Regular and Nonregular Languages

Chapter 8

Regular and Non-Regular Languages

Are all finite languages regular?

Regular and Non-Regular Languages

Are all finite languages regular?

Are all infinite languages non-regular?

Regular and Non-Regular Languages

Are all finite languages regular?

Are all infinite languages non-regular?

What must be true about an FSM that accepts an infinite language or a regular expression that generates an infinite language?

Regular and Non-Regular Languages

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

- cycles (in FSM), or
- Kleene star (in regular expressions)

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

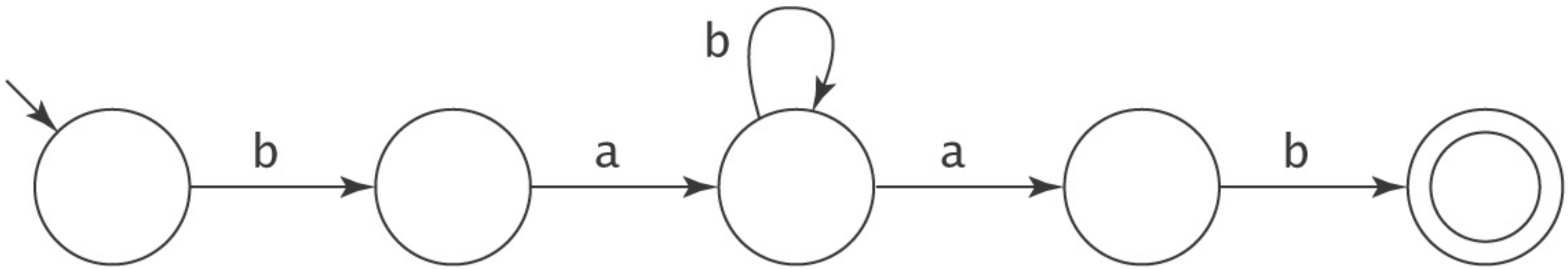
Example 1:

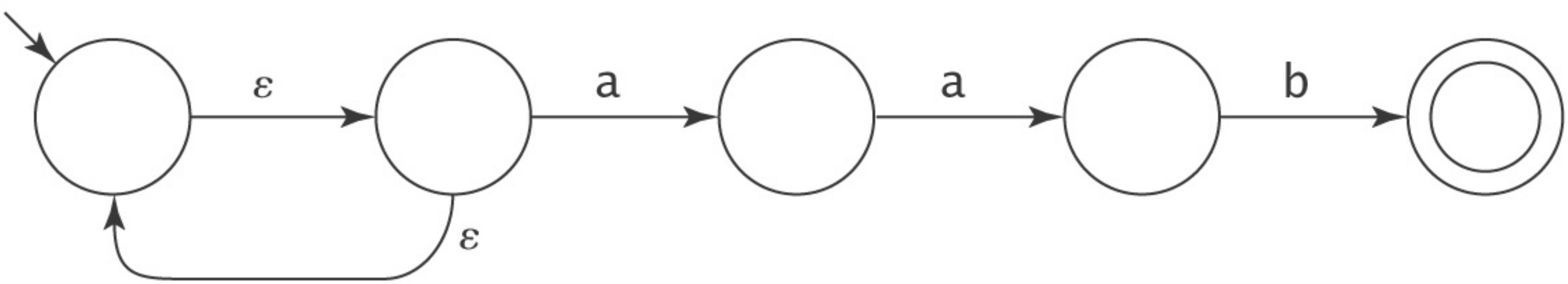
NDFSM with accepting start state and single self loop labeled `a`

Example 2:

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

How Long a String Can Be Accepted?

