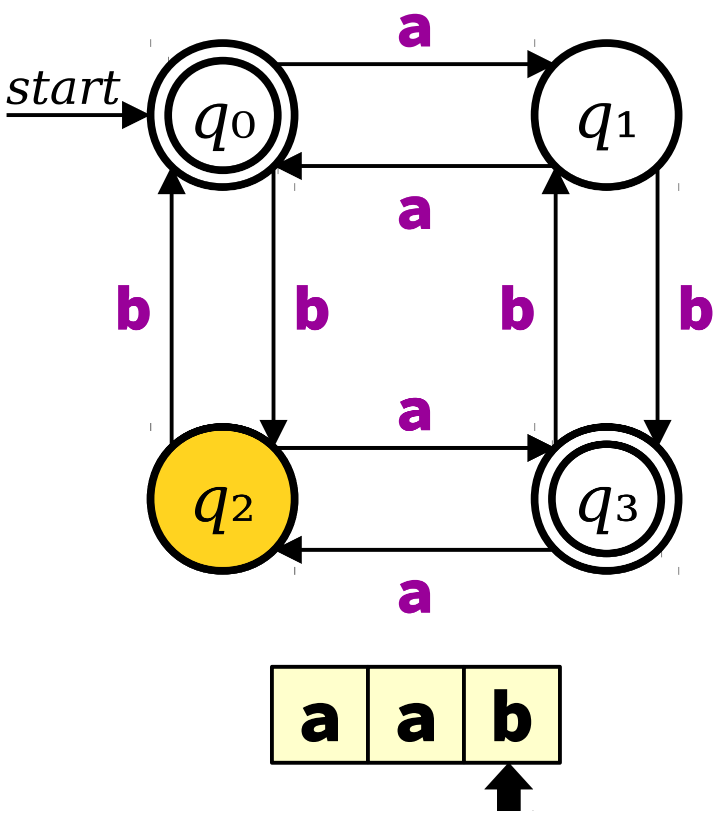
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

aabb

abbababba



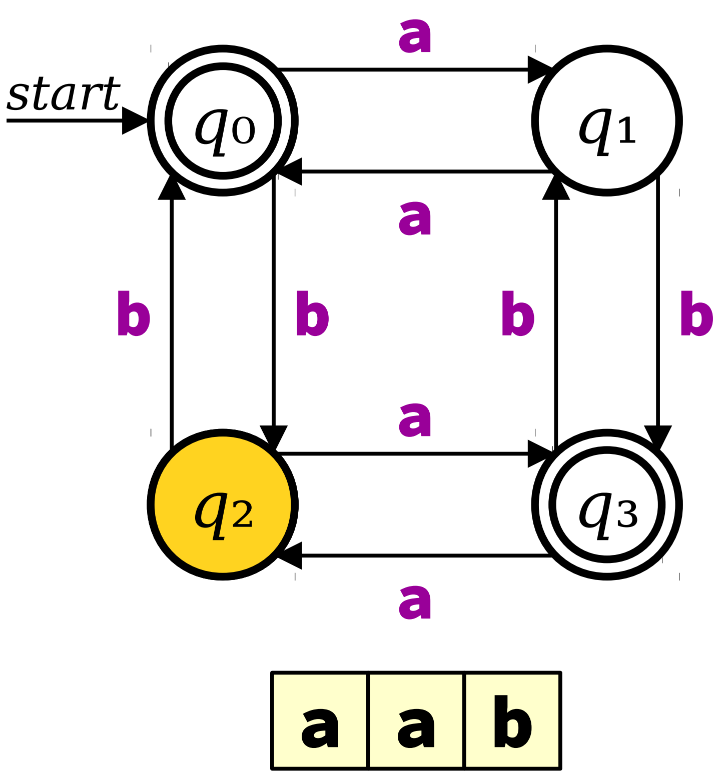
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

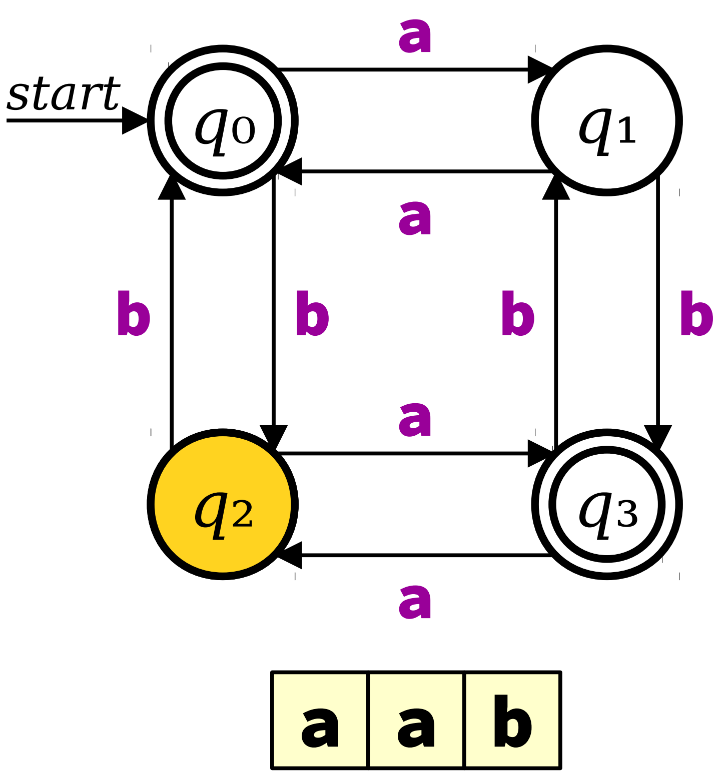
aabb

abbababba



Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?



aab

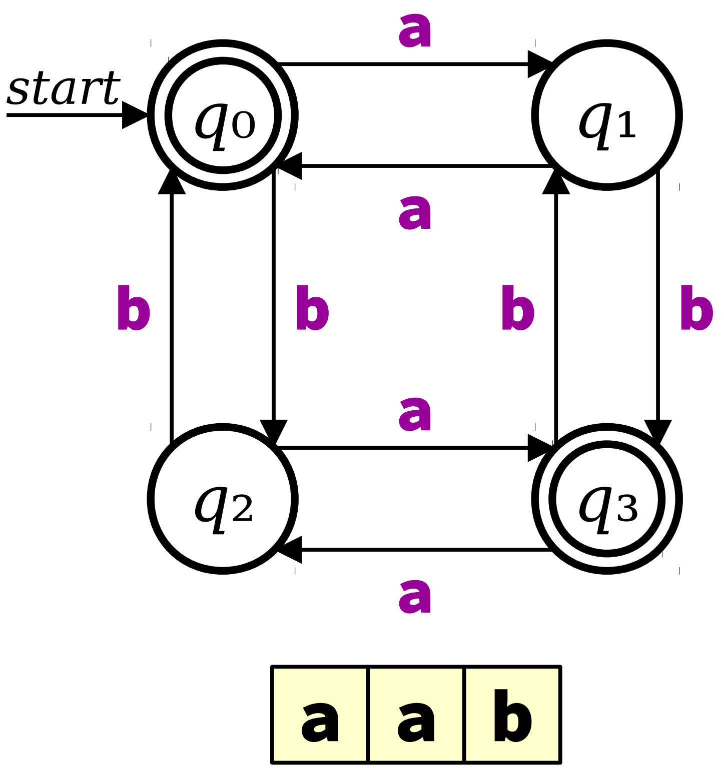
Modeling Finite Computation

- Try it yourself! Which of these strings does this device accept?

aab

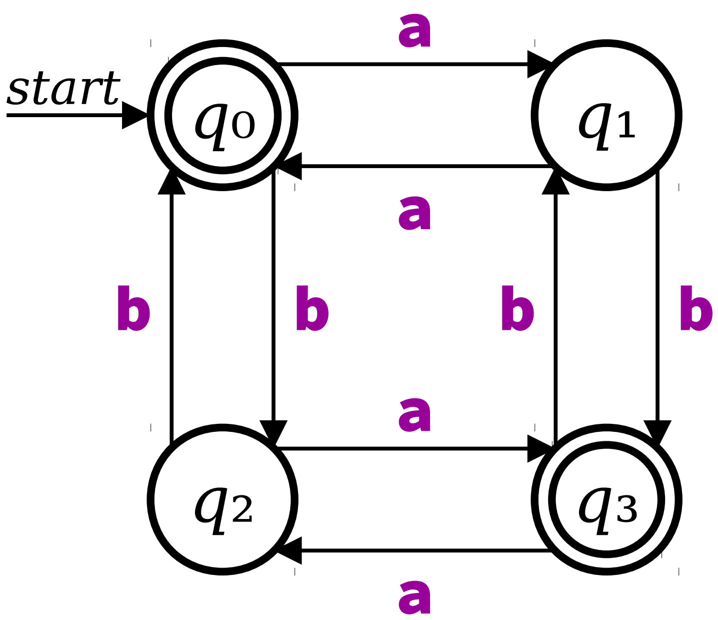
aabb

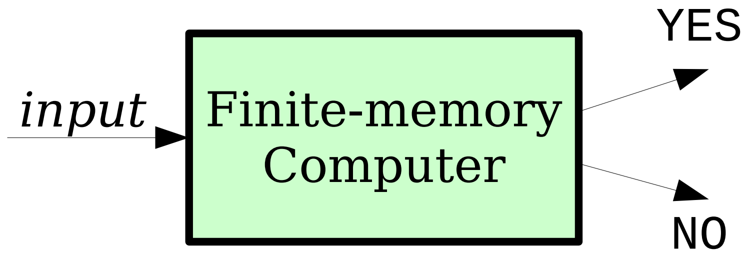
abbababba



Finite Automata

- This type of computational device is called a ***finite automaton*** (plural: ***finite automata***).
- Finite automata model computers where (1) memory is finite and (2) the computation produces as YES/NO answer.
- In other words, finite automata model predicates, and do so with a fixed, finite amount of memory.





Formalizing Things

Strings

- An ***alphabet*** is a finite, nonempty set of symbols called ***characters***.
  - Typically, we use the symbol **Σ** to refer to an alphabet.
- A ***string over an alphabet Σ*** is a finite sequence of characters drawn from Σ.
- Example: Let Σ = {**a**, **b**}. Here are some strings over Σ:

  a   **aabaaabbabaaabaaabbb**   **abbababba**

- The ***empty string*** has no characters and is denoted **ε**.
  - Calling attention to an earlier point: since all strings are finite sequences of characters from Σ, you cannot have a string of infinite length.

Languages

- A ***formal language*** is a set of strings.
- We say that L is a ***language over Σ*** if it is a set of strings over Σ.
- Example: The language of palindromes over Σ = {**a**, **b**, **c**} is the set
  - {ε, **a**, **b**, **c**, **aa**, **bb**, **cc**, **aaa**, **aba**, **aca**, **bab**, … }
- The set of all strings composed from letters in Σ is denoted **Σ***.
- Formally, we say that L is a language over Σ if $L \subseteq \Sigma^*$.

Mathematical Lookalikes

- We now have ∈, ε, Σ, and Σ*. Yikes!
- The symbol ∈ is the ***element-of*** relation.
- The symbol ε is the ***empty string***.
- The symbol Σ denotes an ***alphabet***.

