Introduction to the Theory of Computation

Set 5 — Nonregular Languages

Nonregular Languages

So far, we have explored several ways to identify regular languages

- DFA, NFA, GNFA, Regular Grammar, RE

There are *many* nonregular languages

- $\{0^n1^n \mid n \geq 0\}$
- {101,101001,1010010001,…}
- {w | w has the same number of 0s and 1s}

How can we tell if a language is not regular?

Property of Regular Languages

All regular languages can be generated by finite automata

States must be reused if
the length of a string
is greater than
the number of states

If states are reused, there will be repetition

Proof Idea

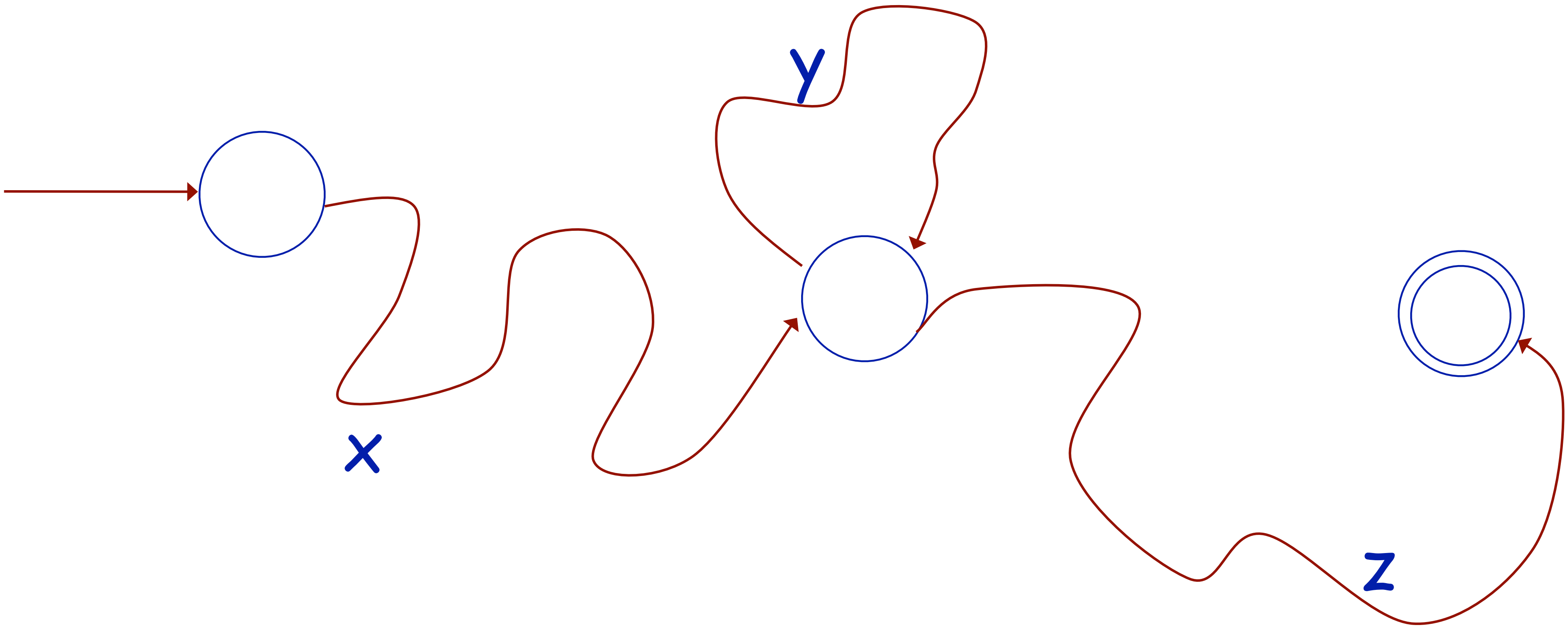
Consider a string in the language whose length is greater than the number of states in the DFA that recognizes the language.

