Non-context-Free Languages

The Pumping Lemma

Theorem. *(Pumping Lemma) Let L be context-free. There exists a positive integer p such that for every $w \in L$ of length at least p, w is divided into five pieces, $w = uvxyz$, such that*

- *for each $i \geq 0$, $uv^ixy^iz \in L$,*
- *$|vy| > 0$, and*
- *$|vxy| \leq p$.*

Proof Let $L = L(G)$ for some CNF grammar $G = (V, \Sigma, R, S)$. Let $m = \parallel V \parallel$ and $p = 2^m$. Let $w, |w| \geq p$, be in L and T be a derivation tree for w.

For any subtree R of T, its non-leaf nodes are all variables and its leaves are symbols with unique parents and form a substring of w.

Proof of Pumping Lemma (cont,d)

Claim. *In every subtree R of T with $\geq 2^{m-1}+1$ leaves there are two nodes α and β that are labeled by the same variable and are on the same downward path from the root to a leaf.*

Proof of Claim Let R be a subtree of T with $\geq 2^{m-1}+1$ leaves. Since the complete binary tree of depth $m-1$ has 2^{m-1} leaves, R has a **downward path of length $\geq m$.** The path has $\geq m+1$ **nodes.** Since there are **only m variables,** by the pigeon hole principle, the path has **two nodes with the same label.** ■Claim

consisting of nodes
labeled by variables

Proof of Pumping Lemma (cont,d)

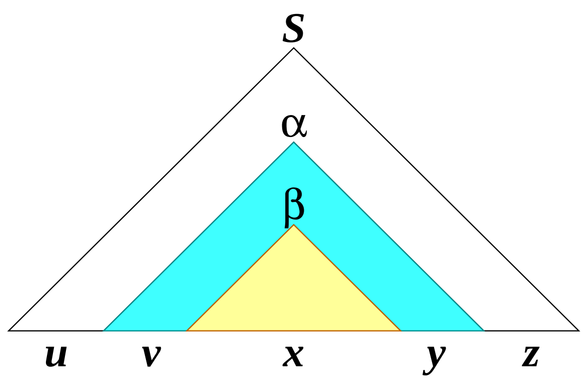
By Claim there is a node in T whose label coincides with that of a descendant. Let α be one such node that is **the farthest from the root.**

Here **neither the left subtree nor the right subtree of α has more than 2^{m-1} leaves**; otherwise, by claim we would find, in one of the two subtrees, a pair of nodes on a downward path labeled by the same variable, which would contradict our assumption that α is the farthest.

Proof of Pumping Lemma (cont,d)

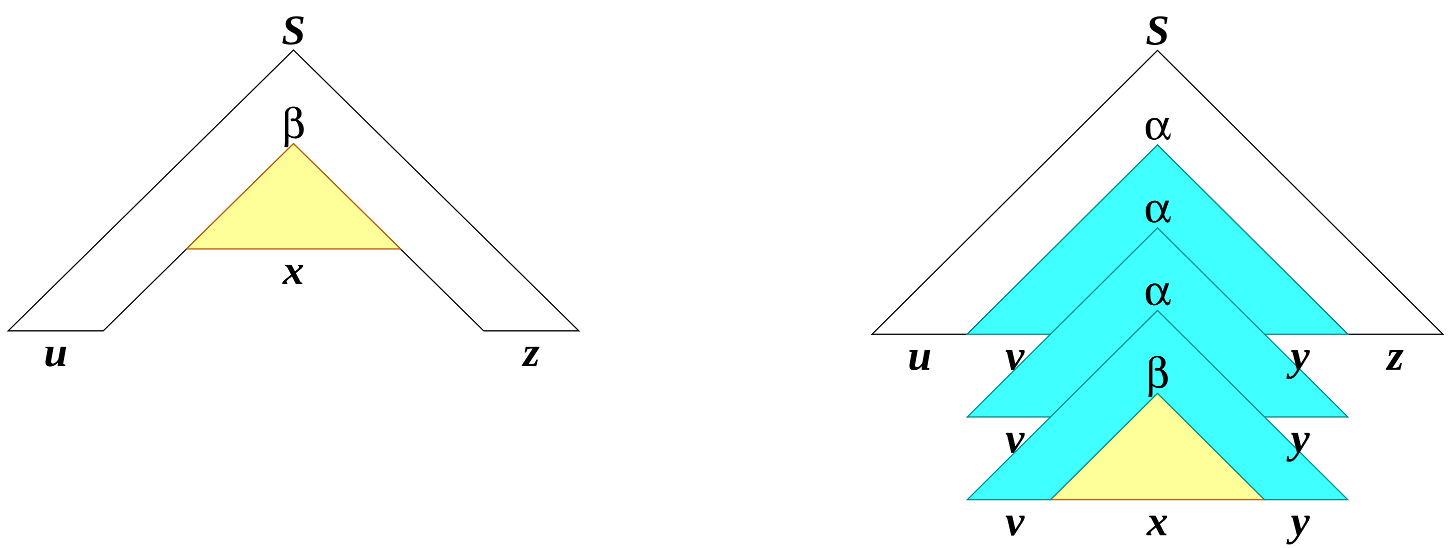
Let β be the descendant of α with the same label as α. **Replacing α by β** as well as **repeatedly β by α** produces **a valid derivation tree.**

