Nonregular Languages

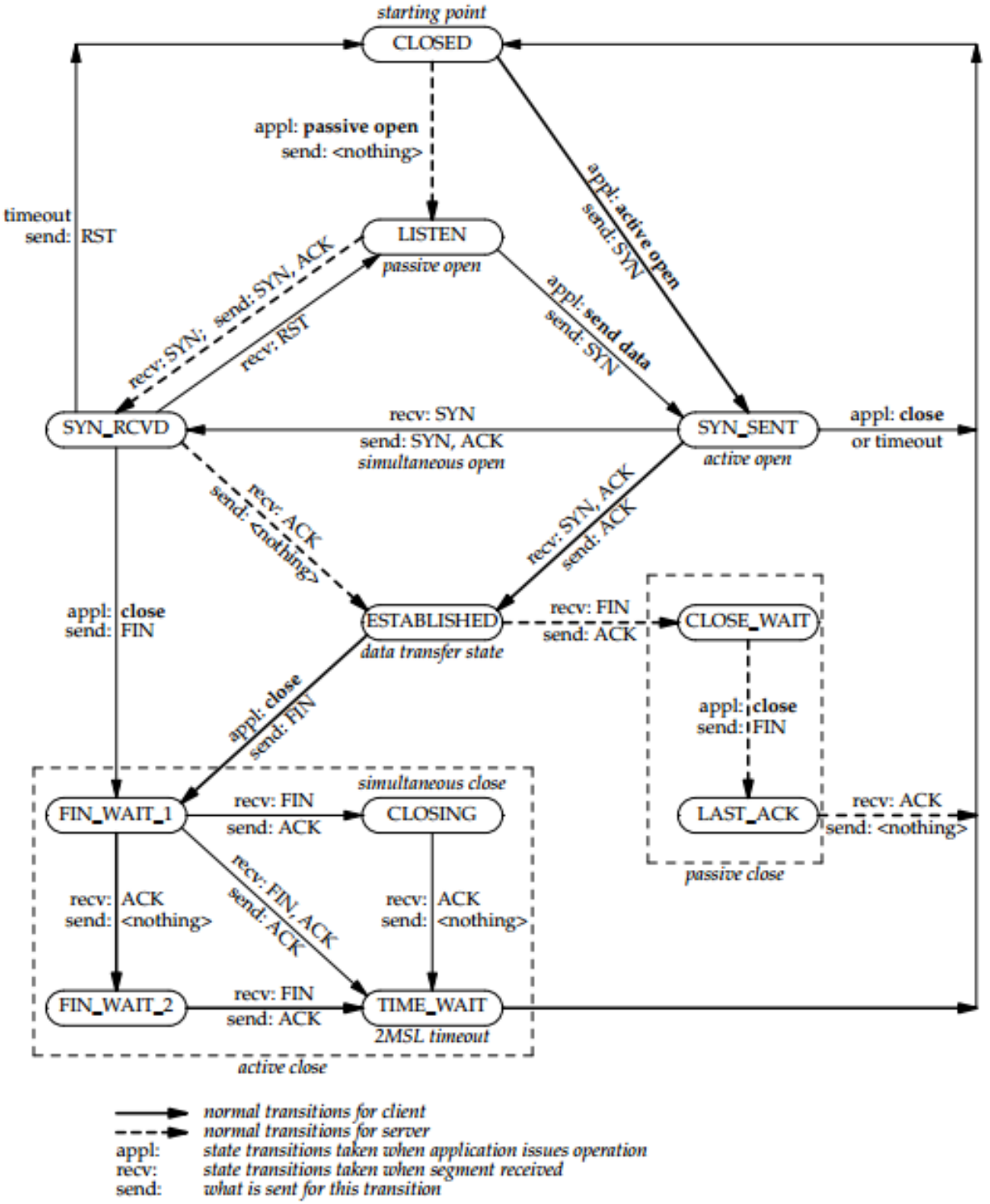
Theorem: The following are all equivalent:

- 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/



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

What exactly is a finite-state machine?

