

CSCI 256: Theory of Computing

CHAPTER 7 *Properties of Context-Free Languages*

Dr. Benjoe A. Juliano

Tel. 530 898-4619 (office)
530 898-6442 (dept)
Fax. 530 898-5995

Juliano@ecst.csuchico.edu
<http://www.ecst.csuchico.edu/~juliano>

B. Juliano © 2002, based on notes by J. Ullman



Normal Forms for CFGs

- A CFG $G = (V, \Sigma, P, S)$ is in **Chomsky Normal Form** (or **CNF**) if all productions in P are of the form $A \rightarrow BC$ or $A \rightarrow a$, where $A, B, C \in V$ and $a \in \Sigma$.
- Preliminary clean-up/simplifications:
 - Eliminate **useless symbols** $x \in VN\Sigma$ that do not appear in any derivation of a terminal string from S .
 - Eliminate **ϵ -productions** $A \rightarrow \epsilon$ for $A \in V$.
 - Eliminate **unit productions** $A \rightarrow B$ for $A, B \in V$.

B. Juliano © 2002, based on notes by J. Ullman



Normal Forms for CFGs

- **Eliminating Useless Symbols**
 - A symbol $X \in VN\Sigma$ is **useful** for a grammar $G = (V, \Sigma, P, S)$ if there is some derivation of the form $S \xRightarrow{*} \alpha X \beta \xRightarrow{*} w$, where $w \in \Sigma^*$.
 - If $X \in VN\Sigma$ is not useful, we say it is **useless**.
 - Eliminating useless symbols from a grammar will not change the language generated.

B. Juliano © 2002, based on notes by J. Ullman



Normal Forms for CFGs

- **Properties of useful symbols**
 - A symbol $X \in VN\Sigma$ is **generating** if $X \xRightarrow{*} w$ for $w \in \Sigma^*$. Note that every $X \in \Sigma$ is generating since $w \in \Sigma^*$ can be X itself.
 - A symbol $X \in VN\Sigma$ is **reachable** if there is a derivation $S \xRightarrow{*} \alpha X \beta$ for $\alpha, \beta \in (V + \Sigma)^*$.

B. Juliano © 2002, based on notes by J. Ullman



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:
 - 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.
 - Round #2:** Eliminate all symbols that are *unreachable* in the grammar G_2 .
- Then G_1 has no useless symbols, and $L(G_1)=L(G)$.

B. Juliano © 2002, based on notes by J. Ullman



5

Normal Forms for CFGs

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 α is already known to be generating. Then A is generating. Note that this rule includes the case where $\alpha=\epsilon$.
- Used for “Round #1” of Theorem 7.2 ...

B. Juliano © 2002, based on notes by J. Ullman



6

Normal Forms for CFGs

Theorem 7.6

- The following algorithm finds all and only the *reachable symbols* of grammar $G=(V,\Sigma,P,S)$:
 - BASIS:** S is reachable.
 - INDUCTION:** Suppose it has been determined that $A\in V$ is reachable. Then for all $(A\rightarrow\alpha)\in P$, all symbols in $\alpha\in(V+\Sigma)^*$ are also reachable.
- Used for “Round #2” of Theorem 7.2 ...

B. Juliano © 2002, based on notes by J. Ullman



7

Normal Forms for CFGs

Example:

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

$$P: \begin{array}{ll} S \rightarrow AB \mid C & B \rightarrow 1 \mid A0 \\ A \rightarrow 0B \mid C & C \rightarrow AC \mid C1 \end{array}$$
- 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.

B. Juliano © 2002, based on notes by J. Ullman



8

Normal Forms for CFGs

Example, *continued ...*

- So, define $G_2=(V_2,\Sigma_2,P_2,S)$ where

$$P_2: S \rightarrow AB \quad B \rightarrow 1 / A0 \\ A \rightarrow 0B$$

- 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.

