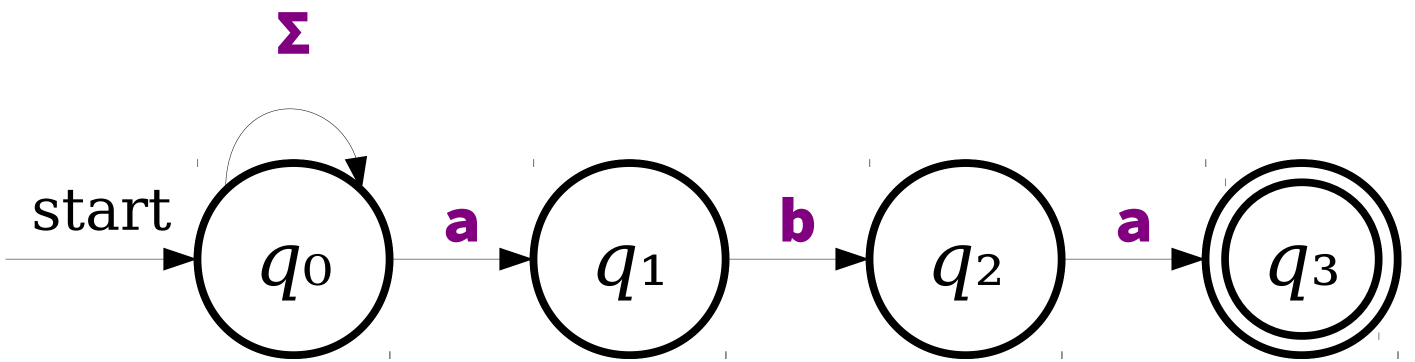
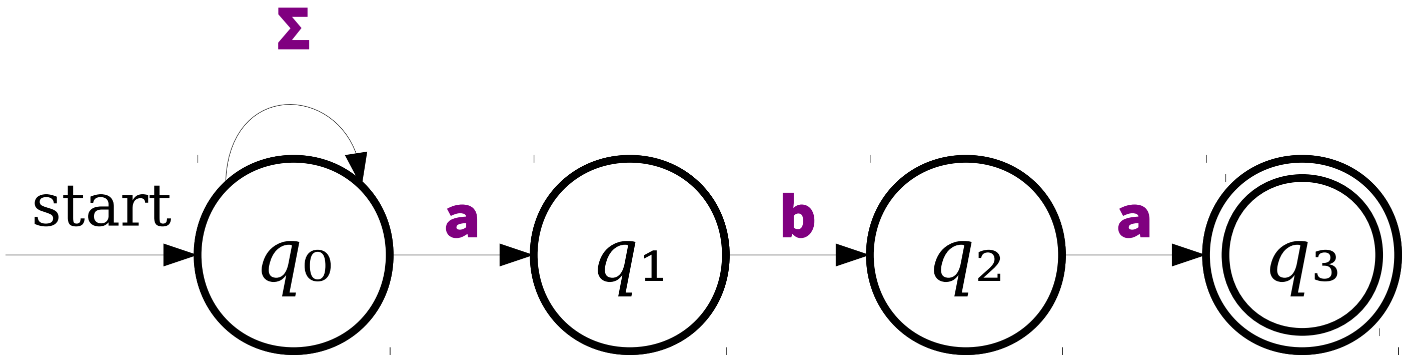
# Intuiting Nondeterminism

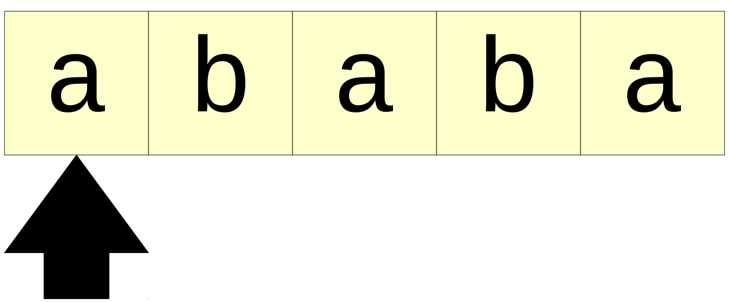
- Nondeterministic machines are a serious departure from physical computers. How can we build up an intuition for them?
- There are two particularly useful frameworks for interpreting nondeterminism:
  - ***Perfect guessing***
  - ***Massive parallelism***

