Lecture 6:
The Myhill-Nerode Theorem and Streaming Algorithms
DFA Minimization Theorem:

For every regular language $A$, there is a unique (up to re-labeling of the states) minimal-state DFA $M^*$ such that $A = L(M^*)$.

Furthermore, there is an efficient algorithm which, given any DFA $M$, will output this unique $M^*$.

If such algorithms existed for more general models of computation, that would be an engineering breakthrough!!
How could we show whether two regular expressions are equivalent?

**Claim:** There is an algorithm which given regular expressions $R$ and $R'$, determines whether $L(R) = L(R')$. 
The Myhill-Nerode Theorem:

For every language:

Either there’s a DFA for it

or there’s a set of strings that “trick” every possible DFA
In DFA Minimization, we defined an equivalence relation between states of a DFA. We can also define a similar equivalence relation over *strings* in a *language*:

Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$

$x \equiv_L y$ means: for all $z \in \Sigma^*$, $xz \in L \iff yz \in L$

**Def.** $x$ and $y$ are indistinguishable to $L$ iff $x \equiv_L y$

**Claim:** $\equiv_L$ ("$L$-equivalent") is an equivalence relation

Proof is easy! Just as in previous lecture.
Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$

$x \equiv_L y$ means: for all $z \in \Sigma^*$, $xz \in L \iff yz \in L$

Suppose we partition all strings in $\Sigma^*$ into their equivalence classes under $\equiv_L$

The Myhill-Nerode Theorem:

If the number of parts is finite $\rightarrow$ can construct a DFA!
If the number of parts is infinite $\rightarrow$ there is no DFA!
Mapping strings to DFA states

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$, we define a function $\Delta : \Sigma^* \rightarrow Q$ as follows:

$\Delta(\epsilon) = q_0$

$\Delta(\sigma) = \delta(q_0, \sigma)$

$\Delta(\sigma_1 \cdots \sigma_{k+1}) = \delta(\Delta(\sigma_1 \cdots \sigma_k), \sigma_{k+1})$

$\Delta(w) = \text{the state of } M \text{ reached after reading in } w$

Note: $\Delta(w) \in F \iff M \text{ accepts } w$
Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$

$x \equiv_L y$ means: for all $z \in \Sigma^*$, $xz \in L \iff yz \in L$

The Myhill-Nerode Theorem:

A language $L$ is regular if and only if the number of equivalence classes of $\equiv_L$ is finite.

Proof ($\Rightarrow$) Let $M = (Q, \Sigma, \delta, q_0, F)$ be any DFA for $L$.

Define the relation: $x \approx_M y \iff \Delta(x) = \Delta(y)$

Claim: $\approx_M$ is an equivalence relation with $|Q|$ classes

Claim: If $x \approx_M y$ then $x \equiv_L y$

Proof: $x \approx_M y$ implies for all $z \in \Sigma^*$, $xz$ and $yz$ reach the same state of $M$. So $xz \in L \iff yz \in L$, and $x \equiv_L y$

Corollary: The number of $\equiv_L$ classes is at most the number of $\approx_M$ classes (which is $|Q|$)
Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$

$x \equiv_L y$ means: for all $z \in \Sigma^*$, $xz \in L \iff yz \in L$

**The Myhill-Nerode Theorem:**
A language $L$ is regular *if and only if* the number of equivalence classes of $\equiv_L$ is *finite.*

**Claim:** If $x \approx_M y$ then $x \equiv_L y$

**Corollary:** The number of $\equiv_L$ classes is *at most* the number of $\approx_M$ classes (which is $|Q|$)

**Proof:** Let $S = \{x_1, x_2, \ldots\}$ be distinct strings, one from every $\equiv_L$-equiv class. $|S| = \text{number of } \equiv_L \text{ classes.}$

Thus for all $i \neq j, x_i \not\equiv_L x_j$. By the claim: $x_i \approx_M x_j$.

So each $x_i \in S$ is in a distinct $\approx_M$ equivalence class.