B. Juliano © 2002, based on notes by J. Ullman

9

Normal Forms for CFGs

- Eliminating ϵ -Productions**
 - If a language L has a CFG, then $L-\{\epsilon\}$ has a CFG without ϵ -productions.
 - If $\epsilon \notin L$, then L itself is $L-\{\epsilon\}$, so L has a CFG without ϵ -productions.
- A variable $A \in V$ is *nullable* if $A \Rightarrow \epsilon$.
- If A is nullable, then whenever A appears in a production body, say $B \rightarrow CAD$, A might (or might not) derive ϵ .

B. Juliano © 2002, based on notes by J. Ullman

10

Normal Forms for CFGs

Theorem 7.7

- In any grammar $G=(V,\Sigma,P,S)$, the only *nullable symbols* are the variables found by the following algorithm:
 - BASIS:** If $(A \rightarrow \epsilon) \in P$, then A is nullable.
 - INDUCTION:** If $(B \rightarrow C_1 C_2 \dots C_k) \in P$, where each $C_i \in V$ is nullable, then B is nullable.
- (Hence, we only need to consider productions with all-variable bodies.)

B. Juliano © 2002, based on notes by J. Ullman

11

Normal Forms for CFGs

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 *eliminating ϵ -productions* given below, then $L(G_1) = L(G)-\{\epsilon\}$.
 - Round #1:** Determine all nullable symbols of G .
 - Round #2:** For each $(A \rightarrow X_1 X_2 \dots X_k) \in P$, $k \geq 1$, 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.

B. Juliano © 2002, based on notes by J. Ullman

12

Normal Forms for CFGs

Example

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

$$P: S \rightarrow AB$$

$$A \rightarrow aAA / \epsilon \quad B \rightarrow bBB / \epsilon$$

- Applying “Round #1” (*Theorem 7.7*):
 - $A, B \in V$ are nullable from $A \rightarrow \epsilon$ and $B \rightarrow \epsilon$.
 - $S \in V$ nullable from $S \rightarrow AB$.
 - So, all variables in V are nullable.

Normal Forms for CFGs

Example, continued ...

- Applying “Round #2” (*from Theorem 7.9*):

- Productions added from $S \rightarrow AB$:

$$S \rightarrow AB / A / B$$

- Productions added from $A \rightarrow aAA$:

$$A \rightarrow aAA / aA / a$$

- Productions added from $B \rightarrow bBB$:

$$B \rightarrow bBB / bB / b$$

- So, define $G_1=(V,\Sigma,P_1,S)$ where

$$P_1: S \rightarrow AB / A / B \quad B \rightarrow bBB / bB / b$$

$$A \rightarrow aAA / aA / a$$

Normal Forms for CFGs

- 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 $A \xRightarrow{*} B$ using only unit productions, is called a **unit pair**.

Normal Forms for CFGs

Theorem 7.11

- The following inductive construction algorithm finds exactly the **unit pairs** for a CFG G :
 - BASIS:** (A, A) 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*:
 - A **Round #1**: Find all the unit pairs of G .
 - A **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, $L(G_1) = L(G)$.

B. Juliano © 2002, based on notes by J. Ullman

17

Normal Forms for CFGs

▪ Cleaning up grammars

1. Eliminate ϵ -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 ϵ , then there is another CFG G_1 such that $L(G_1) = L(G) - \{\epsilon\}$, and G_1 has no ϵ -productions, unit productions, or useless symbols.

B. Juliano © 2002, based on notes by J. Ullman

18

Normal Forms for CFGs

- **Chomsky Normal Form**
 - Every CFL without ϵ 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
 - $A\rightarrow a$, where $A\in V$ and $a\in\Sigma$.
 - Further, G has no useless symbols. Such a grammar is said to be in *Chomsky Normal Form*, or *CNF*.

B. Juliano © 2002, based on notes by J. Ullman

19

Normal Forms for CFGs

▪ 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 .

B. Juliano © 2002, based on notes by J. Ullman

20

Normal Forms for CFGs

Procedure for Converting to CNF, *continued* ...

