Myhill-Nerode Theorem

Deepak D’Souza

Department of Computer Science and Automation
Indian Institute of Science, Bangalore.

23 August 2012
Outline

1. Overview
2. Myhill-Nerode Theorem
3. Correspondence between DA’s and MN relations
4. Canonical DA for $L$
5. Computing canonical DFA
Myhill-Nerode Theorem: Overview

- Every language $L$ has a “canonical” deterministic automaton accepting it.
  - Every other DA for $L$ is a “refinement” of this canonical DA.
  - There is a unique DA for $L$ with the minimal number of states.
- Holds for any $L$ (not just regular $L$).
- $L$ is regular iff this canonical DA has a finite number of states.
- There is an algorithm to compute this canonical DA from any given finite-state DA for $L$. 
Illustrating “refinement” of DA: Example 1

Every DA for $L$ is a “refinement” of this canonical DA:
Every DA for $L$ is a “refinement” of this canonical DA:
Myhill-Nerode Theorem

Canonical equivalence relation $\equiv_L$ on $A^*$ induced by $L \subseteq A^*$:

$$x \equiv_L y \text{ iff } \forall z \in A^*, \, xz \in L \text{ iff } yz \in L.$$ 

$x \not\equiv_L y$ iff

Theorem (Myhill-Nerode)

$L$ is regular iff $\equiv_L$ is of finite index (that is has a finite number of equivalence classes).
Exercise 1

Describe the equivalence classes for $L = \text{“Odd number of } a\text{’s”}$.
Exercise 2

Describe precisely the equivalence classes of $\equiv_L$ for the language $L \subseteq \{a, b\}^*$ comprising strings in which 2nd last letter is a $b$. 
Exercise 2

Describe precisely the equivalence classes of $\equiv_L$ for the language $L \subseteq \{a, b\}^*$ comprising strings in which 2nd last letter is a $b$. 

![Diagram of DFA](image)
Exercise 3

Describe the equivalence classes of \( \equiv_L \) for the language

\[ L = \{ a^n b^n \mid n \geq 0 \}. \]
Exercise 3

Describe the equivalence classes of $\equiv_L$ for the language $L = \{a^n b^n \mid n \geq 0\}$. 

Note: The natural deterministic PDA for $L$ gives this DA.
Exercise 3

Describe the equivalence classes of $\equiv_L$ for the language $L = \{a^n b^n \mid n \geq 0\}$.

Note: The natural deterministic PDA for $L$ gives this DA.
Myhill-Nerode (MN) relations for a language

- An **MN relation** for a language \( L \) on an alphabet \( A \) is an equivalence relation \( R \) on \( A^* \) satisfying
  1. \( R \) is right-invariant (i.e. \( xRy \implies xaRya \) for each \( a \in A \).)
  2. \( R \) refines (or “respects”) \( L \) (i.e. \( xRy \implies x, y \in L \) or \( x, y \notin L \)).
Deterministic Automata for $L$ and MN relations for $L$

DA for $L$ and MN relations for $L$ are in 1-1 correspondence (they represent each other).

Maps $A \rightarrow R_A$ and $A_R \leftarrow R$ are inverses of each other.
Example DA and its induced MN relation

$L$ is “Odd number of $a$’s”:

- **Example DA:**
  - States: $a$, $b$
  - Transitions: $a \rightarrow a$, $b \rightarrow b$, $a \rightarrow b$, $b \rightarrow a$

- **MN Relation $R_A$:**
  - Equivalence classes:
    - $\epsilon$: $\{a\}$
    - $a$: $\{aa, aaa\}$
    - $b$: $\{ab, baa\}$

- **Canonical DFA:**
  - States: $\epsilon, a, \epsilon, a$
  - Transitions: $\epsilon \rightarrow a$, $a \rightarrow a$, $\epsilon \rightarrow \epsilon$, $a \rightarrow a$

- **MN Relation $A_R$:**
  - Equivalence classes:
    - $\epsilon$: $\{\epsilon\}$
    - $a$: $\{aa, aaa\}$
    - $b$: $\{ab, baa\}$


**Deterministic Automata for** \( L \) **and MN relations for** \( L \)

**DA (with no unreachable states) for** \( L \) **and MN relations for** \( L \) **are in 1-1 correspondence.**

\[
\mathcal{A} \rightarrow R_{\mathcal{A}} \quad \mathcal{A}_R \rightarrow R
\]

**Maps** \( \mathcal{A} \rightarrow R_{\mathcal{A}} \) **and** \( \mathcal{A}_R \leftarrow R \) **are inverses of eachother.**
The relation $\equiv_L$ refines all MN-relations for $L$

**Lemma**

Let $L$ be any language over an alphabet $A$. Let $R$ be any MN-relation for $L$. Then $R$ refines $\equiv_L$. 

