Normal Forms for CFGs

- A CFG \( G = (V, \Sigma, P, S) \) is in Chomsky Normal Form (or \( \text{CNF} \)) if all productions in \( P \) are of the form \( A \to BC \) or \( A \to a \), where \( A, B, C \in V \) and \( a \in \Sigma \).

- Preliminary clean-up/simplifications:
  - Eliminate \textit{useless symbols} \( x \in VN \) that do not appear in any derivation of a terminal string from \( S \).
  - Eliminate \textit{\( \varepsilon \)-productions} \( A \to \varepsilon \) for \( A \in V \).
  - Eliminate \textit{unit productions} \( A \to B \) for \( A, B \in V \).

Properties of useful symbols

- A symbol \( X \in VN \) is \textit{generating} if \( X \Rightarrow w \) for \( w \in \Sigma^* \). Note that every \( X \in \Sigma \) is generating since \( w \in \Sigma^* \) can be \( X \) itself.

- A symbol \( X \in VN \) is \textit{reachable} if there is a derivation \( S \Rightarrow \alpha X \beta \) for \( \alpha, \beta \in (V + \Sigma)^* \).
Normal Forms for CFGs

Theorem 7.2

Let \( G=(V,\Sigma,P,S) \) be a CFG, and assume that \( L(G) \neq \emptyset \); i.e. \( G \) generates at least one string. Let \( G_1=(V_1,\Sigma_1,P_1,S) \) be the grammar obtained by the following steps:

A Round #1: Eliminate nongenerating symbols and all productions involving one or more of these symbols. Let \( G_2=(V_2,\Sigma_2,P_2,S) \) be this new grammar.

A Round #2: Eliminate all symbols that are unreachable in the grammar \( G_2 \).

Then \( G_1 \) has no useless symbols, and \( L(G_1)=L(G) \).

Theorem 7.4

The following algorithm finds all and only the generating symbols of grammar \( G=(V,\Sigma,P,S) \):

" BASIS: Every \( a \in \Sigma \) is generating.

" INDUCTION: Suppose \((A \rightarrow \alpha) \in P\), and every symbol of \( \alpha \) is already known to be generating. Then \( A \) is generating. Note that this rule includes the case where \( \alpha = \varepsilon \).

Used for “Round #1” of Theorem 7.2 ...

Example:

Given the grammar \( G=(V,\Sigma,P,S) \) where

\[
P: \quad S \rightarrow AB \mid C \quad B \rightarrow 1 \mid A0 \quad A \rightarrow 0B \mid C \quad C \rightarrow AC \mid C1
\]

Applying “Round #1” (Theorem 7.4):

- Since 0,1 \( \in \Sigma \) are “in”, \( B \rightarrow 1 \) implies \( B \in V \) is “in.”
- So, \( A \rightarrow 0B \) and \( S \rightarrow AB \) imply \( A,S \in V \) are “in.”
- Nothing more can be added, so \( C \in V \) is a nongenerating symbol that can be eliminated along with any production that mentions it.
Normal Forms for CFGs

Example, continued...

- So, define $G_2 = (V_2, \Sigma_2, P_2, S)$ where
  
  \[ P_2: S \rightarrow AB \quad B \rightarrow 1 | A0 \quad A \rightarrow 0B \]

- Applying “Round #2” (Theorem 7.6):
  
  Since $S \in V$ is “in”, then $A, B \in V$ are “in.”
  
  $0, 1 \in \Sigma$ are “in.”
  
  So, all symbols are reachable in $G_2$.

- Letting $G_1 = G_2$, $G_1$ has no useless symbols.

Normal Forms for CFGs

\section*{Theorem 7.7}

- In any grammar $G=(V, \Sigma, P, S)$, the only \textit{nullable symbols} are the variables found by the following algorithm:

  - **Basis:** If $(A \rightarrow \varepsilon) \in P$, then $A$ is nullable.

