Harvard CS 121 and CSCI E-207

Lecture 8: Non-Regular Languages; Minimization

Harry Lewis

September 29, 2009

- **Reading:** Sipser, §1.4.

Goal: Explicit Non-Regular Languages

It **appears** that a language such as

$$\begin{aligned} L &= \{a^{2^n} : n \geq 0\} \\ &= \{a, aa, aaaa, aaaaaaaa, \ldots\} \end{aligned}$$

can't be regular because the "gaps" in the set of possible lengths become arbitrarily large, and no DFA could keep track of them.

But this isn't a proof!

Proof that $L = \{a^{2^n} : n \geq 0\}$ is not regular

- Suppose it were. Then some DFA M accepts L.

- ...

A more general principle so we don't have to repeat essentially the same argument

Approach:

1. Prove some general property P of all regular languages.
2. Show that L does not have P.

- The property P is that for any sufficiently long string in a regular language L, some substring can be repeated to produce more strings in L.

Pumping Lemma (Basic Version)

If L is regular, then there is a number p (the pumping length) such that every string $s \in L$ of length at least p can be divided into $s = xyz$, where $y \neq \varepsilon$ and for every $n \geq 0$, $xy^nz \in L$.

$n = 1$ $\quad \boxed{x}\boxed{\quad y \quad}\boxed{z}$

$n = 0$ $\quad \boxed{x}\boxed{z}$

$n = 2$ $\quad \boxed{x}\boxed{\quad y \quad}\boxed{\quad y \quad}\boxed{z}$

$\cdots$

- Why is the part about p needed?

- Why is the part about $y \neq \varepsilon$ needed?

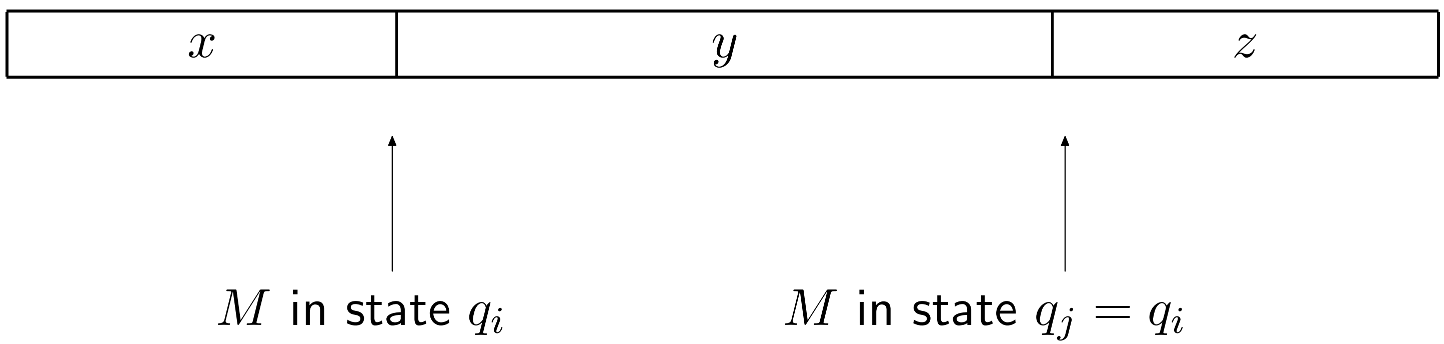
Proof of Pumping Lemma

(Another fooling argument)

- Since L is regular, there is a DFA M recognizing L.
- Let $p =$ # states in M.
- Suppose $s \in L$ has length $l \geq p$.
- M passes through a sequence of $l + 1 > p$ states while accepting s (including the first and last states): say, $q_0, \ldots, q_l$.
- Two of these states must be the same: say, $q_i = q_j$ where $i < j$

Pumping, continued

- Thus, we can break s into x, y, z where $y \neq \varepsilon$ (though x, z may equal ε):



- If more copies of y are inserted, M “can’t tell the difference,” i.e., the state entering y is the same as the state leaving it.

- So since $xyz \in L$, then $xy^nz \in L$ for all n.

Proof also shows (why?):

- We can take p = # states in smallest DFA recognizing L.

- Can guarantee division $s = xyz$ satisfies $|xy| \leq p$ (or $|yz| \leq p$).

Pumping Lemma Example

- Consider

$$L = \{x : x \text{ has an even \# of } a\text{'s and an odd \# of } b\text{'s}\}$$

- Since L is regular, pumping lemma holds.

  (i.e, every sufficiently long string s in L is “pumpable”)

- For example, if $s = aab$, we can write $x = \varepsilon$, $y = aa$, and $z = b$.

Pumping the even a's, odd b's language

- Claim: L satisfies pumping lemma with pumping length $p = 4$.

- Proof:

- **Q:** Can the Pumping Lemma be used to prove that L is regular?

Use PL to Show Languages are NOT Regular

Claim: $L = \{a^n b^n : n \geq 0\} = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$ is not regular.

Proof by contradiction:

- Suppose that L is regular.
- So L has some pumping length $p > 0$.
- Consider the string $s = a^p b^p$. Since $|s| = 2p > p$, we can write $s = xyz$ for some strings x, y, z as specified by lemma.
- Claim: No matter how s is partitioned into xyz with $y \neq \varepsilon$, we have $xy^2z \notin L$.
- This violates the conclusion of the pumping lemma, so our assumption that L is regular must have been false.

Strings of exponential lengths are a nonregular language

Claim: $L = \{a^{2^n} : n \geq 0\}$ is not regular.

Proof:

"Regular Languages Can't Do Unbounded Counting"

Claim: $L = \{w : w$ has the same number of a's and b's$\}$ is not regular.

Proof #1:

- Use pumping lemma on $s = a^p b^p$ with $|xy| \leq p$ condition.

"Regular Languages Can't Do Unbounded Counting"

Claim: $L = \{w : w$ has the same number of a's and b's$\}$ is not regular.

Proof #1:

- Use pumping lemma on $s = a^p b^p$ with $|xy| \leq p$ condition.

Proof #2:

- If L were regular, then $L \cap a^* b^*$ would also be regular.

Reprise on Regular Languages

Which of the following are necessarily regular?

- A finite language
- A union of a finite number of regular languages
- $\{x : x \in L_1$ and $x \notin L_2\}$, L_1 and L_2 are both regular
- A subset of a regular language

What Happens During the Transformations?

- NFA $\rightarrow$ DFA
- DFA $\rightarrow$ Regular Expression
- Regular Expression $\rightarrow$ NFA

Minimizing DFAs

Many different DFAs accept the same language. But there is a smallest one—and we can find it!

- Let M be a DFA
- Say that states p, q of M are *distinguishable* if there is a string w such that exactly one of $\delta^*(p, w)$ and $\delta^*(q, w)$ is final.
- Start by dividing the states of M into two equivalence classes: the final and non-final states

Minimizing DFAs, continued

- Break up the equivalence classes according to this rule: If p, q are in the same equivalence class but $\delta(p, \sigma)$ and $\delta(q, \sigma)$ are not equivalent for some $\sigma \in \Sigma$, then p and q must be separated into different equivalence classes
- When all the states that must be separated have been found, form a new and finer equivalence relation
- Repeat
- How do we know that this process stops?