COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

February 20, 2023

Lecture 13: Myhill-Nerode Theorem

Recap

Define a relation $\equiv$ on the set of states:

Recap

Define a relation $\equiv$ on the set of states:

$$p \equiv q \iff \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F)$$

$\equiv$ is an equivalence relation.

$$[p] := \{q \mid q \equiv p\} \quad \text{Equivalence classes}$$

Every element $p \in Q$ is contained in exactly one equivalence class $[p]$.

$$p \equiv q \iff [p] = [q]$$

An algorithm for DFA minimization

Let M be a DFA with no inaccessible states. We will mark (unordered) pairs of states $\{p, q\}$ if we discover a reason why they are not equivalent.

1. Write down a table of pairs $\{p, q\}$, initially unmarked.
2. Mark $\{p, q\}$ if $p \in F$ and $q \notin F$, or vice-versa.
3. Repeat until no change occurs: if there exists an unmarked pair $\{p, q\}$ such that $\{\delta(p, a), \delta(q, a)\}$ is marked for some $a \in \Sigma$ then mark $\{p, q\}$.
4. When done, $p \equiv q$ iff $\{p, q\}$ is not marked.

Minimization problem

Algorithm

Let $Q = \{q_1, \dots, q_n\}$.

Minimization problem

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i, j) = --$

Minimization problem

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i, j) = --$
2. For each $1 \leq i < j \leq n$

Minimization problem

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i,j) = --$
2. For each $1 \leq i < j \leq n$

   If $(q_i \in F$ AND $q_j \notin F)$ OR $(q_i \in F$ AND $q_j \notin F)$
   $T(i,j) \leftarrow \checkmark$
3. Repeat

Minimization problem

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i,j) = --$
2. For each $1 \leq i < j \leq n$

   If $(q_i \in F$ AND $q_j \notin F)$ OR $(q_i \in F$ AND $q_j \notin F)$
   $T(i,j) \leftarrow \checkmark$
3. Repeat

   $\{$ For each $1 \leq i < j \leq n$

Minimization problem

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \le i < j \le n$, initialize $T(i,j) = --$
2. For each $1 \le i < j \le n$

   If $(q_i \in F$ AND $q_j \notin F)$ OR $(q_i \in F$ AND $q_j \notin F)$

   $T(i,j) \leftarrow \checkmark$
3. Repeat

   $\{$ For each $1 \le i < j \le n$

   If $\exists a \in \Sigma, T(\delta(q_i,a), \delta(q_j,a)) = \checkmark$

   then $T(i,j) \leftarrow \checkmark$

   $\}$

   Untill T stays unchanged.

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

$$\mathcal{Q} = \{\{p, q\} \mid p, q \in Q, p \neq q\}$$

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

$\mathcal{Q} = \{\{p, q\} \mid p, q \in Q, p \neq q\}$

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

$$\Delta(\{p, q\}, a) := \{\{p', q'\} \mid p = \delta(p', a), q = \delta(q', a)\}$$

Proof of Correctness

Claim: The pair $\{p,q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p,x) \in F$ and $\hat{\delta}(q,x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

$\mathcal{Q} = \{\{p,q\} \mid p,q \in Q, p \neq q\}$

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

$$\Delta(\{p,q\},a) := \{\{p',q'\} \mid p = \delta(p',a), q = \delta(q',a)\}$$

$\mathcal{S} := \{\{p,q\} \mid p \in F, q \notin F\}$

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

$\mathcal{Q} = \{\{p, q\} \mid p, q \in Q, p \neq q\}$

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

$$\Delta(\{p, q\}, a) := \{\{p', q'\} \mid p = \delta(p', a), q = \delta(q', a)\}$$

$\mathcal{S} := \{\{p, q\} \mid p \in F, q \notin F\}$

- Step 2 marks elements of $\mathcal{S}$.

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

$\mathcal{Q} = \{\{p, q\} \mid p, q \in Q, p \neq q\}$

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

$$\Delta(\{p, q\}, a) := \{\{p', q'\} \mid p = \delta(p', a), q = \delta(q', a)\}$$

$\mathcal{S} := \{\{p, q\} \mid p \in F, q \notin F\}$

- Step 2 marks elements of $\mathcal{S}$.
- Step 3 marks pairs in $\Delta(\{p, q\}, a)$ when $\{p, q\}$ is marked for some $a \in \Sigma$.

