Theory of Computation
Week 5

Much of the material on this slides comes from the recommended textbook by Elaine Rich

Announcements

- Deadline for Assignment 1
  - 29/03/2020 (Sunday) 23:59.
- Midterm 1
  - Week 7:
    - 06/04/2020 – 8:00 to 9:00
    - **Venue: ~~BSC126~~ online**
  - Syllabus:
    - Contents (lecture/tutorial) covered up to week 5.

Detailed content

Weekly program

- ✓ Week 1 – Background knowledge revision: logic, sets, proof techniques
- ✓ Week 2 – Languages and strings. Hierarchies. Computation. Closure properties
- ✓ Week 3 – Finite State Machines: non-determinism vs. determinism
- ✓ Week 4 – Regular languages: expressions and grammars

➡ **Week 5 – Non regular languages: pumping lemma. Closure**

- ❑ Week 6 – Context-free languages: grammars and parse trees
- ❑ Week 7 – Pushdown automata
- ❑ Week 8 – Non context-free languages: pumping lemma and decidability. Closure
- ❑ Week 9 – Decidable languages: Turing Machines
- ❑ Week 10 – Church-Turing thesis and the unsolvability of the Halting Problem
- ❑ Week 11 – Decidable, semi-decidable and undecidable languages (and proofs)
- ❑ Week 12 – Revision of the hierarchy. Safety-critical systems
- ❑ Week 13 – Extra revision (if needed)

Key concepts from last week

- Regular Expressions (REs) are useful to define patterns

- For every RE there is an equivalent FSM

- A language is ***regular*** iff it can be defined with a regular expression

- A ***regular grammar*** *G* is a quadruple (*V*, Σ, *R*, S)

Key concepts from last week

- Given any DFSM *M*, there exists an algorithm *fsmtoregex* that constructs a regular expression that recognises *L*(*M*).
- Given any grammar *G*, there exists an algorithm *grammartofsm* that constructs a DFSM that also accepts *L*(*G*).
- The class of languages that can be defined with regular grammars, regular expressions and FSMs is exactly the regular languages.

Week 05 Videos

You already know

- ❑ All finite Languages are regular
- ❑ Pumping Lemma/Theorem
  - ❑ Long String
  - ❑ Concept of pumping
  - ❑ Basic understanding of the theorem

Videos to watch before lecture

Additional videos to watch for this week

Week 05 Lecture Outline

Non regular languages: pumping lemma. Closure

- Number of regular and non-regular languages
- How to prove a language L is regular
  - Closure properties of regular languages
- How to prove a language is not regular
  - Pumping lemma

REGULAR AND NONREGULAR LANGUAGES

a^*b^* is regular.

$\{a^n b^n : n \geq 0\}$ is not.

$\{w \in \{a, b\}^* :$ every a is immediately followed by $b\}$ is regular.

$\{w \in \{a, b\}^* :$ every a has a matching b somewhere and no b matches more than one a$\}$ is not

- Showing that a language is regular.
- Showing that a language is not regular.

REGULAR AND NONREGULAR LANGUAGES

Theorem: There is a countably infinite number of regular languages.

Proof:

Upper bound on number of regular languages:
number of FSMs (or regular expressions).

Lower bound on number of regular languages:

{a},{aa},{aaa},{aaaa},{aaaaa},{aaaaaa},…

are all regular. That set is countably infinite.

REGULAR AND NONREGULAR LANGUAGES

There is a countably infinite number of regular languages.

There is an uncountably infinite number of languages over any nonempty alphabet Σ.

- The set of languages defined on Σ is $\mathcal{P}(\Sigma^*)$

So there are *many* more nonregular languages than there are regular ones.

REGULAR LANGUAGES

Theorem: Every finite language is regular.

Proof: If L is the empty set, then it is defined by the regular expression $\varnothing$ and so is regular. If it is any finite language composed of the strings s_1, s_2, … s_n for some positive integer n, then it is defined by the regular expression:

$$s_1 \cup s_2 \cup \ldots \cup s_n$$

So it too is regular.

REGULAR LANGUAGES

Example:

Let $L = L_1 \cap L_2$, where:
$L_1 = \{a^n b^n, n \geq 0\}$, and
$L_2 = \{b^n a^n, n \geq 0\}$

$L = \{\varepsilon\}$

SHOWING THAT *L* IS REGULAR

1. Show that *L* is finite.
2. Exhibit an FSM for *L*.
3. Exhibit a regular expression for *L*.
4. Show that the number of equivalence classes of $\approx_L$ is finite.
5. Exhibit a regular grammar for *L*.
6. Exploit the closure theorems.

CLOSURE PROPERTIES OF REGULAR LANGUAGES

- Union
- Concatenation
- Kleene star

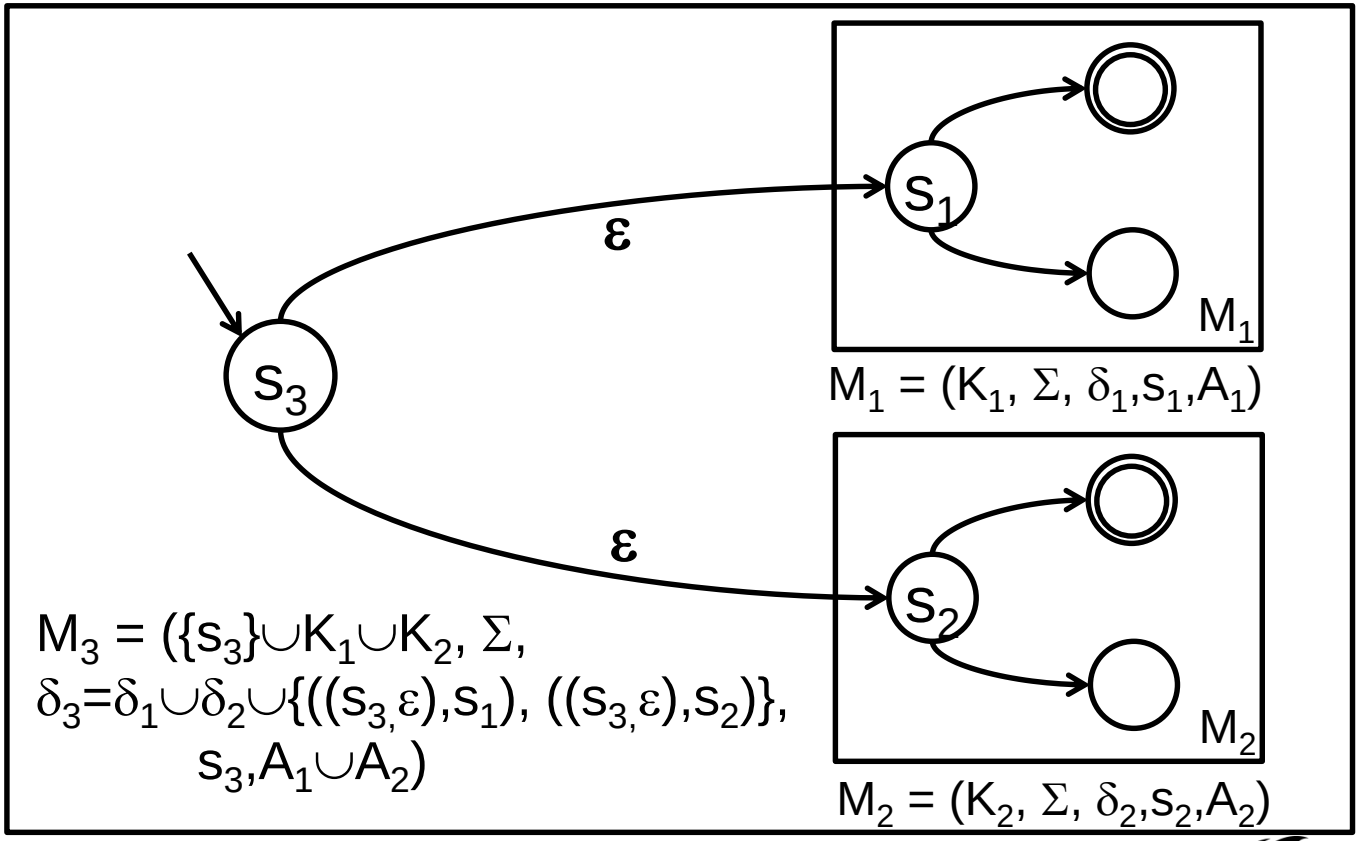
→ *We saw how to prove these last Monday*