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 8GB of RAM and 750GB of hard disk space.
- That's a total of 758GB of memory, which is 6,511,170,420,736 bits.
- There are “only” $2^{6,511,170,420,736}$ possible configurations of my computer.
- 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

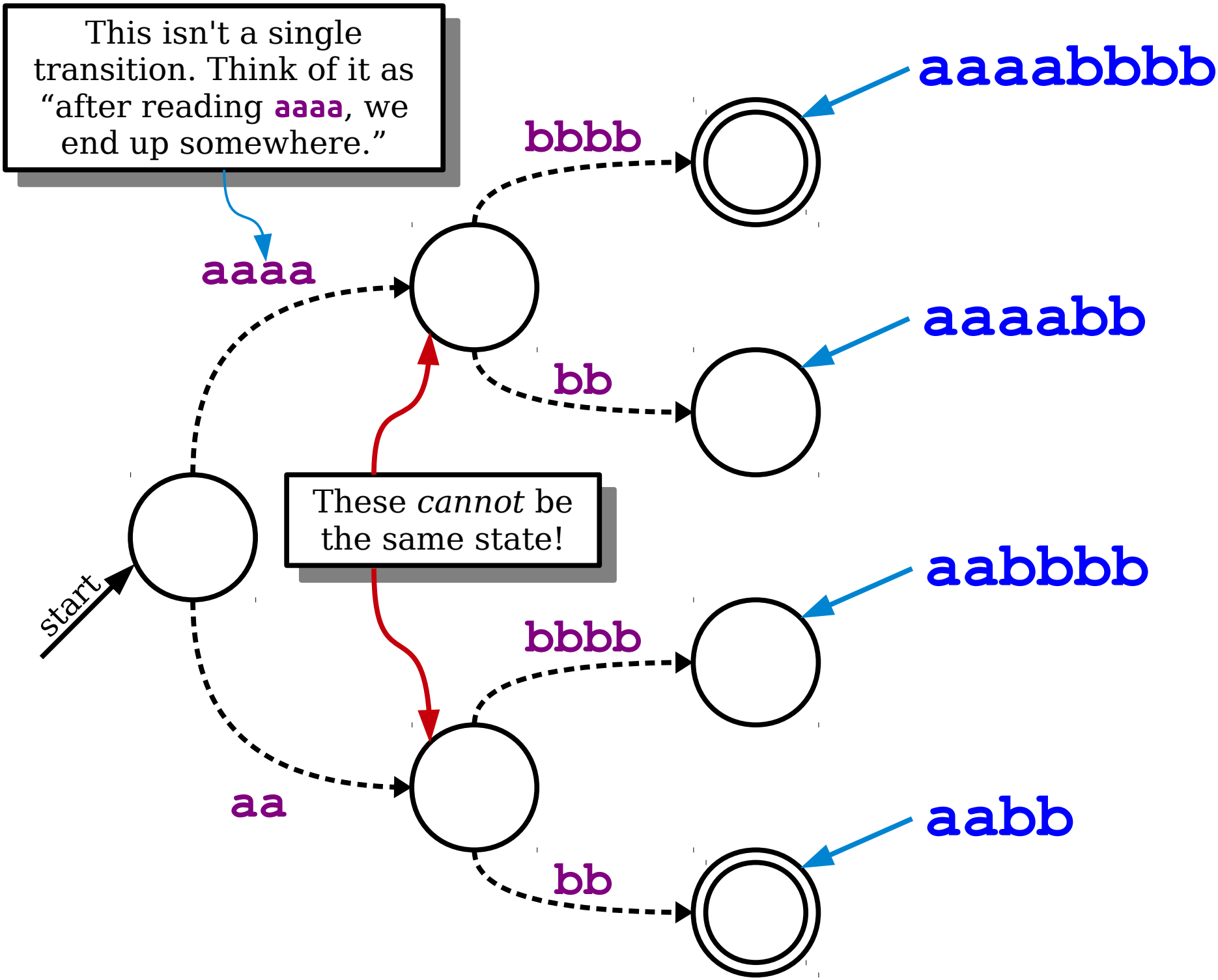
- Let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ and consider the following language:

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

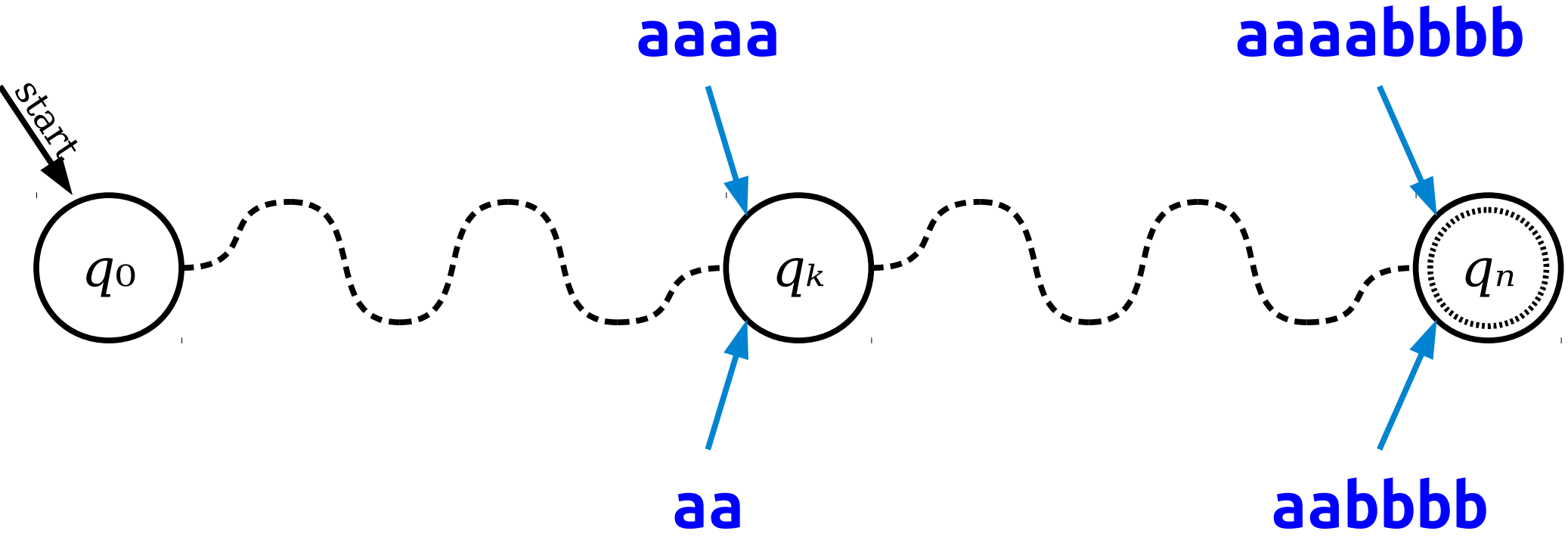
- L is the language of all strings of n **a**'s followed by n **b**'s:

$$\{\ \varepsilon,\ \mathtt{ab},\ \mathtt{aabb},\ \mathtt{aaabbb},\ \mathtt{aaaabbbb},\ \ldots\ \}$$

- Is this language regular?


This isn't a single transition. Think of it as "after reading aaaa, we end up somewhere."
aaaa
bbbb
aaaabbbb
bb
aaaabb
These cannot be the same state!
start
aa
bbbb
aabbbb
bb
aabb

A Different Perspective



What happens if q_n is...

