CS 172: Computability and Complexity

Minimization of DFAs

Sanjit A. Seshia
EECS, UC Berkeley

Acknowledgments: L. von Ahn, L. Blum, M. Blum, A. Sinclair
What is Minimization?

Minimized DFA for language L

= DFA with fewest states that recognizes L

Also called minimal DFA
Why is Minimization Important?

DFAs are how computers manipulate regular languages (expressions)

DFA size determines space/time efficiency
IS THIS MINIMAL?

NO
HOW ABOUT THIS?

YES

1

0

1

0
Equivalent DFAs
Main Result of this Lecture

For every regular language $L$, there exists a unique, minimal DFA that recognizes $L$

- uniqueness up to re-labeling of states
Words $\leftrightarrow$ States

- Let DFA $M = (Q, \Sigma, \delta, q_0, F)$
- Each word $w$ in $\Sigma^*$ corresponds to a unique state in $Q$
  - The ending state of $M$ on $w$
- Given $x, y \in \Sigma^*$
  - $x \sim_M y$ iff $M$ ends in the same state on both $x$ and $y$
  - $\sim_M$ is an equivalence relation (why?)
  - How many equivalence classes are there?
Example:
Is $1 \sim_M 11$? $10 \sim_M 00$?
Indistinguishable Words/Strings

• Let DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognize $L$
• Given $x, y \in \Sigma^*$
  $x \sim_L y$ (x and y are indistinguishable) iff
  $\forall z \in \Sigma^*, xz \in L$ iff $yz \in L$

Compare with

$- x \sim_M y$ iff
  $M$ ends in the same state on both $x$ and $y$
Example:
What are indistinguishable words?
\( \sim_L \) and \( \sim_M \)

- Let DFA \( M = (Q, \Sigma, \delta, q_0, F) \) recognize \( L \)
- Given \( x, y \in \Sigma^* \)
  - \( x \sim_L y \) (\( x \) and \( y \) are indistinguishable) iff 
    \[ \forall z \in \Sigma^*, \ xz \in L \iff yz \in L \]
  - \( x \sim_M y \) iff 
    \( M \) ends in the same state on both \( x \) and \( y \)

- **True or False:**
  - If \( x \sim_M y \) then \( x \sim_L y \) **TRUE**
  - If \( x \sim_L y \) then \( x \sim_M y \) **FALSE**
Indistinguishable Words

• Let DFA $M = (Q, \Sigma, \delta, q_0, F)$ recognize $L$
• Given $x, y \in \Sigma^*$
  $- x \sim_L y$ (x and y are indistinguishable) iff
  $\forall z \in \Sigma^*, xz \in L$ iff $yz \in L$
  $- x \sim_M y$ iff
  $M$ ends in the same state on both $x$ and $y$

Which has more equivalence classes --
$\sim_M$ or $\sim_L$?
Myhill-Nerode Theorem (a version)

The relation $\sim_L$ defines a DFA $M_L$ for language $L$ where the states of $M_L$ correspond to the equivalence classes of $\sim_L$

$M_L$ is the unique, minimal DFA for $L$ (up to isomorphism)
Proof of Myhill-Nerode Thm.
Next:
Algorithm for DFA Minimization
Indistinguishable States

- Idea: Merge “indistinguishable states”
- Recall:
  - States of DFA $M$ map 1-1 to equivalence classes of $\sim_M$
  - Each equivalence class of $\sim_M$ is in some equivalence class of $\sim_L$
- States $p$ and $q$ are indistinguishable iff their corresponding $\sim_M$ equivalence classes are in the same class of $\sim_L$
- We write $p \sim q$
- $p \not\sim q \rightarrow “p$ and $q$ are distinguishable”
The Algorithm We Want

Input: DFA $M$

Output: DFA $M_L$ such that:

\[ M \equiv M_L \]

$M_L$ has no unreachable states

$M_L$ is irreducible

| states of $M_L$ are pairwise distinguishable

Theorem: $M_L$ is the unique minimum
DFA Minimization Algo.: Idea

- States of $M_L$ are equivalence classes of $\sim_L$
- Equivalence classes of $\sim_L$ can be obtained by merging states of $M$
- Our algorithm works in reverse:
  - Start by assuming all states as being merged together
  - Identify pairs of distinguishable states
    - Repeat until no new distinguishable state-pairs identified
TABLE-FILLING ALGORITHM

Input: DFA $M = (Q, \Sigma, \delta, q_0, F)$
Output: Table: $\{ (p,q) \mid p,q \in Q \text{ and } p \not\sim q \}$

States of $M_L = \{ [q] \mid q \in Q \}$

Base Case: $p$ accepts and $q$ rejects $\Rightarrow p \not\sim q$

Recursion:

$p \xrightarrow{\sigma} p'$
$q \xrightarrow{\sigma} q'$

$\Rightarrow p \not\sim q$
Example
Do Try this at Home!
Correctness of Algorithm

1. If algorithm marks \((p, q)\) as “d”, then \(p \not\sim q\)

2. If \(p \not\sim q\), then algorithm marks \((p, q)\) as “d”

Proving (1) is easy. Use induction on the step at which \((p, q)\) was marked “d”.
Part (2):
If $p \not\sim q$, then the algorithm marks $(p, q)$ as “d”

Proof (by contradiction):
Suppose $p \not\sim q$, but the algorithm does not mark $(p, q)$ as “d”

Since $p \not\sim q$ there exists $w$ such that:

$$\hat{\delta}(p, w) \in F \quad \text{and} \quad \hat{\delta}(q, w) \not\in F$$

Of all such “bad pairs” $(p, q)$, let $p, q$ be a pair with the smallest $|w|$
If $p \not\sim q$, then the algorithm marks $(p, q)$ as “d”

Proof (by contradiction):
Suppose $p \not\sim q$, but the algorithm does not mark $(p, q)$ as “d”

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

Of all such “bad pairs” $(p, q)$, let $p, q$ be a pair with the smallest $|w|$

$w = \sigma w'$, where $\sigma \in \Sigma$ (w is not $\varepsilon$, why?)

Let $p' = \delta(p, \sigma)$ and $q' = \delta(q, \sigma)$

Then $(p', q')$ is also a bad pair Contradiction! (why?)
Complexity of Algorithm

• For DFA $M$, let
  – Number of states of $M$ be $n$
  – Size of the input alphabet $\Sigma$ be $k$

• Initialization of table: $O(n^2)$
• Rest of the algorithm: $O(k \ n^2)$
Minimal NFA is NOT Unique
Next Steps

• Read Sipser 2.1 in preparation for next lecture