CS 172: Computability and Complexity Fall 2003

Lecture 7: Myhill-Nerode Theorem

7.1 Word equivalence. Consider a language L over an alphabet Σ. Two words $x, y \in \Sigma^*$ are *L-equivalent*, written $x \equiv_L y$, iff for all words $z \in \Sigma^*$, we have $xz \in L$ iff $yz \in L$. For example, if $\Sigma = \{\mathtt{0}, \mathtt{1}\}$ and $A = \Sigma^*\mathtt{0}\Sigma$, then $\equiv_A$ has four equivalence classes:

$A_1 = \Sigma^*\mathtt{00}$
$A_2 = \Sigma^*\mathtt{01}$
$A_3 = \Sigma^*\mathtt{10} \cup \mathtt{0}$
$A_4 = \Sigma^*\mathtt{11} \cup \mathtt{1} \cup \varepsilon$

If $\Sigma = \{\mathtt{0}, \mathtt{1}\}$ and $B = \{\mathtt{0}^n\mathtt{1}^n : n \geq 0\}$, then $\equiv_B$ has infinitely many equivalence classes:

$B_- = \{\mathtt{0}^n\mathtt{1}^m : m > n \geq 0\} \cup \Sigma^*\mathtt{1}\Sigma^*\mathtt{0}\Sigma^*$
$B_0 = \{\mathtt{0}^n\mathtt{1}^n : n \geq 0\}$
$B_1 = \{\mathtt{0}^n\mathtt{1}^{n-1} : n \geq 1\}$
$B_2 = \{\mathtt{0}^n\mathtt{1}^{n-2} : n \geq 2\}$
$B_3 = \{\mathtt{0}^n\mathtt{1}^{n-3} : n \geq 3\}$
$\vdots$

The number of equivalence classes of $\equiv_L$ is called the *index* of the language L. In particular, the index of A is 4, and the index of B is ∞.

7.2 Word equivalence is a right congruence. The equivalence $\equiv_L$ has the following two properties:

(1) If $x \equiv_L y$ and $x \in L$, then $y \in L$.
(2) If $x \equiv_L y$, then $xa \equiv_L ya$.

They are immedite consequences of the definition.

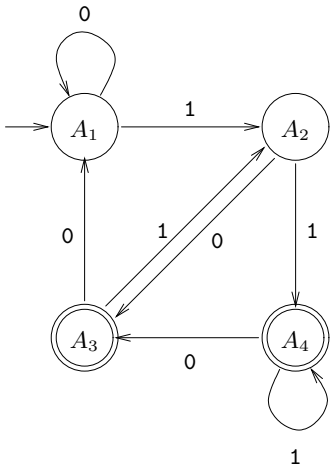
7.2 Myhill-Nerode theorem, part 1. The equivalence relation $\equiv_L$ characterizes exactly what the state of an automaton that defines L needs to remember about the read portion of the input: if the read portion of the input is x, then the state needs to remember the equivalence class $[x]_{\equiv_L}$. This is sufficient, because if $x \equiv_L y$, then it does not matter if the read portion of the input was x or y; all that matters for deciding whether to accept or reject is the future portion of the input, say z, because $xz \in L$ iff $yz \in L$. It is also necessary, because if $x \not\equiv_L y$, then there is some possibe future portion z of the input such that xz needs to be accepted and yz rejected (or vice versa). This is formalized by the following two theorems.

> **Theorem 7A.** If the index of a language A is k, then there is a k-state DFA M_A such that $L(M_A) = A$.

To see this, let $A \subseteq \Sigma^*$ be a language with index k. Define the following finite automaton $M_A = (Q, \Sigma, \delta, q_0, F)$, whose states the $\equiv_A$-equivalence classes:

$Q = \{[x]_{\equiv_A} : x \in \Sigma^*\}$.
$q \in \delta(p, a)$ iff there exists a word $x \in p$ such that $xa \in q$.
$q_0 = [\varepsilon]_{\equiv_A}$.
$q \in F$ iff there exists a word $x \in q$ such that $x \in A$.

From (2) it follows that M_A is deterministic. From (1) it follows that for all states $q \in Q$, either $q \cap A = \emptyset$ or $q \subseteq A$. Therefore, for every word $x \in \Sigma^*$, after reading input x, the DFA M_A ends up in the state $[x]_{\equiv_A}$, which accepts if $x \in A$ and rejects if $x \notin A$. For $A = \Sigma^*\mathtt{0}\Sigma$:



It can be shown that for every DFA D, the quotient automaton $D/_{\approx}$ is equal (up to renaming of states) to the DFA $M_{L(D)}$.