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

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 $L = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$.
- Rationale:
  - Suppose they do.
  - Then $\mathtt{a}^4\mathtt{b}^4$ and $\mathtt{a}^2\mathtt{b}^4$ end up in the same state.
  - If it's an accepting state, then we accept $\mathtt{a}^2\mathtt{b}^4$, which isn't in L.
  - If it's a rejecting state, then we reject $\mathtt{a}^4\mathtt{b}^4$, which is in L.

Distinguishability

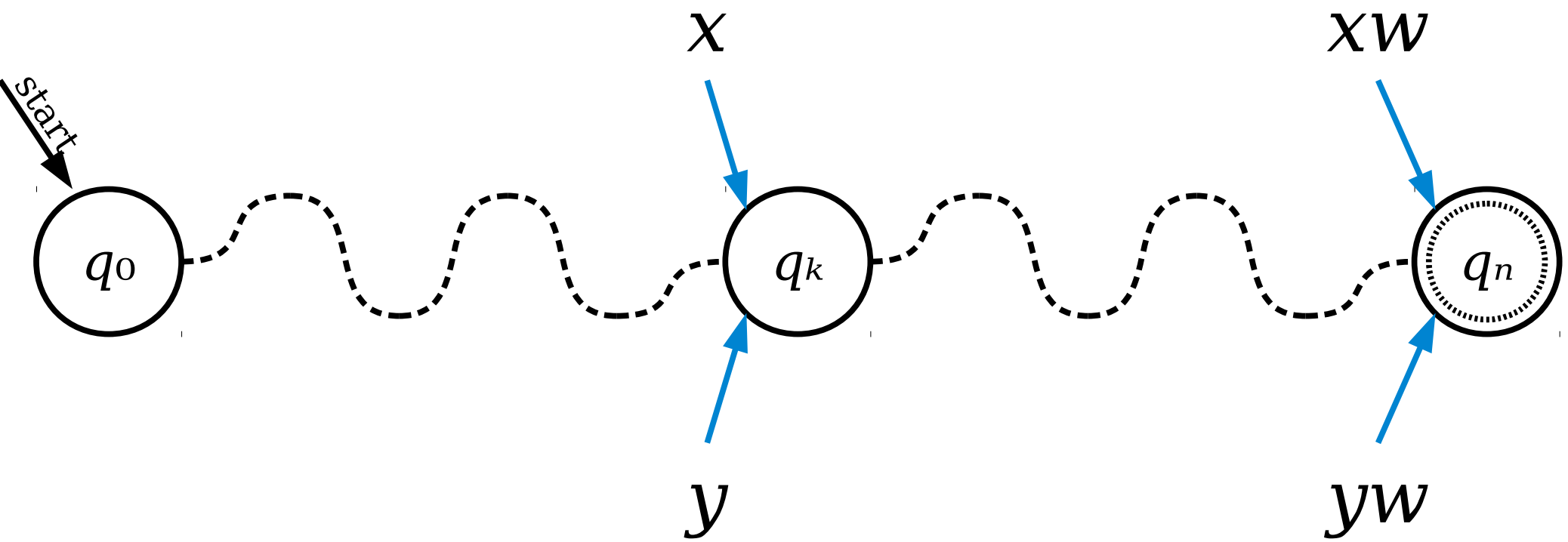
- Let L be a language over Σ.
- Two strings $x \in \Sigma^*$ and $y \in \Sigma^*$ are called ***distinguishable relative to L*** if there is a string $w \in \Sigma^*$ such that exactly one of xw and yw is in L.
- We denote this by writing $\boldsymbol{x \not\equiv_L y}$.
- Formally: $x \not\equiv_L y$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