# Perfect Guessing

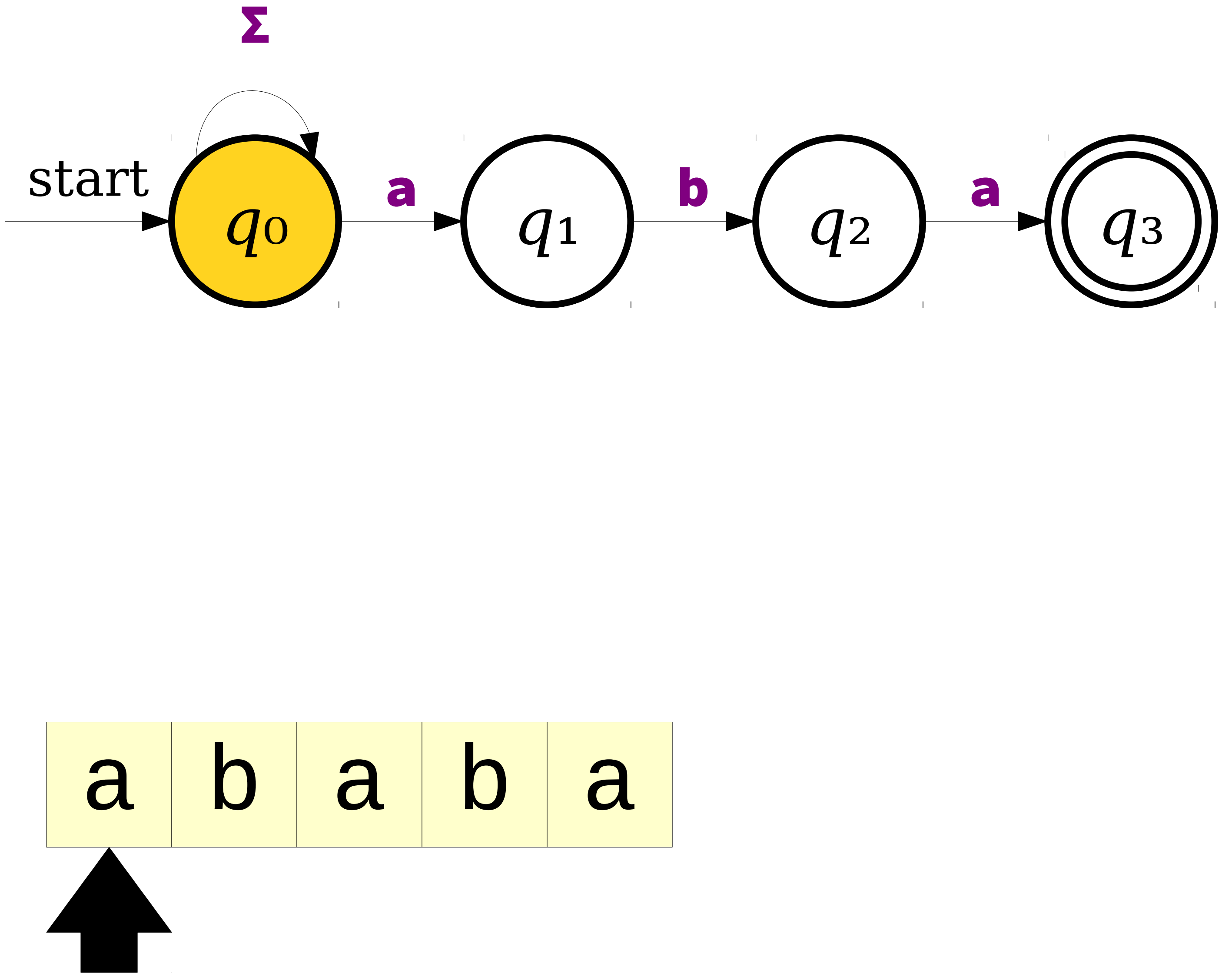


# Perfect Guessing

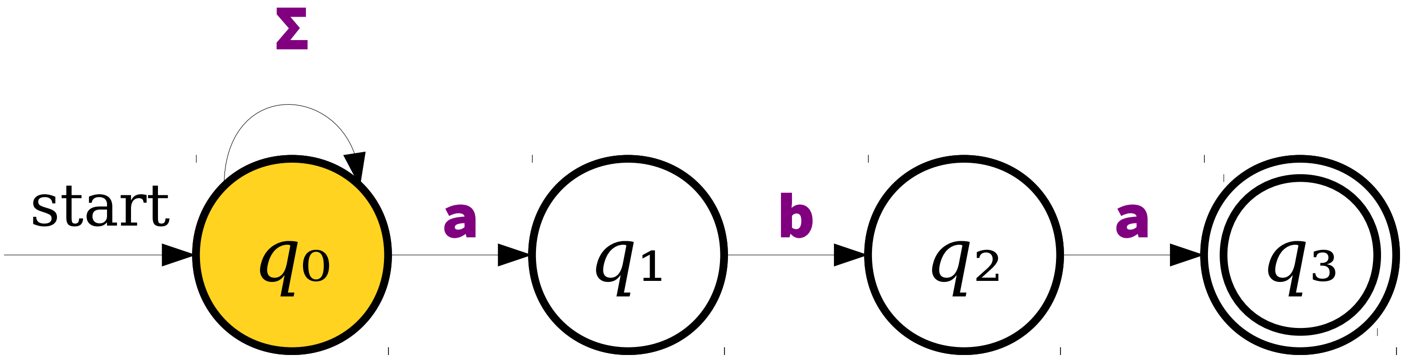


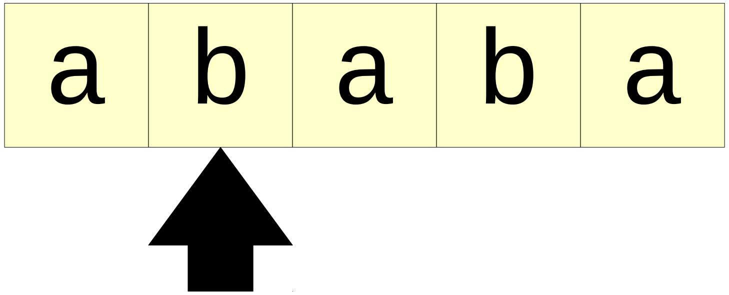



# Perfect Guessing

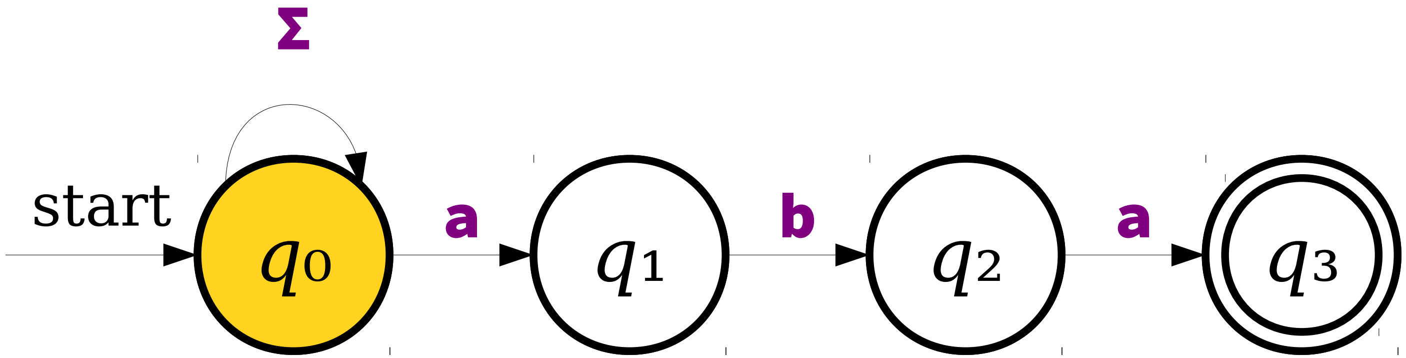


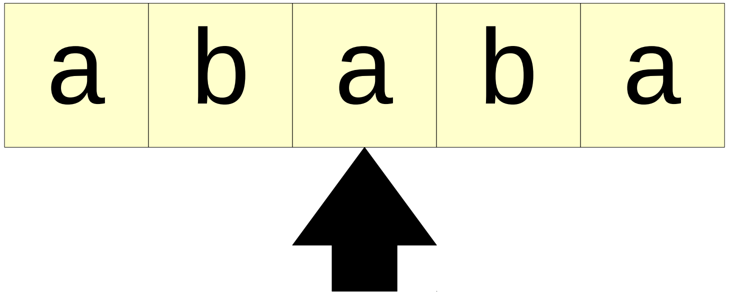
# Perfect Guessing



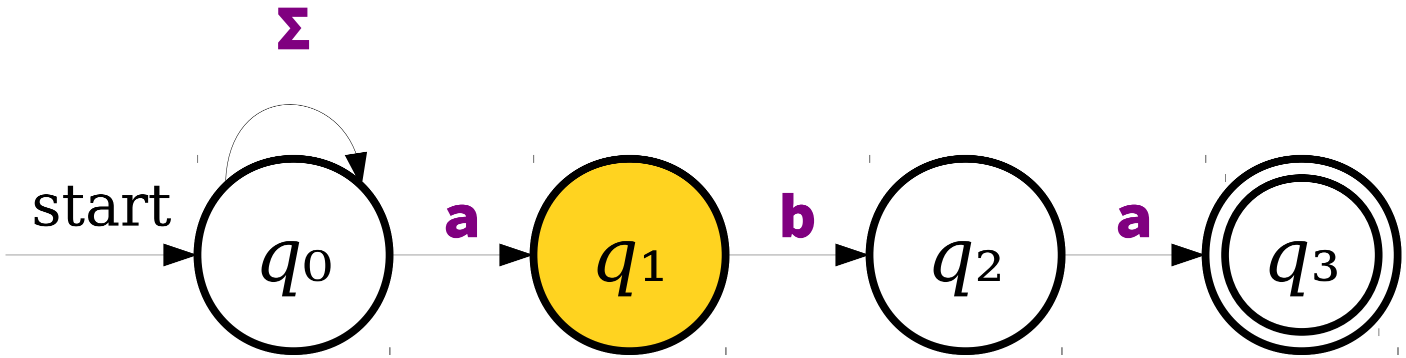



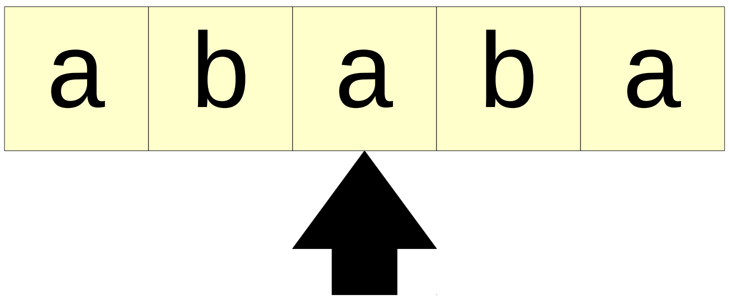
# Perfect Guessing



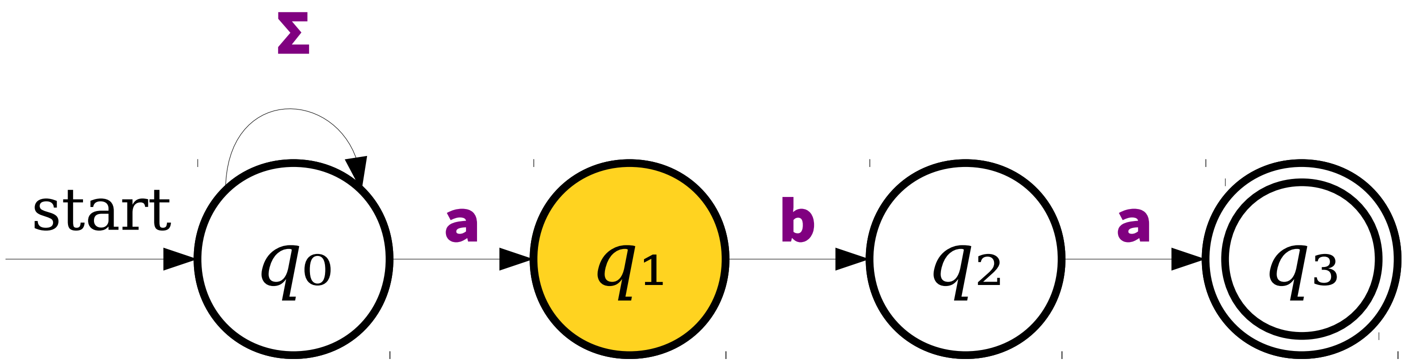



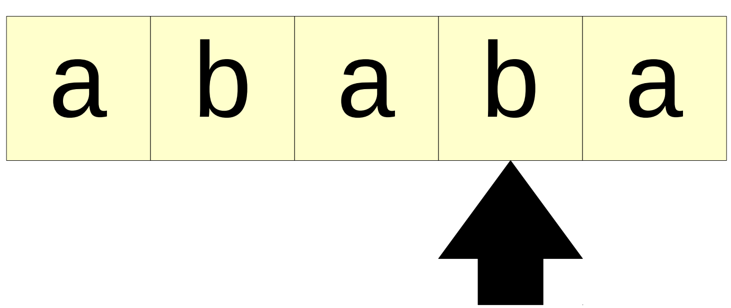
# Perfect Guessing



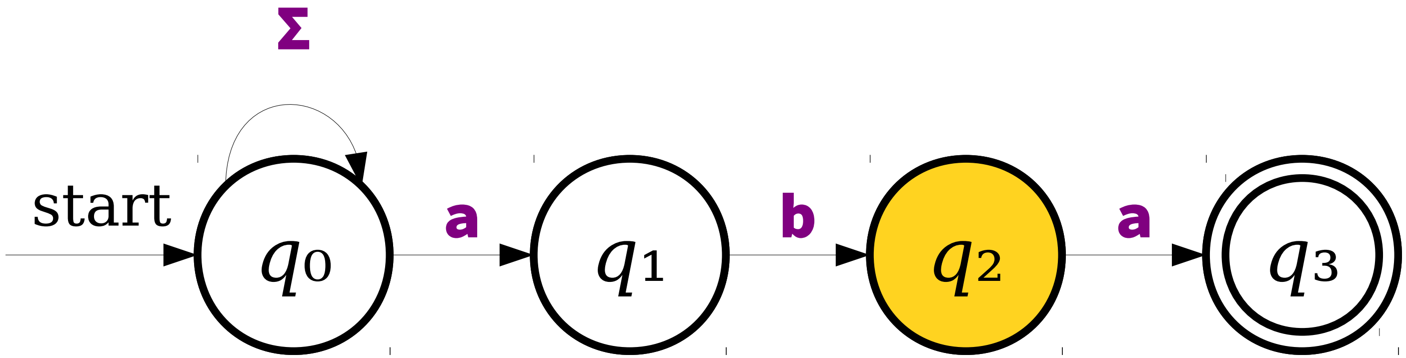



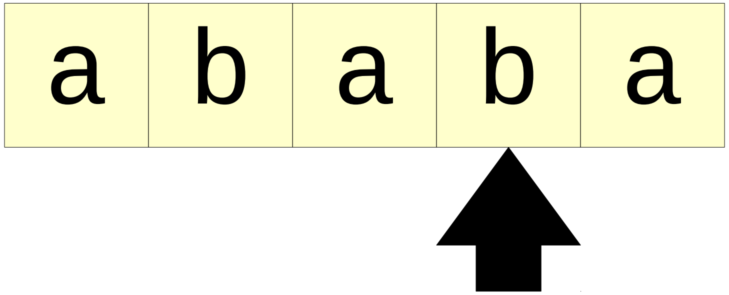
# Perfect Guessing



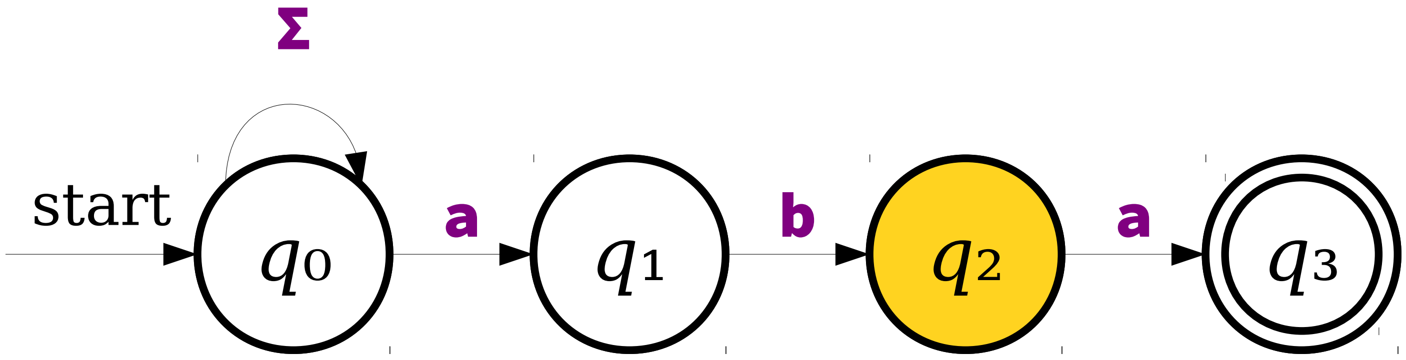



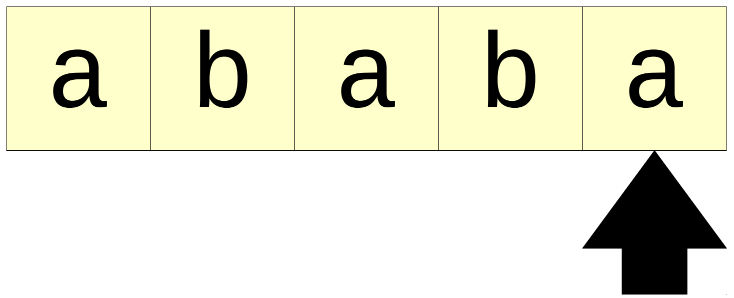
# Perfect Guessing



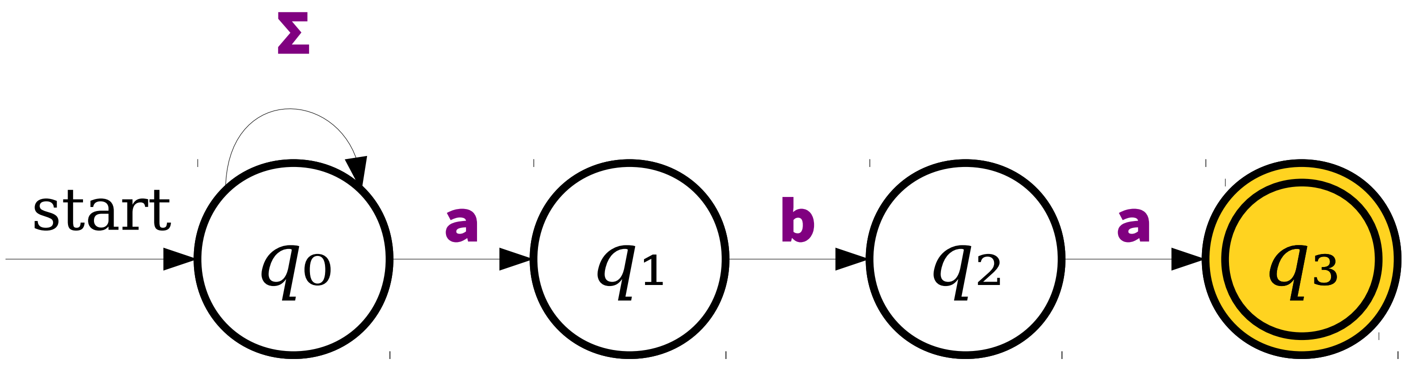



