Limitations of Finite Automata

Lecture 6
Today

Proving that certain languages need DFAs with a large number of states

Proving that certain ("easy") languages cannot be decided using DFAs at all!

Using Closure Properties of regular languages to reason about non-regularity
Need for Memory

e.g., Language $L = \{ 0^{33} \}$ (just one string, of 33 0’s)

“Clearly” a DFA for $L$ will need to keep count of the 0’s seen, up to 33

35 states (count = 0, 1, … , 33 and “crashed”)

How do we rule out the possibility of a clever DFA with fewer states?
Need for Memory

Suppose $M$ with $d < 35$ states s.t. $L(M) = L = \{0^{33}\}$

Consider $\delta^*(s,w)$ for $w \in \{0^i \mid i \in [0,34]\}$

Pigeonhole Principle $\Rightarrow$ at least two values $i < j$ s.t.

$$\delta^*(s,0^i) = \delta^*(s,0^j) = q \text{ (say)}$$

Let $u = 0^i$ and $v = 0^j$

Let $x = 0^k$ where $k = 33 - i$ ($k \geq 0$ since $i < j \leq 34$).

$ux \in L \Rightarrow \delta^*(s,ux) \in F$ and $vx \notin L \Rightarrow \delta^*(s,vx) \notin F$

But $\delta^*(s,ux) = \delta^*(q,x) = \delta^*(s,vx)$ !
Abstracting the proof

Consider $\delta^*(s,w)$ for $w \in \{ 0^i \mid i \in [0,34] \}$

By the pigeonhole principle, at least two values $i < j$ s.t.
$\delta^*(s,0^i) = \delta^*(s,0^j) = q$ (say)

Let $u = 0^i$ and $v = 0^j$

Let $x = 0^k$ where $k = 33 - i$
($k \geq 0$ since $i < j \leq 34$).

$ux \in L \Rightarrow \delta^*(s,ux) \in F$ and $vx \notin L \Rightarrow \delta^*(s,vx) \notin F$

But $\delta^*(s,ux) = \delta^*(q,x) = \delta^*(s,vx)$

Come up with a set $S$, $|S| = d$
s.t. for any two distinct $u, v \in S$

$\exists x s.t$
$| \{ ux, vx \} \cap L | = 1$

Proves that any DFA for $L$
must have at least $d$ states
Abstracting the proof as a Game!

**SuvxGame**(\(L,d\))

<table>
<thead>
<tr>
<th><strong>Defender</strong> (claim: (\exists) DFA for (L) with (&lt; d) states)</th>
<th><strong>Challenger</strong> (claim: no DFA for (L) has (&lt; d) states)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(S) picks a set (S \subseteq \Sigma^*), (</td>
<td>S</td>
</tr>
<tr>
<td>(u,v) picks distinct (u,v \in S)</td>
<td>(x) picks (x \in \Sigma^*)</td>
</tr>
<tr>
<td>Challenger wins if (</td>
<td>{ux,vx} \cap L</td>
</tr>
</tbody>
</table>

**Theorem** : If there is a DFA for \(L\) with \(< d\) states, then Defender has a winning strategy in \(SuvxGame(L,d)\)

\[\forall S \subseteq \Sigma^*, |S| \geq d, \exists u,v \in S, u \neq v, \forall x \in \Sigma^*, ux \in L \iff vx \in L\]
Strings Indistinguishable to a DFA

Let $M = (\Sigma, Q, \delta, s, F)$ be an arbitrary DFA

Suppose $\delta^*(s, u) = \delta^*(s, v) = q$

Then, for every $x$, $\delta^*(s, ux) = \delta^*(s, vx) = \delta^*(q, x)$

i.e., $M$ “can’t tell the difference” between having seen $u$ and $v$

In particular, for every $x$, $ux \in L(M) \iff vx \in L(M)$  \hspace{1cm} (\ast)

**Theorem**: If there is a DFA for $L$ with $< d$ states, then Defender has a winning strategy in $S_{uvxGame}(L, d)$

**Proof**: Winning strategy: Let $L = L(M)$, $M = (\Sigma, Q, \delta, s, F)$, $|Q| < d$.

Receive $S$. Since $|S| > |Q|$, by pigeonhole principle, $\exists u, v \in S$, s.t., $\delta^*(s, u) = \delta^*(s, v)$. Send $(u, v)$. By (\ast), Challenger can’t win the game.
Proving Lowerbound on DFA size

**Theorem**: If there is a DFA for $L$ with $< d$ states, then Defender has a winning strategy in SuvxGame($L, d$)

$$\forall S \subseteq \Sigma^*, |S| \geq d, \exists u, v \in S, u \neq v, \forall x \in \Sigma^*, ux \in L \iff vx \in L$$

**Corollary**: If the Challenger has a winning strategy in the SuvxGame($L, d$) (so that the Defender doesn’t), then there is no DFA for $L$ with $< d$ states.