  - **Induction:** If $(B \rightarrow C_1 C_2 \cdots C_k w) \in P$, where each $C_i \in V$ is nullable, then $B$ is nullable.

  (Hence, we only need to consider productions with all-variable bodies.)

Normal Forms for CFGs

\section*{Theorem 7.9}

- If the grammar $G_1 = (V, \Sigma, P_1, S)$ is constructed from grammar $G=(V, \Sigma, P, S)$ by the construction for \textit{eliminating $\varepsilon$-productions} given below, then $L(G_1) = L(G) - \{\varepsilon\}$.

  - **Round #1:** Determine all nullable symbols of $G$.

  - **Round #2:** For each $(A \rightarrow X_1 X_2 \cdots X_k w) \in P$, $k \geq 1$, suppose $m$ of the $k$ $X_i$‘s are nullable symbols. $G_1$ will have $2^m$ versions of this production, where the nullable $X_i$’s in all possible combinations are either present or absent. If $m=k$, exclude the case when all $X_i$‘s are absent.

Eliminating $\varepsilon$-Productions

- If a language $L$ has a CFG, then $L - \{\varepsilon\}$ has a CFG without $\varepsilon$-productions.

- If $\varepsilon \notin L$, then $L$ itself is $L - \{\varepsilon\}$, so $L$ has a CFG without $\varepsilon$-productions.

- A variable $A \in V$ is \textit{nullable} if $A \Rightarrow \varepsilon$.

- If $A$ is nullable, then whenever $A$ appears in a production body, say $B \rightarrow CAD$, $A$ might (or might not) derive $\varepsilon$. 

April 26, 2002 © B. Juliano, Computer Science @ CSU, Chico
Normal Forms for CFGs

Example

- Given the grammar $G = (V, \Sigma, P, S)$ where
  
  \[ P : \]
  
  \[ S \rightarrow AB \]
  
  \[ A \rightarrow aAA \mid \varepsilon \]
  
  \[ B \rightarrow bBB \mid \varepsilon \]
  
- Applying “Round #1” (Theorem 7.7):
  
  - $A, B \in V$ are nullable from $A \rightarrow \varepsilon$ and $B \rightarrow \varepsilon$.
  - $S \in V$ nullable from $S \rightarrow AB$.
  - So, all variables in $V$ are nullable.

Normal Forms for CFGs, continued ...

- Applying “Round #2” (from Theorem 7.9):
  
  - Productions added from $S \rightarrow AB$:
    
    \[ S \rightarrow AB \mid A \mid B \]
  
  - Productions added from $A \rightarrow aAA$:
    
    \[ A \rightarrow aAA \mid aA \mid a \]
  
  - Productions added from $B \rightarrow bBB$:
    
    \[ B \rightarrow bBB \mid bB \mid b \]
  
  - So, define $G_1 = (V, \Sigma, P_1, S)$ where
    
    \[ P_1 : \]
    
    \[ S \rightarrow AB \mid A \mid B \]
    
    \[ B \rightarrow bBB \mid bB \mid b \]
    
    \[ A \rightarrow aAA \mid aA \mid a \]

Eliminating Unit Productions

- A unit production is a production of the form $A \rightarrow B$ where $A, B \in V$.
- A pair $(A, B)$, where $A \Rightarrow B$ using only unit productions, is called a unit pair.

Theorem 7.11

- The following inductive construction algorithm finds exactly the unit pairs for a CFG $G$:
  
  - **Basis:** $(AA)$ is a unit pair for any variable $A \in V$.
  
  - **Induction:** If $(A, B)$ is a unit pair, and $B \rightarrow C$ is a production where $C \in V$, then $(A, C)$ is a unit pair.
Normal Forms for CFGs

Theorem 7.13

If grammar $G_1 = (V, \Sigma, P_1, S)$ is constructed from grammar $G = (V, \Sigma, P, S)$ by the following algorithm for eliminating unit productions:

- **Round #1:** Find all the unit pairs of $G$.
- **Round #2:** For each unit pair $(A, B)$, add to $P_1$ all the productions $A \rightarrow \alpha$ where $B \rightarrow \alpha$ is a nonunit production in $P$. Note that $A = B$ is possible; in that way, $P_1$ contains all the nonunit productions in $P$.

