Proving Non-regularity

Lecture 6

Thursday, September 8, 2022

LaTeXed: September 13, 2022 10:01

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 M can be represented as a string over a finite alphabet Σ 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

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

Theorem 6.2.

L is not regular.

A Simple and Canonical Non-regular Language

$L = \{0^i1^i \mid i \geq 0\} = \{\epsilon, 01, 0011, 000111, \cdots, \}$

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

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

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

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

M should accept 0^i1^i but then it will also accept 0^j1^i where $i \neq j$.
This contradicts the fact that M accepts L. Thus, there is no DFA for L.

6.2
When two states are equivalent?

Equivalence between states

Definition 6.1.

$M = (Q, \Sigma, \delta, s, A)$: DFA.
Two states $p, q \in Q$ *are* **equivalent** *if for all strings* $w \in \Sigma^*$*, we have that*

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

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

Distinguishing between states

Definition 6.2.

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

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

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

or

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

Distinguishable prefixes

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

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

Definition 6.3.

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

Definition 6.4 (Direct restatement).

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

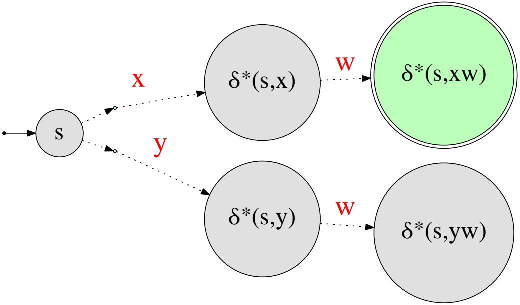
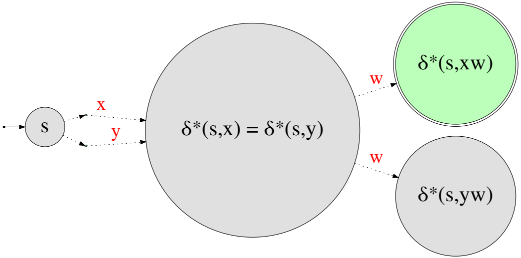
Distinguishable means different states

Lemma 6.5.

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

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

Proof by a figure

Possible	Not possible
	

Distinguishable strings means different states: Proof

Lemma 6.6.

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

Proof.

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

Review questions...

1. Prove for any $i \neq j$ then $\mathbf{0}^i$ and $\mathbf{0}^j$ are distinguishable for the language $\{\mathbf{0}^k\mathbf{1}^k \mid k \geq \mathbf{0}\}$.
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 $\{\mathbf{0}^k\mathbf{1}^k \mid k \geq \mathbf{0}\}$ is not regular.

6.3
Fooling sets: Proving non-regularity

Fooling Sets

Definition 6.1.

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

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

Theorem 6.2.

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

Recall

Already proved the following lemma:

Lemma 6.3.

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

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

Proof of theorem

Theorem 6.4 (Reworded.).

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

Proof.

Let $F = \{w_1, w_2, \ldots, w_m)$ be the fooling set.
Let $M = (Q, \Sigma, \delta, s, A)$ be any DFA that accepts L.
Let $q_i = \nabla w_i = \delta^*(s, x_i)$.
By lemma $q_i \neq q_j$ for all $i \neq j$.
As such, $|Q| \geq |\{q_1, \ldots, q_m\}| = |\{w_1, \ldots, w_m\}| = |F|$. □

Infinite Fooling Sets

Corollary 6.5.

If L has an infinite fooling set F then L is not regular.

Proof.

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

Examples

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

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

$\left\{ \mathbf{0}^k \;\middle|\; k \text{ is a prime number} \right\}$

Hints:

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

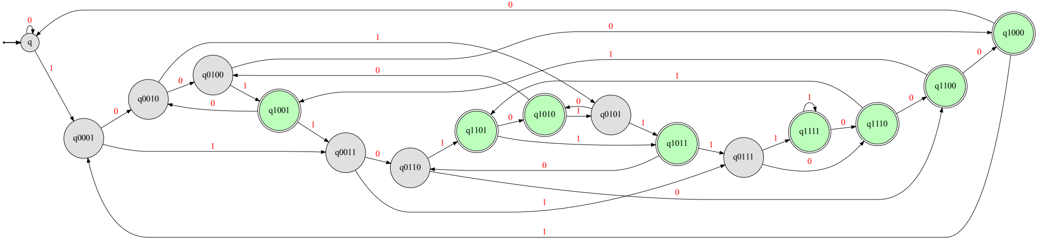
6.3.1

Exponential gap in number of states between DFA and NFA sizes

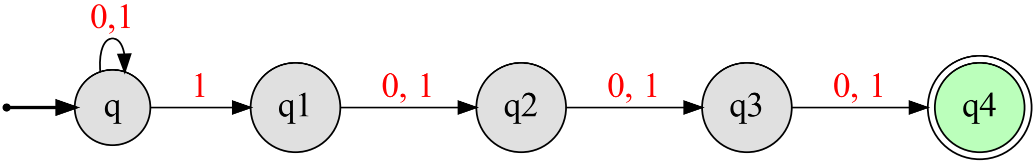
Exponential gap between NFA and DFA size

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

DFA:



NFA:



Exponential gap between NFA and DFA size

$L_k = \{w \in \{0,1\}^* \mid w$ has a 1 k 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 $a_1 a_2 \ldots a_k$ and $b_1 b_2 \ldots b_k$ 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 F?

