4CS4-6: Theory of Computation (Myhill-Nerode Theorem)

Lecture 11: Feb. 24, 2020

Prof. K.R. Chowdhary *: Professor of CS*

Disclaimer: *These notes have not been subjected to the usual scrutiny reserved for formal publications. They may be distributed outside this class only with the permission of the Instructor.*

11.1 Introduction

The idea discussed earlier was to repeatedly combine indistinguishable states, i.e. if p is indistinguishable with q, combine it with q. Similarly, if r is indistinguishable with q then combine it with q, and so on. But suppose that p, q are indistinguishable, and q and r are indistinguishable. We want to combine p with q, and we also want to combine q with r. Then, it is would be better to consider that p and r are also indistinguishable. In fact, they are. This property is formalized in the following lemma.

Lemma 11.1 *State Indistinguishability is an equivalence relation.*

Proof: Consider that p, q, r are the states in a finite automaton. We write $p \approx q$ if p is indistinguishable from q. We need to check the following three conditions for indistinguishability.

Reflexivity: $p \approx p$.

Symmetry: $p \approx q$ implies $q \approx p$.

Transitivity: $p \approx q$ and $q \approx r$ implies $p \approx r$.

All conditions are trivially true, directly from the definition of the indistinguishability relation. ■

Lemma 11.2 *Let $\delta(p, a) = p'$ and $(q, a) = q'$, for any $a \in \Sigma$. Then, if p' and q' are distinguishable then so are p and q.*

Proof: This is simple, for if p and q are distinguished by some string w, then p', q' are distinguished by string wa. ■

Definition 11.3 Equivalent strings. *For a given DFA, $M_D = (\Sigma, Q, q_0, \delta, F)$, two strings $x, y \in \Sigma^*$ are called equivalent with respect to M_D, if x and y both drive M_D from state q_0 to the same state q', i.e.,*

$$\delta^*(q_0, x) = q',$$

and

$$\delta^*(q_0, y) = q'.$$

This relation is represented by $x \approx_{M_D} y$.

All the strings which drive M_D into same state are equivalent, and are related by the equivalence relation $\approx_L$.

11.2 Myhill-Nerode Theorem and its Applications

The pumping lemma holds for some non-regular languages only and does not provide sufficient condition to prove that a language is regular. In fact, if pumping lemma fails to prove nonregularity of a language, it does not mean the otherwise, that is, the language is regular. For example, consider the language $\{uu^R v \mid u, v \in \{0,1\}^+\}$, where strings over the alphabet $\{0,1\}$ consists of nonempty even-palindrome followed by another nonempty string. This language is not regular, but can be still pumped with $|w| = |uu^R v| \geq 4$ as follows: If $|u| = 1$ then $|v| \geq 2$ and we can take y to be the first character of v. Otherwise, take y to be the first character in u and note that y^k for $k \geq 2$ starts with the nonempty palindrome yy.

The *Myhill-Nerode* (MN) theorem provides a necessary and sufficient condition for a language to be regular. Following are some definitions, having understood these, we will be better equipped to make use of interesting applications of MN theorem..

Given a language L, and a pair of strings x and y, a *distinguishing extension* is a string z such that exactly one of the two strings xz or yz belong to L. Based on this, a relation $\approx_L$ is defined on strings as follows.

In the following we present more terminology, necessary for understanding the applications of MN theorem.

Definition 11.4 Distinguishing strings. *Let $M = (Q, \Sigma, \delta, s, F)$ be a finite automaton. The string $w \in \Sigma^*$ is a distinguishing string for states $p, q \in Q$ if exactly one of the states, $\delta^*(p, w)$, $\delta^*(q, w)$ is in F.*

Intuitively, a distinguishing string w for states p and q in an automaton M is a string mapped to an accepting state from exactly one of p and q. In other words, the existence of a distinguishing string for a pair of states p, $q \in Q$ shows that states p and q are not equivalent.

Definition 11.5 Right Invariant. *An equivalence relation $\approx_L$ on strings of symbols from some alphabet Σ is said to be right invariant if $\forall x, \forall y$ $x, y \in \Sigma^*$ with $x \approx_L y$ and all $w \in \Sigma^*$ there is always $xw \approx_L yw$.*

The definition states that an equivalent relation has the right invariant property if two equivalent strings (x and y) that are in the language are still equivalent if a third string (w) is appended to the right of both of them.

Definition 11.6 Equivalence relation for Languages. *Let L be a language over Σ. Two words $x, y \in \Sigma^*$ are called L-equivalent, written as $x \approx_L y$, if and only if there is no distinguishing extension $z \in \Sigma^*$, so that $xz \in L \Leftrightarrow yz \in L$.*

This relation is also called as Canonical Equivalent Relation $\approx_L$ *on* Σ^*.

For $x, y \in \Sigma^*$, there is $x \approx_L \ y$ if and only if there is no distinguishing extension z for x and y. The idea behind the distinguishing extension is that, a given automaton M cannot distinguish between strings x and y. It can be easily shown that $\approx_L$ is an equivalence relation on strings for M, and thus it divides the set of all finite strings, $w \in \Sigma^*$, into equivalent classes.