$p = |Q|$ the number of states in the DFA

If the DFA accepts a string s with $|s| \geq p$, then some state must be entered twice while processing s

"Pigeonhole" principle

Proof Idea



s = xyz

x, y, and z are substrings of s

|s| ≥ |Q|

"Pigeonhole" principle

The Pumping Lemma

Theorem:

If A is a regular language,
then there is a number **p** where,
if ***s*** is any string in A of length at least **p**,
then ***s*** may be divided into three pieces,
$s = xyz$, satisfying the following conditions

1. for each $i \geq 0$, xy^iz is in A
2. $|y| > 0$, and
3. $|xy| \leq p$

p is called the **pumping length** of the language

Proof Idea

Consider a pumping length equal to the number of states in the DFA whose language is A

pumping length = $|Q|$

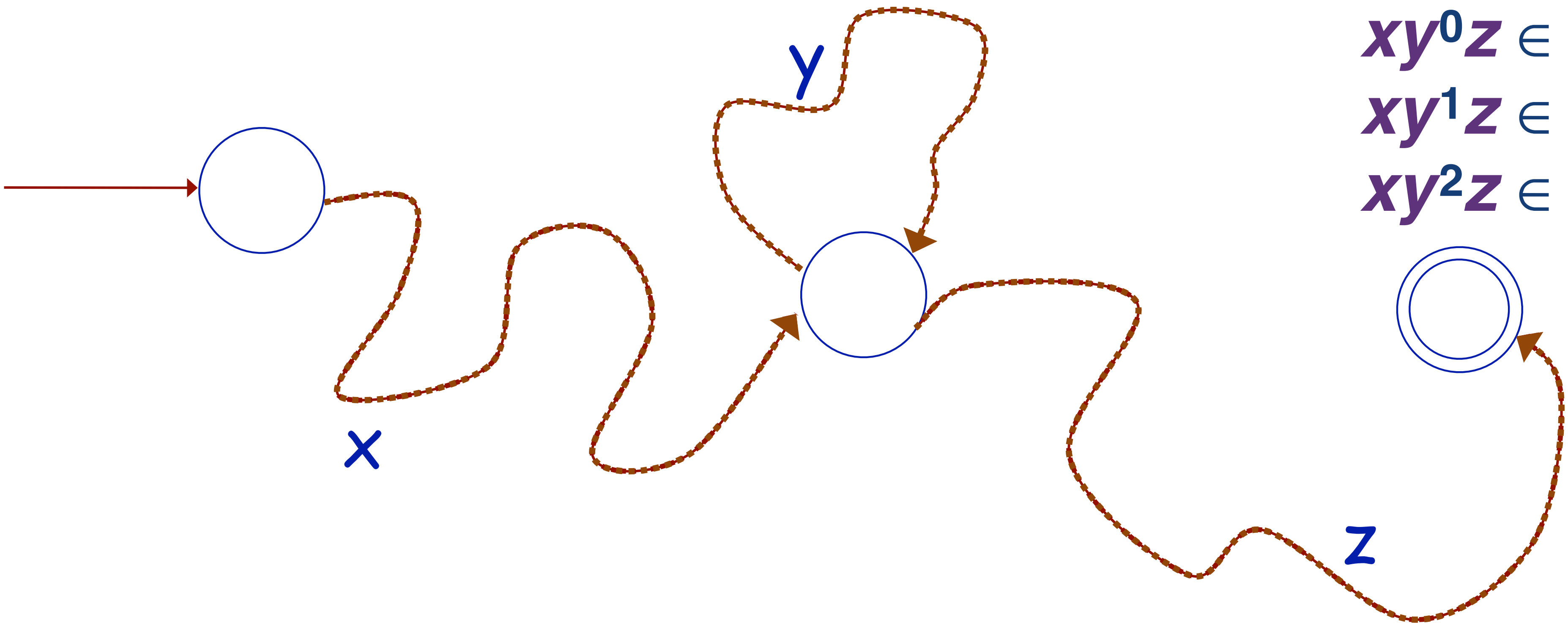
If A accepts a string *s* with $|s| \geq |Q|$, then some state must be entered twice while processing *s*