Proof of Correctness

Claim: The pair $\{p,q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p,x) \in F$ and $\hat{\delta}(q,x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

$\mathcal{Q} = \{\{p,q\} \mid p,q \in Q, p \neq q\}$

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

$$\Delta(\{p,q\},a) := \{\{p',q'\} \mid p = \delta(p',a), q = \delta(q',a)\}$$

$\mathcal{S} := \{\{p,q\} \mid p \in F, q \notin F\}$

- Step 2 marks elements of $\mathcal{S}$.
- Step 3 marks pairs in $\Delta(\{p,q\},a)$ when $\{p,q\}$ is marked for some $a \in \Sigma$.
- Claim above says $p \not\equiv q \iff \{p,q\}$ is reachable from $\mathcal{S}$.

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

$\mathcal{Q} = \{\{p, q\} \mid p, q \in Q, p \neq q\}$

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

$$\Delta(\{p, q\}, a) := \{\{p', q'\} \mid p = \delta(p', a), q = \delta(q', a)\}$$

$\mathcal{S} := \{\{p, q\} \mid p \in F, q \notin F\}$

- Step 2 marks elements of $\mathcal{S}$.
- Step 3 marks pairs in $\Delta(\{p, q\}, a)$ when $\{p, q\}$ is marked for some $a \in \Sigma$.
- Claim above says $p \not\equiv q \iff \{p, q\}$ is reachable from $\mathcal{S}$.

Question: Is the resulting automaton minimal?

Unique Minimized DFA

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Suppose not. Assume there exists A' with fewer states and $L(A') = L(A)$.

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Suppose not. Assume there exists A' with fewer states and $L(A') = L(A)$. Therefore, initial states q_0 and q_0' of A and A' respectively are equivalent.

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Suppose not. Assume there exists A' with fewer states and $L(A') = L(A)$. Therefore, initial states q_0 and q_0' of A and A' respectively are equivalent.
Claim: All states of A are equivalent to some state of A'.

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Suppose not. Assume there exists A' with fewer states and $L(A') = L(A)$. Therefore, initial states q_0 and q_0' of A and A' respectively are equivalent.
Claim: All states of A are equivalent to some state of A'.

- We know all states of A are reachable from its initial state(why?).

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Suppose not. Assume there exists A' with fewer states and $L(A') = L(A)$. Therefore, initial states q_0 and q_0' of A and A' respectively are equivalent.
Claim: All states of A are equivalent to some state of A'.

- We know all states of A are reachable from its initial state(why?).
- Let q be a state in A. Let word w take A to q.
- Let word w take A' to some state q'.

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Suppose not. Assume there exists A' with fewer states and $L(A') = L(A)$. Therefore, initial states q_0 and q_0' of A and A' respectively are equivalent.
Claim: All states of A are equivalent to some state of A'.

- We know all states of A are reachable from its initial state(why?).
- Let q be a state in A. Let word w take A to q.
- Let word w take A' to some state q'.
- q and q' must be equivalent. Otherwise, q_0 and q_0' are not equivalent.

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Suppose not. Assume there exists A' with fewer states and $L(A') = L(A)$. Therefore, initial states q_0 and q_0' of A and A' respectively are equivalent.
Claim: All states of A are equivalent to some state of A'.

- We know all states of A are reachable from its initial state(why?).
- Let q be a state in A. Let word w take A to q.
- Let word w take A' to some state q'.
- q and q' must be equivalent. Otherwise, q_0 and q_0' are not equivalent.
- By pigeonhole principle, there are states q_1 and q_2 of A such that they are equivalent to the same state of A'.

Unique Minimized DFA

Lemma

The DFA A obtained by running the collapsing algorithm is the minimal DFA for $L(A)$.

Suppose not. Assume there exists A' with fewer states and $L(A') = L(A)$. Therefore, initial states q_0 and q_0' of A and A' respectively are equivalent.
Claim: All states of A are equivalent to some state of A'.

- We know all states of A are reachable from its initial state(why?).
- Let q be a state in A. Let word w take A to q.
- Let word w take A' to some state q'.
- q and q' must be equivalent. Otherwise, q_0 and q_0' are not equivalent.
- By pigeonhole principle, there are states q_1 and q_2 of A such that they are equivalent to the same state of A'.
- Therefore, q_1 and q_2 are equivalent. But A is minimized, and no two states of A are equivalent in a minimized DFA. Contradiction!