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 $A \rightarrow B_1 B_2 \dots B_k$, $k \geq 3$, introduce $k-2$ new variables C_1, C_2, \dots, C_{k-2} .

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

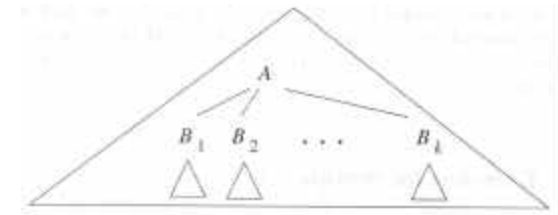
$$\begin{array}{l} A \rightarrow B_1 C_1 \quad \times \\ C_1 \rightarrow B_2 C_2 \quad C_{k-3} \rightarrow B_{k-2} C_{k-2} \\ \times \quad C_{k-2} \rightarrow B_{k-1} B_k \end{array}$$

B. Juliano © 2002, based on notes by J. Ullman

21

From Figure 7.4 (b) of (A.TLC, Hopcroft, Motwani, & Ullman, 2001).

Normal Forms for CFGs

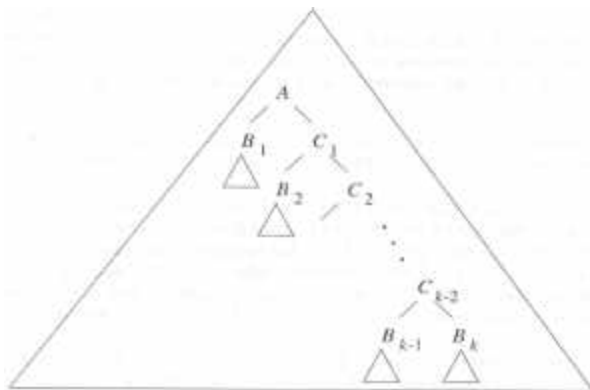


Detail of Step #3: For each production $A \rightarrow B_1 B_2 \dots B_k$, $k \geq 3$, introduce $k-2$ new variables C_1, C_2, \dots, C_{k-2} .

B. Juliano © 2002, based on notes by J. Ullman

22

Normal Forms for CFGs



Detail of Step #3: Replace the original production $A \rightarrow B_1 B_2 \dots B_k$ by $k-1$ cascading productions.

B. Juliano © 2002, based on notes by J. Ullman

23

Normal Forms for CFGs

Theorem 7.16

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

B. Juliano © 2002, based on notes by J. Ullman

24

Pumping Lemma for CFLs



B. Juliano © 2002, based on notes by J. Ullman

25

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 $|w| \leq 2^{n-1}$.

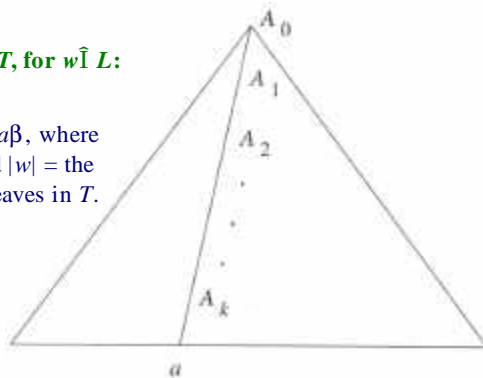
B. Juliano © 2002, based on notes by J. Ullman

26

Pumping Lemma for CFLs

Parse tree, T , for $w \in L$:

Note: $w = \alpha a \beta$, where $\alpha, \beta \in \Sigma^*$, and $|w|$ = the number of leaves in T .



Every sufficiently long string $w \in L$ must have a long path in its parse tree.