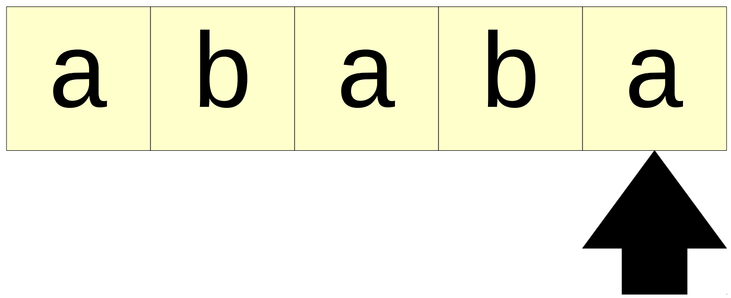
# Perfect Guessing



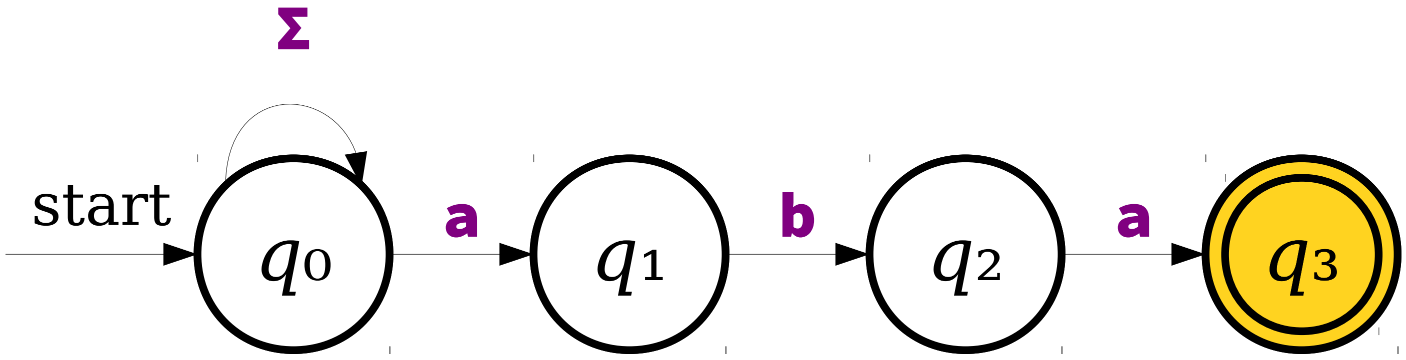



# Perfect Guessing



# Perfect Guessing

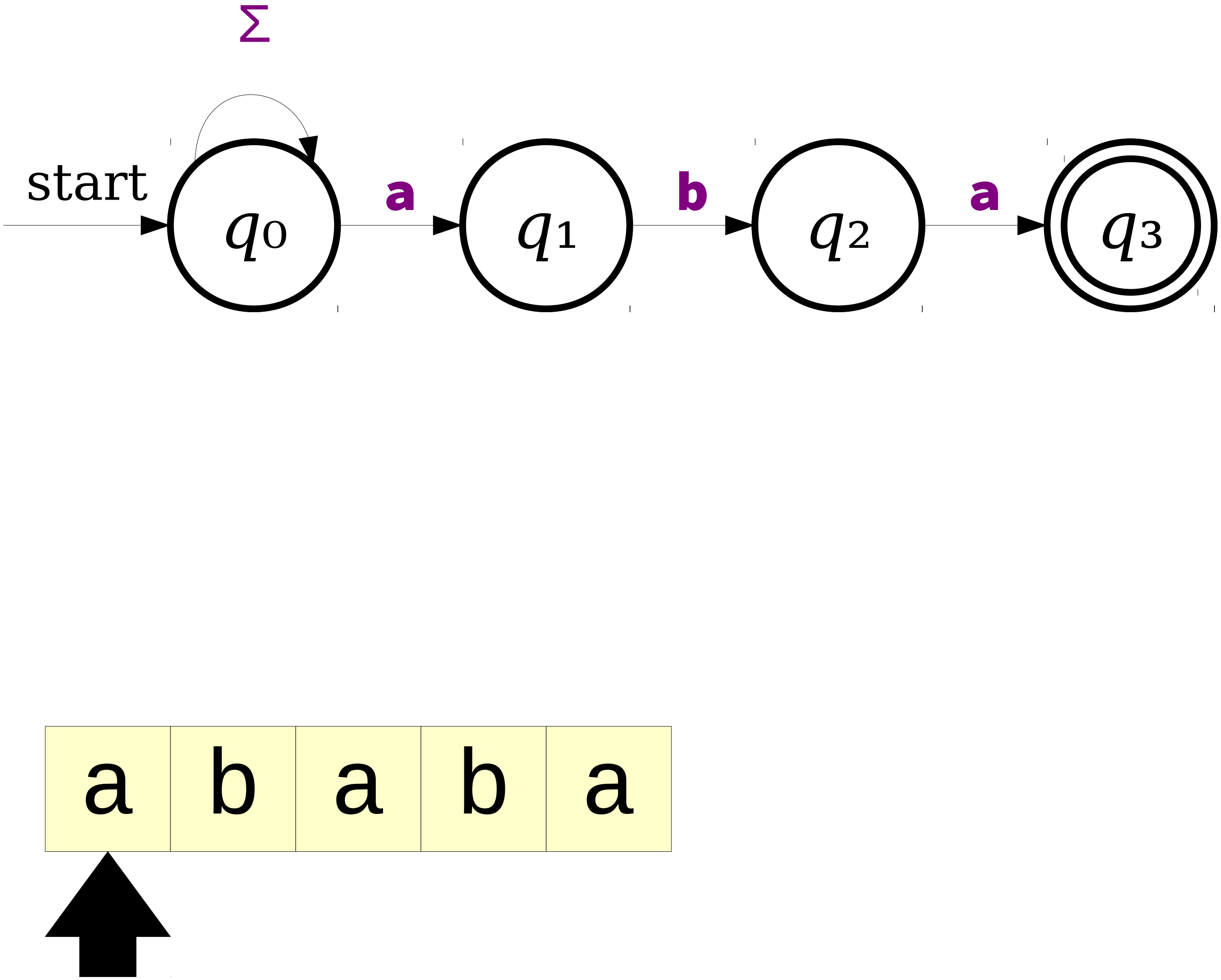



| | | | | |
|---|---|---|---|---|
| a | b | a | b | a |

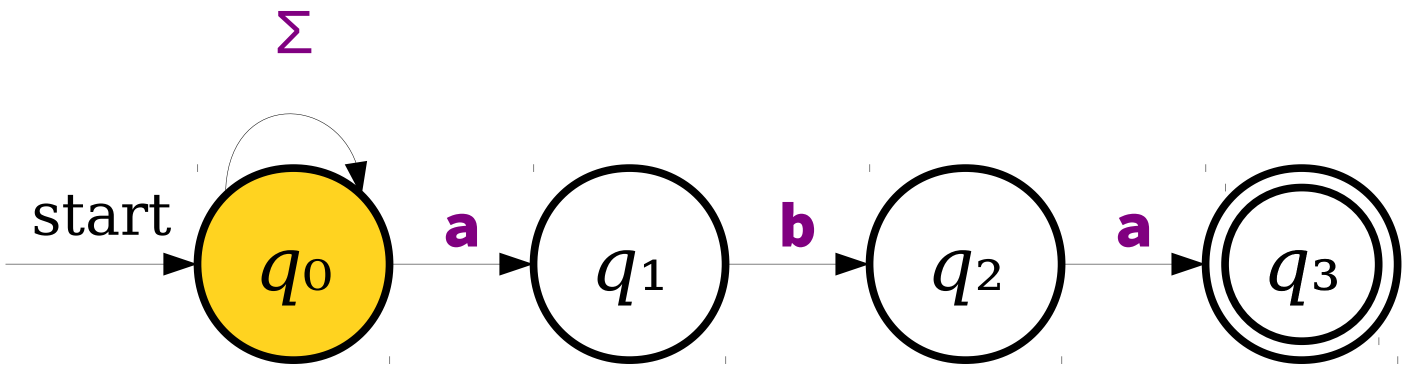
# Perfect Guessing

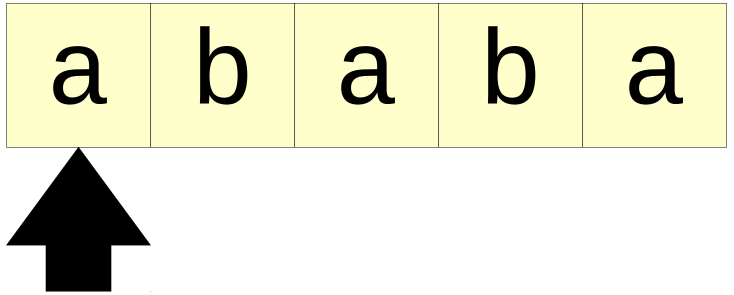
- We can view nondeterministic machines as having ***Magic Superpowers*** that enable them to guess choices that lead to an accepting state.
  - If there is at least one choice that leads to an accepting state, the machine will guess it.
  - If there are no choices, the machine guesses any one of the wrong guesses.
- No known physical analog for this style of computation – this is totally new!