A relation with some strange properties

A relation with some strange properties

Let $R \subseteq \Sigma^*$ be a regular language and $M = (Q, \Sigma, \delta, s, F)$ be a DFA for R. Consider the relation $\equiv_M$:

$$x \equiv_M y \iff \hat{\delta}(s, x) = \hat{\delta}(s, y)$$

A relation with some strange properties

Let $R \subseteq \Sigma^*$ be a regular language and $M = (Q, \Sigma, \delta, s, F)$ be a DFA for R. Consider the relation $\equiv_M$:

$$x \equiv_M y \iff \hat{\delta}(s, x) = \hat{\delta}(s, y)$$

Exercise: $\equiv_M$ is an equivalence relation. Note: This is different from last class! Relation $\equiv$ on the set of states:

$$p \equiv q \iff \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F)$$

A relation with some strange properties

Let $R \subseteq \Sigma^*$ be a regular language and $M = (Q, \Sigma, \delta, s, F)$ be a DFA for R. Consider the relation $\equiv_M$:

$$x \equiv_M y \iff \hat{\delta}(s, x) = \hat{\delta}(s, y)$$

Exercise: $\equiv_M$ is an equivalence relation.

- **Right Congruence:** For any $x, y \in \Sigma^*$ and $a \in \Sigma$

$$x \equiv_M y \implies xa \equiv_M ya$$

A relation with some strange properties

Let $R \subseteq \Sigma^*$ be a regular language and $M = (Q, \Sigma, \delta, s, F)$ be a DFA for R. Consider the relation $\equiv_M$:

$$x \equiv_M y \iff \hat{\delta}(s, x) = \hat{\delta}(s, y)$$

Exercise: $\equiv_M$ is an equivalence relation.

- **Right Congruence:** For any $x, y \in \Sigma^*$ and $a \in \Sigma$

$$x \equiv_M y \implies xa \equiv_M ya$$

A relation with some strange properties

Let $R \subseteq \Sigma^*$ be a regular language and $M = (Q, \Sigma, \delta, s, F)$ be a DFA for R. Consider the relation $\equiv_M$:

$$x \equiv_M y \iff \hat{\delta}(s, x) = \hat{\delta}(s, y)$$

Exercise: $\equiv_M$ is an equivalence relation.

- **Right Congruence:** For any $x, y \in \Sigma^*$ and $a \in \Sigma$

$$x \equiv_M y \implies xa \equiv_M ya$$

Assume $x \equiv_M y$. Then

$$\begin{aligned} \hat{\delta}(s, xa) &= \delta(\hat{\delta}(s, x), a) \\ &= \delta(\hat{\delta}(s, y), a) \qquad \text{(by assumption)} \\ &= \hat{\delta}(s, ya) \end{aligned}$$

A relation with some strange properties

Let $R \subseteq \Sigma^*$ be a regular language and $M = (Q, \Sigma, \delta, s, F)$ be a DFA for R. Consider the relation $\equiv_M$:

$$x \equiv_M y \iff \hat{\delta}(s, x) = \hat{\delta}(s, y)$$

Exercise: $\equiv_M$ is an equivalence relation.

- **Right Congruence:** For any $x, y \in \Sigma^*$ and $a \in \Sigma$

$$x \equiv_M y \implies xa \equiv_M ya$$

- **Refines R:** For any $x, y \in \Sigma^*$,

$$x \equiv_M y \implies (x \in R \iff y \in R)$$

  This is because $\hat{\delta}(s, x) = \hat{\delta}(s, y)$

A relation with some strange properties

Let $R \subseteq \Sigma^*$ be a regular language and $M = (Q, \Sigma, \delta, s, F)$ be a DFA for R. Consider the relation $\equiv_M$:

$$x \equiv_M y \iff \hat{\delta}(s, x) = \hat{\delta}(s, y)$$

Exercise: $\equiv_M$ is an equivalence relation.

- **Right Congruence:** For any $x, y \in \Sigma^*$ and $a \in \Sigma$

$$x \equiv_M y \implies xa \equiv_M ya$$

- **Refines R:** For any $x, y \in \Sigma^*$,

$$x \equiv_M y \implies (x \in R \iff y \in R)$$

  This is because $\hat{\delta}(s, x) = \hat{\delta}(s, y)$
- **Finite index:** There are only finitely many equivalence classes on Σ^* under $\equiv_M$ (There is at exactly one equivalence corresponding to each state of M).