B. Juliano © 2002, based on notes by J. Ullman

27

Pumping Lemma for CFLs

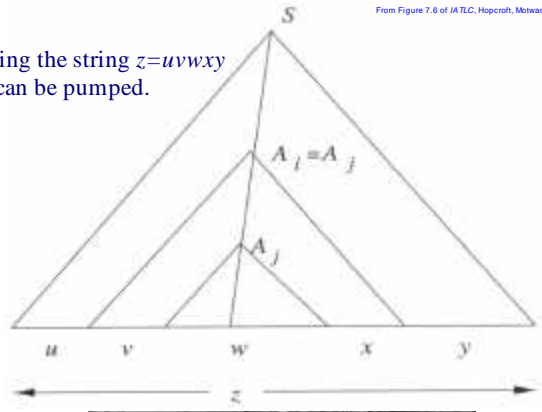
Theorem 7.18 (The pumping lemma for CFLs)

- 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:
 - $|vwx| \leq n$;
 - $vx \neq \epsilon$ – 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 .

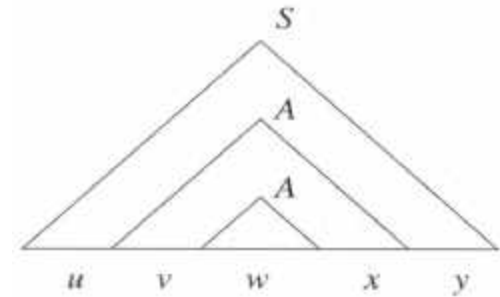
B. Juliano © 2002, based on notes by J. Ullman

28

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

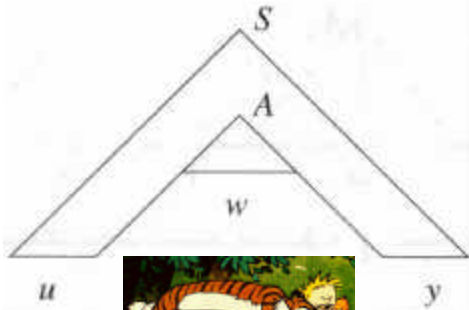


Pumping Lemma for CFLs



Underlying idea: Initial configuration prior to pumping ...

Pumping Lemma for CFLs

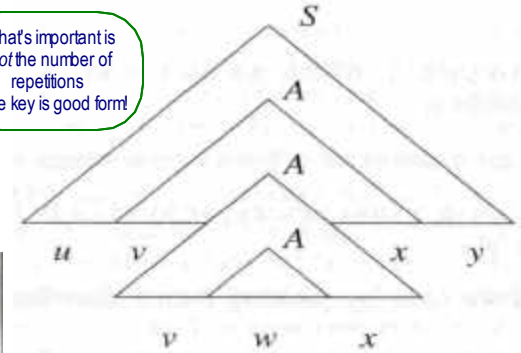


Result of pumping the (sub)strings v and x zero times.



Pumping Lemma for CFLs

What's important is *not* the number of repetitions – the key is good form!



Result of pumping the (sub)strings v and x twice.



Pumping Lemma for CFLs

Proof outline:

- Let L be a CFL. Let there be a CNF CFG for L with m variables. Pick $n=2^m$.
- 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.

B. Juliano © 2002, based on notes by J. Ullman

33

Pumping Lemma for CFLs

Proof outline:

- Focus on some path that is as long as any path in the tree. In this path, we can find some duplication of some variable A among the bottom $m+1$ variables on the path.
 - Let the lower A derive w and the upper A derive vwx .
- CNF guarantees that $|vwx| \leq n$ and $v \neq \epsilon$.
- By repeatedly replacing the lower A 's tree by the upper A 's tree, we see uv^iwx^iy has a parse tree for all $i \geq 0$.