Let x be the substring of β, $r = vxy$ the substring of α, with v and y to the left and to the right of x, respectively, and $w = urz$ with u and z to the left and to the right of r, respectively.



Proof of Pumping Lemma (cont,d)

Then replacing α by β corresponds to eliminating v and y and replacing β by α corresponds to inserting a v before x and a y after x. So, for every $i \geq 0$, $uv^ixy^iz \in L$.



Proof of Pumping Lemma (cont,d)

Since G does not have ϵ rules either v or y is nonempty, so $|vy| > 0$. Since both left and right subtrees of α have at most 2^{m-1} leaves, α has at most 2^m leaves, thus $|vxy| \leq p$. This proves the lemma. ∎

Example 1

$A = \{0^n 1^n 2^n \mid n \geq 0\}$ is not context free.

Proof Assume, to the contrary, that A is context free. By Pumping Lemma there exists a constant p such that every $w \in A$ of length $\geq p$ is divided into $w = uvxyz$ such that $|vxy| \leq p$, $|vy| \geq 1$, and for every $i \geq 0$, $uv^i xy^i z \in A$.

Let $w = 0^p 1^p 2^p$. Since $|vxy| \leq p$, vxy is either in $0^* 1^*$ or $1^* 2^*$. So it is not the case $uv^2 xy^2 z$ has the same number of 0s, 1s, as 2s. ∎

Example 2

$B = \{a\#b\#c \mid a, b$ and c are binary numbers such that $a + b = c\}$ is not context free.

Proof Assume, to the contrary, that B is context free. Let p be the constant from Pumping Lemma for B. Let $w = 10^p\#10^p\#10^{p+1}$, where $a = b = 2^p$ and $c = 2^{p+1}$. Let $uvxyz$ be the decomposition of w as in the lemma.

For "pumping" to be possible, v has to be a nonempty part of a or that of b and y a nonempty part of c. If v either is a part of a or contains the '1' of b, since $|vxy| \le p$, y cannot contain a part of c. Thus, v is a part of b and $v \in 0^*$.

Proof Continued

If y contains the first symbol of c, then uxz is not in B because now c is 0 while $a = 2^p$.

If $y \in 0^*$, then $uv^2xy^2z \notin B$ because now the equation becomes $2^p + 2^q = 2^r$ for some $q > p$.

Thus, B is not context-free. ∎

Example 3

$C = \{ww \mid w \in \{0,1\}^*\}$ is not context free.

Proof Assume C is context free. Let p the constant from the pumping lemma for C.

Let $w = 0^p1^p0^p1^p$, which is in C.

Let $w = uvxyz$ be the decomposition of w such that $|vy| > 0$, $|vxy| \leq p$, and for every $i \geq 0$, $uv^ixy^iz \in C$.