- Complement
- Intersection
- Difference
- Reverse
- Letter substitution

REs vs. FSMs
Kleene's theorem

From last week's lecture

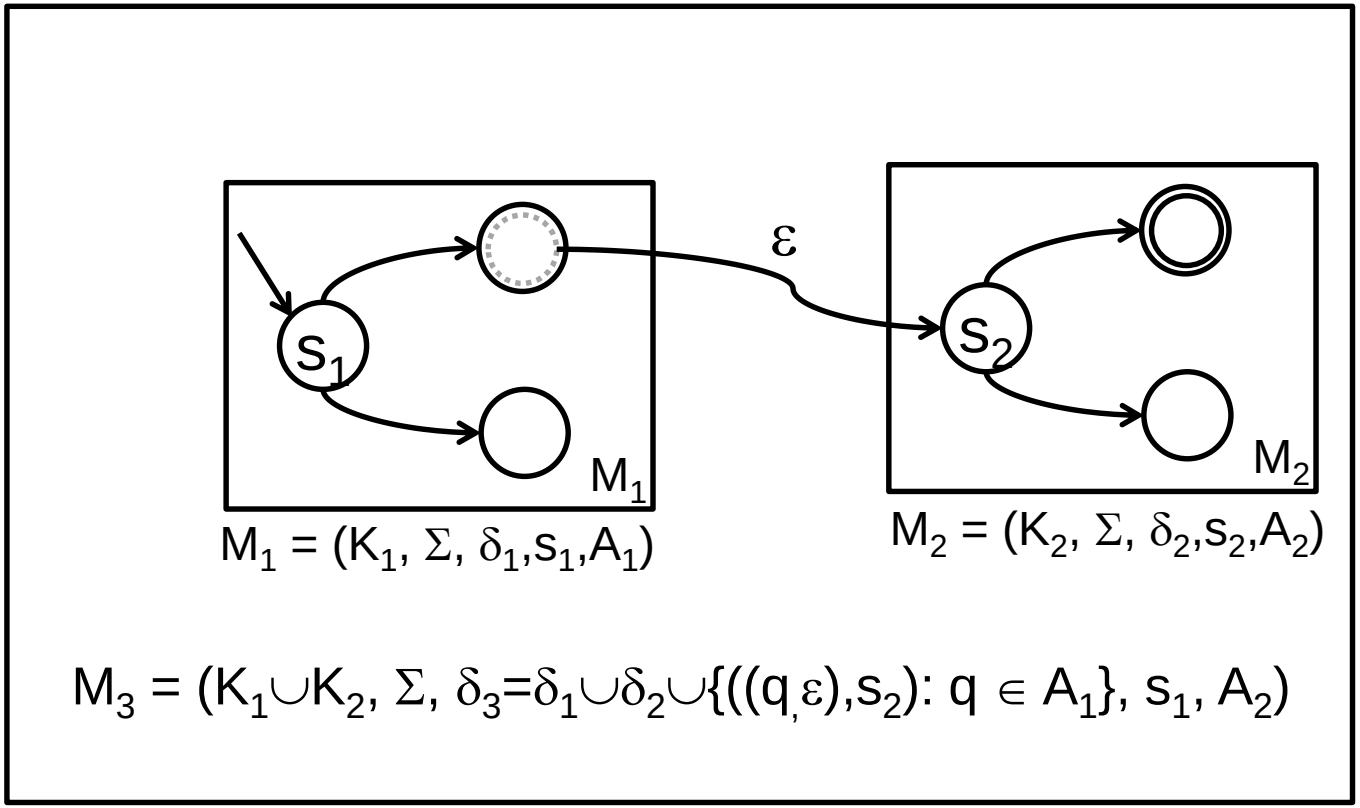
If α is the regular expression $\beta \cup \gamma$ and if both $L(\beta)$ and $L(\gamma)$ are regular, then we construct M_3 such that $L(M_3)=L(\alpha)=L(\beta) \cup L(\gamma)$:



REs vs. FSMs
Kleene's theorem

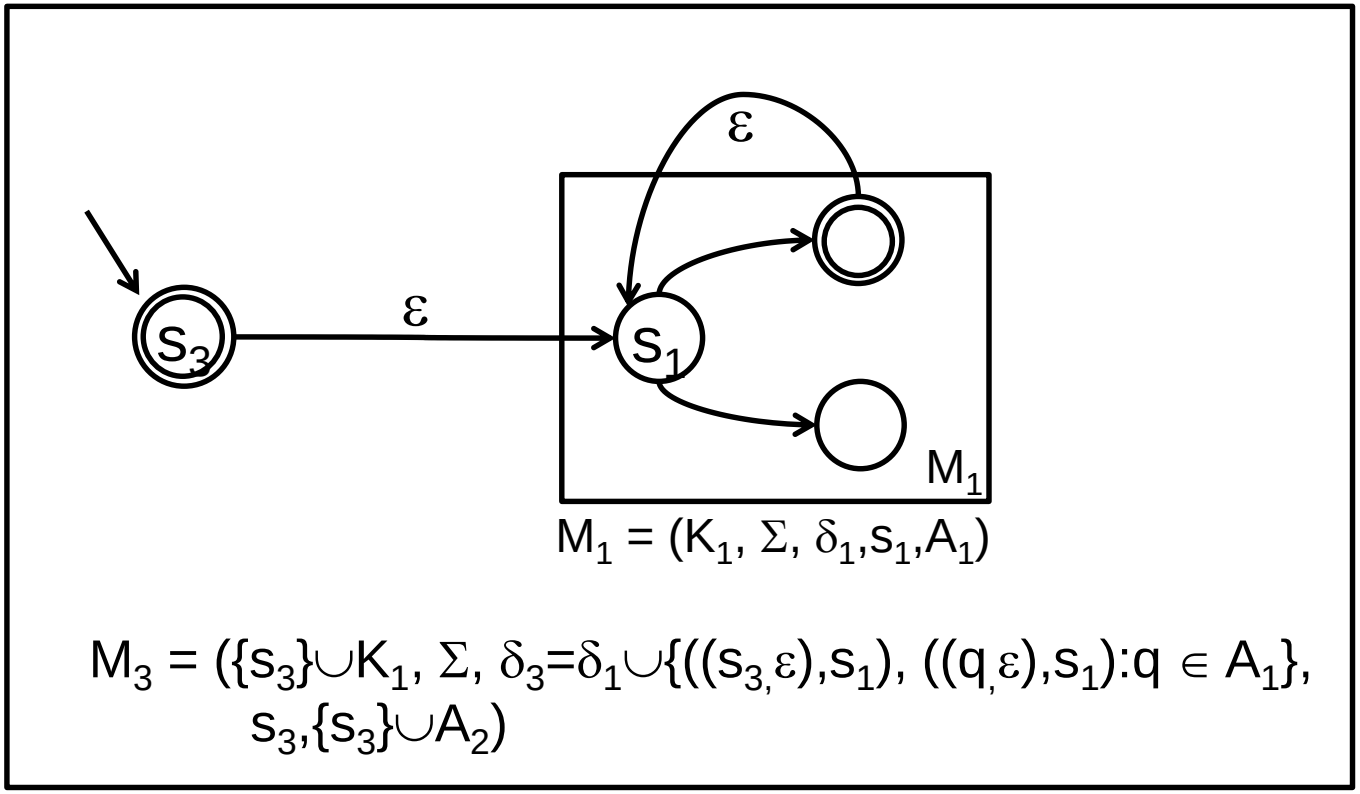
From last week's lecture

If α is the regular expression $\beta\gamma$ and if both $L(\beta)$ and $L(\gamma)$ are regular, then we construct M_3 such that $L(M_3)=L(\alpha)=L(\beta)\ L(\gamma)$:



REs vs. FSMs
Kleene's theorem

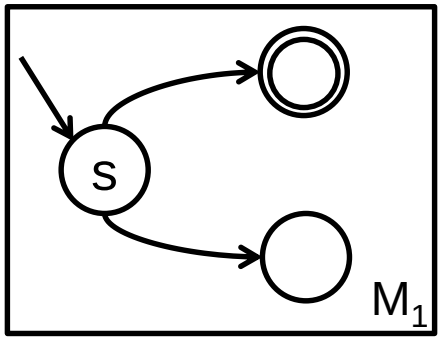
If α is the regular expression β^* and $L(\beta)$ is regular, then we construct M_3 such that $L(M_3)=L(\alpha)=L(\beta)^*$:



$M_1 = (K_1, \Sigma, \delta_1, s_1, A_1)$

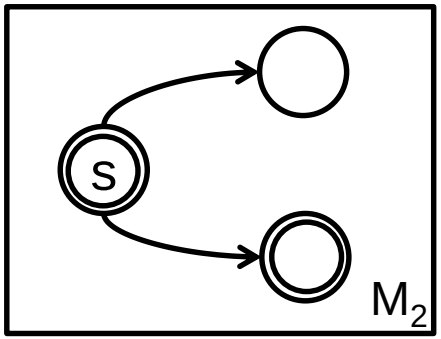
$M_3 = (\{s_3\} \cup K_1, \Sigma, \delta_3 = \delta_1 \cup \{((s_3, \varepsilon), s_1), ((q, \varepsilon), s_1) : q \in A_1\}, s_3, \{s_3\} \cup A_2)$

CLOSURE OF REGULAR LANGUAGES UNDER COMPLEMENT



$M_1 = (K, \Sigma, \delta, s, A)$

$L(M_1)$



$M_2 = (K, \Sigma, \delta, s, K - A)$

$\neg(L(M_1))$

CLOSURE OF REGULAR LANGUAGES UNDER COMPLEMENT

What about:

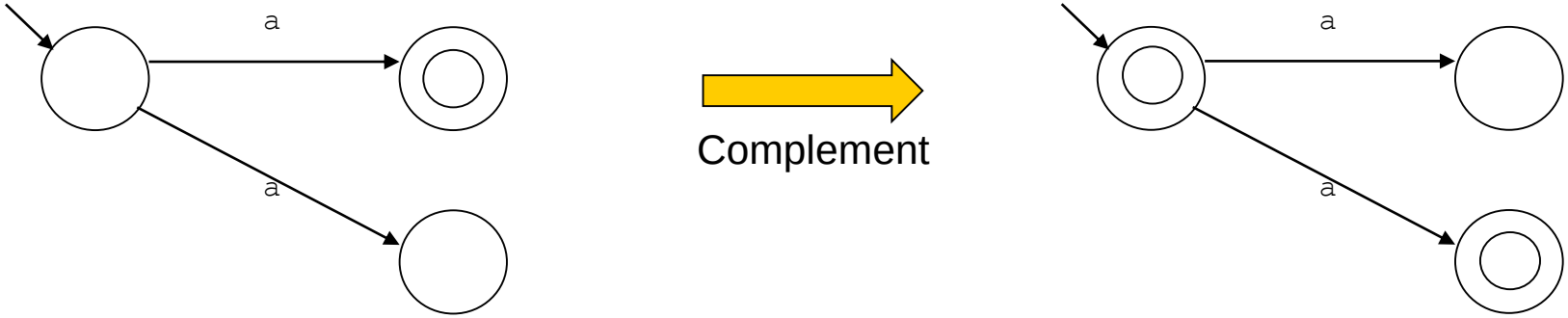
On input `bbb`

Both machine rejects bbb? Why?

Need to make sure all transitions are explicitly represented before we swap accepting / non-accepting states.

CLOSURE OF REGULAR LANGUAGES UNDER COMPLEMENT

What about:



On input a

Both machine accepts a? Why?

Only works for DFSMs. So need to convert any NDFSM into an equivalent DFSM before apply it.

CLOSURE OF REGULAR LANGUAGES UNDER COMPLEMENT

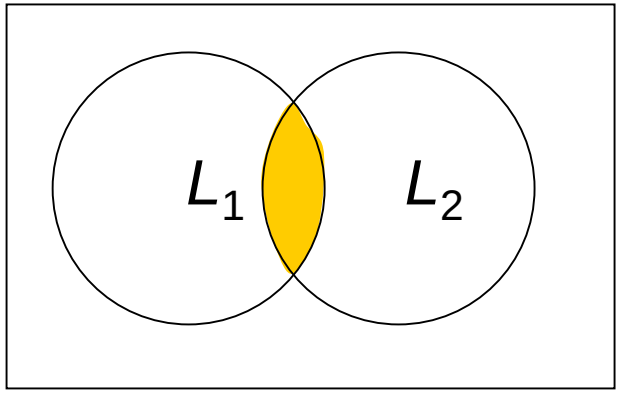
Given: an FSM *M*,

Construct a new machine to accept $\neg L(M)$:

- If necessary, use *ndfsmtodfsm* to construct *M′*, a deterministic equivalent of *M*.
- Make sure that *M′* is described completely (by adding dead states).
- Swap accepting and nonaccepting states.

CLOSURE OF REGULAR LANGUAGES UNDER INTERSECTION

$L_1 \cap L_2 =$



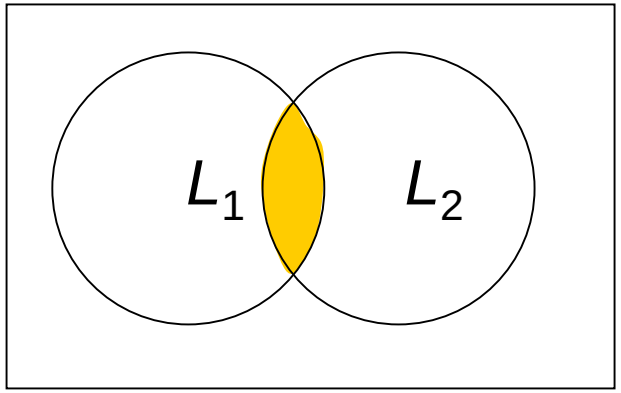
Write this in terms of operations we have already proved closure for:

- Union
- Concatenation
- Kleene star
- Complementation

CLOSURE OF REGULAR LANGUAGES UNDER INTERSECTION

$L_1 \cap L_2 =$

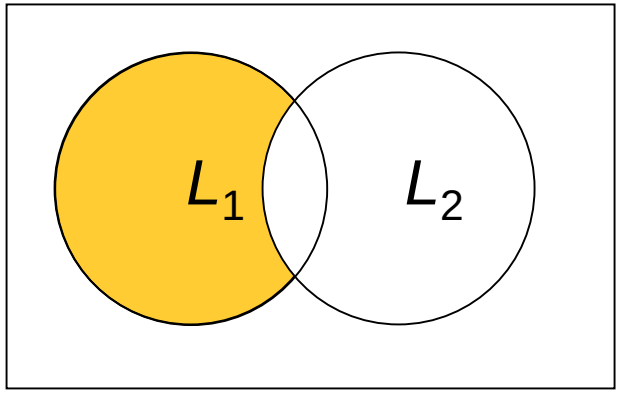
$\neg(\neg L_1 \cup \neg L_2)$



De Morgan's Law!

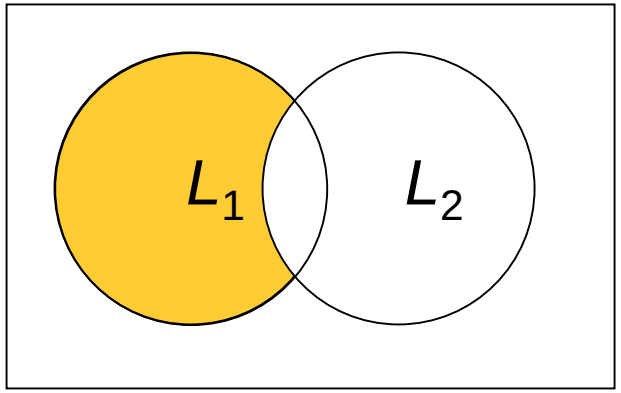
CLOSURE OF REGULAR LANGUAGES UNDER DIFFERENCE

