Nonregular Languages

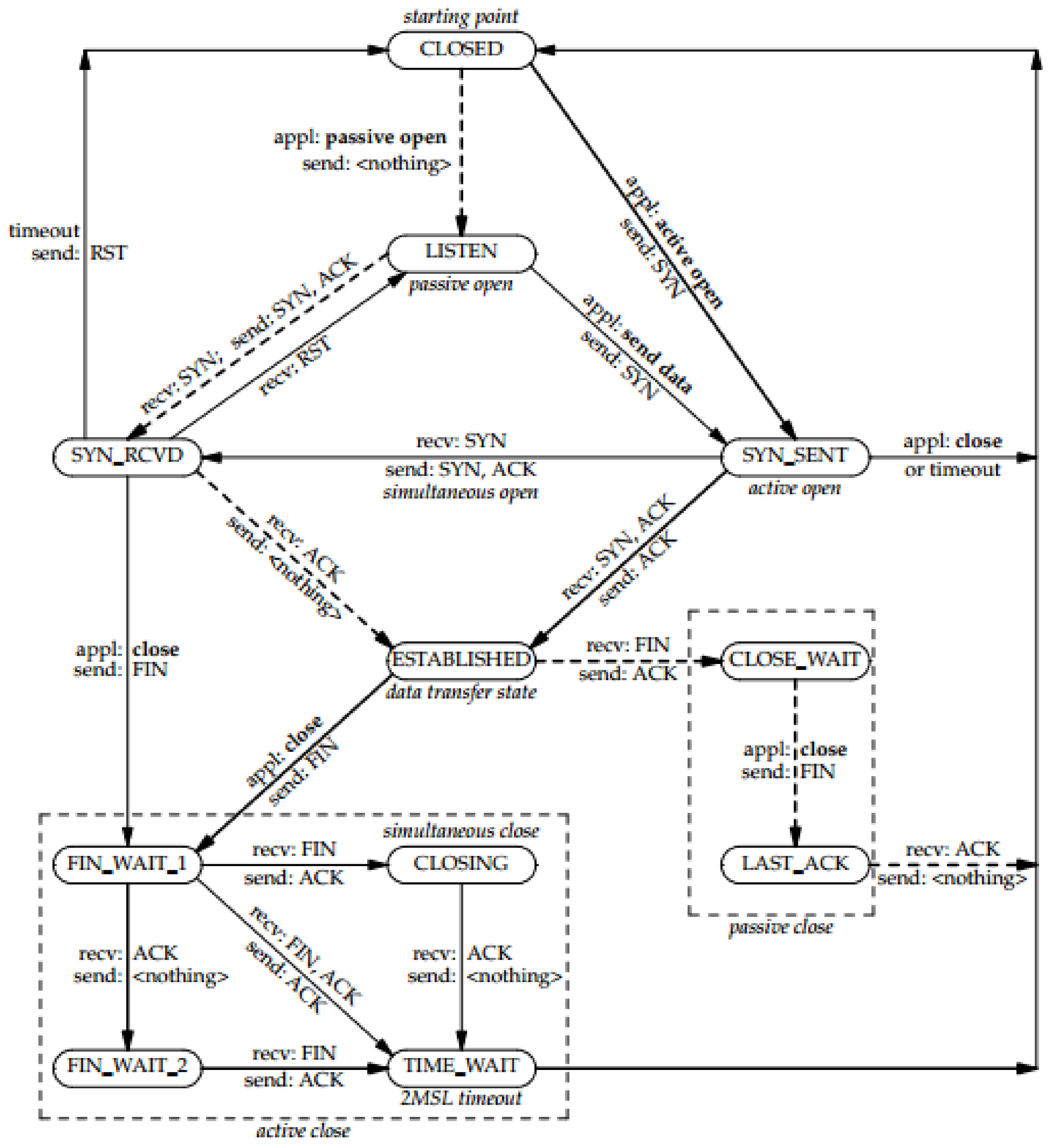
Theorem: The following are all equivalent:

- L is a regular language.
- There is a DFA D such that $\mathscr{L}(D) = L$.
- There is an NFA N such that $\mathscr{L}(N) = L$.
- There is a regular expression R such that $\mathscr{L}(R) = L$.

Why does this matter?

Buttons as Finite-State Machines:

