תן לי הכוח לשנת את מה שאיני יכול לשנות, להבל אינתי מאינים
יוכל לשנות, ואת החכמה להבדיל בין שנייה.
Non-regular languages: two approaches

1. Pumping Lemma
2. Myhill-Nerode Theorem (not in Sipser’s book)

Closure properties

Algorithmic questions for NFAs

Sipser, 1.4, 2.1, 2.2

Hopcroft and Ullman, 3.4
Theorem 1

A language is described by a regular expression, iff it is regular.

We have made a lot of progress understanding what finite automata can do, but what they cannot do?
Negative results

Is there a DFA that accepting the following languages (over \{0, 1\}).

- \(B = \{0^n1^n : n \geq 0\}\)
- \(C = \{w : \#_1(w) = \#_0(w)\}\)
- \(D = \{w : \#_{01}(w) = \#_{10}(w)\}\)

\(#_s(w)\) – the number of times \(s\) appears in \(w\).

Consider \(B\):

- DFA must “remember” how many 0’s it has seen
- Impossible with finite state.

The others languages seem to be exactly the same...

**Question:** Is this a proof?

**Answer:** No, \(D\) is regular.....
Part I

Pumping Lemma
Regular languages can be pumped

For every regular language \( \mathcal{L} \), there exists \( \ell > 0 \), the pumping length, s.t.: Every \( s \in \mathcal{L} \) longer than \( \ell \), can be “pumped” into a longer string in \( \mathcal{L} \).

This is a powerful technique for showing that a language is not regular.
The Pumping Lemma

**Lemma 2**

For every **regular** language $\mathcal{L}$, exists $\ell > 0$ (i.e., the **pumping length**) s.t.:

every $s \in \mathcal{L}$ with $|s| \geq \ell$, can be written as $s = xyz$ with

1. $xy^i z \in \mathcal{L}$ for every $i \geq 0$,
2. $|y| > 0$, and
3. $|xy| \leq \ell$.

**Remarks:** Without the second condition, the theorem would be trivial.

The third condition is technical and sometimes useful.
Proving the Pumping Lemma

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA accepting $L$, and let $\ell = |Q|$.

Let $s \in L$ be with $|s| \geq \ell$, and consider the sequence of states $M$ traverse as it reads $s = s_1 \ldots s_n$:

```
\begin{array}{cccccccccc}
  & s_1 & & s_2 & & s_3 & & s_4 & & s_5 & & s_6 & & \ldots & & s_n \\
\uparrow & q_1 & \uparrow & q_20 & \uparrow & q_9 & \uparrow & q_{17} & \uparrow & q_{12} & \uparrow & q_{13} & \uparrow & q_9 & \uparrow & q_2 & \uparrow & q_5 \in F \\
\end{array}
```

By the pigeonhole principle, at least one of the states in the above sequence repeats.

$s = xyz$
By inspection, \( M \) accepts \( xy^k z \) for every \( k \geq 0 \).

\(|y| > 0\), because the state \( q_9 \) is repeated.

To ensure that \(|xy| \leq \ell\), pick first state repetition, which must occur no later than \( \ell + 1 \) states in sequence.
Corollary 3

\[ B = \{0^n1^n : n > 0\} \text{ is not regular.} \]

Proof: By contradiction. Suppose \( B \) is regular and let \( \ell \) be its pumping length.

- Consider the string \( s = 0^\ell 1^{\ell} \in B \).
- Let \( x, y, z \) be (one possible) strings guaranteed by the pumping lemma (i.e., \( s = xyz \))
  1. \( xy^i z \in B \) for every \( i \geq 0 \),
  2. \( |y| > 0 \), and
  3. \( |xy| \leq \ell \).
- If \( y \) is all 0, then \( xy^2 z \) has too many 0’s.
- If \( y \) is all 1, then \( xy^2 z \) has too many 1’s.
- If \( y \) is mixed, then \( xy^2 z \) is not of right form.

We did not use the third property.
Corollary 4

\( C = \{ w : \#_1(w) = \#_0(w) \} \) is not regular.

Proof: By contradiction. Suppose \( C \) is regular. Let \( \ell \) be the pumping length.

