Theory of Computation

The Myhill-Nerode Theorem

Guest lecture delivered by Cesare Spinoso-Di Piano
Material developed by Prof. Prakash Panangaden

Plan for today

1. Motivation (1)
2. Definitions and lemmas
3. The Myhill-Nerode Theorem
4. Motivation (2)
5. Connection with minimal DFAs

Motivation (1)

What you've seen so far

$\Sigma \neq \emptyset$, $L \subseteq \Sigma^*$ is regular if and only if

How do we prove a language is not regular?

When the PL fails...

Show that the language $L = \{a^i b^j c^k : \text{if } i = 1 \text{ then } j = k\}$ is not regular.

Is all hope lost?

No!

Use closure properties.

Use the Myhill-Nerode Theorem (today).

Definitions and lemmas

Recall - String concatenation

Definition (String concatenation). Let $\Sigma \neq \emptyset, x, y \in \Sigma^*$. The **string concatenation** written either as xy or $x \cdot y$ is the operation of appending the string y to the string x.

Example. $\Sigma = \{a, b, c, \ldots, z\}$, $x = cat$, $y = dog$ then $x \cdot y = catdog$.

Properties of concatenation.

Associativity

Unit element

Together with a (non-empty) alphabet, the string concatenation forms a **monoid**.

Monoid

Definition (Monoid). A **monoid** is a set S with a binary associative operation and an identity element for this operation. Sometimes denoted as the triple $(S, \cdot, e)$.

Examples.

$$(\mathbb{N}, +, 0)$$

$$(\Sigma \neq \emptyset, \cdot, \epsilon)$$

$$(S \to S, \circ, \mathrm{id})$$

Recall - Equivalence relations

Definition (Equivalence relation). Given the set X, the relation $X \times X$ is an **equivalence relation** if it is **reflexive**, **symmetric** and **transitive**.

Definition (Equivalence class). Given the set X, the **equivalence class** of $x \in X$ for the equivalence relation R is the set $[x] := \{y \in X : xRy\}$. The set of equivalence classes is denoted X/R.

Definition (Index of an equivalence relation). Given the set X, the number of equivalence classes for the equivalence relation R is called the **index** of R.

Example. $X = \mathbb{Z}$, $\forall x, y \in X, xRy \iff x \mod 5 = y \mod 5$. What is the index of R?

When concatenation and equivalence relations interact

Definition (Right invariance). An equivalence relation R on Σ^* is said to be **right-invariant** if

$$\forall x, y \in \Sigma^*, xRy \Rightarrow \forall z \in \Sigma^*, xzRyz$$

That is, if a pair of strings are related and we stick a string to the right of each of them, then this new pair of strings will also be related. And this will hold *for any string used to stick to the original pair of strings*.

The "extended" transition function δ^*

For a DFA $M = (Q, \Sigma, q_0, \delta, F)$ we defined $\delta^* : Q \times \Sigma^* \to Q$ inductively as

$$\forall q \in Q, \delta^*(q, \epsilon) = q \text{ and } \forall a \in \Sigma, x \in \Sigma^*, \delta^*(q, ax) = \delta^*(\delta(q, a), x)$$

You can show by induction (on the length of the string) that

$$\forall x, y \in \Sigma^*, \delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$$

δ^* as an equivalence relation

Definition (R_M). For a fixed DFA $M = (Q, \Sigma, q_0, \delta, F)$, define the relation on Σ^*, denoted R_M, as follows

$$\forall x, y \in \Sigma^*, x R_M y \iff \delta^*(q_0, x) = \delta^*(q_0, y)$$

That is, for a given DFA M, two strings are related if M ends at the same state when reading both of them.

Claim 1. R_M is an equivalence relation. Why?

Claim 2. R_M is a **right-invariant** equivalence relation.

Proof.

This right-invariant equivalence relation will come in handy in a few slides from now.

Another very familiar equivalence relation

This equivalence relation will be based on **ANY** language (not just regular languages).

Definition. Given any language $L \subseteq \Sigma^*$, we define a relation $\equiv_L$ on Σ^* as follows

$$x \equiv_L y \iff \forall z \in \Sigma^*, xz \in L \iff yz \in L$$

Claim 1. $\equiv_L$ is an equivalence relation. Prove it!

Claim 2. For any two related strings x, y, they are **either** both in L **or** neither of them is in L. Why?

Example. $\Sigma = \{0, 1\}, L = \{w \in \Sigma^* : |w| \mod 2 = 0\}$

$0 \equiv_L 00$?

