Pumping Lemma for CFLs

**CFL Pumping Lemma**: Let $L$ be a CFL. There exists an integer $p$ ("pumping length") for which every $w \in L$ with $|w| \geq p$ can be written as $w = uvxyz$ such that

1. for every $i \geq 0$, $uv^ixy^iz \in L$, and
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

**Proof**: consider a parse tree for a very long string $w \in L$:
CFL Pumping Lemma

- how large should pumping length $p$ be?
- need to ensure other conditions:
  \[ |vy| > 0 \quad |vxy| \leq p \]
- $b = \text{max } \# \text{ symbols on rhs of any production (assume } b \geq 2)\]
- if parse tree has height $\leq h$, then string generated has length $\leq b^h$ (so length $> b^h$ implies height $> h$)

CFL Pumping Lemma

- let $m$ be the $\#$ of nonterminals in the grammar
- to ensure path of length at least $m+2$, require
  \[ |w| \geq p = b^{m+2} \]
- since $|w| > b^{m+1}$, any parse tree for $w$ has height $> m+1$
- let $T$ be the smallest parse tree for $w$
- longest root-leaf path must consist of $\geq m+1$ non-terminals and 1 terminal.

CFL Pumping Lemma

- must be a repeated non-terminal $A$ on long path
- select a repetition among the lowest $m+1$ non-terminals on path.
- pictures show that for every $i \geq 0$, $uv^ixyz \in L$
- is $|vy| > 0$?
  - smallest parse tree $T$ ensures
- is $|vxy| \leq p$?
  - red path has length $\leq m+2$, so $\leq b^{m+2} = p$ leaves

Chomsky Normal Form

- Useful to deal only with CFGs in a simple normal form
- Most common: Chomsky Normal Form (CNF)
- Definition: every production has form
  \[
  A \to BC \quad \text{or} \quad S \to \epsilon \quad \text{or} \quad A \to a
  \]
  where $A$, $B$, $C$ are any non-terminals (and $B$, $C$ are not $S$) and $a$ is any terminal.

Deciding CFLs

- Useful to have an efficient algorithm to decide whether string $x$ is in given CFL
  - e.g. programming language often described by CFG. Determine if string is valid program.
- If CFL recognized by deterministic PDA, just simulate the PDA.
  - but not all CFLs are (homework)...
- Can simulate NPDA, but this takes exponential time in the worst case.
Deciding CFLs

- Convert CFG into Chomsky Normal Form
- Parse tree for string $x$ generated by nonterminal $A$:

```
A

/ \ \\
B   C

x
```

If $A \rightarrow^k x$ ($k > 1$) then there must be a way to split $x$:

$x = yz$

- $A \rightarrow BC$ is a production and
- $B \rightarrow^l y$ and $C \Rightarrow^j z$ for $i, j < k$

Deciding CFLs

- An algorithm:

```
IsGenerated(x, A)
```

if $|x| = 1$, then return YES if $A \rightarrow x$ is a production,
else return NO

for all $n-1$ ways of splitting $x = yz$

for all $\leq m$ productions of form $A \rightarrow BC$

if IsGenerated(y, B) and IsGenerated(z, C),
return YES

return NO

- worst case running time?

Deciding CFLs

```
IsGenerated(x = x_1 x_2 x_3 \ldots x_n, G)
```

for $i = 1$ to $n$

$T[i, i] = \{ A : \text{“} A \rightarrow x_i \text{” is a production in } G \}$

for $k = 1$ to $n - 1$

for $i = 1$ to $n - k$

for $k$ splittings $x[i, i+k] = x[i,i+j]x[i+j+1,i+k]$

$T[i, i+k] = \{ A : \text{“} A \rightarrow BC \text{” is a production in } G \text{ and } B \in T[i,i+j] \text{ and } C \in T[i+j+1,i+k] \}$

output “YES” if $S \in T[1, n]$, else output “NO”

- worst case running time $\exp(n)$

Deciding CFLs

- Idea: avoid recursive calls
  - build table of YES/NO answers to calls to IsGenerated, in order of length of substring
  - example of general algorithmic strategy called dynamic programming
  - notation: $x[i,j]$ = substring of $x$ from $i$ to $j$
  - table: $T(i,j)$ contains

```
\{ A : \text{a nonterminal such that } A \rightarrow^* x[i,j] \}
```

Deterministic PDA

- A NPDA is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where:
  - $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$ is a function called the transition function
- A deterministic PDA has only one option at every step:
  - for every state $q \in Q$, $a \in \Sigma$, and $t \in \Gamma$, exactly 1 element in $\delta(q, a, t)$, or
  - exactly 1 element in $\delta(q, \varepsilon, t)$, and $\delta(q, a, t)$ empty for all $a \in \Sigma$
**Deterministic PDA**

- A technical detail:
  - we will give our deterministic machine the ability to detect end of input string
    - add special symbol $\bullet$ to alphabet
    - require input tape to contain $x\bullet$–

- language recognized by a deterministic PDA is called a **deterministic CFL (DCFL)**

---

**Example deterministic PDA**

$L = \{0^n1^n : n \geq 0\}$

( unpictured transitions go to a "reject" state and stay there)

---

**Deterministic PDA**

**Theorem**: DCFLs are closed under complement

(complement of $L$ in $\Sigma^*$ is $(\Sigma^* - L)$)

Proof attempt:

- swap accept/non-accept states
- problem: might enter infinite loop before reading entire string
- machine for complement must accept in these cases, and read to end of string

---

**Example of problem**

Language of this DPDA is $\{\varepsilon\}$

---

**Deterministic PDA**

**Proof**:

- convert machine into "normal form"
  - always reads to end of input
  - always enters either an accept state or single distinguished "reject" state
- step 1: keep track of when we have read to end of input
- step 2: eliminate infinite loops
Deterministic PDA

step 1: keep track of when we have read to end of input

for accept state q': replace outgoing "ε, ? → ?" transition with self-loop with same label

step 2: eliminate infinite loops

– add new "reject" states

– on input x, if infinite loop, then:

\[ \text{infinite sequence } i_0 < i_1 < i_2 < \ldots \] such that for all k, stack height never decreases below \( h(t_k) \) after time \( t_k \)