Intro. Algorithms & Models of Computation

CS/ECE 374A, Fall 2022

Proving Non-regularity

Lecture 6

Thursday, September 8, 2022

LaTeXed: October 13, 2022 14:18

6.1
Not all languages are regular

Regular Languages, DFAs, NFAs

Theorem 6.1.

Languages accepted by DFAs, NFAs, and regular expressions are the same.

Question: Is every language a regular language? No.

- Each DFA $\mathbf{M}$ can be represented as a string over a finite alphabet $\boldsymbol{\Sigma}$ by appropriate encoding
- Hence number of regular languages is countably infinite
- Number of languages is uncountably infinite
- Hence there must be a non-regular language!

A direct proof

$\mathbf{L} = \left\{\mathbf{0^i 1^i} \;\middle|\; \mathbf{i} \geq \mathbf{0}\right\} = \{\epsilon, \mathbf{01}, \mathbf{0011}, \mathbf{000111}, \cdots, \}$

Theorem 6.2.

L *is not regular.*

A Simple and Canonical Non-regular Language

$\mathbf{L} = \{0^i 1^i \mid i \geq 0\} = \{\epsilon, 01, 0011, 000111, \cdots, \}$

Theorem 6.3.

L *is not regular.*

Question: Proof?

Intuition: Any program to recognize **L** seems to require counting number of zeros in input which cannot be done with fixed memory.

How do we formalize intuition and come up with a formal proof?

Proof by Contradiction

- Suppose $\mathbf{L}$ is regular. Then there is a DFA $\mathbf{M}$ such that $\mathbf{L(M) = L}$.
- Let $\mathbf{M = (Q, \{0, 1\}, \delta, s, A)}$ where $|\mathbf{Q}| = \mathbf{n}$.

Consider strings $\epsilon, \mathbf{0, 00, 000}, \cdots, \mathbf{0^n}$ total of $\mathbf{n + 1}$ strings.

What states does $\mathbf{M}$ reach on the above strings? Let $\mathbf{q_i} = \delta^*(\mathbf{s, 0^i})$.

By pigeon hole principle $\mathbf{q_i = q_j}$ for some $\mathbf{0 \leq i < j \leq n}$.
That is, $\mathbf{M}$ is in the same state after reading $\mathbf{0^i}$ and $\mathbf{0^j}$ where $\mathbf{i \neq j}$.

$\mathbf{M}$ should accept $\mathbf{0^i 1^i}$ but then it will also accept $\mathbf{0^j 1^i}$ where $\mathbf{i \neq j}$.
This contradicts the fact that $\mathbf{M}$ accepts $\mathbf{L}$. Thus, there is no DFA for $\mathbf{L}$.

6.2
When two states are equivalent?

Equivalence between states

Definition 6.1.

$\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{A})$: DFA.

Two states $\mathbf{p}, \mathbf{q} \in \mathbf{Q}$ *are* **equivalent** *if for all strings* $\mathbf{w} \in \mathbf{\Sigma}^*$, *we have that*

$$\delta^*(\mathbf{p}, \mathbf{w}) \in \mathbf{A} \iff \delta^*(\mathbf{q}, \mathbf{w}) \in \mathbf{A}.$$

One can merge any two states that are equivalent into a single state.

Distinguishing between states

Definition 6.2.

$\mathbf{M = (Q, \Sigma, \delta, s, A)}$: DFA.

Two states $\mathbf{p, q \in Q}$ *are* **distinguishable** *if there exists a string* $\mathbf{w \in \Sigma^*}$*, such that*

$$\delta^*(\mathbf{p, w}) \in \mathbf{A} \qquad \textit{and} \qquad \delta^*(\mathbf{q, w}) \notin \mathbf{A}.$$

or

$$\delta^*(\mathbf{p, w}) \notin \mathbf{A} \qquad \textit{and} \qquad \delta^*(\mathbf{q, w}) \in \mathbf{A}.$$

Distinguishable prefixes

$\mathbf{M = (Q, \Sigma, \delta, s, A)}$: DFA

Idea: Every string $\mathbf{w} \in \mathbf{\Sigma}^*$ defines a state $\nabla \mathbf{w} = \delta^*(\mathbf{s}, \mathbf{w})$.