  - The expression Σ* means "all strings that can be made from characters in Σ."
  - That lets us write things like

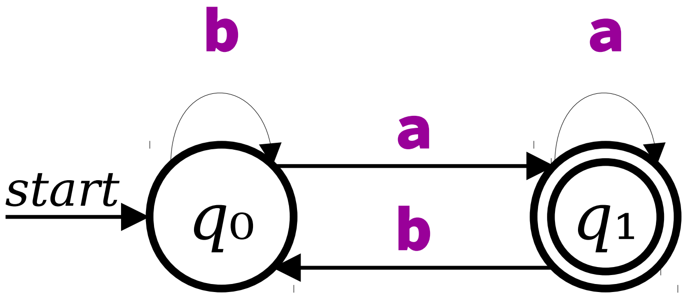
$$\text{We have } \varepsilon \in \Sigma^*, \text{ but } \varepsilon \notin \Sigma.$$

The Cast of Characters

- ***Languages*** are sets of strings.
- ***Strings*** are finite sequences of characters.
- ***Characters*** are individual symbols.
- ***Alphabets*** are sets of characters.

Finite Automata and Languages

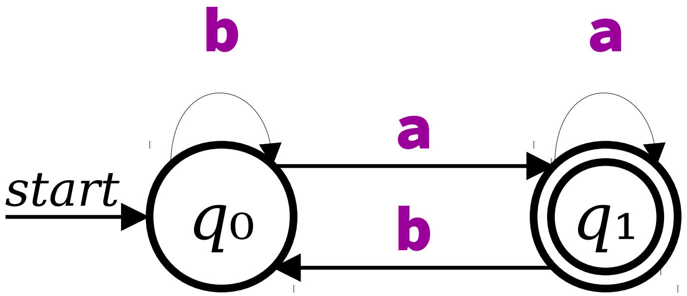
- Let A be an automaton that processes strings drawn from an alphabet Σ.
- The ***language of A***, denoted $\mathscr{L}(A)$, is the set of strings over Σ that A accepts:



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

Finite Automata and Languages

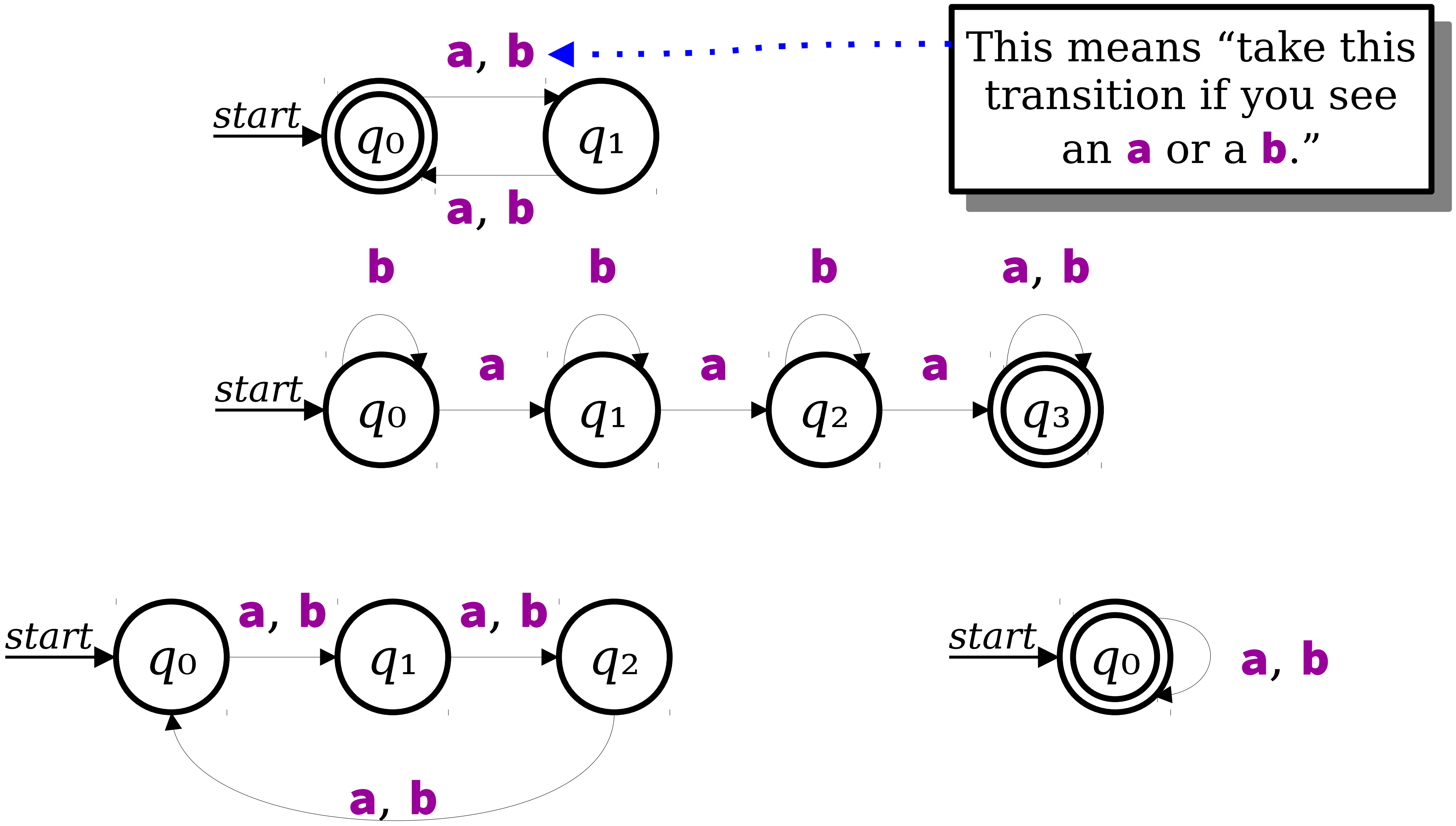
- Let D be the automaton shown to the right. It processes strings over {**a**, **b**}.
- Notice that D accepts all strings of **a**'s and **b**'s that end in **a** and rejects everything else.



- So $\mathscr{L}(D) = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ ends in } \mathbf{a} \}$.

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

Finite Automata and Languages

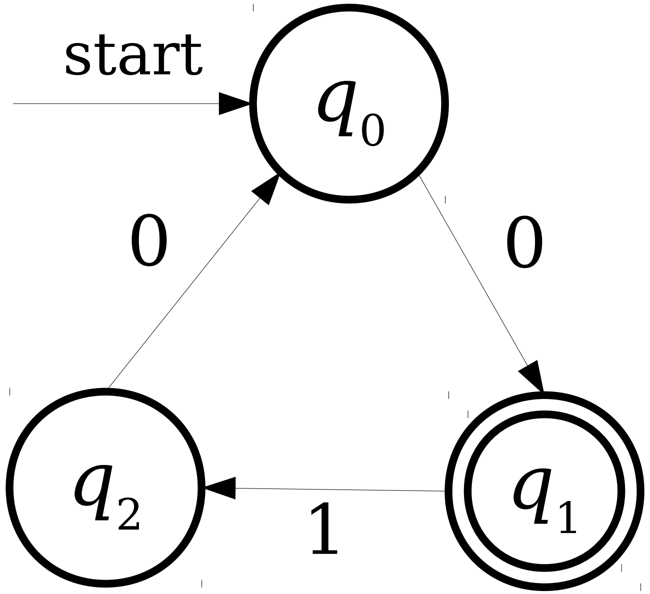


$\mathscr{L}(\boldsymbol{A}) = \{\ \boldsymbol{w} \in \boldsymbol{\Sigma^*} \mid \boldsymbol{A}$ **accepts** $\boldsymbol{w}\ \}$

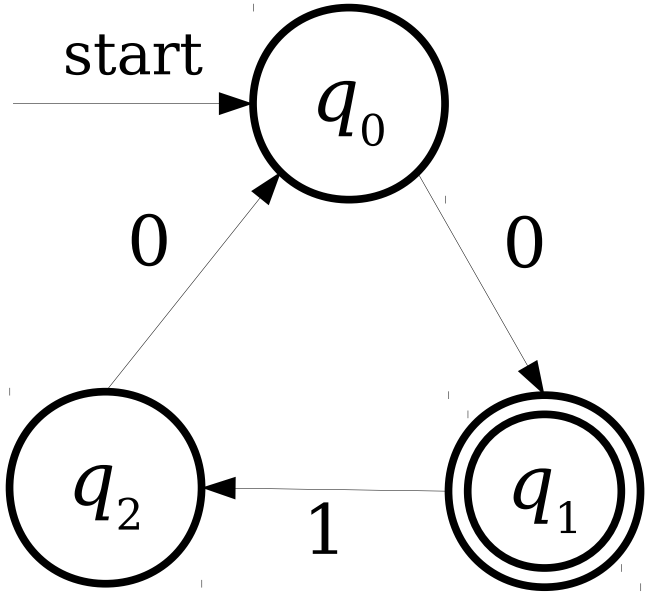
The Story So Far

- A ***finite automaton*** is a set of ***states*** joined by ***transitions***.
- Some state is designated as the ***start state***.
- Some subset of states are designated as ***accepting states***.
- The automaton processes a string by beginning in the start state and following the indicated transitions.
- If the automaton ends in an accepting state, it ***accepts*** the input.
- Otherwise, the automaton ***rejects*** the input.
- The ***language*** of an automaton is the set of strings it accepts.