$L_1 - L_2 =$

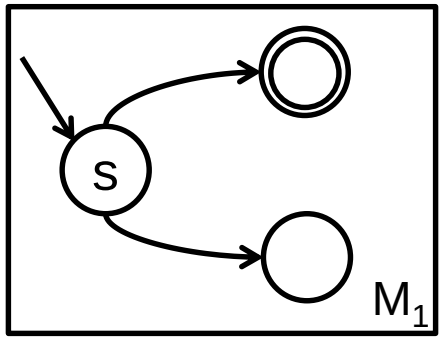


CLOSURE OF REGULAR LANGUAGES UNDER DIFFERENCE

$$L_1 - L_2 = L_1 \cap \neg L_2$$

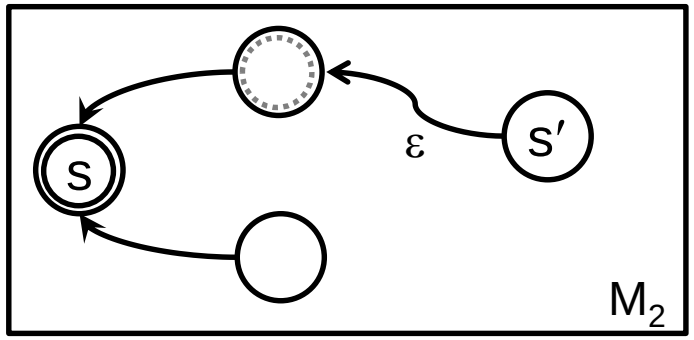


CLOSURE OF REGULAR LANGUAGES UNDER REVERSE



$M_1 = (K, \Sigma, \delta, s, A)$

$L(M_1)$



$M_2 = (K \cup \{s'\}, \Sigma, \delta', s', s)$

$\delta' = \{(s', \varepsilon)=q: q \in A\} \cup$
$\{(p, a)=q: \delta(q,a)=p \ \forall \ q \in K$
and $\forall \ a \in \Sigma\}$

CLOSURE OF REGULAR LANGUAGES UNDER LETTER SUBSTITUTION

Let Σ_1 and Σ_2 be alphabets.

Let *sub* be any function from Σ_1 to Σ_2^*.

Example:

Let: $\Sigma_1 = \{\mathtt{a}, \mathtt{b}\}$,
$\Sigma_2 = \{\mathtt{0}, \mathtt{1}\}$,

sub(a) = 0, and
sub(b) = 11.

CLOSURE OF REGULAR LANGUAGES UNDER LETTER SUBSTITUTION

letsub is a letter substitution function iff:

$$letsub(L_1) = \{w \in \Sigma_2^* : \exists y \in L_1 \text{ and } w = y \text{ except that: every character } c \text{ of } y \text{ is replaced by } sub(c)\}.$$

Example:

$sub(\mathtt{a}) = \mathtt{0}$, and
$sub(\mathtt{b}) = \mathtt{11}$.

Then $letsub(\{\mathtt{a}^n\mathtt{b}^n, n \geq 0\}) = \{\mathtt{0}^n\mathtt{1}^{2n}, n \geq 0\}$

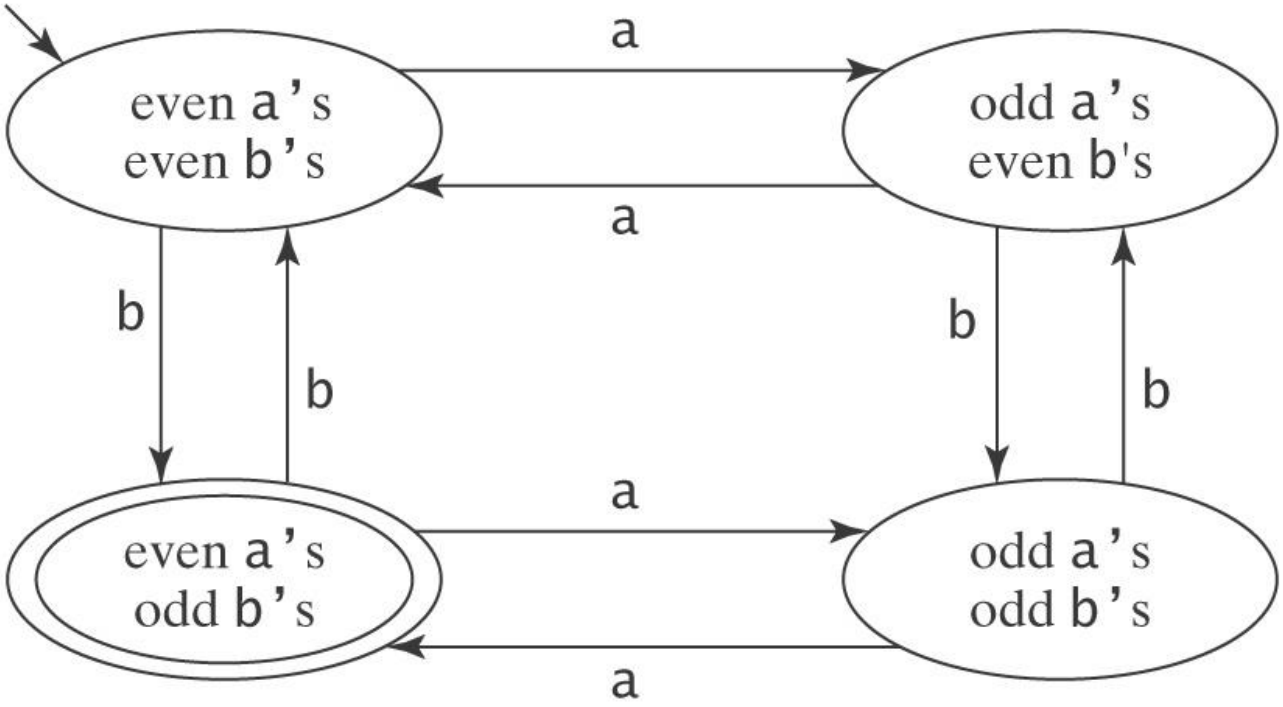
DIVIDE-AND-CONQUER

Let $L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w$ contains an even number of a's and an odd number of b's and all a's come in runs of three$\}$.

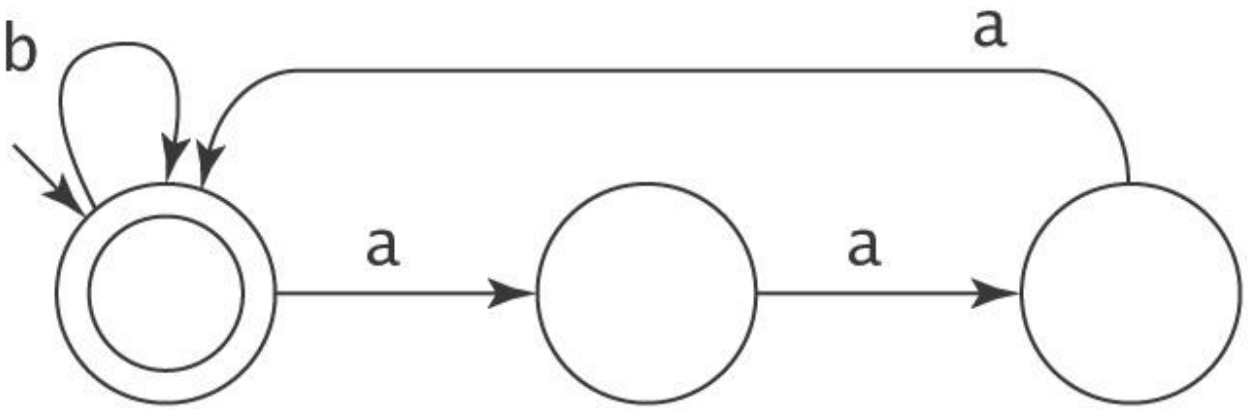
$L = L_1 \cap L_2$, where:

- $L_1 = \{w \in \{\texttt{a}, \texttt{b}\}^* : w$ contains an even number of a's and an odd number of b's$\}$, and
- $L_2 = \{w \in \{\text{a, b}\}^* :$ all a's come in runs of three$\}$

L_1 IS REGULAR



L_2 IS REGULAR



DON'T TRY TO USE CLOSURE BACKWARDS

One Closure Theorem:

If L_1 and L_2 are regular, then so is

$$\underline{L} = \underline{L_1} \cap \underline{L_2}$$

But if L is regular, what can we say about L_1 and L_2?

$$\underline{L} = L_1 \cap L_2$$

DON'T TRY TO USE CLOSURE BACKWARDS

One Closure Theorem:

If L_1 and L_2 are regular, then so is

$$\underline{L} = \underline{L_1} \cap \underline{L_2}$$

But if L is regular, what can we say about L_1 and L_2?