Definition 6.3.

Two strings $\mathbf{u}, \mathbf{w} \in \mathbf{\Sigma}^*$ *are* **distinguishable** *for* $\mathbf{M}$ *(or* $\mathbf{L(M)}$*) if* $\nabla \mathbf{u}$ *and* $\nabla \mathbf{w}$ *are distinguishable.*

Definition 6.4 (Direct restatement).

Two prefixes $\mathbf{u}, \mathbf{w} \in \mathbf{\Sigma}^*$ *are* **distinguishable** *for a language* $\mathbf{L}$ *if there exists a string* $\mathbf{x}$*, such that* $\mathbf{ux} \in \mathbf{L}$ *and* $\mathbf{wx} \notin \mathbf{L}$ *(or* $\mathbf{ux} \notin \mathbf{L}$ *and* $\mathbf{wx} \in \mathbf{L}$*).*

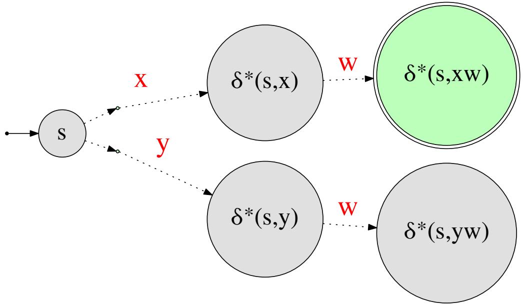
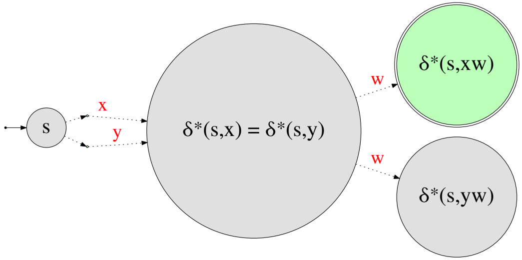
Distinguishable means different states

Lemma 6.5.

$\mathbf{L}$: *regular language.*
$\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{A})$: DFA *for* $\mathbf{L}$.
If $\mathbf{x}, \mathbf{y} \in \mathbf{\Sigma}^*$ *are distinguishable, then* $\nabla \mathbf{x} \neq \nabla \mathbf{y}$.

Reminder: $\nabla \mathbf{x} = \delta^*(\mathbf{s}, \mathbf{x}) \in \mathbf{Q}$ and $\nabla \mathbf{y} = \delta^*(\mathbf{s}, \mathbf{y}) \in \mathbf{Q}$

Proof by a figure

Possible	Not possible
	

Distinguishable strings means different states: Proof

Lemma 6.6.

$\mathbf{L}$: *regular language.*
$\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{A})$: DFA *for* $\mathbf{L}$.
If $\mathbf{x}, \mathbf{y} \in \mathbf{\Sigma}^*$ *are distinguishable, then* $\nabla \mathbf{x} \neq \nabla \mathbf{y}$.

Proof.

Assume for the sake of contradiction that $\nabla \mathbf{x} = \nabla \mathbf{y}$.
By assumption $\exists \mathbf{w} \in \mathbf{\Sigma}^*$ such that $\nabla \mathbf{xw} \in \mathbf{A}$ and $\nabla \mathbf{yw} \notin \mathbf{A}$.
$\implies \mathbf{A} \ni \nabla \mathbf{xw} = \delta^*(\mathbf{s}, \mathbf{xw}) = \delta^*(\nabla \mathbf{x}, \mathbf{w}) = \delta^*(\nabla \mathbf{y}, \mathbf{w})$
$= \delta^*(\mathbf{s}, \mathbf{yw}) = \nabla \mathbf{yw} \notin \mathbf{A}$.
$\implies \mathbf{A} \ni \nabla \mathbf{yw} \notin \mathbf{A}$. Impossible!
Assumption that $\nabla \mathbf{x} = \nabla \mathbf{y}$ is false. □

Review questions...

