cs3102: Theory of Computation

Class 5:
Non-Regular Languages

Spring 2010
University of Virginia
David Evans

Menu

- PS1, Problem 8
- Non-regular languages

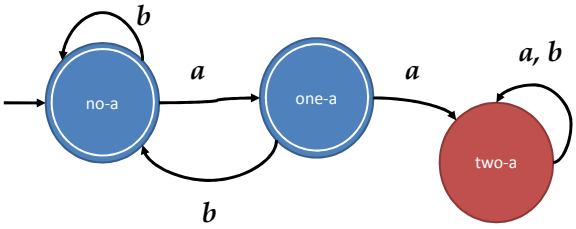
PS1 General Comments

- Proofs are for **making convincing arguments**, not for obfuscation.
  - e.g., If you assumed pizzas can only be cut through their center, it is obvious each cut makes 2 new pieces, and the number of pieces is $2n$. Adding an inductive proof only adds unnecessary confusion!
- Pledges are to remind you to be honorable
  - I assume you are all honorable whether you write a pledge or not
  - Writing a rote pledge (not what the PS collaboration policy says) doesn't work

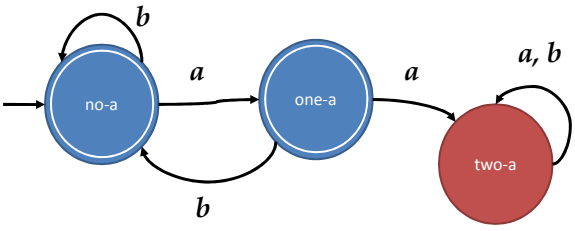
Problem 8

DFA that recognizes:

$\{ w \mid w \in [a, b]^*$ and w does not contain two consecutive as $\}$



How many strings of length n in this language?



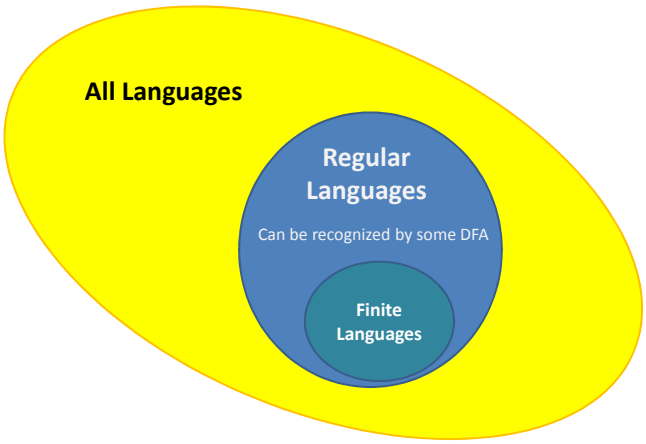
n	End in no-a state	End in one-a state	Total of length n

Fibonacci Strings!

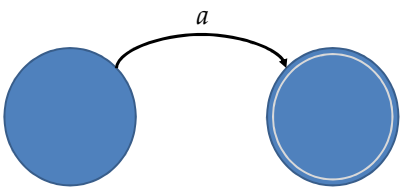
$E_1(n) = E_0(n\text{-}1)$ $E_0(n) = E_0(n\text{-}1)+E_0(n\text{-}2)$

$T(n) = 2E_0(n\text{-}1) + E_0(n\text{-}2)$
$= 2(E_0(n\text{-}2) + E_0(n\text{-}3)) + E_0(n\text{-}2)$
$= \mathbf{3E_0(n\text{-}2) + 2E_0(n\text{-}3)}$

$\mathbf{T(n - 1)} = 2E_0(n\text{-}2) + E_0(n\text{-}3)$
$+ \mathbf{T(n - 2)} = 2E_0(n\text{-}3) + E_0(n\text{-}4)$ $= 2E_0(n\text{-}2) + 3E_0(n\text{-}3) + E_0(n\text{-}4)$
$= 2E_0(n\text{-}2) + 2E_0(n\text{-}3) + (E_0(n\text{-}3) + E_0(n\text{-}4))$
$= 3E_0(n\text{-}2) + 2E_0(n\text{-}3) = \mathbf{T(n)}$



What DFAs **Cannot** Do



Keep track of a non-constant amount of state:
the amount of state to keep track of
cannot scale with the input string

DFAs can have any number of states, but the number of states in any particular DFA is fixed and finite.

