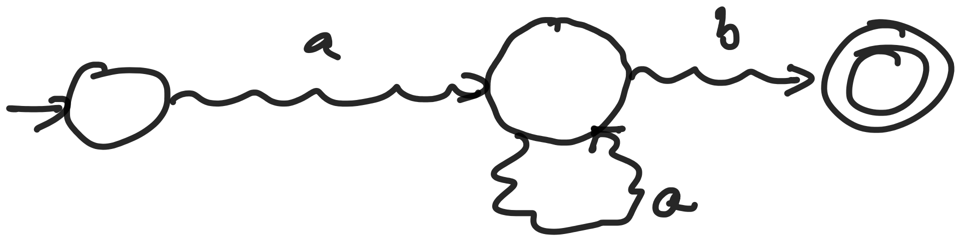
Lecture 7

Thursday, January 28, 2021 11:27 AM

Quick correction: A putative DFA to verify $\{a^n b^n \mid n \geq 0\}$.



EXAMPLE 1 $\Sigma = \{a\}$

$L = \{a^q \mid q$ a prime number$\}$

Demon picks some $p > 0$

I pick a^n where $n > p$ and n is a prime.

Demon has to pick $y = a^k$ where $k > 0$

$0 < k \leq p$

I choose some i (tell you in a minute)

New pumped string is

$a^{n+(i-1)k}$

I pick $i = n+1$

Length $= n + n \cdot k = n(k+1)$

Definitely not prime

This language is not regular.

Ex 2 $L = \{a^n b^m \mid n \neq m\}$ $\Sigma = \{a, b\}$

Intuitively this is not regular but proving it with the pumping lemma directly is not easy.

We can try $\bar{L}$ but this language is messy to describe.

But $\bar{L} \cap a^* b^* = \{a^n b^n \mid n \geq 0\}$

This is clearly not regular.

So $\bar{L}$ cannot be regular.

So L cannot be regular.

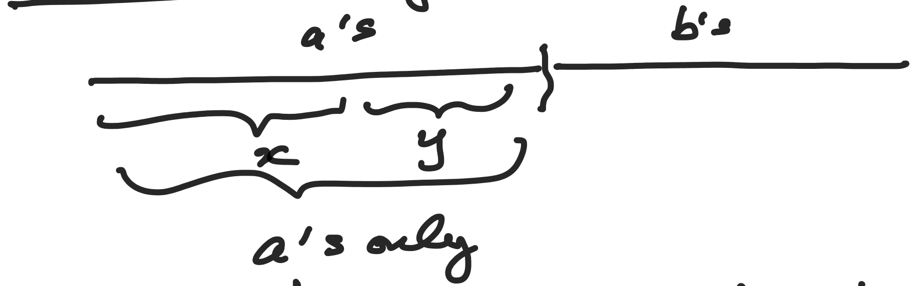
Ex 3 $L = \{a^i b^j \mid i > j\}$ $\Sigma = \{a, b\}$

Demon picks p

I pick $a^p b^{p-1}$

Demon is constrained to choose $|xy| \leq p$ so y has to consist

<u>exclusively</u> of a's.



So $y = a^k$ where $0 < k \le p$

I choose $i = 0$

New "pumped" string is

$a^{p-k} b^{p-1}$

$p-k$ is <u>not</u> strictly bigger than $p-1$.

(((())))) ✗

Balanced parens cannot be regular.

<u>EX 4</u> $\Sigma = \{0\}$ $L = \{0^{2^n} \mid n \geq 0\}$

2^n really does mean exponentiation but 0^x just means repeat 0 x times.

$L = \{0, 00, 0000, 00000000, \dots\}$

Demon chooses p

I choose m s.t. $2^m > p$ and I choose 0^{2^m} as my word w.

Demon chooses $y = 0^k$ $\quad 0 < k \le p$

I choose $i = 2$

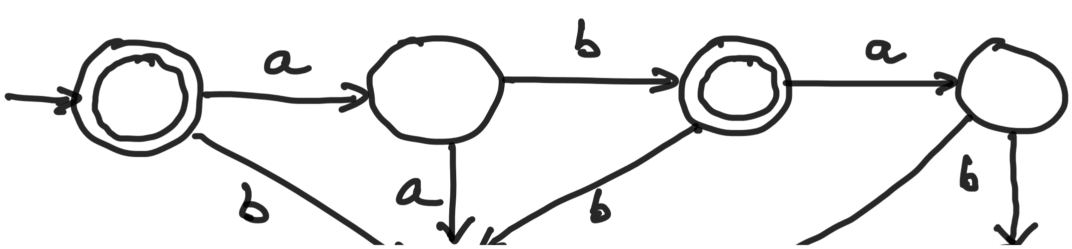
Pumped string $\quad xyyz = 0^{(2^m+k)}$

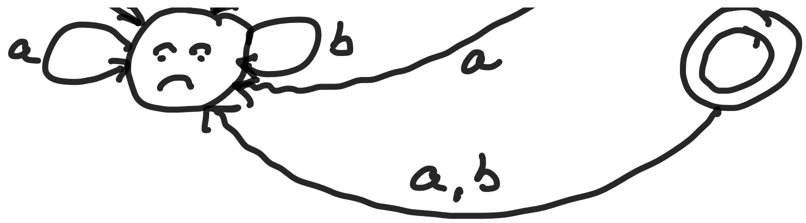
$$2^m < 2^m + k \le 2^m + p < 2^m + 2^m = 2^{m+1}$$

$2^m + k$ lies strictly between two consecutive powers of 2 and hence cannot be a power of 2.