- Consider the string \( s = 0^\ell 1^\ell \in C \).
- Let \( x, y, z \) be (possible) set of strings guaranteed by the pumping lemma (i.e., \( s = xyz \))
  1. \( xy^iz \in B \) for every \( k \geq 0 \),
  2. \( |y| > 0 \), and
  3. \( |xy| \leq \ell \).

- Since \( |xy| \leq \ell \), the string \( y \) is all 0’s.
- Thus, \( xy^2z \notin C \) (more 0’s than 1’s).

Could we have used \( s = (01)^\ell \)?
Corollary 5

\[ \mathcal{E} = \{0^i1^j : i > j\} \text{ is not regular.} \]

Proof: By contradiction. Suppose \( \mathcal{E} \) is regular. Let \( \ell \) be its pumping length.

- Consider the string \( s = 0^\ell 1^{\ell-1} \in \mathcal{E} \).
- By pumping lemma, \( s = xyz \), where \( xy^kz \in \mathcal{E} \) for every \( k \geq 0 \), \( |y| > 0 \) and \( |xy| \leq \ell \).
- But \( xy^0z = xz \notin \mathcal{E} \) \hspace{1cm} (at least as much 1’s as 0’s) \( \square \)
Corollary 6

The language Primes $\subseteq 1^*$ – all strings whose length is a prime number – is not regular.

Proof: Suppose Primes is regular, and let $\ell$ be its pumping length.

- Let $s = 1^p \in \text{Primes}$, where $p \geq \ell$ is a prime (?).
- By pumping lemma, $s = xyz$, where $xy^kz \in \text{Primes}$ for every $k \geq 0$.
- Let $|y| = m$. Hence, $xy^{p+1}z = 1^{p+mp} \in \text{Primes}$ but $p(m + 1)$ is not prime...
Another Example

Consider the language \( \mathcal{L} = \{a^i b^n c^n : n \geq 0, i \geq 1\} \cup \{b^n c^m : n, m \geq 0\} \).

Any non-empty \( s \in \mathcal{L} \) can be pumped:

1. If \( s = a^i b^n c^n \) with \( i > 0 \), then set \( x = \varepsilon \) and \( y = a \).
2. If \( s = b^n c^m \) with \( n > 0 \), then set \( x = \varepsilon \) and \( y = b \).
3. If \( s = c^m \) with \( m > 0 \), then set \( x = \varepsilon \) and \( y = c \).

(in all cases \( z \) is set to the right suffix).

- Is \( \mathcal{L} \) regular? No
- How can we prove it?
Part II

Characterization of Regular Languages
The equivalence relation \( \sim \)

**Definition 7**

For \( \mathcal{L} \subseteq \Sigma^* \), define the equivalence relation \( \sim \) over words in \( \Sigma^* \), by

\[
x \sim y \text{ if for every } z \in \Sigma^*, \text{ it holds that } xz \in \mathcal{L} \iff yz \in \mathcal{L}.
\]

Easy to see that \( \sim \) is indeed an equivalence relation (reflexive, symmetric, transitive) on \( \Sigma^* \).

Hence, \( \sim \) partitions \( \Sigma^* \) into equivalence classes.

For \( x \in \Sigma^* \), let \([x] \subseteq \Sigma^* \) denote its equivalence class with respect to \( \sim \).

How many equivalence classes does \( \sim \) induce? **finite** or **infinite**?

Could be either (depends on \( \mathcal{L} \)).
Three examples

- $\mathcal{L}_1 = \{ w : \#_1(w) \mod 5 = 0 \}$

  $\mathcal{L}_1 \sim$ has finitely many equivalence classes.

  The equivalent classes are: $[\varepsilon], [1], [11], [111], [1111]$.

  Proof:

  - Classes cover $\{0, 1\}^*$: for any $x \in \{0, 1\}^*$: $x \mathcal{L} 1 \#_1(x) \mod 5$.
    
    $xz \in \mathcal{L} \iff \#_1(xz) \mod 5 = 0 \iff (\#_1(x) \mod 5) + \#_1(z) \mod 5 = 0 \iff 1 \#_1(x) \mod 5$ $z \in \mathcal{L}$.

  - Classes are disjoint: $1^i \not\mathcal{L} 1^j$ for $i \neq j \in \{0, 1, 2, 3, 4\}$

- $\mathcal{L}_2 = \{ 0^n1^n : n \in \mathbb{N} \}$

  $\mathcal{L}_2 \sim$ has infinitely many equivalence classes.