If v contains a symbol from the first 0^p then y cannot contain one from the second 0^p, so pumping doesn't work. If v contains only symbols from the first 1^p then y cannot contain one from the second 1^p, so pumping doesn't work. If v contains only symbols from the second 0^p1^p then pumping does not work. ∎

Application

Corollary. *The class of context-free languages is not closed under intersection.*

Proof Let $L_1 = \{0^i 1^j 2^k \mid i = j\}$ and $L_2 = \{0^i 1^j 2^k \mid j = k\}$. Then L_1 and L_2 are both context-free. If the class were closed under intersection then $L_1 \cap L_2 = \{0^n 1^n 2^n \mid n \geq 0\}$ were context-free. ∎

Corollary. *The class of context-free languages is not closed under complement.*

Closure Properties of CFL's

Consider a mapping

$$s : \Sigma \rightarrow 2^{\Delta^*}$$

In other words, we map a letter of Σ to a language over Δ

where Σ and Δ are finite alphabets. Let $w \in \Sigma^*$, where $w = a_1 a_2 \cdots a_n$, and define

$$s(a_1 a_2 \cdots a_n) = s(a_1).s(a_2).\cdots.s(a_n)$$

and, for $L \subseteq \Sigma^*$,

$$s(L) = \bigcup_{w \in L} s(w)$$

Such a mapping s is called a *substitution*.

Example: $\Sigma = \{0, 1\}, \Delta = \{a, b\}$,
$s(0) = \{a^n b^n : n \geq 1\}, s(1) = \{aa, bb\}$.

Let $w = 01$. Then $s(w) = s(0).s(1) =$
$\{a^n b^n aa : n \geq 1\} \cup \{a^n b^{n+2} : n \geq 1\}$

Let $L = \{0\}^*$. Then $s(L) = (s(0))^* =$
$\{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \cdots a^{n_k} b^{n_k} : k \geq 0, n_i \geq 1\}$

Theorem 7.23: Let L be a CFL over Σ, and s a substitution, such that $s(a)$ is a CFL, $\forall a \in \Sigma$. Then $s(L)$ is a CFL.

We start with grammars

$$G = (V, \Sigma, P, S)$$

for L, and

$$G_a = (V_a, T_a, P_a, S_a)$$

for each $s(a)$. We then construct

$$G' = (V', T', P', S\ )$$

where

$V' = (\bigcup_{a \in \Sigma} V_a) \cup V$

$T' = \bigcup_{a \in \Sigma} T_a$

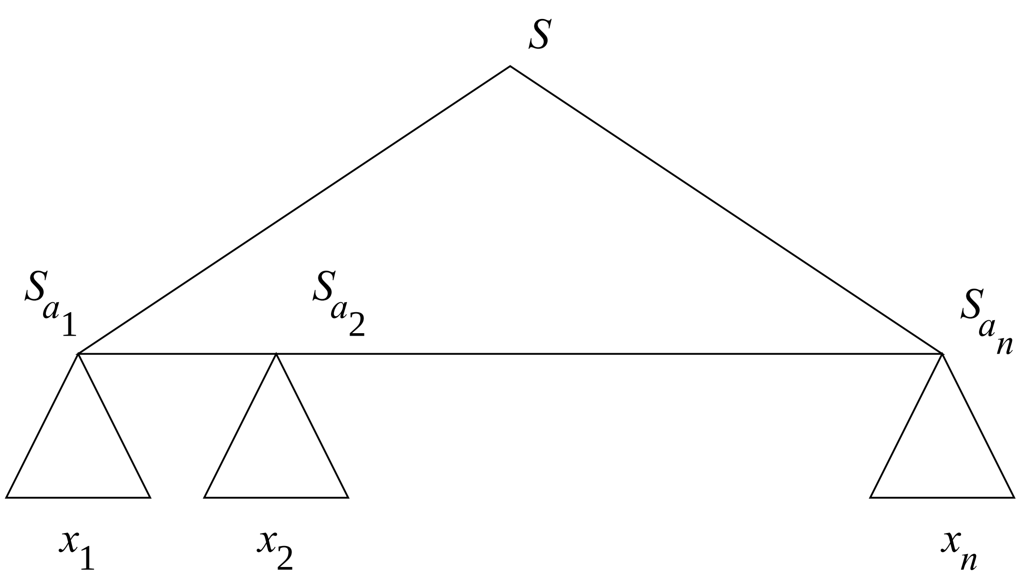
$P' = \bigcup_{a \in \Sigma} P_a$ plus the productions of P with each a in a body replaced with symbol S_a.