To prove that $L$ has no DFA with $< d$ states, enough to show a winning strategy for the Challenger in SuvxGame($L, d$)

**Fooling set** of size $d$

Fooling Set $S$: $\forall u, v \in S, u \neq v, \exists x \in \Sigma^*, |\{ux, vx\} \cap L| = 1$
Example

\[ L_k = \{ w \mid w \text{ ends in } 1(0+1)^{k-1} \} \]

An NFA with \(k+1\) states: \(Q = \{0,1,\ldots,k\}\), \(s = 0\), \(F = \{k\}\).
\(\delta(0,0) = \{0\}\), \(\delta(0,1) = \{0,1\}\). \(\delta(i,b) = \{i+1\} \) for \(1 \leq i < k\) and \(b \in \{0,1\}\).

DFA? Intuitively, need to remember last \(k\) bits, as any of them could turn out to be at the \(k^{th}\) position from end. DFA with a state for each possible prefix: \((1+1+2+4+\ldots+2^{k-1}) = 2^k\) states.

Claim: Any DFA for \(L_k\) must have at least \(d = 2^k\) states!

Winning strategy for challenger?
\(S = \{0,1\}^k\). For any \(u,v\), let \(x\) be as follows.

There is some position where \(u, v\) disagree. Say \(i^{th}\) position from right \((1 \leq i \leq k)\). Let \(x = 0^{k-i}\). Then \(ux, vx\) disagree at the \(k^{th}\) position from right. Hence, exactly one of them will be in \(L_k\).
Non-Regular Languages

e.g., Language $L = \{ 0^n1^n \mid n \geq 0 \}$

“Clearly” an automaton will need to count the number of 0’s and match it against the number of 1’s (if it is scanning the input bit by bit)

Cannot do that in a DFA

How do we prove it?

Show an infinite fooling set!
The Game

dSuvxGame(L)

<table>
<thead>
<tr>
<th>Defender</th>
<th>Challenger</th>
</tr>
</thead>
<tbody>
<tr>
<td>(claim: ∃ DFA for L)</td>
<td>(claim: no DFA for L)</td>
</tr>
</tbody>
</table>

- Defender
  - Picks
  - Picks distinct u,v ∈ S
  - Picks x ∈ Σ*

- Challenger
  - Picks a set S ⊆ Σ*, |S| ≥ d
  - Picks x ∈ Σ*

Challenger wins if |\{ux, vx\} ∩ L | = 1. Else Defender wins

**Theorem**: If there is a DFA for L, then Defender has a winning strategy in dSuvxGame(L)

∃d ∈ Z⁺ ∀S ⊆ Σ*, |S| ≥ d, ∃u,v ∈ S, u≠v, ∀x ∈ Σ*, ux ∈ L ↔ vx ∈ L
Proving no DFA

**Theorem**: If there is a DFA for $L$, then the Defender has a winning strategy in dSuvxGame($L$)

$$\exists d \in \mathbb{Z}^+ \ \forall S \subseteq \Sigma^*, |S| \geq d, \ \exists u,v \in S, u \neq v, \ \forall x \in \Sigma^*, \ ux \in L \iff vx \in L$$

$$\forall d \in \mathbb{Z}^+ \ \exists S \subseteq \Sigma^*, |S| \geq d, \ \forall u,v \in S, u \neq v, \ \exists x \in \Sigma^*, |\{ux,vx\} \cap L| = 1$$

**Corollary**: If the Challenger has a winning strategy in the dSuvxGame($L$) (so that the Defender doesn’t), then there is no DFA for $L$.

To prove that $L$ has no DFA, enough to show a winning strategy for the Challenger in dSuvxGame($L$)

an infinite Fooling set

**Fooling Set** $S$: $\forall u,v \in S, u \neq v, \ \exists x \in \Sigma^*, |\{ux,vx\} \cap L| = 1$
Non-Regularity: Examples

- \( L = \{0^n1^n \mid n \geq 0\} \)
  - Winning strategy for challenger?
  - \( S = \{0\}^* \). For any \( u,v \), let \( x = 1^{\|u\|} \) so that \( ux \in L \), \( vx \notin L \).

- \( L = \{w \mid w \text{ has equal number of 0s and 1s}\} \)
  - Same winning strategy as above

- \( L = \{0^p \mid p \text{ is a prime number}\} \)
  - Same fooling set as above! \( S = \{0\}^* \)
  - Rest of the strategy?
Non-Regularity: Examples

$L = \{ 0^p \mid p \text{ is a prime number} \}$

- Fooling set $S = \{0\}^*$. Rest of the strategy?
  - Given $u=0^i$, $v=0^j$, (say, $i < j$, w.l.o.g.) find a non-negative number $k$ s.t. exactly one of $i+k$ and $j+k$ is prime. Then, let $x=0^k$.
  - **Solution 1**: **Fact**: We can find arbitrarily large gaps between successive prime numbers, for arbitrarily large prime numbers.
    