A Small Problem

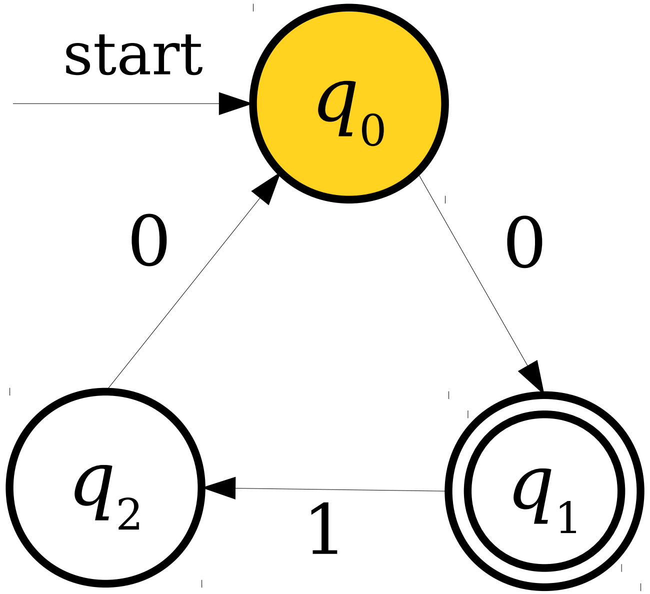


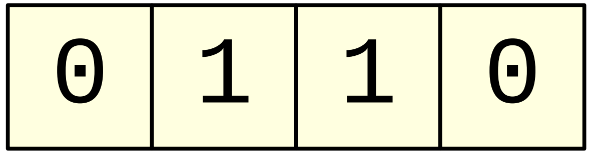
A Small Problem



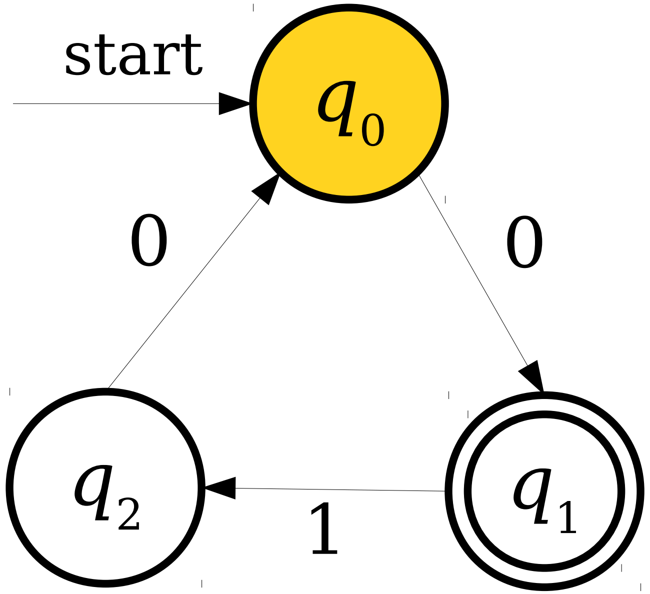
0	1	1	0

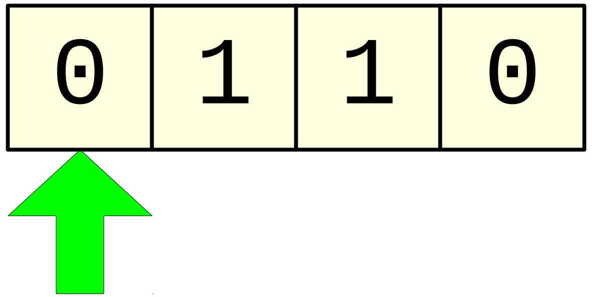
A Small Problem



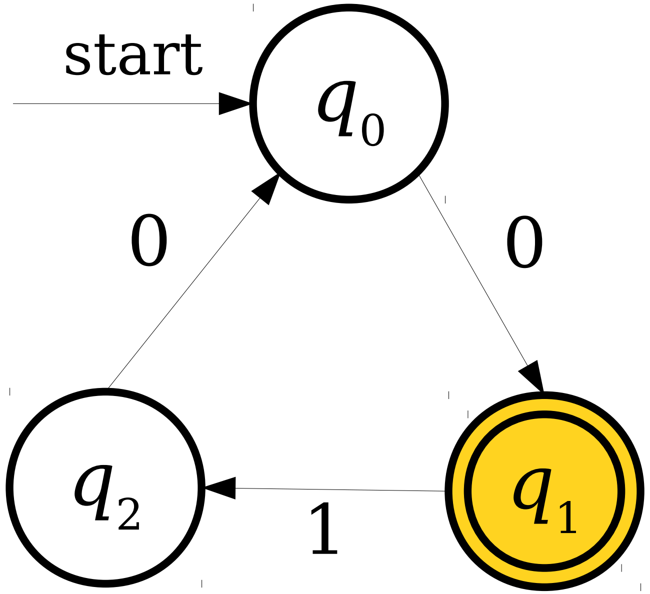


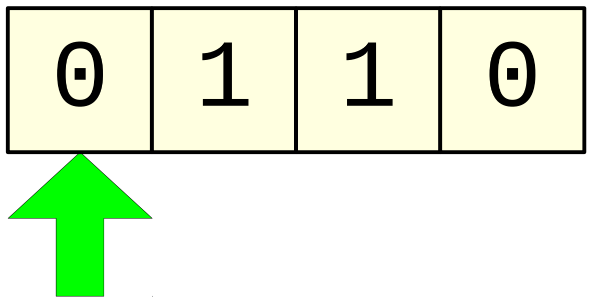
A Small Problem



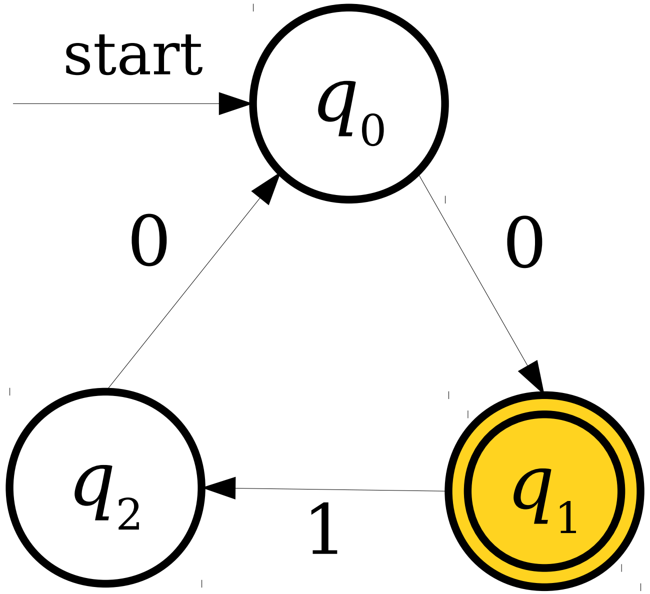


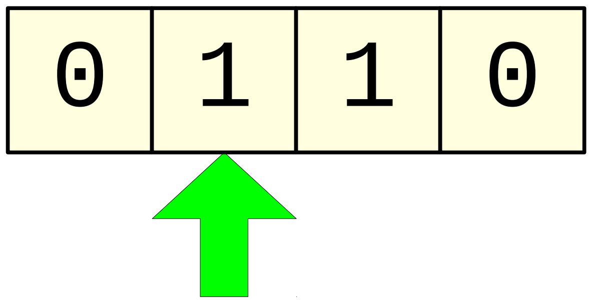
A Small Problem



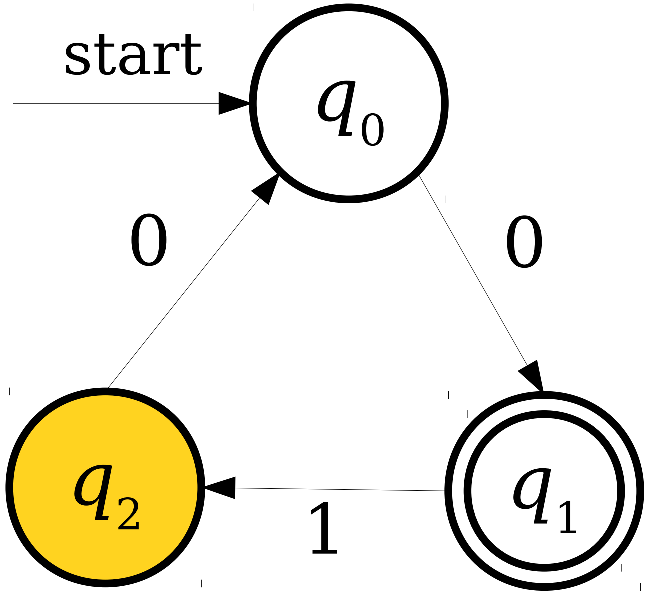


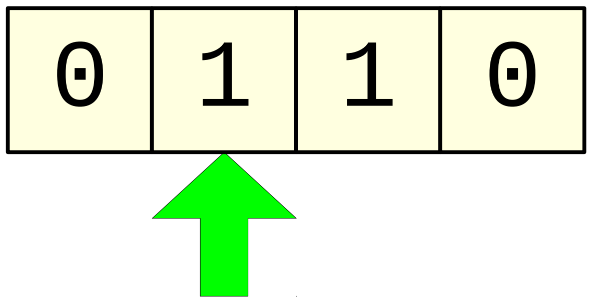
A Small Problem



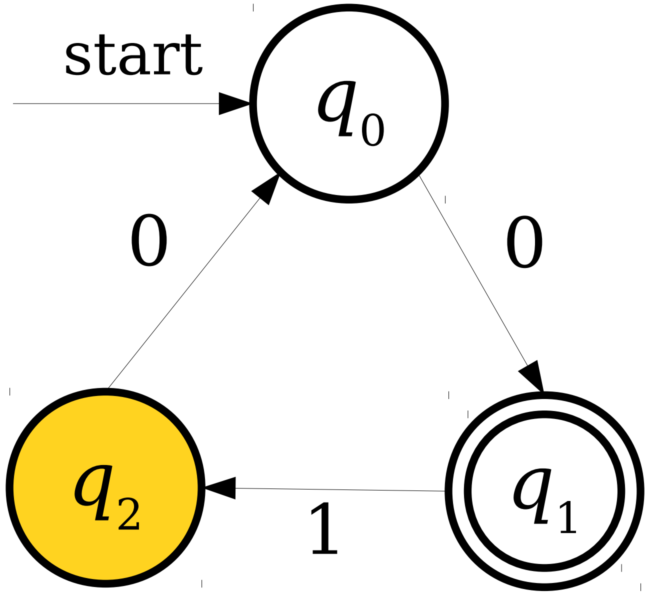


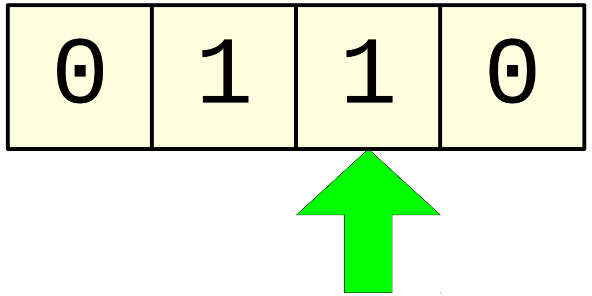
A Small Problem



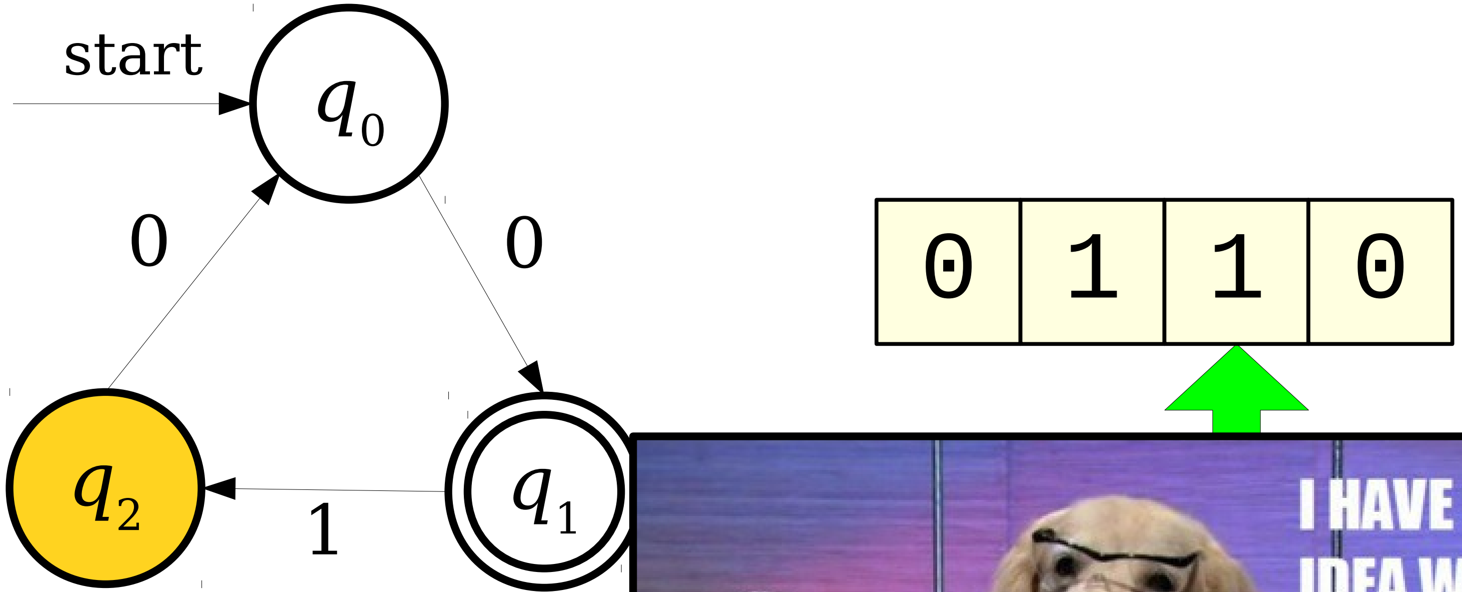


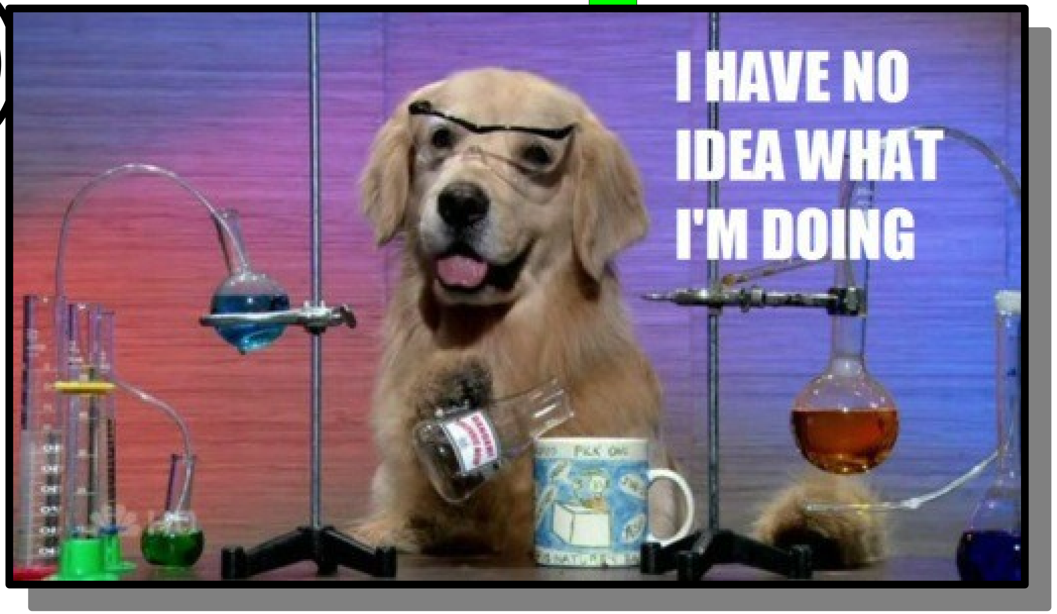
A Small Problem



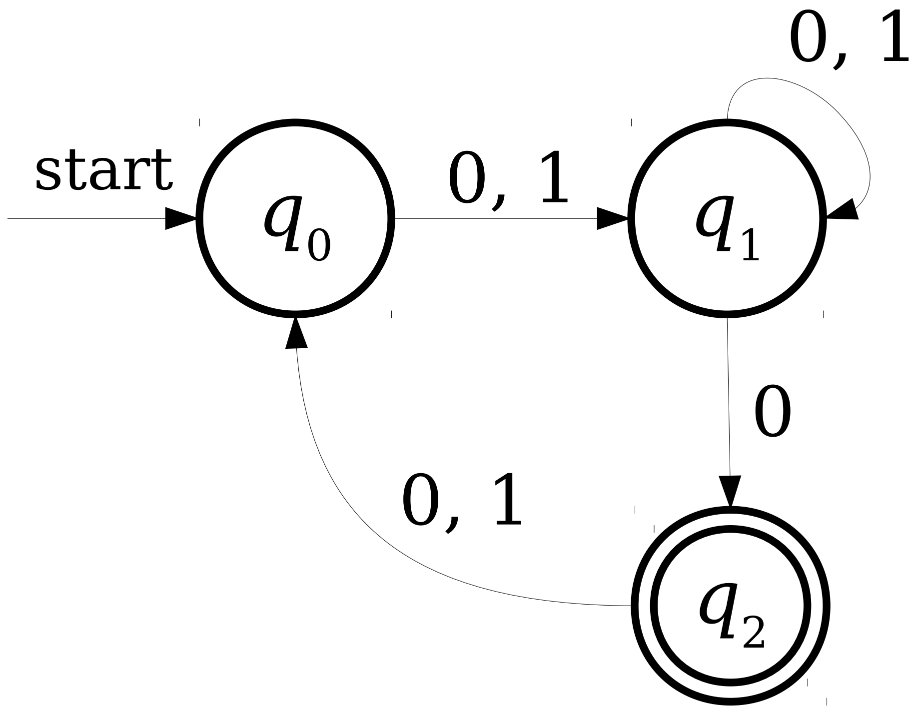


A Small Problem

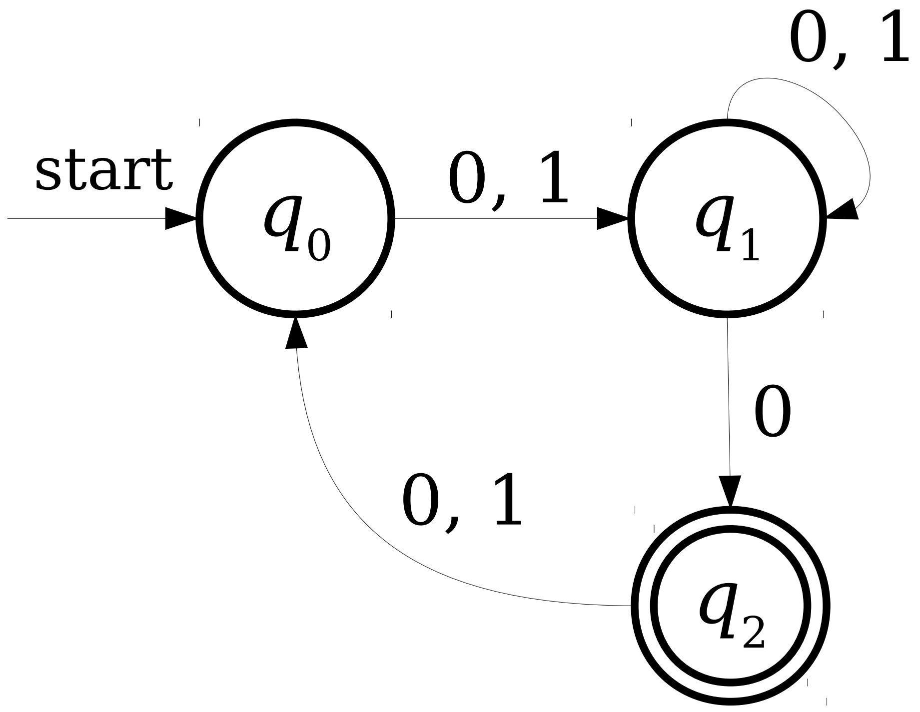


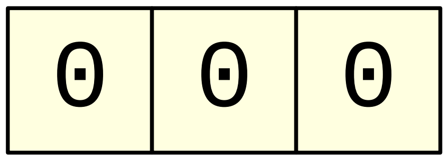


Another Small Problem

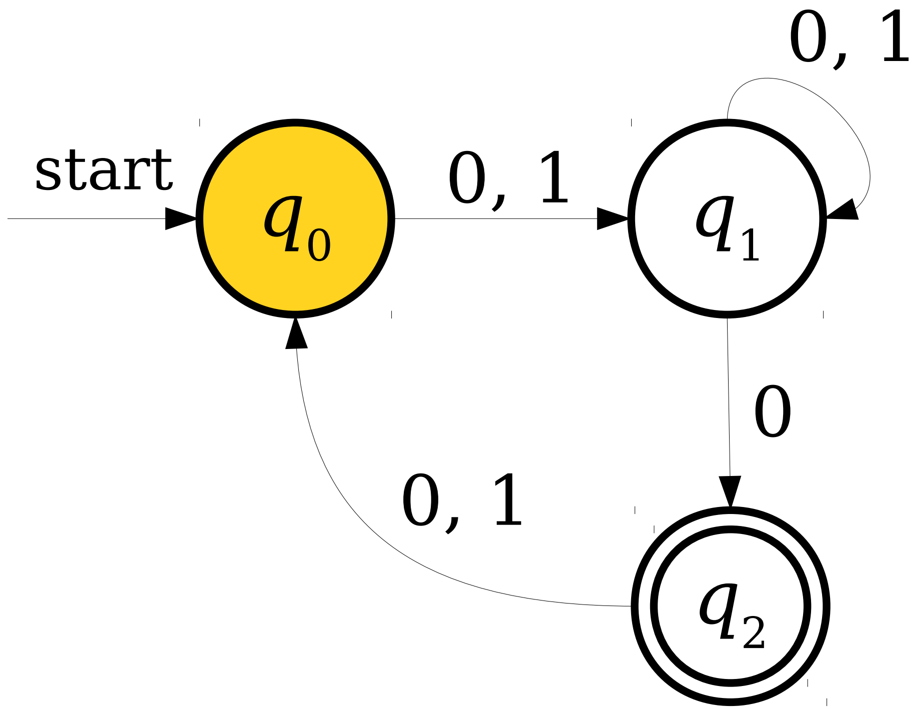


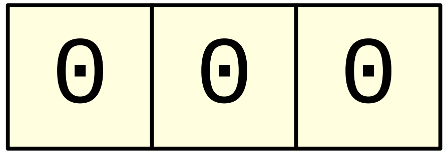
Another Small Problem