“Pigeonhole” principle

$s = xyz$

1. for each $i \geq 0$, xy^iz is in A
2. $|y| > 0$, and
3. $|xy| \leq p$

Proof Idea



$s = xyz$

$xy^0z \in A$

$xy^1z \in A$

$xy^2z \in A$

1. for each $i \geq 0$, xy^iz is in A
2. $|y| > 0$, and
3. $|xy| \leq p$

Using the Pumping Lemma

We can use the pumping lemma to prove a language B is not regular

Proof by contradiction

- **Assume B is regular with pumping length p**
- **Find a string *w* $\in$ B with $|w| \geq p$**
- **Show that the pumping lemma is *not* satisfied**
  - **Show that any *xyz* cannot satisfy all of the properties of the pumping lemma (*cannot* be pumped)**
  - **You can choose a specific *w*,
but you cannot choose a specific *xyz!***

Example

$B=\{sbbs \mid s \in \{a,b\}^*\}$

Assume B is regular and p is the pumping length of B

Consider the string $w = a^p bba^p$

$w \in B$ and $|w| \geq p$ so the pumping lemma applies

$w = xyz$, $|xy| \leq p$, $|y| > 0$, $xy^iz \in B \ \forall i$

Example

Consider the string $w = a^p bba^p$

$w \in B$ and $|w| \geq p$ so pumping lemma applies

$w = xyz$, $|xy| \leq p$, $|y| > 0$, $xy^iz \in B\ \forall i$

Since $|xy| \leq p$ and w begins with *xy*,
$xy = a^k$ for some $k \leq p$

- $y = a^j$ for some $j = 1, 2, \ldots, k$

Is $xy^iz \in B$ when $i=2$?

$xy^2z = a^{p+j}bba^p \notin B$

- Pumping lemma is contradicted, so B is *not regular*

Proof of Pumping Lemma

Let A be any regular language

Find DFA $M=(Q,\Sigma,\delta,q_0,F)$ with $L(M)=A$

Let $p=|Q|$

Let $s=s_1s_2s_3\ldots s_n$ be any string in A with

$|s| = n \geq p$

Proof of Pumping Lemma (cont'd)

$p=|Q|$ $s=s_1s_2s_3\ldots s_n \in A$ $|s| = n \geq p$

Let $r_1, r_2, r_3, \ldots, r_{n+1}$ be the sequence of states entered while processing s

$r_1=q_0$

$r_{n+1}\in F$

$r_{i+1}=\delta(r_i, s_i)$

Proof of Pumping Lemma (cont'd)

$p=|Q|$ $\quad$ $s=s_1s_2s_3\ldots s_n \in A$ $\quad$ $|s| = n \geq p$

$r_1, r_2, r_3, \ldots, r_{n+1}$ $\quad$ $r_1=q_0$ $\quad$ $r_{n+1} \in F$ $\quad$ $r_{i+1}=\delta(r_i, s_i)$

Consider the first p+1 elements of this sequence

- p+1 states must contain a repeated state

Let r_k be the first state to be repeated and let r_t be the second occurrence of this state

- $t \leq p+1$

Proof of Pumping Lemma (cont'd)

$t \le p+1$

$p = |Q|$ $s = s_1 s_2 s_3 \ldots s_n \in A$ $|s| = n \ge p$

$r_1, r_2, r_3, \ldots, r_{n+1}$ $r_1 = q_0$ $r_{n+1} \in F$ $r_{i+1} = \delta(r_i, s_i)$

Let $x = s_1 s_2 \ldots s_{k-1}$

$y = s_k s_{k+1} \ldots s_{t-1}$

$z = s_t s_{t+1} \ldots s_n$

- x takes M from r_1 to r_k

  If $k = 1$, then $x = \varepsilon$

- y takes M from r_k to r_t
- z takes M from r_t to r_{n+1}, which is an accept state

Since r_k and r_t are the *same* state, M must accept xy^iz for any $i = 0, 1, 2, \ldots$

Proof of Pumping Lemma (cont'd)

Have we satisfied the conditions of the theorem?

