CSE 135: Introduction to Theory of Computation
Optimal DFA

Sungjin Im

University of California, Merced

02-19-2015
Optimal Algorithms for Regular Languages

Myhill-Nerode Theorem
There is a “unique” “optimal” “algorithm” for every problem that can be solved using finite memory.

Anil Nerode
Optimal Algorithms for Regular Languages

Myhill-Nerode Theorem

There is a “unique” “optimal” “algorithm” for every problem that can be solved using finite memory.

- “algorithm” here means a deterministic machine
Optimal Algorithms for Regular Languages

Myhill-Nerode Theorem
There is a “unique” “optimal” “algorithm” for every problem that can be solved using finite memory.
- “algorithm” here means a deterministic machine
- “optimal” means requires least memory, i.e., has fewest states
Optimal Algorithms for Regular Languages

Myhill-Nerode Theorem

There is a “unique” “optimal” “algorithm” for every problem that can be solved using finite memory.

- “algorithm” here means a deterministic machine
- “optimal” means requires least memory, i.e., has fewest states
- “unique” means that any two DFAs with fewest states for a language are “isomorphic”
Roadmap

- DFA minimization: Minimize a given DFA $M$ by merging "indistinguishable" states.
Roadmap

- DFA minimization: Minimize a given DFA $M$ by merging “indistinguishable” states. In general, could be only minimal (locally optimal)
- In DFA minimization, a minimal (locally optimal) DFA is a minimum (globally optimal) DFA.
- Test if two given DFAs are equivalent.
- Revisit Myhill-Nerode Theorem.
Many DFAs for the same language
Minimization

Problem
Ultimate goal:

Applications
Algorithms using DFAs run in time directly related to the number of states of DFA. Implementation of the DFA itself takes memory proportional to log number of states. So constructing small DFAs is very critical.
Minimization

Problem
Ultimate goal: Given a DFA $M$, construct the DFA with fewest states $M'$ such that $L(M') = L(M)$. 

Intermediate goal: Minimize a given DFA $M$ by merging "indistinguishable" states.

Applications
Algorithms using DFAs run in time directly related to the number of states of DFA. Implementation of the DFA itself takes memory proportional to log number of states. So constructing small DFAs is very critical.
Minimization

Problem
Ultimate goal: Given a DFA $M$, construct the DFA with fewest states $M'$ such that $L(M') = L(M)$. 

Applications
Algorithms using DFAs run in time directly related to the number of states of DFA. Implementation of the DFA itself takes memory proportional to log number of states. So constructing small DFAs is very critical.
Minimization

Problem
Ultimate goal: Given a DFA $M$, construct the DFA with fewest states $M'$ such that $L(M') = L(M)$.
Intermediate goal: Minimize a given DFA $M$ by merging “indistinguishable” states.
Minimization

Problem
Ultimate goal: Given a DFA $M$, construct the DFA with fewest states $M'$ such that $L(M') = L(M)$.
Intermediate goal: Minimize a given DFA $M$ by merging “indistinguishable” states.

Applications
Algorithms using DFAs run in time directly related to the number of states of DFA. Implementation of the DFA itself takes memory proportional to log number of states. So constructing small DFAs is very critical.
Algorithm

Step 1: Remove all unreachable states.
Algorithm

Step 1: Remove all unreachable states.
* Use DFS or BFS.
Algorithm

Step 1: Remove all unreachable states.
* Use DFS or BFS.
Step 2: Merge “similar” states.
Some possible approaches

Want to merge “similar” states.
Some possible approaches

Want to merge “similar” states.
Attempt 1: Focus on what each state remembers/encodes.
Some possible approaches

Want to merge “similar” states.
Attempt 1: Focus on what each state remembers/encodes.
Seems hard when the DFA is large.
Some possible approaches

Want to merge “similar” states.
Attempt 1: Focus on what each state remembers/encodes. Seems hard when the DFA is large.
Attempt 2: State Characterization? Two states are indistinguishable if $\hat{\delta}(p, w) = \hat{\delta}(q, w)$ for all strings $w$. 
Some possible approaches

Want to merge “similar” states.
Attempt 1: Focus on what each state remembers/encodes. Seems hard when the DFA is large.
Attempt 2: State Characterization? Two states are indistinguishable if $\hat{\delta}(p, w) = \hat{\delta}(q, w)$ for all strings $w$. Seems not strong enough.
Distinguishability

When must two states \( p \) and \( q \) of \( M \) not be collapsed?
Distinguishability

