CS103
FALL 2025

Lecture 18:

Nonregular Languages

Recap from Last Time

Theorem: The following are all equivalent:

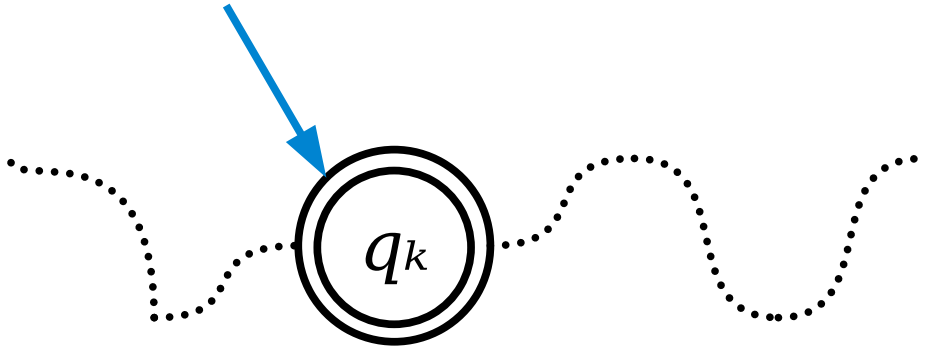
- L is a regular language.
- There is a DFA D such that $\mathscr{L}(D) = L$.
- There is an NFA N such that $\mathscr{L}(N) = L$.
- There is a regular expression R such that $\mathscr{L}(R) = L$.

New Stuff!

A Warm-Up Exercise

Consider a DFA for $\mathscr{L}$(**a* ∪ b***)...

Suppose we land here upon reading **aaaa**.



Not knowing what the rest of the DFA looks like, which of the following can we say are true?

(1) **aa** ***must*** also land in q_k.

(2) **aa** ***might*** also land in q_k.

(3) **aa** ***cannot*** land land in q_k.

(4) **bb** ***must*** also land in q_k.

(5) **bb** ***might*** also land in q_k.

(6) **bb** ***cannot*** land in q_k.

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

Nonregular Languages

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!
  - Though there's a bit of an asterisk here we need to address. Hold tight!

Finding Nonregular Languages

Finding Nonregular Languages

- To prove that a language is regular, we can just find a DFA, NFA, or regex for it.
- To prove that a language is not regular, we need to prove that there are no possible DFAs, NFAs, or regexes for it.
  - ***Claim:*** We can actually just prove that there's no DFA for it. Why is this?
- ***This sort of argument will be challenging.*** Our arguments will be somewhat technical in nature, since we need to rigorously establish that no amount of creativity could produce a DFA for a given language.
- Let's see an example of how to do this.

A Simple Language

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

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

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

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

A Simple Language

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

None of these regular expressions are regexes for the language E. Explain why not.