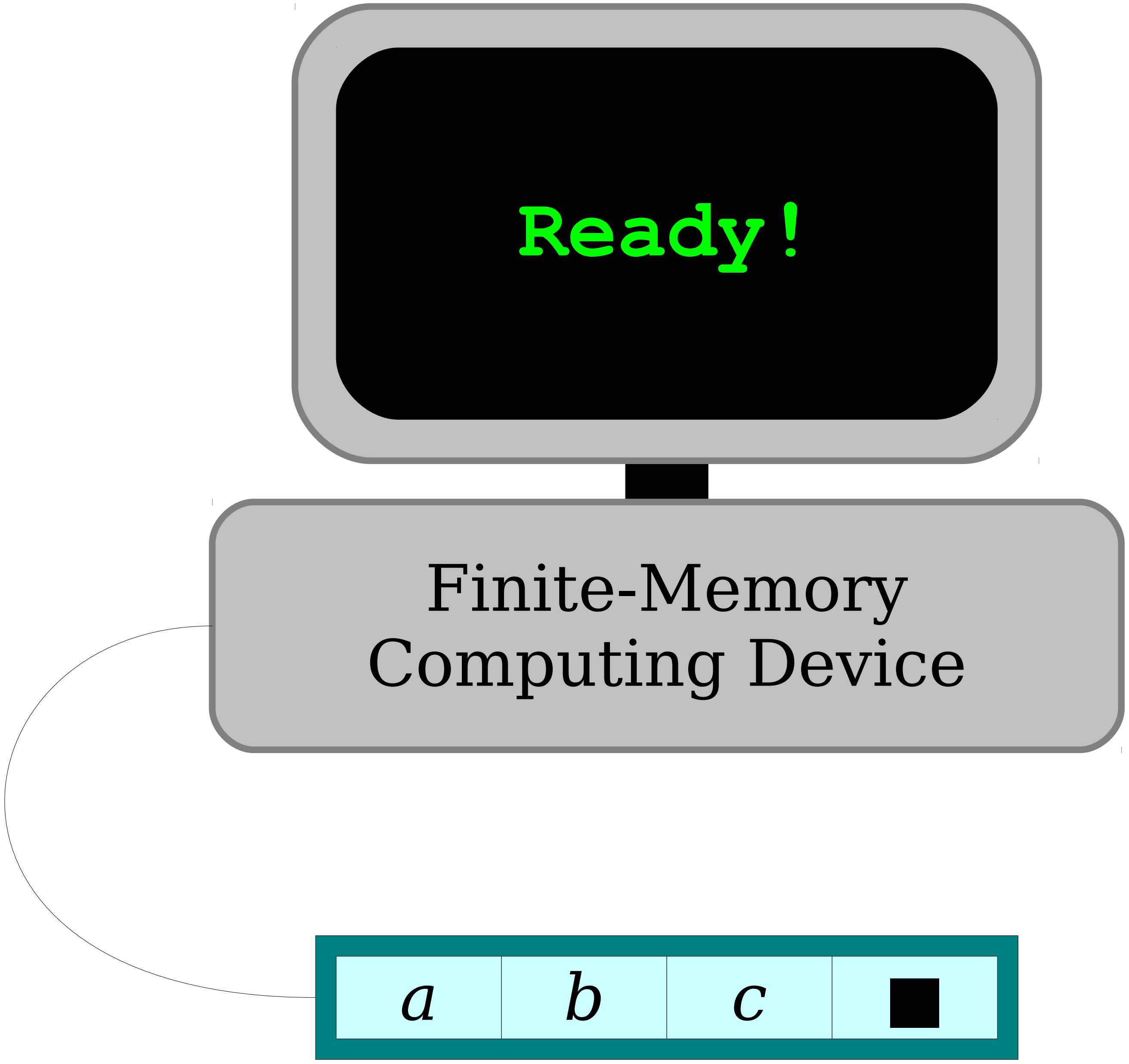
http://cs103.stanford.edu/tools/button-fsm/



⟶	*normal transitions for client*
- - →	*normal transitions for server*
appl:	*state transitions taken when application issues operation*
recv:	*state transitions taken when segment received*
send:	*what is sent for this transition*

http://www.tti.unipa.it/~gneglia/ip_networks06/slides/TCPIP_State_Transition_Diagram.pdf

What exactly is a finite-state machine?


Ready!
Finite-Memory Computing Device
a
b
c
■

The Model

- The computing device has internal workings that can be in one of finitely many possible configurations.
  - Each ***state*** in a DFA corresponds to some possible configuration of the internal workings.
- After each button press, the computing device does some amount of processing, then gets to a configuration where it's ready to receive more input.
  - Each ***transition*** abstracts away how the computation is done and just indicates what the ultimate configuration looks like.
- After the user presses the “done” button, the computer outputs either YES or NO.
  - The ***accepting*** and ***rejecting*** states of the machine model what happens when that button is pressed.

Computers as Finite Automata

- My computer has 12GB of RAM and 750GB of hard disk space.
- That's a total of 766GB of memory, which is 26,319,559,589,888 bits.
- There are “only” $2^{26,319,559,589,888}$ possible configurations of the memory in my computer.
- You could in principle build a DFA representing my computer, where there's one symbol per type of input the computer can receive.

A Powerful Intuition

- ***Regular languages correspond to problems that can be solved with finite memory.***
  - At each point in time, we only need to store one of finitely many pieces of information.
- Nonregular languages, in a sense, correspond to problems that cannot be solved with finite memory.
- Since every computer ever built has finite memory, in a sense, nonregular languages correspond to problems that cannot be solved by physical computers!

Finding Nonregular Languages

A Simple Language

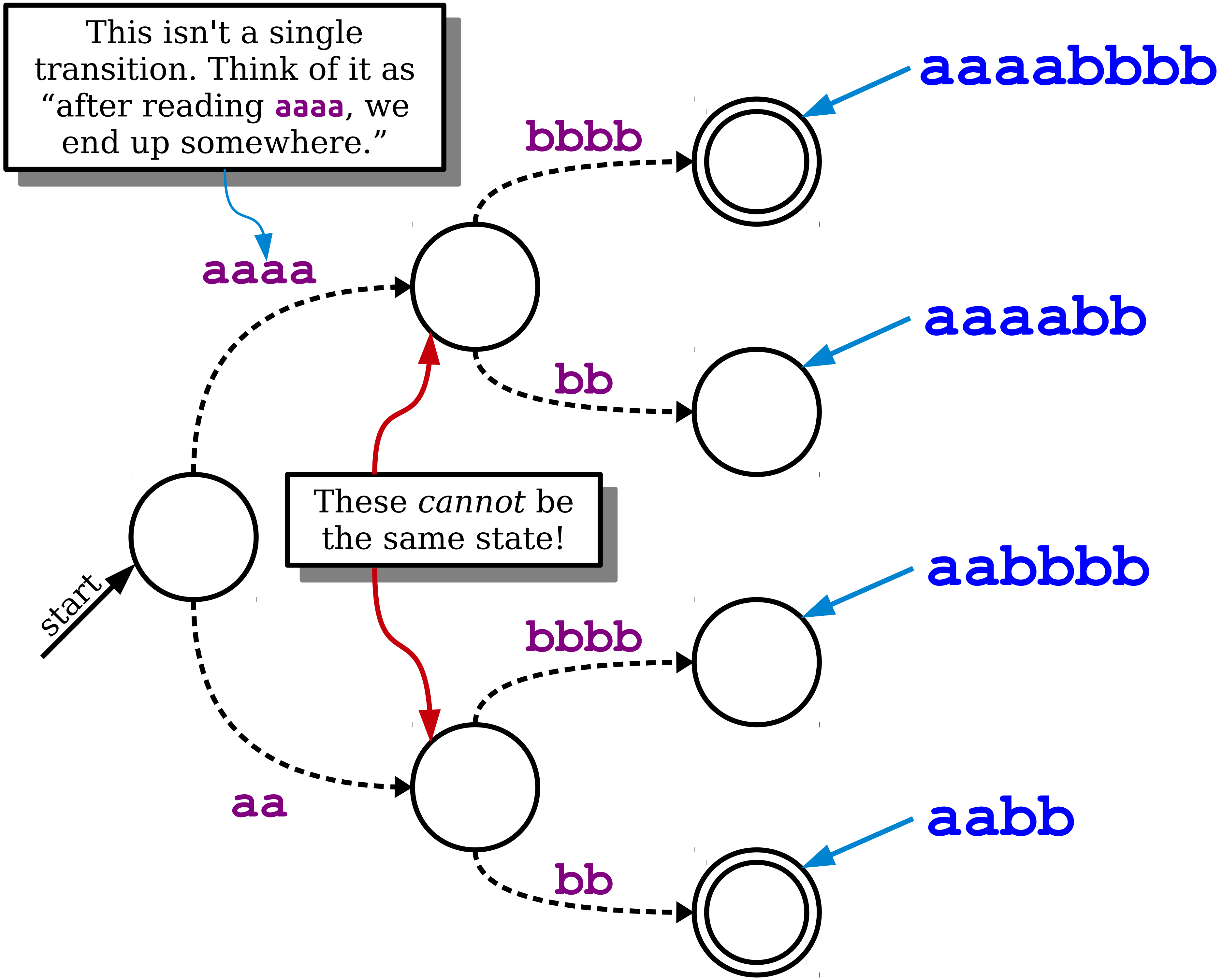
- Let $\Sigma = \{$**`a`**, **`b`**$\}$ and consider the following language:

$$E = \{ \mathtt{a}^n \mathtt{b}^n \mid n \in \mathbb{N} \}$$

- *E* is the language of all strings of *n* **`a`**'s followed by *n* **`b`**'s:

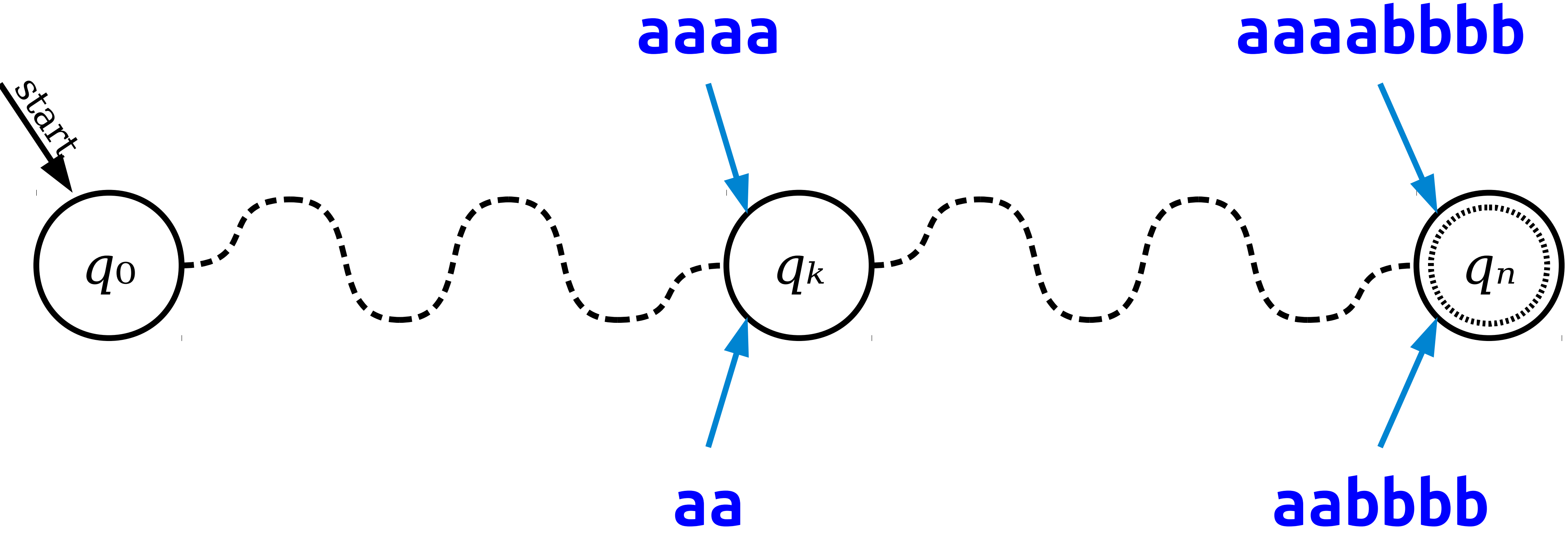
{ ε, **`ab`**, **`aabb`**, **`aaabbb`**, **`aaaabbbb`**, … }

- Is this language regular?


This isn't a single transition. Think of it as "after reading **aaaa**, we end up somewhere."

These *cannot* be the same state!

A Different Perspective



What happens if q_n is…

…an accepting state? We accept **aabbbb** $\notin E$!
…a rejecting state? We reject **aaaabbbb** $\in E$!

The Intuition

- As you just saw, the strings $\mathtt{a}^4$ and $\mathtt{a}^2$ can't end up in the same state in any DFA for the language $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$.
- Two proof routes:
    - *Direct:* The states you reach for $\mathtt{a}^4$ and $\mathtt{a}^2$ have to behave differently when reading $\mathtt{b}^4$ – in one case it should lead to an accept state, in the other it should lead to a reject state.
- *Contradiction:* Suppose they don't. Then $\mathtt{a}^4\mathtt{b}^4$ and $\mathtt{a}^2\mathtt{b}^4$ end up in the same state, so we either reject $\mathtt{a}^4\mathtt{b}^4$ (oops) or accept $\mathtt{a}^2\mathtt{b}^4$ (oops).