Myhill-Nerode Relations

Definition

An equivalence relation on Σ^* is said to be a Myhill-Nerode relation for the language R, if it is a right congruence of finite index refining R.

Myhill-Nerode Relations

Definition

An equivalence relation on Σ^* is said to be a Myhill-Nerode relation for the language R, if it is a right congruence of finite index refining R.

- Definition characterises exactly those relations on Σ^* that are $\equiv_M$ for some automaton M.

Myhill-Nerode Relations

Definition

An equivalence relation on Σ^* is said to be a Myhill-Nerode relation for the language R, if it is a right congruence of finite index refining R.

- Definition characterises exactly those relations on Σ^* that are $\equiv_M$ for some automaton M.
- Given $\equiv_M$, one can reconstruct M just using the fact that it is Myhill-Nerode.

Myhill-Nerode Relations

Definition

An equivalence relation on Σ^* is said to be a Myhill-Nerode relation for the language R, if it is a right congruence of finite index refining R.

- Definition characterises exactly those relations on Σ^* that are $\equiv_M$ for some automaton M.
- Given $\equiv_M$, one can reconstruct M just using the fact that it is Myhill-Nerode. In fact,

$$\begin{array}{ccc} M & \rightarrow & \equiv_M \\ \equiv_M & \rightarrow & M \end{array}$$

are both inverses upto isomorphism of the automata.

Myhill-Nerode theorem

Theorem (John Myhill, Anil Nerode (1958))

L is regular if and only if there exists a Myhill-Nerode relation for L.

Proof

- Suppose $\equiv$ is a Myhill-Nerode relation on Σ^*.

Myhill-Nerode theorem

Theorem (John Myhill, Anil Nerode (1958)

L is regular if and only if there exists a Myhill-Nerode relation for L.

Proof

- Suppose $\equiv$ is a Myhill-Nerode relation on Σ^*.
- Let $[x] = \{y \in \Sigma^* \mid y \equiv x\}$.

Myhill-Nerode theorem

Theorem (John Myhill, Anil Nerode (1958))

L is regular if and only if there exists a Myhill-Nerode relation for L.

Proof

- Suppose $\equiv$ is a Myhill-Nerode relation on Σ^*.
- Let $[x] = \{y \in \Sigma^* \mid y \equiv x\}$.
- We will build $M_\equiv = (Q, \Sigma, q_0, \delta, F)$

Myhill-Nerode theorem

Theorem (John Myhill, Anil Nerode (1958)

L is regular if and only if there exists a Myhill-Nerode relation for L.

Proof

- Suppose $\equiv$ is a Myhill-Nerode relation on Σ^*.
- Let $[x] = \{y \in \Sigma^* \mid y \equiv x\}$.
- We will build $M_{\equiv} = (Q, \Sigma, q_0, \delta, F)$
  - $Q = \{[x] \mid x \in \Sigma^*\}$

Myhill-Nerode theorem

Theorem (John Myhill, Anil Nerode (1958)

L is regular if and only if there exists a Myhill-Nerode relation for L.

Proof

- Suppose $\equiv$ is a Myhill-Nerode relation on Σ^*.
- Let $[x] = \{y \in \Sigma^* \mid y \equiv x\}$.
- We will build $M_\equiv = (Q, \Sigma, q_0, \delta, F)$
  - $Q = \{[x] \mid x \in \Sigma^*\}$
  - $q_0 = [\epsilon]$

Myhill-Nerode theorem

Theorem (John Myhill, Anil Nerode (1958))

L is regular if and only if there exists a Myhill-Nerode relation for L.

Proof

- Suppose $\equiv$ is a Myhill-Nerode relation on Σ^*.
- Let $[x] = \{y \in \Sigma^* \mid y \equiv x\}$.
- We will build $M_\equiv = (Q, \Sigma, q_0, \delta, F)$
  - $Q = \{[x] \mid x \in \Sigma^*\}$
  - $q_0 = [\epsilon]$
  - $\delta([x], a) = [xa]$.

Myhill-Nerode theorem

Theorem (John Myhill, Anil Nerode (1958)

L is regular if and only if there exists a Myhill-Nerode relation for L.