1. for each $i \geq 0$, xy^iz is in A
   - Yes
2. $|y| > 0$
   - Yes, since $t > k$ and $y = s_k s_{k+1} \ldots s_{t-1}$
3. $|xy| \leq p$
   - Yes, since $t \leq p+1$ and $xy = s_1 s_2 \ldots s_{t-1}$

Regular Languages — Summary

Let R be any language.
The following are equivalent:

1. R is a regular language
2. R = L(M) for some finite automaton M, where M is a DFA, an NFA, or a GNFA
3. R is described by some regular grammar
4. R is described by some regular expression

If R can be shown *not* to have a finite pumping length, then R is *not regular*.

Pumping Lemma Example Use

B={🌧️^{n}👟^{n} | $n \geq 0$}

Assume B is regular. Let p be the pumping length given by the lemma.

Consider the string *s* = 🌧️^{p}👟^{p}

$s \in B$ and $|s| \geq p$ so the pumping lemma guarantees that *s* can be split into three pieces

$s = xyz$, $|xy| \leq p$, $|y| > 0$, $xy^iz \in B \ \forall i$

Pumping Lemma Example

B={🌧^{n}👟^{n} | n≥0}

s = 🌧^{p}👟^{p}

$s = xyz$

|xy| ≤ p

Cases to consider

- The string *y* consists only of 🌧's.
- ~~The string *y* consists only of 👟's.~~
- ~~The string *y* consists of both 🌧's and 👟~~'s.

y consists only of 🌧's:

xyyz has more 🌧's than 👟's, so *xyyz* ∉ B

y consists only of 👟's:

xyyz has more 👟's than 🌧's, so *xyyz* ∉ B

y consists of 🌧's and 👟's:

xyyz may have same number of 🌧's and 👟's, but some 👟's will come before some 🌧's, so *xyyz* ∉ B

Minimum Pumping Length

See Sipser Problem 1.55

- The Pumping Lemma for Regular Languages states that
  - every $L \in RL$ has a pumping length p, such that
  - every string $\in L$ of length p or more can be pumped.
- If p is a pumping length for L, so is any length $\geq$ p
- The **minimum pumping length** for L is the ***smallest*** p

- The Pumping Lemma for Regular Languages states that
  - every $L \in RL$ has a pumping length p, such that
  - every string $\in$ L of length p or more can be pumped.
- If p is a pumping length for L, so is any length $\geq$ p
- The **minimum pumping length** for L is the ***smallest*** p

Consider language $B = a^*b^*$

$s = \lambda$, $s \in B$, $|s| = 0$

but s cannot be pumped

(minimum pumping length of B is 1)

Consider language $A = aab^*$

$s1 = aa$, $s1 \in A$, $|s1| = 2$

but s1 cannot be pumped

(minimum pumping length of A is 3)

Unless there is a proof for the minimum pumping length of a language, can only depend on the *existence* of such a value, *not* a specific value itself.

- The Pumping Lemma for Regular Languages states that
  - every L ∈ RL has a pumping length p, such that
  - every string ∈ L of length p or more can be pumped.
- If p is a pumping length for L, so is any length ≥ p
- The **minimum pumping length** for L is the ***smallest*** p

If a language is *not* regular, there is no such guaranteed pumping length.

Using proof by contradiction to show that a language is not regular involves demonstrating that there is no such pumping length.

A variable represents the pumping length and the proof demonstrates a string of that length or greater which does not pump.

- The Pumping Lemma for Regular Languages states that
  - every $L \in RL$ has a pumping length p, such that
  - every string $\in L$ of length p or more can be pumped.
- If p is a pumping length for L, so is any length $\geq p$
- The **minimum pumping length** for L is the ***smallest*** p

- The Pumping Lemma for CF Languages states that
  - every L ∈ CFL has a pumping length p, such that
  - every string ∈ L of length p or more can be pumped.
- If p is a pumping length for L, so is any length ≥ p
- The **minimum pumping length** for L is the ***smallest*** p

If a language is *not* context-free, there is no such guaranteed pumping length.

Using proof by contradiction to show that a language is not context-free involves demonstrating that there is no such pumping length.

A variable represents the pumping length and the proof demonstrates a string of that length or greater which does not pump.