  $[0] \neq [0^2] \neq [0^3] \ldots$

- $\mathcal{L}_3 = \{ a^i b^n c^n : n \geq 0, i \geq 1 \} \cup \{ b^n c^m : n, m \geq 0 \}$

  $\mathcal{L}_3 \sim$ has infinitely many equivalence classes.

  $[ab] \neq [ab^2] \neq [ab^3] \neq \ldots$
Myhill-Nerode Theorem

Theorem 8 (Myhill-Nerode Theorem)

\( \mathcal{L} \subseteq \Sigma^* \) is regular iff \( \sim \) finitely many equivalence classes.

Hence

- \( \mathcal{L}_1 = \{ w \in \{0, 1\}^* : \#_1(w) \mod 5 = 0 \} \) is regular.
- \( \mathcal{L}_2 = \{ 0^n1^n : n \in \mathbb{N} \} \) is not regular.
- \( \mathcal{L}_3 = \{ a^i b^n c^n : n \geq 0, i \geq 1 \} \cup \{ b^n c^m : n, m \geq 0 \} \) is not regular.
**Fact 9 (right invariance)**

If $x \sim y$, then $xw \sim yw$ for every $w \in \Sigma^*$

Proof: $(xw)z \in \mathcal{L} \iff x(wz) \in \mathcal{L} \iff y(wz) \in \mathcal{L} \iff (yw)z \in \mathcal{L}$
Proving Myhill-Nerode Theorem

Let \( \mathcal{L} \) be a regular language and let \( M = (Q, \Sigma, \delta, q_0, F) \) be a DFA accepting it.

- Define the binary relation \( \sim_M \) by \( x \sim_M y \) if \( \widehat{\delta}(q_0, x) = \widehat{\delta}(q_0, y) \).
- \( \sim_M \) is an equivalence relation.
- \( x \sim_M y \implies xz \sim_M yz \) for every \( z \in \Sigma^* \).
  \[ \implies xz \in \mathcal{L} \iff yz \in \mathcal{L}. \]
- Hence, \( x \sim_M y \implies x \sim \mathcal{L} y \).
- Each equivalence class of \( \sim_M \) corresponds to union of classes of \( \sim_M \).
  Namely, \( \sim_M \) is a refinement of \( \sim_M \). (see drawing on board)

- Specifically, \( \# \) of equivalence classes of \( \sim_M \) is less or equal than \( \# \) of equivalence classes of \( \sim_M \).
- \( \sim_M \) has finitely many equivalence classes. (?)
- Therefore, \( \sim_M \) has finitely many equivalence classes.
Proving Myhill-Nerode theorem \(\iff\)

Assume \(\mathcal{L}\) has finitely many equivalence classes and let \(x_1, \ldots, x_n \in \Sigma^*\) be their representatives.

We’ll construct a DFA \(M = (Q, \Sigma, \delta, q_0, F)\) that accepts \(\mathcal{L}\).

For \(y \in \Sigma^*\), let \(C(y)\) be the index \(i \in \{1, \ldots, n\}\) with \(y \in [x_i]\).

- \(Q = \{1, \ldots, n\}\).
- \(\delta(i, \sigma) = C(x_i\sigma)\).
- \(q_0 = C(\varepsilon)\).
- \(F = \{i: x_i \in \mathcal{L}\}\).

Claim. For any \(y \in \Sigma^*\): \(\hat{\delta}(q_0, y) = C(y)\).

Hence, \(y \in \mathcal{L}(M) \iff \hat{\delta}(q_0, y) \in F \iff C(y) \in F \iff x_{C(y)} \in \mathcal{L} \iff y \in \mathcal{L}\). \(\square\)

This is \(the\) optimal DFA, number of states wise, for \(\mathcal{L}\). (?)

Proof: (of claim) By induction on word length. Base case: by definition.

- Let \(y = w\sigma \in \Sigma^*\), and assume \(w \in [x_i]\) and \(y \in [x_j]\).
- \(\delta(i, \sigma) := C(x_i\sigma) = \) (by right invariance) \(C(w\sigma) = C(y) = j\).