# Massive Parallelism

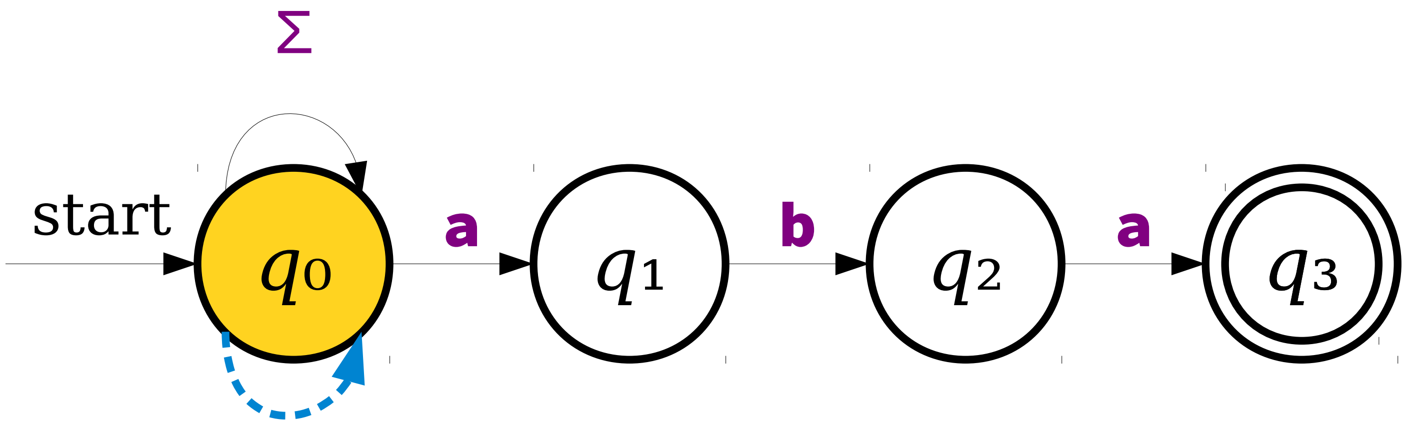


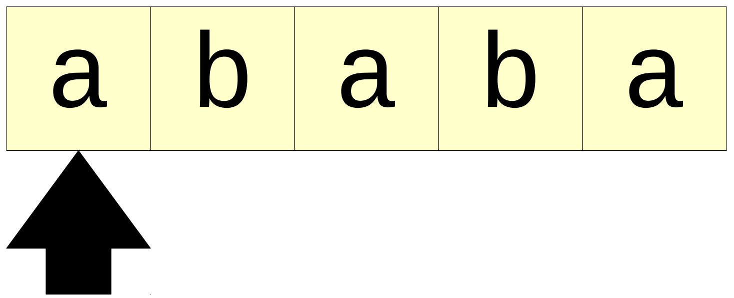
# Massive Parallelism



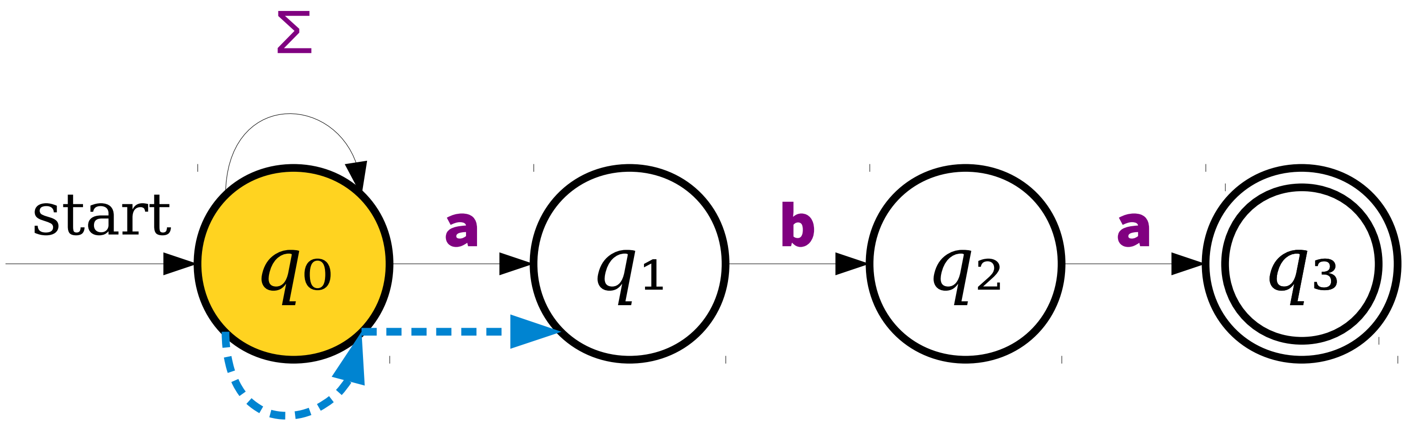



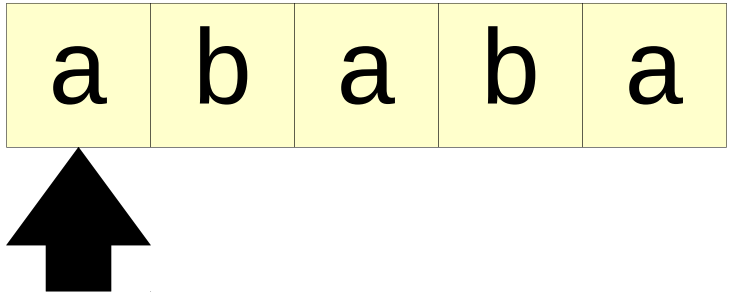
# Massive Parallelism



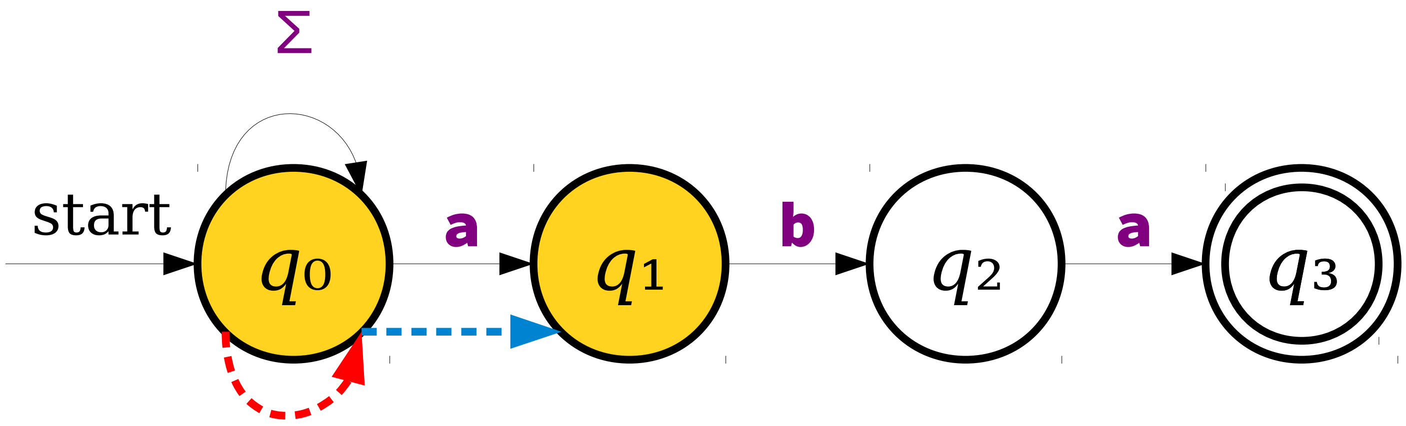



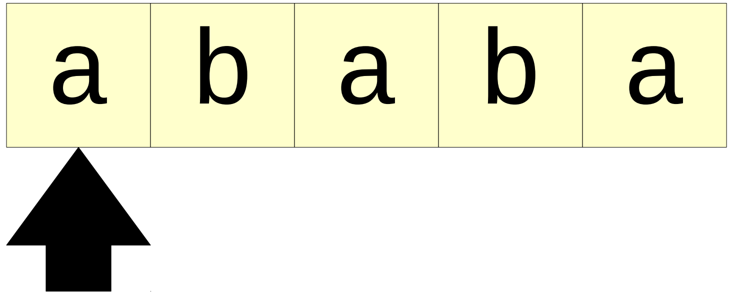
# Massive Parallelism



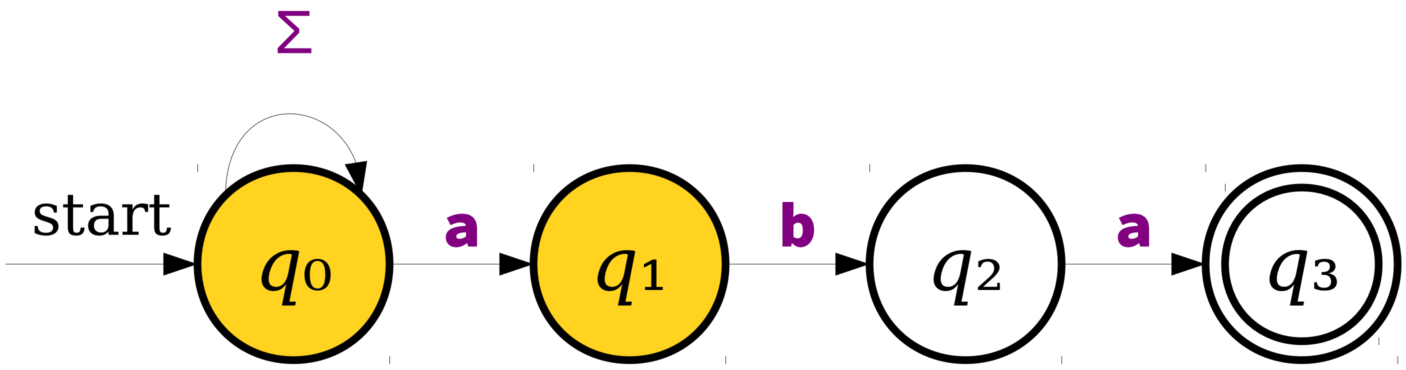



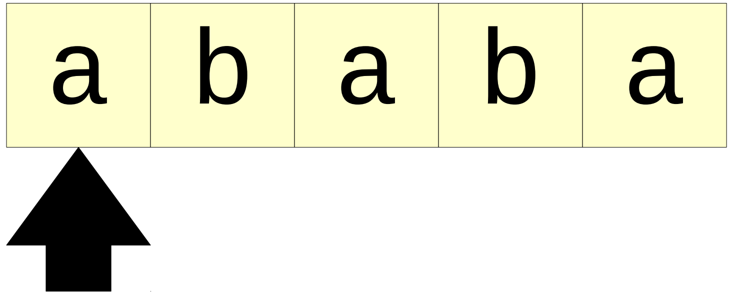
# Massive Parallelism



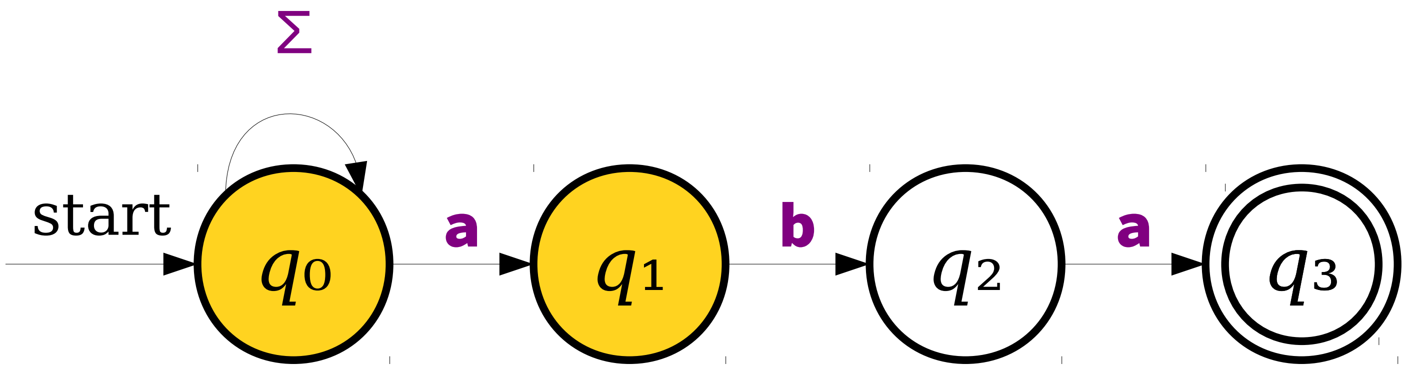



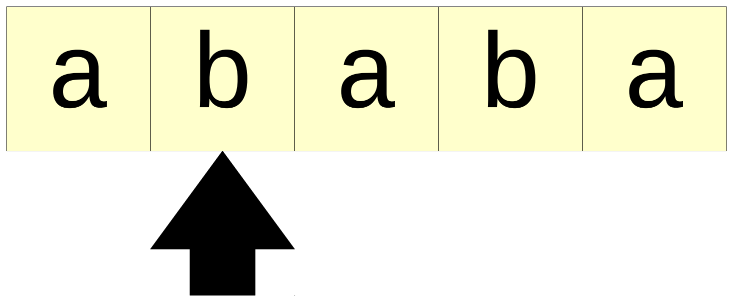
# Massive Parallelism



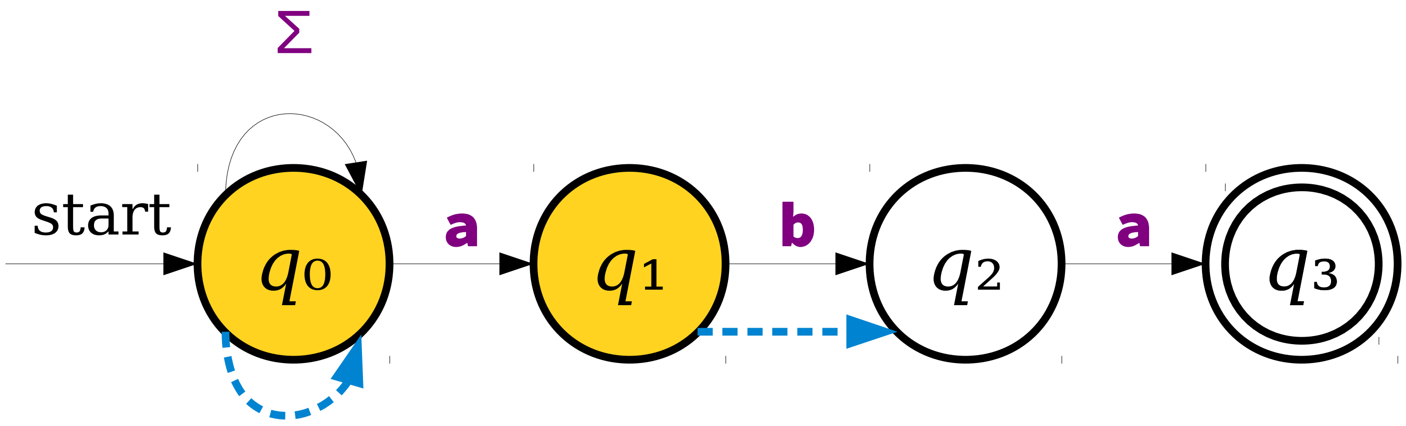



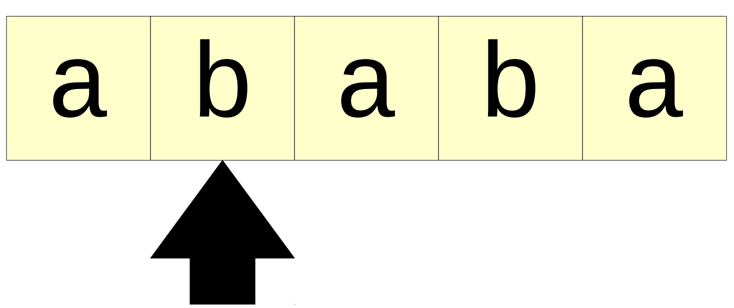
# Massive Parallelism



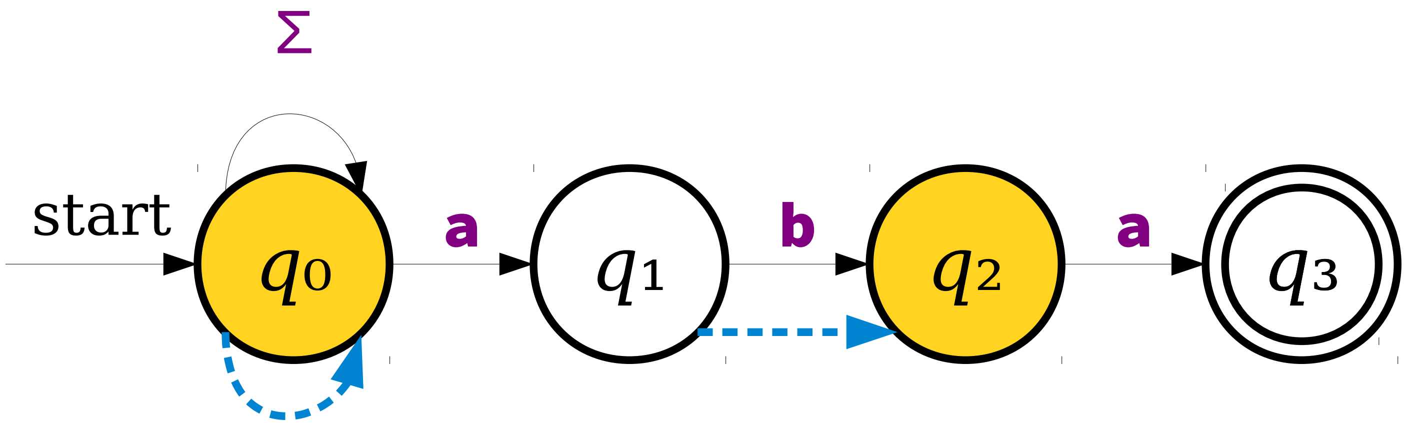



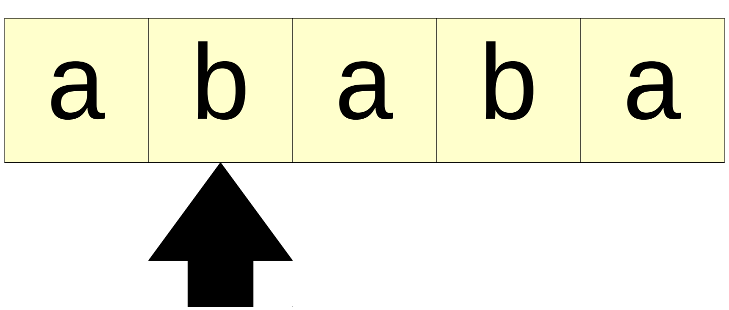
# Massive Parallelism



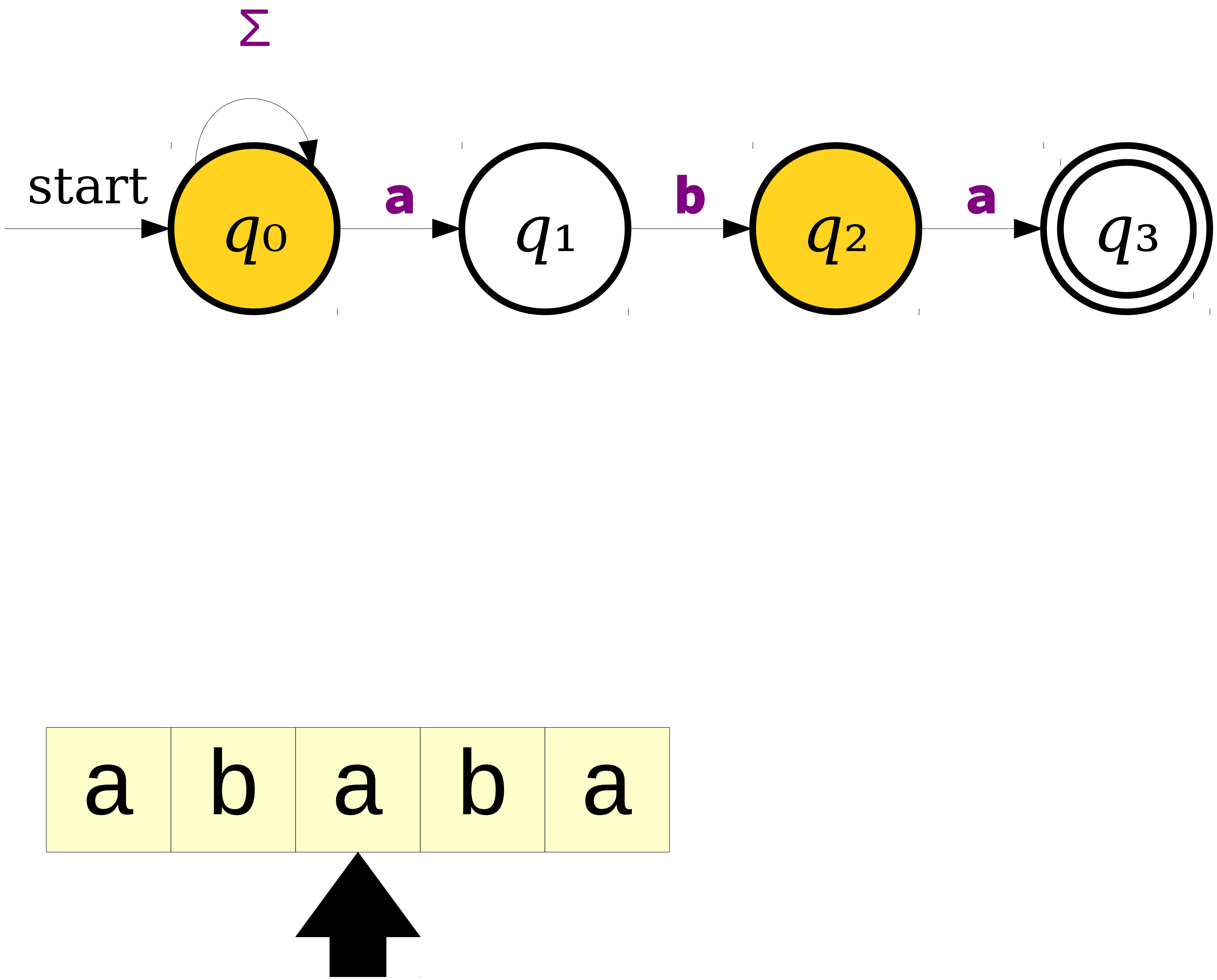