a*b*

(ab)*

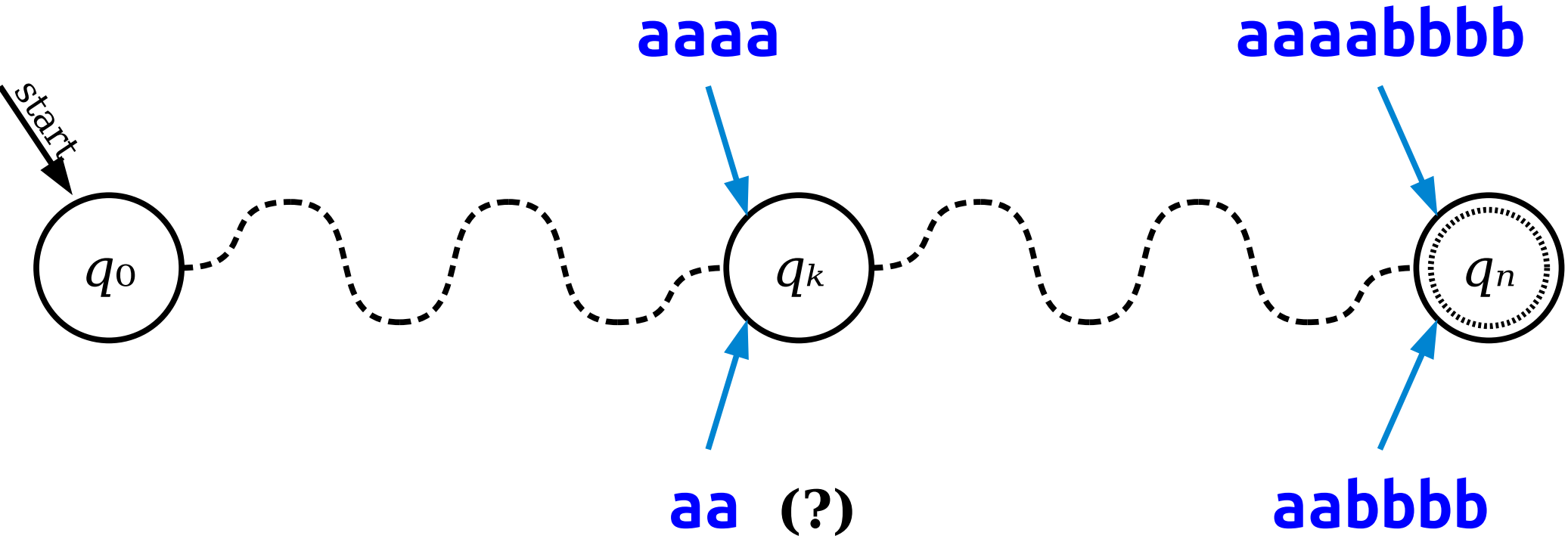
ε U ab U a²b² U a³b³

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

Let's imagine what a DFA for the language $\{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$ would have to look like.

Can we say anything about it?

Keeping Things Separated



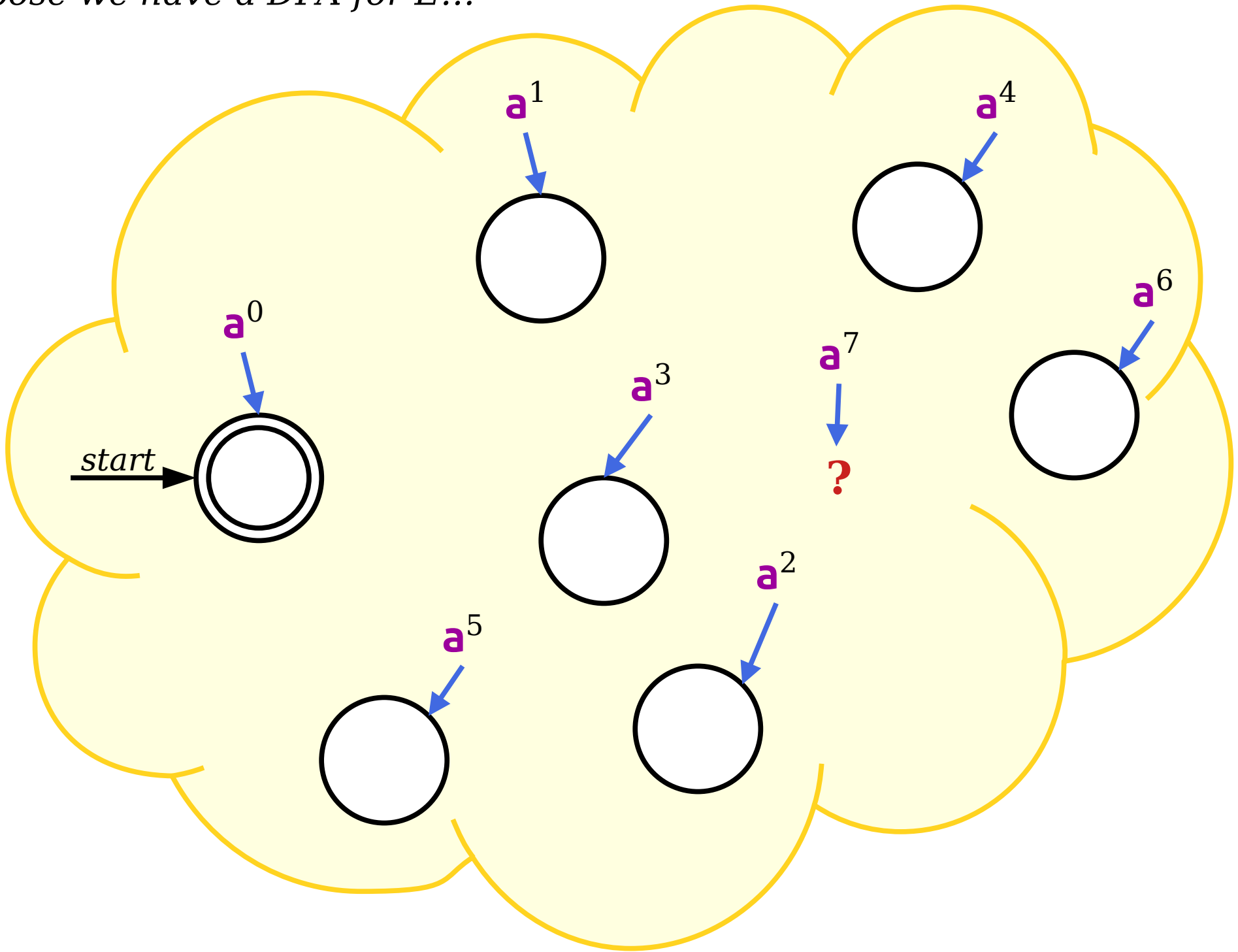
What happens if q_n is...