Then, $L(G_1) = L(G)$.

Normal Forms for CFGs

Chomsky Normal Form

- Every CFL without $\epsilon$ has a grammar $G = (V, \Sigma, P, S)$ in which all productions are in one of two simple forms, either
  - $A \rightarrow BC$, where $A, B, C \in V$; or
  - $A \rightarrow \alpha$, where $A \in V$ and $\alpha \in \Sigma$.
- Further, $G$ has no useless symbols. Such a grammar is said to be in **Chomsky Normal Form**, or CNF.

Normal Forms for CFGs

Cleaning up grammars

1. Eliminate $\epsilon$-productions
2. Eliminate unit productions
3. Eliminate useless symbols

Theorem 7.14

If $G$ is a CFG generating a language that contains at least one string other than $\epsilon$, then there is another CFG $G_1$ such that $L(G_1) = L(G) - \{\epsilon\}$, and $G_1$ has no $\epsilon$-productions, unit productions, or useless symbols.

Procedure for Converting to CNF

1. Clean-up the grammar (via Theorem 7.14)
2. Arrange that all productions with bodies of length 2 or more consist only of variables.
   - For each $a \in \Sigma$, introduce a new variable $A_a$ and a production $A_a \rightarrow a$.
   - Replace $a$ in any body, where it is not the entire body, by $A_a$. 
Normal Forms for CFGs

3. Break productions with bodies of length 3 or more into a cascade of productions, each with a body consisting of two variables.

"For each production $A \rightarrow B_1 B_2 w B_k$, $k \geq 3$, introduce $k-2$ new variables $C_1, C_2, w, C_{k-2}$.

"Replace the original production $A \rightarrow B_1 B_2 w B_k$ by the $k-1$ productions

$A \rightarrow B_1 C_1$

$C_1 \rightarrow B_2 C_2$

$C_{k-3} \rightarrow B_{k-2} C_{k-2}$

$x$

$C_{k-2} \rightarrow B_{k-1} B_k$

Theorem 7.16

If $G = (V, \Sigma, P, S)$ is a CFG whose language contains at least one string other than $\epsilon$, then there is a grammar $G_1$ in Chomsky Normal Form, such that $L(G_1) = L(G) - \{\epsilon\}$. 
**Pumping Lemma for CFLs**

**Theorem 7.17**
- Suppose we have a parse tree $T$ according to a CNF grammar $G=(V, \Sigma, P, S)$, and suppose that the yield of the tree is $w \in \Sigma^*$. If the length of the longest path in $T$ is $n$, then $|w| \leq 2^{n-1}$.

**Theorem 7.18** (*The pumping lemma for CFLs*)
- Let $L$ be a CFL. Then there exists a constant of the pumping lemma, $n$, such that if $z$ is any string in $L$ such that $|z| \geq n$, we can write $z=uvwxy$, subject to the following conditions:
  - $|vx| \leq n$;
  - $vx \neq \epsilon$ – at least one of the strings pumped must not be empty; and
  - for all $i \geq 0$, $uv^iwx^iy \in L$ – the two substrings $v$ and $x$ may be “pumped” any number of times and the resulting string is still in $L$. 

---

*Note: $w = \alpha \beta$, where $\alpha, \beta \in \Sigma^*$, and $|w|$ is the number of leaves in $T$. Every sufficiently long string $w \in L$ must have a long path in its parse tree.*
Chapter 7: Properties of CFLs

Dividing the string $z=uvwxy$ so it can be pumped.

Underlying idea: Initial configuration prior to pumping ...

Result of pumping the (sub)strings $v$ and $x$ zero times.

