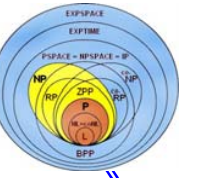
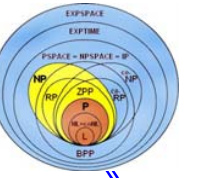
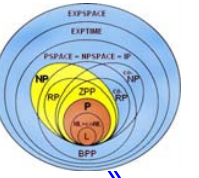


Session 11

- Pumping Lemma for Context-Free Languages
- Closure Properties of Context-Free Languages
- Decision Properties of Context-Free Languages



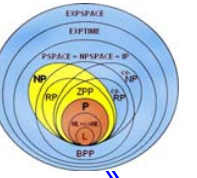
Pumping Lemma
for Context-Free Grammars



Now, we 'll develop tool for showing that certain language are not context-free.

The theorem, called the pumping lemma for context-free languages, says that in any sufficiently long string in a CFL, it is possible to find at most two short, nearby substrings, that we can "pump" in tandem.

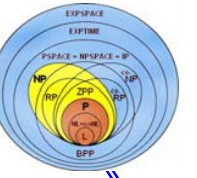
This theorem is similar to the pumping lemma for regular languages, which says we can always find one small string to pump.



Parse tree of a CFN grammar is a binary tree, which has some convenient properties, one of which deals with the shape and size of tree.

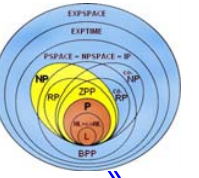
Theorem 7.9 *Suppose we have a parse tree according to a CFN grammar* $G = (V, T, P, S)$, *and suppose that the yield of the tree is a terminal string* w. *If the length of the longest path in* n, *then* $|w| \leq 2^{n-1}$.

Proof The proof ia a simple induction on n. ◄



Theorem 7.10 *Let L be a CFL. Then there exists a constant n such that if z is any string in L such that $|z| > n$, the we can write $z = uvwxy$, subject to the following conditions:*

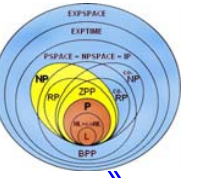
1. $|vwx| \leq n$. *Thai is, the middle portion is not too long.*
2. $|vx| > 0$. *That is, at least one of strings we pump must not be empty.*
3. $\forall\, i \geq 0$, $uv^iwx^iy \in L$. *That is, the two strings v and x may be "pumped" any number of times, including 0, the resulting string will still in L.*



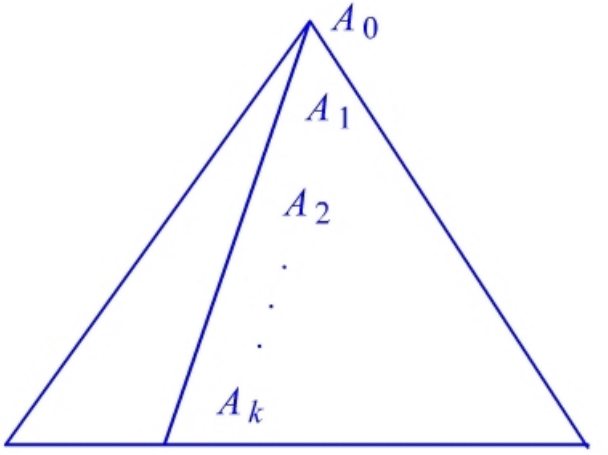
Proof Let $L \neq \emptyset$, $L \neq \{\epsilon\}$, we can find a CNF grammar G for L such that $L(G) = L/\{\epsilon\}$. Let G have m variables and choose $n = 2^m$. Suppose that z in L is of length at least n.

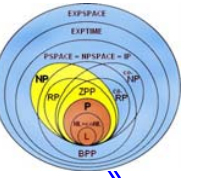
By the Theorem 7.9, any parse tree whose longest path is of length m or less must have a yield of length $2^{m-1} = n/2$ or less. Such a parse tree cannot have yield z, since it is too long.