Now we have to show that

- $L(G') = s(L)$.

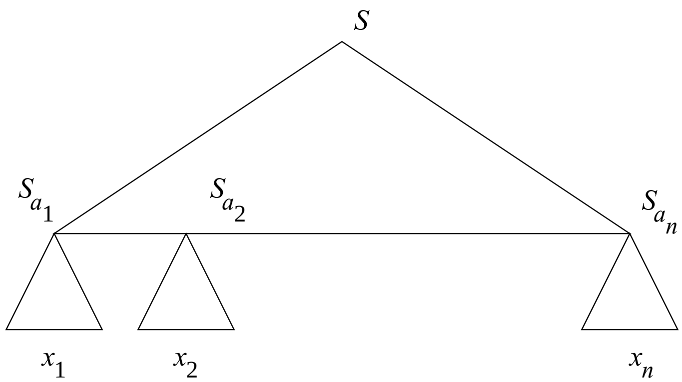
Let $w \in s(L)$. Then $\exists x = a_1 a_2 \cdots a_n$ in L, and $\exists x_i \in s(a_i)$, such that $w = x_1 x_2 \cdots x_n$.

A derivation tree in G' will look like



Thus we can generate $S_{a_1} S_{a_2} \cdots S_{a_n}$ in G' and from there we generate $x_1 x_2 \cdots x_n = w$. Thus $w \in L(G')$.

Then let $w \in L(G')$. Then the parse tree for w must again look like



Now delete the dangling subtrees. Then you have yield

$$S_{a_1} S_{a_2} \cdots S_{a_n}$$

where $a_1 a_2 \cdots a_n \in L(G)$. Now w belongs to $s(a_1 a_2 \cdots a_n)$, which is contained in *S(L)*.

Applications of the Substitution Theorem

Theorem 7.24: The CFL's are closed under (i) : union, (ii) : concatenation, (iii) : Kleene closure and positive closure $+$, and (iv) : homomorphism.

Proof: (i): Let L_1 and L_2 be CFL's, let $L = \{1, 2\}$, and $s(1) = L_1, s(2) = L_2$.
Then $L_1 \cup L_2 = s(L)$.

(ii) : Here we choose $L = \{12\}$ and s as before.
Then $L_1.L_2 = s(L)$

(iii) : Suppose L_1 is CF. Let $L = \{1\}^*, s(1) = L_1$. Now $L_1^* = s(L)$. Similar proof for $+$.

(iv) : Let L_1 be a CFL over Σ, and h a homomorphism on Σ. Then define s by

$$a \mapsto \{h(a)\}$$

Then $h(L_1) = s(L_1)$.

Theorem: If L is CF, then so is L^R.

Proof: Suppose L is generated b $G = (V, T, P, S)$. Construct $G^R = (V, T, P^R, S)$, where

$$P^R = \{A \rightarrow \alpha^R : A \rightarrow \alpha \in P\}$$

Show at home by inductions on the lengths of the derivations in G (for one direction) and in G^R (for the other direction) that $(L(G))^R = L(G^R)$.

Let $L_1 = \{0^n 1^n 2^i : n \geq 1, i \geq 1\}$. The L_1 is CF with grammar

$$S \rightarrow AB$$
$$A \rightarrow 0A1|01$$
$$B \rightarrow 2B|2$$

Also, $L_2 = \{0^i 1^n 2^n : n \geq 1, i \geq 1\}$ is CF with grammar

$$S \rightarrow AB$$
$$A \rightarrow 0A|0$$
$$B \rightarrow 1B2|12$$

However, $L_1 \cap L_2 = \{0^n 1^n 2^n : n \geq 1\}$ which is not CF (see the handout on course-page).