1. Prove for any $i \neq j$ then 0^i and 0^j are distinguishable for the language $\{0^k 1^k \mid k \geq 0\}$.
2. Let L be a regular language, and let $w_1, \ldots, w_k$ be strings that are all pairwise distinguishable for L. Prove that any DFA for L must have at least k states.
3. Prove that $\{0^k 1^k \mid k \geq 0\}$ is not regular.

6.3
Fooling sets: Proving non-regularity

Fooling Sets

Definition 6.1.

For a language $\mathbf{L}$ over $\mathbf{\Sigma}$ a set of strings $\mathbf{F}$ (could be infinite) is a fooling set or distinguishing set for $\mathbf{L}$ if every two distinct strings $\mathbf{x}, \mathbf{y} \in \mathbf{F}$ are distinguishable.

Example: $\mathbf{F} = \{0^i \mid i \geq 0\}$ is a fooling set for the language $\mathbf{L} = \{0^k 1^k \mid k \geq 0\}$.

Theorem 6.2.

Suppose $\mathbf{F}$ *is a fooling set for* $\mathbf{L}$. *If* $\mathbf{F}$ *is finite then there is no* DFA $\mathbf{M}$ *that accepts* $\mathbf{L}$ *with less than* $|\mathbf{F}|$ *states.*

Recall

Already proved the following lemma:

Lemma 6.3.

$\mathbf{L}$: *regular language.*
$\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{A})$: DFA *for* $\mathbf{L}$.
If $\mathbf{x}, \mathbf{y} \in \mathbf{\Sigma}^*$ *are distinguishable, then* $\nabla\mathbf{x} \neq \nabla\mathbf{y}$.

Reminder: $\nabla\mathbf{x} = \delta^*(\mathbf{s}, \mathbf{x})$.

Proof of theorem

Theorem 6.4 (Reworded.).

$\mathbf{L}$: *A language*
$\mathbf{F}$: *a fooling set for* $\mathbf{L}$.
If $\mathbf{F}$ *is finite then any* DFA $\mathbf{M}$ *that accepts* $\mathbf{L}$ *has at least* $|\mathbf{F}|$ *states.*

Proof.

Let $\mathbf{F} = \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$ be the fooling set.
Let $\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{A})$ be any DFA that accepts $\mathbf{L}$.
Let $\mathbf{q}_i = \nabla \mathbf{w}_i = \delta^*(\mathbf{s}, \mathbf{x}_i)$.
By lemma $\mathbf{q}_i \neq \mathbf{q}_j$ for all $\mathbf{i} \neq \mathbf{j}$.
As such, $|\mathbf{Q}| \geq |\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}| = |\{\mathbf{w}_1, \ldots, \mathbf{w}_m\}| = |\mathbf{F}|$. $\square$

Infinite Fooling Sets

Corollary 6.5.

If $\mathbf{L}$ has an infinite fooling set $\mathbf{F}$ then $\mathbf{L}$ is not regular.

Proof.

Let $\mathbf{w_1}, \mathbf{w_2}, \ldots \subseteq \mathbf{F}$ be an infinite sequence of strings such that every pair of them are distinguishable.
Assume for contradiction that $\exists$ $\mathbf{M}$ a DFA for $\mathbf{L}$.
Let $\mathbf{F_i} = \{\mathbf{w_1}, \ldots, \mathbf{w_i}\}$.
By theorem, $\#$ states of $\mathbf{M} \geq |\mathbf{F_i}| = \mathbf{i}$, for all $\mathbf{i}$.
As such, number of states in $\mathbf{M}$ is infinite.
Contradiction: DFA = deterministic finite automata. But $\mathbf{M}$ not finite. □

Examples

- $\{0^k 1^k \mid k \geq 0\}$
- {bitstrings with equal number of 0s and 1s}
- $\{0^k 1^\ell \mid k \neq \ell\}$