# Massive Parallelism

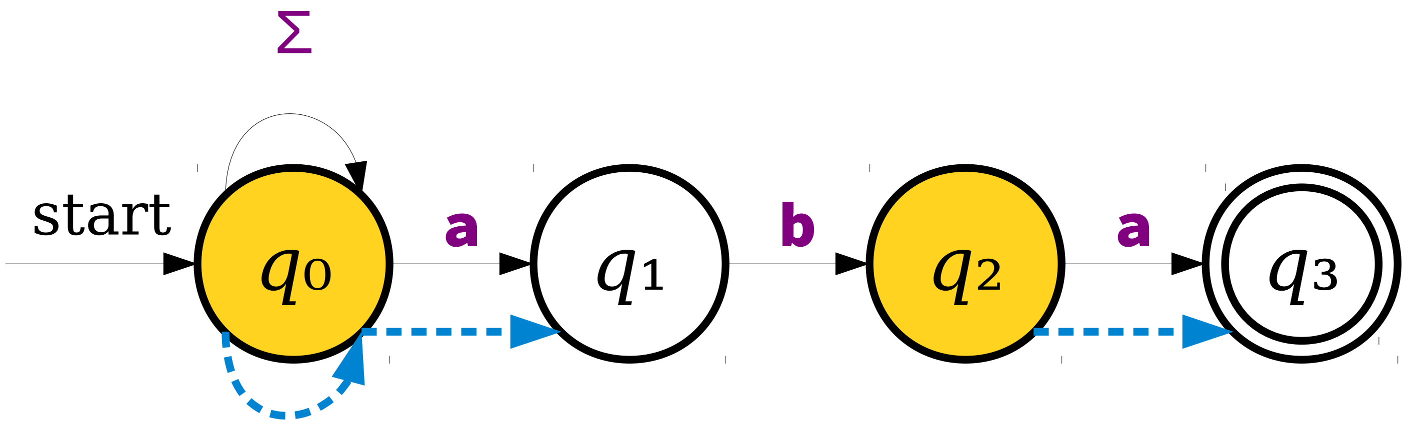


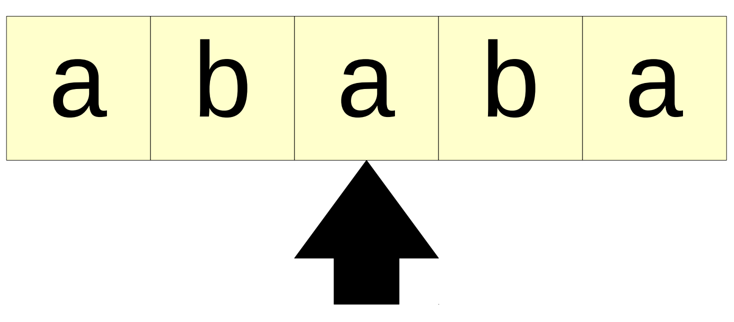



# Massive Parallelism

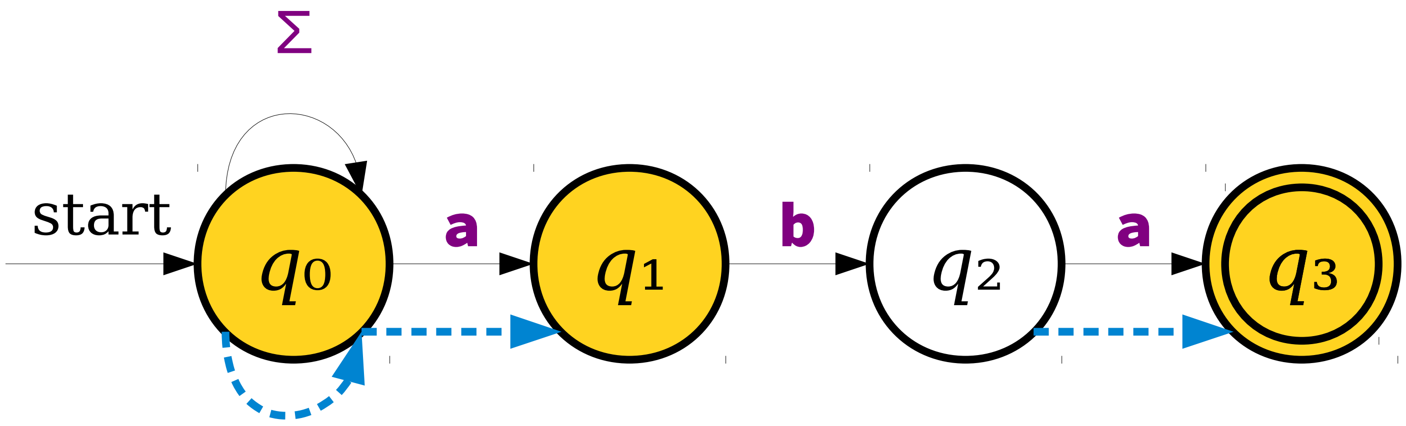


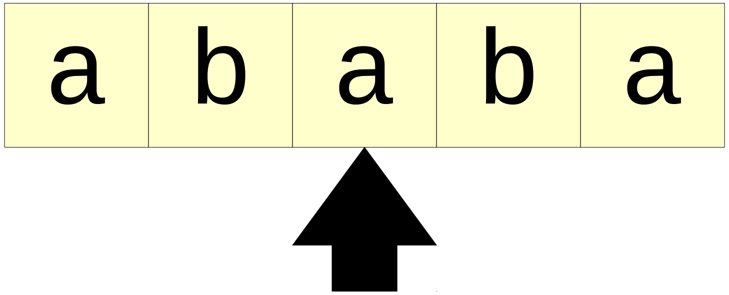
# Massive Parallelism



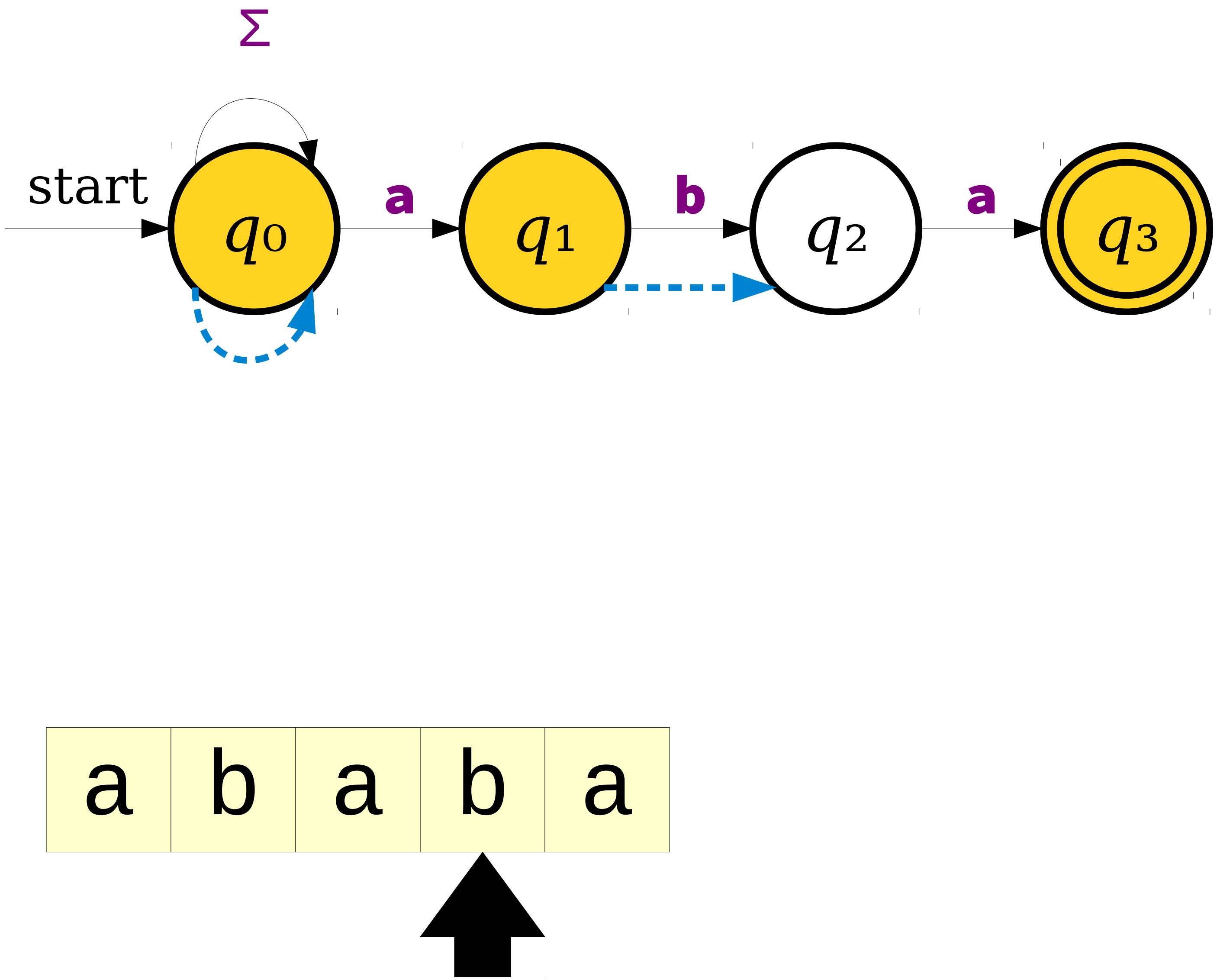



# Massive Parallelism

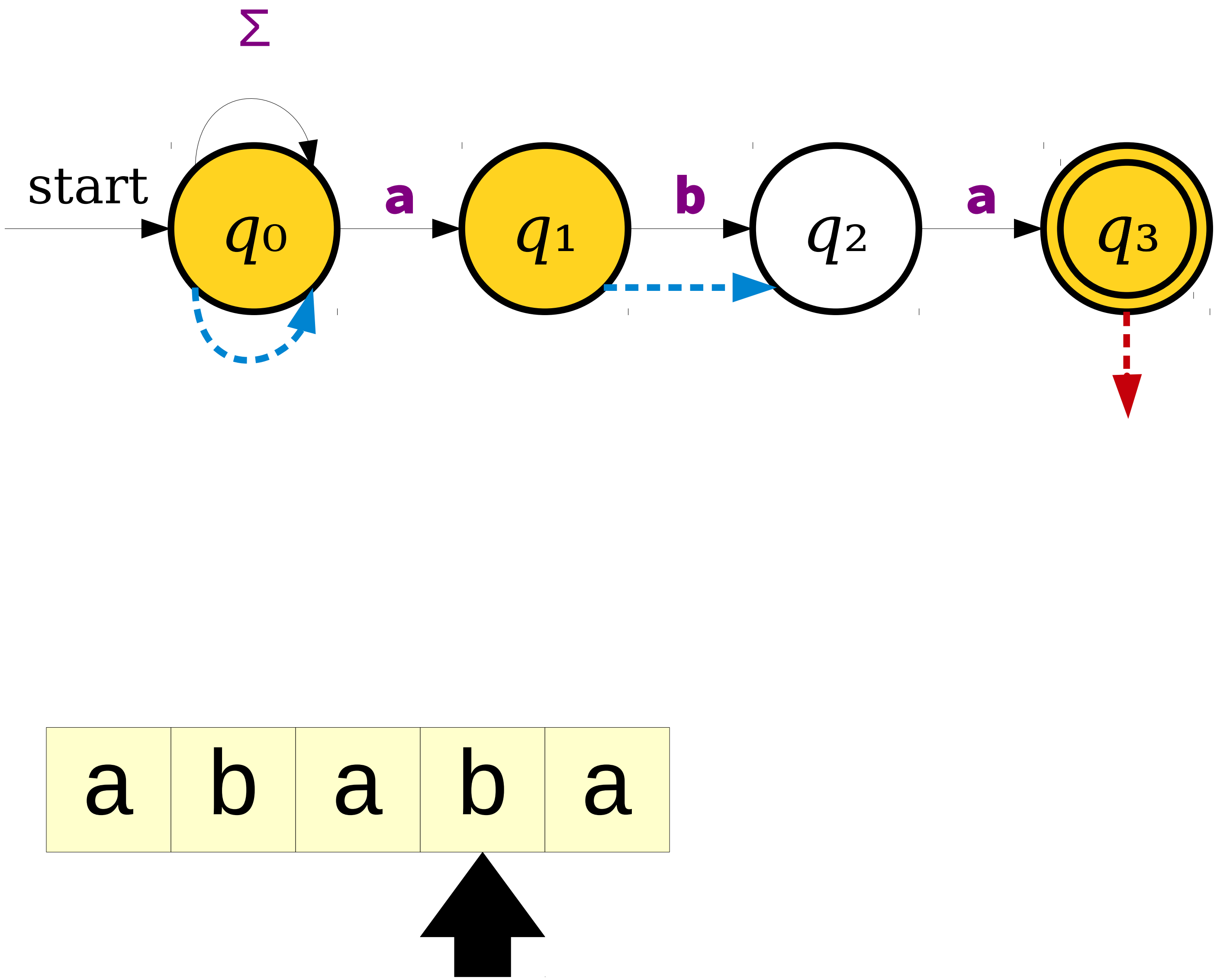


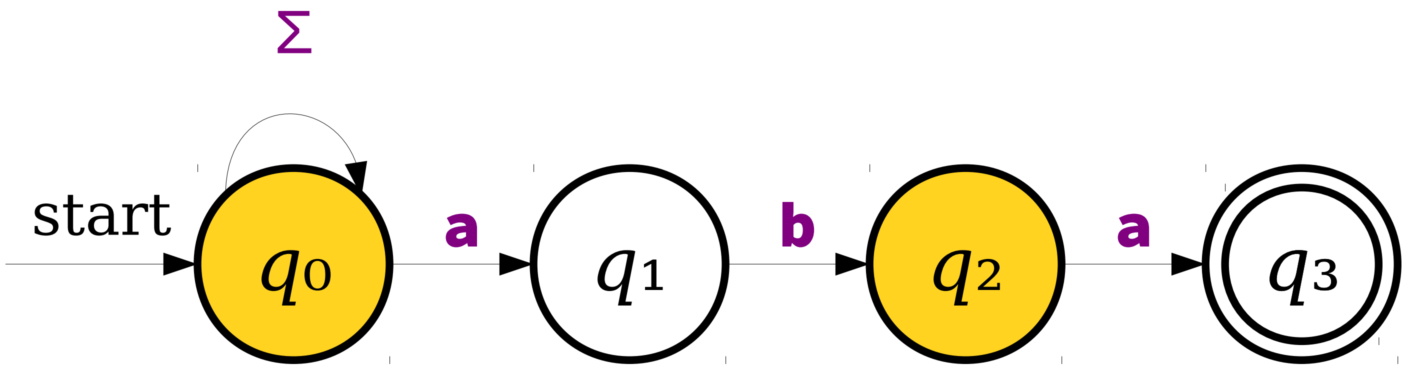
# Massive Parallelism

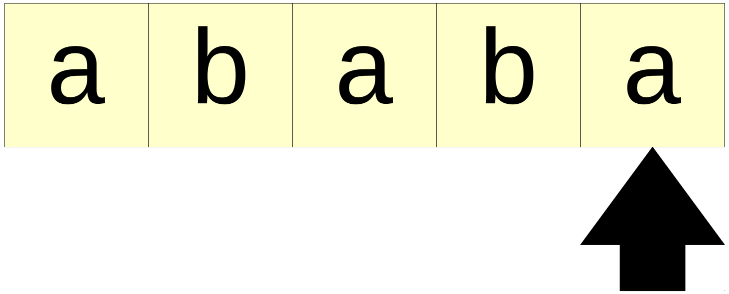


# Massive Parallelism

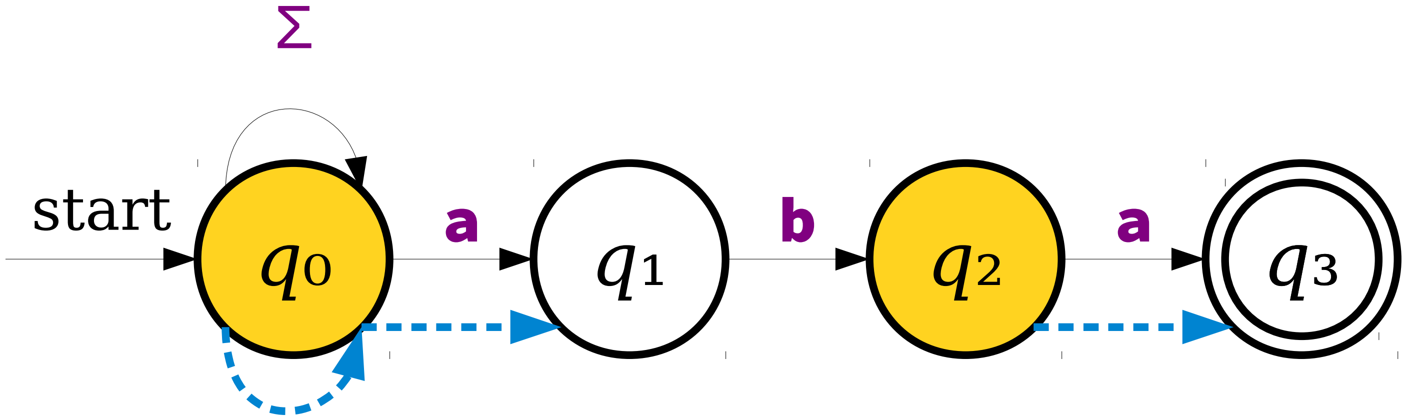


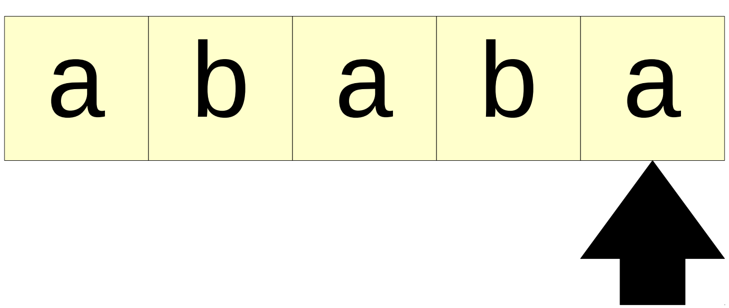
# Massive Parallelism



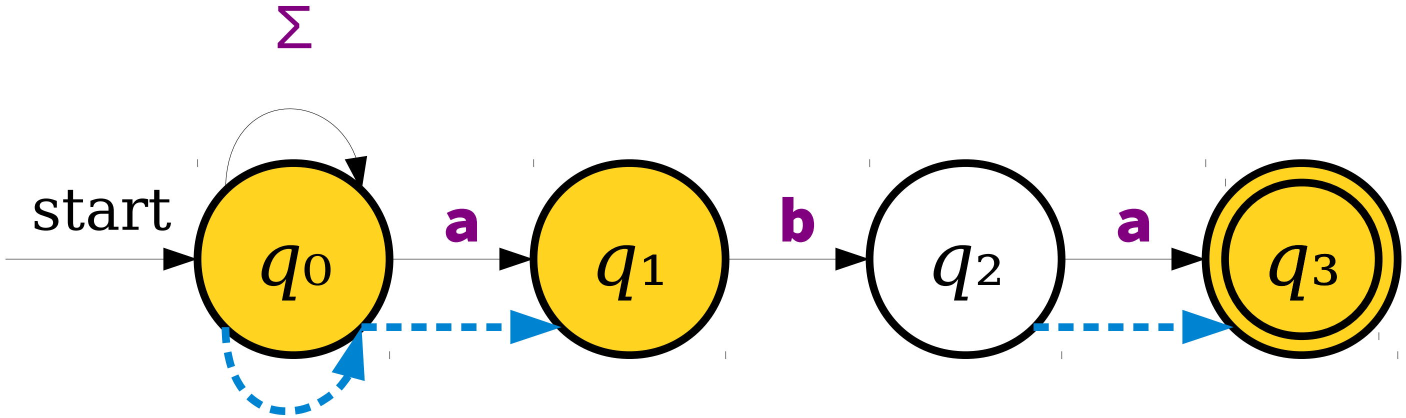



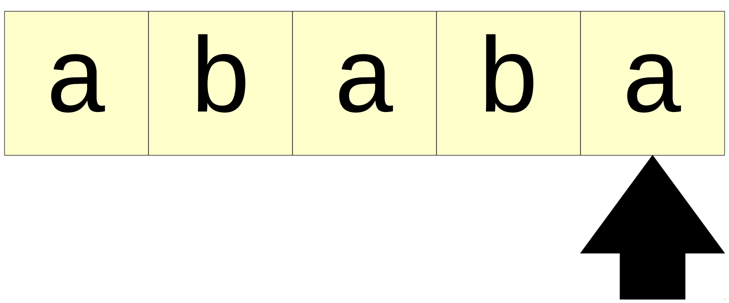
# Massive Parallelism



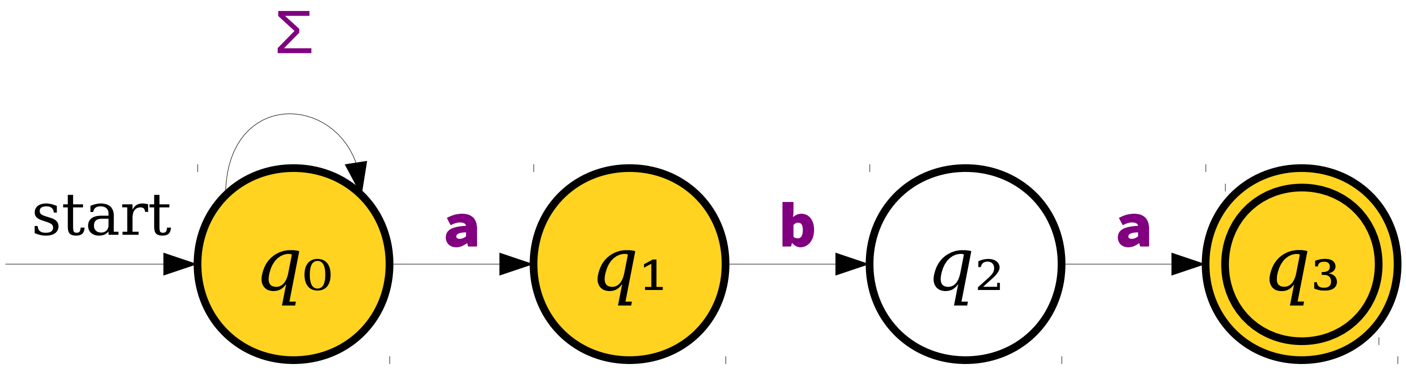



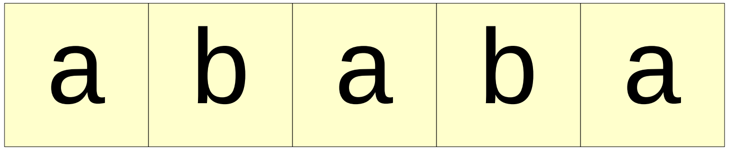
# Massive Parallelism



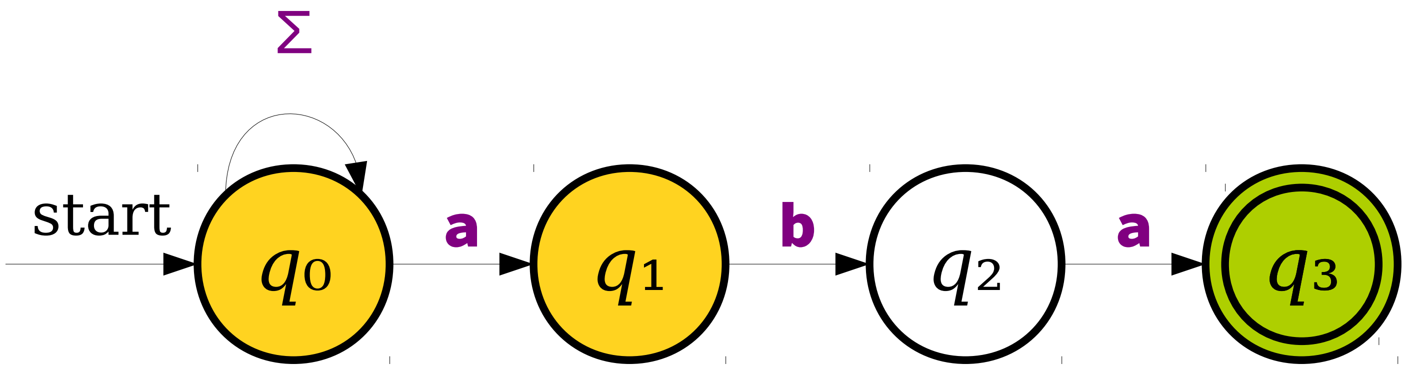



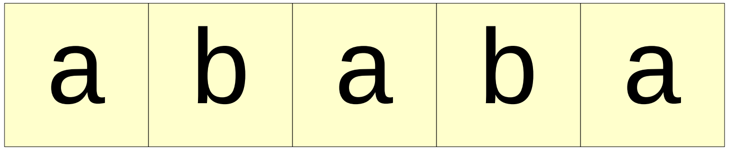
# Massive Parallelism



# Massive Parallelism




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

Therefore, we accept!

# 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.

# So What?

- Each intuition of nondeterminism is useful in a different setting:
  - Perfect guessing is a great way to think about how to design a machine.
  - Massive parallelism is a great way to test machines – and has nice theoretical implications.
- Nondeterministic machines may not be feasible, but they give a great basis for interesting questions:
  - Can any problem that can be solved by a nondeterministic machine be solved by a deterministic machine?
  - Can any problem that can be solved by a nondeterministic machine be solved ***efficiently*** by a deterministic machine?
- The answers vary from automaton to automaton.