Theorem 7.27: If L is CF, and R regular, then $L \cap R$ is CF.

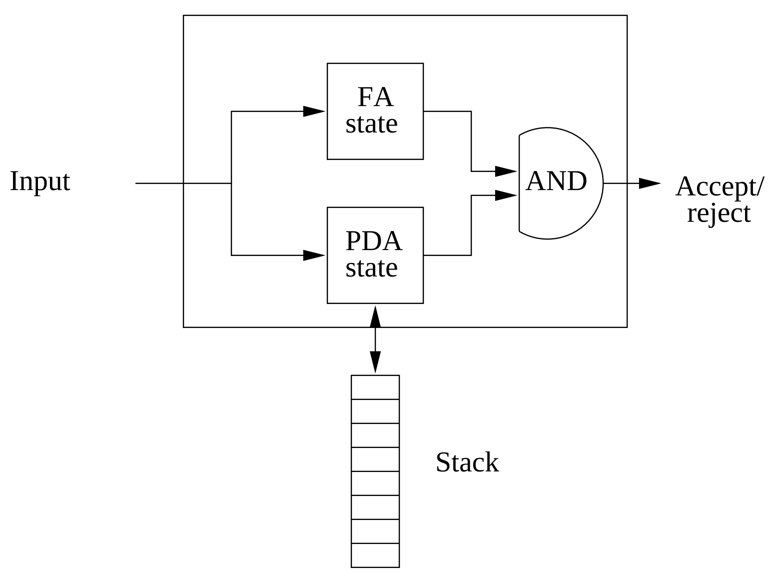
Proof: Let L be accepted by PDA

$$P = (Q_P, \Sigma, \Gamma, \delta_P, q_P, Z_0, F_P)$$

by final state, and let R be accepted by DFA

$$A = (Q_A, \Sigma, \delta_A, q_A, F_A)$$

We'll construct a PDA for $L \cap R$ according to the picture



Formally, define

$$P' = (Q_P \times Q_A, , \Sigma, \Gamma, \delta, (q_P, q_A), Z_0, F_P \times F_A)$$

where

$$\delta((q,p), a, X) = \{((r, \hat{\delta}_A(p,a)), \gamma) : (r,\gamma) \in \delta_P(q,a,X)\}$$

where a is in $\Gamma \cup \{\varepsilon\}$

Prove at home by an induction $\overset{*}{\vdash}$, both for P and for P' that

$$(q_P, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \gamma) \text{ in } P$$

if and only if

$$((q_P, q_A), w, Z_0) \overset{*}{\vdash} \Big((q, \hat{\delta}_A(q_A, w)), \epsilon, \gamma\Big) \text{ in } P'$$

The claim then follows (Why?)

Theorem 7.29: Let L, L_1, L_2 be CFL's and R regular. Then

1. $L \setminus R$ is CF
2. $\bar{L}$ is not necessarily CF
3. $L_1 \setminus L_2$ is not necessarily CF

Proof:

1. $\bar{R}$ is regular, $L \cap \bar{R}$ is CF, and $L \cap \bar{R} = L \setminus R$.
2. If $\bar{L}$ always was CF, it would follow that
$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$
always would be CF.

an example?
non-squares!