    Let $\Delta = j-i$. Let $p_1$ and $p_2$ be successive primes s.t. $p_1 \geq i$ and $p_2 - p_1 > \Delta$. Let $k = p_1 - i$. Then $k+i = p_1$ is prime, $k+j = p_1 + \Delta < p_2$ is not.

  - **Solution 2**: Let $\Delta = j-i$. Let $p \geq i$ be a prime, Note that $p+r\Delta$ is prime for $r=0$, and non-prime for $r = p$. Hence $\exists$ some $r \in [0, p]$ s.t. $p+r\Delta$ is prime but $p + (r+1)\Delta$ is a non-prime. Set $k = p+r\Delta-i$. 

Fun exercise!

Hint: **primorial**
Non-Regularity via Closure

Recall: Several operations on languages that preserve regularity

If $L_1, L_2$ regular, so is $L_1 \text{ op } L_2$

Gives a way to prove non-regularity!

Suppose we already knew that $L_2$ is regular, but $L_1 \text{ op } L_2$ is not.

Then $L_1$ is not regular!
Non-Regularity: Examples

- Let $L = \{ w \mid w \text{ has unequal number of 0s and 1s} \}$.  
  - We know that $\overline{L}$ is not regular. Hence $L$ can’t be.

- Let $L = \{ 0^n1^n \mid n \geq 0 \} \cup \{ w \mid |wl| \text{ is odd} \}$
  - Let $L' = \{0^n1^n \mid n \geq 0 \}$ and $L'' = \{w \mid |wl| \text{ is odd} \}$. Then $L' = L - L''$. Now, $L''$ is regular. If $L$ regular, $L'$ will be too!

- Let $L_1 = \{ 0^n1^n \mid n \geq 0 \}$. Let $L = \{ w \mid w \in L_1 \text{ and } |wl| \equiv 0 \pmod{3} \}$. Prove that $L$ is not regular.
  - Use $L = L_1 \cap L_2$ where $L_2 = \{ w \mid |wl| \equiv 0 \pmod{3} \}$ ?
  - **No!** ($L_1$ not regular, $L_2$ regular) $\not\Rightarrow$ ($L_1 \cap L_2$ not regular)!
    - e.g., $L_2$ is a finite set
Non-Regularity via Closure

Recall: Several operations on languages that preserve regularity

If $L_1, L_2$ regular, so is $L_1 \text{ op } L_2$

Gives a way to prove non-regularity!

Suppose we already knew that $L_2$ is regular, but $L_1 \text{ op } L_2$ is not.

Then $L_1$ is not regular!
Non-Regularity: Examples

Let $L_1 = \{0^n1^n | n \geq 0\}$. Let $L = \{w | w \in L_1 \text{ and } |w| \equiv 0 \pmod{3}\}$. Prove that $L$ is not regular.

- Let $L' = \{0\}L\{1\}$ and $L'' = \{00\}L\{11\}$.
- If $L$ is regular, so are $L'$ and $L''$
- But $L_1 = L \cup L' \cup L''$
- Since $L_1$ not regular, $L$ is not regular
- Or directly use the fooling set argument: e.g., $S = \{0^{3i} | i \geq 0\}$
How to Pick a Fooling Set

Make sure you don’t put in any two strings which can let the defender win!

In particular, all the strings you include (except maybe one) should be prefixes of strings in the language e.g., for $L = \{ 0^n1^n \mid n \geq 0 \}$ don’t include 1 and 10 (say)

Intuitively, each prefix in the fooling set has a different value for a parameter that a machine will need to remember
Non-Regular ⇒ Infinite Fooling Set?

We saw that infinite fooling set ⇒ non-regular. Converse?

(Will this proof strategy always work, in principle? Or maybe for some non-regular languages the Defender has a winning strategy in dSuvxGame?)

Myhill-Nerode theorem

Converse does hold!
(Defender has a winning strategy only if L is regular.)

But not necessarily easy (or even possible) to compute the winning strategy from say, an English description.
FYI: Another Game

**PumpingGame**(L)

<table>
<thead>
<tr>
<th>Defender</th>
<th>Challenger</th>
</tr>
</thead>
<tbody>
<tr>
<td>(claim: ∃ DFA for L)</td>
<td>(claim: no DFA for L)</td>
</tr>
<tr>
<td>Picks</td>
<td>Picks w ∈ L,</td>
</tr>
<tr>
<td></td>
<td>x,y,z ∈ Σ* s.t.</td>
</tr>
<tr>
<td></td>
<td>w=xyz,</td>
</tr>
<tr>
<td></td>
<td>Picks i ∈ N</td>
</tr>
</tbody>
</table>

**Theorem**: If there is a DFA for L, then Defender has a winning strategy in PumpingGame(L). Hence if Challenger has a winning strategy, L not regular. [ Converse doesn’t hold. ]