$$\underline{L} = L_1 \cap L_2$$

$\texttt{ab} = \texttt{ab} \cap (\texttt{a} \cup \texttt{b})^*$ (they may be regular)

DON'T TRY TO USE CLOSURE BACKWARDS

One Closure Theorem:

If L_1 and L_2 are regular, then so is

$$\underline{L} = \underline{L_1} \cap \underline{L_2}$$

But if L is regular, what can we say about L_1 and L_2?

$$\underline{L} = L_1 \cap L_2$$

$\mathtt{ab} = \mathtt{ab} \cap (\mathtt{a} \cup \mathtt{b})^*$ (they may be regular)

$\mathtt{ab} = \mathtt{ab} \cap \{\mathtt{a}^n\mathtt{b}^n, n \geq 0\}$ (they may not be regular)

DON'T TRY TO USE CLOSURE BACKWARDS

Another Closure Theorem:

If L_1 and L_2 are regular, then so is

$$L = \underline{L_1} \ \underline{L_2}$$

But if L_2 is not regular, what can we say about L?

$$L = L_1 \, L_2$$

DON'T TRY TO USE CLOSURE BACKWARDS

Another Closure Theorem:

If L_1 and L_2 are regular, then so is

$$L = \underline{L_1} \ \underline{L_2}$$

But if L_2 is not regular, what can we say about L?

$$L = L_1 L_2$$

$$\{\mathtt{aba}^n\mathtt{b}^n : n \geq 0\} = \{\mathtt{ab}\}\{\mathtt{a}^n\mathtt{b}^n : n \geq 0\}$$

DON'T TRY TO USE CLOSURE BACKWARDS

Another Closure Theorem:

If L_1 and L_2 are regular, then so is

$$L = L_1 \; L_2$$

But if L_2 is not regular, what can we say about L?

$$L = L_1 \, L_2$$

$$\{\mathtt{aba}^n\mathtt{b}^n : n \geq 0\} = \{\mathtt{ab}\} \, \{\mathtt{a}^n\mathtt{b}^n : n \geq 0\}$$

$$\{\mathtt{aaa}^*\} = \{\mathtt{a}^*\} \, \{\mathtt{a}^p : p \text{ is prime}\}$$

SHOWING THAT A LANGUAGE IS NOT REGULAR

Every regular language can be accepted by some FSM.

- Finite or infinite

FSM can only use a finite amount of memory to record essential properties.

- Can accept infinite number of strings of a regular language

SHOWING THAT A LANGUAGE IS NOT REGULAR

The only way to generate/accept an infinite language with a finite description is to use:

- Kleene star (in regular expressions), or
- cycles (in automata).

This forces some kind of simple repetitive cycle within the strings.

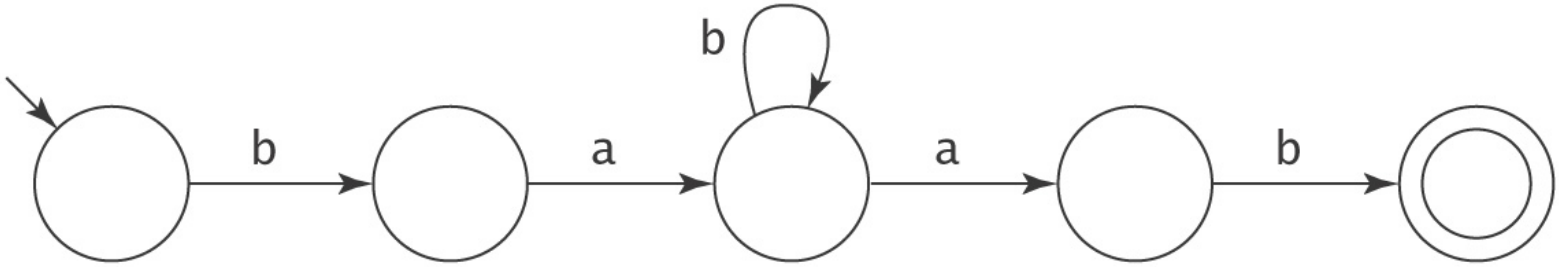
Example:

`ab*a` generates `aba`, `abba`, `abbba`, `abbbba`, etc.

Example:

$\{a^n : n \geq 1$ is a prime number$\}$ is not regular.

EXPLOITING THE REPETITIVE PROPERTY



If an FSM with n states accepts any string of length $\geq n$, how many strings does it accept?

L = `bab*ab`

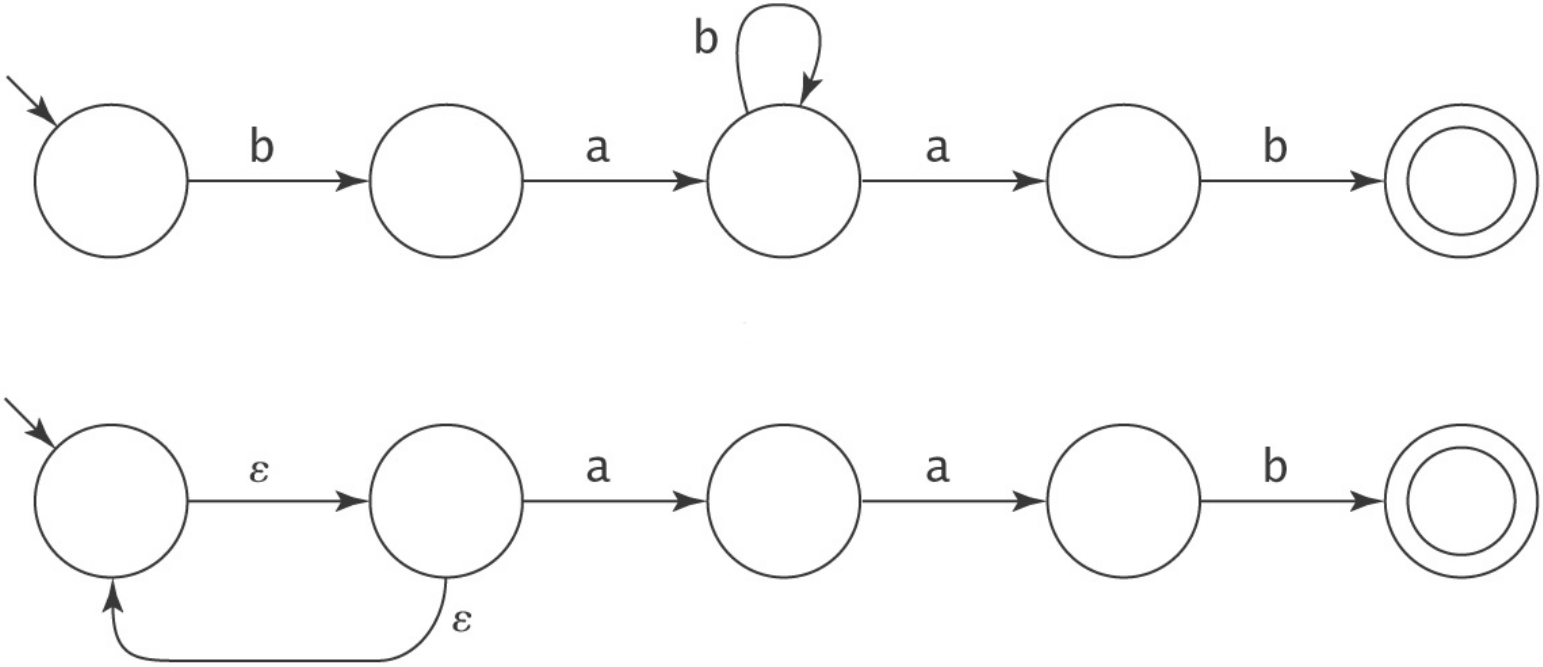
`b a b a b`

x y z

xy^*z must be in L.

So L includes: `baab`, `babab`, `babbab`, `babbbbbbbbbbab`

EXPLOITING THE REPETITIVE PROPERTY



What is the longest string that a 5-state FSM can accept?

THEOREM – LONG STRINGS FORCE REPEATED STATES

Theorem: Let $M = (K, \Sigma, \delta, s, A)$ be any DFSM. If M accepts any string of length $|K|$ or greater, then that string will force M to visit some state more than once (thus traversing at least one loop).

