# PART 1:

## Automata:

**Finite State Machines (Finite Automata)**

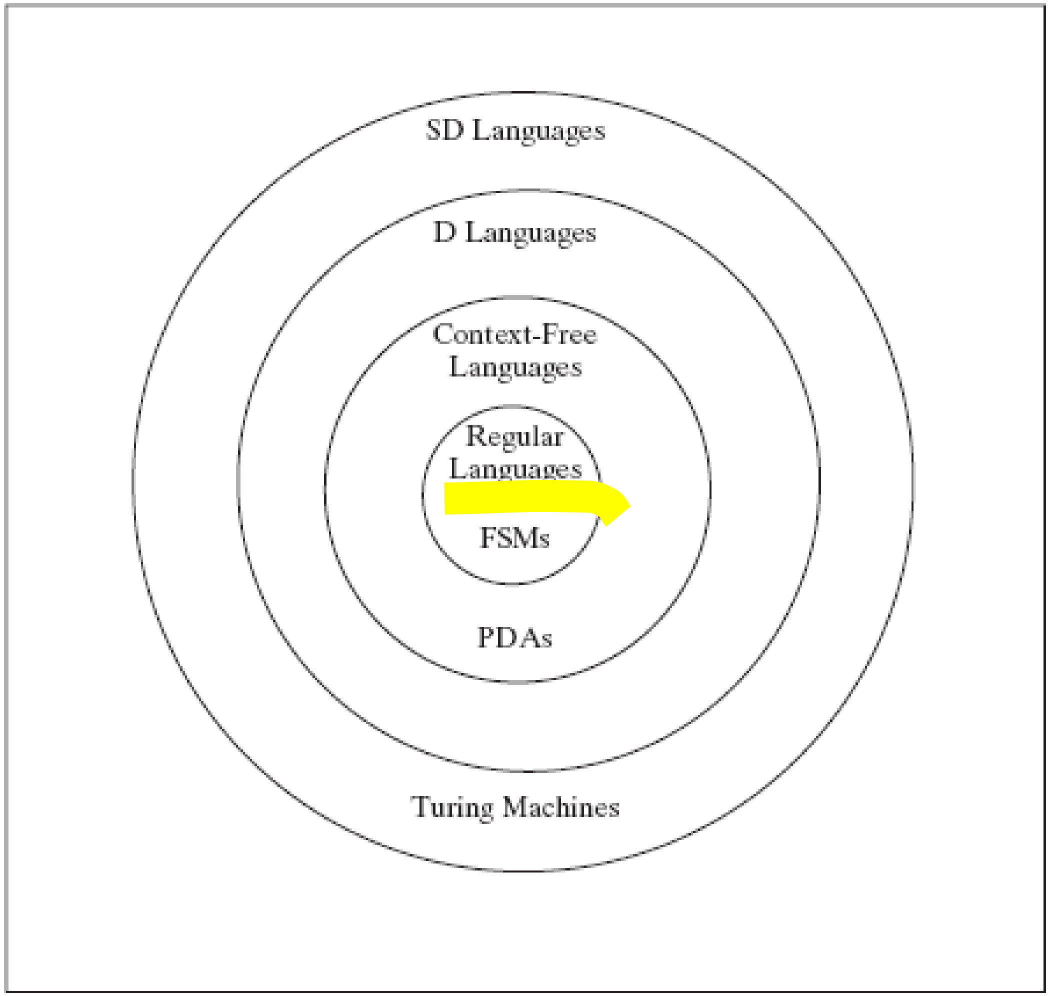
## Formal Language:

