Finite Automata

Part Two

Outline for Today

- ***Recap from Last Time***
  - Where are we, again?
- ***Designing a DFA***
  - How to think about finite memory.
- ***Regular Languages***
  - A fundamental class of languages.
- ***NFAs***
  - Automata with Magic Superpowers.
- ***Designing NFAs***
  - Harnessing an awesome power.

Recap from Last Time

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 $\mathscr{L}(\boldsymbol{A})$, is the set of all strings A accepts.
- Formally:

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

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.

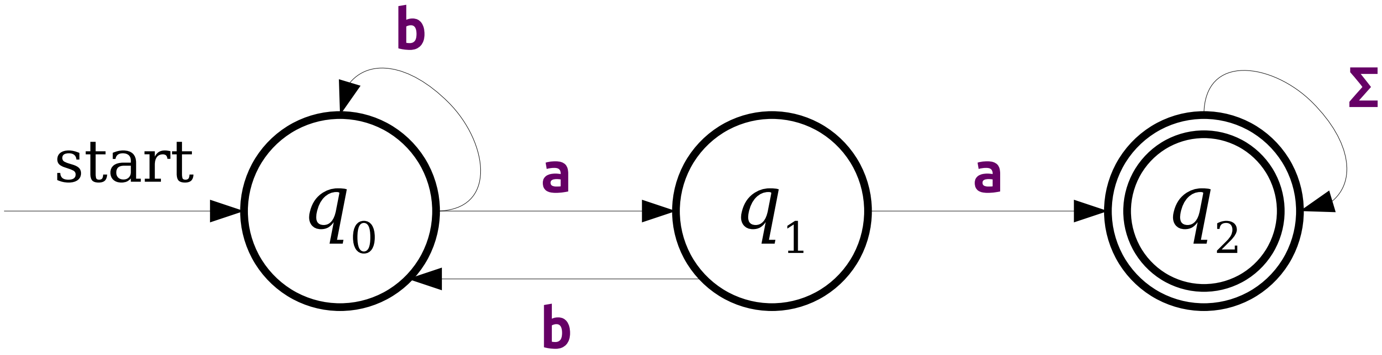
DFAs

- 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.

New Stuff!

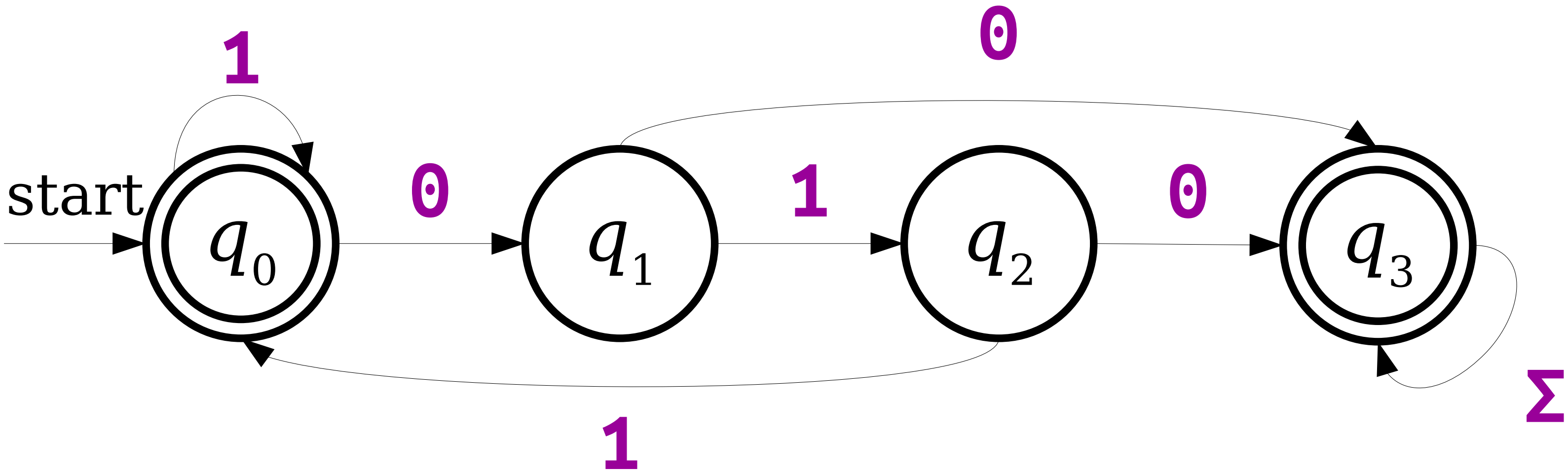
Recognizing Languages with DFAs

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



Tabular DFAs

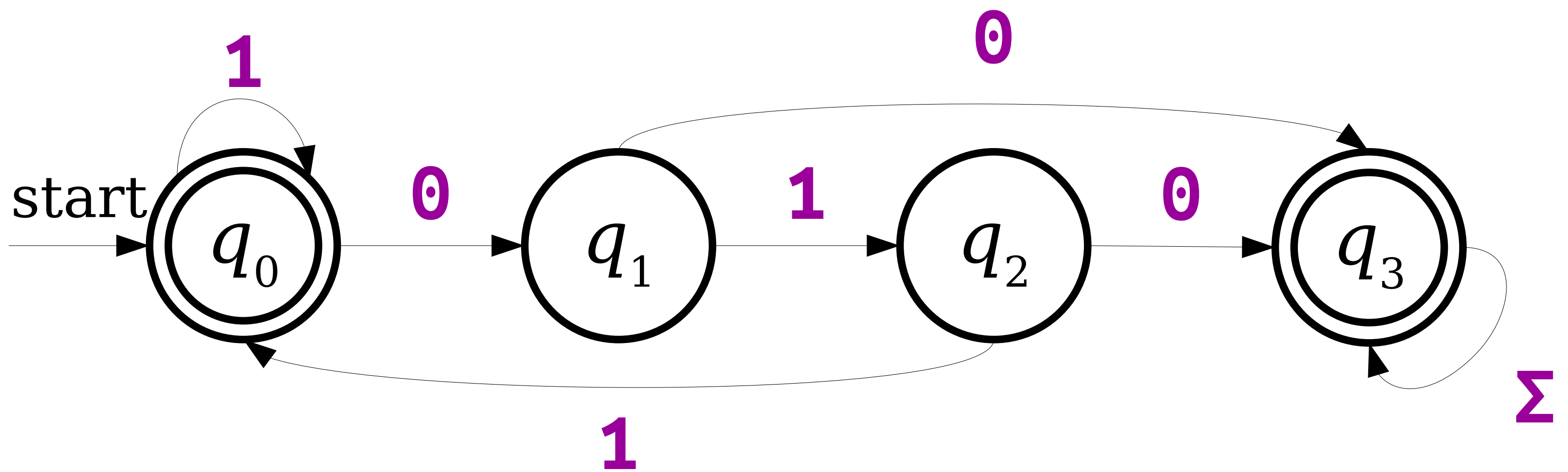
Tabular DFAs



	0	1
$*q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

These stars indicate accepting states.