$\Rightarrow$ The number of $\approx_M$ classes is *at least* $|S|.$
Let $L \subseteq \Sigma^*$ and $x, y \in \Sigma^*$

$x \equiv_L y$ means: for all $z \in \Sigma^*$, $xz \in L \Leftrightarrow yz \in L$

($\Leftarrow$) If the number of equivalence classes of $\equiv_L$ is $k$
then there is a DFA for $L$ with $k$ states

**Idea:** Build a DFA whose states are the equivalence classes of $\equiv_L$

Define a DFA $M$ where:

- $Q$ is the set of equivalence classes of $\equiv_L$
- $q_0 = [\varepsilon] = \{ y | y \equiv_L \varepsilon \}$
- for all $x \in \Sigma^*$, $\delta([x], \sigma) = [x \sigma]$ \hspace{1cm} (well-defined??)
- $F = \{ [x] | x \in L \}$

**Claim:** $M$ accepts $x$ if and only if $x \in L$
Define a DFA $M$ where:

- $Q$ is the set of equivalence classes of $\equiv_L$
- $q_0 = [\varepsilon] = \{y | y \equiv_L \varepsilon\}$
- $\delta([x], \sigma) = [x \sigma]$ (well-defined??)
- $F = \{[x] | x \in L\}$

Claim: $M$ accepts $x$ if and only if $x \in L$

Proof: Let $M$ run on $x = x_1 \cdots x_n \in \Sigma^*$, for $x_i \in \Sigma$. $M$ starts in state $[\varepsilon]$, reads $x_1$ and moves to $[x_1]$, reads $x_2$ and moves to $[x_1 x_2]$, ..., and ends in state $[x_1 \cdots x_n]$.

So, $M$ accepts $x_1 \cdots x_n \iff [x_1 \cdots x_n] \in F$

By definition of the set $F$, $[x_1 \cdots x_n] \in F \iff x \in L$
The Myhill-Nerode Theorem gives us a new way to prove that a given language is not regular:

L is not regular if and only if there are infinitely many equiv. classes of $\equiv_L$

L is not regular if and only if there are infinitely many strings $w_1, w_2, \ldots$ so that for all $w_i \neq w_j$, $w_i$ and $w_j$ are distinguishable to L: there is a $z \in \Sigma^*$ such that exactly one of $w_iz$ and $w_jz$ is in L
L is not regular if and only if
There are infinitely many strings $w_1, w_2, \ldots$ so that for all $w_i \neq w_j$, $w_i$ and $w_j$ are distinguishable to $L$.

To prove that L is regular, we have to show that a special finite object (DFA/NFA/regex) exists.

To prove that L is not regular, it is sufficient to show that a special infinite set of strings exists!

We can prove the nonexistence of a DFA/NFA/regex by proving the existence of this special string set!
Using **Myhill-Nerode** to prove non-regularity:

**Theorem:** $L = \{0^n 1^n \mid n \geq 0\}$ is not regular.

**Proof:** Consider the infinite set of strings

$$S = \{0, 00, 000, \ldots, 0^n, \ldots\}$$

Claim: $S$ is a distinguishing set for $L$.

Take any pair $(0^m, 0^n)$ of distinct strings in $S$

Let $z = 1^m$

Then $0^m 1^m$ is in $L$, but $0^n 1^m$ is *not* in $L$

So all pairs of strings in $S$ are distinguishable to $L$

Hence there are infinitely many equivalence classes of $\equiv_L$, and $L$ is not regular!
Streaming Algorithms
Streaming Algorithms

Have three components

Initialize:
  <variables and their assignments>

When next symbol seen is $\sigma$:
  <pseudocode using $\sigma$ and vars>

When stream stops (end of string):
  <accept/reject condition on vars>
  (or: <pseudocode for output>)

Algorithm $A$ computes $L \subseteq \Sigma^*$ if

$A$ accepts the strings in $L$, rejects strings not in $L$
Streaming Algorithms

Streaming algorithms differ from DFAs in several significant ways:

1. Streaming algorithms could output more than one bit
2. The “memory” or “space” of a streaming algorithm can (slowly) increase as it reads longer strings
3. Could also make multiple passes over the input, could be randomized

Can recognize non-regular languages!
L = \{x \mid x\text{ has more }1\text{'s than }0\text{'s}\}

Initialize: C := 0 and B := 0

When next symbol seen is \(\sigma\):

- If (C = 0) then B := \(\sigma\), C := 1
- If (C \neq 0) and (B = \(\sigma\)) then C := C + 1
- If (C \neq 0) and (B \neq \(\sigma\)) then C := C – 1

When stream stops:

- accept if B=1 and C > 0, else reject

B = the majority bit

C = how many more times B appears

On all strings of length n, the algorithm uses \((\log_2 n)+O(1)\) bits of space (to store B and C)
How to think of memory usage

The program is *not considered* as part of the memory

Space usage of A:

$$S(n) = \text{maximum \# of bits needed to store vars in A, over all inputs of length up to } n$$
\[ L = \{0^n1^n \mid n \geq 0\} \]