Harder example: The language of squares is not regular

$\{0^{k^2} \mid k \geq 0\}$

Really hard: Primes are not regular

An exercise left for your enjoyment

$\{\mathbf{0}^{\mathbf{k}} \mid \mathbf{k} \text{ is a prime number}\}$

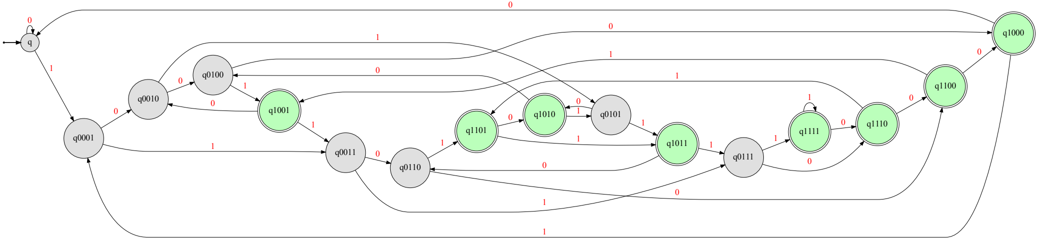
Hints:

1. Probably easier to prove directly on the automata.
2. There are infinite number of prime numbers.
3. For every $\mathbf{n} > \mathbf{0}$, observe that $\mathbf{n!}, \mathbf{n!} + \mathbf{1}, \ldots, \mathbf{n!} + \mathbf{n}$ are all composite – there are arbitrarily big gaps between prime numbers.

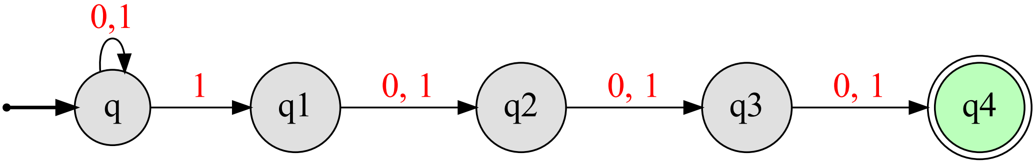
6.3.1
Exponential gap in number of states between DFA and NFA sizes

Exponential gap between NFA and DFA size

$\mathbf{L_4 = \{w \in \{0, 1\}^* \mid w}$ has a $\mathbf{1}$ located 4 positions from the end$\}$



DFA:



NFA:

Exponential gap between NFA and DFA size

$L_k = \{w \in \{0,1\}^* \mid w \text{ has a } 1 \ k \text{ positions from the end}\}$
Recall that L_k is accepted by a NFA N with $k+1$ states.

Theorem 6.6.

Every DFA that accepts L_k has at least 2^k states.

Claim 6.7.

$F = \{w \in \{0,1\}^* : |w| = k\}$ *is a fooling set of size* 2^k *for* L_k.

Why?

- Suppose $a_1 a_2 \dots a_k$ and $b_1 b_2 \dots b_k$ are two distinct bitstrings of length k
- Let i be first index where $a_i \neq b_i$
- $y = 0^{k-i-1}$ is a distinguishing suffix for the two strings

How do pick a fooling set

How do we pick a fooling set $\mathbf{F}$?

- If $\mathbf{x}, \mathbf{y}$ are in $\mathbf{F}$ and $\mathbf{x} \neq \mathbf{y}$ they should be distinguishable! Of course.
- All strings in $\mathbf{F}$ except maybe one should be prefixes of strings in the language $\mathbf{L}$. For example if $\mathbf{L} = \{\mathbf{0}^{\mathbf{k}}\mathbf{1}^{\mathbf{k}} \mid \mathbf{k} \geq \mathbf{0}\}$ do not pick $\mathbf{1}$ and $\mathbf{10}$ (say). Why?

6.4
Closure properties: Proving non-regularity

Non-regularity via closure properties

$\mathbf{H} = \{\text{bitstrings with equal number of 0s and 1s}\}$

$\mathbf{H}' = \{\mathbf{0}^k\mathbf{1}^k \mid k \geq 0\}$

Suppose we have already shown that $\mathbf{L}'$ is non-regular. Can we show that $\mathbf{L}$ is non-regular without using the fooling set argument from scratch?