Thus, any parse tree with yield z has a path of length at least $m + 1$.

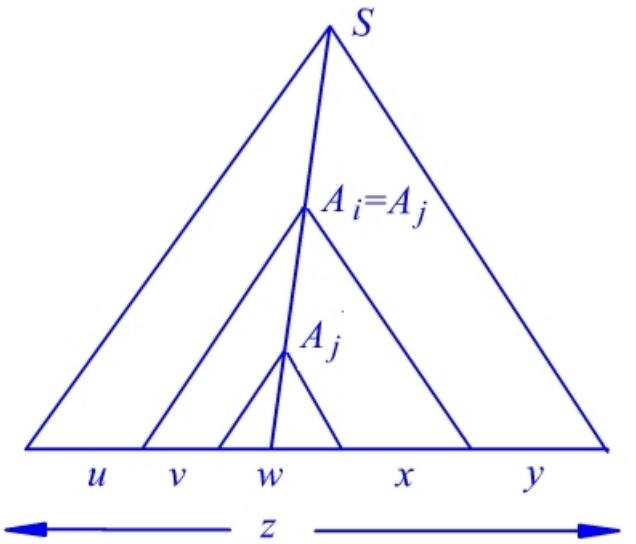


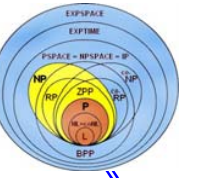
Now we have the longest path in the tree for z, there are at least $m + 1$ occurrences of variables $A_0, A_1, \cdots, A_k$ $(k \geq m)$ on the path.



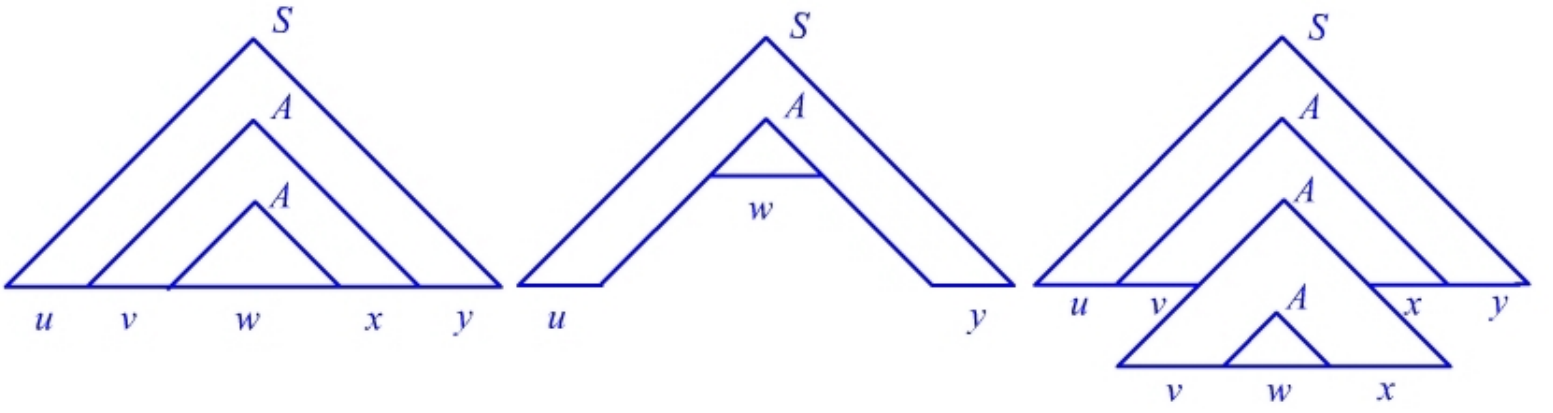


As there are only m different variables in V, there are at least two variables $A_i = A_j$, where $k - m \leq i < j \leq k$. We divide the tree as

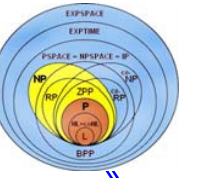




If $A_i = A_j = A$, then we can construct new parse trees from original tree. And we have proved the pumping lemma.