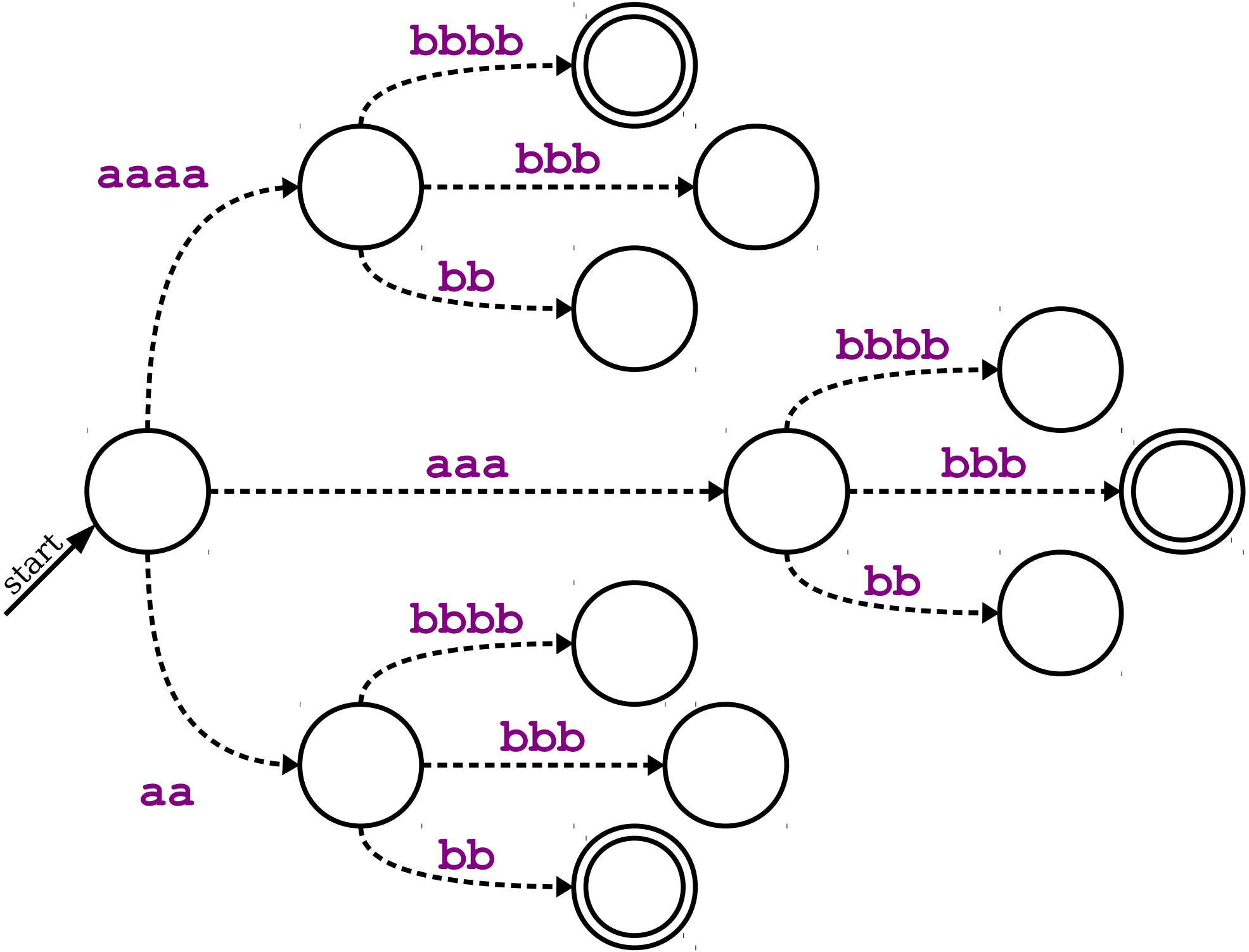
$$\boldsymbol{\exists w \in \Sigma^*. (xw \in L \leftrightarrow yw \notin L)}$$

