

CS 373: Theory of Computation

Manoj Prabhakaran

Mahesh Viswanathan

Fall 2008

1 Introduction

Optimal Algorithms



Figure 1: Manuel Blum

Best Solutions

If a problem can be solved computationally, is there always a “best” method for solving it?

Example 1. Merge Sort and Heap Sort achieve the asymptotically best time possible for sorting (in a certain model).

Blum’s Speedup Theorem

No! There are computationally solvable problems that have no optimal algorithms! In other words, there are problems for which any algorithm can always be made faster.

Optimal Algorithms

Regular Languages



Figure 2: Anil Nerode

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”

1.1 Suffix Languages

Suffix Language of a State

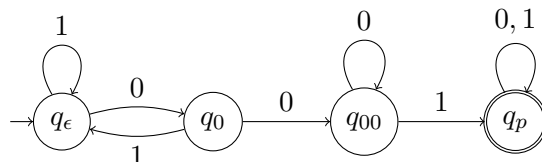


Figure 3: DFA M

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$, $\text{suffix}(M, q) = \{w \in \Sigma^* \mid q \xrightarrow{w}_M q' \text{ and } q' \in F\}$. In other words, it is the collection of all words accepted if q were the initial state.

For example, $\text{suffix}(M, q_{00}) = 0^*1(0 \cup 1)^*$.

Suffix Languages

Definition 2. For a language $L \subseteq \Sigma^*$, and a string $x \in \Sigma^*$, the *suffix language of L* with respect to x , is defined as

$$\text{suffix}(L, x) = \{y \in \Sigma^* \mid xy \in L\}$$

In other words, $\text{suffix}(L, x)$ is the collection of strings y which when prefixed by x , result in a string in L .

The *class of suffix languages of L* is

$$\mathcal{C}_{\text{suf}}(L) = \{\text{suffix}(L, x) \mid x \in \Sigma^*\}$$

1.2 Examples

Example: L_{odd}

Example 3. Consider $L_{\text{odd}} = \{w \in \{0, 1\}^* \mid w \text{ has an odd number of 1s}\}$

- $\text{suffix}(L_{\text{odd}}, \epsilon) = L_{\text{odd}}$
- $\text{suffix}(L_{\text{odd}}, 0) = L_{\text{odd}}$
- $\text{suffix}(L_{\text{odd}}, 1) = \{w \in \{0, 1\}^* \mid w \text{ has an even number of 1s}\} = L_{\text{even}}$

Example: L_{odd}

Class of Suffix Languages

Example 4. Consider $L_{\text{odd}} = \{w \in \{0, 1\}^* \mid w \text{ has an odd number of 1s}\}$

- For x that has an even number of 1s, $\text{suffix}(L_{\text{odd}}, x) = L_{\text{odd}}$
- For x that has an odd number of 1s, $\text{suffix}(L_{\text{odd}}, x) = L_{\text{even}}$
- Thus, $C_{\text{suf}}(L_{\text{odd}}) = \{L_{\text{odd}}, L_{\text{even}}\}$

Example: L_{odd}

DFA and Suffix Languages

Example 5. Recall, $C_{\text{suf}}(L_{\text{odd}}) = \{L_{\text{odd}}, L_{\text{even}}\}$. A DFA for L_{odd} is

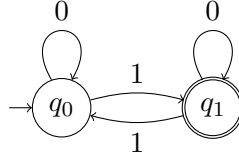


Figure 4: DFA for L_{odd}

Observe that $\text{suffix}(M, q_0) = L_{\text{odd}}$, and $\text{suffix}(M, q_1) = L_{\text{even}}$.

Example: L_{001}

Example 6. Consider $L_{001} = (0 \cup 1)^*001(0 \cup 1)^*$.