# Designing NFAs

# Designing NFAs

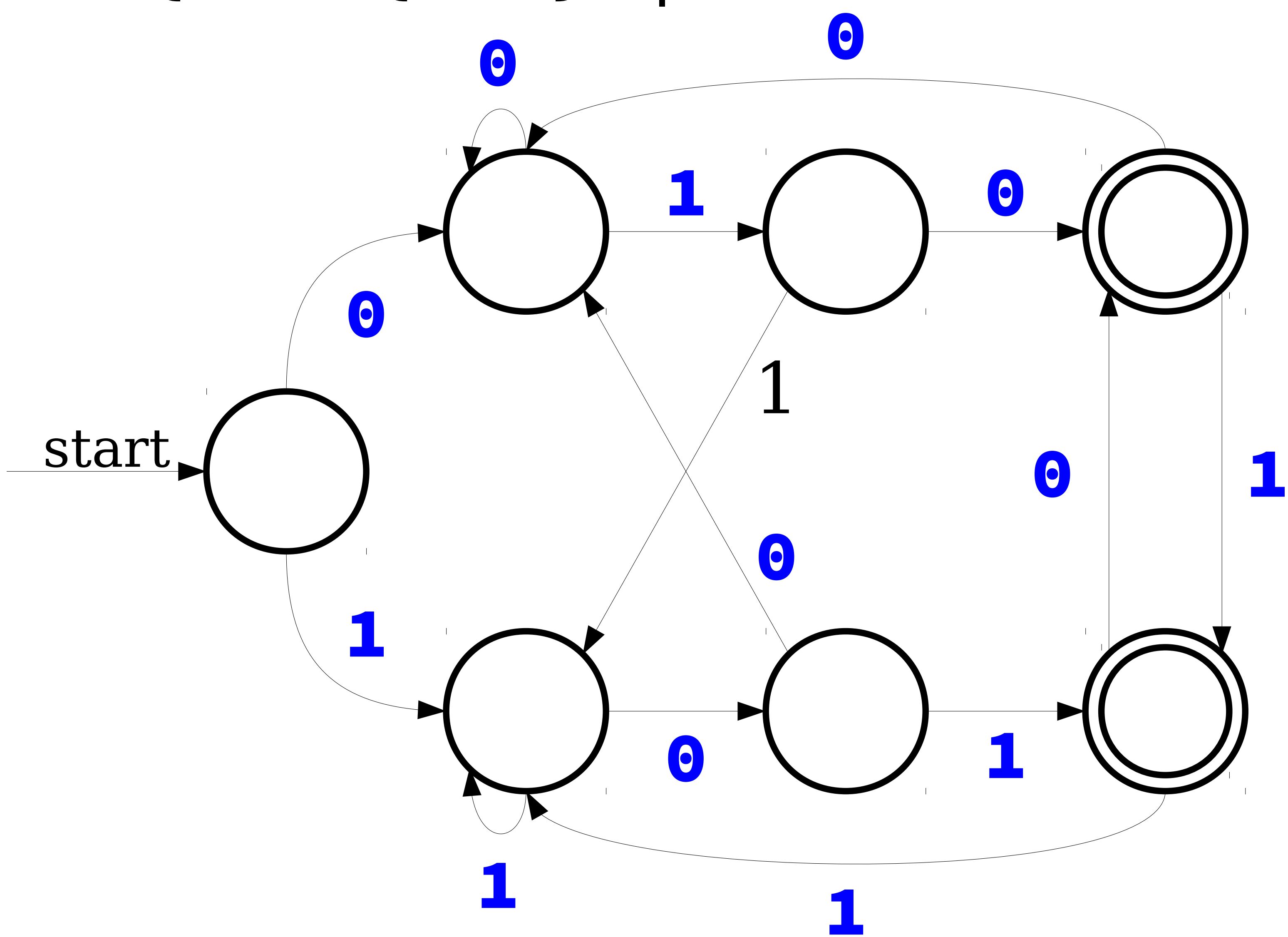
- When designing NFAs, *embrace the nondeterminism!*
- Good model: ***Guess-and-check***:
  - Is there some information that you'd really like to have? Have the machine *nondeterministically guess* that information.
  - Then, have the machine *deterministically check* that the choice was correct.
- The *guess* phase corresponds to trying lots of different options.
- The *check* phase corresponds to filtering out bad guesses or wrong options.

# Guess-and-Check

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

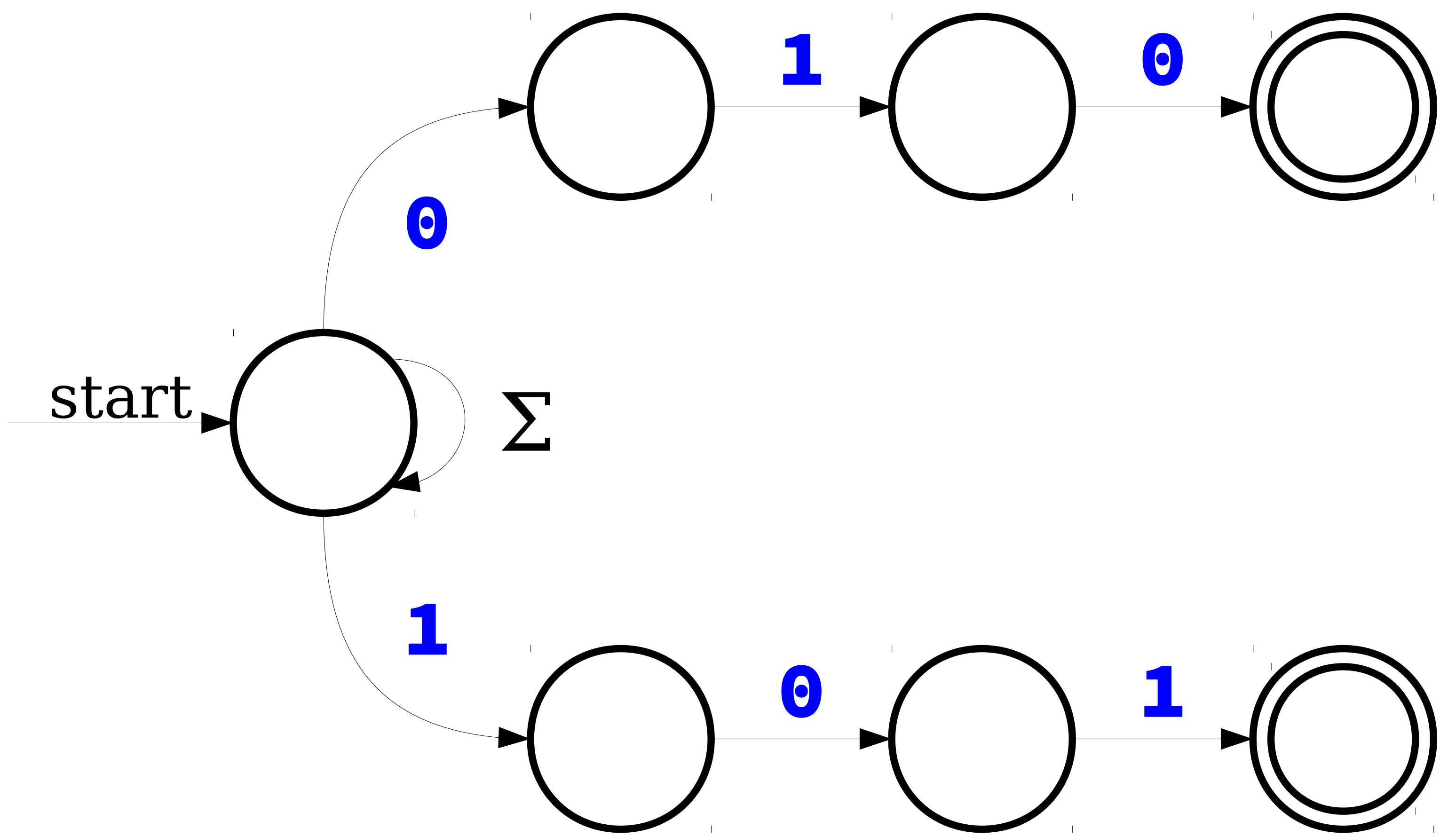
# Guess-and-Check

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



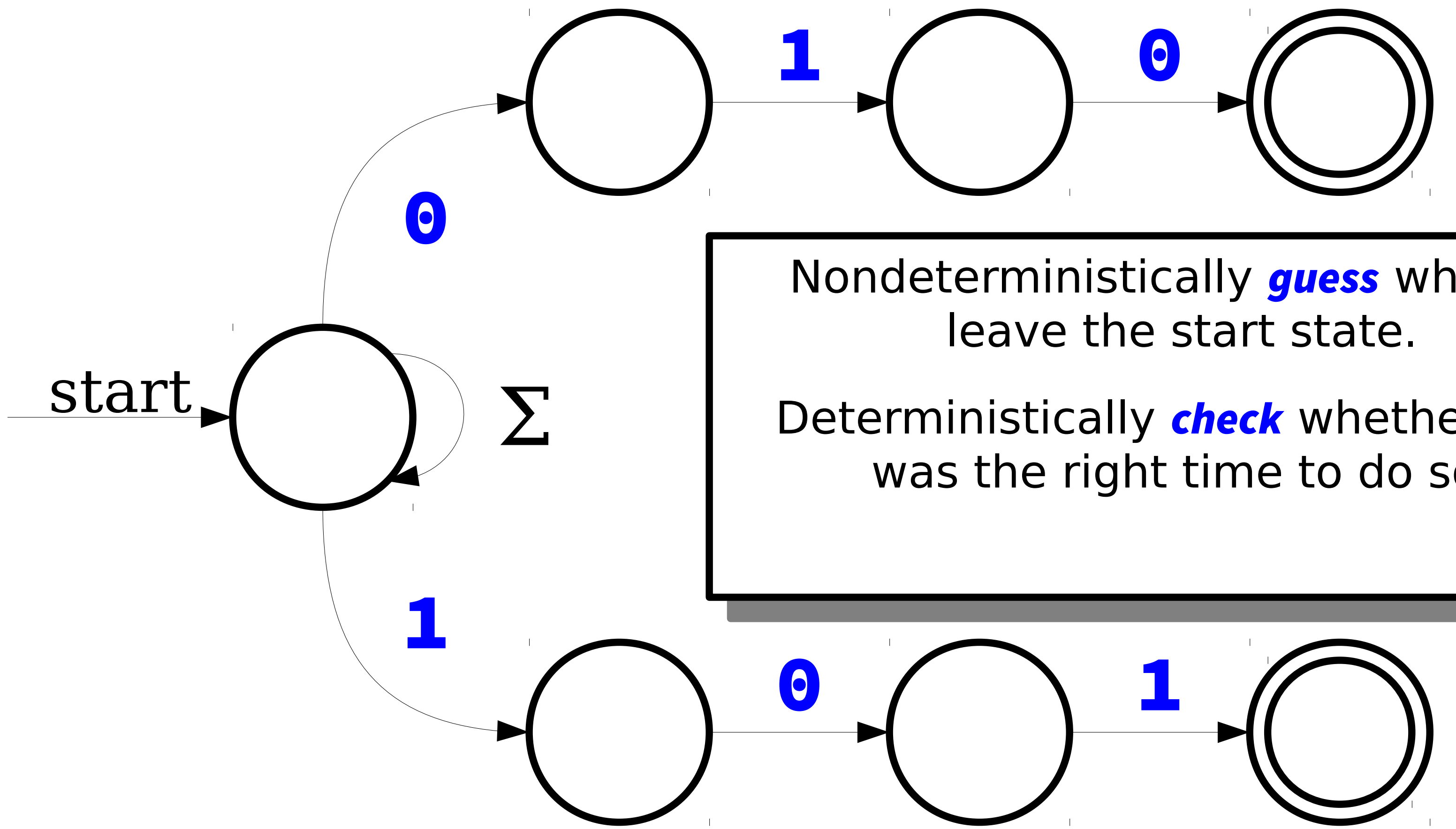
# Guess-and-Check

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



# Guess-and-Check

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




Nondeterministically ***guess*** when to leave the start state.

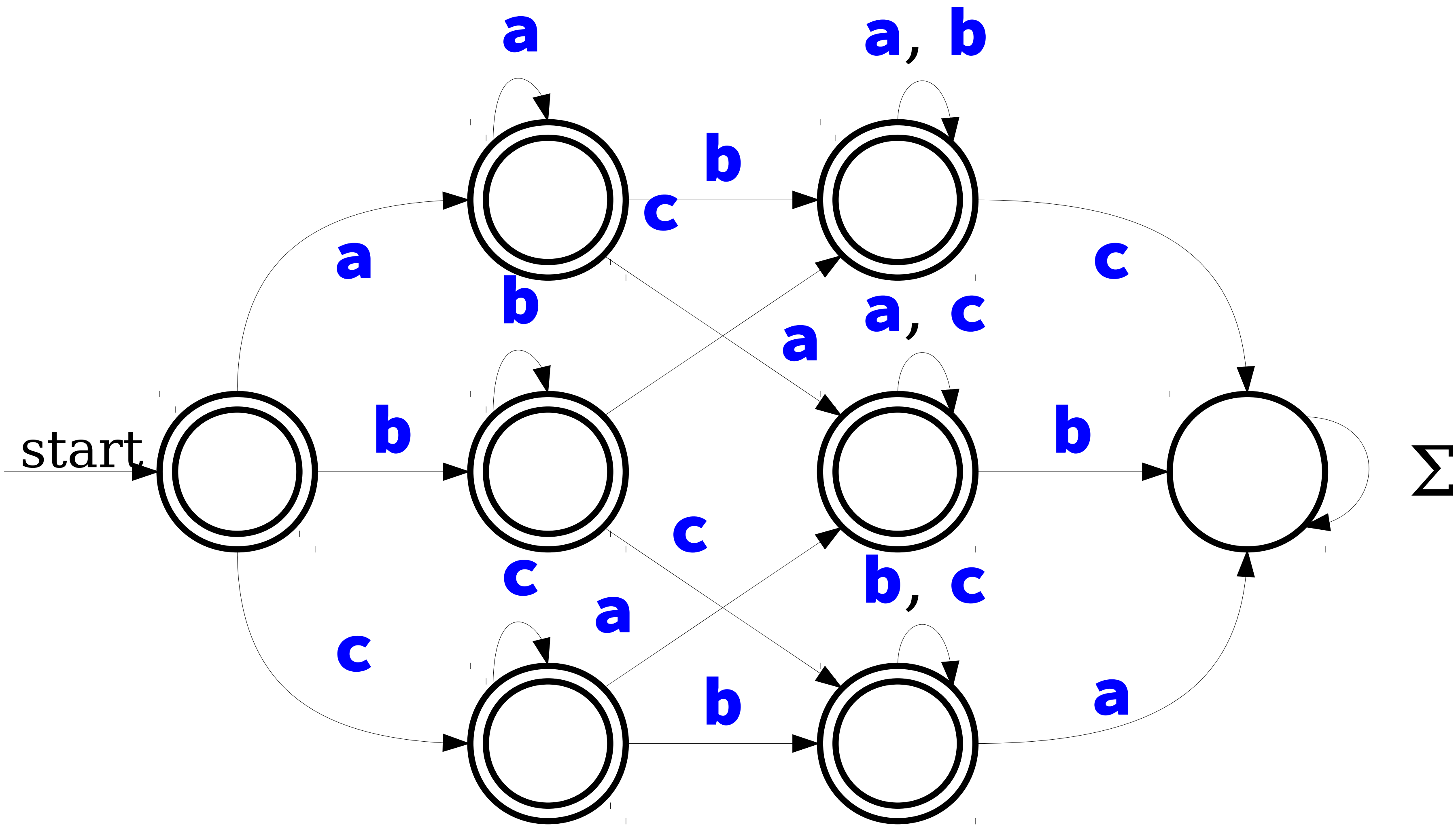
Deterministically ***check*** whether that was the right time to do so.