Distinguishability

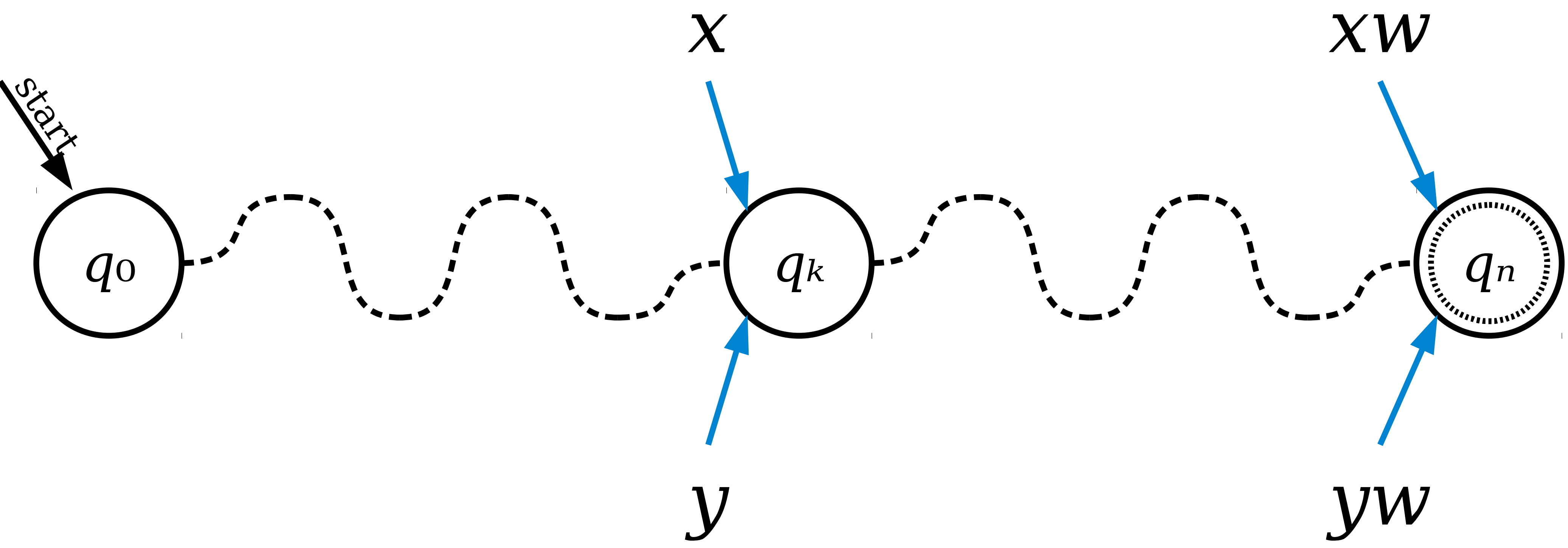
- Let L be an arbitrary language over Σ.
- Two strings $x \in \Sigma^*$ and $y \in \Sigma^*$ are called ***distinguishable relative to L*** if there is a string $w \in \Sigma^*$ such that exactly one of xw and yw is in L.
- We denote this by writing $\boldsymbol{x \not\equiv_L y}$.
- Formally: $x \not\equiv_L y$ if

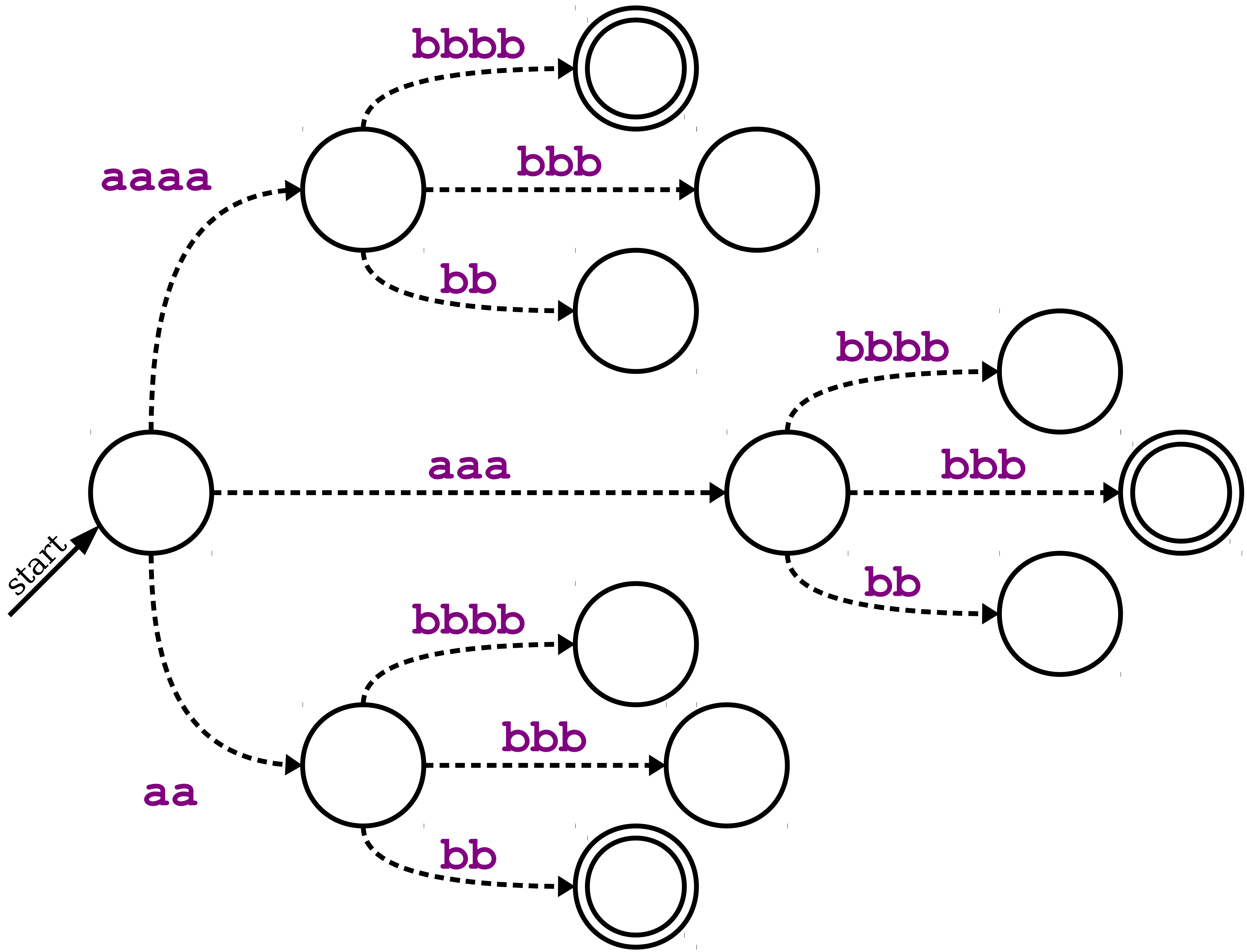
$$\exists w \in \Sigma^*.\ (xw \in L \leftrightarrow yw \notin L)$$

- In our previous case example, $\mathtt{a}^2$ and $\mathtt{a}^4$ are distinguishable relative to $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$.
  - Try appending $\mathtt{b}^4$ to both of them.

