PROPERTIES of CONTEXT-FREE LANGUAGES

Chap. 8

Summary

- The general properties for CFL to compare
  with the properties of Regular Languages (RL).
- The property of closure and decision algorithms
  – more complicated for CFL.
- More complicate for the two Pumping Lemmas,
  – one for CFL, the other for Linear Languages ($\supset$ RL).
- While the basic idea is the same as for Regular languages, specific arguments normally involve much more detail.

Learning Objectives

- Apply the *Pumping Lemma* to show that a language is ***not*** *Context-Free*.
- State the *Closure properties* applicable to CFL.
- Prove that CFLs are *closed* under the operations of *union, concatenation*, and *star-closure*.
- Prove that CFLs are ***not*** *closed* under either *intersection* or *complementation*.
- Describe a *Membership Algorithm* for CFL.
- Describe an *algorithm* to determine if a CFL is *empty*.
- Describe an *algorithm* to determine if a CFL is *infinite*.

A Pumping Lemma for Context-Free Language

- Theorem 8.1: Given an *infinite* Context-Free Laguage L, there exists some positive integer $\boldsymbol{m}$ such that any sufficiently long string $w \in L$ with $|w| \geq m$ can be decomposed as
  - $w = uvxyz$, $u, v, x, y, z \in T^*$ with
  - $|vy| \geq 1$ and $|vxy| \leq m$

    where m is an arbitrary integer, $0 < m \leq |w|$
  - an arbitrary, but equal number of repetitions of v and y yields another string in L: $w_i = uv^ixy^iz \in L$, $i \geq 0$.
- The "pumped" string consists of two separate parts (v and y) and can occur anywhere in the string.
- The Pumping Lemma can be used to show, by contradiction, that a certain language is *not context-free*.

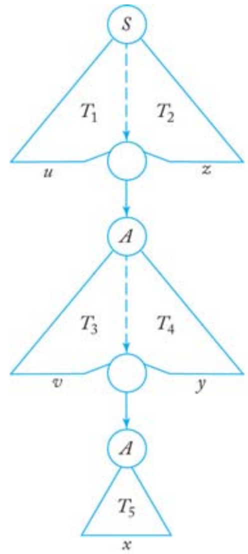
An Illustration of the Pumping Lemma for CFL

As shown in Fig. 8.1,

the pumping lemma for CFLs can be illustrated by sketching

a general derivation tree that shows a decomposition of the string into the required components.

Proof: ?



Proof: Pumping Lemma for CFL

<u>Proof</u>: Consider L – {λ} and assume a CFG G without unit-productions or λ-productions.

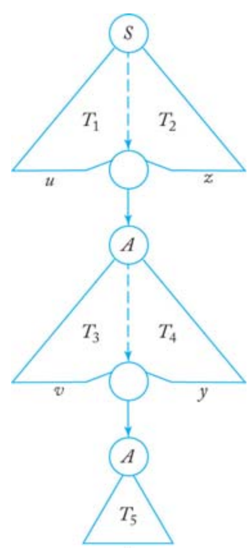
Since the length of string on the right side of any production is bounded, say by *k*, the length of the derivation of any $w \in L$ must be at least $|w|/k$. Thus, since L is infinite, there exist arbitrarily long derivations and corresponding derivation trees of arbitrary height.

Consider such a high derivation tree and some sufficiently long path from the root to a leaf. Since the number of variables in G is finite, there must be some variable that repeats on this path.

Corresponding to the derivation tree in Fig.,

$\exists\ S \Rightarrow^* uAz \Rightarrow^* uvAyz \Rightarrow^* uvxyz$ for $u,v,x,y,z \in T^*$

From this, we see $A \Rightarrow^* vAy$ and $A \Rightarrow^* x$, so all the strings uv^ixy^iz, i = 0,1,2, .., can be generated by the grammar and are therefore in *L*.