Proof

- Suppose $\equiv$ is a Myhill-Nerode relation on Σ^*.
- Let $[x] = \{y \in \Sigma^* \mid y \equiv x\}$.
- We will build $M_\equiv = (Q, \Sigma, q_0, \delta, F)$
  - $Q = \{[x] \mid x \in \Sigma^*\}$
  - $q_0 = [\epsilon]$
  - $\delta([x], a) = [xa]$.
  - $F = \{[x] \mid x \in L\}$

Myhill-Nerode theorem

Theorem (John Myhill, Anil Nerode (1958))

L is regular if and only if there exists a Myhill-Nerode relation for L.

Proof

- Suppose $\equiv$ is a Myhill-Nerode relation on Σ^*.
- Let $[x] = \{y \in \Sigma^* \mid y \equiv x\}$.
- We will build $M_\equiv = (Q, \Sigma, q_0, \delta, F)$
  - $Q = \{[x] \mid x \in \Sigma^*\}$
  - $q_0 = [\epsilon]$
  - $\delta([x], a) = [xa]$.
  - $F = \{[x] \mid x \in L\}$
- Proof of correctness: By induction.

$\square$

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots,\}$ has only finitely many equivalence classes.

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots, \}$ has only finitely many equivalence classes.
  - Then there exists $1 \leq p < q$ such that, $[a^p] = [a^q]$.

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots,\}$ has only finitely many equivalence classes.
  - Then there exists $1 \leq p < q$ such that, $[a^p] = [a^q]$.
  - $[a^p][a^{q!-p}] = [a^q][a^{q!-p}]$

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots,\}$ has only finitely many equivalence classes.
  - Then there exists $1 \leq p < q$ such that, $[a^p] = [a^q]$.
  - $[a^p][a^{q!-p}] = [a^q][a^{q!-p}]$
  - $[a^{q!}] = [a^q][a^{q!-p}]$

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots,\}$ has only finitely many equivalence classes.
  - Then there exists $1 \leq p < q$ such that, $[a^p] = [a^q]$.
  - $[a^p][a^{q!-p}] = [a^q][a^{q!-p}]$
  - $[a^{q!}] = [a^q][a^{q!-p}]$
  - LHS $\in L$, RHS $\notin L$. (Why?)
  - Suffices to show $q! - p + q < (q+1)!$

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots,\}$ has only finitely many equivalence classes.
  - Then there exists $1 \le p < q$ such that, $[a^p] = [a^q]$.
  - $[a^p][a^{q!-p}] = [a^q][a^{q!-p}]$
  - $[a^{q!}] = [a^q][a^{q!-p}]$
  - LHS $\in L$, RHS $\notin L$. (Why?)
  - Suffices to show $q! - p + q < (q+1)!$

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots,\}$ has only finitely many equivalence classes.
  - Then there exists $1 \leq p < q$ such that, $[a^p] = [a^q]$.
  - $[a^p][a^{q!-p}] = [a^q][a^{q!-p}]$
  - $[a^{q!}] = [a^q][a^{q!-p}]$
  - LHS $\in L$, RHS $\notin L$. (Why?)
  - Suffices to show $q! - p + q < (q+1)!$

What about $\{a^p \mid p \in \mathbb{N}, p \text{ prime}\}$?

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots,\}$ has only finitely many equivalence classes.
  - Then there exists $1 \leq p < q$ such that, $[a^p] = [a^q]$.
  - $[a^p][a^{q!-p}] = [a^q][a^{q!-p}]$
  - $[a^{q!}] = [a^q][a^{q!-p}]$
  - LHS $\in L$, RHS $\notin L$. (Why?)
  - Suffices to show $q! - p + q < (q+1)!$

What about $\{a^p \mid p \in \mathbb{N}, p \text{ prime}\}$?

$$[a^p] = [a^q] \implies [a^p][a^{q-p}] = [a^q][a^{q-p}] = [a^{2q-p}] = [a^p][a^{2(q-p)}] \in L$$

Example: Factorial

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a\}^*$.
- Suppose it is a right congruence and it refines L.
- Then it cannot have finite index.
  - Suppose $\{a, a^2, \ldots,\}$ has only finitely many equivalence classes.
  - Then there exists $1 \leq p < q$ such that, $[a^p] = [a^q]$.
  - $[a^p][a^{q!-p}] = [a^q][a^{q!-p}]$
  - $[a^{q!}] = [a^q][a^{q!-p}]$
  - LHS $\in L$, RHS $\notin L$. (Why?)
  - Suffices to show $q! - p + q < (q+1)!$

What about $\{a^p \mid p \in \mathbb{N}, p \text{ prime}\}$?

$$[a^p] = [a^q] \implies [a^p][a^{q-p}] = [a^q][a^{q-p}] = [a^{2q-p}] = [a^p][a^{2(q-p)}] \in L$$

In general, $a^{p+i(q-p)]} \in L$ for every i.