When must two states $p$ and $q$ of $M$ not be collapsed?

$$\exists w. \hat{\delta}(p, w) \in F \text{ and } \hat{\delta}(q, w) \notin F; \text{ or}$$

$$\exists w. \hat{\delta}(p, w) \notin F \text{ and } \hat{\delta}(q, w) \in F$$
Distinguishability

When must two states $p$ and $q$ of $M$ not be collapsed?

$$\exists w. \hat{\delta}(p, w) \in F \text{ and } \hat{\delta}(q, w) \notin F; \text{ or}$$
$$\exists w. \hat{\delta}(p, w) \notin F \text{ and } \hat{\delta}(q, w) \in F$$

We will say that $p$ and $q$ are distinguishable when this happens.
Indistinguishability

We say that two states $p$ and $q$ of $M$ are indistinguishable/equivalent if

$$\forall w. \hat{\delta}(p, w) \in F \text{ iff } \hat{\delta}(q, w) \in F$$
Indistinguishability

We say that two states $p$ and $q$ of $M$ are indistinguishable/equivalent if

$$\forall w. \delta(p, w) \in F \text{ iff } \delta(q, w) \in F$$
Indistinguishability

We say that two states \( p \) and \( q \) of \( M \) are indistinguishable/equivalent if

\[
\forall w. \hat{\delta}(p, w) \in F \text{ iff } \hat{\delta}(q, w) \in F
\]
Indistinguishability

We say that two states $p$ and $q$ of $M$ are indistinguishable/equivalent if

$$\forall w. \hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F$$

Indistinguishability defines an equivalence class: $A \equiv A$ (reflexivity), $A \equiv B \iff B \equiv A$ (symmetricity), $A \equiv B$ and $B \equiv C \Rightarrow$ (transivity). So let’s use $p \equiv q$ to say that two states $p$ and $q$ are indistinguishable.
Gradually Refine Indistinguishability

Recall

\[ p \equiv q : \quad \forall w. \ \hat{\delta}(p, w) \in F \ \text{iff} \ \hat{\delta}(q, w) \in F \]

For each \( k \geq 0 \), define

\[ p \equiv_k q : \quad \forall w \text{ with } |w| \leq k. \ \hat{\delta}(p, w) \in F \ \text{iff} \ \hat{\delta}(q, w) \in F \]

An equivalence class partitions states into disjoint groups. \( \equiv_0 \) has two groups: \( Q \setminus F \) and \( F \).
Suppose that we know for each pair of two states $p, q$ if $p \equiv_k q$ or not. How can we examine if $p \equiv_{k+1} q$ or not?

If $p \not\equiv_k q$, then we have $p \not\equiv_{k+1} q$. Each group of states can be only refined!

If $p \equiv_k q$, then we need to do more work:
Gradually Refine Indistinguishability

Suppose that we know for each pair of two states $p, q$ if $p \equiv_k q$ or not. How can we examine if $p \equiv_{k+1} q$ or not?

If $p \not\equiv_k q$, then we have $p \not\equiv_{k+1} q$. Each group of states can be only refined!

If $p \equiv_k q$, then we need to do more work:

$$p \equiv_{k+1} q \iff \forall a \in \Sigma. \delta(p, a) \equiv_k \delta(q, a)$$
Gradually Refine Indistinguishability

Suppose that we know for each pair of two states \( p, q \) if \( p \equiv_k q \) or not. How can we examine if \( p \equiv_{k+1} q \) or not?

If \( p \not\equiv_k q \), then we have \( p \not\equiv_{k+1} q \). Each group of states can be only refined!

If \( p \equiv_k q \), then we need to do more work:

\[
p \equiv_{k+1} q \iff \forall a \in \Sigma. \delta(p, a) \equiv_k \delta(q, a)
\]

Do you see why?
Gradually Refine Indistinguishability

Suppose that we know for each pair of two states \( p, q \) if \( p \equiv_k q \) or not. How can we examine if \( p \equiv_{k+1} q \) or not?

If \( p \not\equiv_k q \), then we have \( p \not\equiv_{k+1} q \). Each group of states can be only refined!