Proof: Pumping Lemma for CFL

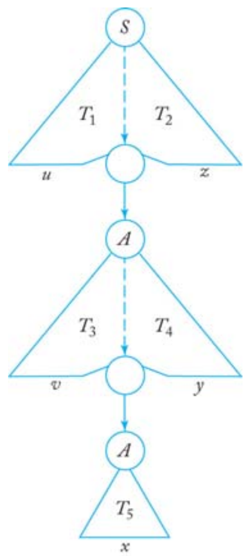
<u>Cont.</u>: $S \Rightarrow^* uAz \Rightarrow^* uvAyz \Rightarrow^* uvxyz$, $u,v,x,y,z \in T^*$

In $A \Rightarrow^* vAy$ and $A \Rightarrow^* x$, assume that no variable repeats. In Fig., no variable repeats in T_5. Similarly, assume no variable repeats in T_3 and T_4. Thus, the lengths of v, x and y depend only on the productions of the grammar and can be bounded independently of w, so $|vxy| \leq m$ holds.

Since there are no unit/λ-productions, v and y can't both be empty, so $|vy| \geq 1$.

This completes the argument that w = $uvxyz$ and

$w_i = uv^ixy^iz \in L$, $i \geq 0$ holds.



Example: Pumping Lemma

- <u>Example 8.1</u>: L = $\{a^nb^nc^n \mid n \geq 0\}$ is not context free.

Pf) Let's choose $n=m$ for an arbitrary m, so $w=a^mb^mc^m \in L$.

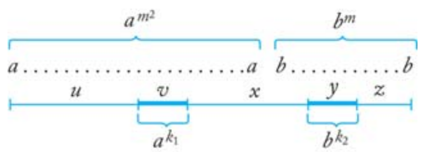
For $w=uvxyz$, if we choose vxy contains only a's since $|vxy| \leq m$, then $u=a^j$, $v=a^k$, $x=a^p$, $y=a^q$, $z=a^{m-j-k-p-q}b^mc^m$.

Then, by Pumping Lemma, $w_i = uv^ixy^iz$.

With i=0, $w_0 = uv^0xy^0z = uxz = a^{j+p}\ a^{m-j-k-p-q}b^mc^m = a^{m-k-q}b^mc^m \notin L$.

So, L = $a^nb^nc^n$ is *not* a *CFL.*

- <u>Example 8.3</u>: L = $\{a^{n!} \mid n \geq 0\}$ is not context free.
- <u>Example 8.4</u>: L = $\{a^nb^j \mid n = j^2\}$ is not context free.



Example: Pumping Lemma

- <u>Example 8.2</u>: L = $\{ww \mid w \in \{a,b\}^*\}$ is not context free.

Pf) Let's choose $ww=a^m b^m a^m b^m \in L$.

For $w=uvxyz$, let's choose each substring

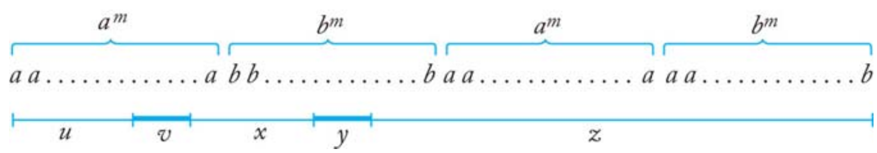
$u=a^j, v=a^k, x=a^{m-j-k}b^p, y=b^q, z=b^{m-p-q}a^m b^m$

where $m \geq j, k, p, q > 0$ and $|vxy|=m-j+p+q = m-(j-p-q) \leq m$.

By Pumping Lemma, $w_i = uv^ixy^iz$.

Thus, with i=0, $w_0 = uv^0xy^0z = uxz = a^{m-k}b^{m-q}a^m b^m \notin L$.