◀

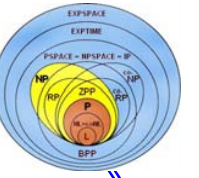


Applications of the Pumping Lemma for CFL's

Example Consider language $L = \{0^m 1^m 2^m \mid m \geq 1\}$.

Suppose L were context-free. Then there is an integer n given by pumping lemma. Pick $z = 0^n 1^n 2^n$ and break it as $z = uvwxy$, where $|vwx| \leq n$, and v and x are not both ϵ.

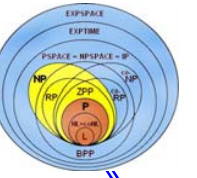
Then we know that vwx cannot involve both 0's and 2's, since the last 0 and first 2 are separated by $n + 1$ positions. In following two cases, we'll find a contradiction, thus L is not CFL.



1. vwx has no 2's. Then vx consists of only 0's and 1's, and has at least one of these symbols. By pumping lemma $uwy \in L$, but it is impossible since it has n 2's, and fewer than n 0's or 1's, or both.
2. vwx has no 0's. Similarly.

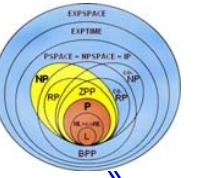
Example Let $L = \{ww \mid w \in \{0,1\}^*\}$. If L is context-free, then let n is pumping lemma constant. Consider the string $z = 0^n1^n0^n1^n$.

Break $z = uvwxy$, with $|vwx| \leq n$ and $|vx| > 0$. We'll show that $uwy \notin L$, and thus show L not to be CFL, by contradiction.



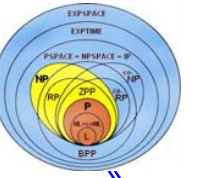
Since $|vwx| \leq n$, $|uwy| \geq 3n$. Thus, if uwy is repeating string, say tt, then t is of length at least $3n/2$. There are several cases to consider:

1. Suppose vwx is within the first n 0's. Let $|vx| = k$, then uwy begins with $0^{n-k}1^n$. Since $|uwy| = 4n - k$, we know that if $uwy = tt$, then $|t| = 2n - k/2$. Thus, t does not end until after the first block of 1's; i.e. t ends in 0. But uwy ends in 1, and so it cannot equal tt.



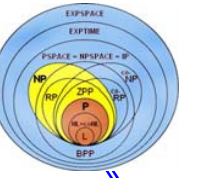
2. Suppose vwx straddles the first block of 0's and the first block of 1's. It may be that vx consists only of 0's, if $x = \epsilon$. Then, the argument that uwy is not of the form tt is the same as case (1). If vx has at least one 1, then we note that t, which is of length at least $3n/2$, must end in 1^n, because uwy ends in 1^n. However, these is no block of n 1's except the final block, so t cannot repeat in uwy.

3. If vwx is contained in the first block of 1's, then the argument that uwy is not in L is like the second part of case (2).



4. Suppose vwx straddles the first block of 1's and the second block of 0's. If vx actually has no 0's, then the argument is the same as if vwx were contained in the first block of 1's. If vx has at least one 0, then uwy starts with a block of n 0's, and so does t if $uwy = tt$. However, these is no other block of n 0's in uwy for the second copy t. We conclude in this case too, that uwy is not in L.

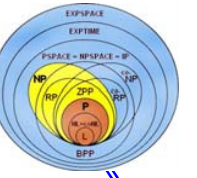
5. If the other cases, where vwx is the second half of z, the argument is symmetric to the cases where uwy is contained in the first first half of z.



Example Prove language $L = \{0^i 1^{i^2}\}$ not to be context-free.

Let n be the pumping lemma constant and consider $z = 0^n 1^{n^2}$. We break $z = uvwxy$ according to the pumping lemma. If vwx consists only of 0's, the uwy has n^2 1's and fewer that n 0's; it is not in the language.

If vwx has only 1's, then we derive a contradiction similarly. If either v or x has both 0's and 1's, then uv^2wx^2y is not in 0^*1^*, and thus could not be in the language.