B. Juliano © 2002, based on notes by J. Ullman

34

Closure Properties of CFLs

Substitutions

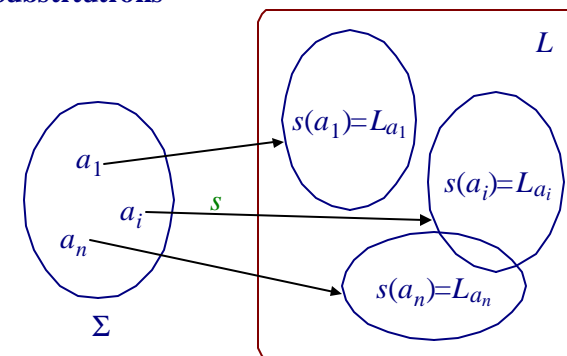
- Let Σ be an alphabet, and suppose that for every $a \in \Sigma$, we choose a language L_a . These chosen languages can be over any alphabets, not necessarily Σ and not necessarily the same.
- This choice of languages defines a function s (a **substitution**) on Σ , and we shall refer to L_a as $s(a)$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

B. Juliano © 2002, based on notes by J. Ullman

35

Closure Properties of CFLs

Substitutions



B. Juliano © 2002, based on notes by J. Ullman

36

Closure Properties of CFLs

Substitutions

- If $w = a_1 a_2 \dots a_n$, $a_n \in \Sigma^*$, then $s(w)$ is the language over all strings $x_1 x_2 \dots x_n$ 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 $s(a_1) s(a_2) \dots s(a_n)$.
 - $s(L)$ is the *union* of $s(w)$ for all $w \in L$.

B. Juliano © 2002, based on notes by J. Ullman

37

Closure Properties of CFLs

Theorem 7.23 (*Substitution Theorem*)

- If L is a CFL over alphabet Σ , and s is a substitution on Σ 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.

B. Juliano © 2002, based on notes by J. Ullman

38

Closure Properties of CFLs

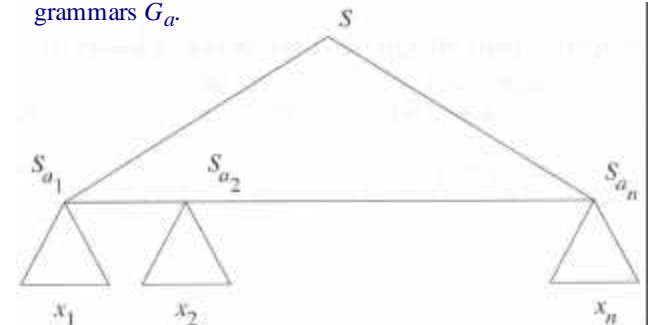
- Construct a new grammar $G' = (V', T', P', S')$ for $s(L)$ as follows:
 - V' is the union of V and all the V_a 's for $a \in \Sigma$.
 - T' is the union of all the T_a 's for $a \in \Sigma$.
 - P' consists of:
 - all productions in any P_a for $a \in \Sigma$.
 - the productions of P , but with each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

B. Juliano © 2002, based on notes by J. Ullman

39

Closure Properties of CFLs

- 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 .



B. Juliano © 2002, based on notes by J. Ullman

40

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

Closure Properties of CFLs

- Applications of the Substitution Theorem

Proof:

- Union:* Let L_1 and L_2 be CFLs. Then L_1NL_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_1L_2 is the language $s(L)$, where language $L = \{12\}$, and substitution s is defined by $s(1) = L_1$ and $s(2) = L_2$.

Closure Properties of CFLs

- Applications of the Substitution Theorem

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 $L_1^+ = s(L)$.

Closure Properties of CFLs

- Applications of the Substitution Theorem

Proof:

- Homomorphism:* Let L be CFL over Σ and h a homomorphism on Σ . 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.

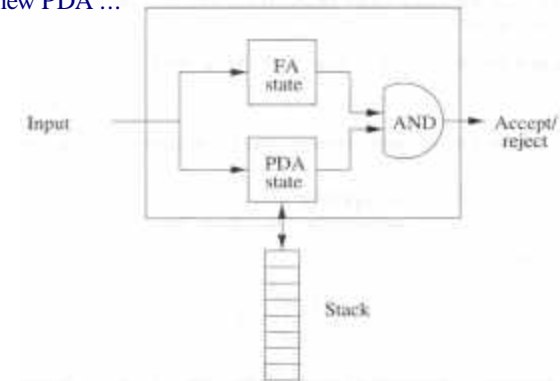
B. Juliano © 2002, based on notes by J. Ullman

45

From Figure 7.9 of (ATLC, Hopcroft, Motwani, & Ullman, 2001).