**Regular Languages**
**Non-regular Languages**

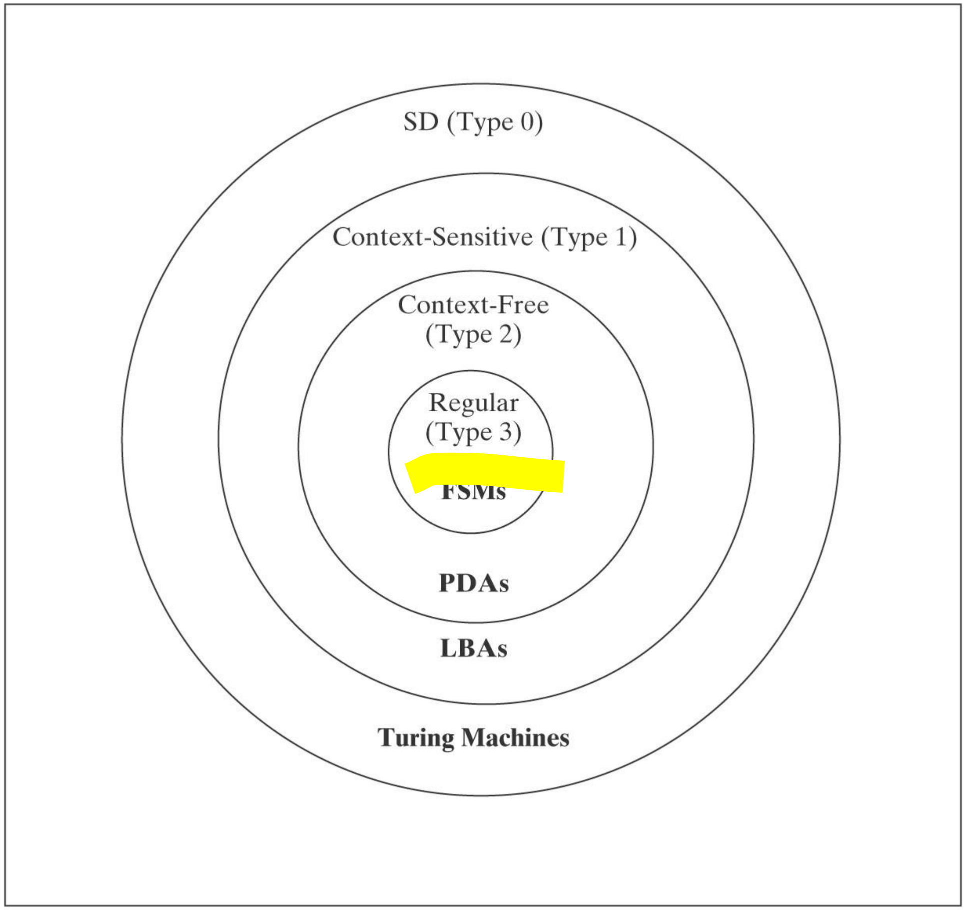
## Grammar:

**Regular Expressions**
**Regular Grammars**

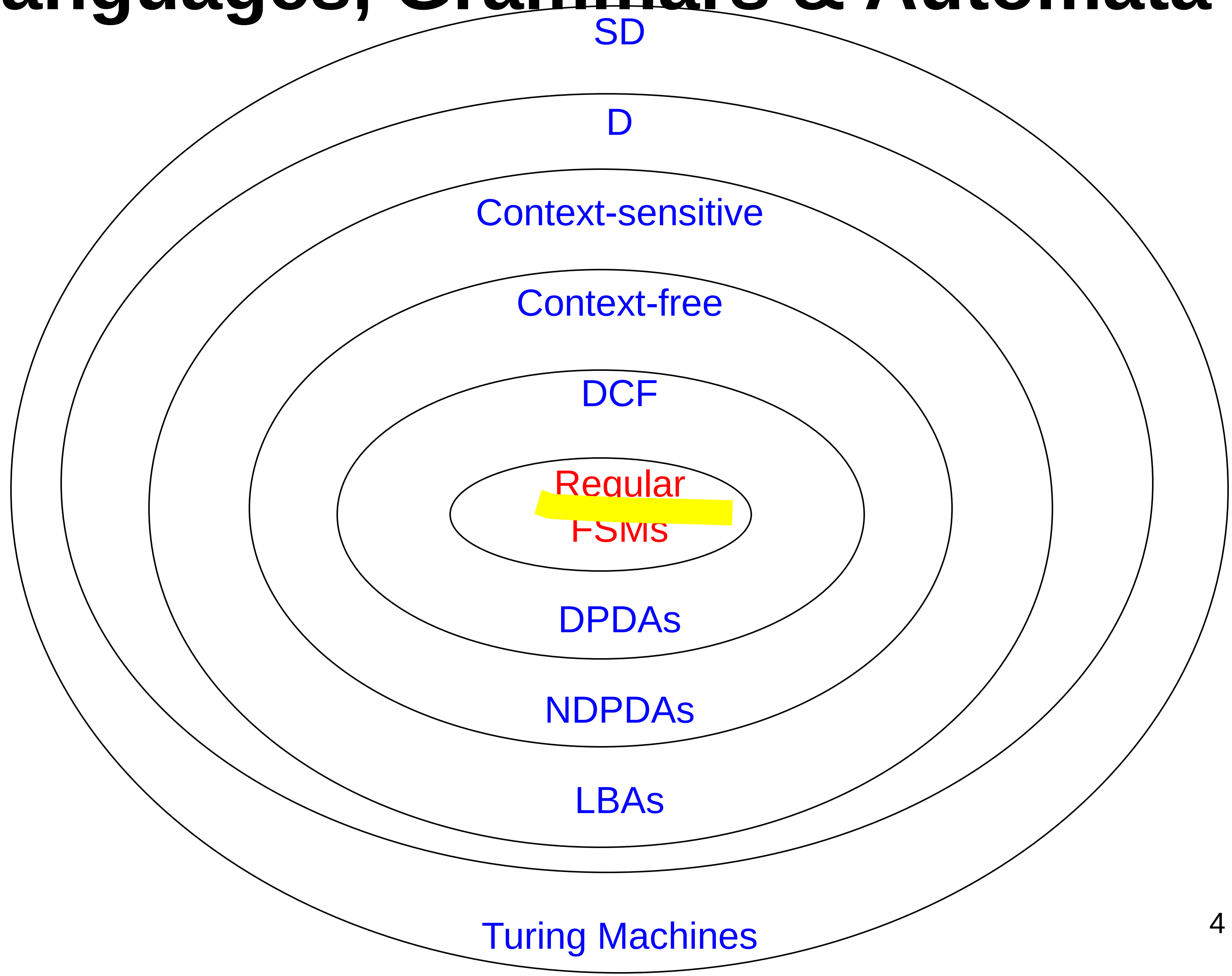
# Languages, Grammars & Automata



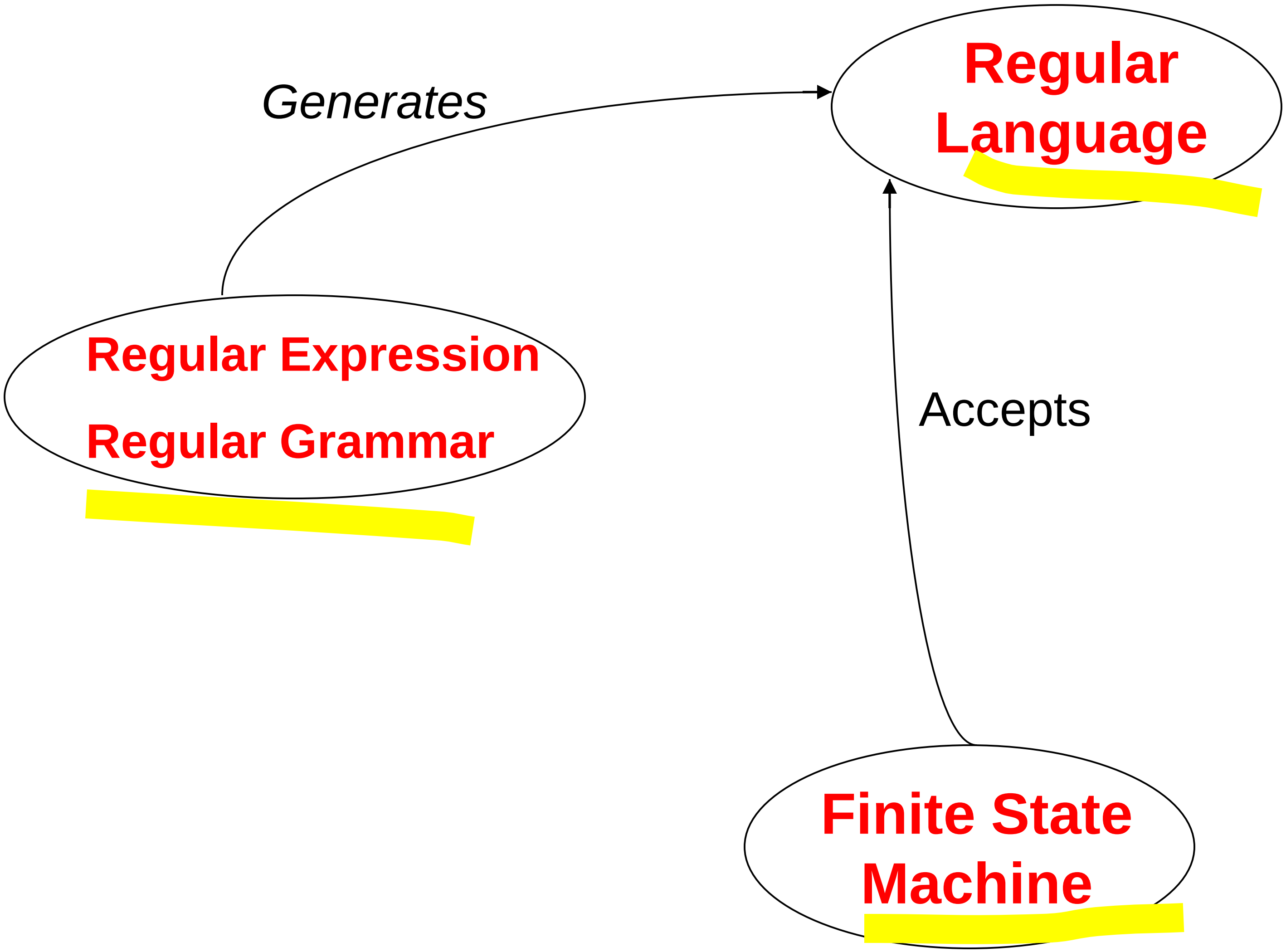
# Languages, Grammars & Automata



# Languages, Grammars & Automata



# Regular Languages



# Closure Properties & Pumping of Regular Languages

# Non-regular Languages

# Languages: Regular or Not?

- L= $a^*b^*$ regular?

- L= $\{a^n b^n : n \geq 0\}$ regular?

- L= $\{w \in \{a, b\}^* :$ every a is immediately followed by b$\}$ regular?

- L= $\{w \in \{a, b\}^* :$ every a has a matching b somewhere and no b matches more than one a $\}$ regular?

# How Many Regular Languages?

***Theorem 8.1:*** There is a countably infinite number of regular languages.

***Proof Idea:***

# There Exist Nonregular Languages

- There is a <u>countably</u> <u>infinite</u> number of **regular languages**.

- There is an <u>uncountably</u> <u>infinite</u> number of **languages** over any nonempty alphabet $\Sigma$.

✓ So there are *many* more non-regular languages than there are regular ones!

# Languages: Regular or Not?

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

# Showing that a Language is Regular

# Showing that a Language is Regular

***Theorem 8.2:*** Every finite language is regular.

***Proof Idea:***

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.

# 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.

# Showing That *L* is Regular

- ✓ Show that *L* is finite.
- ✓ Exhibit an FSM for *L*.
- ✓ Exhibit a regular expression for *L*.
- ✓ Exhibit a regular grammar for *L*.

# Closure Properties of Regular Languages

- Operations that preserve the Property of being a Regular Language!

# Closure Theorems of Regular Languages

The regular languages are closed under

- Union
- Concatenation
- Kleene star

- Complement
- Intersection
- Difference

- Reverse
- Letter Substitution

# Showing That *L* is Regular

- Exploit the closure theorems.

✓ To show that L is regular, show that L can be constructed from other regular languages using the closure operations!

# Example 8.5

L = $\{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ contains an even number of $\mathtt{a}$'s and an odd number of $\mathtt{b}$'s and all $\mathtt{a}$'s come in runs of three$\}$ is **regular**!