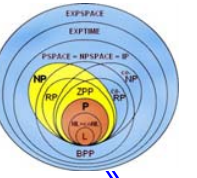


Finally, consider the case where v consists of 0's only, say k 0's, and x consists of m 1's only, where k and m are both positive.

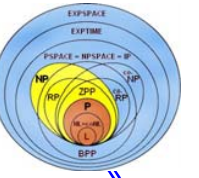
Then for all i, $uv^{i+1}wx^{i+1}y$ consists of $n + ik$ 0's and $n^2 + im$ 1's. If the number of 1's is always to be the square of the number of 0's, we must have, for some positive k and m:

$$(n + ik)^2 = n^2 + im, \quad \text{or} \quad 2ink + i^2k^2 = im.$$

But the left side grows quadratically in i, while the right side grows linearly, and so this equality for all i is impossible. We conclude that for at least some i, $uv^{i+1}wx^{i+1}y$ is not in the language and have thus derived a contradiction in all case.



Closure Properties of Context-Free Grammars



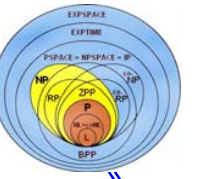
Substitutions

Consider a mapping $s : \Sigma \to 2^{\Delta^*}$ where Σ and Δ are finite alphabets. That is, for any $a \in \Sigma$, $s(a)$ is a language over Δ.

Let $w \in \Sigma^*$, where $w = a_1 a_2 \cdots a_n$, and define $s(w) = s(a_1).s(a_2).\cdots.s(a_n)$ $(s(\epsilon) = \{\epsilon\})$ and for $L \subseteq \Sigma^*$,

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

Such a mapping s is called a substitution.



Example Let $\Sigma = \{0, 1\}$ and $\Delta = \{a, b\}$, define

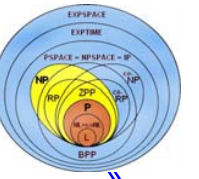
$$s(0) = \{a^n b^n \mid n \geq 1\}, \quad s(1) = \{aa, bb\}.$$

For $w = 01$, we have

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

Let $L = \{0\}^*$. Then

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

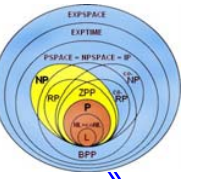


Theorem 7.11 *Let L be a CFL over Σ, and s a substitution, such that $s(a)$ is a CFL, $\forall\, a \in \Sigma$. Then $s(L)$ is a CFL.*

Proof 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 CFG $G' = (V', T', P', S)$ where

$$V' = V \cup \left(\bigcup_{a \in \Sigma} V_a \right), \quad T' = \bigcup_{a \in \Sigma} T_a.$$

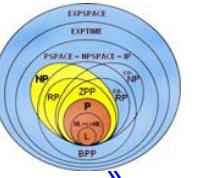


And P' consists of:

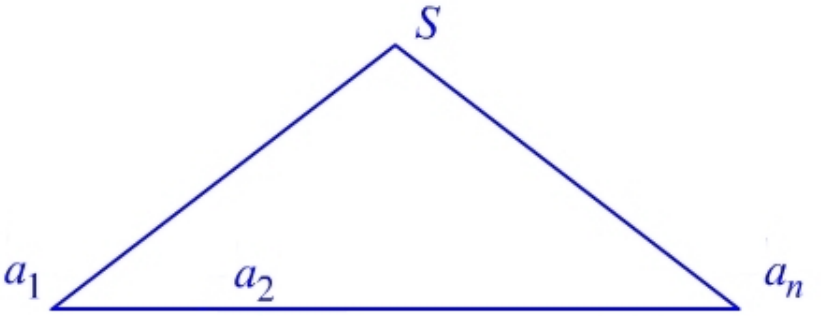
1. All productions in any P_a, for $a \in \Sigma$.
2. The productions of P, but with each terminal a in their bodies replaced by S_a everywhere a occurs.

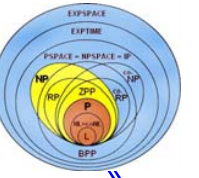
Now we have to show that $L(G') = s(L)$.