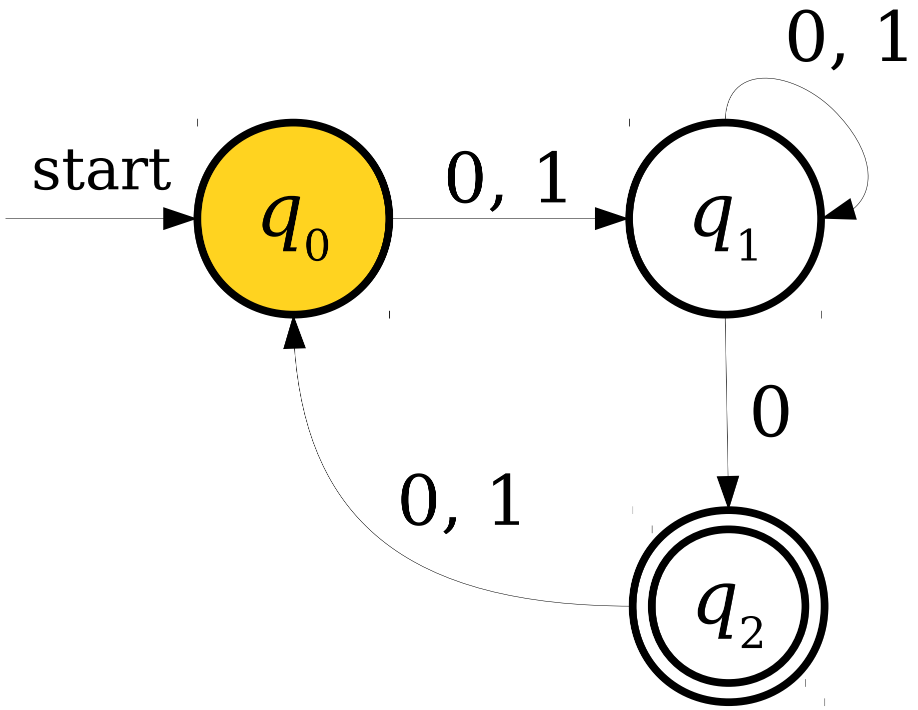


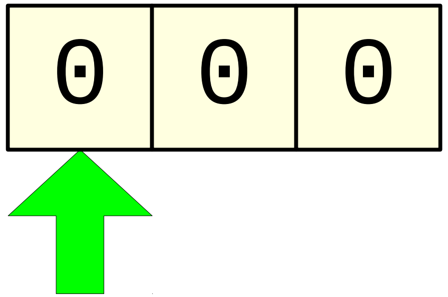
Another Small Problem



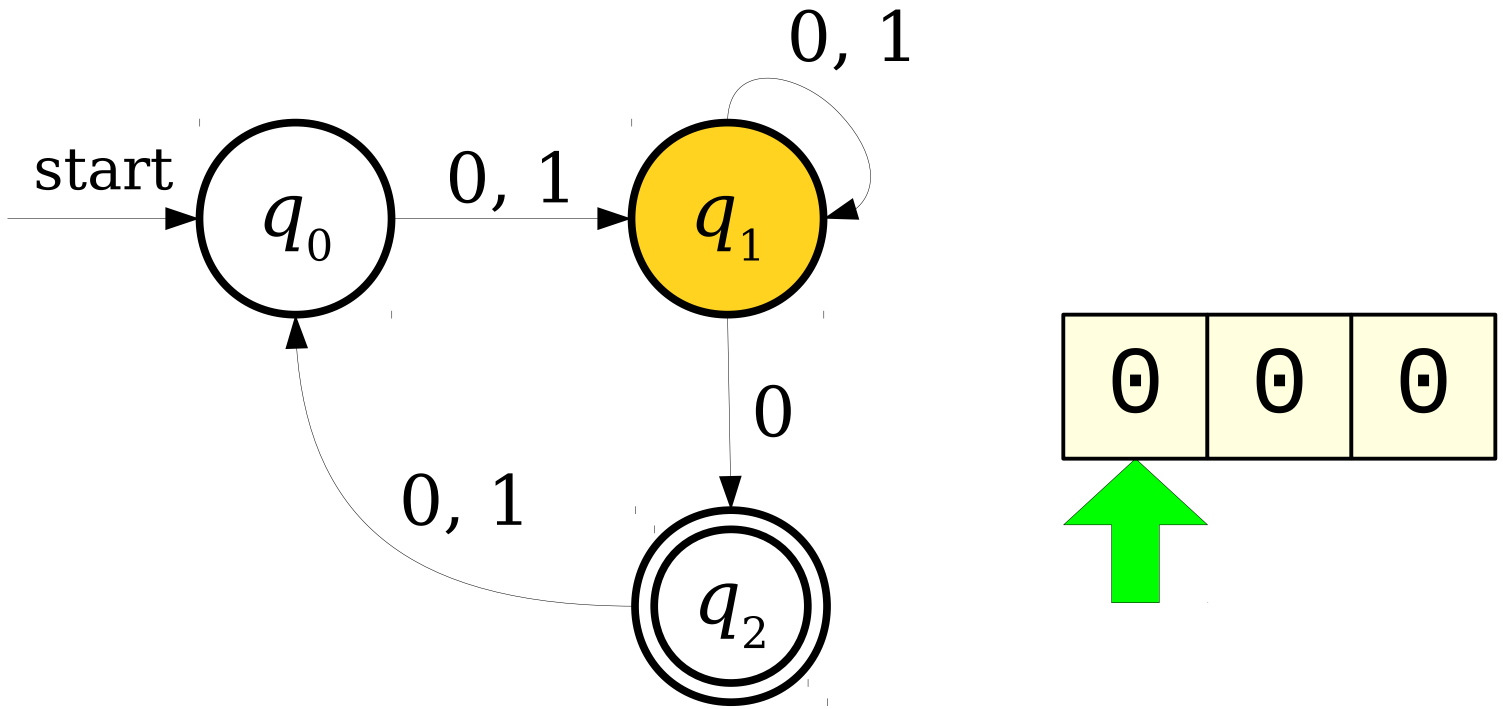


Another Small Problem

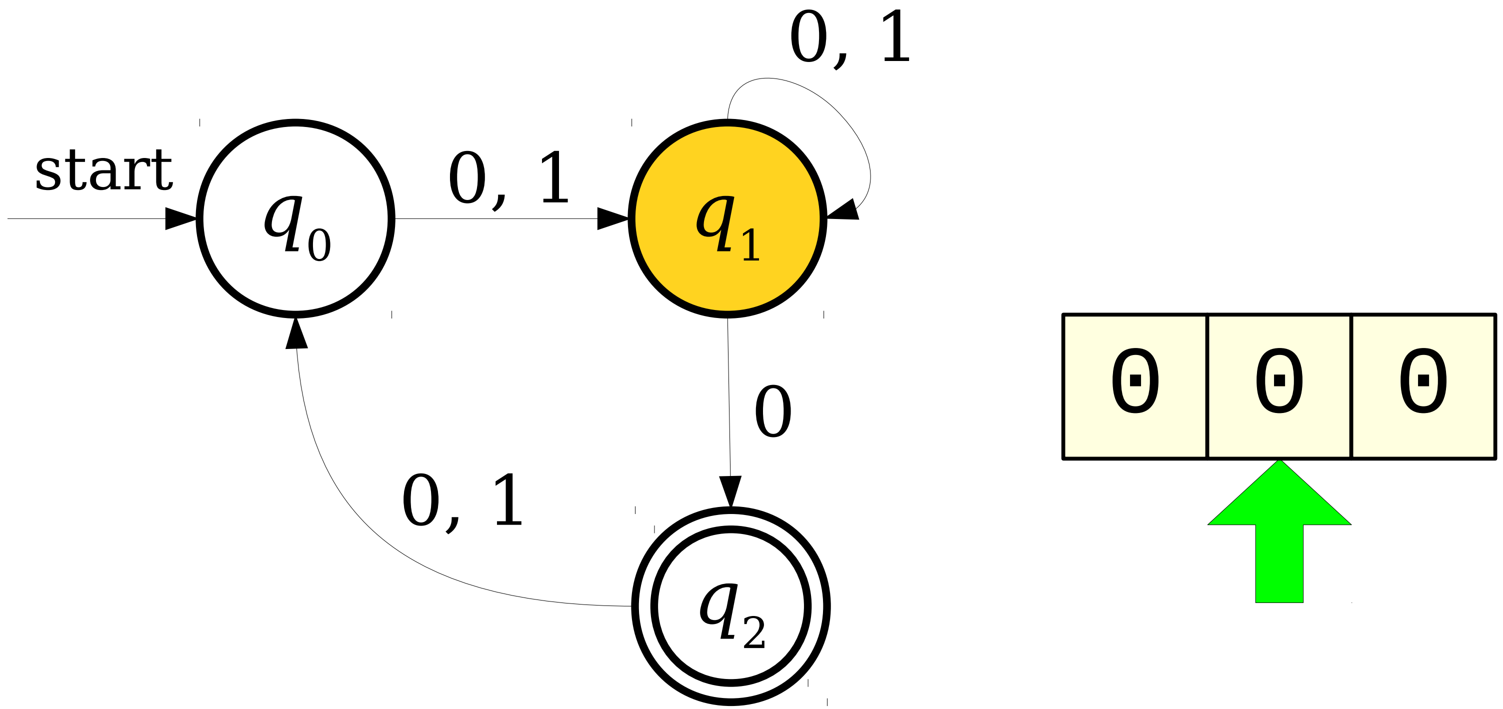




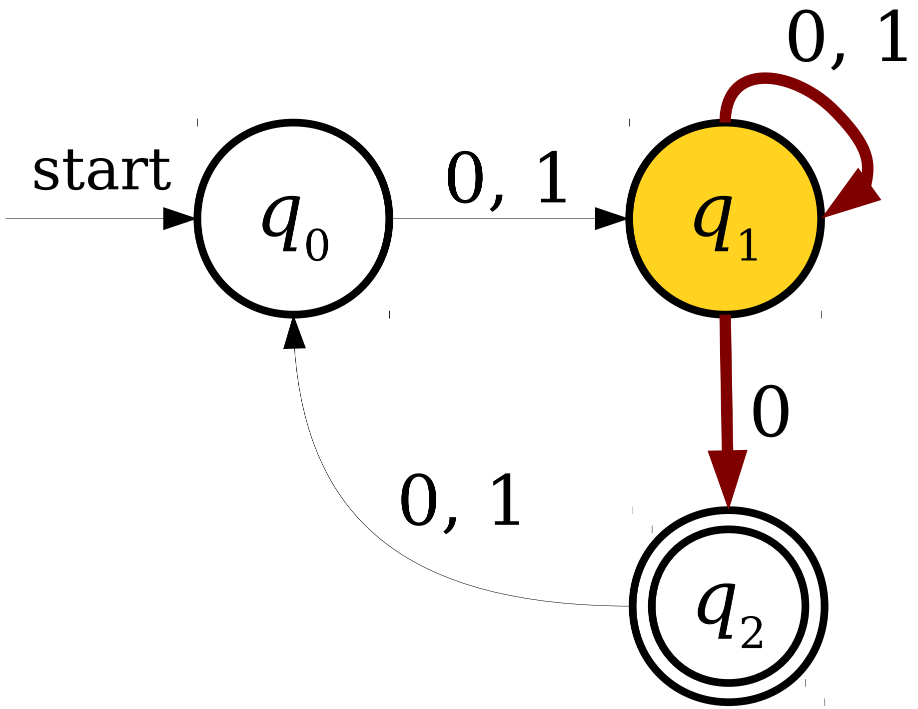
Another Small Problem

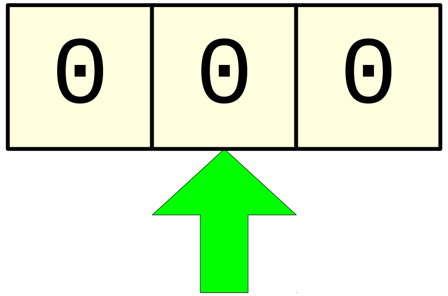


Another Small Problem

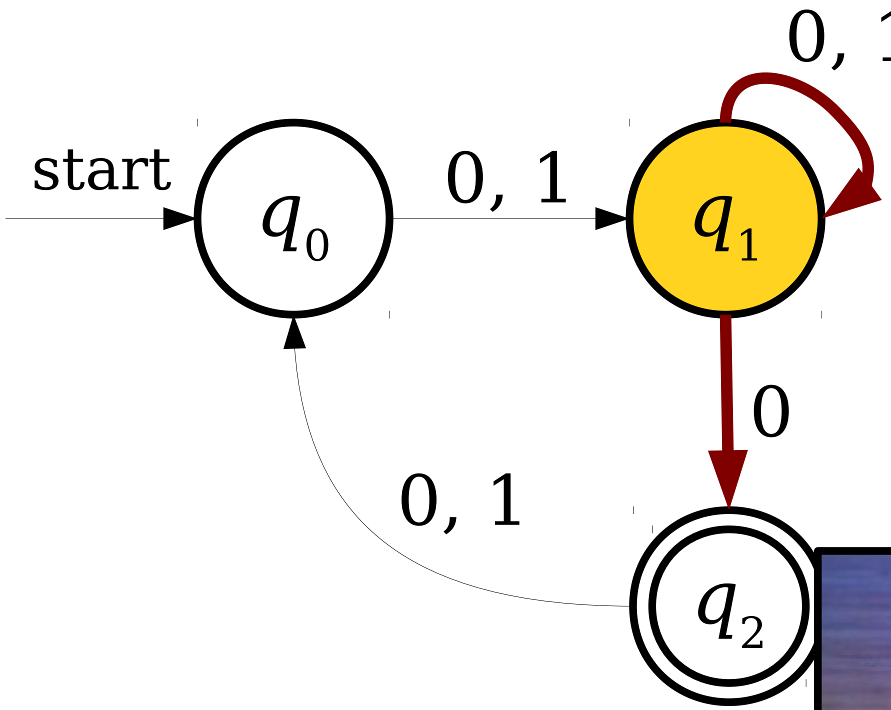


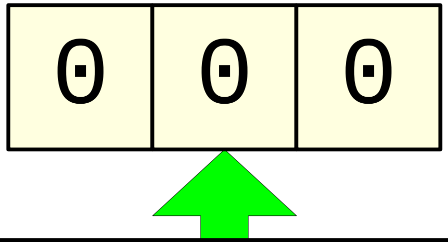
Another Small Problem

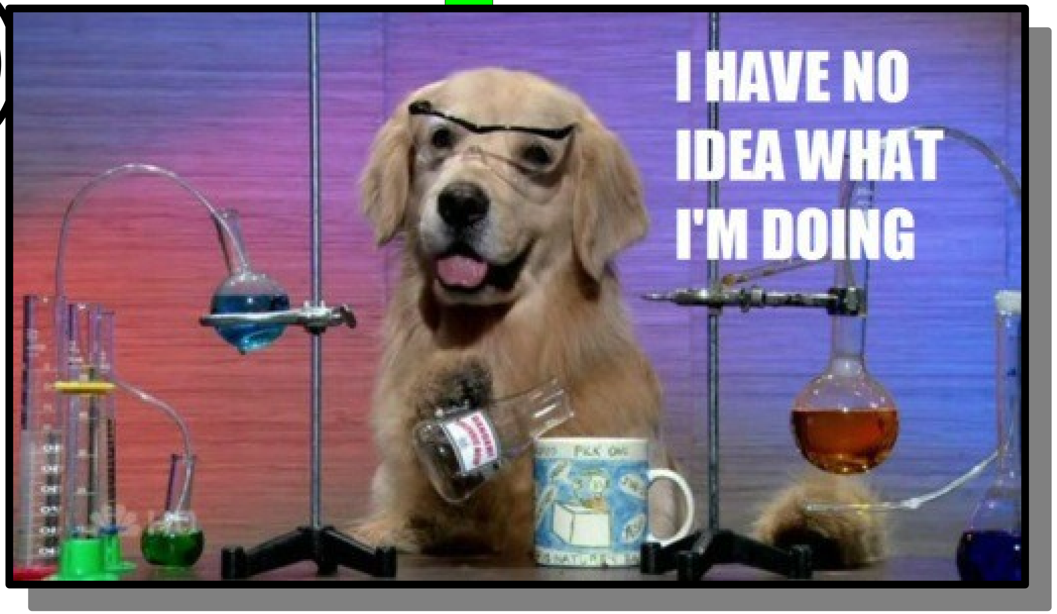




Another Small Problem







The Need for Formalism

- In order to reason about the limits of what finite automata can and cannot do, we need to formally specify their behavior in *all* cases.
- All of the following need to be defined or disallowed:
  - What happens if there is no transition out of a state on some input?
  - What happens if there are *multiple* transitions out of a state on some input?

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.

Is this a DFA?

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