What's important is not the number of repetitions – the key is good form.
Pumping Lemma for CFLs

Proof outline:
- Let $L$ be a CFL. Let there be a CNF CFG for $L$ with $m$ variables. Pick $n = 2m$.
- Because CNF grammars have bodies of no more than 2 symbols, a string $z$ where $|z| \geq n$ must have some path with at least $m+1$ variables.
- Thus, some variable must appear (at least) twice on the path.
- Compare with the PL for RL using a DFA argument about a path longer than the number of states.

Closure Properties of CFLs

Substitutions
- Let $\Sigma$ be an alphabet, and suppose that for every $a \in \Sigma$, we choose a language $L_a$. These chosen languages can be over any alphabets, not necessarily $\Sigma$ and not necessarily the same.
- This choice of languages defines a function $s$ (a substitution) on $\Sigma$, and we shall refer to $L_a$ as $s(a)$ for each $a \in \Sigma$. 
Closure Properties of CFLs

Substitutions
- If \( w = a_1 a_2 w a_n \in \Sigma^* \), then \( s(w) \) is the language over all strings \( x_1 x_2 w x_n \) such that string \( x_i \) is in the language \( s(a_i) \), for \( 1 \leq i \leq n \).
- In other words, \( s(w) \) is the concatenation of the languages \( s(a_1)s(a_2)w s(a_n) \).
- \( s(L) \) is the union of \( s(w) \) for all \( w \in L \).

Theorem 7.23 (Substitution Theorem)
- If \( L \) is a CFL over alphabet \( \Sigma \), and \( s \) is a substitution on \( \Sigma \) such that \( s(a) \) is a CFL for each \( a \in \Sigma \), then \( s(L) \) is a CFL.

Proof:
- Idea: Take a CFG \( G=(V, \Sigma, P, S) \) for \( L \) and replace each \( a \in \Sigma \) by the start symbol \( S_a \) of a CFG \( G_a=(V_a, T_a, P_a, S_a) \) for the language \( s(a) \).
- Assuming no (variable) symbol \( A \) is in two or more of \( V \) and any of the \( V_a \)'s.

Closure Properties of CFLs

Construct a new grammar \( G'=(V', T', P', S) \) for \( s(L) \) as follows:
- \( V' \) is the union of \( V \) and all the \( V_a \)'s for \( a \in \Sigma \).
- \( T' \) is the union of all the \( T_a \)'s for \( a \in \Sigma \).
- \( P' \) consists of:
  - all productions in any \( P_a \) for \( a \in \Sigma \).
  - the productions of \( P \), but with each \( a \in \Sigma \) replaced by \( S_a \) everywhere \( a \) occurs.

A parse tree in \( G' \) begins with a parse tree in \( G \) and finishes with many parse trees, each one in one of the grammars \( G_a \).
Closure Properties of CFLs

Applications of the Substitution Theorem

Theorem 7.24

CFLs are closed under the following operations:
- Union
- Concatenation
- Closure (*) and Positive Closure (+)
- Homomorphism

Proof:
- **Union:** Let $L_1$ and $L_2$ be CFLs. Then $L_1 \cup L_2$ is the language $s(L)$, where language $L = \{1, 2\}$, and substitution $s$ is defined by $s(1) = L_1$ and $s(2) = L_2$.
- **Concatenation:** Let $L_1$ and $L_2$ be CFLs. Then $L_1 L_2$ is the language $s(L)$, where language $L = \{12\}$, and substitution $s$ is defined by $s(1) = L_1$ and $s(2) = L_2$.

Proof:
- **Closure and Positive Closure:** If $L_1$ is a CFL, $L$ is the language $\{1\}^*$, and substitution $s$ is defined as $s(1) = L_1$, then $L_1^* = s(L)$. Similarly, if $L$ is instead the language $\{1\}^+$, then $L_1^+ = s(L)$.