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

A More General Result

- ***Lemma:*** If D is a DFA for $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$ and we run D on *any* strings $\mathtt{a}^m \neq \mathtt{a}^n$, then those strings do not end in the same state.
- ***Proof Idea:***
  - Suppose you do end up in the same state. Then $\mathtt{a}^m\mathtt{b}^m$ and $\mathtt{a}^n\mathtt{b}^m$ end up in the same state (DFAs are deterministic, so we follow the same transitions). But then we either reject $\mathtt{a}^m\mathtt{b}^m$ (oops) or accept $\mathtt{a}^m\mathtt{b}^n$ (oops).
- ***Powerful intuition:*** *Any* DFA for E must keep $\mathtt{a}^m$ and $\mathtt{a}^n$ separated. It needs to remember something fundamentally different after reading those strings.

Suppose we have a DFA for E…



Theorem: The language $E = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$ 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 $\mathtt{a}^0$, $\mathtt{a}^1$, $\mathtt{a}^2$, ..., $\mathtt{a}^k$. 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. But this is impossible, since we know that $\mathtt{a}^m$ and $\mathtt{a}^n$ cannot 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. ■

We're going to see a simpler proof of this result later on once we've built more machinery. If (hypothetically speaking) you want to prove something like this in the future, we'd recommend not using this proof as a template.

What Just Happened?

- ***We've just hit the limit of finite-memory computation.***
  - A DFA for $E = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$ needs to keep infinitely many strings separated.
  - There's no way to do this with finitely many possible states!
- And so…

  … you can’t build a DFA for E,

  … or design an NFA for E,

  … or write a regex for E.
- ***Wow!***

Time-Out for Announcements!

Second Midterm Logistics

- Our second midterm exam is next ***Monday, November 10*** from ***7:00PM – 10:00PM***

Seating assignments have changed.

☞ ***Check the seating assignment page again.*** ☜

Write down your new seat.

- Topic coverage is primarily lectures 06 – 13 (functions through induction) and PS3 – PS5. Finite automata and onward won't be tested here.
  - Because the material is cumulative, topics from PS1 – PS2 and Lectures 00 – 05 are also fair game.
- The exam is closed-book and closed-computer. You can bring one double-sided 8.5" × 11" sheet of notes with you.
- Contact us ASAP if you need an alternate exam and haven't heard from us with date/time/place.