$10 \equiv_L 01$?

A consequential lemma

Lemma. The equivalence relation $\equiv_L$ is right-invariant.

Proof.

This lemma will be **crucial** in the proof of the following theorem.

The Myhill-Nerode Theorem

The main theorem

Theorem (Myhill-Nerode). The following three statements are equivalent:

(1) The language L is accepted by a DFA.

(2) The language L is equal to the union of *some* equivalence classes for *some* right-invariant equivalence relation of finite index.

(3) The equivalence relation $\equiv_L$ has finite index. In fact, any right-invariant equivalence relation R with the property that L is the union of some of the equivalence classes of R will have index greater than $\equiv_L$. (This will come in handy when proving uniqueness of minimality.)

What is the theorem telling us?

We will prove this by showing $(1) \Rightarrow (2), (2) \Rightarrow (3), (3) \Rightarrow (1)$.

Proof - (1) $\Rightarrow$ (2)

Statement: The language L is accepted by a DFA. $\Rightarrow$ The language L is equal to the union of *some* equivalence classes for *some* right-invariant equivalence relation of finite index.

Proof - (2) $\Rightarrow$ (3)

Statement: The language L is equal to the union of *some* equivalence classes for *some* right-invariant equivalence relation of finite index. $\Rightarrow$ The equivalence relation $\equiv_L$ has finite index. (Plus some other stuff)

Proof - $(3) \Rightarrow (1)$

Statement: The equivalence relation $\equiv_L$ has finite index. $\Rightarrow$ The language L is accepted by a DFA.

Motivation (2)

Who cares?

Recall(!) the language $L = \{a^i b^j c^k : \text{if } i = 1 \text{ then } j = k\}$. We can use M-N to *easily* prove that L is not regular.

How?

1. Pick an infinite set of strings S
2. Show that $\forall x, y \in S, x \neq y \Rightarrow x \not\equiv_L y$
3. This implies that each element in S belongs to a different equivalence class of $\equiv_L$.
4. Therefore $\equiv_L$ is not finite, so by M-N L is not regular!

Example - Continued

Showing that $L = \{a^i b^j c^k : \text{if } i = 1 \text{ then } j = k\}$ is not regular.

Exercise

Exercise. Using M-N, show that $L = \{a^n b^{n^2} : n \geq 0\}$ is not regular.

Exercise. Using M-N, show that $L = \{w \in \{0,1\}^* : |w| \bmod 3 = 0\}$ is regular. (Hint: $\equiv_L$ should have index 3.)

Connection with minimal DFAs

Uniqueness of minimal DFA

- A few lectures ago we saw a DFA minimization algorithm that we claimed (without proof) produced the **unique** minimal DFA.
- We want to show that the algorithm could not have stumbled on a *different* minimal DFA that accepted the same language.
- To do this, we first have to define what it means for a DFA to be the same as (and, by extension, different than) another DFA.

NOTE This is not true for NFAs! Two NFAs can be “minimal“ while being completely “different”.

Isomorphic DFAs

We call the concept of two DFAs being the same a DFA "isomorphism".[1]

Definition (DFA isomorphism). We say two DFAs $M = (Q, \Sigma, q_0, \delta, F)$ and $M' = (Q', \Sigma, q_0', \delta', F')$ are **isomorphic** if there is a bijection ϕ where $\phi : Q \to Q'$ such that

1. $\phi(q_0) = q_0'$
2. $\phi(\delta(q, a)) = \delta'(\phi(q), a)$
3. $q \in F \iff \phi(q) \in F'$

Fact. If $f : X \to Y$ and $|X| = |Y|$ then f is injective $\iff$ f is surjective. (Why? Try induction on $|X| = |Y|$).

[1] Different mathematical objects have different definitions of isomorphism, for instance, graph isomorphisms.

Uniqueness proposition

The following proposition will allow us to show that the minimal DFA found in the DFA minimization lecture was unique *up to isomorphism*.

Proposition. The machine described in the last part of the M-N proof ($(3) \Rightarrow (1)$) is the *unique* minimal DFA that recognizes the language L.

Proof of the proposition

Proof.

Again, so what?

You should now be able to answer the following questions:

- How do I create an algorithm (and prove its correctness) that checks whether two NFAs N_1, N_2 accept the same language?
- How do I create an algorithm (and prove its correctness) that checks that two regular expressions R_1, R_2 recognize the same language?