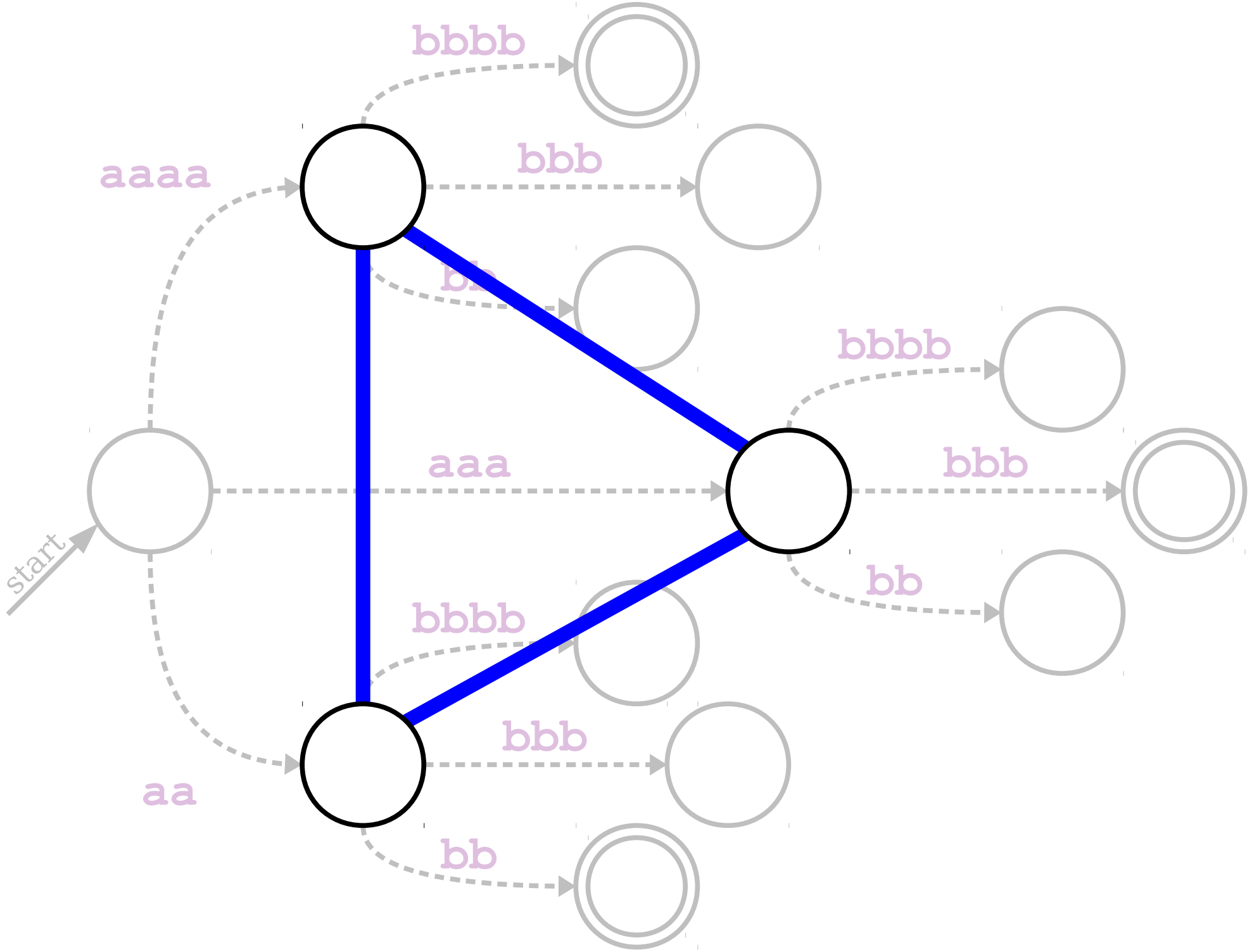
- In our previous case example, $\mathtt{a}^2$ and $\mathtt{a}^4$ are distinguishable relative to $\{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$.
  - Try appending $\mathtt{b}^4$ to both of them.