Closure Properties of CFLs

- Combining a PDA and a FA to run in parallel to create a new PDA ...



B. Juliano © 2002, based on notes by J. Ullman

46

Closure Properties of CFLs

Formally ...

- Given:
 - PDA $P = (Q_P, \Sigma, \Gamma, \delta_P, q_P, Z_0, F_P)$ 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 $\delta((q,p), a, X)$ is defined to be the set of all pairs $((r,s), \gamma)$ such that:
 - " $s = \Delta_A(p,a)$; and
 - " $(r,\gamma) \in \delta_P(q,a,X)$

B. Juliano © 2002, based on notes by J. Ullman

47

Closure Properties of CFLs

Theorem 7.29

- The following are true about CFLs L , L_1 , and L_2 , and a RL R :
 - $L \cap R$ is a CFL.
 - \bar{L} is not necessarily a CFL.
 - $L_1 \cap L_2$ is not necessarily a CFL.

B. Juliano © 2002, based on notes by J. Ullman

48

Closure 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 $a \in \Sigma$.

B. Juliano © 2002, based on notes by J. Ullman

49

Closure Properties of CFLs

- Construction of $P' = (Q', \Sigma, \Gamma, \delta', (q_0, \epsilon), Z_0, F \times \{\epsilon\}) \dots$
 - δ' is defined by the following rules:
 - $\delta'((q, \epsilon), a, X) = \{(q, h(a)), X\}$ for all $a \in \Sigma$ and $a \neq \epsilon$, $q \in Q$, and $X \in \Gamma$.
 - If $(p, \gamma) \in \delta(q, b, X)$, where $b \in T$ or $b = \epsilon$, then $((p, x), \gamma) \in \delta'((q, bx), \epsilon, X)$.
 - The start state of P' , (q_0, ϵ) , is the start state of P with an empty buffer.

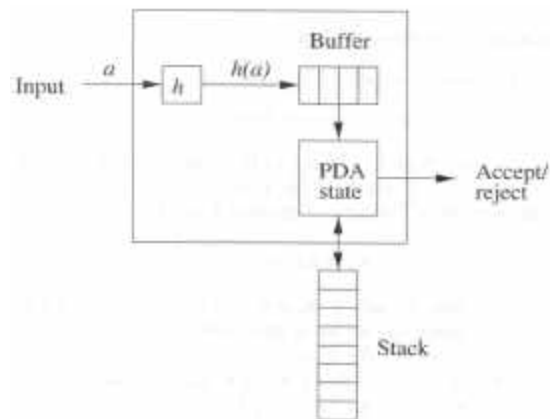
Hence, to show $L(P') = h^{-1}(L(P))$,

- $(q_0, h(w), Z_0) \vdash_P^* (p, \epsilon, \gamma) \Leftrightarrow ((q_0, \epsilon), w, Z_0) \vdash_{P'}^* ((p, \epsilon), \epsilon, \gamma)$

B. Juliano © 2002, based on notes by J. Ullman

50

Closure Properties of CFLs



B. Juliano © 2002, based on notes by J. Ullman

51

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?

B. Juliano © 2002, based on notes by J. Ullman

52

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.

B. Juliano © 2002, based on notes by J. Ullman

53

Decision Properties of CFLs

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.