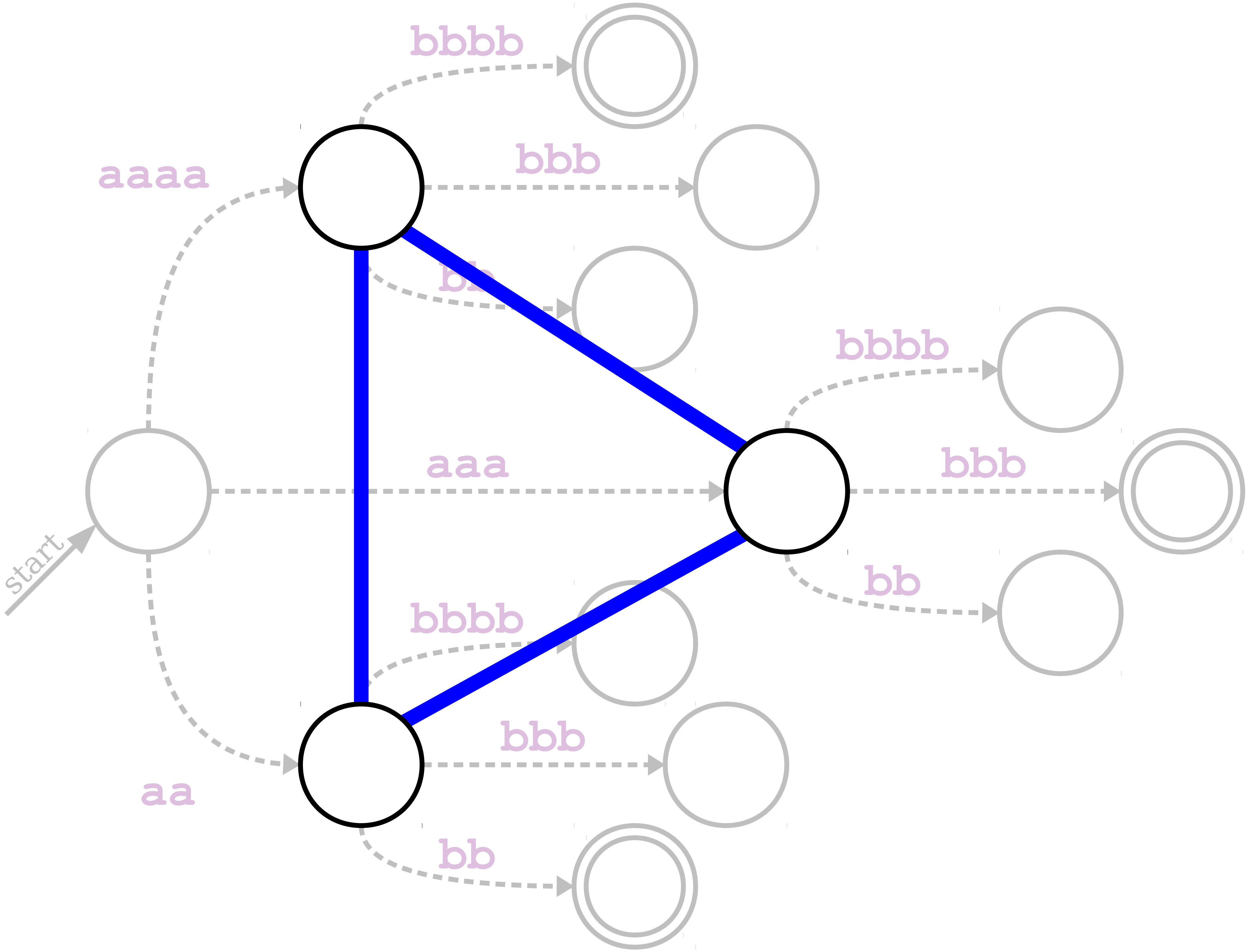
Distinguishability

- ***Theorem:*** Let L be a language over Σ. Let $x \in \Sigma^*$ and $y \in \Sigma^*$ be strings where $x \not\equiv_L y$. Then if D is any DFA for L, then D must end in different states when run on inputs x and y.
- ***Proof sketch:***





bbbb
aaaa
bbb
bb
bbbb
start
aaa
bbb
bb
bbbb
bbb
aa
bb


bbbb
aaaa
bbb
bb
bbbb
aaa
bbb
start
bb
bbbb
bbb
aa
bb

Distinguishability

- Let's focus on this language for now:

$$E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$$

- ***Lemma:*** If $m, n \in \mathbb{N}$ and $m \neq n$, then $\mathtt{a}^m \not\equiv_E \mathtt{a}^n$.
- ***Proof:*** Let $\mathtt{a}^m$ and $\mathtt{a}^n$ be strings where $m \neq n$. Then $\mathtt{a}^m\mathtt{b}^m \in E$ and $\mathtt{a}^n\mathtt{b}^m \notin E$. Therefore, by definition, we see that $\mathtt{a}^m \not\equiv_E \mathtt{a}^n$. ■

A Bad Combination

- Suppose there is a DFA D for the language $E = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$.
- We know the following:
  - Any two strings of the form $\mathtt{a}^m$ and $\mathtt{a}^n$, where $m \neq n$, cannot end in the same state when run through D.
  - There are infinitely many pairs of strings of the form $\mathtt{a}^m$ and $\mathtt{a}^n$.
  - However, there are only *finitely many* states they can end up in, since D is a deterministic ***finite*** automaton!
- If we put the pieces together, we see that…

Theorem: The language $E = \{ \texttt{a}^n\texttt{b}^n \mid n \in \mathbb{N} \}$ is not regular.