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
aaa
bbb
bb
start
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:

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

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

A Bad Combination

- Suppose there is a DFA D for the language $L = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$.
- 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 $L = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$ is not regular.

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 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 $\mathtt{a}^m \not\equiv_L \mathtt{a}^n$, so 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, L is not regular. ■

What Just Happened?

- ***We've just hit the limit of finite-memory computation.***
- To build a machine for $\{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$, 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 $\Sigma = \{\mathtt{a}, \mathtt{b}, \stackrel{?}{=}\}$:

$$L = \{\ w \stackrel{?}{=} w \mid w \in \{\mathtt{a}, \mathtt{b}\}^*\}$$

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

$\mathtt{ab} \stackrel{?}{=} \mathtt{ab} \in L$ $\quad \mathtt{bbb} \stackrel{?}{=} \mathtt{bbb} \in L$ $\quad \stackrel{?}{=} \in L$

$\mathtt{ab} \stackrel{?}{=} \mathtt{ba} \notin L$ $\quad \mathtt{bbb} \stackrel{?}{=} \mathtt{aaa} \notin L$ $\quad \mathtt{b} \stackrel{?}{=} \notin L$

Another Language

$$L = \{ w \stackrel{?}{=} w \mid w \in \{\texttt{a}, \texttt{b}\}^* \}$$

- This language corresponds to the following problem:

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

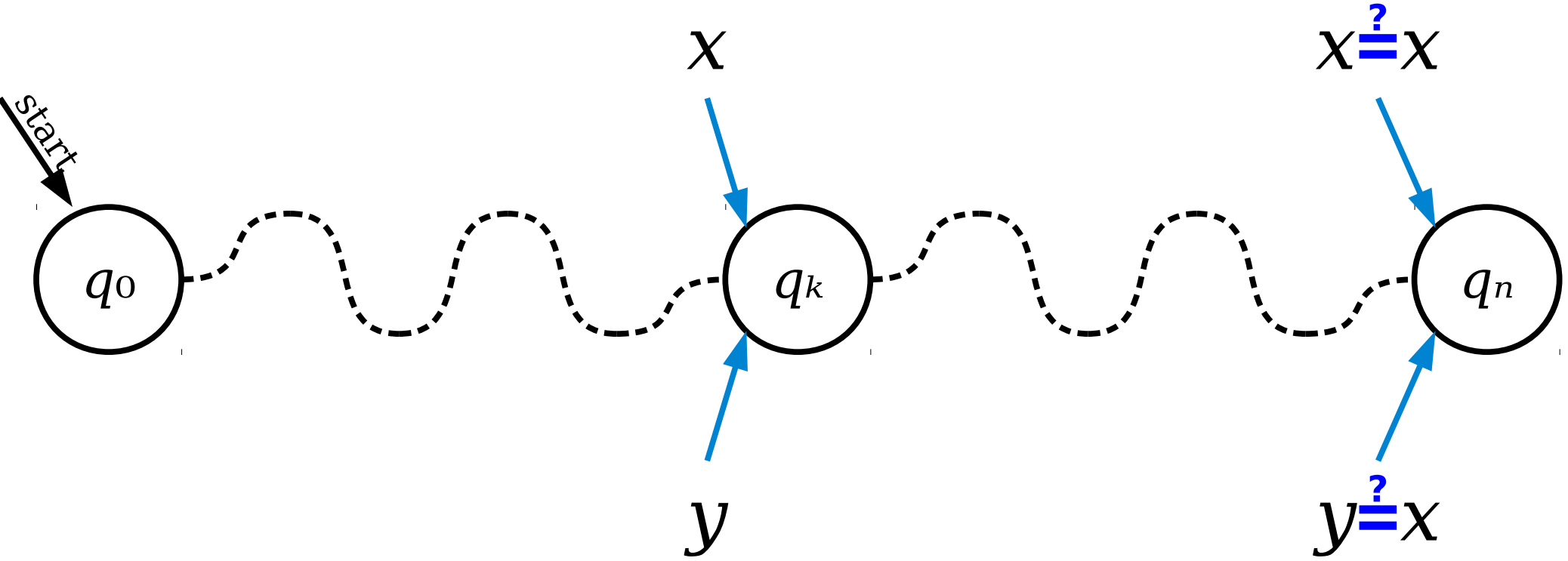
- Justification: $x = y$ iff $x \stackrel{?}{=} y \in L$.
- Is this language regular?