Review Session

- Kenneth is holding a review session tomorrow (Thursday). Location / time are up on Ed.
  - As with last time, this is not recorded.
  - As with last time, come prepared with questions you want to ask.
- We also have a ton of practice exams up on the course website.
- ***Best of luck – you can do this!***

Back to CS103!

Generalizing the Proof

What We Did

- Our proof that $E = \{\mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N}\}$ is not regular relied on several observations:
  - No two strings $\mathtt{a}^m \neq \mathtt{a}^n$ can end in the same state in any DFA for E. Appending $\mathtt{b}^m$ puts one string into E and keeps the other out.
  - DFAs only have finitely many states, so there simply isn't room to keep all these strings separated.
  - So there can't be a DFA for E.
- ***Question:*** Can we generalize this idea?

Distinguishability

- Let L be an arbitrary language over Σ.
- Strings $x \in \Sigma^*$ and $y \in \Sigma^*$ are ***distinguishable relative to L*** when 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, we say that $x \not\equiv_L y$ when

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

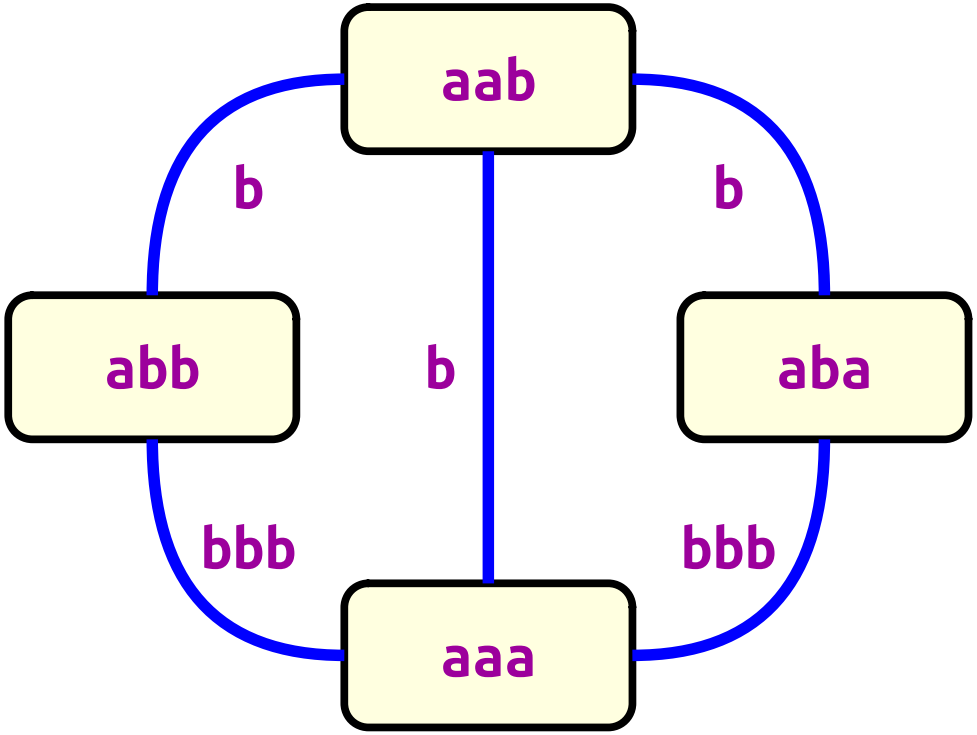
This is how we express exclusive "OR" in propositional logic.

Distinguishability

- Consider the language

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

- There's a collection of strings to the right.
- Which pairs of these strings are distinguishable relative to E? What would you append to distinguish them?
- (Two strings x and y are distinguishable relative to E when there's a string w where ***exactly one*** of xw and yw belongs to E.)