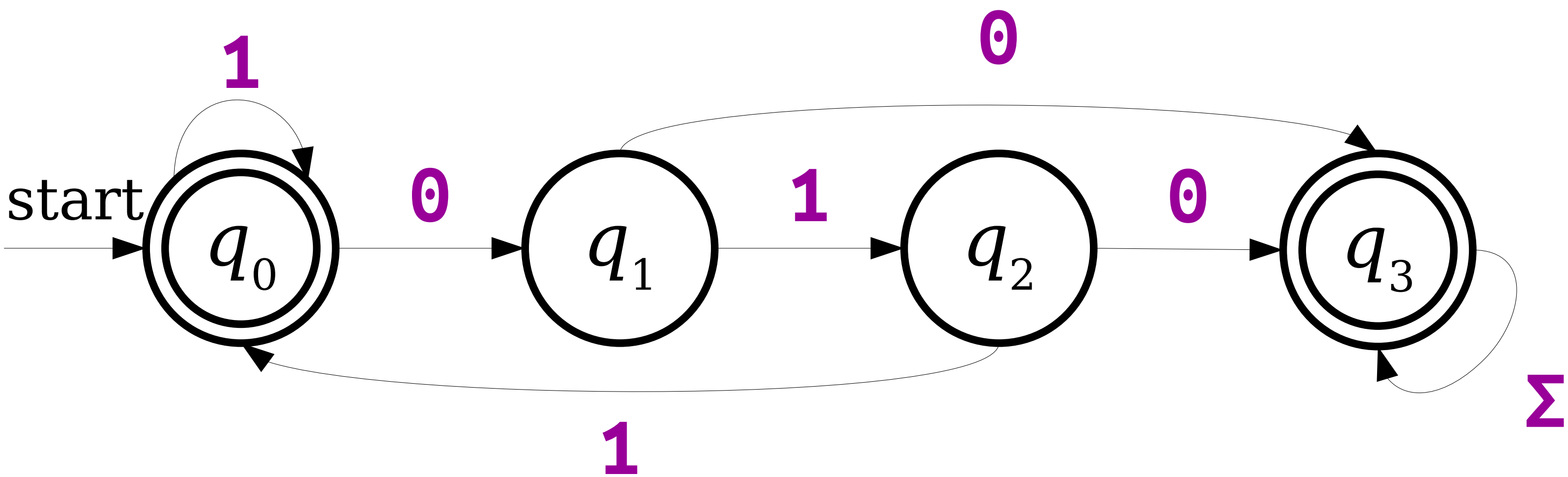
Tabular DFAs



	0	1
$*q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

Since this is the first row, it's the start state.

Tabular DFAs



	0	1
$*q_0$	q_1	q_0
q_1	q_3	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

Why isn't there a column here for Σ?

Answer at ***https://cs103.stanford.edu/pollev***

My Turn to Code Things Up!

```
int kTransitionTable[kNumStates][kNumSymbols] = {
     {0, 0, 1, 3, 7, 1, …},
      …
};
bool kAcceptTable[kNumStates] = {
    false,
    true,
    true,
    …
};
bool doesAccept(string input) {
    int state = 0;
    for (char ch: input) {
        state = kTransitionTable[state][ch];
    }
    return kAcceptTable[state];
}
```

The Regular Languages

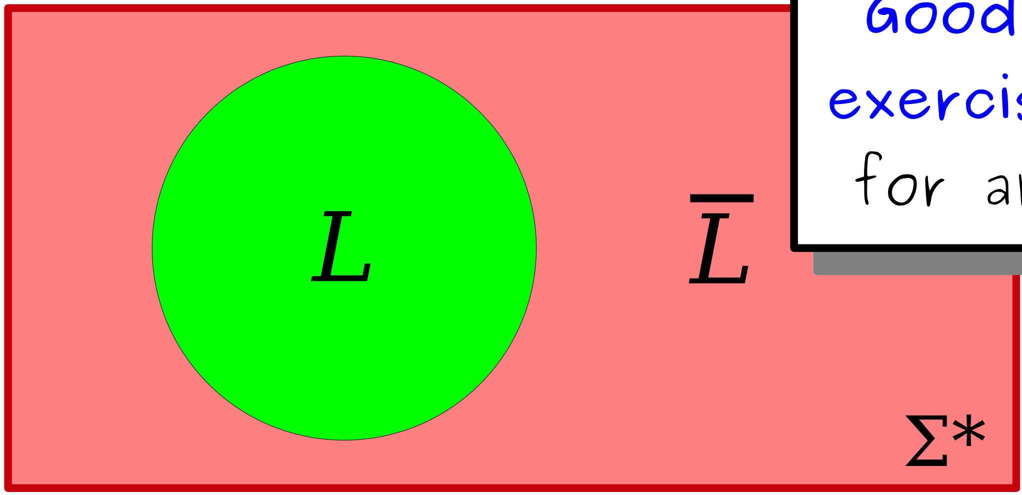
A language L is called a ***regular language*** if there exists a DFA D such that $\mathscr{L}(D) = L$.