If \( p \equiv_k q \), then we need to do more work:

\[
p \equiv_{k+1} q \iff \forall a \in \Sigma. \hat{\delta}(p, a) \equiv_k \delta(q, a)
\]

Do you see why?

\[
p \equiv_{k+1} q
\]
\[
\iff \forall u \text{ with } |u| \leq k \forall a \in \Sigma. \hat{\delta}(p, au) \in F \iff \hat{\delta}(q, au) \in F
\]
\[
\iff \forall u \text{ with } |u| \leq k \forall a \in \Sigma. \hat{\delta}(\delta(p, a), u) \in F \iff \hat{\delta}(\delta(q, a), u) \in F
\]
\[
\iff \forall a \in \Sigma \forall u \text{ with } |u| \leq k. \hat{\delta}(\delta(p, a), u) \in F \iff \hat{\delta}(\delta(q, a), u) \in F
\]
\[
\iff \forall a \in \Sigma. \delta(p, a) \equiv_k \delta(q, a)
\]
Distinguishability

Inductive Definition

Distinguishability can be inductively defined as follows
Distinguishability
Inductive Definition

Distinguishability can be inductively defined as follows

- If $p \in F$ and $q \not\in F$ then $p$ and $q$ are distinguishable
Distinguishability can be inductively defined as follows

- If $p \in F$ and $q \notin F$ then $p$ and $q$ are distinguishable.
- If for some $a$, $\delta(p, a) = p'$ and $\delta(q, a) = q'$, and $p'$ and $q'$ are distinguishable, then $p$ and $q$ are distinguishable.
Distinguishability
An Algorithm

Let distinct be a table with an entry for each pair of states. Initially all entries are 0.

\[
\text{if } p \in F \text{ and } q \notin F \text{ (or vice versa)} \\
\text{then } \text{distinct}(p, q) := 1
\]

repeat
for each pair \((p, q)\) and symbol \(a\)
    if distinct(\(\delta(p, a)\), \(\delta(q, a)\)) = 1,
        then distinct\((p, q)\) := 1
until no changes in table
Minimization Algorithm

1. Remove states that are not reachable from the initial state
Minimization Algorithm

1. Remove states that are not reachable from the initial state
2. Find all pairs of states that are distinguishable
Minimization Algorithm

1. Remove states that are not reachable from the initial state
2. Find all pairs of states that are distinguishable
3. Collapse pairs that are not distinguishable
Example
Example
Example

\[
\begin{array}{cccccc}
B & & & & & \\
C & \ast & \ast & & & \\
E & & \ast & & & \\
F & & & \ast & & \\
g & & & & \ast & \\
H & & & & & \\
A & B & C & E & F & G \\
\end{array}
\]

\[
\begin{array}{cccccc}
0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
\]
Example

\[
\begin{array}{cccccc}
B & * \\
C & * & * \\
E & * & * & * \\
F & * & * & * & * \\
G & * & * & * & * & * \\
H & * & * & * & * & * & * \\
\end{array}
\]
Example

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>C</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

\[ \begin{array}{cccccc}
B & C & E & F & G
\end{array} \]

![Diagram](image)
Example

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

![Graph Diagram]
Example

\[\equiv_0: \quad \{A, B, E, F, G, H\}, \{C\}\]

\[\equiv_1: \quad \{A, E, G\}, \{B, H\}, \{F\}, \{C\}\]

\[\equiv_2: \quad \{A, E\}, \{G\}, \{B, H\}, \{F\}, \{C\}\]

\[\equiv_3: \quad \{A, E\}, \{G\}, \{B, H\}, \{F\}, \{C\}\]

No change from \(\equiv_2\) to \(\equiv_3\), so stop.
Example (2)

Remove unreachable states.

DFA $M$
Example (2)

Remove unreachable states.

DFA $M$
Example (2)

\[\equiv_0:\]

\[
\begin{align*}
\{A, B, C\}, \{D, E\}.
\end{align*}
\]

\[
\begin{align*}
\{A\}, \{B, C\}, \{D, E\}.
\end{align*}
\]

\[
\begin{align*}
\{A\}, \{B, C\}, \{D, E\}.
\end{align*}
\]
Example (2)

\[ \equiv_0: \{ A, B, C \}, \{ D, E \}. \]
Example (2)

\[\equiv_0: \{A, B, C\}, \{D, E\}.\]

\[\equiv_1: \]
Example (2)

\[ \equiv_0: \{A, B, C\}, \{D, E\} \]

\[ \equiv_1: \{A\}, \{B, C\}, \{D, E\} \]
Example (2)

- \( \equiv_0 : \{A, B, C\}, \{D, E\} \).
- \( \equiv_1 : \{A\}, \{B, C\}, \{D, E\} \).
- \( \equiv_2 : \)
Example (2)