Proof: Suppose for the sake of contradiction that E is regular. Let D be a DFA for E, and let k be the number of states in D. Consider the strings $\texttt{a}^0$, $\texttt{a}^1$, $\texttt{a}^2$, …, $\texttt{a}^k$. This is a collection of $k+1$ strings and there are only k states in D. Therefore, by the pigeonhole principle, there must be two distinct strings $\texttt{a}^m$ and $\texttt{a}^n$ that end in the same state when run through D.

Our lemma tells us that $\texttt{a}^m \not\equiv_E \texttt{a}^n$, so by our earlier theorem we know that $\texttt{a}^m$ and $\texttt{a}^n$ cannot end in the same state when run through D. But this is impossible, since we know that $\texttt{a}^m$ and $\texttt{a}^n$ must end in the same state when run through D.

We have reached a contradiction, so our assumption must have been wrong. Therefore, E is not regular. ■

What Just Happened?

- ***We've just hit the limit of finite-memory computation.***
- To build a DFA for $E = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$, we need to have different memory configurations (states) for all possible strings of the form $\mathtt{a}^n$.
- There's no way to do this with finitely many possible states!

More Nonregular Languages

Another Language

- Consider the following language L over the alphabet $\Sigma = \{\mathtt{a}, \mathtt{b}, \overset{?}{=}\}$:

$$EQ = \{\ w\overset{?}{=}w \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$$

- EQ is the language all strings consisting of the same string of **a**'s and **b**'s twice, with a $\overset{?}{=}$ symbol in-between.
- Examples:

$$\mathtt{ab}\overset{?}{=}\mathtt{ab} \in EQ \qquad \mathtt{bbb}\overset{?}{=}\mathtt{bbb} \in EQ \qquad \overset{?}{=} \in EQ$$

$$\mathtt{ab}\overset{?}{=}\mathtt{ba} \notin EQ \qquad \mathtt{bbb}\overset{?}{=}\mathtt{aaa} \notin EQ \qquad \mathtt{b}\overset{?}{=} \notin EQ$$

Another Language

$$EQ = \{\ w\overset{?}{=}w \mid w \in \{\texttt{a}, \texttt{b}\}^*\}$$

- This language corresponds to the following problem:

Given strings x and y, does $x = y$?

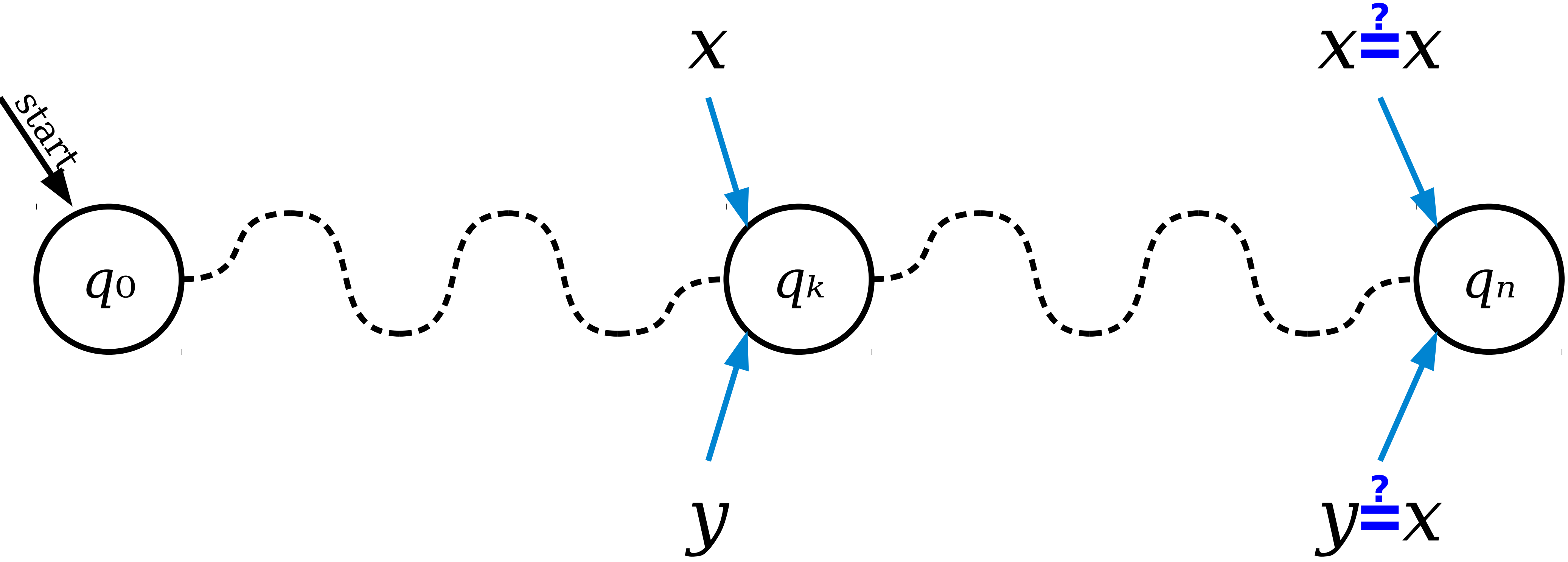
- Justification: $x = y$ iff $x\overset{?}{=}y \in EQ$.
- Is this language regular?

The Intuition

$$EQ = \{\ w\overset{?}{=}w \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$$

- Intuitively, any machine for EQ has to be able to remember the contents of everything to the left of the $\overset{?}{=}$ so that it can match them against the contents of the string to the right of the $\overset{?}{=}$.
- There are infinitely many possible strings we can see, but we only have finite memory to store which string we saw.
- That's a problem… can we formalize this?

The Intuition



What happens if q_n is…

…an accepting state? We accept $y \stackrel{?}{=} x \notin EQ$!

…a rejecting state? We reject $x \stackrel{?}{=} x \in EQ$!