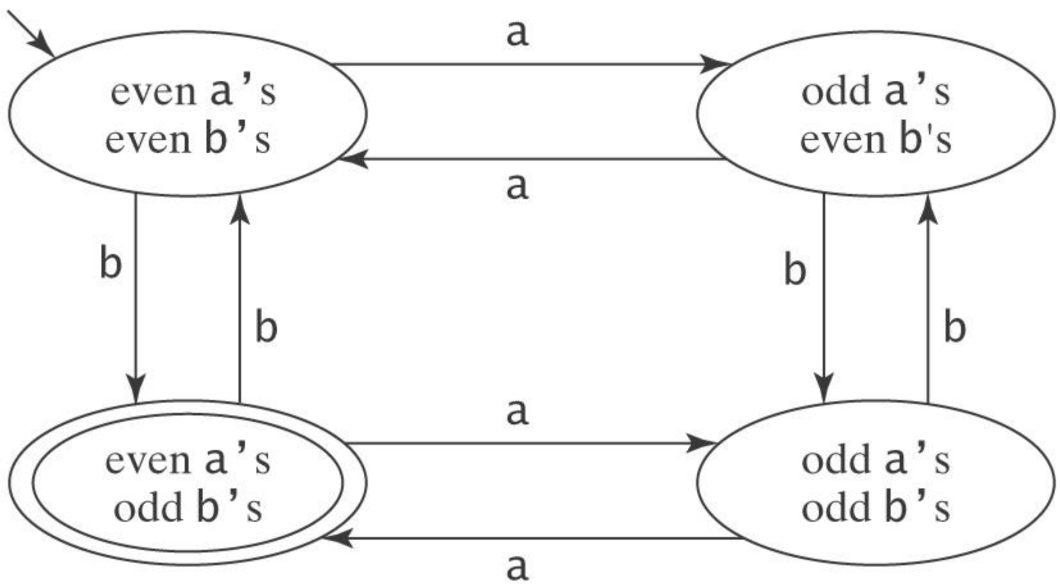
***Proof:***

L = $L_1 \cap L_2$, where:

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

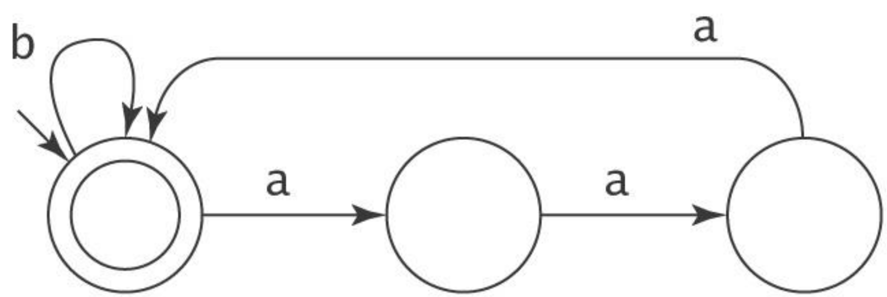
# Example 8.5

$L_1 = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w$ contains an even number of $\mathtt{a}$'s and an odd number of $\mathtt{b}$'s$\}$ is **regular**.



# Example 8.5

$L_2 = \{w \in \{a, b\}^*$ : all a's come in runs of three$\}$ is **regular**.



# Example 8.5

- $L_1$ and $L_2$ are regular.
- Then, $L_1 \cap L_2$ is also regular.
- So, L is regular!

# Showing That *L* is Regular

- ✓ Show that *L* is finite.
- ✓ Exhibit an FSM for *L*.
- ✓ Exhibit a regular expression for *L*.
- ✓ Exhibit a regular grammar for *L*.
- ✓ Exploit the closure theorems.

# Showing that a Language is Not Regular

# Regular Languages

- Every regular language can be accepted by some FSM with a finite # of states.

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

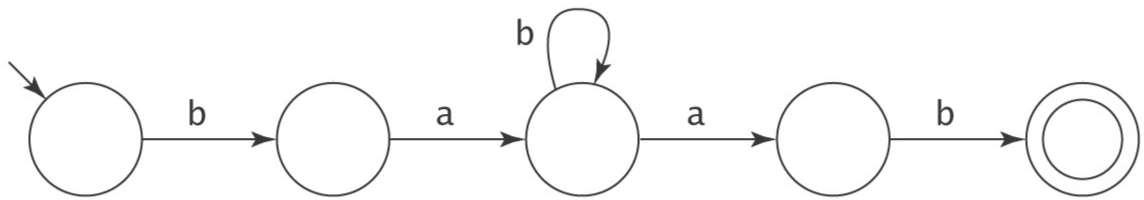
# Regular Languages

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

  ✓ **Kleene star** (in regular expressions), or
  ✓ **cycles** (in FSM).

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

# Repetitive Property of DFSM




L = `bab*ab`

$\underbrace{\texttt{b a}}_{x}\ \underbrace{\texttt{b b b b}}_{y}\ \underbrace{\texttt{a b}}_{z}$

*xy*z* must be in L.

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

# Long Strings Forces Repeated States

***Theorem 8.5*** 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 <u>visit some state more than once</u> (thus traversing at least one loop).