So, L = ww is *not* a *CFL.*



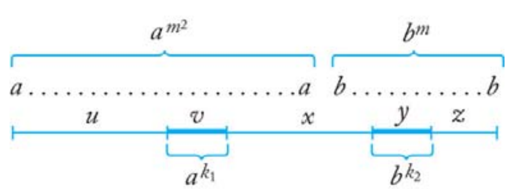
Example: Pumping Lemma

- <u>Example 8.3</u>: L = $\{a^{n!} \mid n \geq 0\}$ is not context free.

Pf) Similar to the Examples 8.1 & 8.2. See textbook.

- <u>Example 8.4</u>: L = $\{a^n b^j \mid n = j^2\}$ is not context free.

Pf) See the textbook.



Pumping Lemma for Linear Language

- Definition 8.1: A CFL L is said to be linear
  if there exists a *linear* CFG G s.t. $L = L(G)$.
  (*at most one nonterminal* in the right side of production)

Cf) A grammar G = (V, T, S, P) is *linear CFG*
  if all productions in P are linear with the form
  $A \rightarrow xBy$, where $A,B \in V$ and $x, y \in T^*$.

- Example 8.5(A):
  $L = \{a^n b^n \mid n \geq 0\}$ is a linear language, generated by a linear CFG,
  $G = (\{S\}, \{a, b\}, S, P)$ with P given by $\{ S \rightarrow aSb \mid \lambda \}$.
- Example 8.5(B):
  $L = \{w \mid n_a(w) = n_b(w)\}$ is not linear because the grammar G,
  s.t. L = L(G), is not linear where $G = (\{S\}, \{a, b\}, S, P)$
  with P given by $\{ S \rightarrow SS \mid aSb \mid bSa \mid \lambda \}$.

Pumping Lemma for Linear Language

- Theorem 8.2: Let L be an infinite linear language.

Then, $\exists\, m > 0$, s.t. for any $w \in L$, with $|w| \geq m$ can be decomposed as $w = uvxyz$ with

$|vy| \geq 1$ and $|uvyz| \leq m$,

s.t. $w_i = uv^i xy^i z \in L \;\; \forall i = 0, 1, 2, \ldots$.

- The updated condition, $|uvyz| \leq m$, implies v and y must be within m symbols of the left & right ends of w, respectively.
- x can be of arbitrary length.

Pumping Lemma for Linear Language

- Theorem 8.2: Let L be an infinite linear language.

Then, $\exists\, m > 0$, s.t. for any $w \in L$, with $|w| \geq m$ can be decomposed as $w = uvxyz$ with $|uvyz| \leq m$, $|vy| \geq 1$, s.t. $w_i = uv^ixy^iz \in L \ \forall i$.

Proof: Since the language is linear there exists a linear grammar G for it. For the argument it is convenient to assume that G has no unit-productions and no λ-productions. An examination of the proofs of Theorem 6.3 and 6.4 makes it clear that removing unit-productions and λ-productions does not destroy the linearity of the grammar. We can therefore assume that G has the required property.

Consider now the derivation of a string $w \in L(G)$

$$S \stackrel{*}{\Rightarrow} uAz \stackrel{*}{\Rightarrow} uvAyz \stackrel{*}{\Rightarrow} uvxyz = w.$$

Assume, for the moment, that for every $w \in L(G)$, there is a variable A, such that

1. in the partial derivation $S \stackrel{*}{\Rightarrow} uAz$ no variable is repeated,
2. in the partial derivation $S \stackrel{*}{\Rightarrow} uAz \stackrel{*}{\Rightarrow} uvAyz$ no variable except A is repeated,
3. the repetition of A must occur in the first m steps, where m can depend on the grammar, but not on w.

If this is true, then the lengths of u, v, y, z must be bounded independent of w. This in turn implies that (8.5), (8.6), and (8.7) must hold.

Pumping Lemma for Linear Language

- Example 8.6: $L = \{w \mid n_a(w) = n_b(w)\}$ is not linear.