Proof:
- **Homomorphism:** Let $L$ be CFL over $\Sigma$ and $h$ a homomorphism on $\Sigma$. Let substitution $s$ be defined as $s(a) = \{h(a)\}$ for all $a \in \Sigma$. Then $h(L) = s(L)$.
Closure Properties of CFLs

Theorem 7.25 (Closure under Reversal)

- If \( L \) is a CFL, then so is \( L^R \).

Theorem 7.27 (Intersection with a Regular Language)

- If \( L \) is CFL and \( R \) is RL, the \( L \cap R \) is a CFL.

Formally ...

- Given:
  - PDA \( P = (Q_P, \Sigma, \Gamma, \delta_P, q_0, F_P) \) that accepts \( L \) by final state; and
  - DFA \( A = (Q_A, \Sigma, \delta_A, F_A) \) for \( R \).
- Construct
  - PDA \( P' = (Q_P \times Q_A, \Sigma, \Gamma, \delta, (q, p, q_A), Z_0, F_P \times F_A) \)
  - where \( \delta((q, p), a, X) \) defined to be the set of all pairs \((r, s), (r', \gamma)\) such that:
    - \( s = \Delta_A(p, a) \); and
    - \( (r, \gamma) \in \delta_P(q, a, X) \)

Theorem 7.29

- The following are true about CFLs \( L, L_1, \) and \( L_2 \), and a RL \( R \):
  - \( L - R \) is a CFL.
  - \( \overline{L} \) is not necessarily a CFL.
  - \( L_1 - L_2 \) is not necessarily a CFL.
Chapter 7: Properties of CFLs

Theorem 7.30 (Closure under Inverse Homomorphisms)

Let $L$ be a CFL and $h$ a homomorphism. Then $h^{-1}(L)$ is a CFL.

Proof:

Let CFL $L$ be defined over alphabet $T$, $h: \Sigma \rightarrow T^*$, and PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where $L(P) = L$.

Construct $P' = (Q', \Sigma, \Gamma, \delta', (q_0, \epsilon), Z_0, F \times \{\epsilon\})$ where

- $Q' \subseteq Q \times T^*$ where for every $(q, x) \in Q$, $x \in T^*$ is a suffix (not necessarily proper) of $h(a) \in T^*$ for $a \in \Sigma$.

"$\delta'$ is defined by the following rules:

- $\delta'((q, \epsilon), a, X) = \{(q, h(a), X)\}$ for all $a \in \Sigma$ and $a \not= \epsilon$, $q \in Q$, and $X \in \Gamma$.
- If $(p, \gamma) \in \delta(q, b, X)$, where $b \in T$ or $b = \epsilon$, then $(p, \gamma) \in \delta'(q, bx, \epsilon, X)$.

The start state of $P'$, $(q_0, \epsilon)$, is the start state of $P$ with an empty buffer.

Hence, to show $L(P') = h^{-1}(L(P))$,

$(q_0, h(w), Z_0) \xrightarrow{\delta} (p, \epsilon, \gamma) \iff ((q_0, \epsilon), w, Z_0) \xrightarrow{\delta'} ((p, \epsilon), \epsilon, \gamma)$

Decision Properties of CFLs

Recall the following (linear!) conversion algorithms:

- CFG to PDA conversion (see Theorem 6.13)
- PDA that accepts by final state to PDA that accepts by empty stack (see Theorem 6.11)
- PDA that accepts by empty stack to PDA that accepts by final state (see Theorem 6.9)

What about conversions between PDAs and CFGs?
Decision Properties of CFLs

**Theorem 7.31 (Complexity of Converting among CFGs and PDAs)**

- There is an $O(n^3)$ algorithm that takes a PDA $P$ whose representation has length $n$ and produces a CFG of length at most $O(n^3)$. This CFG generates the same language as $P$ accepts by empty stack.
- Optionally, we can cause $G$ to generate the language that $P$ accepts by final state.