# Guess-and-Check

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

# Guess-and-Check

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

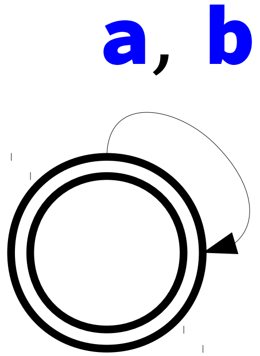


# Guess-and-Check

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

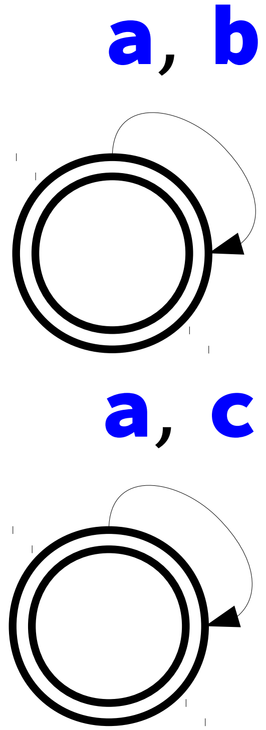
# Guess-and-Check

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



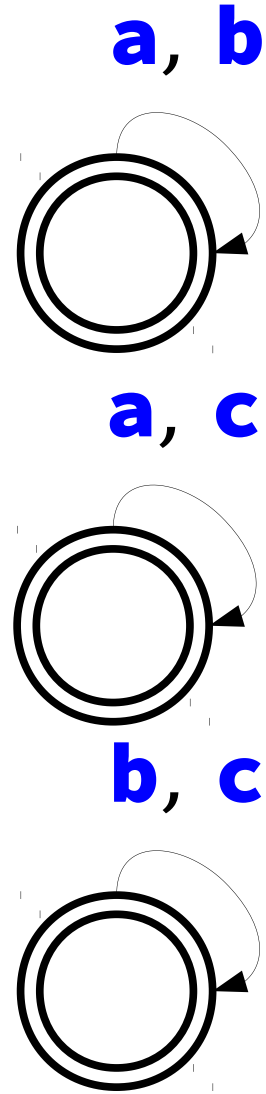
# Guess-and-Check

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



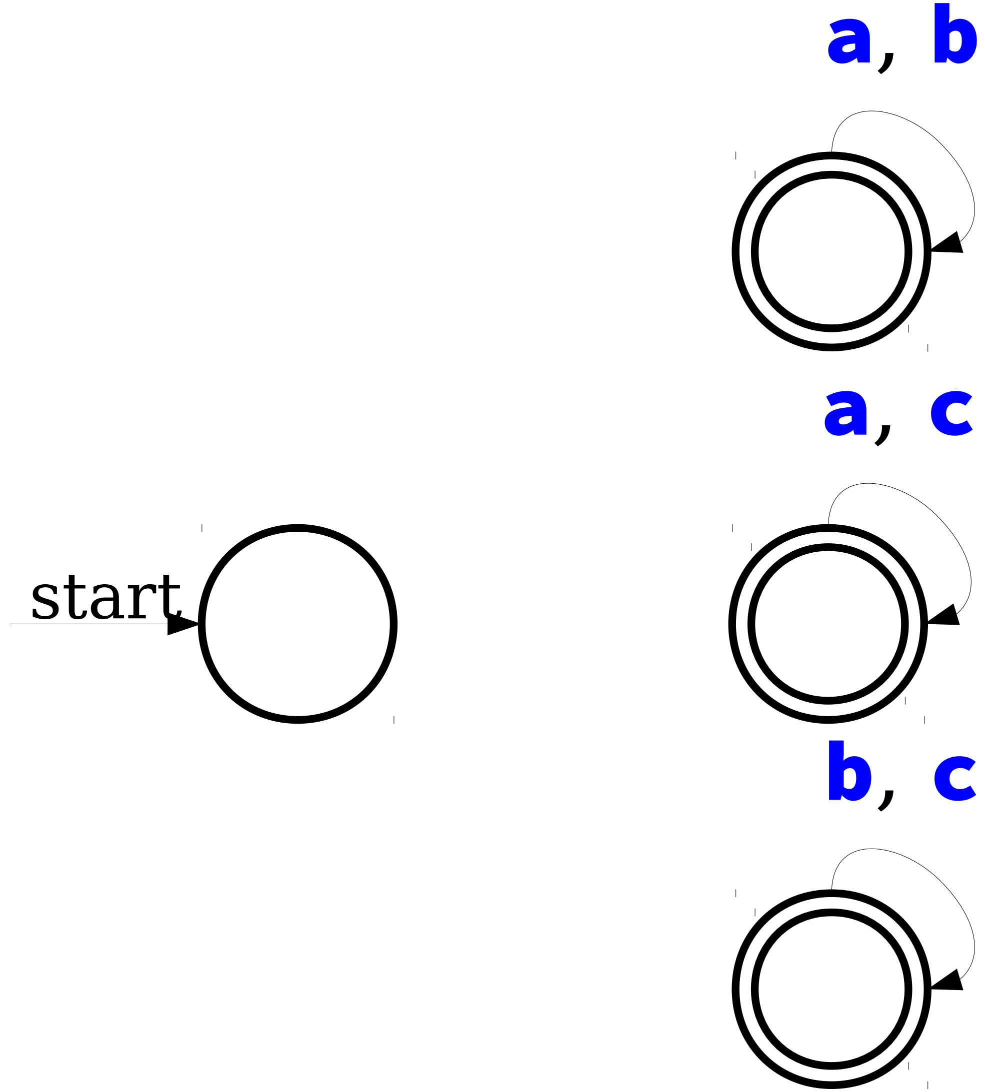
# Guess-and-Check

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



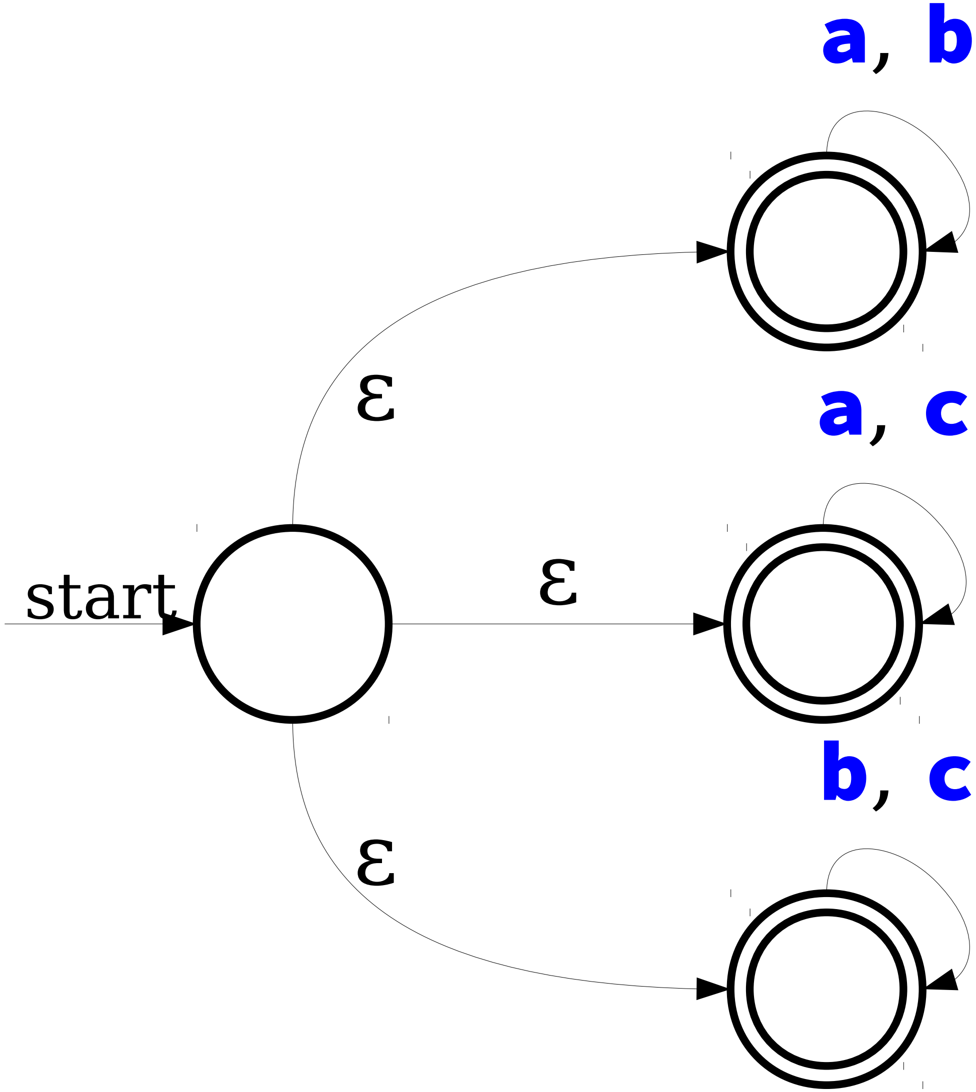
# Guess-and-Check

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



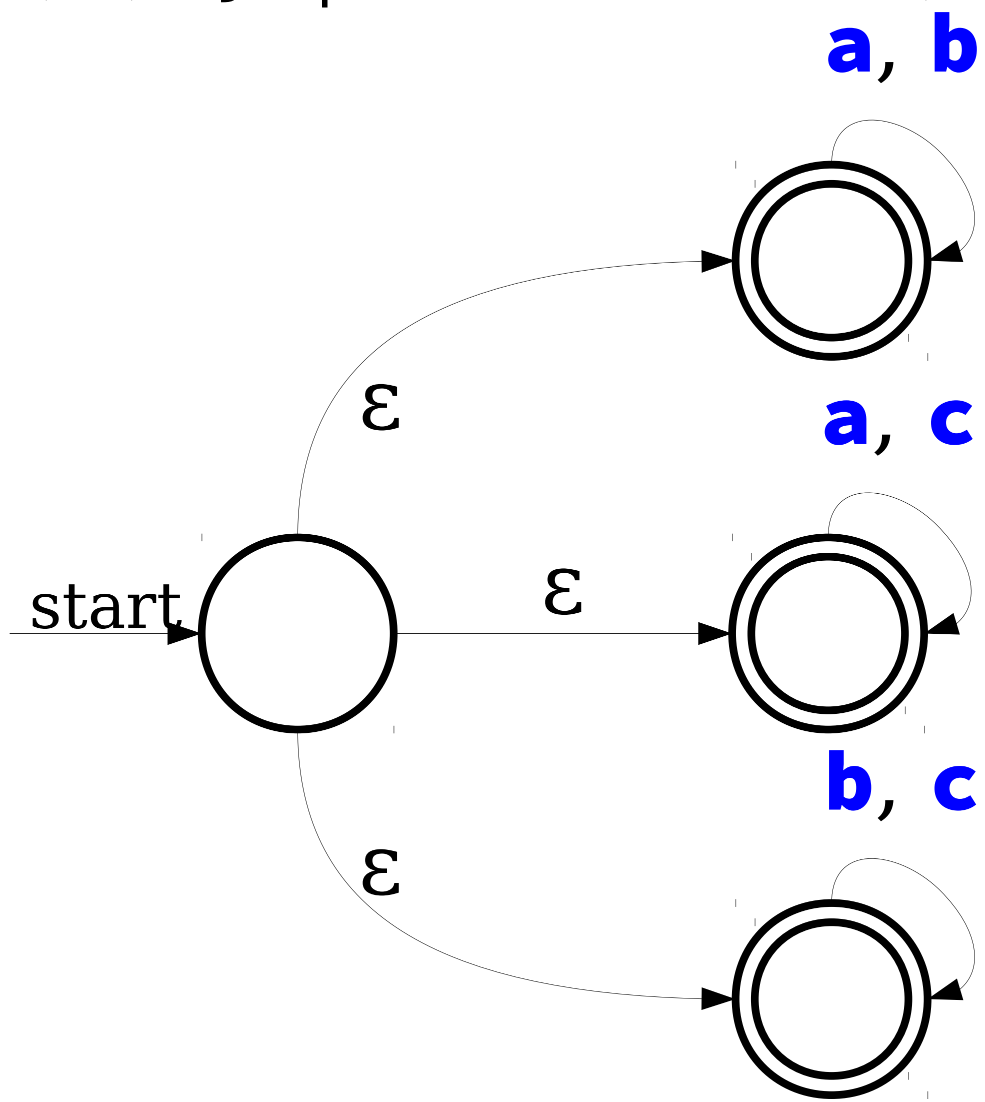
# Guess-and-Check

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



# Guess-and-Check

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




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

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

# Just how powerful are NFAs?