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

Recognizing Languages with DFAs

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

Ask yourself these design questions:

What trait(s) of the string so far do I need to keep track of while processing?

For each trait, how many meaningfully distinct configurations of that trait are there that I need to keep track of?

Recognizing Languages with DFAs

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

Ask yourself these design questions:

What trait(s) of the string so far do I need to keep track of while processing?

For each trait, how many meaningfully distinct configurations of that trait are there that I need to keep track of?

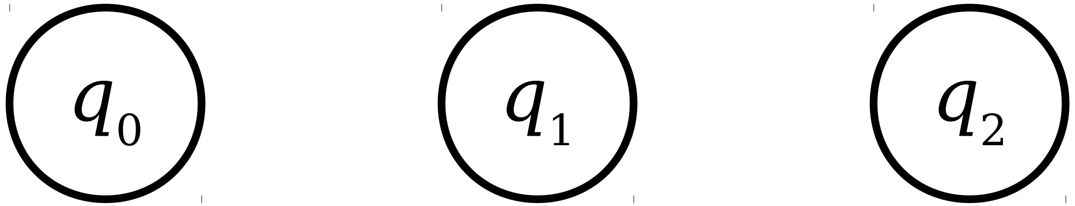
1 trait: count of b's mod 3

3 configurations: 0, 1, 2

Conclusion: we'll make 3 states: one each for 0, 1, 2

Recognizing Languages with DFAs

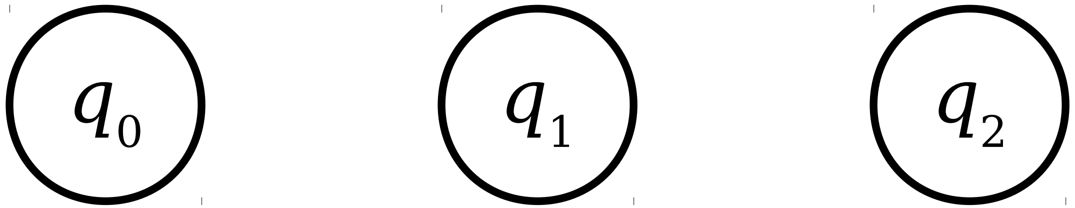
$L = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid$ the number of **b**'s in w is congruent to two modulo three $\}$