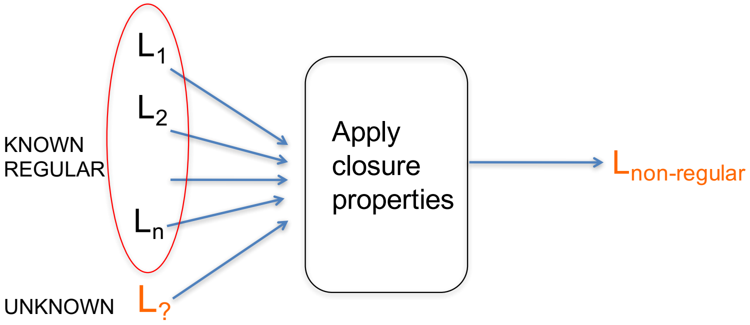
$\mathbf{H}' = \mathbf{H} \cap \mathbf{L}(\mathbf{0}^*\mathbf{1}^*)$

Claim: The above and the fact that $\mathbf{L}'$ is non-regular implies $\mathbf{L}$ is non-regular. Why?

Suppose $\mathbf{H}$ is regular. Then since $\mathbf{L}(\mathbf{0}^*\mathbf{1}^*)$ is regular, and regular languages are closed under intersection, $\mathbf{H}'$ also would be regular. But we know $\mathbf{H}'$ is not regular, a contradiction.

Non-regularity via closure properties

General recipe:



Proving non-regularity: Summary

- Method of distinguishing suffixes. To prove that L is non-regular find an infinite fooling set.
- Closure properties. Use existing non-regular languages and regular languages to prove that some new language is non-regular.
- Pumping lemma. We did not cover it but it is sometimes an easier proof technique to apply, but not as general as the fooling set technique.

6.5
Myhill-Nerode Theorem

One automata to rule them all

"Myhill-Nerode Theorem": A regular language **L** has a unique (up to naming) minimal automata, and it can be computed efficiently once any DFA is given for **L**.

6.5.1
Myhill-Nerode Theorem: Equivalence between strings

Indistinguishability

Recall:

Definition 6.1.

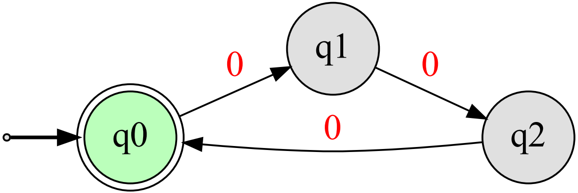
For a language $\mathbf{L}$ over $\mathbf{\Sigma}$ and two strings $\mathbf{x}, \mathbf{y} \in \mathbf{\Sigma}^*$ we say that $\mathbf{x}$ and $\mathbf{y}$ are distinguishable with respect to $\mathbf{L}$ if there is a string $\mathbf{w} \in \mathbf{\Sigma}^*$ such that exactly one of $\mathbf{xw}, \mathbf{yw}$ is in $\mathbf{L}$. $\mathbf{x}, \mathbf{y}$ are indistinguishable with respect to $\mathbf{L}$ if there is no such $\mathbf{w}$.

Given language $\mathbf{L}$ over $\mathbf{\Sigma}$ define a relation $\equiv_\mathbf{L}$ over strings in $\mathbf{\Sigma}^*$ as follows: $\mathbf{x} \equiv_\mathbf{L} \mathbf{y}$ iff $\mathbf{x}$ and $\mathbf{y}$ are indistinguishable with respect to $\mathbf{L}$.

Definition 6.2.

$\mathbf{x} \equiv_\mathbf{L} \mathbf{y}$ *means that* $\forall \mathbf{w} \in \mathbf{\Sigma}^*$: $\mathbf{xw} \in \mathbf{L} \iff \mathbf{yw} \in \mathbf{L}$.
In words: $\mathbf{x}$ *is* **equivalent** *to* $\mathbf{y}$ *under* $\mathbf{L}$.

Example: Equivalence classes



Indistinguishability

Claim 6.3.

$\equiv_L$ *is an equivalence relation over* Σ^*.

Proof.