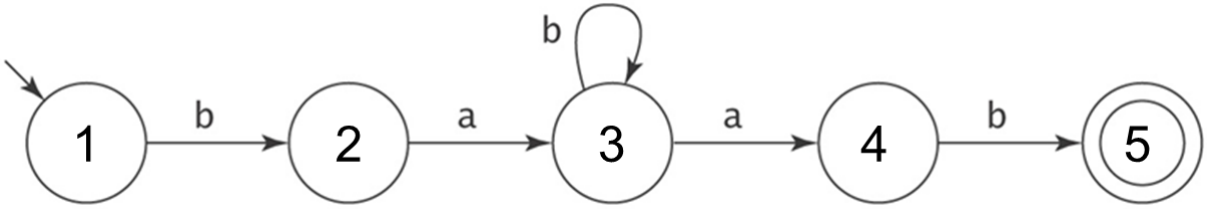
Proof: M must start in one of its states. Each time it reads an input character, it visits some state. So, in processing a string of length n, M creates a total of $n + 1$ state visits. If $n+1 > |K|$, then, by the pigeonhole principle, some state must get more than one visit. So, if $n \geq |K|$, then M must visit at least one state more than once.

The Concept of Pumping in and out

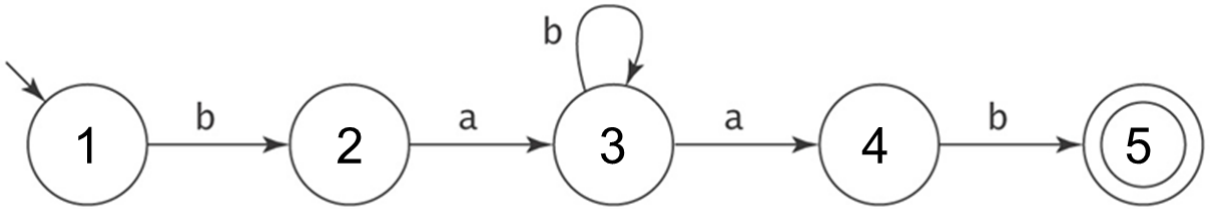
Suppose M = (K, Σ, δ, s, A) and there exist a "long" string w (i.e. $|w| \geq |K|$) and $w \in$ L(M).

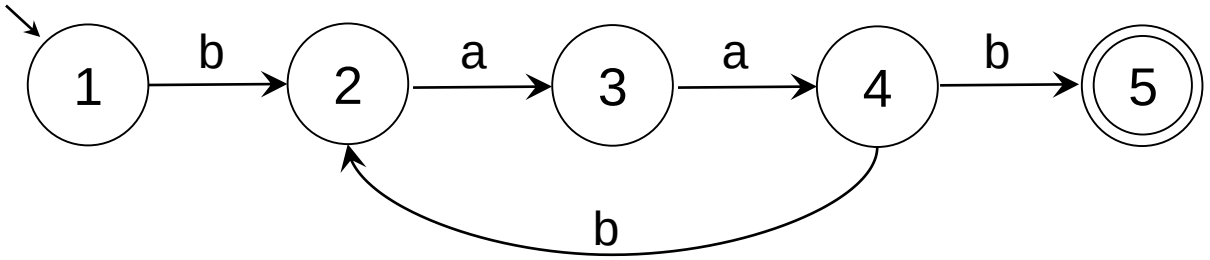
e.g. $w = babab$ for the DFA below

- Then M must go through at least one loop to process w
  - Here the machine visits the state 3 twice
- There is some string y of w that drove M through at least one loop
  - Here $y = b$ in $w = ba\mathbf{b}ab$
- If we excise y from w then the remaining string also be in L(M)
  - $baab$ is also accepted by the machine
- If we splice in one or more extra copies of y immediately adjacent to the original one those string will also be in L(M)
  - $ba\mathbf{bb}ab$ or $ba\mathbf{bbbbb}ab$ I also acceped by the machine



The Concept of Pumping in and out





The Concept of Pumping in and out

This property of FSM, and the languages they can accept, is the basis for a powerful tool for showing that a language is not regular

If a language L contains even a single **long string** that cannot be pumped in then it is not accepted by any FSM

Therefore L is not regular

THE PUMPING THEOREM FOR REGULAR LANGUAGES

If L is regular, then every **long string** in L is pumpable.

So, $\exists k \geq 1$

($\forall$ strings $w \in L$, where $|w| \geq k$

($\exists\ x, y, z$ ($w = xyz$,
$|xy| \leq k$,
$y \neq \varepsilon$, and
$\forall q \geq 0$ ($xy^q z$ is in L)))).

THE PUMPING THEOREM : PROOF

If L is regular, L is accepted by some DFSM M= M = (K,Σ,δ, s, A)

Suppose k = |K| and $w \in$ L where $|w| \geq$ k

To accept w, M must traverse some loop at least once.

We carve w up as follows:

$$w = x\ y\ z$$

y : the first substring to drive M through a loop
x : the part of w that precedes y
z : the part of w that follows y

For example, if w = `bab*ab`

b a	b	a b
x	y	z

THE PUMPING THEOREM : PROOF

We need to show that each of the three conditions must hold:

- $|xy| \leq k$:
  - M must traverse a loop for the first time by the time it has read k characters.
  - It can read k-1 characters without revisiting any state
  - But the k-th character must, if not earlier character already has, take M to a state that it has visited before

- $y \neq \varepsilon$: Since M is deterministic, no loop can be traversed by ε

- $\forall q \geq 0$ $(xy^q z \in L)$:
  - y can be pumped out and still the resulting string is in L
  - y can be pumped in and still the resulting string is in L

THE PUMPING THEOREM FOR REGULAR LANGUAGES

Important:

We use the pumping theorem to prove that a language is ***not*** regular.

Why?

THE PUMPING THEOREM FOR REGULAR LANGUAGES

Important:

We use the pumping theorem to prove that a language is ***not*** regular.

Why?

We would have to prove that $\forall$ (for all) strings in L the 'pumping' property holds… and there is an infinite number of them.

We use the pumping theorem to prove that a language is not regular constructing a ***proof by contradiction***

EXAMPLE: L={$a^N b^N$: $N \geq 0$} IS NOT REGULAR

If we can find <u>one</u> k such that |w| ≥ k and the conditions of the Pumping Theorem are not satisfied

Lets choose $w = a^k b^k$. Since |w| = 2k, w is "long enough" and it is in L.

So |w| must satisfy the conditions of the Pumping Theorem

Show that there is no x, y, z where $w=xyz$ with the required properties:

1. $|xy| \leq k$,
2. $y \neq \varepsilon$,
3. $\forall\ q \geq 0$ ($xy^q z$ is in L).

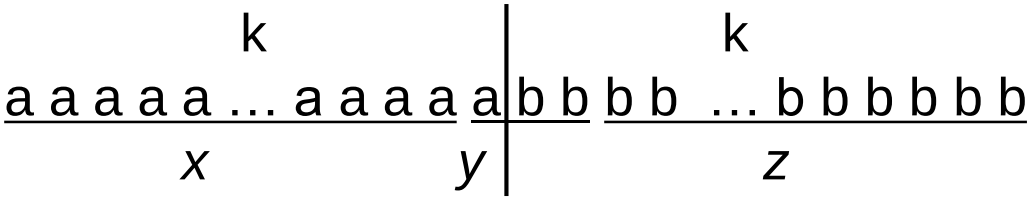
1. Since $|xy| \leq k$, y must occur within the first k characters
   assume $y = a^p$ for some p
2. Since $y \neq \varepsilon$, p must be greater than 0
3. Since $\forall\ q \geq 0$ ($xy^q z$ is in L), lets assume q=2 [pumping in one q]
   resulting string is $a^{k+p} b^k$ which is not in L

Hence L is not regular.

EXAMPLE: $\{a^N b^N : N \geq 0\}$ IS NOT REGULAR

k is the number from the Pumping Theorem.