Conclusion: we'll make 3 states: one each for 0, 1, 2

Recognizing Languages with DFAs

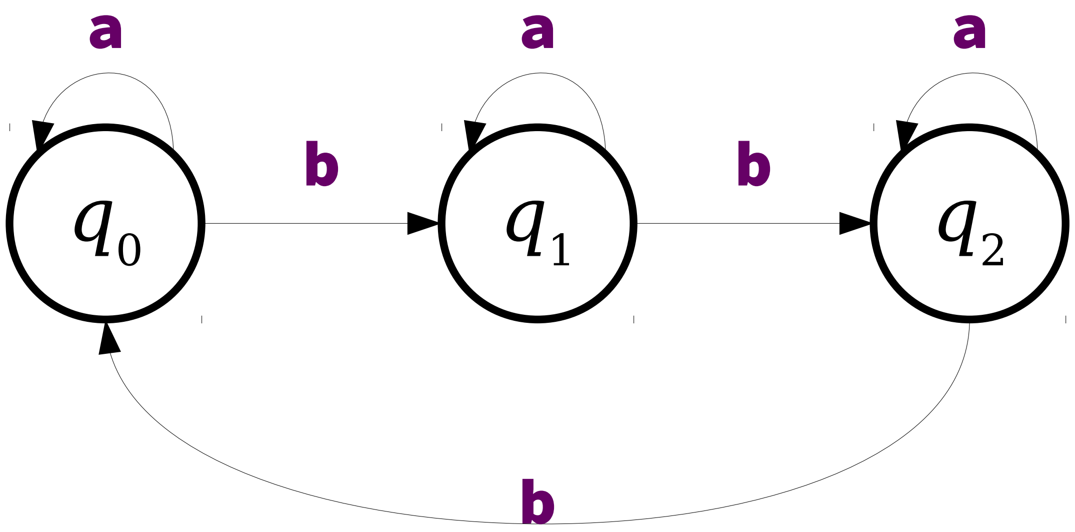
$L = \{\ w \in \{$**a**, **b**$\}^* \mid$ the number of **b**'s in w is congruent to two modulo three $\}$



Only after putting down all the states you'll need, think about what moves you from state to state, and connect them.

Recognizing Languages with DFAs

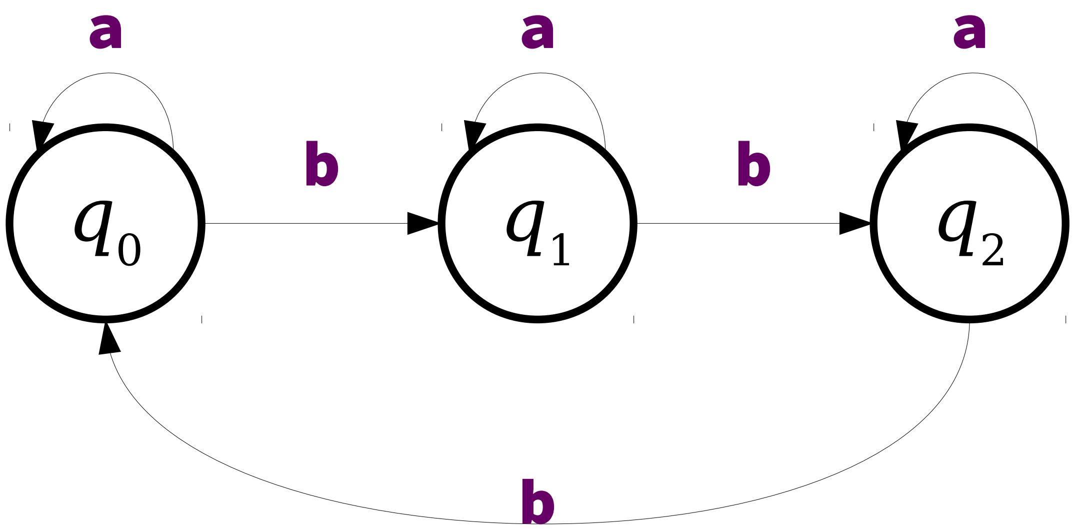
$L = \{ w \in \{$**a**, **b**$\}^*|$ the number of **b**'s in w is congruent to two modulo three $\}$



Seeing a letter **b** changes our status in terms of count of **b**'s modulo three, but seeing **a** doesn't change our status.

Recognizing Languages with DFAs

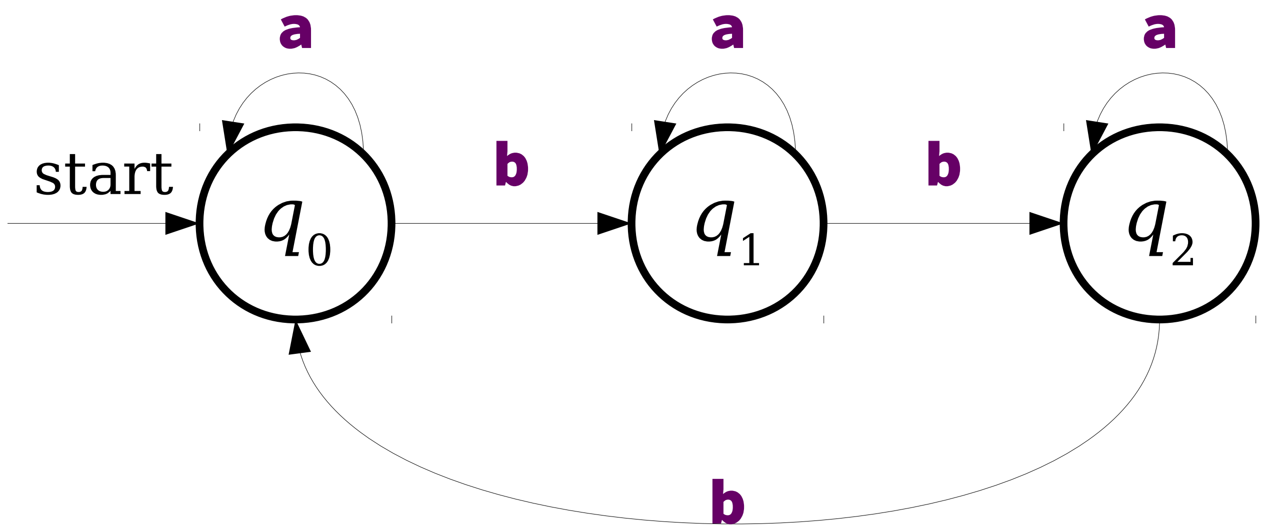
$L = \{ w \in \{$**a**, **b**$\}^* \mid$ the number of **b**'s in w is congruent to two modulo three $\}$



Now ask yourself: what is your status before you read any input? That configuration is our start state.