Initialize: \( z := 0, s := \text{false}, \text{fail} := \text{false} \)

When next symbol seen is \( \sigma \):

1. If (not \( s \)) and (\( \sigma = 0 \)) then \( z := z + 1 \)
2. If (not \( s \)) and (\( \sigma = 1 \)) then \( s := \text{true}; z := z - 1 \)
3. If (\( s \)) and (\( \sigma = 0 \)) then \( \text{fail} := \text{true} \)
4. If (\( s \)) and (\( \sigma = 1 \)) then \( z := z - 1 \)

When stream stops:

\textit{accept} if and only if (not \text{fail}) and (\( z = 0 \))

\( z \) = how many more times 0 appears than 1
\( s \) = “Started reading 1s yet?”
\( \text{fail} \) = “Reject for certain?”

On all strings of length \( n \), uses \((\log_2 n) + O(1)\) space
DFAs and Streaming

**Thm:** Let \( L' \) be recognized by DFA \( M \) with \( \leq 2^p \) states. Then \( L' \) is computable by a streaming algorithm \( A \) using \( \leq p \) bits of space.

**Proof Idea:** Define algorithm \( A \) as follows.

**Initialize:** Encode the *start state of \( M \)* in memory.

**When next symbol seen is \( \sigma \):**

*Update state of \( M \) using \( M \)'s transition function*

**When stream stops:**

*Accept if current state of \( M \) is final, else reject*
DFAs and Streaming

For any $L \subseteq \Sigma^*$ define $L_n = \{ x \in L \mid |x| \leq n \}$

**Theorem:** Let $L'$ be computable by streaming algorithm $A$ using $\leq S(n)$ bits of space on all strings of length up to $n$.
Then for all $n$, there is a DFA $M$ with $< 2^{S(n)+1}$ states such that $L'_n = L(M)_n$

That is, for all streaming algorithms $A$ using $S(n)$ space, there’s a DFA $M$ of $< 2^{S(n)+1}$ states such that $A$ and $M$ agree on all strings of length up to $n$. 
Theorem: Let $L'$ be computable by streaming algorithm $A$ using $\leq S(n)$ bits of space on all strings of length up to $n$. Then for all $n$, there is a DFA $M$ with $< 2^{S(n)+1}$ states such that $L'_n = L(M)_n$

Proof Idea: States of $M = \text{all } 2^{S(n)+1} - 1 \text{ possible memory configurations of } A, \text{ over strings of length up to } n$
Start state of $M = \text{Initialized memory of } A$
Transition function = Mimic how $A$ updates its memory
Final states of $M = \text{Subset of memory configurations in which } A \text{ would accept, if the string ended there}$

For any $L \subseteq \Sigma^*$ define $L_n = \{x \in L \mid |x| \leq n\}$
Initialize: $C := 0$ and $B := 0$

When the next symbol $\sigma$ is read,
If $(C = 0)$ then $B := \sigma$, $C := 1$
If $(C \neq 0)$ and $(B = \sigma)$ then $C := C + 1$
If $(C \neq 0)$ and $(B \neq \sigma)$ then $C := C - 1$

When the stream stops,
accept if $B=1$ and $C > 0$, else reject

Example: $L = \{x \mid x$ has more 1's than 0's$\}$

Example: A 6-state DFA that agrees with $L$ on all strings of length $\leq 3$
Streaming Lower Bounds via DFAs

For any $L \subseteq \Sigma^*$ define $L_n = \{x \in L \mid |x| \leq n\}$

**Theorem:** Let $L'$ be computable by streaming algorithm $A$ using $S(n)$ bits of space on all strings of length up to $n$.
Then for all $n$, there is a DFA $M$ with $< 2^{S(n)+1}$ states such that $L'_n = L(M)_n$.

**Corollary:** Suppose every DFA $M$ such that $L'_n = L(M)_n$ requires at least $Q(n) := 2^{S(n)+1}$ states. Then $L'$ is not computable by a streaming algorithm using $S(n) = \log_2(Q(n)/2) = \log_2(Q(n))-1$ space! That is, $L'$ requires at least $\log_2(Q(n))$ space.
L = \{x | x \text{ has more 1’s than 0’s}\}

Is there a streaming algorithm for L using much \textit{less than} \((\log_2 n)\) space?

\textbf{Theorem:} Every streaming algorithm for L needs at least \((\log_2 n)\) bits of space over all strings of length up to n.