\(\implies\) \(\hat{\delta}(q_0, w\sigma) = \hat{\delta}(\hat{\delta}(q_0, w), \sigma) = \) (by i.h) \(\delta(i, \sigma) = j\).
Example

Construct a DFA for \( \{ w \in \{0, 1\}^\ast : \#_1(w) \equiv 0 \pmod{5} \} \), via the latter proof method.

- Equivalent class representatives: \( \{ \varepsilon, 1, 11, 111, 1111 \} \)
- \( Q = \{0, 1, 2, 3, 4\} \)
- \( q_0 = 0 \)
- \( F = \{0\} \)
- \( \delta(i, 0) = C(1^i 0) = i \) and \( \delta(i, 1) = C(1^i 1) = i + 1 \pmod{5} \)
Part III

Closure Properties of Regular Languages
Simple closure properties

- Regular languages are closed under complement.
  1. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts $L$.
  2. Then $M' = (Q, \Sigma, \delta, q_0, Q \setminus F)$ is a DFA that accepts $\overline{L} = \Sigma^* \setminus L$.
  3. NFA ?!

- Regular languages are closed under intersection.
  1. $L_1 \cap L_2 = \overline{L_1 \cup L_2}$.
  2. Proof with automata ?
Division

For languages $\mathcal{L}_1, \mathcal{L}_2 \subseteq \Sigma^*$, define

$$\mathcal{L}_1 / \mathcal{L}_2 = \{x \in \Sigma^* : \exists y \in \mathcal{L}_2, xy \in \mathcal{L}_1\}$$

Examples:

- $\mathcal{L}_1 = \{abc, dec, gg\}$ and $\mathcal{L}_2 = \{c\}$. Then $\mathcal{L}_1 / \mathcal{L}_2 = \{ab, de\}$
- $\mathcal{L}_1 = \mathcal{L}(01 \cup 1)^*$ and $\mathcal{L}_2 = \mathcal{L}(00)$. Then $\mathcal{L}_1 / \mathcal{L}_2 = \emptyset$
- $\mathcal{L}_3 = \mathcal{L}(a^*b^*c^*)$ and $\mathcal{L}_4 = \mathcal{L}(b)$. Then $\mathcal{L}_3 / \mathcal{L}_4 = \mathcal{L}(a^*b^*)$
Closure under division

Recall, \( L_1/L_2 = \{ x : \exists y \in L_2, xy \in L_1 \} \)

**Theorem 10**

Regular languages are closed under division with any language: 
\( L_1 \) is regular \( \implies L_1/L_2 \) is regular.

**Proof:** Let \( L_1 \) be a regular language and let \( L_2 \) be an arbitrary language.

1. \( \sim \) is a refinement of \( L_1/L_2 \). Proof: Assume \( x \sim y \). For \( z \in \Sigma^* \):
   
   \[ xz \in L_1/L_2 \iff xzw \in L_1 \iff yzw \in L_1 \iff yz \in L_1/L_2. \implies x \sim y. \]

2. Hence, \( L_1/L_2 \) has finite number of equivalent states, and thus regular.

Another proof.

1. Let \( M = (Q, \Sigma, \delta, q_0, F) \) be a DFA for \( L_1 \).
2. Let \( F' = \{ q \in Q : \exists y \in L_2, \hat{\delta}(q, y) \in F \} \)
3. The DFA \( M' = (Q, \Sigma, \delta, q_0, F') \) accepts \( L_1/L_2 \).

\( F' \) is well defined, but might be hard to compute – “non constructive proof”.

Iftach Haitner (TAU)
Computational Models, Lecture 3
October 29, 2018 27 / 46
Homomorphism

**Definition 11 (Homomorphism)**

An **homomorphism** from alphabet $\Delta$ to **words** over alphabet $\Sigma$, is a function $h: \Delta \mapsto \Sigma^*$.  
For $w \in \Delta^*$, let $h(w = w_1, \ldots, w_n) = h(w_1) \cdots h(w_n)$.  
For $L \subseteq \Delta^*$, let $h(L) = \{h(w): w \in L\}$.

By definition, $h(\varepsilon) = \varepsilon$ and $h(\emptyset) = \emptyset$.

**Examples:**

- Let $h: \{0, 1\} \mapsto \{a, b\}^*$ be defined by $h(1) = aba$ and $h(0) = aa$.  
  $h(010) = aa \, aba \, aa$. For $L_1 = (01)^*$, $h(L_1) = (aaaba)^*$.