The Language Recognition Game

- Players agree on a language A
- Player 1: draw a DFA M that attempts to recognize language A.
- Player 2: find a string s that M decides incorrectly.
  - Either, $s \in A$ but M **rejects**
  - Or, $s \notin A$ but M **accepts**

If Player 2 can find such a string, Player 2 wins. Otherwise, Player 1 wins.

If A is regular, Player 1 should always win.
If A is non-regular, Player 2 should always win.

Playing the Language Game

$A = a^*b^*$

Playing the Language Game

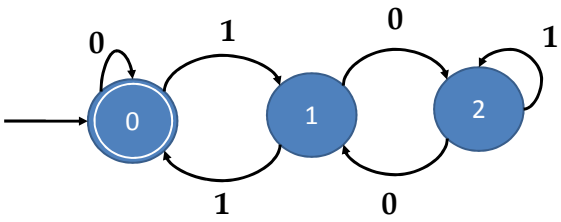
$A = \{ a^n b^n \mid n >= 0 \}$

Regular Language Game

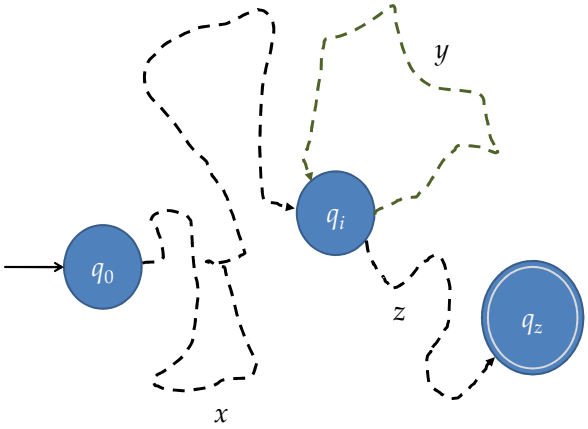
$A = \{ w \mid w \in [a,b]^*$ and w has more as than bs. $\}$

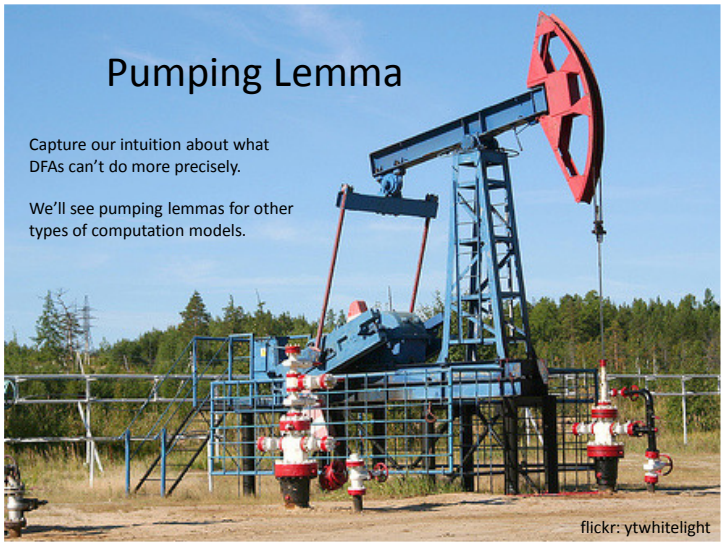
Regular Language Game

$A = \{ w \mid w \in [0,1]^*$ and w is divisible by 3 when interpreted as a binary number $\}$



If input string is longer than $|Q|$, some state must repeat.