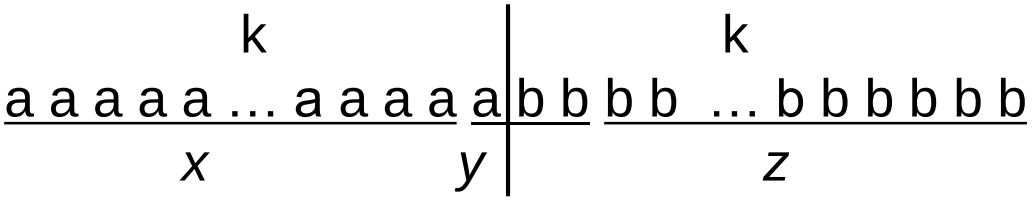
Lets choose $w = a^k b^k$. Since $|w| = 2k$, w is "long enough" and it is in L.



$y = a^p$ for some $p > 0$. $x = ? : [|xy| \leq k]$; $z = ?$

a a a a a a a a … a a a a a a a b b b b … b b b b b b √
a a a a a a a a … a a a a a a a b b b b … b b b b b b √
a a a a a a a a … a a a a a a a b b b b … b b b b b b √
a a a a a a a a … a a a a a a a b b b b … b b b b b b √
a a a a a a a a … a a a a a a a b b b b … b b b b b b ×
a a a a a a a a … a a a a a a a b b b b … b b b b b b ×
a a a a a a a a … a a a a a a a b b b b … b b b b b b ×
…..

EXAMPLE: $\{a^N b^N : N \geq 0\}$ IS NOT REGULAR



$y = a^p$ for some $p > 0$. x = ? : [$|xy| \leq$ k]; z = ?

a a a a a a a a … a a a a a a a b b b b … b b b b b b

$\forall$ $q \geq 0$ ($xy^q z$ is in L):

lets assume q=2 [pumping in one q] : resulting string is $a^{k+p}b^k \notin$ L

a a a a a a a a a a … a a a a a a a b b b b … b b b b b b

lets assume q=0 [pumping out q] : resulting string is $a^{k-p}b^k \notin$ L

a a a a a … a a a a a a a b b b b … b b b b b b

EXAMPLE: Tricks for Using Pumping Theorem

- It is sufficient to find one long string w that is in L but not pumpable.
- We need to show that there is **no way** to curve w up into x, y, z such that all the three conditions hold.
- It is **not sufficient** to pick an arbitrary y and show that it does not work.
- We need to show that there is **no value** of y that works.
- We need to consider all the possible classes of value for y.
  - Sometimes there is only one class
  - Sometimes there are more than one

EXAMPLE: $\{a^N b^N : N \geq 0\}$ IS NOT REGULAR

k is the number from the Pumping Theorem.

Choose w to be $a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$ ("long enough").

1 : $\{\lceil k/2 \rceil\}$		2 : $\{\lceil k/2 \rceil\}$	
a a a a a … a a a a	a	b b	b b … b b b b b b
x	y		z
x		y	z
x			y z

We show that there is no x, y, z with the required properties:

$|xy| \leq k$, $y \neq \varepsilon$, $\forall\ q \geq 0$ ($xy^q z$ is in L).

Three cases to consider:

- y falls in region 1:
- y falls across regions 1 and 2:
- y falls in region 2:

What You Need to Write

We prove that $L = \{a^n b^n : n \geq 0\}$ is not regular

Let $w = a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$. (If not completely obvious, as in this case, show that w is in fact in L.)

There are 3 cases for where y could occur:

```
aaaaa.....aaaaaa  | bbbbb.....bbbbbb
      1           |       2
```

So y can fall in:

(1): $y = a^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^{k+p}b^k$. But this string is not in L, since it has more a's than b's. .

(2): $y = b^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^k b^{k+p}$. But this string is not in L, since it has more b's than a's.

(1, 2): $y = a^p b^r$ for some non-zero p and r. Let $q = 2$. The resulting string will have interleaved a's and b's, and so is not in L.

Thus L is not regular.

THE PUMPING THEOREM FOR REGULAR LANGUAGES

If L is regular, then every "long" string in L is pumpable.

To show that L is not regular, we find one that isn't.

To use the Pumping Theorem to show that a language L is not regular, we must:

1. Choose a string w where $|w| \geq k$. Since we do not know what k is, we must state w in terms of k.
2. Divide the possibilities for y into a set of equivalence classes that can be considered together.
3. For each such class of possible y values where $|xy| \leq k$ and $y \neq \varepsilon$:

   Choose a value for q such that xy^qz is not in L.

THE PUMPING THEOREM FOR REGULAR LANGUAGES: EXAMPLES

Bal = {$w \in$ (), ()* :the parenthesis are balanced}

Is it regular?

THE PUMPING THEOREM FOR REGULAR LANGUAGES: EXAMPLES

Bal = {$w \in$ {), (}* :the parenthesis are balanced}

Is it regular?
Why not?
Prove it!
A good string to use is $w = (^k)^k$

USING THE PUMPING THEOREM EFFECTIVELY

- To choose *w*:
  - Choose a *w* that is in the part of *L* that makes it not regular.
  - Choose a *w* that is only barely in *L*.
  - Choose a *w* with as homogeneous as possible an initial region of length at least *k*.
- To choose *q*:
  - Try letting *q* be either 0 or 2.
  - If that doesn't work, analyze *L* to see if there is some other specific value that will work.

THE PUMPING THEOREM FOR REGULAR LANGUAGES: EXAMPLES

PalEven = $\{ww^R : w \in \{a, b\}^*\}$

Is it regular?

THE PUMPING THEOREM FOR REGULAR LANGUAGES: EXAMPLES

PalEven = $\{ww^R : w \in \{a, b\}^*\}$

Is it regular?
Why not?
Prove it!
A good string to use is $w = a^k b^k b^k a^k$

THE PUMPING THEOREM FOR REGULAR LANGUAGES: EXAMPLES

$\{a^n b^m : n \geq m\}$

Is **not** regular

- A good string to use is $w = a^{k+1}b^k$
- A bad one is $w = a^{2k}b^k$

$\{aba^n b^n : n \geq 0\}$

Choose $w = aba^k b^k$

$$\underbrace{a\ b\ a\ a\ a\ a\ a}_{k}\ \underbrace{a\ a\ a}_{k}\ \underbrace{a\ a\ b\ b\ b\ b\ b\ b\ b\ b\ b}_{k}\ b\ b$$

x y z

What are the choices for (x, y):

(ε, a)
(ε, ab)
(ε, aba$^+$)
(a, b)
(a, ba$^+$)
(aba*, a$^+$)

A DIFFERENT APPROACH TO $\mathtt{aba^nb^n}$?

Can we argue as follows?

We already know that $\{\mathtt{a}^n\mathtt{b}^n : n \geq 0\}$ is not regular. So neither is L.

Can we defend this argument by appeal to the fact that the regular languages are closed under concatenation:

L = $\mathtt{ab}$ || $\{\mathtt{a}^n\mathtt{b}^n : n \geq 0\}$

? regular not regular

A DIFFERENT APPROACH TO $\mathtt{aba}^n\mathtt{b}^n$?

$L_1 = \{\mathtt{aba}^n\mathtt{b}^n : n \geq 0\}$

Consider:

$L_2 = \{\mathtt{b}^n\mathtt{a}^n\mathtt{ba} : n \geq 0\}$

What is L_1 L_2? How can we use this fact to proof L_1 and L_2 are not regular?

USING THE CLOSURE PROPERTIES

The two most useful ones are closure under:

- Intersection
- Complement

USING THE CLOSURE PROPERTIES

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w)\}$

If L were regular, then:

$L' = L \cap$ _____

would also be regular. But it isn't.

USING THE CLOSURE PROPERTIES

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w)\}$

If L were regular, then:

$$\{\mathtt{a}^n\mathtt{b}^n : n \geq 0\} = L \cap \mathtt{a}^*\mathtt{b}^*$$

would also be regular. But it isn't.

EXPLOITING PROBLEM-SPECIFIC KNOWLEDGE

$L = \{w \in \{0, 1, 2, 3, 4, 5, 6, 7\}^*$: w is the octal representation of a nonnegative integer that is divisible by 7$\}$

The first strings are 0,7,16, 25, 34, 43, 52 and 61

Is L regular?

EXPLOITING PROBLEM-SPECIFIC KNOWLEDGE

$L = \{w \in \{0, 1, 2, 3, 4, 5, 6, 7\}^*$: w is the octal representation of a nonnegative integer that is divisible by 7$\}$

The first strings are 0,7,16, 25, 34, 43, 52 and 61

Is L regular? YES!