- Let $h(0) = a$, $h(1) = a$. For $L_2 = \{0^n1^n: n \geq 0\}$, $h(L_2) = \{a^{2n}: n \geq 0\}$. 
Closure under homomorphism

**Theorem 12**

*Regular languages are closed under homomorphism.*

**Proof idea:** Using regular expressions.

Let $\mathcal{L} \subseteq \Delta^*$ be regular language and let $h : \Delta \mapsto \Sigma^*$.

1. $h(\emptyset) = \emptyset$, $h(\{\varepsilon\}) = \{\varepsilon\}$, and $h(\{a\}) = \{h(a)\}$ for any $a \in \Delta$.
2. $h(\mathcal{L}_1 \cup \mathcal{L}_2) = h(\mathcal{L}_1) \cup h(\mathcal{L}_2)$, $h(\mathcal{L}_1 \parallel \mathcal{L}_2) = h(\mathcal{L}_1) \parallel h(\mathcal{L}_2)$ and $h(\mathcal{L}^*) = h(\mathcal{L})^*$

Let $R$ be a regular expression with $\mathcal{L} = \mathcal{L}(R)$. The proof is by induction on $|R|$.

- $|R| = 1$. By Item (1), $h(\mathcal{L}) = h(\mathcal{L}(R))$ is regular.
- $|R| > 1$. Assume $R = (R_1 \cup R_2)$ (other cases are similar).
  - By item (2), $h(\mathcal{L}) = h(\mathcal{L}(R_1) \cup \mathcal{L}(R_2)) = h(\mathcal{L}(R_1)) \cup h(\mathcal{L}(R_2))$.
  - By i.h., $h(\mathcal{L}(R_1))$ and $h(\mathcal{L}(R_2))$ are regular.

Thus, $h(\mathcal{L})$ is regular. □
Closure under homomorphism, proof using Automata

Let \( M = (Q, \Delta, \delta, q_0, F) \) be a DFA for \( \mathcal{L} \).

Define NFA \( N = (Q', \Sigma, \delta', q_0, F) \) for \( h(\mathcal{L}) \) as follows:

- if \( \delta(q_i, \sigma) = q_j \) and \( h(\sigma) = w_1, \ldots, w_t \), then \( \delta'(d_{i\sigma}^k, w_i) = d_{i\sigma}^{k+1} \) for all \( k \in \{1, \ldots, t-1\} \), letting \( d_{i\sigma}^1 = q_i \) and \( d_{i\sigma}^t = q_j \).
- \( Q' \) includes \( Q \) and all new states.

Claim 13

\( \mathcal{L}(N) = h(\mathcal{L}) \).

Proof idea:

- For \( h(\mathcal{L}) \subseteq \mathcal{L}(N) \):
  \[ w \in \mathcal{L} \implies \exists r_1, \ldots, r_{|w|+1} \text{ s.t. } r_1 = q_0, r_{|w|+1} \in F \text{ and } r_{i+1} = \delta(r_i, w_{i+1}). \ldots \implies h(w) \in \mathcal{L}(N). \]

- For \( \mathcal{L}(N) \subseteq h(\mathcal{L}) \):
  \[ w \in \mathcal{L}(N) \implies \exists r_1, \ldots, r_{|w|+1} \text{ s.t. } r_1 = q_0, r_{|w|+1} \in F \text{ and } r_{i+1} \in \delta'(r_i, w_{i+1}). \ldots \implies \exists w' \in \mathcal{L} \text{ with } h(w') = w. \]
Inverse homomorphism

**Definition 14 (Inverse homomorphism)**

For homomorphism \( h : \Delta \mapsto \Sigma^* \), define its inverse homomorphism \( h^{-1} : \Sigma^* \mapsto P(\Delta^*) \), by \( h^{-1}(w) = \{ x \in \Delta^* : h(x) = w \} \).

For \( \mathcal{L} \subseteq \Sigma^* \), let \( h^{-1}(\mathcal{L}) = \bigcup_{x \in \mathcal{L}} h^{-1}(x) = \{ x \in \Delta^* : h(x) \in \mathcal{L} \} \)

▶ Example: \( h(0) = a, h(1) = b \) and \( h(2) = a \). Then \( h^{-1}(\{a^n ba^n : n \geq 0\}) = \{\{0, 2\}^n 1\{0, 2\}^n : n \geq 0\} \).