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

The Complement of a Language

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

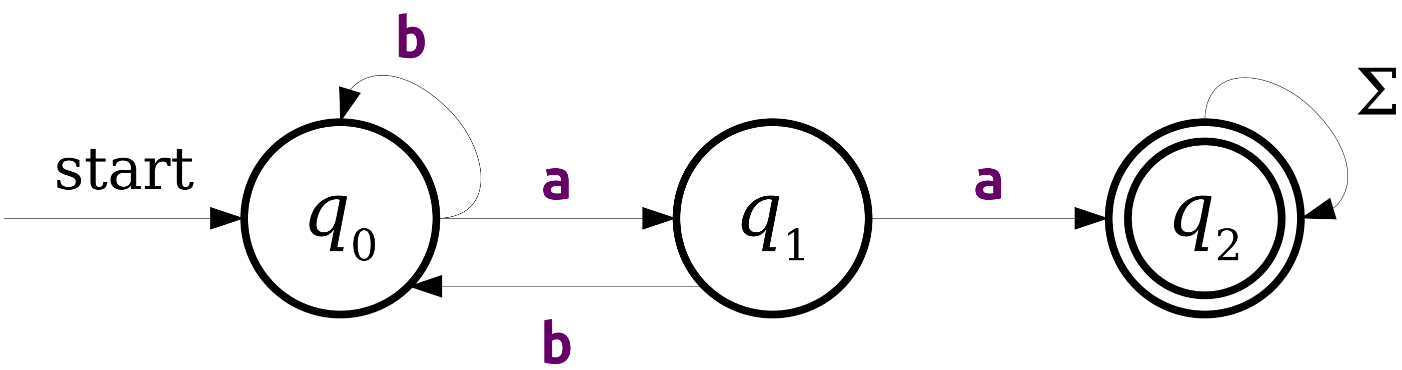
$$\overline{L} = \Sigma^* - L$$



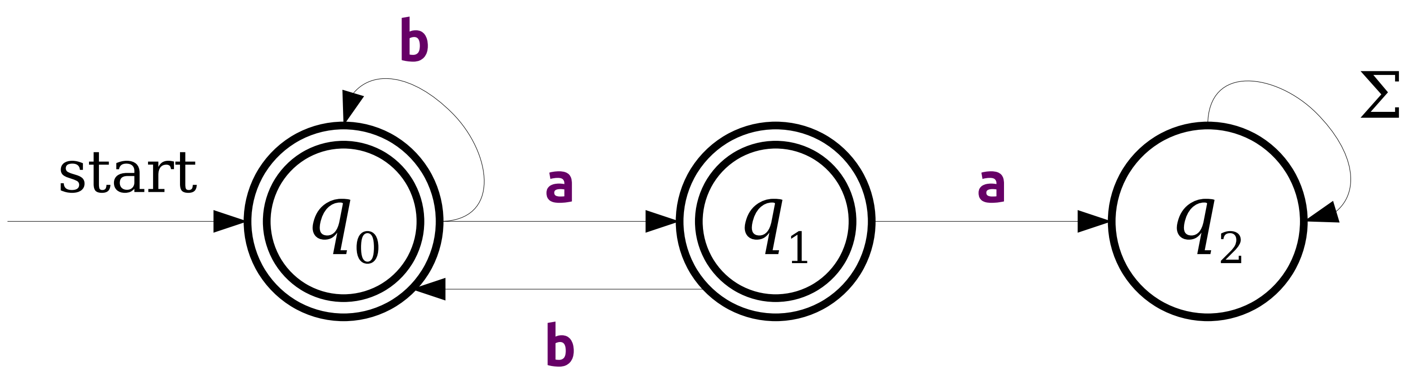
Good proofwriting exercise: prove $\overline{\overline{L}} = L$ for any language L.

Complementing Regular Languages

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

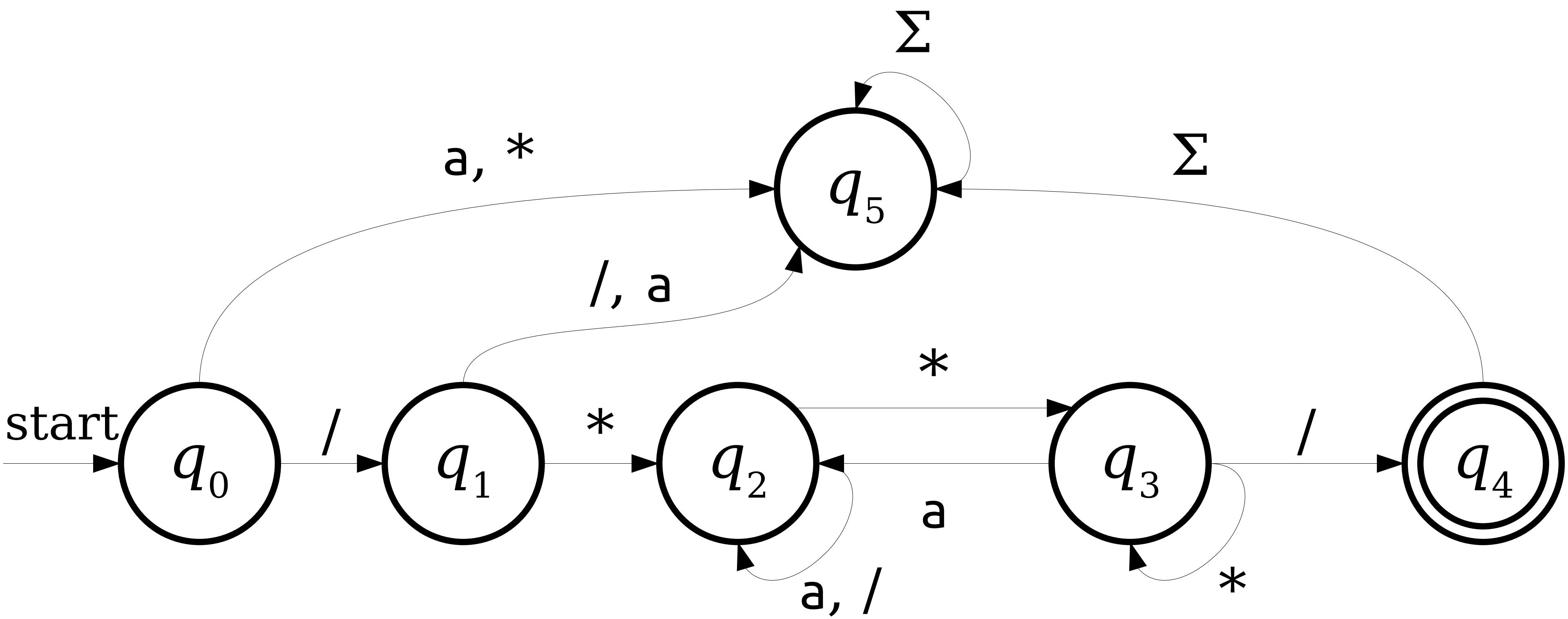


$\overline{L} = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \textbf{\textit{ does not }} \text{contain } \texttt{aa} \text{ as a substring } \}$