1. Reflexive: $\forall x \in \Sigma^*: \forall w \in \Sigma^*: xw \in L \iff xw \in L. \implies x \equiv_L x.$
2. Symmetry: $x \equiv_L y$ then $\forall w \in \Sigma^*: xw \in L \iff yw \in L$
   $\forall w \in \Sigma^*: yw \in L \iff xw \in L \implies y \equiv_L x$.
3. Transitivity: $x \equiv_L y$ and $y \equiv_L z$
   $\forall w \in \Sigma^*: xw \in L \iff yw \in L$ and $\forall w \in \Sigma^*: yw \in L \iff zw \in L$
   $\implies \forall w \in \Sigma^*: xw \in L \iff zw \in L$
   $\implies x \equiv_L z.$

□

Equivalences over automatas...

Claim 6.4 (Just proved.).

$\equiv_{\mathsf{L}}$ *is an equivalence relation over* Σ^*.

Therefore, $\equiv_{\mathsf{L}}$ partitions Σ^* into a collection of equivalence classes.

Definition 6.5.

L: *A language For a string* $\mathsf{x} \in \Sigma^*$, *let*

$$[x] = [x]_{\mathsf{L}} = \{y \in \Sigma^* \mid x \equiv_{\mathsf{L}} y\}$$

be the **equivalence class** *of* x *according to* L.

Definition 6.6.

$[L] = \{[x]_{\mathsf{L}} \mid x \in \Sigma^*\}$ *is the set of* **equivalence classes** *of* L.

Strings in the same equivalence class are indistinguishable

Lemma 6.7.

Let $\mathbf{x}, \mathbf{y}$ *be two distinct strings.*
$\mathbf{x} \equiv_{\mathbf{L}} \mathbf{y} \iff \mathbf{x}, \mathbf{y}$ *are indistinguishable for* $\mathbf{L}$.

Proof.

$\mathbf{x} \equiv_{\mathbf{L}} \mathbf{y} \implies \forall \mathbf{w} \in \mathbf{\Sigma}^*\colon \mathbf{xw} \in \mathbf{L} \iff \mathbf{yw} \in \mathbf{L}$
$\mathbf{x}$ and $\mathbf{y}$ are indistinguishable for $\mathbf{L}$.

$\mathbf{x} \not\equiv_{\mathbf{L}} \mathbf{y} \implies \exists \mathbf{w} \in \mathbf{\Sigma}^*\colon \mathbf{xw} \in \mathbf{L}$ and $\mathbf{yw} \notin \mathbf{L}$
$\implies \mathbf{x}$ and $\mathbf{y}$ are distinguishable for $\mathbf{L}$.

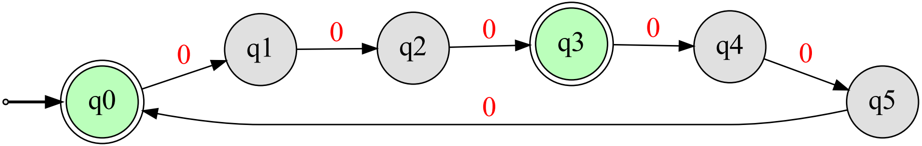
$\square$

All strings arriving at a state are in the same class

Lemma 6.8.

$\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{A})$ *a* DFA *for a language* $\mathbf{L}$.
For any $\mathbf{q} \in \mathbf{A}$, *let* $\mathbf{L_q} = \{\mathbf{w} \in \mathbf{\Sigma}^* \mid \nabla \mathbf{w} = \delta^*(\mathbf{s}, \mathbf{w}) = \mathbf{q}\}$.
Then, there exists a string $\mathbf{x}$, *such that* $\mathbf{L_q} \subseteq [\mathbf{x}]_{\mathbf{L}}$.

An inefficient automata



6.5.2
Stating and proving the Myhill-Nerode Theorem

Equivalences over automatas...

Claim 6.9 (Just proved).

Let $\mathbf{x}, \mathbf{y}$ be two distinct strings.
$\mathbf{x} \equiv_{\mathbf{L}} \mathbf{y} \iff \mathbf{x}, \mathbf{y}$ *are indistinguishable for* $\mathbf{L}$.

Corollary 6.10.