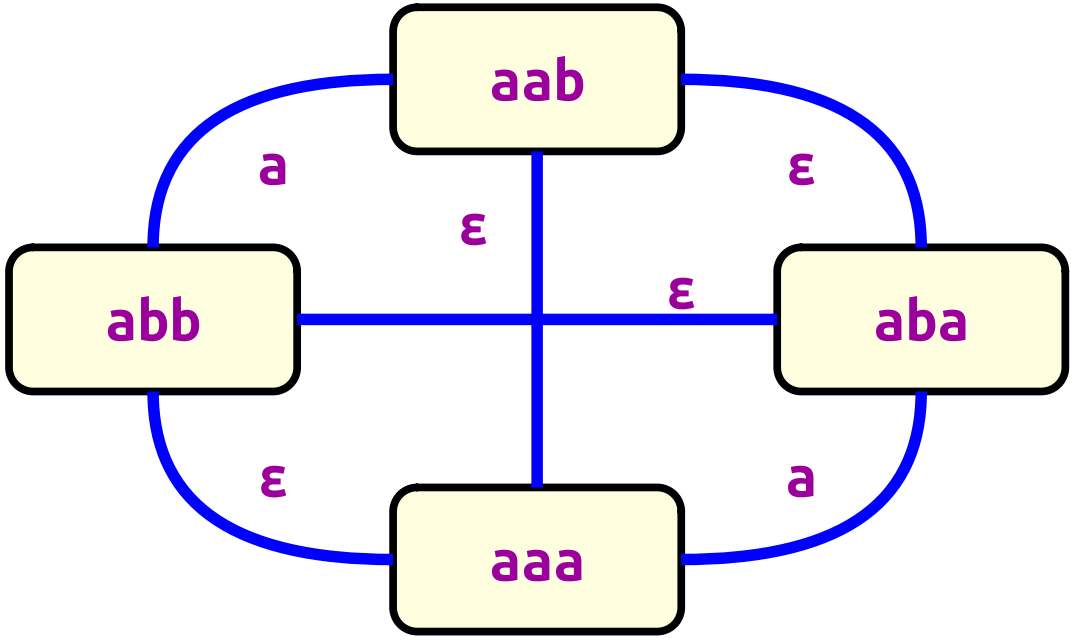


Distinguishability

- A ***palindrome*** is a string that is the same when the characters are read left-to-right and right-to-left.
- Consider the language

$$L = \{ w \in \{\texttt{a}, \texttt{b}\}^* \mid w \text{ is a palindrome} \}$$

- Which pairs of the strings below are distinguishable relative to L? What would you append to distinguish them?