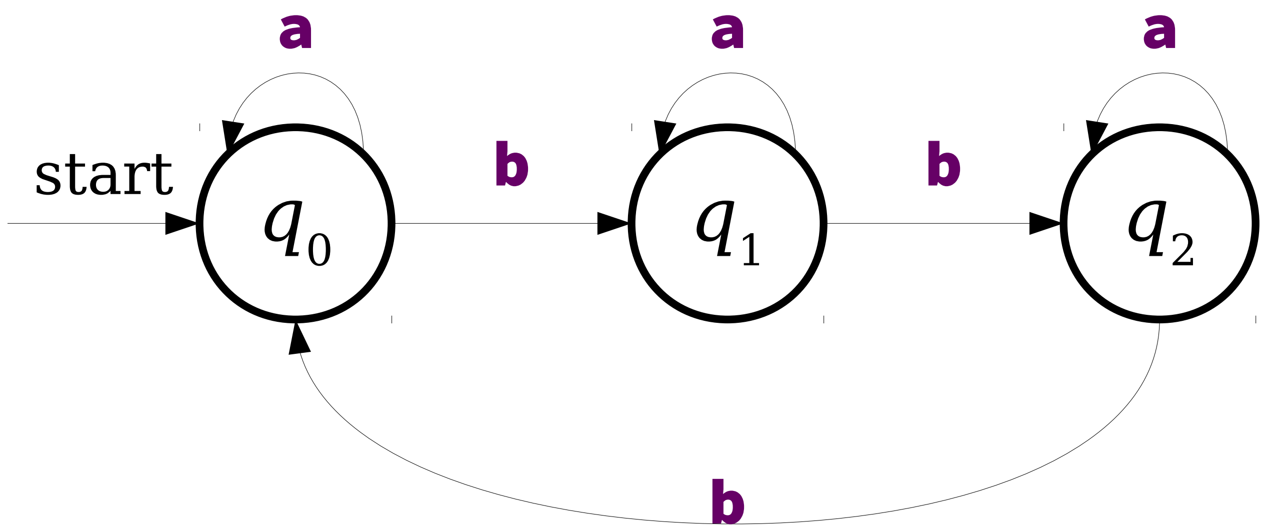
Recognizing Languages with DFAs

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



Recognizing Languages with DFAs

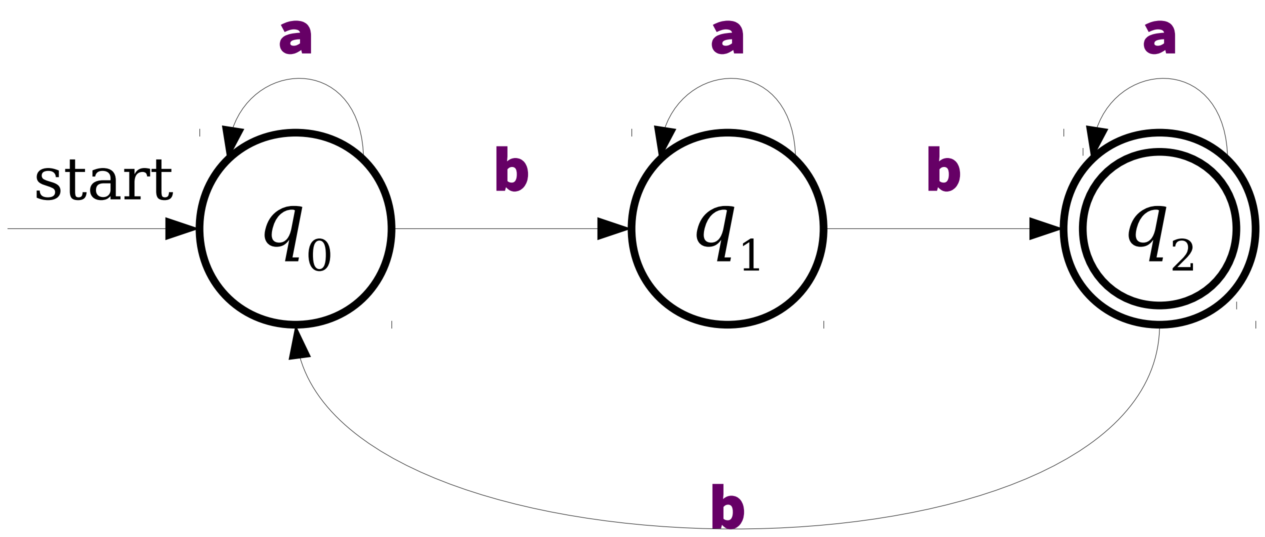
$L = \{\ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid$ the number of $\mathbf{b}$'s in w is congruent to two modulo three $\}$



Now ask yourself: what status do we want to have at the end? Those configuration(s) are our accepting states.

Recognizing Languages with DFAs

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



Recognizing Languages with DFAs

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

Ask yourself these design questions:

What trait(s) of the string so far do I need to keep track of while processing?

For each trait, how many meaningfully distinct configurations of that trait are there that I need to keep track of?

Recognizing Languages with DFAs

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

Ask yourself these design questions:

What trait(s) of the string so far do I need to keep track of while processing?

For each trait, how many meaningfully distinct configurations of that trait are there that I need to keep track of?

1 trait: count of consecutive a's seen so far

3 configurations: 0, 1, 2+

Recognizing Languages with DFAs

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

Ask yourself these design questions:

What trait(s) of the string so far do I need to keep track of while processing?

For each trait, how many meaningfully distinct configurations of that trait are there that I need to keep track of?

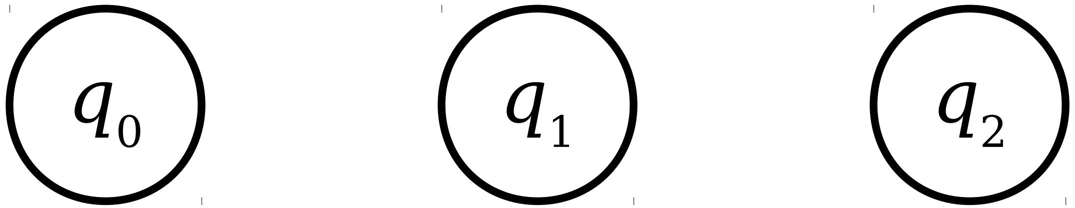
1 trait: count of consecutive a's seen so far

3 configurations: 0, 1, 2+

Conclusion: we'll make 3 states: one each for 0, 1, 2+

Recognizing Languages with DFAs

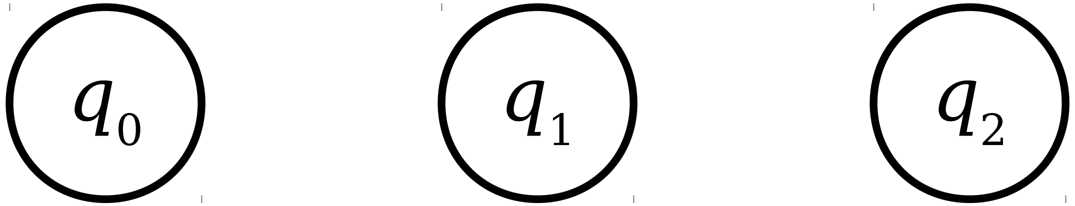
$L = \{\ w \in \{$**a**, **b**$\}^* \mid w$ contains **aa** as a substring $\}$



Conclusion: we'll make 3 states: one each for 0, 1, 2+

Recognizing Languages with DFAs

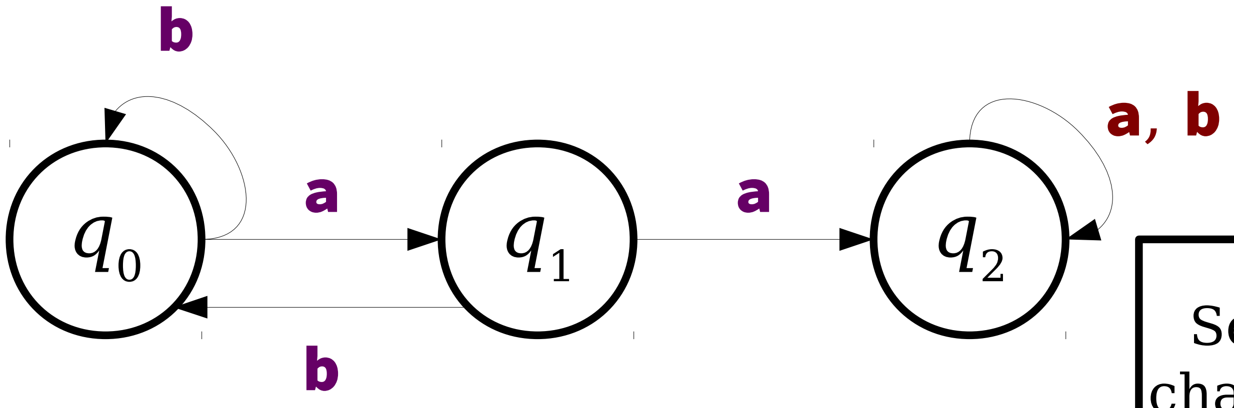
$L = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w$ contains **aa** as a substring $\}$



Only after putting down all the states you'll need, think about what moves you from state to state, and connect them.

Recognizing Languages with DFAs

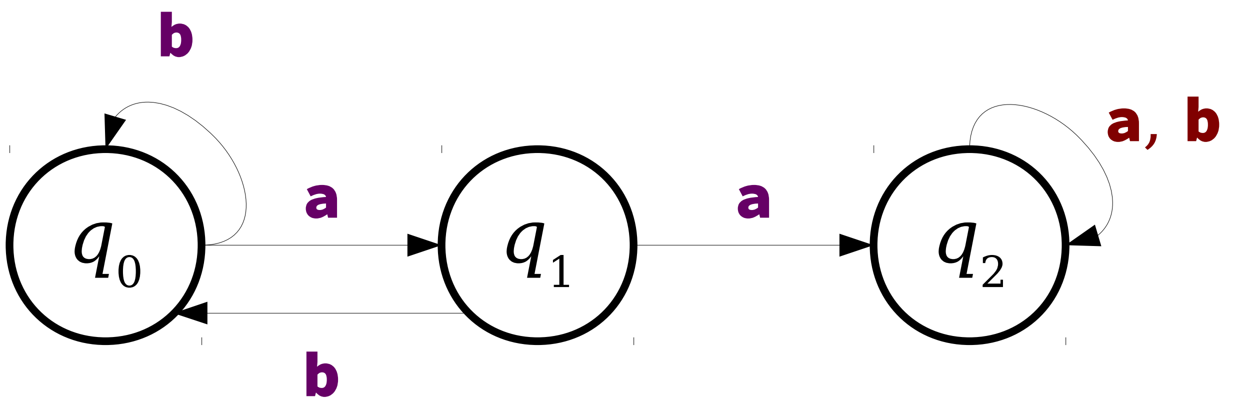
$L = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ contains } \mathbf{aa} \text{ as a substring } \}$



Seeing a letter **b** at the beginning changes nothing. Seeing an **a** means we advance our count. Seeing a **b** restarts our count. Once we've seen it, nothing changes our status.