A 7 state FSM can accept L. Trick: use the fact that $w \in L$ if and only if the sum of its digits is divisible by 7. So the states correspond to the modulo 7 of the sum of the digits so far.

IS ENGLISH REGULAR?

Is English finite?

IS ENGLISH REGULAR?

In the event that the Purchaser defaults in the payment of any installment of purchase price, taxes, insurance, interest, or the annual charge described elsewhere herein, or shall default in the performance of any other obligations set forth in this Contract, the Seller may: at his option: (a) Declare immediately due and payable the entire unpaid balance of purchase price, with accrued interest, taxes, and annual charge, and demand full payment thereof, and enforce conveyance of the land by termination of the contract or according to the terms hereof, in which case the Purchaser shall also be liable to the Seller for reasonable attorney's fees for services rendered by any attorney on behalf of the Seller, or (b) sell said land and premises or any part thereof at public auction, in such manner, at such time and place, upon such terms and conditions, and upon such public notice as the Seller may deem best for the interest of all concerned, consisting of advertisement in a newspaper of general circulation in the county or city in which the security property is located at least once a week for Three (3) successive weeks or for such period as applicable law may require and, in case of default of any purchaser, to re-sell with such postponement of sale or resale and upon such public notice thereof as the Seller may determine, and upon compliance by the Purchaser with the terms of sale, and upon judicial approval as may be required by law, convey said land and premises in fee simple to and at the cost of the Purchaser, who shall not be liable to see to the application of the purchase money; and from the proceeds of the sale: First to pay all proper costs and charges, including but not limited to court costs, advertising expenses, auctioneer's allowance, the expenses, if any required to correct any irregularity in the title, premium for Seller's bond, auditor's fee, attorney's fee, and all other expenses of sale occurred in and about the protection and execution of this contract, and all moneys advanced for taxes, assessments, insurance, and with interest thereon as provided herein, and all taxes due upon said land and premises at time of sale, and to retain as compensation a commission of five percent (5%) on the amount of said sale or sales; SECOND, to pay the whole amount then remaining unpaid of the principal of said contract, and interest thereon to date of payment, whether the same shall be due or not, it being understood and agreed that upon such sale before maturity of the contract the balance thereof shall be immediately due and payable; THIRD, to pay liens of record against the security property according to their priority of lien and to the extent that funds remaining in the hands of the Seller are available; and LAST, to pay the remainder of said proceeds, if any, to the vendor, his heirs, personal representatives, successors or assigns upon the delivery and surrender to the vendee of possession of the land and premises, less costs and excess of obtaining possession.

IS ENGLISH REGULAR?

The rat ran.
The rat that the cat saw ran.
The rat that the cat that the dog chased saw ran.

Let:
A = {cat, rat, dog, bird, bug, pony}
V = {ran, saw, chased, flew, sang, frolicked}.

Let L = English $\cap$ {The A (that the A)* V* V}.

L = {The A (that the A)^n V^n V, $n \geq 0$}.

Let w = The cat (that the rat)^k saw^k ran.

References

- **Automata, Computability and Complexity. Theory and Applications**
  - By Elaine Rich
- Chapter 8:
  - Page : 162-181.

REGULAR LANGUAGES
Examples

- $L_1 = \{w \in \{0 - 9\}^* : w$ is the social security number of the current US president$\}$.
- $L_2 = \{1$ if Santa Claus exists and 0 otherwise$\}$
- $L_3 = \{1$ if there were people in North America more than 10,000 years ago and 0 otherwise$\}$
- $L_4 = \{1$ if there were people in North America more than 15,000 years ago and 0 otherwise$\}$
- $L_5 = \{w \in \{0 - 9\}^+ : w$ is the decimal representation, no leading 0's, of a prime Fermat number$\}$

REGULAR LANGUAGES
Prime Fermat Numbers

$\Sigma = \{0 - 9\}$

$L = \{w \in \{0 - 9\}^+ : w$ is the decimal representation, no leading 0's, of a prime Fermat number$\}$

The Fermat numbers are defined by

$$F_n = 2^{2^n} + 1, n \geq 0$$

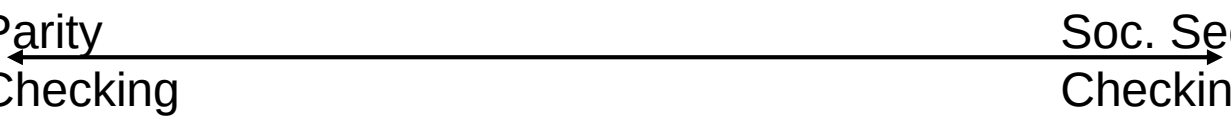
Example elements of L:

$$F_0 = 3, F_1 = 5, F_2 = 17, F_3 = 257, F_4 = 65{,}537$$

Is L regular?

FINITENESS - THEORETICAL VS. PRACTICAL

Any finite language is regular. The size of the language doesn't matter.



But, from an implementation point of view, it very well may.

When is an FSM a good way to encode the facts about a language?

FSM's are good at looking for repeating patterns. They don't bring much to the table when the language is just a set of unrelated strings.

PUMPING THEOREM REVISITED

$L = \{a^i b^j : i, j \geq 0 \text{ and } i \neq j\}$

Try to use the Pumping Theorem by letting $w = a^{k+1}b^k$:

Then $y = a^p$ for some nonzero p.

For $p=2$ we can pump the string and $a^{k+1+2(q-1)}b^k$ *is in the language for all non-negative values of q*.

PUMPING THEOREM REVISITED

Try to use the Pumping Theorem by letting $w = \mathtt{a}^k\mathtt{b}^{k+k!}$.

Then $y = \mathtt{a}^p$ for some nonzero p.

Let $q = (k!/p) + 1$ (i.e., pump in $(k!/p)$ times).

Note that $(k!/p)$ must be an integer because $p < k$.

The number of a's in the new string is $k + (k!/p)p = k + k!$.

So the new string is $\mathtt{a}^{k+k!}\mathtt{b}^{k+k!}$, which has equal numbers of a's and b's and so is not in L.

USING THE CLOSURE PROPERTIES

An easier way:

If L is regular then so is $\neg L$. Is it?

USING THE CLOSURE PROPERTIES

An easier way:

If L is regular then so is $\neg L$. But, what is $\neg L$?

$$\neg L = A^n B^n \cup \{\text{out of order}\}$$

If $\neg L$ is regular, then so is $L' = \neg L \cap a^*b^*$

$$= \_\_\_\_\_$$

USING THE CLOSURE PROPERTIES

An easier way:

If L is regular then so is $\neg L$. But, what is $\neg L$?

$$\neg L = A^n B^n \cup \{\text{out of order}\}$$

If $\neg L$ is regular, then so is $L' = \neg L \cap a^*b^*$

$$= A^n B^n$$

$L = \{a^i b^j c^k : i, j, k \geq 0 \text{ and (if } i = 1 \text{ then } j = k)\}$

Every string in *L* of length at least 1 is pumpable:

But is *L* regular?

$L = \{a^i b^j c^k : i, j, k \geq 0 \text{ and (if } i = 1 \text{ then } j = k)\}$

Every string in L of length at least 1 is pumpable:

Rewrite the final condition as: $(i \neq 1)$ or $(j = k)$

$L = \{a^i b^j c^k : i, j, k \geq 0 \text{ and (if } i = 1 \text{ then } j = k)\}$

Every string in L of length at least 1 is pumpable:

- If $i = 0$ then: if $j \neq 0$, let y be `b`; otherwise, let y be `c`. Pump in or out. Then i will still be 0 and thus not equal to 1, so the resulting string is in L.
- If $i = 1$ then: let y be `a`. Pump in or out. Then i will no longer equal 1, so the resulting string is in L.
- If $i = 2$ then: let y be `aa`. Pump in or out. Then i cannot equal 1, so the resulting string is in L.
- If $i > 2$ then: let y = `a`. Pump out once or in any number of times. Then i cannot equal 1, so the resulting string is in L.

$L = \{a^i b^j c^k : i, j, k \geq 0 \text{ and (if } i = 1 \text{ then } j = k)\}$

But the closure theorems help. Suppose we guarantee that $i = 1$. If L is regular, then so is:

$$L' = L \cap ab^*c^*.$$

$$L' = \{ab^j c^k : j, k \geq 0 \text{ and } j = k\}$$

From here we can use the pumping theorem to show that L' is not regular