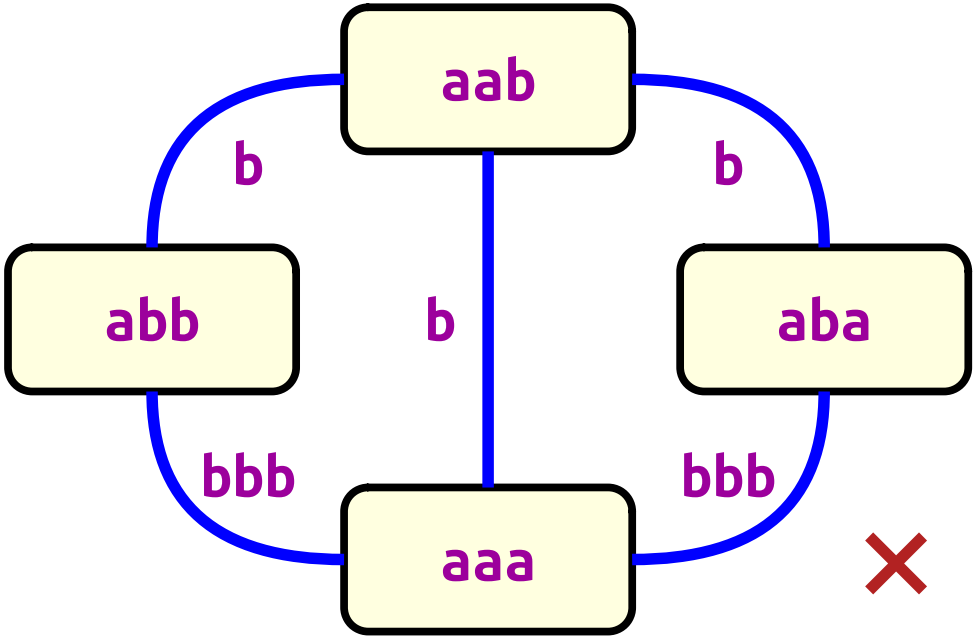


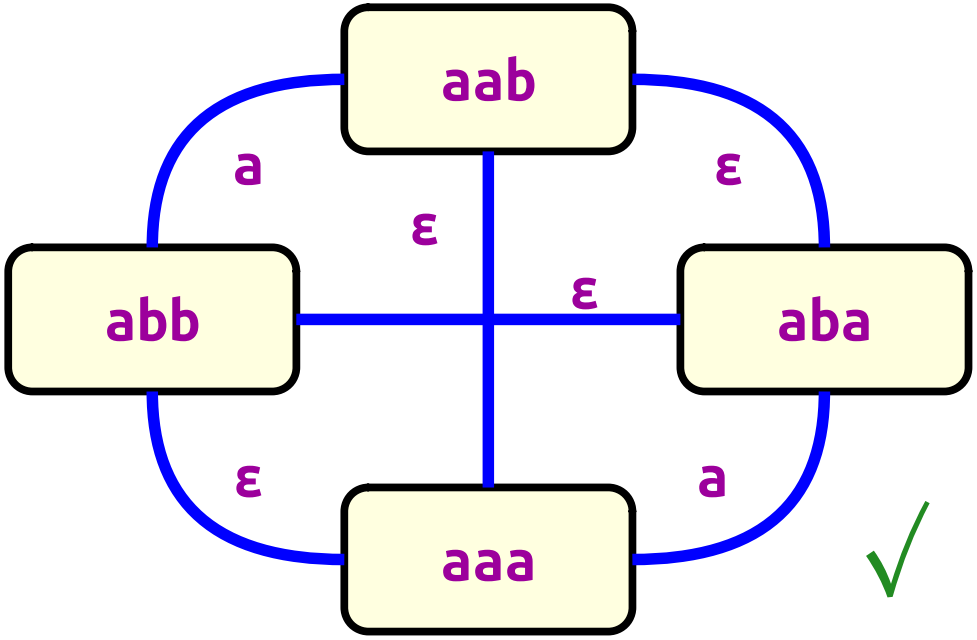
Distinguishing Sets

- Let $L \subseteq \Sigma^*$ be a language. A ***distinguishing set for L*** is set $S \subseteq \Sigma^*$ where the following is true:

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

- In other words, it's a set of strings S where all pairs of distinct strings in S are distinguishable relative to L.





Distinguishing Sets

- Let $E = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$.
- Which of the following are distinguishing sets for E?

$$\{ \varepsilon, \mathtt{a}, \mathtt{ab} \}$$

$$\mathtt{a}^*$$

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

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

Distinguishing Sets

- Let $L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is a palindrome} \}$.
- Which of the following are distinguishing sets for L?