If $\equiv_{\mathbf{L}}$ *is finite with* $\mathbf{n}$ *equivalence classes then there is a fooling set* $\mathbf{F}$ *of size* $\mathbf{n}$ *for* $\mathbf{L}$.

Corollary 6.11.

If $\equiv_{\mathbf{L}}$ *has infinite number of equivalence classes* $\implies \exists$ *infinite fooling set for* $\mathbf{L}$.
$\implies \mathbf{L}$ *is not regular.*

Equivalence classes as automata

Lemma 6.12.

For all $\mathbf{x}, \mathbf{y} \in \mathbf{\Sigma}^*$, *if* $[\mathbf{x}]_{\mathbf{L}} = [\mathbf{y}]_{\mathbf{L}}$, *then for any* $\mathbf{a} \in \mathbf{\Sigma}$, *we have* $[\mathbf{xa}]_{\mathbf{L}} = [\mathbf{ya}]_{\mathbf{L}}$.

Proof.

$[\mathbf{x}] = [\mathbf{y}] \implies \forall \mathbf{w} \in \mathbf{\Sigma}^*\colon \mathbf{xw} \in \mathbf{L} \iff \mathbf{yw} \in \mathbf{L}$

$\implies \forall \mathbf{w}' \in \mathbf{\Sigma}^*\colon \mathbf{xaw}' \in \mathbf{L} \iff \mathbf{yaw}' \in \mathbf{L}$ // $\mathbf{w} = \mathbf{aw}'$

$\implies [\mathbf{xa}]_{\mathbf{L}} = [\mathbf{ya}]_{\mathbf{L}}$. $\square$

Set of equivalence classes

Lemma 6.13.

If $\mathbf{L}$ has $\mathbf{n}$ distinct equivalence classes, then there is a DFA *that accepts it using $\mathbf{n}$ states.*

Proof.

Set of states: $\mathbf{Q} = [\mathbf{L}]$
Start state: $\mathbf{s} = [\varepsilon]_{\mathbf{L}}$.
Accept states: $\mathbf{A} = \{[\mathbf{x}]_{\mathbf{L}} \mid \mathbf{x} \in \mathbf{L}\}$.
Transition function: $\delta([\mathbf{x}]_{\mathbf{L}}, \mathbf{a}) = [\mathbf{xa}]_{\mathbf{L}}$.
$\mathbf{M} = (\mathbf{Q}, \boldsymbol{\Sigma}, \delta, \mathbf{s}, \mathbf{A})$: The resulting DFA.
Clearly, $\mathbf{M}$ is a DFA with $\mathbf{n}$ states, and it accepts $\mathbf{L}$. $\square$

Myhill-Nerode Theorem

Theorem 6.14 (Myhill-Nerode).

$\mathbf{L}$ is regular $\iff$ $\equiv_{\mathbf{L}}$ has a finite number of equivalence classes.
If $\equiv_{\mathbf{L}}$ is finite with $\mathbf{n}$ equivalence classes then there is a DFA $\mathbf{M}$ accepting $\mathbf{L}$ with exactly $\mathbf{n}$ states and this is the minimum possible.

Corollary 6.15.

A language $\mathbf{L}$ is non-regular if and only if there is an infinite fooling set $\mathbf{F}$ for $\mathbf{L}$.

Algorithmic implication: For every DFA $\mathbf{M}$ one can find in polynomial time a DFA $\mathbf{M}'$ such that $\mathbf{L}(\mathbf{M}) = \mathbf{L}(\mathbf{M}')$ and $\mathbf{M}'$ has the fewest possible states among all such DFAs.

What was that all about

Summary: A regular language **L** has a unique (up to naming) minimal automata, and it can be computed efficiently once any DFA is given for **L**.

Exercise

1. Given two DFAs $\mathbf{M_1}, \mathbf{M_2}$ describe an efficient algorithm to decide if $\mathbf{L(M_1) = L(M_2)}$.
2. Given DFA $\mathbf{M}$, and two states $\mathbf{q}, \mathbf{q'}$ of $\mathbf{M}$, show an efficient algorithm to decide if $\mathbf{q}$ and $\mathbf{q'}$ are distinguishable. (Hint: Use the first part.)
3. Given a DFA $\mathbf{M}$ for a language $\mathbf{L}$, describe an efficient algorithm for computing the minimal automata (as far as the number of states) that accepts $\mathbf{L}$.