Distinguishability

- Let's focus on this language for now:

$$EQ = \{ w \overset{?}{=} w \mid w \in \{\mathtt{a}, \mathtt{b}\}^* \}$$

- ***Lemma:*** If $x, y \in \{\mathtt{a}, \mathtt{b}\}^*$ and $x \neq y$, then $x \not\equiv_{EQ} y$.
- ***Proof:*** Let x and y be two distinct, arbitrary strings from $\{\mathtt{a}, \mathtt{b}\}^*$. Then $x \overset{?}{=} x \in L$ and $y \overset{?}{=} x \notin L$, so by definition we know $x \not\equiv_{EQ} y$. ■

Theorem: The language $EQ = \{\ w \overset{?}{=} w \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$ is not regular.

Proof: Suppose for the sake of contradiction that L is regular. Let D be a DFA for EQ and let k be the number of states in D. Consider any $k+1$ distinct strings in $\{\mathtt{a}, \mathtt{b}\}^*$. Because D only has k states, by the pigeonhole principle there must be at least two strings x and y that, when run through D, end in the same state.

Our lemma tells us that $x \not\equiv_{EQ} y$. By our earlier theorem, this means that x and y cannot end in the same state when run through D. But this is impossible, since specifically chose x and y to end in the same state when run through D.

We have reached a contradiction, so our assumption must have been wrong. Thus EQ is not regular. ■

Time-Out for Announcements!

Midterm Pickup

- Didn't pick up your midterm after class last time? Stop by the Gates building to pick it up!
- Graded exams, plus solutions and statistics, are available in a filing cabinet in the Gates building near the normal CS103 cabinets.
- Please pick up your exam and review it – it's important to see how to improve!

Regrade Requests

- If you'd like to submit a problem set for a regrade, email the regrades list (cs103-aut1516-regrades@lists.stanford.edu) and let us know
  - which problems you'd like us to review and
  - why you believe that your answer is correct.
- If you'd like to submit a regrade request for the midterm, print out a midterm regrade request form from the website, attach it to your midterm, and hand it to me at lecture or during office hours.
- In both cases, all regrade requests need to be submitted at most one week after we've returned the problem set or exam.

Upcoming Talk

- The Department of Classics is hosting a talk on Friday, November 6 on the Antikythera Mechanism.
- This looks like it will be an interesting talk and might be fun – it's a physical computing device!
- Reception is at 5:30 in Levinthal Hall in the Stanford Humanities Center.

Your Questions

“Feelin' Kinda burned out from Midterm season. What'd you use to do as an undergrad to recover?”

Don't do what I did as an undergrad. Do this instead:

Find something that makes you happy and just go do it. Take a few hours and read a book. Go for a bike ride. Rent a car and drive to a park. Host a party. Bake a cake. Take some time to recover, recharge, and remind yourself what you like to do.

Once you've done that, look over your exams, figure out what areas you need to improve in, and focus your efforts there.

Hang in there. Everything's going to be okay.

“I'm a freshman and I really want to work in tech this summer. I'm in 106B and I feel like I have a decent amount of experience. How do I even get to the phone call stage? I've given my resumé to so many people and it seems really futile at this point.”

Applying for internships as a freshman can be a bit of an uphill battle – you haven't yet established a track record and the economics don't work in your favor.

If you haven't yet done so, consider looking at Explore Microsoft, Google Summer Engineering Practicum, or Facebook U. Also, remember that there's another round of internship hiring in winter, so don't despair!

“Why does the campus CS culture seem so strongly geared towards software engineering/startups/other consumer-centric applications as opposed to scientific computing? Does this branch not publicize itself as well, or is it just the Silicon Valley effect?”

Part of this is the politics of how the School of Engineering is positioned relative to industry – there are by design strong ties between SoE and the tech companies.

Scientific computing has (generally speaking) started taking more of a back seat to consumer technologies due to a combination of the end of the Cold War and the rise of personal computers. It's still there, but it's usually compartmentalized into other departments.

Back to CS103!

Comparing Proofs

Theorem: The language $E = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$ is not a regular language.

Proof: Suppose for the sake of contradiction that E is regular. Let D be a DFA for E and let k be the number of states in D.

Consider the strings $\mathtt{a}^0$, $\mathtt{a}^1$, $\mathtt{a}^2$, …, $\mathtt{a}^k$. This is a collection of $k+1$ strings and there are only k states in D. Therefore, by the pigeonhole principle, there must be two distinct strings $\mathtt{a}^m$ and $\mathtt{a}^n$ that end in the same state when run through D.

Our lemma tells us that $\mathtt{a}^m \not\equiv_E \mathtt{a}^n$. By our earlier theorem we know that $\mathtt{a}^m$ and $\mathtt{a}^n$ cannot end in the same state when run through D. But this is impossible, since we know that $\mathtt{a}^m$ and $\mathtt{a}^n$ must end in the same state when run through D.

We have reached a contradiction, so our assumption must have been wrong. Therefore, E is not regular. ■

Theorem: The language $EQ = \{ w \overset{?}{=} w \mid w \in \{\texttt{a}, \texttt{b}\}^*\}$ is not a regular language.

Proof: Suppose for the sake of contradiction that L is regular. Let D be a DFA for EQ and let k be the number of states in D.

Consider any $k+1$ distinct strings in $\{\texttt{a}, \texttt{b}\}^*$.These are $k+1$ strings and there are only k states in D. By the pigeonhole principle, there must be two distinct strings x and y from this group that end in the same state when run through D.

Our lemma tells us that $x \not\equiv_{EQ} y$. By our earlier theorem we know that x and y cannot end in the same state when run through D. But this is impossible, since specifically chose x and y to end in the same state when run through D.

We have reached a contradiction, so our assumption must have been wrong. Therefore, EQ is not regular. ■

Theorem: The language L = **[fill in the blank]** is not a regular language.

Proof: Suppose for the sake of contradiction that L is regular. Let D be a DFA for L and let k be the number of states in D.

Consider **[some *k*+1 specific strings.]** This is a collection of $k+1$ strings and there are only k states in D. Therefore, by the pigeonhole principle, there must be two distinct strings x and y that end in the same state when run through D.

[Somehow we know] that $x \not\equiv_L y$. By our earlier theorem we know that x and y cannot end in the same state when run through D. But this is impossible, since we know that x and y must end in the same state when run through D.