Example: $\{a^n b^n\}$

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a, b\}^*$.

Example: $\{a^n b^n\}$

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a, b\}^*$.
- Assume that it is a right congruence and refines L.

Example: $\{a^n b^n\}$

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a, b\}^*$.
- Assume that it is a right congruence and refines L.
- Now we will show that it does not have finite index.

Example: $\{a^n b^n\}$

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a, b\}^*$.
- Assume that it is a right congruence and refines L.
- Now we will show that it does not have finite index.
  - For $n \neq m$, say $a^n \equiv a^m$.

Example: $\{a^n b^n\}$

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a, b\}^*$.
- Assume that it is a right congruence and refines L.
- Now we will show that it does not have finite index.
  - For $n \neq m$, say $a^n \equiv a^m$.
  - By right congruence $a^n \cdot b^n \equiv a^m \cdot b^n$.

Example: $\{a^n b^n\}$

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a, b\}^*$.
- Assume that it is a right congruence and refines L.
- Now we will show that it does not have finite index.
  - For $n \neq m$, say $a^n \equiv a^m$.
  - By right congruence $a^n \cdot b^n \equiv a^m \cdot b^n$.
  - But $a^n b^n \in L$ and $a^m b^n \notin L$.

Example: $\{a^n b^n\}$

Consider

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

- Consider an equivalence relation $\equiv$ on $\{a, b\}^*$.
- Assume that it is a right congruence and refines L.
- Now we will show that it does not have finite index.
  - For $n \neq m$, say $a^n \equiv a^m$.
  - By right congruence $a^n \cdot b^n \equiv a^m \cdot b^n$.
  - But $a^n b^n \in L$ and $a^m b^n \notin L$.

Exercise: Work out the Myhill-Nerode proof for PAL.

More consequences of Myhill-Nerode

What are the properties of the automaton constructed from L?

More consequences of Myhill-Nerode

What are the properties of the automaton constructed from L?

- DFA representation of languages is not unique!

More consequences of Myhill-Nerode

What are the properties of the automaton constructed from L?

- DFA representation of languages is not unique!
- What does Myhill-Nerode construction give?

More consequences of Myhill-Nerode

What are the properties of the automaton constructed from L?

- DFA representation of languages is not unique!
- What does Myhill-Nerode construction give?

Theorem

Given L, the DFA constructed from L using the Myhill-Nerode consruction has the minimum number of states possible.

Algorithms for Regular Sets

Algorithms for Regular Sets

Membership testing

Given a regular language L over Σ and a string $w \in \Sigma^*$, check if $w \in L$

Algorithms for Regular Sets

Membership testing

Given a regular language L over Σ and a string $w \in \Sigma^*$, check if $w \in L$

Emptiness checking

Given a regular language L over Σ, check if $L = \varnothing$

Algorithms for Regular Sets

Membership testing

Given a regular language L over Σ and a string $w \in \Sigma^*$, check if $w \in L$

Emptiness checking

Given a regular language L over Σ, check if $L = \emptyset$

Language Equivalence

Given two regular languages L_1 and L_2 over Σ, check if $L_1 = L_2$.

Algorithms for Regular Sets

Membership testing

Given a regular language L over Σ and a string $w \in \Sigma^*$, check if $w \in L$

Emptiness checking

Given a regular language L over Σ, check if $L = \varnothing$

Language Equivalence

Given two regular languages L_1 and L_2 over Σ, check if $L_1 = L_2$.

$$L_1 = L_2 \iff (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$$

Algorithms for Regular Sets

Membership testing

Given a regular language L over Σ and a string $w \in \Sigma^*$, check if $w \in L$

Emptiness checking

Given a regular language L over Σ, check if $L = \varnothing$

Language Equivalence

Given two regular languages L_1 and L_2 over Σ, check if $L_1 = L_2$.

$$L_1 = L_2 \iff (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \varnothing$$

Membership testing

- **DFA**

Membership testing

- **DFA**
  - Run it on w and check if an accepting state is reached.