Assume that L is linear and apply Pumping Lemma in Thm. 8.2 to $w = a^m b^{2m} a^m$.

$|uvyz| \leq m$ shows that the substrings u, v, y, z must all consist of all a's: $u=a^j$, $v=a^k$, $y=a^p$, $z=a^q$, $x=a^{m-j-k}b^{2m}a^{m-p-q}$.

Then, by Pumping lemma for Linear language, $w_i = uv^ixy^iz$.

With i=0, $w_0 = uv^0xy^0z = uxz = a^j\, a^{m-j-k}b^{2m}a^{m-p-q}\, a^q = a^{m-k}b^{2m}a^{m-p} \notin L$.

So, $L = a^nb^nc^n$ is not a CFL.

Closure Properties for CFL

- Theorem 8.3: If L_1 and L_2 are context-free languages, so are the languages that result from the following operations:
  - $L_1 \cup L_2$: union
  - $L_1 \cdot L_2$: concatenation.
  - $L_1{}^*$: star-closure
- In other words, the family of CFLs is closed under union, intersection, and star-closure.
- To prove these properties, we assume the existence of two context-free grammars G_1 and G_2 that generate the respective languages

Proof of Closure under Union

- Let L_1 and L_2 are CFLs generated by the Context-Free Grammars $G_1 = (V_1, T_1, S_1, P_1)$ and $G_2 = (V_2, T_2, S_2, P_2)$, respectively.
  Assume V_1 and V_2 are disjoint: $V_1 \cap V_2 = \emptyset$.
- Create a new variable $S_3 \notin V_1 \cup V_2$.
- Construct a new grammar $G_3 = (V_3, T_3, S_3, P_3)$ so that
  - $V_3 = V_1 \cup V_2 \cup \{ S_3 \}$
  - $T_3 = T_1 \cup T_2$
  - $P_3 = P_1 \cup P_2$
- Add to P_3 a production that allows the new start symbol to derive either of the start symbols for L_1 and L_2
  $S_3 \to S_1 \mid S_2$. So, $P_3 = P_1 \cup P_2 \cup \{S_3 \to S_1 \mid S_2\}$
- Clearly, G_3 is Context-Free grammar so that the generated language, $L(G_3)$ is a CFL.

Proof of Closure under Union (cont.)

- Clearly, G_3 is CFG so that the generated language, $L(G_3)$, is a CFL.
- Let's show that $L(G_3) = L_1 \cup L_2$.

$\leftarrow$) if $w \in L_1$ or $w \in L_2$ (i.e. $S_1 \Rightarrow_{G1}^* w$ or $S_2 \Rightarrow_{G2}^* w$)

then w is derived by G_3, i.e. $S_3 \Rightarrow_{G3}^* w$

Suppose $w \in L_1$. Then, $S_3 \Rightarrow S_1 \Rightarrow^* w$ is a possible derivation in G_3.

Similarly, for $w \in L_2$, $S_3 \Rightarrow S_2 \Rightarrow^* w$ is a possible derivation in G_3.

Thus, $w \in L(G_3)$.

$\rightarrow$) If $w \in L(G_3)$, then *either* $S_3 \Rightarrow S_1$ *or* $S_3 \Rightarrow S_2$ must be the 1st step of derivation.

- If $S_3 \Rightarrow S_1$ *is used, since sentential forms* derived from S_1 have variables in V_1, the derivation $S_1 \Rightarrow^* w$ can involve productions in P_1 only. Thus, w must be in L_1.
- Similarly, If $S_3 \Rightarrow S_2$ *is used,* w must be in L_2.
- Thus, $w \in L_1 \cup L_2$. *Therefore,* $L(G_3) = L_1 \cup L_2$.