- $\text{suffix}(L_{001}, \epsilon) = L_{001}$
- $\text{suffix}(L_{001}, 0) = 00^*1(0 \cup 1)^* \cup 1L_{001}$
- $\text{suffix}(L_{001}, 00) = 0^*1(0 \cup 1)^*$
- $\text{suffix}(L_{001}, 001) = (0 \cup 1)^*$

Example: L_{001}

Class of Suffix Languages

Example 7. Consider $L_{001} = (0 \cup 1)^*001(0 \cup 1)^*$.

- For $x = \epsilon$ or $x \in (\Sigma^* \setminus L_{001}) \cap (0 \cup 1)^*1$, $\text{suffix}(L_{001}, x) = L_{001}$
- For $x \in (\Sigma^* \setminus L_{001}) \cap (0 \cup 1)^*0$, $\text{suffix}(L_{001}, x) = 00^*1(0 \cup 1)^* \cup 1L_{001}$
- For $x \in (\Sigma^* \setminus L_{001}) \cap (0 \cup 1)^*00$, $\text{suffix}(L_{001}, x) = 0^*1(0 \cup 1)^*$
- For $x \in L_{001}$, $\text{suffix}(L_{001}, x) = (0 \cup 1)^*$

Example: L_{001}

DFA and Suffix Languages

Example 8. A DFA for L_{001} is

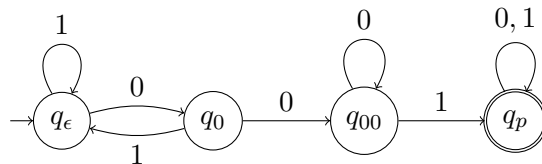


Figure 5: DFA for L_{odd}

Observe that the suffix languages of the states correspond to the class of suffix languages.

Example: L_{0n1n}

Example 9. Consider $L_{0n1n} = \{0^n 1^n \mid n \geq 0\}$.

- $\text{suffix}(L_{0n1n}, 0) = \{0^{n-1} 1^n \mid n \geq 1\}$
- $\text{suffix}(L_{0n1n}, 0^i) = \{0^{n-i} 1^n \mid n \geq i\}$

Proposition 10. $\mathcal{C}_{\text{suf}}(L_{0n1n})$ has infinitely many languages.

Proof. Observe that for $i \neq j$, $\text{suffix}(L_{0n1n}, 0^i) \neq \text{suffix}(L_{0n1n}, 0^j)$. □

Recap ...

Observations

In the previous examples,

- For regular L ,
 - $\mathcal{C}_{\text{suf}}(L)$ has only finitely many languages
 - There is a DFA D for L , the suffix languages of the states of D , correspond to the languages in $\mathcal{C}_{\text{suf}}(L)$
- For non-regular L ,
 - $\mathcal{C}_{\text{suf}}(L)$ has infinitely many languages

Are these observations true in general?

2 Myhill-Nerode Theorem

2.1 Regular Languages have few Suffix Languages

States and Suffix Languages

Proposition 11. *Let $M = (Q, \Sigma, \delta, q_0, F)$ and let $L = L(M)$. Then if $q_0 \xrightarrow{x}_M q$ and $q_0 \xrightarrow{y}_M q$ (i.e., both x and y take M to the same state), then $\text{suffix}(L, x) = \text{suffix}(L, y)$.*

Proof. Consider $z \in \text{suffix}(L, x)$. Then $xz \in L$.

- $q_0 \xrightarrow{x}_M q \xrightarrow{z}_M q_1$ for some $q_1 \in F$
- Therefore, $q_0 \xrightarrow{y}_M q \xrightarrow{z}_M q_1$, and $yz \in L$, which means $z \in \text{suffix}(L, y)$.

Similarly, we can show that if $z \in \text{suffix}(L, y)$ then $z \in \text{suffix}(L, x)$. □

Regularity and Suffix Languages

Corollary 12. *If L is regular then $\mathcal{C}_{\text{suf}}(L)$ is finite.*

Proof. If L is regular then there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ such that $L = L(M)$.

- We have shown that, if x, y reach the same state in M then $\text{suffix}(L, x) = \text{suffix}(L, y)$
- Thus, $|\mathcal{C}_{\text{suf}}(L)| \leq |Q|$, which is finite.