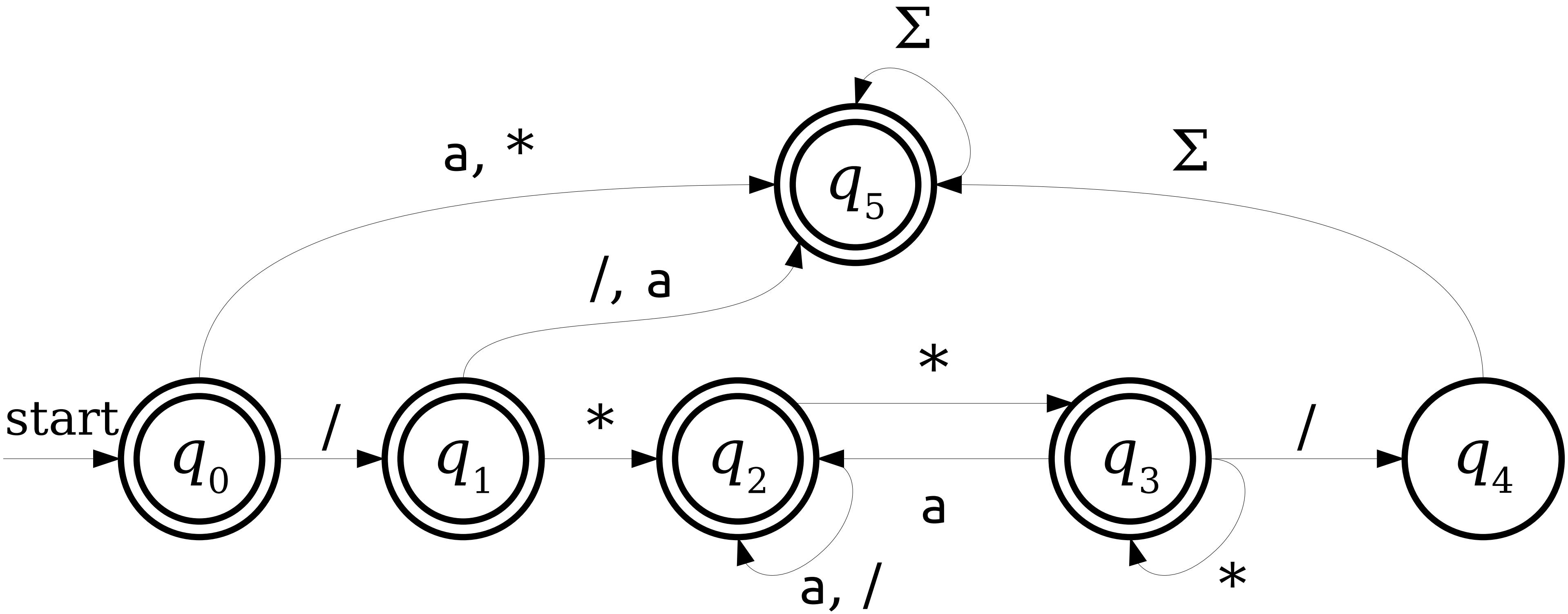
Complementing Regular Languages

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



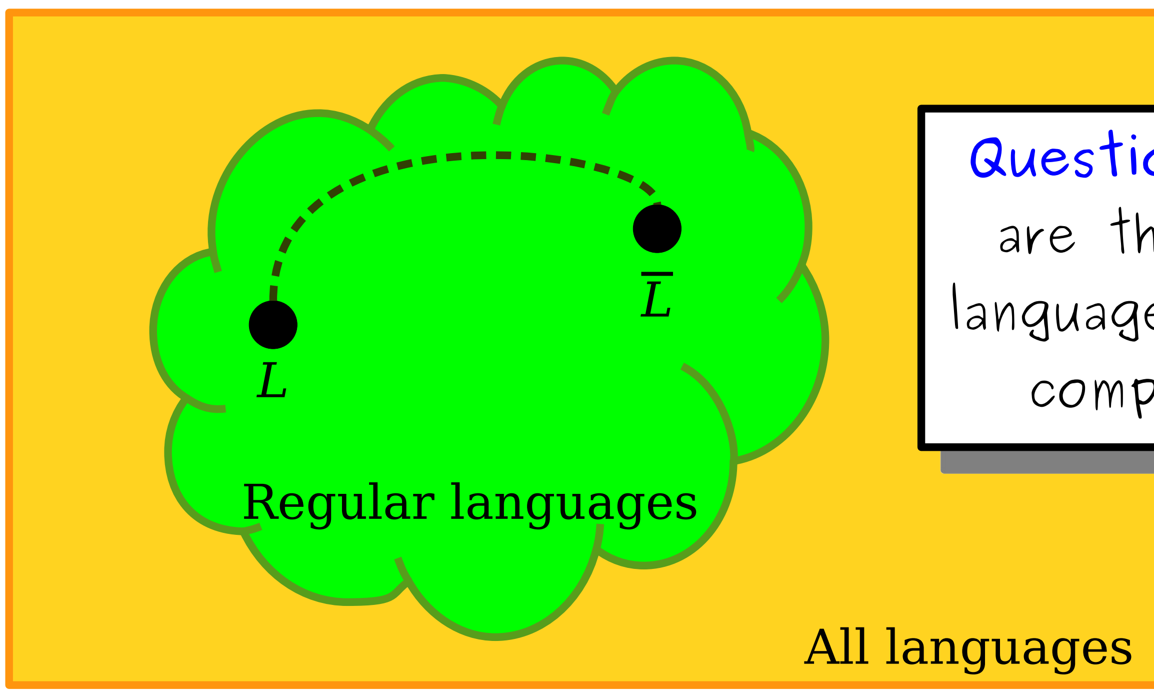
Complementing Regular Languages

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



Closure Properties

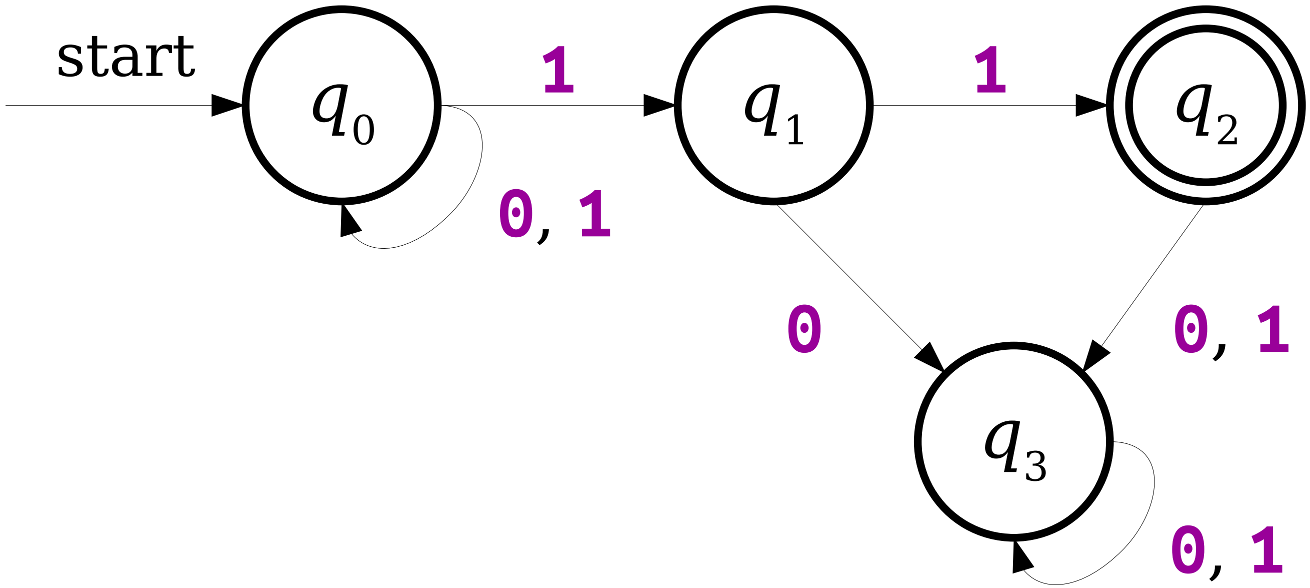
- ***Theorem:*** If L is a regular language, then $\overline{L}$ is also a regular language.
- As a result, we say that the regular languages are ***closed under complementation***.



Question to ponder: are the *nonregular* languages closed under complementation?

NFAs

The Motivation



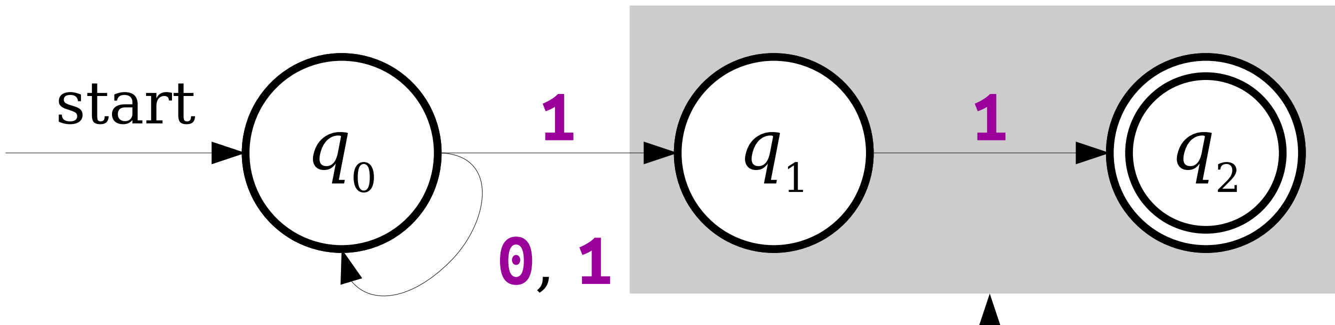
NFAs