Proof: To prove that $xRy$ implies $x \equiv_L y$. Suppose $x \not\equiv_L y$. Then there exists $z$ such that (WLOG) $xz \in L$ and $yz \not\in L$. Suppose $xRy$. Since its an MN relation for $L$, it must be right invariant; and hence $xzRyz$. But this contradicts the assumption that $R$ respects $L$.

As a corollary we have:

**Theorem (Myhill-Nerode)**

$L$ is regular iff $\equiv_L$ is of finite index (that is has a finite number of equivalence classes).
Lemma

Let $L$ be any language over an alphabet $A$. Let $R$ be any MN-relation for $L$. Then $R$ refines $\equiv_L$.

Proof: To prove that $xRy$ implies $x \equiv_L y$. Suppose $x \not\equiv_L y$. Then there exists $z$ such that (WLOG) $xz \in L$ and $yz \not\in L$. Suppose $xRy$. Since its an MN relation for $L$, it must be right invariant; and hence $xzRyz$. But this contradicts the assumption that $R$ respects $L$. As a corollary we have:

Theorem (Myhill-Nerode)

$L$ is regular iff $\equiv_L$ is of finite index (that is has a finite number of equivalence classes).
The relation $\equiv_L$ refines all MN-relations for $L$

Lemma

Let $L$ be any language over an alphabet $A$. Let $R$ be any MN-relation for $L$. Then $R$ refines $\equiv_L$.

Proof: To prove that $xRy$ implies $x \equiv_L y$. Suppose $x \not\equiv_L y$. Then there exists $z$ such that (WLOG) $xz \in L$ and $yz \not\in L$. Suppose $xRy$. Since its an MN relation for $L$, it must be right invariant; and hence $xzRyz$. But this contradicts the assumption that $R$ respects $L$.

As a corollary we have:

Theorem (Myhill-Nerode)

$L$ is regular iff $\equiv_L$ is of finite index (that is has a finite number of equivalence classes).
We call $A_{\equiv L}$ the “canonical” DA for $L$.

In what sense is $A_{\equiv L}$ canonical?

- Every other DA for $L$ is a refinement of $A_{\equiv L}$.
- $A$ is a refinement of $B$ if there is a stable partitioning $\sim$ of $A$ such that quotient of $A$ under $\sim$ (written $A/\sim$) is isomorphic to $B$.
- Stable partitioning of $A = (Q, s, \delta, F)$ is an equivalence relation $\sim$ on $Q$ such that:
  - $p \sim q$ implies $\delta(p, a) \sim \delta(q, a)$.
  - If $p \sim q$ and $p \in F$, then $q \in F$ also.
- Note that if $\sim$ is a stable partitioning of $A$, then $A/\sim$ accepts the same language as $A$. 
Example: 1

A stable partitioning shown by pink and light pink classes, and below, the quotiented automaton:
Example: 2
Proving canonicity of $A_{\equiv L}$

Let $A$ be a DA for $L$ with no unreachable states. Then $A_{\equiv L}$ represents a stable partitioning of $A$. (Use the refinement of $\equiv_L$ by the MN relation $R_A$.)

$A_{\equiv L} \leftarrow \equiv_L$

$A \mapsto R_A$

$\epsilon$ 

$a$

$aaa$ 

$aa$

$\epsilon$ 

$a$

$aaa$ 

$aa$
Stable partitioning \( \approx \)

- Let \( \mathcal{A} = (Q, s, \delta, F) \) be a DA for \( L \) with no unreachable states.
- The canonical MN relation for \( L \) (i.e. \( \equiv_L \)) induces a “coarsest” stable partitioning \( \approx_L \) of \( \mathcal{A} \) given by
  \[
  p \approx_L q \quad \text{iff} \quad \exists x, y \in A^* \text{ such that } \hat{\delta}(s, x) = p \text{ and } \hat{\delta}(s, y) = q, \text{ with } x \equiv_L y.
  \]
- Define a stable partitioning \( \approx \) of \( \mathcal{A} \) by
  \[
  p \approx q \quad \text{iff} \quad \forall z \in A^* : \hat{\delta}(p, z) \in F \text{ iff } \hat{\delta}(q, z) \in F.
  \]
Example of $\approx$ partitioning relation
Stable partitioning $\approx$ is coarsest

Claim: $\approx$ coincides with $\approx_L$.