Proof of Closure under Concatenation

- Let L_1 and L_2 are CFLs generated by the Context-Free Grammars $G_1 = (V_1, T_1, S_1, P_1)$ and $G_2 = (V_2, T_2, S_2, P_2)$, respectively.
  Assume V_1 and V_2 are disjoint: $V_1 \cap V_2 = \emptyset$.
- Create a new variable $S_4 \notin V_1 \cup V_2$.
- Construct a new grammar $G_4 = (V_4, T_4, S_4, P_4)$ so that
  - $V_4 = V_1 \cup V_2 \cup \{ S_4 \}$
  - $T_4 = T_1 \cup T_2$
  - $P_4 = P_1 \cup P_2$
- Add to P_4 a production that allows the new start symbol to derive the concatenation of the start symbols for L_1 and L_2
  $S_4 \rightarrow S_1S_2$. So, $P_4 = P_1 \cup P_2 \cup \{S_4 \rightarrow S_1S_2\}$
- Clearly, G_4 is context-free and generates the concatenation of L_1 and L_2, i.e. $L(G_4) = L(G_1)L(G_2)$.

Proof of Closure under Star-Closure

- Let L_1 be CFL generated by $G_1 = (V_1, T_1, S_1, P_1)$.
- Create a new variable $S_5 \notin V_1$.
- Construct a new grammar $G_5 = (V_5, T_5, S_5, P_5)$ so that
  - $V_5 = V_1 \cup \{ S_5 \}$
  - $T_5 = T_1$
  - $P_5 = P_1$
- Add to P_5 a production that allows the new start symbol S_5 to derive the repetition of the start symbol for L_1 any number of times

  $S_5 \rightarrow S_1S_5 \mid \lambda$. So, $P_5 = P_1 \cup P_2 \cup \{S_5 \rightarrow S_1S_5 \mid \lambda\}$
- Clearly, G_5 is context-free and generates the star-closure of L_1, i.e. $L(G_5) = L(G_1)^*$.

No Closure under Intersection

- Unlike regular languages, the intersection of two CFLs L_1 and L_2 does not necessarily produce a context-free language.
- As a counter example, consider the CFLs

  $L_1 = \{ a^nb^nc^m \mid n \geq 0, m \geq 0 \}$ where a grammar for L_1 is

  $S \rightarrow S_1S_2,\ S_1 \rightarrow aS_1b \mid \lambda,\ S_2 \rightarrow cS_2 \mid \lambda.$

  $L_2 = \{ a^nb^mc^m \mid n \geq 0, m \geq 0 \}$ whose CFL is, similarly,

  $S \rightarrow S_1S_2,\ S_1 \rightarrow aS_1 \mid \lambda,\ S_2 \rightarrow bS_2c \mid \lambda.$
- We note that L_1 is the concatenation of two CFLs, $\{a^nb^n \mid n \geq 0\}$ and $\{c^m \mid m \geq 0\}$, L_1 is context free by Th^m. 8.3. Similarly, so is L_2.
- However, the intersection of L_1 and L_2

  $L_3 = L_1 \cap L_2 = \{ a^nb^nc^n \mid n \geq 0 \}$

  has been proven *not context-free* by Pumping Lemma in Ex. 8.1.
- Therefore, CFL is not closed under intersection.

No Closure under Complementation

- The complement of a context-free language L_1 does not necessarily produce a context-free language.
- The proof is by contradiction: given CFLs L_1 and L_2, assume that their complements are also context-free.
- By Theorem 8.3, the union of the complements must also produce a context-free language L_3.
- Using our assumption, the complement of L_3 is also context-free.
- However, using the set identity below, we conclude that the complement of L_3 is the intersection of L_1 and L_2, which has been shown not to be context-free, thus contradicting our assumption.

$$L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}$$

Intersection of CFL and Regular Language

- Theorem 8.5: Let L_1 be a CFL and L_2 be a Regular language. Then, $L_1 \cap L_2$ is context free.

- Example 8.7: $L = \{a^n b^n : n \geq 0, n \neq 100\}$ is context free.

Proof) Let $L_1 = \{a^{100}b^{100}\}$.