For any \( h : \Delta \mapsto \Sigma^* : \)

**Claim 15**

\( h(h^{-1}(\mathcal{L})) \subseteq \mathcal{L} \), for any \( \mathcal{L} \subseteq \Sigma^* \)

Proof: Immediate.

**Claim 16**

\( \mathcal{L} \subseteq h^{-1}(h(\mathcal{L})) \), for any \( \mathcal{L} \subseteq \Delta^* \)

Proof: Holds since \( w \in h^{-1}(h(w)) \) for any \( w \in \Delta^* \)
Closure under inverse homomorphism

**Theorem 17**

*Regular languages are closed under inverse homomorphism.*

**Proof idea:** Let \( \mathcal{L} \) be a regular language, let \( M \) be a DFA for \( \mathcal{L} \) and let \( h: \Delta \mapsto \Sigma^* \).

- \( w \in h^{-1}(\mathcal{L}) \iff h(w) \in \mathcal{L}(M) \).
- Hence, to decide \( w \) simply emulate \( M(h(w)) \).

**Algorithm 18 (DFA for \( h^{-1}(\mathcal{L}) \), Informal)**

On input \( w \):

1. Initialized a “buffer” \( \text{Buff} \) to \( h(a) \), where \( a \) is the first letter of \( w \).
2. Emulate a running of \( M \) with \( \text{Buff} \) as its input string.
   - Each time \( \text{Buff} \) is fully read by \( M \), set \( \text{Buff} = h(a) \), where \( a \) is the next letter in \( w \) (if exists).
3. **Accept** iff \( M \) does

How do we implement \( \text{Buff} \)?
Closure under inverse homomorphism, the DFA definition

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$ and homomorphism $h: \Delta \rightarrow \Sigma^*$, define NFA $N = (Q', \Delta, \delta', q'_0, F')$.

1. Let $k = \max_{a \in \Delta} |h(a)|$ and $\tilde{\Sigma} = \bigcup_{0 \leq i \leq k} \Sigma^i$
2. $Q' = Q \times \tilde{\Sigma}$
3. $q'_0 = [q_0, \varepsilon]$
4. $F' = F \times \{\varepsilon\}$
5. $\delta'$ is defined as follows:
   - $\delta'([q, \varepsilon], a) = \{([q, h(a)])\}$
   - $\delta'([q, ax], \varepsilon) = ([\delta(q, a), x])$

It is not hard to prove that $L(N) = h^{-1}(L(M))$, but we show it using a simpler approach.
Closure under inverse homomorphism, DFA definition take 2

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$ and homomorphism $h: \Delta \to \Sigma^*$, define DFA $M' = (Q, \Delta, \delta', q_0, F)$ by

$$\delta'(q, a) = \widehat{\delta}(q, h(a))$$

Easy to verify that $\widehat{\delta'}(q, w) = \widehat{\delta}_M(q, h(w))$

Hence, $w \in \mathcal{L}(M') \iff h(w) \in \mathcal{L}(M) \iff w \in h^{-1}(\mathcal{L})$
Part IV

Algorithmic Questions for NFAs
Algorithmic Questions for NFAs

Q: Given an NFA, \( N \), and a string \( w \), is \( w \in \mathcal{L}(N) \)?

Answer: Construct the DFA equivalent to \( N \) and run it on \( w \).

Q: Is \( \mathcal{L}(N) = \emptyset \)?

Answer: This is a reachability question in graphs: Is there a path in the states’ graph of \( N \) from the start state to some accepting state?

There are simple, efficient algorithms for this task.
More Questions

Q.: Is $\mathcal{L}(\mathcal{N}) = \Sigma^*$?

Answer: Check if $\overline{\mathcal{L}(\mathcal{N})} = \emptyset$.

Q.: Given $\mathcal{N}_1$ and $\mathcal{N}_2$, is $\mathcal{L}(\mathcal{N}_1) \subseteq \mathcal{L}(\mathcal{N}_2)$?

Answer: Check if $\overline{\mathcal{L}(\mathcal{N}_2)} \cap \mathcal{L}(\mathcal{N}_1) = \emptyset$.