□

2.2 Canonical DFAs for Languages

Canonical DFAs for Regular Languages

Proposition 13. *For a language $L \subseteq \Sigma^*$, if $\mathcal{C}_{\text{suf}}(L)$ is finite then there is a DFA M^L such that $L(M^L) = L$.*

Proof. Define $M^L = (Q^L, \Sigma, \delta^L, q_0^L, F^L)$ as follows.

- $Q^L = \mathcal{C}_{\text{suf}}(L)$
- $q_0^L = \text{suffix}(L, \epsilon)$
- $F^L = \{\text{suffix}(L, x) \mid \epsilon \in \text{suffix}(L, x)\}$
- $\delta^L(\text{suffix}(L, x), a) = \text{suffix}(L, xa)$

Is δ well-defined? Same state can have multiple names (i.e., x, y s.t. $\text{suffix}(L, x) = \text{suffix}(L, y)$). □

Canonical DFAs for Regular Languages

Transition function is well-defined

Proof (contd). • $\delta^L(\text{suffix}(L, x), a) = \text{suffix}(L, xa)$

- δ is well-defined because if $\text{suffix}(L, x) = \text{suffix}(L, y)$ then, for all $a \in \Sigma$, $\text{suffix}(L, xa) = \text{suffix}(L, ya)$.

– Suppose $\text{suffix}(L, x) = \text{suffix}(L, y)$. Then,

$$\begin{aligned} z \in \text{suffix}(L, xa) &\iff xaz \in L \\ &\iff az \in \text{suffix}(L, x) = \text{suffix}(L, y) \\ &\iff yaz \in L \iff z \in \text{suffix}(L, ya) \end{aligned}$$

– Hence, $\text{suffix}(L, xa) = \text{suffix}(L, ya)$. □

Canonical DFAs for Regular Languages

Proof (contd). • For any string $x \in \Sigma^*$, $q_0^L \xrightarrow{x}_{ML} \text{suffix}(L, x)$.

– $q_0^L = \text{suffix}(L, \epsilon) \xrightarrow{x_1}_{ML} \text{suffix}(L, x_1) \xrightarrow{x_2}_{ML} \text{suffix}(L, x_1x_2) \dots \xrightarrow{x_n}_{ML} \text{suffix}(L, x)$

– Formally, by induction on $|x|$

- $x \in L$ iff $\epsilon \in \text{suffix}(L, x)$ iff $\text{suffix}(L, x) \in F^L$

- Hence, $x \in L$ iff M^L accepts x . □

Example of Canonical DFA

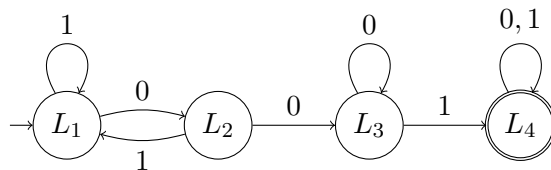
Example 14. Consider $L_{001} = (0 \cup 1)^*001(0 \cup 1)^*$. Recall that the suffix languages are

$$L_1 = L_{001} = \text{suffix}(L_{001}, \epsilon) = \text{suffix}(L_{001}, (0 \cup \epsilon)1)$$

$$L_2 = 00^*1(0 \cup 1)^* \cup 1L_{001} = \text{suffix}(L_{001}, 0)$$

$$L_3 = 0^*1(0 \cup 1)^* = \text{suffix}(L_{001}, 00) = \text{suffix}(L_{001}, 000)$$

$$L_4 = (0 \cup 1)^* = \text{suffix}(L_{001}, 001) = \text{suffix}(L_{001}, 001(0 \cup 1))$$



2.3 Minimality and Uniqueness of Canonical DFA

Canonical DFA is the smallest DFA

Proposition 15. For any regular language L , M^L is the unique smallest DFA that recognizes L upto isomorphism.

Isomorphism

Definition 16. 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.