3. Note that Σ^* is CF, so if $L_1 \setminus L_2$ was always CF, then so would $\Sigma^* \setminus L = \bar{L}$.

Inverse homomorphism

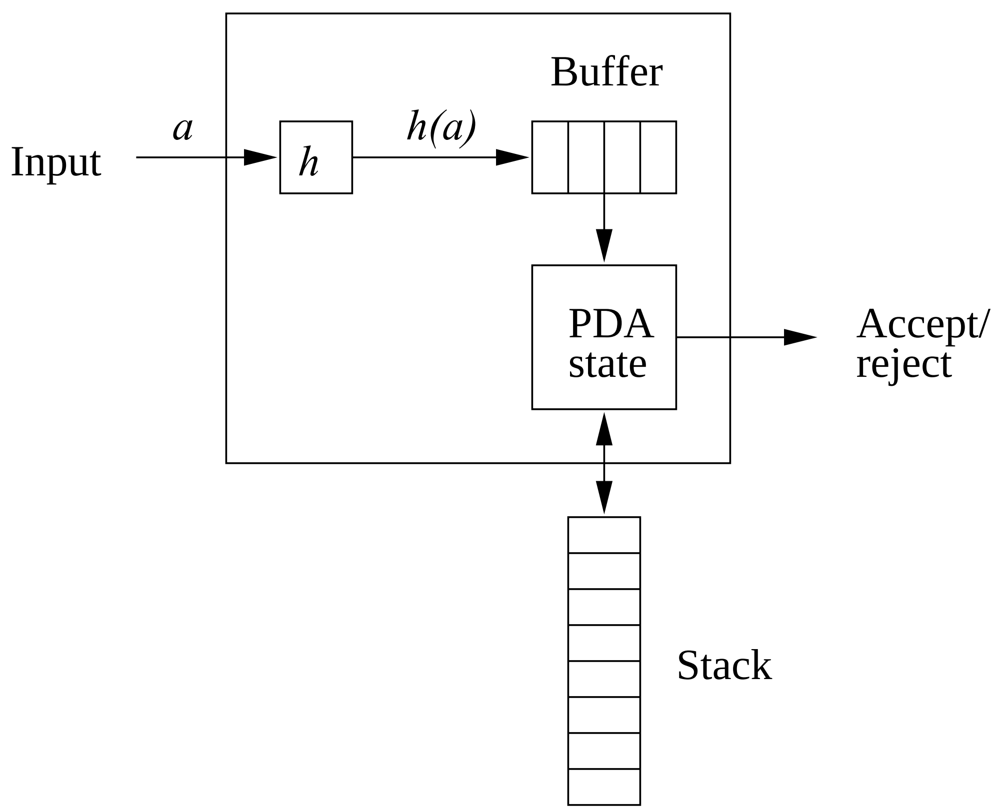
Let $h : \Sigma \to \Theta^*$ be a homom. Let $L \subseteq \Theta^*$, and define

$$h^{-1}(L) = \{w \in \Sigma^* : h(w) \in L\}$$

Now we have

Theorem 7.30: Let L be a CFL, and h a homomorphism. Then $h^{-1}(L)$ is a CFL.

Proof: The plan of the proof is



Let L be accepted by PDA

$$P = (Q, \Theta, \Gamma, \delta, q_0, Z_0, F)$$

We construct a new PDA

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

where

$$Q' = \{(q, x) : q \in Q, x \in \textit{suffix}(h(a)), a \in \Sigma\}$$

$$\delta'((q, \epsilon), a, X) = \{((q, h(a)), X) : \epsilon \neq a \in \Sigma, q \in Q, X \in \Gamma\}$$

$$\delta'((q, bx), \epsilon, X) = \{(((p, x), \gamma) : (p, \gamma) \in \delta(q, b, X), b \in \Sigma \cup \{\epsilon\}, q \in Q, X \in \Gamma\}$$

Show at home by suitable inductions that

- $(q_0, h(w), Z_0) \overset{*}{\vdash} (p, \epsilon, \gamma)$ in P if and only if $((q_0, \epsilon), w, Z_0) \overset{*}{\vdash} ((p, \epsilon), \epsilon, \gamma)$ in P'.

Note that h(ε) = ε.

Decision Properties of CFL's

We'll look at the following:

- Complexity of converting among CFG's and PDA 's

- Converting a CFG to CNF

- Testing $L(G) \neq \emptyset$, for a given G

- Testing $w \in L(G)$, for a given w and fixed G.

- Preview of undecidable CFL problems

Converting between CFG's and PDA's

- Input size is n.
- n is the *total* size of the input CFG or PDA.