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

6.4
Closure properties: Proving non-regularity

Non-regularity via closure properties

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

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

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

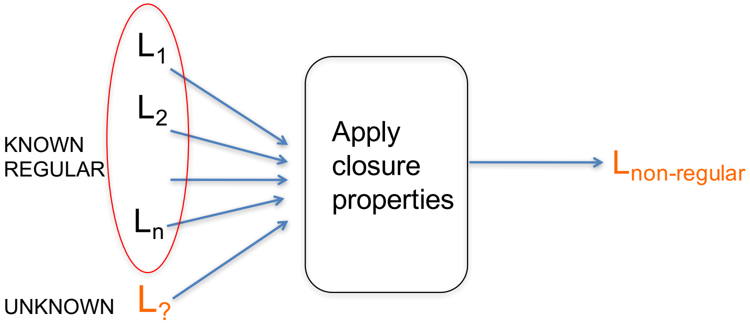
$H' = H \cap L(\mathbf{0}^*\mathbf{1}^*)$

Claim: The above and the fact that L' is non-regular implies L is non-regular. Why?

Suppose H is regular. Then since $L(\mathbf{0}^*\mathbf{1}^*)$ is regular, and regular languages are closed under intersection, H' also would be regular. But we know 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.

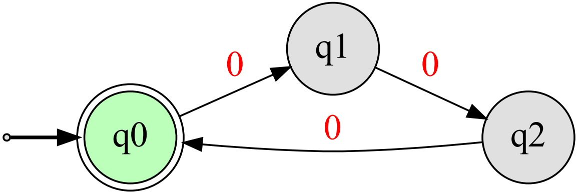
For a language L over Σ and two strings $x, y \in \Sigma^*$ we say that x and y are distinguishable with respect to L if there is a string $w \in \Sigma^*$ such that exactly one of xw, yw is in L. x, y are indistinguishable with respect to L if there is no such w.

Given language L over Σ define a relation $\equiv_L$ over strings in Σ^* as follows: $x \equiv_L y$ iff x and y are indistinguishable with respect to L.

Definition 6.2.

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

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

Therefore, $\equiv_L$ partitions Σ^* into a collection of equivalence classes.

Definition 6.5.

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

$$[x] = [x]_L = \{ y \in \Sigma^* \mid x \equiv_L y \}$$

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

Definition 6.6.

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

Strings in the same equivalence class are indistinguishable

Lemma 6.7.

Let x, y be two distinct strings.
$x \equiv_L y \iff x, y$ *are indistinguishable for* L.

Proof.

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

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

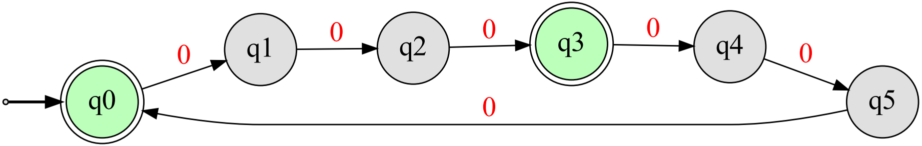
□

All strings arriving at a state are in the same class

Lemma 6.8.

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

An inefficient automata



6.5.2
Stating and proving the Myhill-Nerode Theorem

Equivalences over automatas...

Claim 6.9 (Just proved).

Let x, y be two distinct strings.
$x \equiv_L y \iff x, y$ *are indistinguishable for* L.

Corollary 6.10.

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

Corollary 6.11.

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

Equivalence classes as automata

Lemma 6.12.

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

Proof.

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

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

$\implies [xa]_L = [ya]_L$. □

Set of equivalence classes

Lemma 6.13.

If L has n distinct equivalence classes, then there is a DFA that accepts it using n states.

Proof.

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

Myhill-Nerode Theorem

Theorem 6.14 (Myhill-Nerode).

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

Corollary 6.15.

A language L is non-regular if and only if there is an infinite fooling set F for L.

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

What was that all about

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

Exercise

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

6.6
Roads not taken: Non-regularity via pumping lemma

Non-regularity via "looping"

Claim 6.1.

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

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

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

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

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