***Proof Idea:***

**By the Pigeonhole Principle!**

*If you put >n pigeons into n holes, then some hole must have > 1 pigeon.*

# Long Strings Forces Repeated States

***Proof Idea:***

- $M$ 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 Pumping Theorem/Lemma for Regular Languages

***Theorem 8.6*** If $L$ is regular, then every long string in $L$ is pumpable.

So, $\exists k \geq 1$

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

$(\exists\, x, y, z\ (w = xyz,$

$\boldsymbol{|xy| \leq k},$

$\boldsymbol{y \neq \varepsilon}$, and

$\boldsymbol{\forall q \geq 0\ (xy^qz \text{ is in } L)})))$.

# Showing That *L* is not Regular

- The pumping theorem is true for every regular language!
- If we could show the pumping theorem is not true of some language L, then L is not regular!

- **Proof by Contraction:**
  1. Suppose some language L is regular, then it would possess certain properties.
  2. Show that L does not posses those properties.
  3. Therefore, L is not regular.

# Example 8.8

L = $\{\mathtt{a}^n\mathtt{b}^n : n \geq 0\}$ is **not regular**!

***Proof:*** Proof by Contradiction.

If L were regular, there exists some k st any string w, $|w| \geq k$, must satisfy the conditions of the pumping theorem.

Choose $w = \mathtt{a}^k\mathtt{b}^k$ Since $|w| \geq k$, $w$ must satisfy the conditions of the pumping theorem.

So, there must exist for some $x$, $y$, and $z$ st $w = xyz$, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall q \geq 0$, $xy^qz$ is in $L$.

We will show that no such $x$, $y$, and $z$ exist!

# Example 8.8

We will show that no such *x*, *y*, and *z* exist!

$$\overbrace{\underbrace{a\ a\ a\ a\ a \ldots a\ a}_{x}\ \underbrace{a\ a}_{y}\ a}^{1}\ \overbrace{b\ b\ b\ b\ \ldots b\ b\ b\ b\ b\ b}^{2}$$

Since $|xy| \le k$, $y$ must be in region 1.
So $y = \mathtt{a}^p$ for some $p \ge 1$.

Let $q = 2$, producing: $\mathtt{a}^{k+p}\mathtt{b}^k$ which $\notin L$, since it has more $\mathtt{a}$'s than $\mathtt{b}$'s.
There exists at least one long string that fails to satisfy the pumping theorem.

So, L is not regular!

# Example 8.9

L = $\{a^n b^n : n \geq 0\}$ is **not regular**!

***Proof:*** Proof by Contradiction.

Choose $w = a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$. Since $|w| \geq k$, $w$ must satisfy the conditions of the pumping theorem. So, for some $x$, $y$, and $z$, $w = xyz$, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall q \geq 0$, $xy^q z$ is in $L$.

We show that no such $x$, $y$, and $z$ exist.
There are 3 cases for where $y$ could occur:

aaaaa.....aaaaaa | bbbbb.....bbbbbb

1 | 2

# Example 8.9

So $y$ can fall in:

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

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

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

There exists one long string in $L$ for which no $x$, $y$, $z$ exist.
So $L$ is not regular!

# Using the Pumping Theorem

- 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 $L$.

# Example 8.10

Bal = {w $\in$ {), (}* : the parens are balanced}
is **not regular**!

***Proof:*** Proof by Contradiction.

Choose $w = (^k)^k$

# Example 8.11

PalEven = $\{ww^R : w \in \{\mathtt{a}, \mathtt{b}\}^*\}$ is **not regular**!

***Proof:*** Proof by Contradiction.

Choose $w = \mathtt{a}^k\mathtt{b}^k\mathtt{b}^k\mathtt{a}^k$

# Example 8.12

L = $\{a^n b^m : n \geq m\}$ is **not regular**!

***Proof:*** Proof by Contradiction.

Choose $w = a^{k+1}b^k$

# 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.

# Using the Closure Properties

The two most useful ones are closure under:

- Intersection $\cap$
- Complement $\neg$

# Example 8.14

L = $\{w \in \{\mathtt{a}, \mathtt{b}\}^*: \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w)\}$ is **not regular**!

***Proof:***

If L were regular, then: Intersection

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

would also be regular. But it isn't.
So, L is not regular.

# Example 8.15

L = $\{a^i b^j : i, j \geq 0 \text{ and } i \neq j\}$ is **not regular**!

***Proof:***

If L is regular then so is Complement

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

If $\neg L$ is regular, then so is Intersection

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

But, $L'$ is not regular.

Then, $\neg L$ is not regular.