The following work in time $O(n)$

1. Converting a CFG to a PDA (slide 203)
2. Converting a "final state" PDA to a "null stack" PDA (slide 199)
3. Converting a "null stack" PDA to a "final state" PDA (slide 195)

Avoidable exponential blow-up

For converting a PDA to a CFG we have

(slide 210)

At most n^3 variables of the form $[pXq]$

If $(\mathbf{r}, Y_1 Y_2 \cdots Y_k) \in \delta(\mathbf{q}, a, X)\}$, we'll have $O(n^n)$ rules of the form

$$[\mathbf{q}Xr_k] \rightarrow a[\mathbf{r}Y_1 r_1] \cdots [r_{k-1}Y_k r_k]$$

- By introducing $k-2$ new states we can modify the PDA to push at most *one* symbol per transition. Illustration on blackboard in class.

$(r_{Y2...Yk}, Y_1)$ is in $\delta(q,a,X)$
$(r_{Y3...Yk}, Y_2Y_1)$ is in $\delta(r_{Y2...Yk}, \varepsilon, Y_1)$
...

- Now, k will be ≤ 2 for all rules.

- Total length of all transitions is still $O(n)$.

- Now, each transition generates at most n^2 productions

- Total size (and time to calculate) the grammar is therefore $O(n^3)$.

Converting into CNF

Good news:

1. Computing $r(G)$ and $g(G)$ and eliminating useless symbols takes time $O(n)$. This will be shown shortly

   (slides 229,232,234)

2. Size of $u(G)$ and the resulting grammar with productions P_1 is $O(n^2)$

   (slides 244,245)

3. Arranging that bodies consist of only variables is $O(n)$

   (slide 248)

4. Breaking of bodies is $O(n)$ (slide 248)

Bad news:

- Eliminating the nullable symbols can make the new grammar have size $O(2^n)$

(slide 236)

The bad news are avoidable:

Break bodies first before eliminating nullable symbols

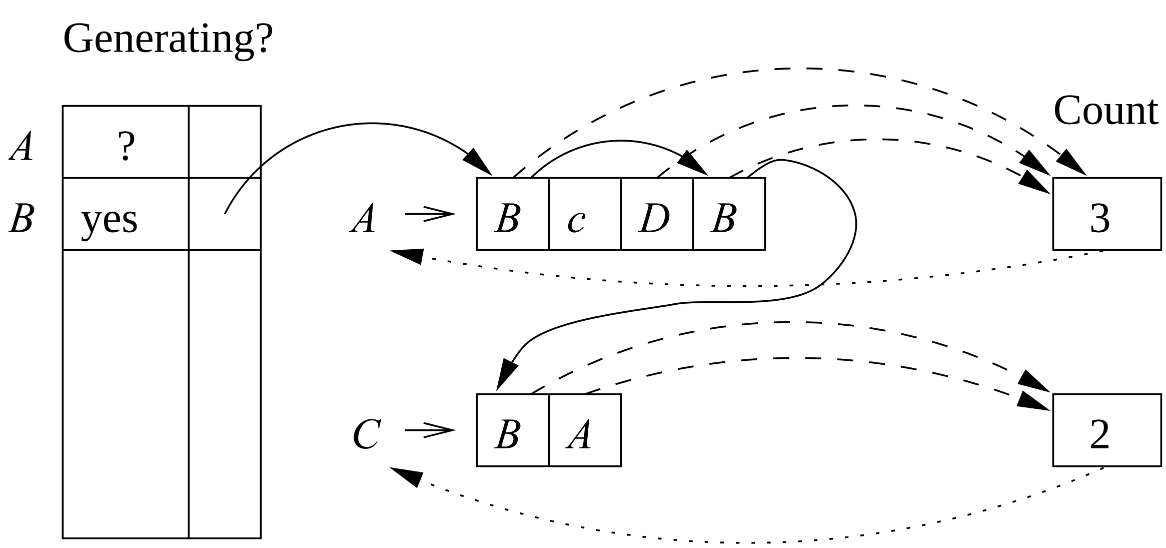
- Conversion into CNF is $O(n^2)$

Testing emptiness of CFL's

$L(G)$ is non-empty if the start symbol S is generating.