$$\{ \varepsilon, \mathtt{a}, \mathtt{ab} \}$$

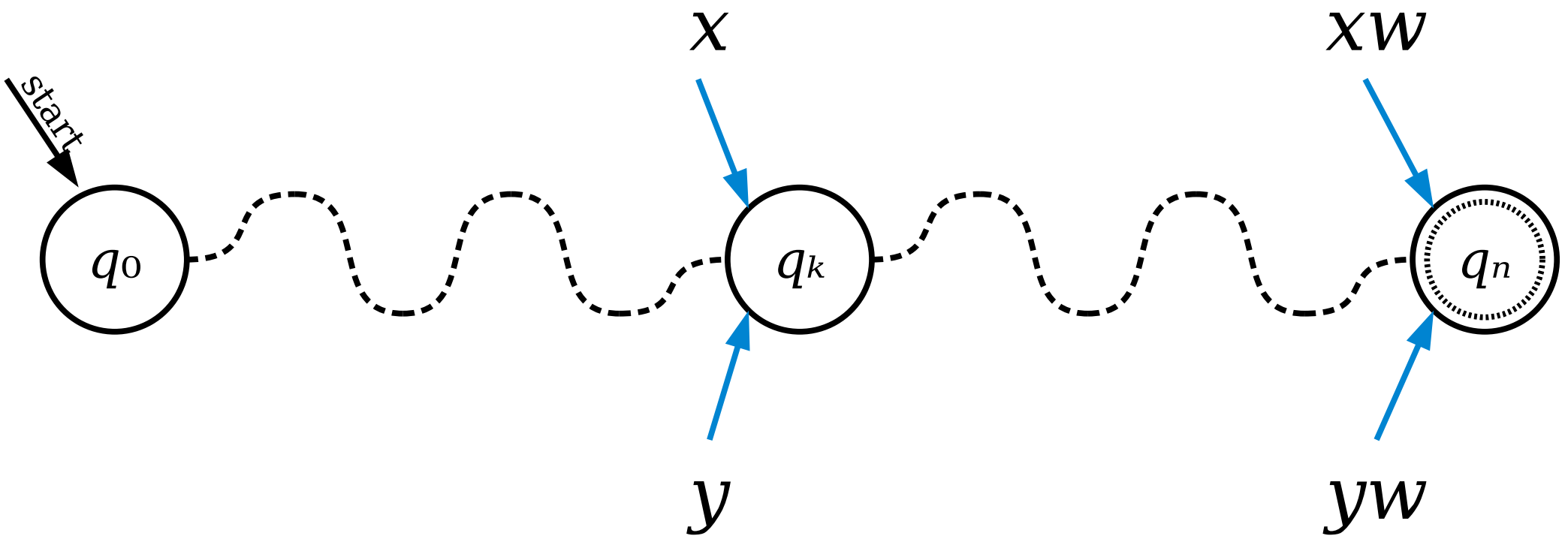
$$\mathtt{a}^*$$

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

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

Distinguishability

- ***Theorem:*** Let L be an arbitrary language over Σ. Let $x \in \Sigma^*$ and $y \in \Sigma^*$ be strings where $x \not\equiv_L y$. If D is a DFA where $\mathscr{L}(D) = L$, then D must end in different states when run on inputs x and y.
- ***Proof sketch:***



Theorem (Myhill-Nerode):

Let L be a language. If L has an infinite distinguishing set, then L is not regular.

(A distinguishing set containing infinitely many strings.)

Theorem: Let L be a language. If L has an infinite distinguishing set, then L is not regular.

Proof: Assume for the sake of contradiction that there is an infinite distinguishing set S for L but that L is regular.

We know L is regular, so there is a DFA D for L. Let k be the number of states in D. Since there are infinitely many strings in S, we can pick $k+1$ distinct strings w_1, w_2, …, and w_{k+1} from S.

Consider what happens when we run D on all those strings. There are only k states in D and we have $k+1$ strings, so by the pigeonhole principle there are strings $w_i \neq w_j$ in S that end in the same state when run through D.

Because $w_i \neq w_j$ and S is a distinguishing set for L, we know that $w_i \not\equiv_L w_j$. As we saw earlier, when we run w_i and w_j through D, they therefore end up in different states. But this is impossible: w_i and w_j 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 $E = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$ is not regular.

Proof: Let $S = \{ \mathtt{a}^n \mid n \in \mathbb{N} \}$. We will prove that S is infinite and that S is a distinguishing set for E.

To see that S is infinite, note that S contains one string for each natural number.