We will use:

• Myhill-Nerode Theorem
• The connection between DFAs and streaming
L = \{ x \mid x \text{ has more 1's than 0's}\}

**Theorem:** Every streaming algorithm for L requires at least $(\log_2 n)$ bits of space.

**Proof Idea:** Let $n$ be even, and $L_n = \{ x \in L \mid |x| \leq n \}$

We will give a set $T_n$ of $n$ strings such that each pair in $T_n$ is distinguishable to $L_n$.

By the Myhill-Nerode Thm

- Every DFA for $L_n$ needs at least $n$ states.
- Every streaming algorithm for $L$ needs at least $(\log n)$ space on strings of length $\leq n$. 

$L = \{x \mid x \text{ has more 1's than 0's}\}$

**Theorem:** Every streaming algorithm for $L$ requires at least $(\log_2 n)$ bits of space

Suppose we partition all strings into their equivalence classes under $\equiv_{L_n}$

Construct $T_n$

The number of states in a DFA recognizing $L_n$ is at least the number of equivalence classes under $\equiv_{L_n}$
L = \{x \mid x \text{ has more 1's than 0's}\}

**Theorem:** Every streaming algorithm for L requires at least \((\log_2 n)\) bits of space

**Proof (Slide 1):** Let \(T_n = \{0^i, 1^i \mid i = 1, \ldots, n/2\}\)

**Claim:** \(T_n\) is distinguishable to \(L_n\)

**Case 1:** Let \(x = 0^a\) and \(y = 1^b\) be any strings from \(T_n\)

**Claim:** \(z = \varepsilon\) distinguishes \(x\) and \(y\) in \(L_n\)

\(xz\) has more 0s than 1s \(\Rightarrow xz \not\in L_n\)

\(yz\) has length \(\leq n\) and more 1s than 0s \(\Rightarrow yz \in L_n\)

So the string \(z\) distinguishes \(x\) and \(y\), and \(x \not\sim_{L_n} y\)
\[ L = \{ x \mid x \text{ has more 1's than 0's} \} \]

**Theorem:** Every streaming algorithm for \( L \) requires at least \((\log_2 n)\) bits of space

**Proof (Slide 2):** Let \( T_n = \{0^i, 1^i \mid i = 1, \ldots, n/2\} \)

**Claim:** \( T_n \) is distinguishable to \( L_n \)

**Case 2:** Let \( x=0^a \) and \( y=0^b \) be from \( T_n \), with \( a < b \)

**Claim:** \( z = 1^b \) distinguishes \( x \) and \( y \) in \( L_n \)

\( xz \) has length \( \leq n \), \( a \) 0's and \( b \) 1's \( \Rightarrow xz \in L_n \)

\( yz \) has \( b \) zeroes and \( b \) ones \( \Rightarrow yz \notin L_n \)

So the string \( z \) distinguishes \( x \) and \( y \), and \( x \not\equiv_{L_n} y \)
$L = \{ x \mid x \text{ has more } 1\text{'s than } 0\text{'s} \}$

**Theorem:** Every streaming algorithm for $L$ requires at least $(\log_2 n)$ bits of space

**Proof (Slide 3):** Let $T_n = \{0^i, 1^i \mid i = 1, \ldots, n/2\}$

**Claim:** $T_n$ is *distinguishable* to $L_n$

**Case 3:** Let $x = 1^a$ and $y = 1^b$ be from $T_n$, with $a < b$

**Claim:** $z = 0^a$ distinguishes $x$ and $y$ in $L_n$

$xz$ has $a$ ones and $a$ zeroes $\Rightarrow xz \notin L_n$

$yz$ has length $\leq n$, $a$ zeroes and $b$ ones $\Rightarrow yz \in L_n$

So the string $z$ distinguishes $x$ and $y$, and $x \not\equiv_{L_n} y$
Let $L = \{x \mid x \text{ has more 1's than 0's}\}$

**Theorem:** Every streaming algorithm for $L$ requires at least $(\log_2 n)$ bits of space

**Proof (Slide 4):** Let $T_n = \{0^i, 1^i \mid i = 1, \ldots, n/2\}$

All pairs of strings in $T_n$ are distinguishable to $L_n$

$\Rightarrow$ There are at least $|T_n|$ equiv classes of $\equiv_{L_n}$

By the Myhill-Nerode Theorem:

$\Rightarrow$ All DFAs recognizing $L_n$ need $\geq |T_n|$ states

$\Rightarrow$ Every streaming algorithm for $L$ needs at least $S(n) = (\log_2 |T_n|)$ bits of space.

Finally, note $|T_n| = n$ and we’re done!