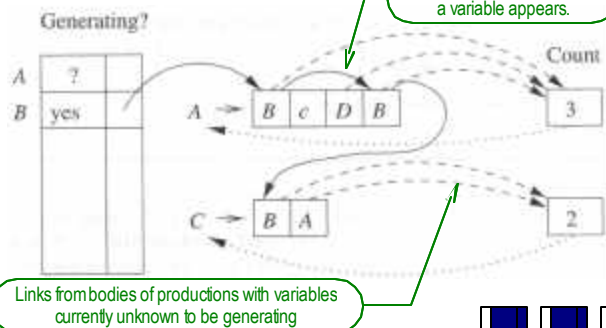
B. Juliano © 2002, based on notes by J. Ullman

54

Decision Properties of CFLs

Testing Emptiness of CFLs

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



B. Juliano © 2002, based on notes by J. Ullman

55

Decision Properties of CFLs

Testing Membership in a CFL

- CYK Algorithm** by J. Cocke, D. Younger (1967) and T. Kasami (1965).
 - “dynamic programming” algorithm
 - Inputs:**
 - CNF grammar $G = (V, \Sigma, P, S)$ for CFL L
 - $w = a_1 a_2 \dots a_n$, $a_i \in \Sigma^*$
 - Output:**
 - Decision, in $O(n^3)$ time, whether $w \in L$.

B. Juliano © 2002, based on notes by J. Ullman

56

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 $L(G)$.

Decision Properties of CFLs

- Idea behind CYK Algorithm ...

$$\begin{array}{r}
 S \rightarrow AB \\
 S \rightarrow BC \\
 A \rightarrow BA \\
 B \rightarrow CC \\
 C \rightarrow AB \\
 A \rightarrow a \\
 B \rightarrow b \\
 C \rightarrow a
 \end{array}
 \qquad
 \begin{array}{r}
 w = baaba \\
 \\
 baab \quad aaba \\
 baa \quad aab \quad aba \\
 ba \quad aa \quad ab \quad ba \\
 b \quad a \quad a \quad b \quad a
 \end{array}$$

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

Decision Properties of CFLs

- Idea behind CYK Algorithm ...

$$\begin{array}{r}
 S \rightarrow AB \\
 S \rightarrow BC \\
 A \rightarrow BA \\
 B \rightarrow CC \\
 C \rightarrow AB \\
 A \rightarrow a \\
 B \rightarrow b \\
 C \rightarrow a
 \end{array}
 \qquad
 \begin{array}{r}
 w = baaba \\
 \\
 baab \quad aaba \\
 baa \quad aab \quad aba \\
 ba \quad aa \quad ab \quad ba \\
 \{B\} \quad \{A,C\} \quad \{A,C\} \quad \{B\} \quad \{A,C\} \\
 b \quad a \quad a \quad b \quad a
 \end{array}$$

For substrings of w with length = 1

Decision Properties of CFLs

- Idea behind CYK Algorithm ...

$$\begin{array}{r}
 S \rightarrow AB \\
 S \rightarrow BC \\
 A \rightarrow BA \\
 B \rightarrow CC \\
 C \rightarrow AB \\
 A \rightarrow a \\
 B \rightarrow b \\
 C \rightarrow a
 \end{array}
 \qquad
 \begin{array}{r}
 w = baaba \\
 \\
 baab \quad aaba \\
 baa \quad aab \quad aba \\
 \{S,A\} \quad \{B\} \quad \{S,C\} \quad \{S,A\} \\
 ba \quad aa \quad ab \quad ba \\
 \{B\} \quad \{A,C\} \quad \{A,C\} \quad \{B\} \quad \{A,C\} \\
 b \quad a \quad a \quad b \quad a
 \end{array}$$

For substrings of w with length = 2

Decision Properties of CFLs

- Idea behind CYK Algorithm ...

$S \rightarrow AB$	$w = baaba$			
$S \rightarrow BC$				
$A \rightarrow BA$	<i>baab</i>	<i>aaba</i>		
$B \rightarrow CC$	ϵ	$\{B\}$	$\{B\}$	
$C \rightarrow AB$	<i>baa</i>	<i>aab</i>	<i>aba</i>	
$A \rightarrow a$	$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$
$B \rightarrow b$	<i>ba</i>	<i>aa</i>	<i>ab</i>	<i>ba</i>
$C \rightarrow a$	$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$
	<i>b</i>	<i>a</i>	<i>a</i>	<i>b</i>
				<i>a</i>

For substrings of w with length = 3

B. Juliano © 2002, based on notes by J. Ullman

61

Decision Properties of CFLs

- Idea behind CYK Algorithm ...