**Theorem 7.32 (Running Time of Conversion to CNF)**

- Given a grammar $G$ of length $n$, we can find an equivalent CNF grammar for $G$ in time $O(n^2)$; the resulting grammar has length $O(n^2)$.
- Primarily due to the subalgorithm for unit pair construction and elimination of all unit productions.

Testing Emptiness of CFLs

CFL $L$ is empty if and only if $S$ of $G = (V, \Sigma, P, S)$ is not generating.

Testing Membership in a CFL


- “dynamic programming” algorithm
- **Inputs:**
  - CNF grammar $G = (V, \Sigma, P, S)$ for CFL $L$
  - $w = a_1a_2w a_n \in \Sigma^*$
- **Output:**
  - Decision, in $O(n^3)$ time, whether $w \in L$. 
Decision Properties of CFLs

Example:
Given the following productions of CNF $G$:

$S \rightarrow AB \mid BC$
$A \rightarrow BA \mid a$
$B \rightarrow CC \mid b$
$C \rightarrow AB \mid a$

Test membership of $baaba$ in $L(G)$.

Idea behind CYK Algorithm …

$S \rightarrow AB$
$S \rightarrow BC$
$A \rightarrow BA\quad baab\quad aaba$
$B \rightarrow CC$
$C \rightarrow AB\quad baa\quad aab\quad aba$
$A \rightarrow a\quad ba\quad aa\quad ab\quad ba$
$B \rightarrow b\quad b\quad a\quad a\quad b\quad a$

Determine the parse tree for $w$, $|w| = n$, by incrementally considering substrings of $w$ of length 1, 2, $\ldots$, $n$.

For substrings of $w$ with length = 1

For substrings of $w$ with length = 2
Decision Properties of CFLs

Idea behind CYK Algorithm ...

\[
\begin{align*}
S &\rightarrow AB \quad w = baaba \\
S &\rightarrow BC \\
A &\rightarrow BA \quad baab \quad aaba \\
B &\rightarrow CC \quad \emptyset \quad \{B\} \quad \{B\} \\
C &\rightarrow AB \quad baa \quad aab \quad aba \\
A &\rightarrow a \quad \{SA\} \quad \{B\} \quad \{S.C\} \quad \{SA\} \\
B &\rightarrow b \quad ba \quad aa \quad ab \quad ba \\
C &\rightarrow a \quad \{B\} \quad \{A,C\} \quad \{A,C\} \quad \{B\} \quad \{A,C\} \\
\end{align*}
\]

For substrings of \( w \) with length = 3

\[
\begin{align*}
\emptyset \\
\{B\} \\
\{B\} \\
\{B\} \\
\{B\}
\end{align*}
\]

Page 16 April 26, 2002 © B. Juliano, Computer Science @ CSU, Chico
Decision Properties of CFLs

Going back to our example earlier …

1 $b \{B\}$ 2 $a \{A,C\}$ 3 $a \{A,C\}$ 4 $b \{B\}$ 5 $a \{A,C\}$

$S \rightarrow AB$
$S \rightarrow BC$
$A \rightarrow BA$
$B \rightarrow CC$
$C \rightarrow AB$
$A \rightarrow a$
$B \rightarrow b$
$C \rightarrow a$

Copyright and Intellectual Property Notice

This document and parts of its contents are the Intellectual Property (IP) of Dr. Benjoe A. Juliano of the Department of Computer Science at California State University, Chico (CSUC). Dr. Juliano claims exclusive moral rights of ownership under current Copyright Laws (Title 17 of the United States Code and 1998 Digital Millenium Copyright Act) and IP Policies/Guidelines (CSUC EM83-08, EM97-07, and Article 39 of the CFACSU Contract) including, but not limited to:

- the exclusive right to copy, reproduce, and/or distribute this document;
- the right to be identified as the creator of this work (the right of attribution);
- the right to take action against false attribution; and
- the right to object to derogatory treatment of this work (the right of integrity).