Recognizing Languages with DFAs

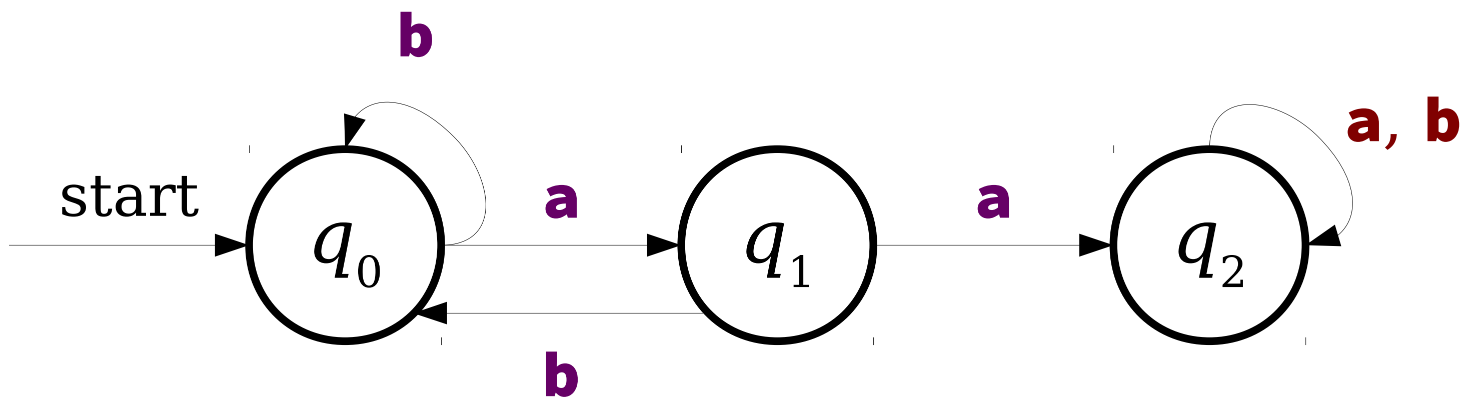
$L = \{ \, w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ contains } \mathbf{aa} \text{ as a substring } \}$



Now ask yourself: what is your status before you read any input? That configuration is our start state.

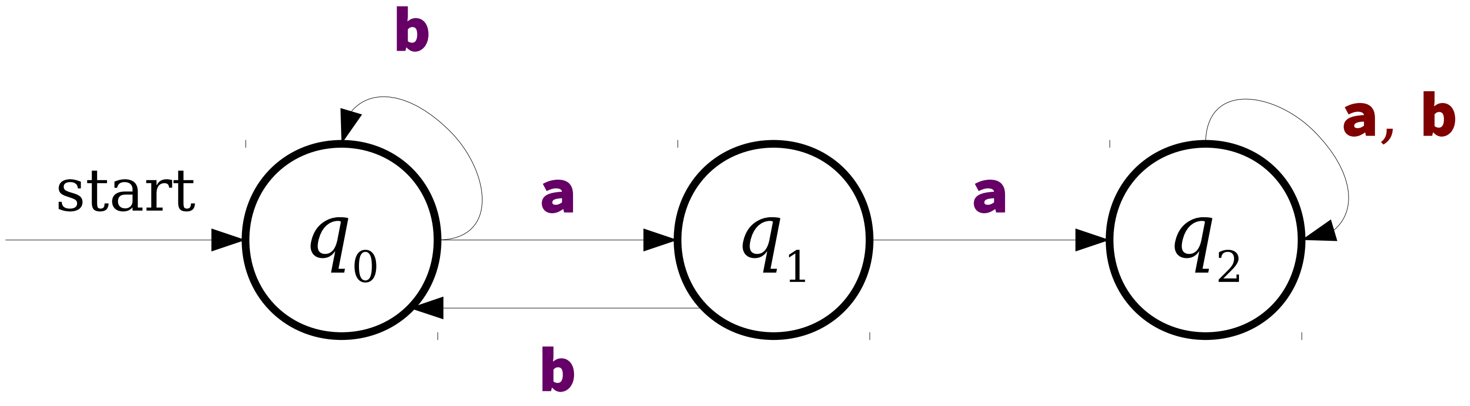
Recognizing Languages with DFAs

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



Recognizing Languages with DFAs

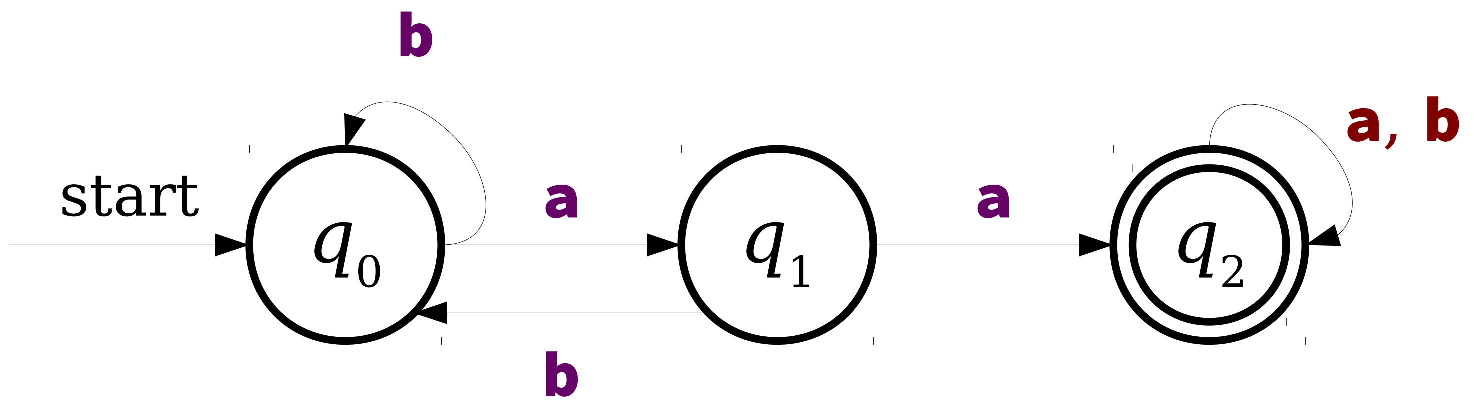
$L = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ contains } \mathbf{aa} \text{ as a substring } \}$



Now ask yourself: what status do we want to have at the end? Those configuration(s) are our accepting states.

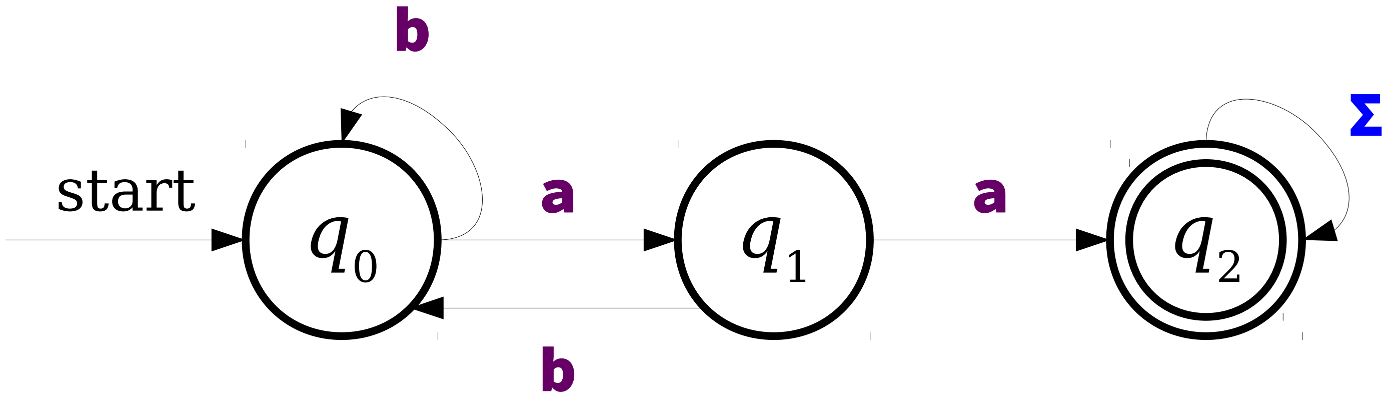
Recognizing Languages with DFAs

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



Recognizing Languages with DFAs

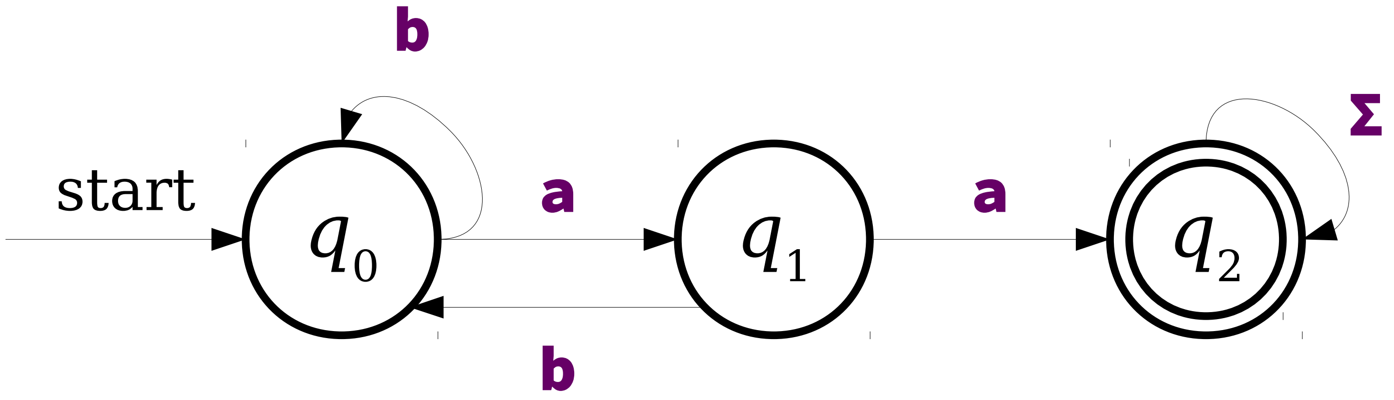
$L = \{ w \in \{\mathbf{a}, \mathbf{b}\}^* \mid w \text{ contains } \mathbf{aa} \text{ as a substring } \}$



Style note: if a transition label includes every character in the alphabet, we can just use this shorthand instead of a long list of characters.

Recognizing Languages with DFAs

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



Next Time

- ***Regular Languages***
  - An important class of languages.
- ***Nondeterministic Computation***
  - Why must computation be linear?
- ***NFAs***
  - Automata with Magic Superpowers.