To see that S is a distinguishing set for E, consider any two strings $\mathtt{a}^m, \mathtt{a}^n \in S$ where $m \neq n$. Note that $\mathtt{a}^m\mathtt{b}^m \in E$ and that $\mathtt{a}^n\mathtt{b}^m \notin E$. Therefore, we see that $\mathtt{a}^m \not\equiv_E \mathtt{a}^n$, as required.

Since S is infinite and is a distinguishing set for E, by the Myhill-Nerode theorem we see that E is not regular. ■

Theorem: The language $L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is a palindrome} \}$ is not regular.

Proof: Let $S = \{ \mathtt{a}^n\mathtt{b}^n \mid n \in \mathbb{N} \}$. We will prove that S is infinite and that S is a distinguishing set for L.

To see that S is infinite, note that S contains one string for each natural number.

To see that S is a distinguishing set for L, consider any two strings $\mathtt{a}^m\mathtt{b}^m, \mathtt{a}^n\mathtt{b}^n \in S$ where $m \neq n$. Note that $\mathtt{a}^m\mathtt{b}^m\mathtt{b}^m\mathtt{a}^m \in L$ and that $\mathtt{a}^n\mathtt{b}^n\mathtt{b}^m\mathtt{a}^m \notin L$. Therefore, we see that $\mathtt{a}^m\mathtt{b}^m \not\equiv_L \mathtt{a}^n\mathtt{b}^n$, as required.

Since S is infinite and is a distinguishing set for L, by the Myhill-Nerode theorem we see that L is not regular. ■

Theorem: The language $L = \{ w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is a palindrome} \}$ is not regular.

Proof: Let $S = \{ \mathtt{a}^n \mid n \in \mathbb{N} \}$. We will prove that S is infinite and that S is a distinguishing set for L.

To see that S is infinite, note that S contains one string for each natural number.

To see that S is a distinguishing set for L, consider any two strings $\mathtt{a}^m, \mathtt{a}^n \in S$ where $m \neq n$. Note that $\mathtt{a}^m\mathtt{b}\mathtt{a}^m \in L$ and that $\mathtt{a}^n\mathtt{b}\mathtt{a}^m \notin L$. Therefore, we see that $\mathtt{a}^m \not\equiv_L \mathtt{a}^n$, as required.

Since S is infinite and is a distinguishing set for L, by the Myhill-Nerode theorem we see that L is not regular. ■

Approaching Myhill-Nerode

- The challenge in using the Myhill-Nerode theorem is finding the right set of strings.
- ***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 you have infinitely many strings an that the group of strings is a distinguishing set.
- Check our online "***Guide to the Myhill-Nerode Theorem***" for more details.

Tying Everything Together

- ***Key Intuition for DFAs:*** 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: the states correspond to those pieces of information.
  - If you must remember one of infinitely many pieces of information, your language is not regular: use the information that "must be remembered" to build a distinguishing set for the Myhill-Nerode theorem.
- This can be made rigorous! Take CS154 for details.

Where We Stand

Where We Stand

- We've ended up where we are now by trying to answer the question “what problems can you solve with a computer?”
- We defined a computer to be DFA, which means that the problems we can solve are precisely the regular languages.
- We've discovered several equivalent ways to think about regular languages (DFAs, NFAs, and regular expressions) and used that to reason about the regular languages.
- We now have a powerful intuition for where we ended up: DFAs are finite-memory computers, and regular languages correspond to problems solvable with finite memory.
- Putting all of this together, we have a much deeper sense for what finite memory computation looks like – *and what it doesn't look like!*

Where We're Going

- What does computation look like with unbounded memory?
- What problems can you solve with unbounded-memory computers?
- What does it even mean to “solve” such a problem?
- And how do we know the answers to any of these questions?

Your Action Items

- ***Read "Guide to the Myhill-Nerode Theorem."***
  - It's a useful refresher and deep-dive into all the topics we covered today.
  - And it has worked exercises to give you a sense of how the theorem works!
- ***Finish Problem Set 6.***
  - Slow and steady progress is key here.
  - Come talk to us if you have questions!

Next Time

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