- $\equiv_0$: \(\{A, B, C\}, \{D, E\}\).
- $\equiv_1$: \(\{A\}, \{B, C\}, \{D, E\}\).
- $\equiv_2$: \(\{A\}, \{B, C\}, \{D, E\}\).
Optimal Algorithms for Regular Languages

- $\equiv_0$: \{A, B, C\}, \{D, E\}.
- $\equiv_1$: \{A\}, \{B, C\}, \{D, E\}.
- $\equiv_2$: \{A\}, \{B, C\}, \{D, E\}.
Roadmap

- DFA minimization: Minimize a given DFA $M$ by merging “indistinguishable” states (possibly locally optimal) – done.
- In DFA minimization, a minimal (locally optimal) DFA is a minimum (globally optimal) DFA.
- Test if the two given DFAs are equivalent.
- Revisit Myhill-Nerode Theorem.
We would like to test if two DFAs $M = (Q^M, \Sigma^M, \delta^M, q_0^M, F^M)$ and $N = (Q^N, \Sigma^N, \delta^N, q_0^N, F^N)$ accept the same language or not.

1. Run the table-filling algorithm on both DFAs simultaneously.
2. $M$ and $N$ accept the same language iff $q_0^M \equiv q_0^N$. 

Decide if two given DFAs accept the same language

We would like to test if two DFAs $M = (Q^M, \Sigma^M, \delta^M, q_0^M, F^M)$ and $N = (Q^N, \Sigma^N, \delta^N, q_0^N, F^N)$ accept the same language or not.

1. Run the table-filling algorithm on both DFAs simultaneously.
Decide if two given DFAs accept the same language

We would like to test if two DFAs $M = (Q^M, \Sigma^M, \delta^M, q_0^M, F^M)$ and $N = (Q^N, \Sigma^N, \delta^N, q_0^N, F^N)$ accept the same language or not.

1. Run the table-filling algorithm on both DFAs simultaneously.
2. $M$ and $N$ accept the same language iff $q_0^M \equiv q_0^N$. 
Table-filling Algorithm Gives a Globally Optimal DFA

Very short proof sketch: $M = (Q^M, \Sigma^M, \delta^M, q_0^M, F^M)$: DFA output by the algorithm
$N = (Q^N, \Sigma^N, \delta^N, q_0^N, F^N)$: a globally optimal DFA
For the sake of contradiction suppose $|Q^N| < |Q^M|$. Then one can find $q_i^M \neq q_j^M$ such that $q_i^M \equiv q_t^N \equiv q_j^M$ for some $q_t^N \in Q^N$. 
Isomorphism

Definition
Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be two DFAs. A function $f : Q_1 \rightarrow Q_2$ is said to be isomorphism iff

- $f$ is bijective, i.e., one-to-one and onto
- $f(q_1) = q_2$
- For every $p \in Q_1$ and $a \in \Sigma$, $f(\delta_1(p, a)) = \delta_2(f(p), a)$
- $q \in F_1$ iff $f(q) \in F_2$

Thus, if $M_1$ and $M_2$ are isomorphic then they are the “same” machine except for possibly renaming states.
Isomorphism

Definition
Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be two DFAs. A function $f : Q_1 \rightarrow Q_2$ is said to be isomorphism iff

- $f$ is bijective, i.e., one-to-one and onto
- $f(q_1) = q_2$
- For every $p \in Q_1$ and $a \in \Sigma$, $f(\delta_1(p, a)) = \delta_2(f(p), a)$
- $q \in F_1$ iff $f(q) \in F_2$

$M_1$ and $M_2$ are said to be isomorphic if there is an isomorphism $f$ from $M_1$ to $M_2$. 
Isomorphism

Definition
Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be two DFAs. A function $f : Q_1 \rightarrow Q_2$ is said to be isomorphism iff

- $f$ is bijective, i.e., one-to-one and onto
- $f(q_1) = q_2$
- For every $p \in Q_1$ and $a \in \Sigma$, $f(\delta_1(p, a)) = \delta_2(f(p), a)$
- $q \in F_1$ iff $f(q) \in F_2$

$M_1$ and $M_2$ are said to be isomorphic if there is an isomorphism $f$ from $M_1$ to $M_2$.
Thus, if $M_1$ and $M_2$ are isomorphic then they are the “same” machine except for possibly renaming states.
Theorem

For any regular language \( L \), there is a unique (upto isomorphism) DFA with fewest states that recognizes \( L \).