Q.: Given $\mathcal{N}_1$ and $\mathcal{N}_2$, is $\mathcal{L}(\mathcal{N}_1) = \mathcal{L}(\mathcal{N}_2)$?

Answer: Check if $\mathcal{L}(\mathcal{N}_1) \subseteq \mathcal{L}(\mathcal{N}_2)$ and $\mathcal{L}(\mathcal{N}_2) \subseteq \mathcal{L}(\mathcal{N}_1)$.

In the future, we will see that for stronger models of computations, many of these problems cannot be solved by any algorithm.
Part V

Summary — Regular Languages
Summary - Regular Languages

So far we saw

- Finite automata,
- Regular languages,
- Regular expressions,
- Myhill-Nerode theorem and pumping lemma for regular languages.

Did not do (see appendix): Given a DFA $M$, the minimal (in # of states) DFA $M'$ with $L(M') = L(M)$.

Next class we introduce stronger machines and languages with more expressive power:

- pushdown automata,
- context-free languages,
- context-free grammars
Part VII

Appendix

Not Taught in Class
Section 1

Finding the Minimal Automata
Finding the minimal automata

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$, find the minimal (with respect to \# of states) DFA $M'$ with $L(M') = L(M)$.

**Definition 19 (Equivalent states)**

For DFA $(Q, \Sigma, \delta, q_1, F)$, states $q_1, q_2 \in Q$ are equivalent, if for all $x_1, x_2 \in \Sigma^*$ with $\hat{\delta}(q_0, x_i) = q_i$, it holds that $x_1 \sim x_2$.

**Idea**: keep merging equivalent states in $Q$, until all states are non-equivalent.

**Actual idea**:

1. Start with the two sets $F$ and $Q \setminus F$.

2. Keep *splitting* the sets until all states in the same set are equivalent.

   To check whether states $q$ and $q'$ are equivalent, check if $\delta(q, \sigma)$ and $\delta(q', \sigma)$ are in the same set, for all $\sigma \in \Sigma$.

3. Merge all states in the same set.

**Is this a minimal automate?** Yes, if $M$ has no unreachable states.

But we can assume that wlg.
Algorithm 20

Input: DFA $M = (Q, \Sigma, \delta, q_0, F)$

1. Let $T = \{F, Q \setminus F\}$.

2. While $\exists S \in T$, $q_1, q_2 \in S$ and $\sigma \in \Sigma^*$ s.t,
   $\delta(q_1, \sigma) \in S'$ and $\delta(q_2, \sigma) \notin S'$ for some $S' \in T$:
   
   2.1 Let $S_{sp} = \{q \in S : \delta(q, \sigma) \in S'\}$.
   
   2.2 Set $T = T \cup S_{sp} \cup (S \setminus S_{sp}) \setminus S$.

3. Output DFA $M' = (Q', \delta', q_0', F')$, where

   $Q' = T$
   
   $q_0' = S_0 \in T$, where $q_0 \in S_0$.
   
   $F' = \{S \in T : S \subseteq F\}$
   
   $\delta'(S, \sigma) = S' \in T$, s.t. $\delta(q, \sigma) \in S'$ for (every) $q \in S$.

Claim 21

The above algorithm outputs the minimal automata for $L(M)$. 
Example

On board...
Section 2

Using Homomorphism
Using homomorphism

We know that $L_1 = \{0^n1^n : n \geq 1\}$ is not regular, show that $L_2 = \{a^nba^n : n \geq 1\}$ is not regular.

We will prove using homomorphism and inverse homomorphism. Let

1. $h_1(a) = a, h_1(b) = b, h_1(c) = a. \quad (h_1: \{a, b, c\} \mapsto \{a, b, c\}^*)$
2. $h_2(a) = 0, h_2(b) = \epsilon, h_2(c) = 1. \quad (h_1: \{a, b, c\} \mapsto \{0, 1\}^*)$

We prove $h_2(h_1^{-1}(L_2) \cap a^*b^*c^*) = L_1$. Thus, $L_2$ is not regular (?)

1. $h_1^{-1}(L_2) = L((a \cup c)^n) b L((a \cup c)^n)$
2. $h_1^{-1}(L_2) \cap a^*b^*c^* = \{a^nbc^n : n \geq 1\}$
3. $h_2(h_1^{-1}(L_2) \cap a^*b^*c^*) = \{0^n1^n : n \geq 1\}$