6.6
Roads not taken: Non-regularity via pumping lemma

Non-regularity via "looping"

Claim 6.1.

The language $\mathbf{L} = \{\mathbf{a^n b^n} \mid \mathbf{n} \geq 0\}$ *is not regular.*

Proof: Assume for contradiction $\mathbf{L}$ is regular.
$\implies \exists$ DFA $\mathbf{M} = (\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{q_0}, \mathbf{F})$ for $\mathbf{L}$. That is $\mathbf{L} = \mathbf{L(M)}$.
$\mathbf{n} = |\mathbf{Q}|$: number of states of $\mathbf{M}$.
Consider the string $\mathbf{a^n b^n}$. Let $\mathbf{p}_\tau = \delta^*(\mathbf{q_0}, \mathbf{a}^\tau)$, for $\tau = 0, \ldots, \mathbf{n}$.
$\mathbf{p_0 p_1 \ldots p_n}$: $\mathbf{n} + 1$ states. $\mathbf{M}$ has $\mathbf{n}$ states.
By pigeon hole principle, must be $\mathbf{i} < \mathbf{j}$, such that $\mathbf{p_i} = \mathbf{p_j}$.
$\implies \delta^*(\mathbf{p_i}.\mathbf{a^{j-i}}) = \mathbf{p_i}$ (its a loop!).
For $\mathbf{x} = \mathbf{a^i}$, $\mathbf{y} = \mathbf{a^{j-i}}$, $\mathbf{z} = \mathbf{a^{n-j} b^n}$, we have

$$\delta^*(\mathbf{q_0}, \mathbf{a^{n+j-i} b^n}) = \delta^*(\mathbf{q_0}, \mathbf{xyyz}) = \delta^*\Big(\delta^*\big(\delta^*(\delta^*(\mathbf{q_0}, \mathbf{x}), \mathbf{y}), \mathbf{y}\big), \mathbf{z}\Big)$$

Proof continued

Non-regularity via "looping"

We have: $p_i = \delta^*(q_0, a^i)$ and $\delta^*(p_i.a^{j-}) = p_i$.

$$\begin{aligned}
\delta^*(q_0, a^{n+j-i}b^n) &= \delta^*\left(\delta^*\left(\delta^*\left(\delta^*(q_0, a^i), a^{j-i}\right), a^{j-i}\right), a^{n-j}b^n\right) \\
&= \delta^*\left(\delta^*\left(\delta^*\left(\delta^*\left(p_i, a^{j-i}\right), a^{j-i}\right), a^{n-j}b^n\right)\right) \\
&= \delta^*\left(\delta^*\left(\delta^*\left(\delta^*(q_0, a^i), a^{j-i}\right), a^{n-j}b^n\right)\right) \\
&= \delta^*\left(\delta^*\left(\delta^*\left(p_i, a^{j-i}\right), a^{n-j}b^n\right)\right) \\
&= \delta^*(q_0, a^n b^n) \in A.
\end{aligned}$$

$\implies a^{n+j-i}b^n \in L$, which is false. Contradiction. $\square$

The pumping lemma

The previous argument implies that any regular language must suffer from loops (we omit the proof):

Theorem 6.2 (Pumping Lemma.).

Let $\mathbf{L}$ *be a regular language. Then there exists an integer* $\mathbf{p}$ *(the "pumping length") such that for any string* $\mathbf{w} \in \mathbf{L}$ *with* $|\mathbf{w}| \geq \mathbf{p}$, $\mathbf{w}$ *can be written as* $\mathbf{xyz}$ *with the following properties:*

- $|\mathbf{xy}| \leq \mathbf{p}$.
- $|\mathbf{y}| \geq \mathbf{1}$ *(i.e.* $\mathbf{y}$ *is not the empty string).*
- $\mathbf{xy^kz} \in \mathbf{L}$ *for every* $\mathbf{k} \geq \mathbf{0}$.