Membership testing

- **DFA**
  - Run it on w and check if an accepting state is reached.
  - Cost of algorithm: $O(|w|)$.

Membership testing

- **DFA**
  - Run it on w and check if an accepting state is reached.
  - Cost of algorithm: $O(|w|)$.
- **NFA**

Membership testing

- **DFA**
  - Run it on w and check if an accepting state is reached.
  - Cost of algorithm: $O(|w|)$.
- **NFA**
  - Compute reachable states on w.

Membership testing

- **DFA**
  - Run it on w and check if an accepting state is reached.
  - Cost of algorithm: $O(|w|)$.
- **NFA**
  - Compute reachable states on w.
  - Cost of algorithm: $O(|w|n^2)$ where n is number of states.

Membership testing

- **DFA**
  - Run it on w and check if an accepting state is reached.
  - Cost of algorithm: $O(|w|)$.
- **NFA**
  - Compute reachable states on w.
  - Cost of algorithm: $O(|w|n^2)$ where n is number of states.
- **RegEx**

Membership testing

- **DFA**
  - Run it on w and check if an accepting state is reached.
  - Cost of algorithm: $O(|w|)$.
- **NFA**
  - Compute reachable states on w.
  - Cost of algorithm: $O(|w|n^2)$ where n is number of states.
- **RegEx**
  - Convert to a NFA and test membership.

Membership testing

- **DFA**
  - Run it on w and check if an accepting state is reached.
  - Cost of algorithm: $O(|w|)$.
- **NFA**
  - Compute reachable states on w.
  - Cost of algorithm: $O(|w|n^2)$ where n is number of states.
- **RegEx**
  - Convert to a NFA and test membership.
  - Cost of algorithm: $O(|w|n^2)$?.

Language Equivalence and Emptiness checking

- **DFA**

Language Equivalence and Emptiness checking

- **DFA**
  - Complementation is trivial

Language Equivalence and Emptiness checking

- **DFA**
  - Complementation is trivial
  - Construct DFA for $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ and test if any any of the final states are reachable from the start state.

Language Equivalence and Emptiness checking

- **DFA**
  - Complementation is trivial
  - Construct DFA for $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ and test if any any of the final states are reachable from the start state.
  - Cost of algorithm: Exercise.

Language Equivalence and Emptiness checking

- **DFA**
  - Complementation is trivial
  - Construct DFA for $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ and test if any any of the final states are reachable from the start state.
  - Cost of algorithm: Exercise.
- **NFA and Regex**

Language Equivalence and Emptiness checking

- **DFA**
  - Complementation is trivial
  - Construct DFA for $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ and test if any any of the final states are reachable from the start state.
  - Cost of algorithm: Exercise.
- **NFA and Regex**
  - Complementation is expensive!

Language Equivalence and Emptiness checking

- **DFA**
  - Complementation is trivial
  - Construct DFA for $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ and test if any any of the final states are reachable from the start state.
  - Cost of algorithm: Exercise.
- **NFA and Regex**
  - Complementation is expensive!
  - Given NFA A with n states, NFA for $\overline{L(A)}$ exists with 2^n states.

Language Equivalence and Emptiness checking

- **DFA**
  - Complementation is trivial
  - Construct DFA for $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ and test if any any of the final states are reachable from the start state.
  - Cost of algorithm: Exercise.
- **NFA and Regex**
  - Complementation is expensive!
  - Given NFA A with n states, NFA for $\overline{L(A)}$ exists with 2^n states.
  - **Exercise:** There exist NFA A with n states such that the smallest NFA for $\overline{L(A)}$ requires 2^n states

Language Equivalence and Emptiness checking

- **DFA**
  - Complementation is trivial
  - Construct DFA for $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ and test if any any of the final states are reachable from the start state.
  - Cost of algorithm: Exercise.
- **NFA and Regex**
  - Complementation is expensive!
  - Given NFA A with n states, NFA for $\overline{L(A)}$ exists with 2^n states.
  - **Exercise:** There exist NFA A with n states such that the smallest NFA for $\overline{L(A)}$ requires 2^n states