Then, L_1 is regular since it's finite.

Then, L may be defined as $L = \{a^n b^n : n \geq 0\} \cap \bar{L}_1$

By the closure of regular languages (L_1) under complementation and the closure of CFL under regular intersection,

L is context free. Q.E.D.

Proof: Let $M_1 = (Q, \Sigma, \Gamma, \delta_1, q_0, z, F_1)$ be an npda that accepts L_1 and $M_2 = (P, \Sigma, \delta_2, p_0, F_2)$ be a dfa that accepts L_2. We construct a push-down automaton $\widehat{M} = \left(\widehat{Q}, \Sigma, \Gamma, \hat{\delta}, \hat{q}_0, z, \widehat{F}\right)$ that simulates the parallel action of M_1 and M_2: Whenever a symbol is read from the input string, $\widehat{M}$ simultaneously executes the moves of M_1 and M_2. To this end we let

$$\widehat{Q} = Q \times P,$$

$$\hat{q}_0 = (q_0, p_0),$$

$$\widehat{F} = F = F_1 \times F_2,$$

and define $\hat{\delta}$ such that

$$((q_k, P_l), x) \in \hat{\delta}\ ((q_i, p_j), a, b)$$

if and only if

$$(q_k, x) \in \delta_1\ (q_i, a, b)$$

and

$$\delta_2\ (p_j, a) = p_l.$$

In this, we also require that if $a = \lambda$, then $p_j = p_l$. In other words, the states of $\widehat{M}$ are labeled with pairs (q_i, p_j), representing the respective states in which M_1 and M_2 can be after reading a certain input string. It is a straightforward induction argument to show that

$$\left(\left(q_0, p_0\right), w, z\right) \overset{*}{\vdash}_{\widehat{M}} \left(\left(q_r, p_s\right), \lambda, x\right),$$

with $q_r \in F_1$ and $p_s \in F_2$ if and only if

$$\left(q_0, w, z\right) \overset{*}{\vdash}_{M_1} \left(q_r, \lambda, x\right),$$

and

$$\delta^*\ (p_0, w) = p_s.$$

Therefore, a string is accepted by $\widehat{M}$ if and only if it is accepted by M_1 and M_2, that is, if it is in $L\ (M_1) \cap L\ (M_2) = L_1 \cap L_2$.

Intersection of CFL and Regular Language

- Theorem 8.5: Let L_1 be a CFL and L_2 be a Regular language. Then, $L_1 \cap L_2$ is context free.
- Example 8.7: $L = \{a^n b^n : n \geq 0, n \neq 100\}$ is context free.
- Example 8.8: $L = \{w \in \{a, b, c\}^* \mid n_a(w) = n_b(w) = n_c(w)\}$ is *not* context free.

Proof) Suppose L is context free. Then,

$$L \cap L(a^*b^*c^*) = \{a^n b^n c^n \mid n \geq 0\}$$

would be context free since CFL $\cap$ Regular Language is CFL by Th^m 8.5. But, it's proven not CFL in Ex. 8.1. Contradiction!

Thus, L is not context free.   Q.E.D.

More Properties of Closures: (N)CFL, DCFL, Regular Lang, and Linear Lang.

$\forall$L1, L2 $\in$ (N)CFL, $\forall$L3 $\in$ RL, $\forall$L4 $\in$ DCFL

- The family of (N)CFL is closed under:
  - Homomorphism, Reversal.
- The family of (N)CFL is closed with the Regular lang. under:
  - Regular Difference: L1 – L2 $\notin$ CFL, but L1 – L3 $\in$ CFL.
    Also, If L4 is deterministic CFL, then L4 – L3 $\in$ DCFL.
- The family of Deterministic CFL is not closed under:
  - Union, intersection, *reversal,* but
  - Close under: homomorphism, regular difference.
- The family of Linear Language is closed under:
  - Union, but
  - Not closed under: concatenation, intersection
- The family of unambiguous CFL is not closed under:
  - Union, intersection