The Intuition

$$L = \{ w \stackrel{?}{=} w \mid w \in \{\texttt{a}, \texttt{b}\}^* \}$$

- Intuitively, any machine for L has to be able to remember the contents of everything to the left of the $\stackrel{?}{=}$ so that it can match them against the contents of the string to the right of the $\stackrel{?}{=}$.
- 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 $y \stackrel{?}{=} x \notin L$!

...a rejecting state? We reject $x \stackrel{?}{=} x \in L$!

Distinguishability

- Let's focus on this language for now:

$$L = \{ w \stackrel{?}{=} w \mid w \in \{\texttt{a}, \texttt{b}\}^* \}$$

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

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

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 any $k+1$ distinct strings in $\{\texttt{a}, \texttt{b}\}^*$. 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 $x \not\equiv_L y$. 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 L 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!
- Solutions and statistics available in the normal return filing cabinet (directions online).
- Graded exams in the drawer just below that.

Regrade Requests

- If you'd like to submit a problem set for a regrade, email the regrades list (cs103-spr1415-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.

Your Questions

“What class(es) at Stanford have you not taken, but wish you had?”

In CS: CS147, CS229, and CS246.

Outside CS: English 90, Bio 150, CEE 151, SLE.

“We thoroughly enjoyed your last talk on nixtamalization. Speak (again) on something that fascinates you! Teach us something new (again)!”

I'd like to show you an odd building with an odd story…

Back to CS103!

The General Pattern

- These previous two proofs have the following shape:
  - Find an infinite collection of strings that cannot end up in the same state in any DFA for a language L.
  - Conclude that since any DFA for L has only finitely many states, that L cannot be regular.
- Two questions:
  - Is there a bigger picture here?
  - Can we abstract these proofs away from machine specifics?

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

- S is infinite (i.e. it contains infinitely many strings).
- If $x, y \in S$ and $x \neq y$, then $x \not\equiv_L y$.

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 S is an infinite set, we can choose $k+1$ distinct strings from S. Because there are only k states in D and we've chosen $k+1$ distinct 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. Since $x \in S$ and $y \in S$ and $x \neq y$, we know that $x \not\equiv_L y$. Consequently, by our earlier theorem, we know that x and y must end in different states when run through D. But this is impossible – we chose x and y specifically 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. ■

Theorem: The language $L = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$ 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 $\mathtt{a}^n\mathtt{b}^n \in L$ and $\mathtt{a}^m\mathtt{b}^n \notin L$. Consequently, $\mathtt{a}^n \not\equiv_L \mathtt{a}^m$. Therefore, by the Myhill-Nerode theorem, L is not regular. ■

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

Proof: Let $S = \{\texttt{a}, \texttt{b}\}^*$. This set is infinite because it contains one string for each natural number. Now, consider any $x, y \in S$ where $x \neq y$. Then $x \stackrel{?}{=} x \in L$ and $y \stackrel{?}{=} x \notin L$. Consequently, $x \not\equiv_L y$. Therefore, by the Myhill-Nerode theorem, L is not regular. ■

Why it Works

- The Myhill-Nerode Theorem is, essentially, a generalized version of the argument from before.
  - If there are infinitely many distinguishable strings and only finitely many states, two distinguishable strings must end up in the same state.
  - Therefore, two strings that cannot be in the same state must end in the same state.
- Proof focuses on the infinite set of strings, not the DFA mechanics.

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.

Next Time

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