- 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

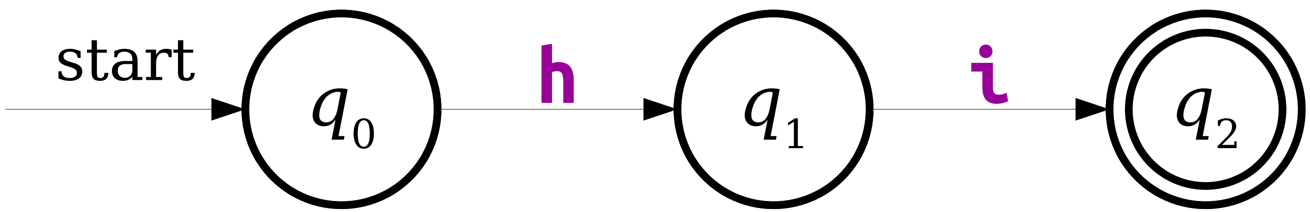
- A model of computation is ***deterministic*** if at every point in the computation, there is exactly one choice that can make.
  - The machine accepts if that series of choices leads to an accepting state.
- A model of computation is ***nondeterministic*** if the computing machine has a finite number 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.)

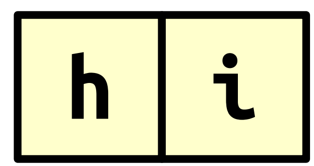
A More Complex NFA



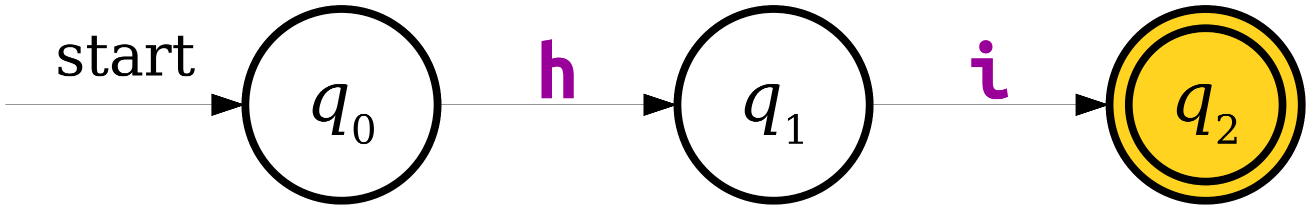
If a NFA needs to make a transition when no transition exists, the automaton dies and that particular path does not accept.

