Pumping Lemma
(Regular Languages)

So far

We have studied regular languages,
and the equivalent regular expressions, NFA, DFA

Are there any languages that are not regular?
Or is there a limitation of finite automata?

Consider $L = \{\alpha^n \beta^n \mid n = 0, 1, \ldots\}$
Is it possible to design a finite automaton to recognize L?

Need a math proof. Our intuition can be misleading:
$L = \{(\alpha\beta)^n \mid n = 0, 1, \ldots\}$ is regular

Aside: logic

Propositional logic:
negation $\neg$, conjunction $\wedge$, disjunction $\vee$
De Morgan's Law

truth table:

A	B	$\neg A$	$\neg B$	$A \wedge B$	$A \vee B$	$(\neg A) \vee (\neg B)$
0	0	1	1	0	0	1
1	0	0	1	0	1	1
0	1	1	0	0	1	1
1	1	0	0	1	1	0

"not" "and" "or"

$\neg(A \wedge B) = (\neg A) \vee (\neg B)$

$(A \wedge B) = \neg[(\neg A) \vee (\neg B)]$

$\neg(A \vee B) = (\neg A) \wedge (\neg B)$

Aside: logic

Propositional logic:

implication $\Rightarrow$

"if A, then B"

truth table:

A	B	$A \Rightarrow B$	$\neg A$	$(\neg A) \vee B$
0	0	1	1	1
1	0	0	0	0
0	1	1	1	1
1	1	1	0	1

equivalence $\Leftrightarrow$ $(\Rightarrow) \wedge (\Leftarrow)$

Aside: logic

First-order logic:
universal quantifier $\forall$, exitential quantifier $\exists$

e.g. $(\exists N > 0)(\forall n \geq 1)|a_n| \leq N$

1) Each variable n, N must be quantified by $\forall$ or $\exists$

2) Order of $\forall$, $\exists$ is crucial

$$(\forall n \geq 1)(\exists N > 0)|a_n| \leq N$$

3) Negation of a first-order logical statement

$$\neg(\exists N > 0)(\forall n \geq 1)|a_n| \leq N$$

$(\forall N > 0)(\exists n \geq 1)\ \neg(|a_n| \leq N)$

Pumping lemma

If L is a regular language, then there is a number
p (pumping length) s.t. $(\forall s \in L)|s| \geq p \Rightarrow (s = xyz) \wedge$
$((\forall i \geq 0) xy^i z \in L) \wedge$
$(|y| > 0) \wedge$
$(|xy| \leq p)$

y^i: $y^0 = \epsilon$, $y^1 = y$, $y^2 = yy, \ldots$ (this is why "pumping")

$|y| > 0$: $y \neq \epsilon$

Pumping lemma is a *necessary* condition for regular languages

(Proof of the pumping lemma: Sipser's book p.78)

Pumping lemma

Now we use pumping lemma to show
$L = \{\alpha^n \beta^n \mid n = 0, 1, \ldots\}$ is not a regular language

Assume on the contrary L *is* regular.
Then by pumping lemma, there is a pumping length p s.t.

$$(\forall s \in L)|s| \geq p \Rightarrow (s = xyz) \wedge$$
$$((\forall i \geq 0) xy^i z \in L) \wedge \quad \cdots \text{ condition 1)}$$
$$(|y| > 0) \wedge \quad \cdots \text{ condition 2)}$$
$$(|xy| \leq p) \quad \cdots \text{ condition 3)}$$

Consider the string $s = \alpha^p \beta^p$.
Since $s \in L$ and $|s| > p$, s can be split into x, y, z
satisfying the three conditions

We consider three cases to show this is impossible.

Pumping lemma

Case 1: y contains only α.
Then string $xyyz$ has more α than β,
and $xy^2z \notin L$. This violates condition 1).

Case 2: y contains only β.
Then string $xyyz$ has more β than α,
and $xy^2z \notin L$. This violates condition 1).

Case 3: y contains both α and β.
Then string $xyyz$ has β before α,
and $xy^2z \notin L$. This violates condition 1).