Pumping Lemma *for Regular Languages*

If A is a regular language, then there is some number p (the *pumping length*) where for any string $s \in A$ and $|s| \geq p$, s may be divided into three pieces, $s = xyz$, such that $|y| > 0$, $|xy| \leq p$, and for any $i \geq 0$, $xy^iz \in A$.

Intuitively: you can go around the circle any number of times.

Using the PLRL: Proof by **Contradiction**

1. **Assume** A is **regular.**
2. Then, the pumping lemma is true for A: there is some number p (the *pumping length*) where for
   We can condense the first 2 steps into the statement: ``Assume A is regular and p is the pumping length for A.''
   and for any $i \geq 0$, $xy^iz \in A$.
3. **The creative part:** Identify a string s that can be built using p and show that there is no way to divide $s = xyz$ that satisfies the pumping lemma: for all possible divisions where $|y| > 0$ there is some $i \geq 0$ such that $xy^iz \notin A$.

Example: $A = \{ a^n b^n \mid n >= 0 \}$

$A = \{ a^n b^n \mid n >= 0 \}$ is not regular

Proof. Assume $\{a^n b^n | n \geq 0\}$ is regular and p is the pumping length for A.

Choose $s = a^p b^p$. $s \in A$ since we can choose $n = p$.

The pumping lemma requires that the repeating string is found in the first p symbols: $|xy| \leq p$. However we choose x and y, y can only contain as since the first p symbols in s are all as.

Since y only contains as and the pumping lemma requires $|y| > 0$, $xy^i z$ always contains p bs, but the number of as increases with i. But, A only contains strings with equal numbers of as and bs, so xy^2z must not be in A. □

$A = \{ w \mid w$ has more as than bs. $\}$

$A = \{ w \mid w \in [0,1]^*$ and w is divisible by 3 when interpreted as a binary number $\}$

Pumping lemma can be used to show a language is **not regular**. It starts by **assuming a language is regular and gets a contradiction**. This is no way to show a language is regular.

Regular Languages One-Slide Summary

A language is a *set of strings*.

A language is **regular** iff some DFA recognizes it.

DFAs, NFAs, and Regular Expressions are **equally powerful**

- They can recognize exactly the same set of languages = the regular languages
- Prove by construction: show how to construct a DFA that recognizes the same language as NFA, etc.

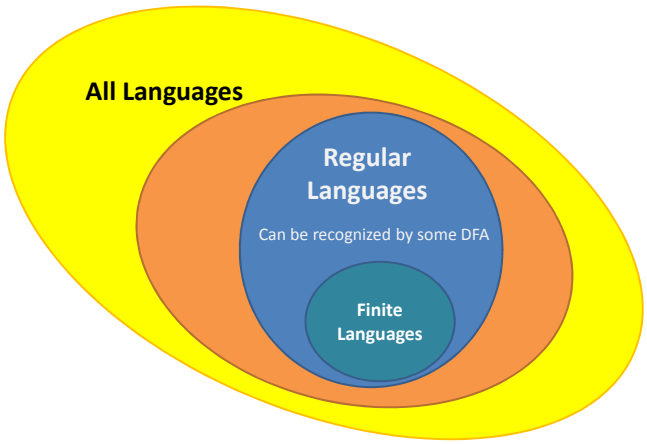
To prove a language is **regular**:

construct a DFA, NFA or RE that recognizes it

To prove a language is **not regular**:

show recognizing it requires keeping track of *infinite* state

use the *pumping lemma* to get a **contradiction**



Next week: machines that can recognize some non-regular language.

Return PS1

afg2s (Arthur Gordon) – dk8p

dr7jx (David Renardy) – jmd9xk

jth2ey (James Harrison) - pmc8p

ras3kd (Robyn Short) – yyz5w