Definition 11.7 Equivalence relation for DFAs. *Let M be a DFA. For strings $x, y \in \Sigma^*$, let there is $x \approx_M y$ if and only if the same state in the DFA is reachable from the initial state by reading each of the strings x and y.*

Definition 11.8 Equivalence classes Index. *An equivalent relation '$\approx$', on a set Σ^* imposes a partition $\Sigma^*/\approx$ on Σ^*, such that elements $[x], [y] \in \Sigma^*$ are in the same part of $\Sigma^*/\approx$ if $[x] \approx [y]$. The parts of Σ^* are called equivalent classes, and we write $[x] \approx$ to denote the equivalence class of $\Sigma^*/\approx_x$ to which x belongs. The index of equivalent relation $\approx$ is $|\delta/\approx|$; which is total number of unique equivalent classes.*

Definition 11.9 Canonical DFA. *$M \approx_L$ is canonical DFA for language L, with finite index $\equiv_L$, defined as follows:*

The canonical DFA $M_L = (Q_L, \Sigma, q_{0L}, \delta_L, F_L)$ for L is defined as:

$Q_L = \{[x] \mid x \in \Sigma^*\}$, *(the finite set of $\approx_L$-classes),*

$q_{0L} = \varepsilon$,

$\delta_L([x], a) = [xa]$, *and*

$F_L = \{[x] \mid x \in L\}$.

From above we note that since L is a regular language there exists a DFA $M = (Q, \Sigma, \delta, s, F)$ that accepts L. We know that $\approx_M$ is a right invariant relation of finite index, hence

$$L = \bigcup_{x:\delta(q_0,x)\in F} [x], \tag{11.1}$$

where $x \in \Sigma^*$.

11.3 Myhill-Nerode Theorem

Definition 11.10 *For $L \subseteq \Sigma^*$, the word equivalence is a right congruence, i.e., $\approx_L$ has the following two properties:*

1. *if $x \approx_L y$ and $x \in L$, then $y \in L$,*
2. *if $x \approx_L y$ iff $xz \approx_L yz$, for all $z \in \Sigma^*$.*

This means that for any future string z, the x and y need not to be distinguished with respect to L. We often use the negation of this equivalence to prove that two words belong to two different equivalence classes, i.e., $x \not\approx_L y$ iff there exists z with $xz \in L$ and $yz \notin L$ (or $xz \notin L$ and $yz \in L$).

The above relations are called *Myhill-Nerode* relations (also *Nerode Congruence*), are proved below, as theorem's part I and II. However, we will be proving the part-I only.

Theorem 11.11 *(Myhill-Nerode theorem Part I.) If index of a language L is k, then there is a k-state DFA M such that $L(M) = L$ (that is, every language with finite index is regular).*

Proof: To show this, let $L \subseteq \Sigma^*$ be a language with index k. Define the following finite automaton $M_L = (Q, \Sigma, \delta, q_0, F)$ whose states are the L-equivalence classes, and defined as,

$$\begin{aligned}
&Q = \{[x] \approx_L \mid [x] \subseteq \Sigma^*\},\\
&\delta \in \delta(p, a) \text{ iff there exists a word } x \in p \text{ such that } xa \in q,\\
&q_0 = [\varepsilon] \approx_L,\\
&q \in F \text{ iff there exists a word } x \in q \text{ such that } x \in L.
\end{aligned}$$

Since, for the language L with finite index k there exist a finite automaton of k-state DFA that accepted it, hence the language L is regular. This proves the theorem.

It may be noted that the table-filling algorithm discussed earlier is based on above definitions and lemmas.

Example 11.12 *Show that the set $R = \{a^n b^n \mid n \geq 0\}$ is non-regular.*

The Myhill-Nerode theorem can be used to show whether R is regular or non-regular by determining number of $\approx_R$-equivalence classes. If $k \neq m$, then $a^k \not\approx_R a^m$, since $a^k b^k \in R$ but $a^m b^k \notin R$. Therefore, there are infinitely many $\approx_R$-classes, at least one for each $a^k, k \geq 0$. By the Myhill-Nerode theorem, R is not regular.

Review Questions

1. For a given regular expression there can be more than one DFA, which recognize the corresponding language (True/False).

2. For a given DFA there can be only one regular expression (True/False).
3. What is maximum number of states in an equivalent DFA for an NFA of 4 states?
4. Is it possible to determine approximate number of the states in the NFA if an equivalent DFA has n states? Justify your answer for Yes or No.
5. Two states in a DFA are indistinguishable from each other if their behavior is indistinguishable from each other for a particular input string (True/false).
6. What conclusions you draw from the following definitions ? Are these deductions valid?

   (a) $\delta^*(p, u) \in F \Rightarrow \delta^*(q, u) \in F$

   (b) $\delta^*(p, v) \notin F \Rightarrow \delta^*(q, u) \notin F$

7. If p and q are two indistinguishable states in a finite automata, then one of these can be removed including its incoming and outgoing edges, without effecting its language recognition capabilities of the FA (True/False).
8. Pumping-lemma is only sufficient for ttesting non-regularity of all languages (True/False).