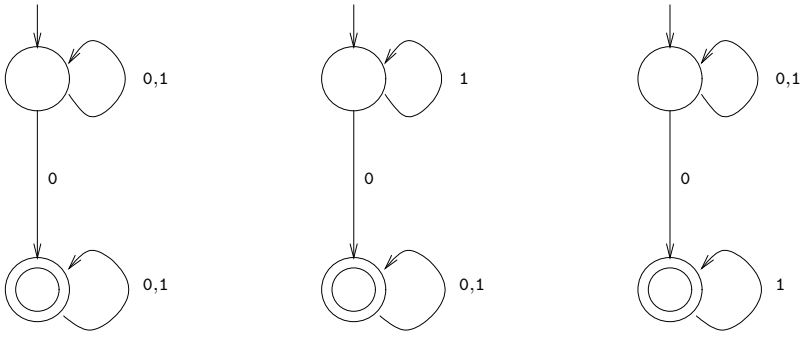
7.3 Myhill-Nerode theorem, part 2. In the previous section we saw that every language with finite index is regular; now we will see that every regular language has a finite index.

Theorem 7B. For every k-state DFA M, the index of $L(M)$ is at most k.

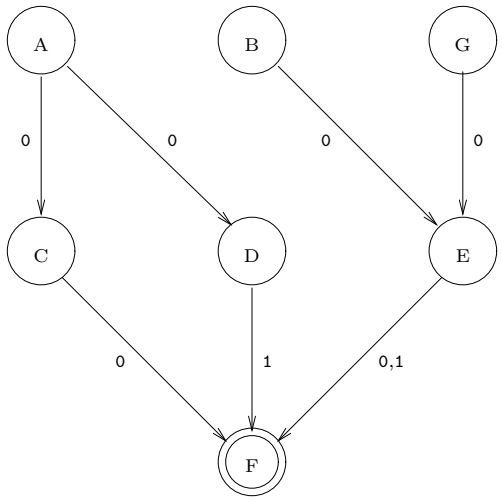
To see this, consider a DFA M over the alphabet Σ. For two words $x, y \in \Sigma^*$, let $x \equiv_M y$ iff after reading input x, the DFA M ends up in the same state as after reading input y. The number of $\equiv_M$-equivalence classes is equal to the number of states of M, say k. If $x \equiv_M y$, then $x \equiv_{L(M)} y$, because inputs xz and yz always lead to the same state. Hence there can be no more $\equiv_{L(M)}$-equivalence classes than there are $\equiv_M$-equivalence classes; that is, the index of $L(M)$ is at most k.

Theorem 7B can be used to show that a language is nonregular: it suffices to argue that its index is ∞. Theorem 7B can be strengthened to show that if M is reduced (i.e., $M = M/_{\approx}$), then the index of $L(M)$ is exactly k. From that and Section 7.1 it follows that for every regular language L there is a unique (up to renaming of states) minimal DFA (with the fewest number of states), and that this DFA can be obtained by applying the quotient construction to *any* DFA for L.

7.4 Nondeterminism and minimality. There may not be a unique minimal NFA for a regular language. For example, the following three automata all define the same language, namely $\Sigma^*0\Sigma$, and are all reduced:



7.5 Bisimilarity. For DFAs, if two states p and q are equivalent, then for all input letters a, also $\delta(p, a)$ and $\delta(q, a)$ are equivalent. By contrast, for NFAs, two states p and q may be equivalent even though there are no equivalent states in $\delta(p, a)$ and $\delta(q, a)$. For instance, for the following NFA N, states A and B are equivalent even though no two of C, D, and E are equivalent:



$L(N_{\mathrm{A}}) = L(N_{\mathrm{B}}) = L(N_{\mathrm{G}}) = \{\mathtt{00}, \mathtt{01}\}$
$L(N_{\mathrm{C}}) = \{\mathtt{0}\}$
$L(N_{\mathrm{D}}) = \{\mathtt{1}\}$
$L(N_{\mathrm{E}}) = \{\mathtt{0}, \mathtt{1}\}$
$L(N_{\mathrm{F}}) = \{\varepsilon\}$

For NFAs we can define a stronger concept than state equivalence called state bisimilarity, which corresponds to "recursive state equivalence": two states p and q of an NFA $M = (Q, \Sigma, \delta, q_0, F)$ are *bisimilar*, written $p \approx^B_M q$, iff the following three conditions are satisfied:

(1) $p \approx_M q$ (this can be weakened to $p \approx^0_M q$).
(2) For all $a \in \Sigma$ and all $p' \in \delta(p, a)$, there exists a state $q' \in \delta(q, a)$ such that $p' \approx^B_M q'$.
(3) For all $a \in \Sigma$ and all $q' \in \delta(q, a)$, there exists a state $p' \in \delta(p, a)$ such that $p' \approx^B_M q'$.

For the example automaton N, we have $\textsc{a} \not\approx^B_N \textsc{b}$ but $\textsc{b} \approx^B_N \textsc{g}$.

7.6 Minimization of nondeterministic automata. When applied to NFAs, the minimization algorithm (suitably adjusted) computes state bisimilarity, not state equivalence. Recall the NFA N from Section 7.5:

A							
B	2						
C	1	1					
D	1	1	1				
E	1	1	1	1			
F	0	0	0	0	0		
G	2		1	1	1	0	
	A	B	C	D	E	F	G