Canonical DFA is the smallest DFA

Proposition 17. For any regular language L , M^L is the unique (upto isomorphism) smallest DFA that recognizes L .

Proof. Recall $M^L = (Q^L, \Sigma, \delta^L, q_0^L, F^L)$. Let $M = (Q, \Sigma, \delta, q_0, F)$ be some DFA such that $L(M) = L$.

- Define $f : Q \rightarrow Q^L$ as follows: $f(q) = \text{suffix}(L, x)$ iff $q_0 \xrightarrow{x}_M q$.
 - f is well-defined, because we showed that if x, y both take M to q , then $\text{suffix}(L, x) = \text{suffix}(L, y)$.
- f is onto because a state $\text{suffix}(L, x) \in Q^L$ is the image of q' under f , where $q_0 \xrightarrow{x}_M q'$.
- Thus, $|Q| \geq |Q^L|$ □

Canonical DFA is the smallest DFA

f preserves transitions

Proof (contd). Suppose $|Q| = |Q^L|$. Then we need to show that M and M^L are isomorphic. Recall, $f(q) = \text{suffix}(L, x)$ iff $q_0 \xrightarrow{x}_M q$.

- Since f is onto, f must be one-to-one. Thus, f is bijective.
- Since $q_0 \xrightarrow{\epsilon}_M q_0$, $f(q_0) = \text{suffix}(L, \epsilon) = q_0^L$
- Suppose $q \xrightarrow{a}_M q'$ and $f(q) = \text{suffix}(L, x)$. Then, $q_0 \xrightarrow{x}_M q$, and so $q_0 \xrightarrow{xa}_M q'$. Thus, $f(q') = \text{suffix}(L, xa)$. So, $\delta^L(\text{suffix}(L, x), a) = \text{suffix}(L, xa)$.
- Suppose $q \in F$, and $f(q) = \text{suffix}(L, x)$. Hence, $x \xrightarrow{q}_M$, and so $x \in L$, and $\epsilon \in \text{suffix}(L, x)$. Thus, $f(q) \in F^L$. The converse holds by reversing this argument, and so f is an isomorphism. \square

Myhill-Nerode Theorem

In summary ...

Theorem 18. *The following facts are true.*

1. *If L is regular then $\mathcal{C}_{\text{suf}}(L)$ is finite.*
2. *If $\mathcal{C}_{\text{suf}}(L)$ is finite then L is regular; more precisely, there is a DFA M^L , whose states are the languages in $\mathcal{C}_{\text{suf}}(L)$, such that $L(M^L) = L$.*
3. *For any regular language L , M^L is the unique (upto isomorphism) DFA with fewest states that recognizes L .*

3 DFA Minimization

3.1 Distinguishability

Minimization

Problem

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.

Ideas

Let $M = (Q, \Sigma, \delta, q_0, F)$, and $L = L(M)$. Recall that from the proof of Myhill-Nerode Theorem

- M^L is the unique smallest DFA
- The function $f : Q \rightarrow Q^L$ maps a state q to the state of Q^L that corresponds to the language $\text{suffix}(M, q)$.

Thus M can be “minimized” by collapsing states q_1 and q_2 if $\text{suffix}(M, q_1) = \text{suffix}(M, q_2)$.

Distinguishability

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

$$\begin{aligned} \text{suffix}(M, p) \neq \text{suffix}(M, q) &\text{ iff} \\ \exists w. w \in \text{suffix}(M, p) \text{ and } w \notin \text{suffix}(M, q) &\text{ (or vice versa) iff} \\ \exists w. p \xrightarrow{w}_M p' \text{ and } p' \in F \text{ and } q \xrightarrow{w}_M q' &\text{ and } q' \notin F \end{aligned}$$

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

Distinguishability

Inductive Definition

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
-

3.2 The Algorithm

Distinguishability

An Algorithm

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

```
if  $p \in F$  and  $q \notin F$  (or vice versa)
then 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
 2. Find all pairs of states that are distinguishable
 3. Collapse pairs that are not distinguishable
-