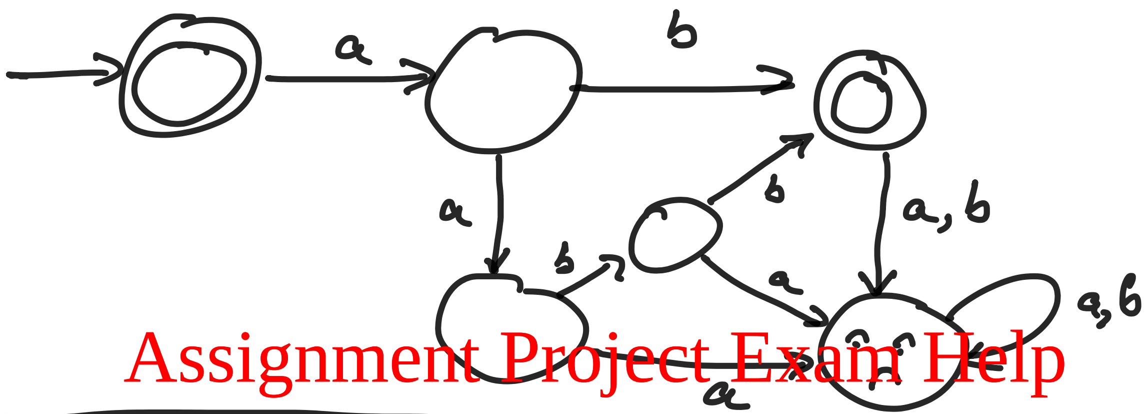
DFA's can do bounded counting or modular arithmetic.

$$L = \{a^n b^n \ \cancel{2 \ge n \ge 0}\} = \{\varepsilon, ab, abab\}$$





$$L = \{ a^n b^n \mid 0 \leq n \leq 2 \} = \{ \varepsilon, ab, aabb \}$$



MINIMIZATION of DFA

Size = # of states

We must count dead states.

GOAL Design a DFA with the fewest states.

FACT (I) There is a UNIQUE smallest DFA for any regular language

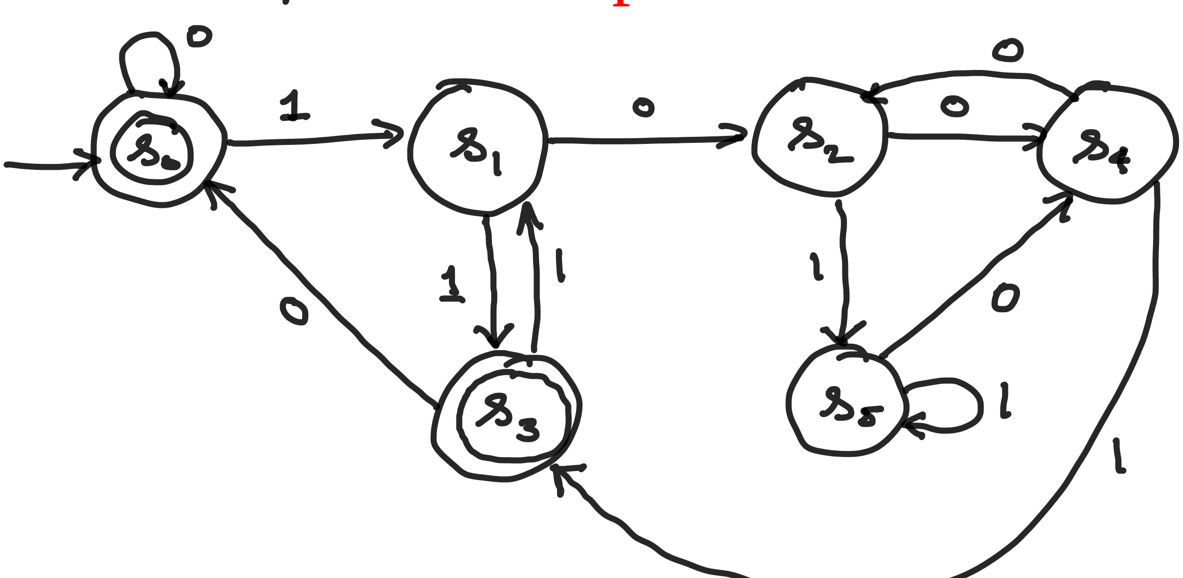
DFA for any regular language.

FACT (II) There is an algorithm to find the smallest DFA for a given regular language. This algorithm works by starting with any (correct) DFA and "combining" states.

MACHINE for recognizing whether numbers are divisible by 3:

$\Sigma = \{0, 1\}$

Let's keep track of the remainder mod 6



This does the job but

it has 6 states instead of 3

There must be unnecessary states.
There must be states that can be combined.

s_0 & s_3 : do we really need both of these?

$s_0 \xrightarrow{0} s_0$

$s_3 \xrightarrow{0} s_0$

$s_0 \xrightarrow{1} s_1$

$s_3 \xrightarrow{1} s_1$

Once you are in s_0 or s_3 at the very next step you are in the same state. Once you are in a state the future behaviour does not depend on how you got there.

We can deem (s_0, s_3) as

equivalent.

Similarly (s_1, s_4) behave "similarly" and (s_2, s_5) behave "similarly".

We can construct a new m/c by lumping together

$\{s_0, s_3\}$, $\{s_1, s_4\}$, $\{s_2, s_5\}$.