$S \rightarrow AB$	$w = baaba$			
$S \rightarrow BC$	ϵ	$\{S,A,C\}$		
$A \rightarrow BA$	<i>baab</i>	<i>aaba</i>		
$B \rightarrow CC$	\emptyset	$\{B\}$	$\{B\}$	
$C \rightarrow AB$	<i>baa</i>	<i>aab</i>	<i>aba</i>	
$A \rightarrow a$	$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$
$B \rightarrow b$	<i>ba</i>	<i>aa</i>	<i>ab</i>	<i>ba</i>
$C \rightarrow a$	$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$
	<i>b</i>	<i>a</i>	<i>a</i>	<i>b</i>
				<i>a</i>

For substrings of w with length = 4

B. Juliano © 2002, based on notes by J. Ullman

62

Decision Properties of CFLs

- Idea behind CYK Algorithm ...

	$\{S,A,C\}$			
$S \rightarrow AB$	$w = baaba$			
$S \rightarrow BC$	\emptyset	$\{S,A,C\}$		
$A \rightarrow BA$	<i>baab</i>	<i>aaba</i>		
$B \rightarrow CC$	\emptyset	$\{B\}$	$\{B\}$	
$C \rightarrow AB$	<i>baa</i>	<i>aab</i>	<i>aba</i>	
$A \rightarrow a$	$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$
$B \rightarrow b$	<i>ba</i>	<i>aa</i>	<i>ab</i>	<i>ba</i>
$C \rightarrow a$	$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$
	<i>b</i>	<i>a</i>	<i>a</i>	<i>b</i>
				<i>a</i>

For substrings of w with length = 5

B. Juliano © 2002, based on notes by J. Ullman

63

Decision Properties of CFLs

- CYK Algorithm detail

- Given CFG $G = (V, \Sigma, P, S)$ in CNF, $w = a_1 a_2 \dots a_n$, $a_n \in \Sigma^*$
- Fill out $n \times n$ matrix X as follows:

$$X_{i,i} = \text{AgV}, \{A\} a_i \text{fgP} \quad |$$

$$X_{i,j} = \text{AgV}, \begin{matrix} \{A\} BC \text{fg} P \\ B \text{g} X_{i,k}, C \text{g} X_{kA1,j} \\ iRkPj \end{matrix} \quad |$$

- $w \in L(G)$ if and only if $S \in X_{1,n}$.

B. Juliano © 2002, based on notes by J. Ullman

64

Decision Properties of CFLs

- Going back to our example earlier ...

	1	2	3	4	5
1	b {B}	{S,A}	\emptyset	\emptyset	{S,A,C}
$S \rightarrow AB$	2	a {A,C}	{B}	{B}	{S,A,C}
$S \rightarrow BC$			3	a {A,C}	{S,C}
$A \rightarrow BA$				4	b {B}
$B \rightarrow CC$					5
$C \rightarrow AB$					a {A,C}
$A \rightarrow a$					
$B \rightarrow b$					
$C \rightarrow a$					

B. Juliano © 2002, based on notes by J. Ullman

65

Decision Properties of CFLs

- Preview of Undecidable CFL Problems

- Undecidability** – there are problems we cannot solve by *any* algorithm that can run on a computer.
- The following are undecidable:
 - Is a given CFG G ambiguous?
 - Is a given CFL L inherently ambiguous?
 - Are two CFLs, L_1 and L_2 , the same?
 - Is a given CFL L equal to Σ^* , where Σ is the alphabet of L ?

B. Juliano © 2002, based on notes by J. Ullman

66

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 Millennium 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).

B. Juliano © 2002, based on notes by J. Ullman



67