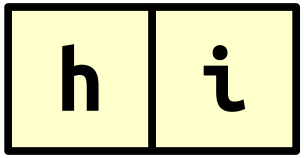
Hello, NFA!



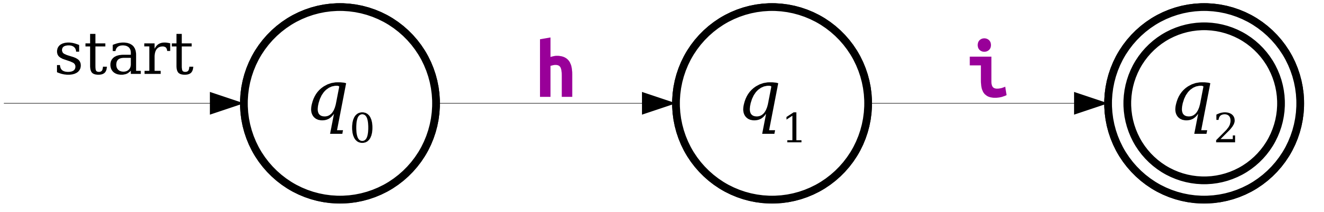


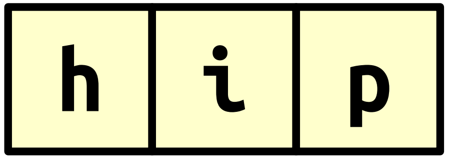
Hello, NFA!