$\approx_L = \approx$.

Proof:

$p \not\approx q$ iff $\exists x, y, z : \hat{\delta}(s, x) = p, \hat{\delta}(s, y) = q$, and

$\hat{\delta}(p, z) \in F$ but $\hat{\delta}(q, z) \not\in F$.

iff $p \not\approx_L q$. 
Algorithm to compute \( \approx \) for a given DFA

Input: DFA \( \mathcal{A} = (Q, s, \delta, F) \).
Output: \( \approx \) for \( \mathcal{A} \).

1. Initialize entry for each pair in table to “unmarked”.
2. Mark \((p, q)\) if \(p \in F\) and \(q \notin F\) or vice-versa.
3. Scan table entries and repeat till no more marks can be added:
   1. If there exists unmarked \((p, q)\) with \(a \in A\) such that \(\delta(p, a)\) and \(\delta(q, a)\) are marked, then mark \((p, q)\).
4. Return \(\approx\) as: \(p \approx q\) iff \((p, q)\) is left unmarked in table.
Example

Run minimization algorithm on DFA below:

![DFA Diagram]

<table>
<thead>
<tr>
<th></th>
<th>u</th>
<th>p</th>
<th>t</th>
<th>q</th>
<th>s</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example

Run minimization algorithm on DFA below:

```
<table>
<thead>
<tr>
<th></th>
<th>u</th>
<th>p</th>
<th>t</th>
<th>q</th>
<th>s</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>√</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>√</td>
<td>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q</td>
<td>.</td>
<td>√</td>
<td>√</td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>.</td>
<td>√</td>
<td>√</td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>√</td>
<td>.</td>
<td>√</td>
<td>√</td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>
```
Example

Run minimization algorithm on DFA below:

![DFA Diagram]

<table>
<thead>
<tr>
<th></th>
<th>u</th>
<th>p</th>
<th>t</th>
<th>q</th>
<th>s</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>.</td>
<td></td>
<td></td>
<td>.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>✓</td>
<td>.</td>
<td></td>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>t</td>
<td>✓</td>
<td>.</td>
<td></td>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>q</td>
<td>.</td>
<td>✓</td>
<td>✓</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>s</td>
<td>✓</td>
<td>.</td>
<td>✓</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>r</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>.</td>
</tr>
</tbody>
</table>
Run minimization algorithm on DFA below:

```
        p
   a        a
   b        b
   t        t
   s        s
   u        u
   r        r

<table>
<thead>
<tr>
<th></th>
<th>u</th>
<th>p</th>
<th>t</th>
<th>q</th>
<th>s</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>q</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
```
Correctness of minimization algorithm

Claim: Algo always terminates.

- \( n(n - 1)/2 \) table entries in each scan, and at most \( n(n - 1)/2 \) scans.
- In fact, number of scans in algo is \( \leq n \), where \( n = |Q| \).

1. Consider modified step 3.1 in which mark check is done wrt the table at the end of previous scan.
2. Argue that at end of \( i \)-th scan algo computes \( \approx_i \), where
   \[
   p \approx_i q \text{ iff } \forall w \in A^* \text{ with } |w| \leq i : \hat{\delta}(p, w) \in F \text{ iff } \hat{\delta}(q, w) \in F.
   \]
3. Observe that \( \approx_{i+1} \) strictly refines \( \approx_i \), unless the algo terminates after scan \( i + 1 \). So modified algo does at most \( n \) scans.
4. Both versions mark the same set of pairs. Also if modified algo marks a pair, original algo has already marked it.
Correctness of minimization algorithm

Claim: Algo marks \((p, q)\) iff \(p \not\sim q\).

- \((\Rightarrow)\)
- \((\Leftarrow)\)