Elementary Questions about CFL

- Given a CFL *L* and an arbitrary string *w*, is there an algorithm to determine whether or not *w* is in *L*? : $w \in L(G)$? *Membership*
- Given a CFL *L*, is there an algorithm to determine if *L* is empty? : $L = \varnothing$? *Emptiness*
- Given a CFL *L*, is there an algorithm to determine if *L* is infinite? : *(In)finiteness*
- Given two CFGs G_1 and G_2, is there an algorithm to determine if $L(G_1) = L(G_2)$? : *Equality of grammar*

A Membership Algorithm for CFL

- The combination of Theorems 5.2 and 6.5 confirms the existence of a membership algorithm for CFL.
- By Theorem 5.2, exhaustive parsing is guaranteed to give the correct result for any context-free grammar that contains neither λ-productions nor unit-productions.
- By Theorem 6.5, such a grammar can always be produced if the language does not include λ.
- Alternatively, a NDPA to accept the language can be constructed as established by Theorem 7.1

Theorem 5.2:
Exhaustive parsing is guaranteed to yield all $w \in L(G)$ eventually, but may fail to stop for strings $w \notin L(G)$, unless we restrict the productions in the grammar to avoid the forms

$A \to \lambda$ (λ production) and $A \to B$ (unit production).

i.e. If the CFG G with *no* rule of the form

$A \to \lambda$ (λ production) or $A \to B$ (unit production),
the exhaustive parsing decides/stops for $w \notin L(G)$.

Theorem 6.5: For any CFL that *does not include λ*,
there exists a CFG *without useless, λ-, or unit-productions.*

Theorem 7.1: $\forall$CFL L, $\exists$ NDPA M s.t. $L(M) = L$.
For any CFL L, there exists a NDPA that accepts L.

Deciding Whether a CFL is Empty

- Theorem 8.6: Given a CFL G=(V, T, S, P), there exists an algorithm to decide if a context-free language L(G) = $\emptyset$.

Proof) For simplicity, assume that $\lambda \notin L(G)$

- Apply the algorithm for removing useless symbols and productions.
- If the start symbol, S, is found to be useless, then $L(G)$ is empty; otherwise, $L(G)$ contains at least one string.

Deciding Whether a CFL is Infinite

- Theorem 8.7: Given a CFL $G=(V, T, S, P)$, there exists an algorithm to decide if a CFL $L(G)$ is infinite.

Proof) Apply the algorithms for removing λ-productions, unit-productions, and useless productions.

- Suppose G has a repeating variable $A \in V$ for which there is a derivation $A \Rightarrow^{*1} xAy$. Since G has no λ-productions and no unit-productions, x and y can't be simultaneously empty.
- Since A is neither nullable nor a useless symbol, we have
  $S \Rightarrow^{*2} uAv \Rightarrow^{*} w$ and $A \Rightarrow^{*3} z$, where $u, v, z \in T^*$.
  But, then $S \Rightarrow^{*2} uAv \Rightarrow^{*1} ux^nAy^nv \Rightarrow^{*3} ux^nzy^nv$, $\forall n$ *is possible.*
  So, $L(G)$ is infinite.
- If no variable in G repeats, the length of any derivation is bounded by $|V|$. So, L(G) is finite.
- Note: The grammar has a repeating variable iff the dependency graph has a cycle as well as any variable that is at the base of a cycle is repeating variable. So, it can be an algorithm to decide if a grammar has a repeating variable to be used for the algorithm of infiniteness/finiteness.

Deciding Whether Two CFLs are Equal

- Given two CFGs G_1 and G_2, is there an algorithm to decide if $L(G_1) = L(G_2)$?
- If the languages are finite, the answer can be found by performing a string-by-string comparison.
- However, for general CFLs, *no algorithm exists to decide an equality of CFGs, thus an equality of CFLs.*