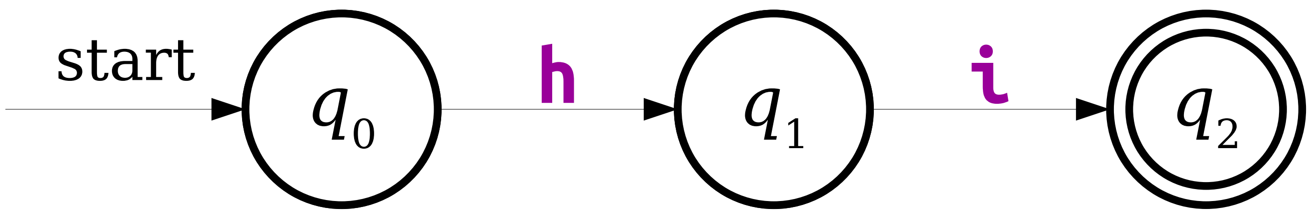


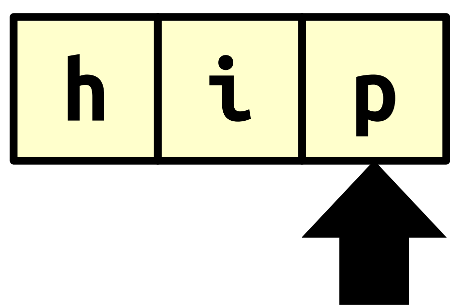
Tragedy in Paradise

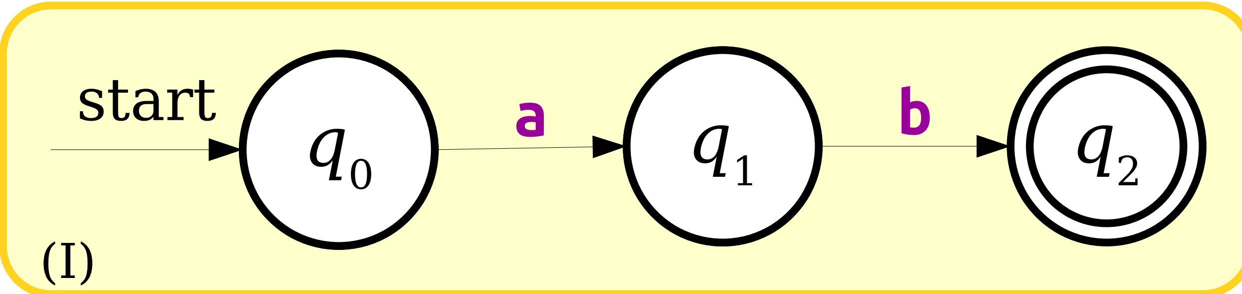




Tragedy in Paradise

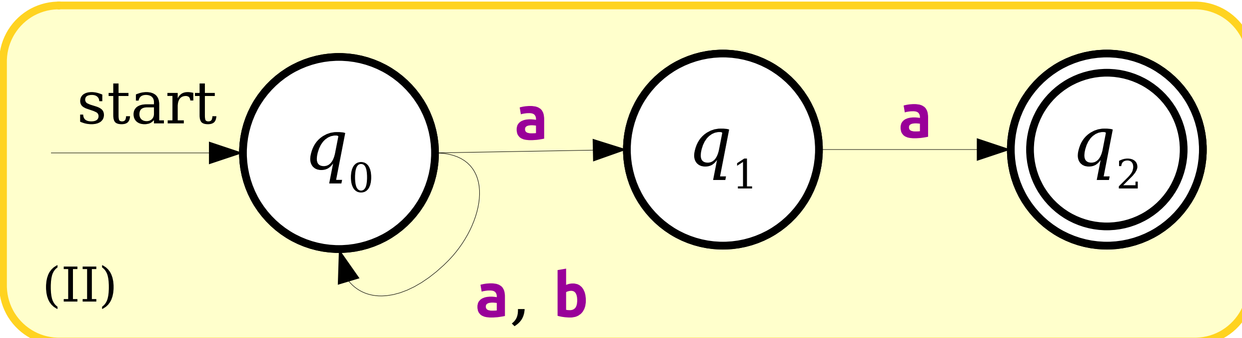




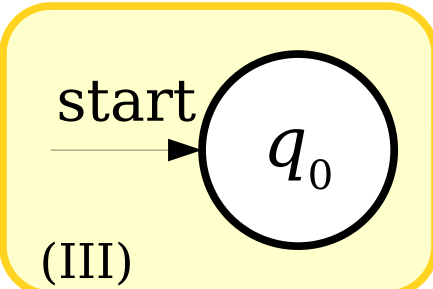


{ab}

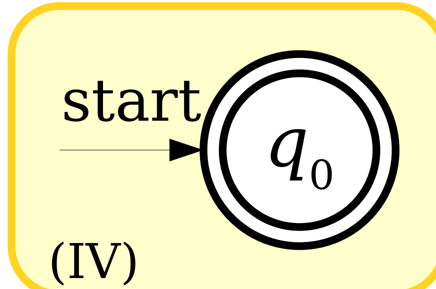
Question to ponder:
Why is the answer
$\{ \ w \in \Sigma^* \mid w$ ends in aaa $\}$
not correct?



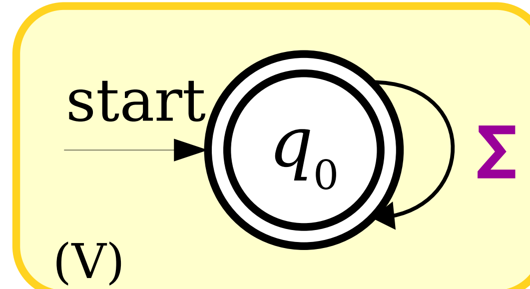
$\{ \ w \in \Sigma^* \mid w$ ends in aa $\}$



$\emptyset$



$\{\varepsilon\}$



Σ^*

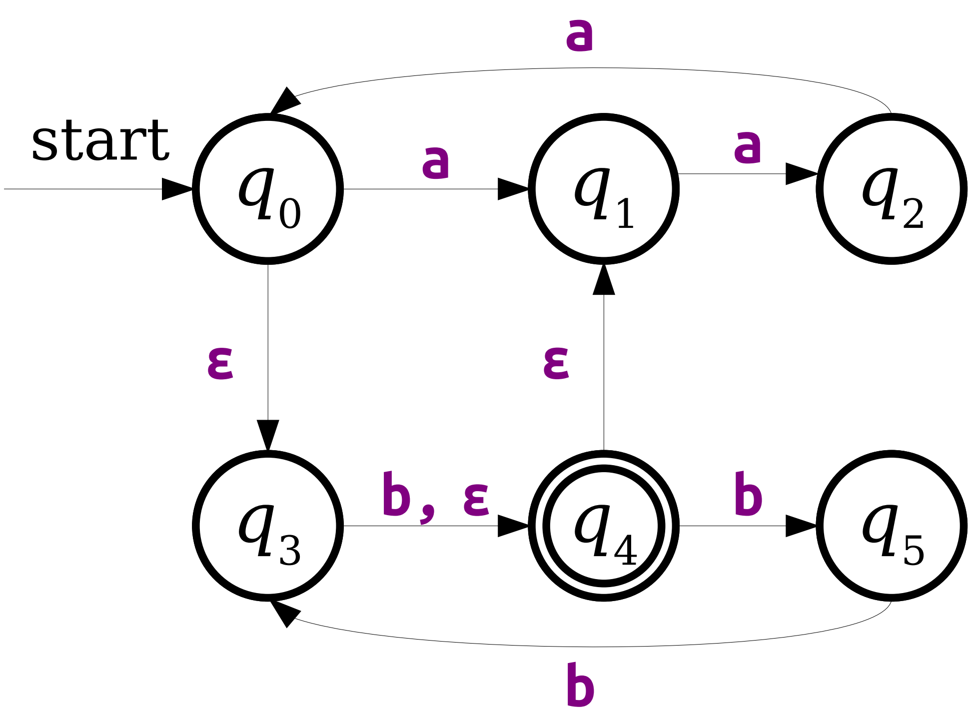
The ***language of an NFA*** is

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

What is the language of each NFA? (Assume $\Sigma = \{$a, b$\}$.)
Answer at [***https://cs103.stanford.edu/pollev***](https://cs103.stanford.edu/pollev)

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.

NFAs

- An NFA is defined relative to some alphabet Σ.
- For each state in the NFA, there may be ***any number of*** transitions defined for each symbol in Σ, plus any number of ε-transitions.
  - This is the "nondeterministic" part of NFA.
- There is a unique start state.
- There are zero or more accepting states.

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. Additionally, ε-transitions are not allowed.
  - This is the “deterministic” part of DFA.
- There is a unique start state.
- There are zero or more accepting states.

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

- 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

- 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.