A naive implementation on $g(G)$ takes time $O(n^2)$.

$g(G)$ can be computed in time $O(n)$ as follows:



Creation and initialzation of the array is $O(n)$

Creation and initialzation of the links and counts is $O(n)$

When a count goes to zero, we have to

1. Finding the head variable A, checking if it already is "yes" in the array, and if not, queueing it is $O(1)$ per production. Total $O(n)$

2. Following links for A, and decreasing the counters. Takes time $O(n)$.

Total time is $O(n)$.

What if L is given as a PDA?

The membership question

$$w \in L(G)\textbf{?}$$

Inefficient way:

Suppose G is CNF, test string is w, with $|w| = n$. Since the parse tree is binary, there are $2n-1$ internal nodes.

Generate *all* binary parse trees of G with $2n-1$ internal nodes.

Check if any parse tree generates w

CYK-algo for membership testing

The grammar G is fixed

Input is $w = a_1 a_2 \cdots a_n$

We construct a triangular table, where X_{ij} contains all variables A, such that

$$A \overset{*}{\underset{G}{\Rightarrow}} a_i a_{i+1} \cdots a_j$$

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
a_1	a_2	a_3	a_4	a_5

To fill the table we work row-by-row, upwards

The first row is computed in the basis, the subsequent ones in the induction.

Basis: $X_{ii} == \{A : A \rightarrow a_i$ is in $G\}$

Induction:

We wish to compute X_{ij}, which is in row $j - i + 1$.

$A \in X_{ij}$, if
$A \overset{*}{\Rightarrow} a_i a_i + 1 \cdots a_j$, if
for some $k < j$, and $A \rightarrow BC$, we have
$B \overset{*}{\Rightarrow} a_i a_{i+1} \cdots a_k$, and $C \overset{*}{\Rightarrow} a_{k+1} a_{k+2} \cdots a_j$, if
$B \in X_{ik}$, and $C \in X_{(k+1)j}$

Example:

G has productions

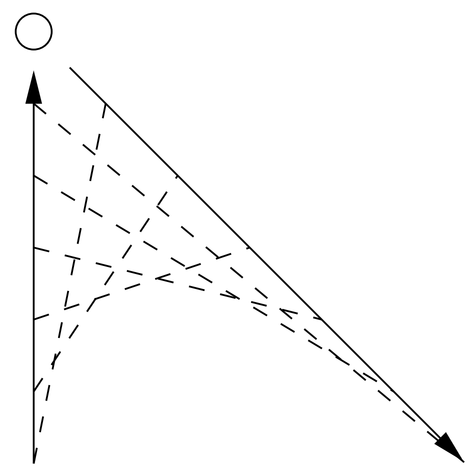
$$
\begin{aligned}
S &\rightarrow AB|BC \\
A &\rightarrow BA|a \\
B &\rightarrow CC|b \\
C &\rightarrow AB|a
\end{aligned}
$$

{S,A,C}				
-	{S,A,C}			
-	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	*a*	*a*	*b*	*a*

To compute X_{ij} we need to compare at most n pairs of previously computed sets:

$$(X_{ii}, X_{i\,+1,j}), (X_{i,i+1}, X_{i+2,j}), \ldots, (X_{i,j-1}, X_{jj})$$

as suggested below



For $w = a_1 \cdots a_n$, there are $O(n^2)$ entries X_{ij} to compute.

For each X_{ij} we need to compare at most n pairs $(X_{ik}, X_{k+1,j})$.

Total work is $O(n^3)$.

Preview of undecidable CFL problems

The following are undecidable:

1. Is a given CFG G ambiguous?

2. Is a given CFL inherently ambiguous?

3. Is the intersection of two CFL's empty?

4. Are two CFL's the same?

5. Is a given CFL universal (equal to Σ^*)?

∞