So, L is not regular!

# Showing That *L* is not Regular

- ✓ Use the Pumping Theorem for regular languages.
- ✓ Exploit the closure theorems for regular languages.

# Language Summary

**IN**

R grammar
Regular Expression
FSM

**Regular**
a*b*

**OUT**

Pumping
Closure

# Reading Assignment

## Chapter 8:

Sections
8.1
8.2
8.3
8.4

# In-Class Exercises

## Chapter 8:

1 – a & b & n
21 – a & b & e

# Algorithms and Decision Procedures for Regular Languages

# Decision Procedures

- A decision procedure is an algorithm whose result is a boolean value.
- It must be guaranteed to halt on all inputs and be correct

# Membership

***Theorem 9.1*** Given a regular language L and a string w, there exists a decision procedure that answers the question, is $w \in L$?

# Emptiness

***Theorem 9.2*** Given an FSM M, there exists a decision procedure that answers the question, is $L(M) = \varnothing$ ?

# Totality

***Theorem 9.3*** Given an FSM M, there exists a decision procedure that answers the question, is $L(M) = \Sigma^*$?

# Finiteness

***Theorem 9.4*** Given an FSM M, there exists a decision procedure that answers the question, is L(M) finite and is L(M) infinite?

# Equivalence

***Theorem 9.5*** Given two FSMs $M_1$ and $M_2$, there exists a decision procedure that answers the question, is $L(M_1) = L(M_2)$.

# Minimality

***Theorem 9.6*** Given an FSM M, there exists a decision procedure that answers the question, is M minimal?

# Decision Procedures for RLs

Decision procedures that answer questions about languages defined by FSMs:

- Given an FSM $M$ and a string $s$, is $s$ accepted by $M$?
- Given an FSM $M$, is $L(M) = \varnothing$?
- Given an FSM $M$, is $L(M) = \Sigma^*$ ?
- Given an FSM $M$, is $L(M)$ finite (infinite)?
- Given two FSMs, $M_1$ and $M_2$, is $L(M_1) = L(M_2)$?
- Given an FSM $M$, is $M$ minimal?

# Decision Procedures for RLs

Decision procedures that answer questions about languages defined by regular expressions:

- Convert the regular expressions to FSMs and apply the FSM algorithms!

# Decision Procedures for RLs

Decision procedures that answer questions about languages defined by regular grammars:

- Convert the regular grammars to FSMs and apply the FSM algorithms!

# Algorithms for RLs

- *Ndfsmtodfs*
- *MinDFSM*

# Algorithms for RLs

- Given FSMs $M_1$ and $M_2$, construct a FSM $M_3$ st $L(M_3) = L(M_2) \cup L(M_1)$.

- Given FSMs $M_1$ and $M_2$, construct a new FSM $M_3$ st $L(M_3) = L(M_2)\, L(M_1)$.

- Given FSM $M$, construct an FSM $M^*$ st $L(M^*) = (L(M))^*$.

# Algorithms for RLs

- Given a DFSM $M$, construct an FSM $M^*$ st $L(M^*) = \neg L(M)$.

- Given two FSMs $M_1$ and $M_2$, construct an FSM $M_3$ st $L(M_3) = L(M_2) \cap L(M_1)$.

- Given two FSMs $M_1$ and $M_2$, construct an FSM $M_3$ st $L(M_3) = L(M_2) - L(M_1)$.

- Given an FSM $M$, construct an FSM $M^*$ st $L(M^*) = (L(M))^R$.

# Algorithms for RLs

- Given a regular expression $\alpha$, construct an FSM $M$ st $L(\alpha) = L(M)$.

- Given an FSM $M$, construct a regular expression $\alpha$ st $L(\alpha) = L(M)$.

- Given a regular grammar $G$, construct an FSM $M$ st $L(G) = L(M)$.

- Given an FSM $M$, construct a regular grammar $G$ st $L(G) = L(M)$.

# Reading Assignment

**Chapter 9:**

Sections
9.1
9.2

# In-Class Exercises

## Chapter 9:

1 - c

# Regular Languages: Summary

## Regular Languages

- regular exprs.
  or
- regular grammars
- = DFSMs
- recognize
- minimize FSMs

- closed under:
  - ♦ concatenation
  - ♦ union
  - ♦ Kleene star
  - ♦ complement
  - ♦ intersection
- pumping theorem
  D = ND