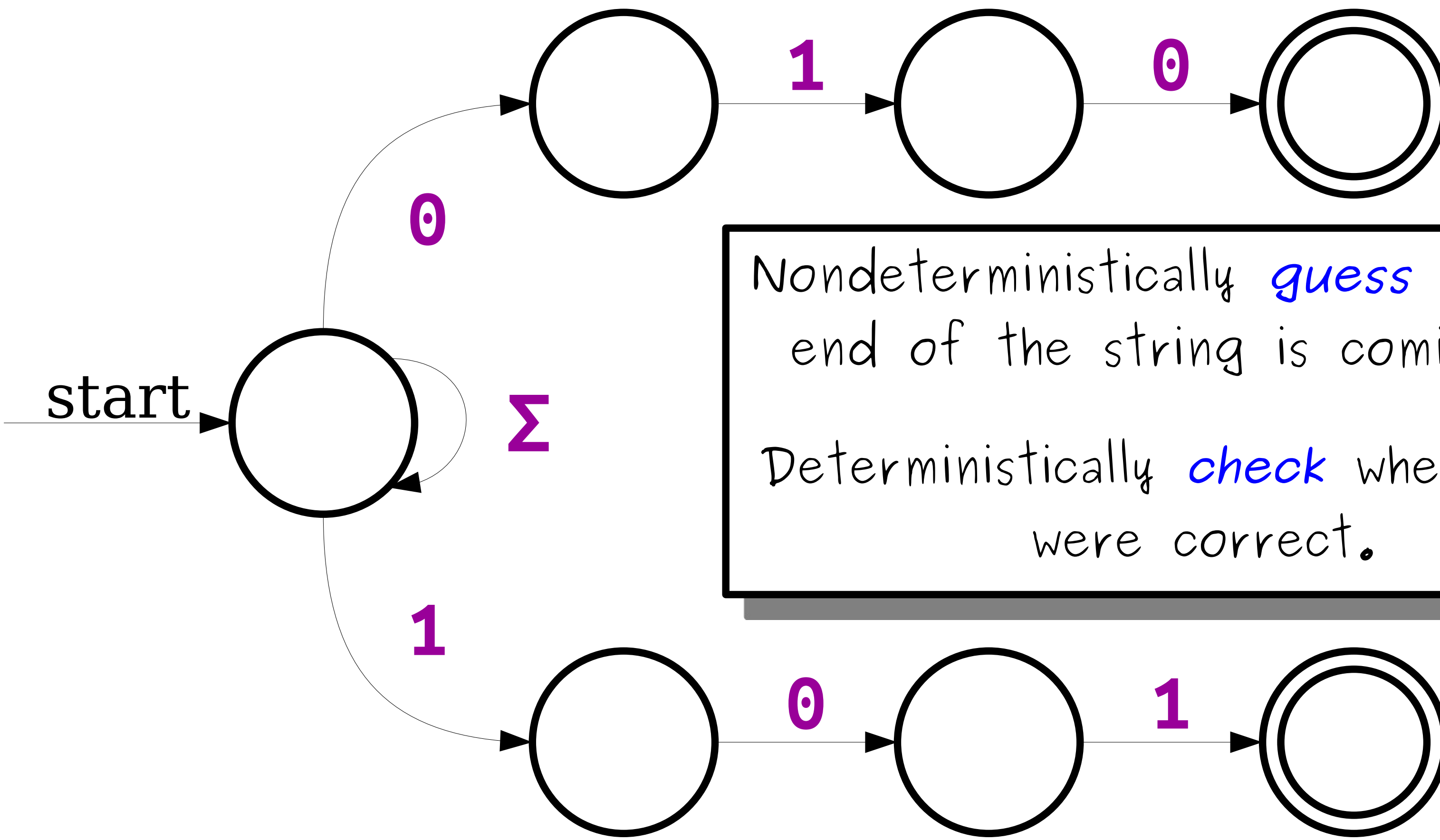
Designing NFAs

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

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

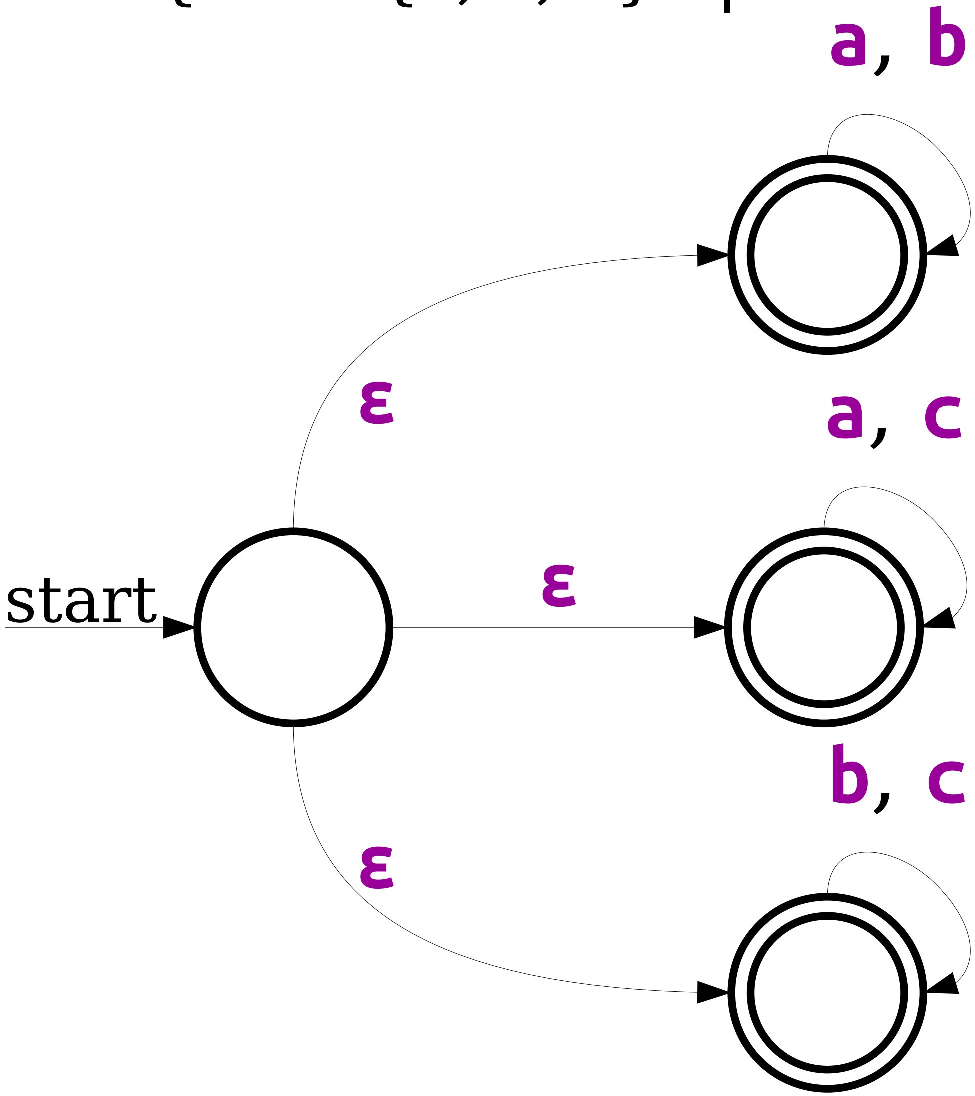


Nondeterministically *guess* when the end of the string is coming up.

Deterministically *check* whether you were correct.

Guess-and-Check

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



Nondeterministically *guess* which character is missing.

Deterministically *check* whether that character is indeed missing.

Just how powerful are NFAs?

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^*?