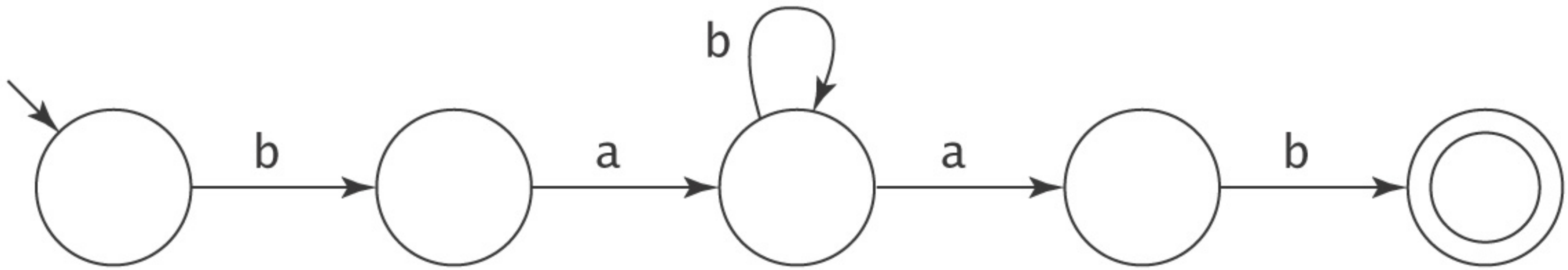




What is the longest string that a 5-state FSM can accept?

How about with no loops?

Exploiting the Repetitive Property



If an FSM with *n* states accepts any string of length ≥ *n*, how many strings does it accept?

L = `bab*ab`

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

xy^*z must be in L.

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

Theorem – Long Strings

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).

Theorem – Long Strings

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 Pumping Theorem for Regular Languages

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

To be precise, if L is regular, then

$\exists k \geq 1$

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

$$
\begin{aligned}
(\exists\, x, y, z\, (w = xyz \wedge\ & \\
& |xy| \leq k \wedge \\
& y \neq \varepsilon \wedge \\
& \forall q \geq 0\ (xy^q z \text{ is in } L)))).
\end{aligned}
$$

The Pumping Theorem for Regular Languages

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

To be precise, if L is regular, then

$\exists k \geq 1,$

$\{\forall\ w \in \Sigma^*,$

$[(\mathrm{w} \in \mathrm{L}\ \wedge |\mathrm{w}| \geq \mathrm{k}) \Rightarrow$

$\exists\ x, y, z \in \Sigma^*,$

$(w = xyz\ \wedge$

$|xy| \leq k\ \wedge$

$y \neq \varepsilon\ \wedge$

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

Showing a Language is Not Regular

If the following is true, then L is not regular:

$\forall k \geq 1,$

$\{\exists\, w \in \Sigma^*,$

$[(w \in L \wedge |w| \geq k) \wedge$

$\forall\, x, y, z \in \Sigma^*,$

$(w = xyz \wedge$

$|xy| \leq k \wedge$

$y \neq \varepsilon) \wedge$

$\exists\, q \geq 0\ (xy^q z \text{ is NOT in } L)]\}.$

No matter what k is (no matter how many states are in the DFSM), we can find a string in L with length at least k that is not "pumpable."

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

Choose w to be $a^k b^k$ (Given k, we get to choose any w.)

$$\overset{1}{\underbrace{a\ a\ a\ a\ a\ \ldots\ a\ a}_{x}\ \underbrace{a\ a}_{y}}\ \Big|\ \overset{2}{\underbrace{a\ b\ b\ b\ b\ \ldots\ b\ b\ b\ b\ b\ b}_{z}}$$

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

$w = xyz$

$|xy| \leq k$,

$y \neq \varepsilon$,

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

Since $|xy| \leq k$, y must be in region 1. So $y = a^p$ for some $p \geq 1$. Let $q = 2$, producing:

$$a^{k+p} b^k$$

which $\notin L$, since it has more a's than b's.

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 possible cases that need to be considered.
3. For each such case where $|xy| \leq k$ and $y \neq \varepsilon$: choose a value for q such that xy^qz is not in L.

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

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

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

Using the Pumping Theorem Effectively

- To choose *w*:
  - Choose a *w* that is in the part of *L* that makes it not regular, e.g. not `aaaaaa` for palindrome.
  - Choose a *w* that is only barely in *L*, i.e. pumping part of it will produce a string not in *L*, e.g. $a^{k+1}b^{k}$ for a^nb^m, n > m
  - Choose a *w* with as homogeneous as possible an initial region of length at least *k*, e.g. a^kb^k for a^nb^n, n >= 0 and Balanced Parens.

    This can mean a string longer than *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
- 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

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

If L were regular, then the complement of L would also be regular. Is it?