Conclusion: $L = \{\alpha^n \beta^n \mid n = 0, 1, \ldots\}$ is not regular

Pumping lemma

Next we use pumping lemma to show
$L = \{s | \#\alpha(s) = \#\beta(s)\}$ is not a regular language

Assume on the contrary L *is* regular.
Then by pumping lemma, there is a pumping length p s.t.

$$(\forall s \in L)|s| \geq p \Rightarrow (s = xyz) \wedge$$
$$((\forall i \geq 0) xy^i z \in L) \wedge \quad \cdots \text{ condition 1)}$$
$$(|y| > 0) \wedge \quad \cdots \text{ condition 2)}$$
$$(|xy| \leq p) \quad \cdots \text{ condition 3)}$$

Consider the string $s = \alpha^p \beta^p$.
Since $s \in L$ and $|s| > p$, s can be split into x, y, z
satisfying the three conditions

Pumping lemma

Due to $|xy| \leq p$ (condition 3)), y can contain only α.
Then string $xyyz$ has more α than β,
and $xy^2z \notin L$. This violates condition 1).

Conclusion: $L = \{s | \#\alpha(s) = \#\beta(s)\}$ is not regular

Context-Free Languages

Context-free grammar

We introduce a more powerful way that
can describe a more general class of languages

e.g. a *context-free grammar* (CFG) G:

$$A \longrightarrow 0A1$$
$$A \longrightarrow B$$
$$B \longrightarrow \epsilon$$

(or $A \longrightarrow 0A1 \mid B$)

1) each line is a substitution rule

2) each substitution rule is symbol $\longrightarrow$ string

3) each symbol on the left is a variable: A, B

4) each string on the right contains variables and terminal symbols: $0, 1, \epsilon$

5) the variable on the left of the 1st substitution rule is the start variable: A

Context-free grammar

e.g. a *context-free grammar* (CFG) G:

$$A \longrightarrow 0A1$$
$$A \longrightarrow B$$
$$B \longrightarrow \epsilon$$

(or $A \longrightarrow 0A1 \mid B$)

A *derivation* is a sequence of substitutions:

e.g. $A \longrightarrow 0A1 \longrightarrow 0B1 \longrightarrow 0\epsilon 1 = 01$

e.g. $A \longrightarrow 0A1 \longrightarrow 00A11 \longrightarrow 00B11 \longrightarrow 0011$

Parse tree:

Context-free grammar

Defn. A context-free grammar is a 4-tuple $G = (\mathcal{V}, \Sigma, \mathcal{R}, S)$

$\mathcal{V}$: finite set of variables
Σ: finite set of terminal symbols
$(\mathcal{V} \cap \Sigma = \emptyset)$
$\mathcal{R}$: finite set of substitution rules of the form
$V \longrightarrow s$ where $V \in \mathcal{V}$ and $s \in (\mathcal{V} \cup \Sigma)^*$
$S \in \mathcal{V}$: start variable

e.g. a *context-free grammar* (CFG) G:

$$A \longrightarrow 0A1$$
$$A \longrightarrow B$$
$$B \longrightarrow \epsilon$$

Context-free grammar

Let $G = (\mathcal{V}, \Sigma, \mathcal{R}, S)$ be a CFG.

Let $V \in \mathcal{V}$, $s, u, v \in (\mathcal{V} \cup \Sigma)^*$, and $(V \longrightarrow s) \in \mathcal{R}$
Then $uVv \longrightarrow usv$ is a *derivation*

If $s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_k$,
say s_0 derives s_k, written $s_0 \longrightarrow^* s_k$

e.g. a *context-free grammar* (CFG) G: $A \longrightarrow 0A1$
$A \longrightarrow B$
$B \longrightarrow \epsilon$

Context-free grammar

Let $G = (\mathcal{V}, \Sigma, \mathcal{R}, S)$ be a CFG.
The language generated by G is $L(G) = \{s \in \Sigma^* \mid S \longrightarrow^* s\}$

e.g. a *context-free grammar* (CFG) G: $A \longrightarrow 0A1$
$A \longrightarrow B$
$B \longrightarrow \epsilon$

$$L(G) = \{0^n 1^n \mid n \geq 0\}$$