We have reached a contradiction, so our assumption must have been wrong. Therefore, L is not regular. ■

Theorem: The language L = `[ fill in the blank ]` is not a regular language.

Proof: Suppose for the sake of contradiction that L is regular. Let D be a DFA for L and let k be the number of states in D.

Consider `[ some k+1 specific strings. ]` This is a collection of $k+1$ strings and there are only k states in D. Therefore, by the pigeonhole principle, there must be two distinct strings x and y that end in the same state when run through D.

`[ Somehow we know ]` that $x \not\equiv_L y$. By our earlier theorem we know that x and y cannot end in the same state when run through D. But this is impossible, since we know that x and y must end in the same state when run through D.

We have reached a contradiction, so our assumption must have been wrong. Therefore, L is not regular. ■

Imagine we have an infinite set of strings S with the following property:

$$\forall x \in S.\ \forall y \in S.\ (x \neq y \to x \not\equiv_L y)$$

What happens?

For any number of states k, we need a way to find k+1 strings so that two of them get into the same state…

[

of contradiction that L is regular.
et k be the number of states in D.

Consider [**some k+1 specific strings.**] This is a collection of k+1 strings and there are only k states in D. Therefore, by the pigeonhole principle, there must be two distinct strings x and y that end in the same state when run through D.

[**Somehow we know**] that $x \not\equiv_L y$. By our earlier theorem we know that x and y cannot end in the same state when run through D. But this is impossible, since we know that x and y must end in the same state when run through D.

We have reached a contradiction, s
have been wrong. Therefore, L is r

… and all those strings need to be distinguishable so that we get a contradiction.

The Myhill-Nerode Theorem

Theorem: Let L be a language over Σ. If there is a set $S \subseteq \Sigma^*$ with the following properties, then L is not regular:

- S is infinite (that is, S contains infinitely many strings).
- The strings in S are *pairwise distinguishable relative to L*. That is,

$$\forall x \in S.\ \forall y \in S.\ (x \neq y \to x \not\equiv_L y).$$

Proof: Let L be an arbitrary language over Σ. Let $S \subseteq \Sigma^*$ be an infinite set of strings with the following property: if $x, y \in S$ and $x \neq y$, then $x \not\equiv_L y$. We will show that L is not regular.

Suppose for the sake of contradiction that L is regular. This means that there must be some DFA D for L. Let k be the number of states in D. Since there are infinitely many strings in S, we can choose $k+1$ distinct strings from S and consider what happens when we run D on all of those strings. Because there are only k states in D and we've chosen $k+1$ strings from S, by the pigeonhole principle we know that at least two strings from S must end in the same state in D. Choose any two such strings and call them x and y.

Because x and y are distinct strings in S, we know that $x \not\equiv_L y$. Our earlier theorem therefore tells us that when we run D on inputs x and y, they must end up in different states. But this is impossible – we chose x and y precisely because they end in the same state when run through D.

We have reached a contradiction, so our assumption must have been wrong. Thus L is not a regular language. ■

Using the Myhill-Nerode Theorem

Theorem: The language $E = \{ \texttt{a}^n\texttt{b}^n \mid n \in \mathbb{N} \}$ is not regular.

To use the Myhill-Nerode theorem, we need to find an infinite set of strings that are pairwise distinguishable relative to E.

We know that any two strings of the form $\texttt{a}^n$ and $\texttt{a}^m$, where $n \neq m$, are distinguishable.

So pick the set $S = \{ \texttt{a}^n \mid n \in \mathbb{N} \}$.

Notice that S isn't a subset of E. That's okay: we never said that S needs to be a subset of E!

Theorem: The language $E = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$ is not regular.

Proof: Let $S = \{ \mathtt{a}^n \mid n \in \mathbb{N} \}$. This set is infinite because it contains one string for each natural number. Now, consider any strings $\mathtt{a}^n, \mathtt{a}^m \in S$ where $\mathtt{a}^n \neq \mathtt{a}^m$. Then $\mathtt{a}^n\mathtt{b}^n \in E$ and $\mathtt{a}^m\mathtt{b}^n \notin E$. Consequently, $\mathtt{a}^n \not\equiv_E \mathtt{a}^m$. Therefore, by the Myhill-Nerode theorem, L is not regular. ■

Theorem: The language $EQ = \{ w \overset{?}{=} w \mid w \in \{\mathtt{a}, \mathtt{b}\}^* \}$ is not regular.

To use the Myhill-Nerode theorem, we need to find an infinite set of strings that are pairwise distinguishable relative to EQ.

We know that any two distinct strings over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ are distinguishable.

So pick the set $S = \{\mathtt{a}, \mathtt{b}\}^*$.

Notice that S isn't a subset of EQ. That's okay: we never said that S needs to be a subset of EQ!

Theorem: The language $EQ = \{\ w \overset{?}{=} w \mid w \in \{\texttt{a}, \texttt{b}\}^*\}$ is not regular.

Proof: Let $S = \{\texttt{a}, \texttt{b}\}^*$. This set contains infinitely many strings. Now, consider any $x, y \in S$ where $x \ne y$. Then $x \overset{?}{=} x \in EQ$ and $y \overset{?}{=} x \notin EQ$. Consequently, $x \not\equiv_{EQ} y$. Therefore, by the Myhill-Nerode theorem, EQ is not regular. ■

Approaching Myhill-Nerode

- The challenge in using the Myhill-Nerode theorem is finding the right set of strings to use.
- ***General intuition:***
  - Start by thinking about what information a computer "must" remember in order to answer correctly.
  - Choose a group of strings that all require different information.
  - Prove that those strings are distinguishable relative to the language in question.

Tying Everything Together

- One of the intuitions we hope you develop for DFAs is to have each state in a DFA represent some key piece of information the automaton has to remember.
- If you only need to remember one of finitely many pieces of information, that gives you a DFA.
- If you need to remember one of infinitely many pieces of information, you can use the Myhill-Nerode theorem to prove that the language has no DFA.

Next Time

- **Context-Free Languages**
  - Context-Free Grammars
  - Generating Languages from Scratch