($\supseteq$-direction) Let $w \in s(L)$. Then $\exists\, y \in L$ such that $w \in s(y)$. Let $y = a_1 a_2 \cdots a_n$, $a_i \in \Sigma$, then $w \in s(a_1).s(a_2).\cdots.s(a_n)$. That is, $\exists\, x_i \in s(a_i)$, such that $w = x_1 x_2 \cdots x_n$.

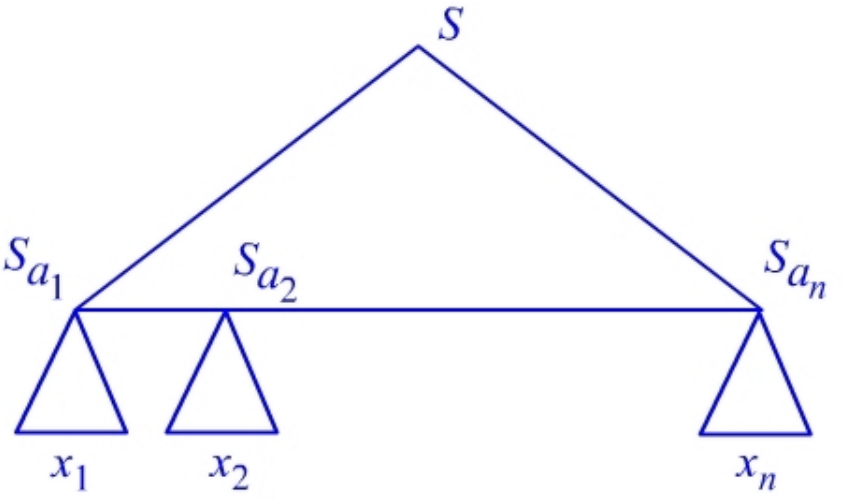


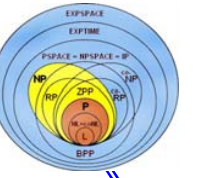
A derivation tree in G' will look like





A derivation tree in G' will look like





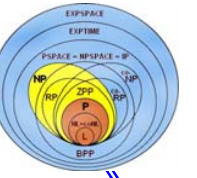
Thus we can generate $S_{a_1}S_{a_2}\cdots S_{a_n}$ in G' and from there we generate $x_1x_2\cdots x_n = w$. Thus $w \in L(G')$.

($\subseteq$-direction) Let $w \in L(G')$. Then the parse tree for w must again look like the figure above slide.

Now delete the dangling subtrees. Then we have yield

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

where $a_1a_2\cdots a_n \in L(G)$. Now w is in $s(a_1a_2\cdots a_n)$, which is in $s(L)$. ◀

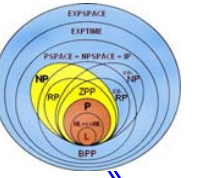


Applications of the Substitution Theorem

Theorem 7.12 *The CFL's are closed under the following operations:*

1. *Union.*
2. *Concatenation.*
3. *Closure (*), and positive closure (+).*
4. *Homomorphism.*

Proof Each requires only that we set up the proper substitution.



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

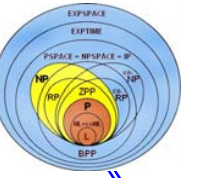
2. Here we choose $L = \{12\}$ and s as before. Then $L_1.L_2 = s(L)$.

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

4. Let L be a CFL over Σ, and h a homomorphism on Σ. Then define s by

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

Then $h(L) = s(L)$. ◀



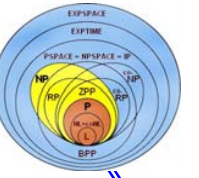
Other Closure Properties

Theorem 7.13 *If L is CFL, then so is L^R.*

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

$$P^R = \{A \to \alpha^R \mid A \to \alpha \in P\}$$

It is an easy induction on the lengths of the derivations in G (for one direction) and in G^R (for the other direction) that $(L(G))^R = L(G^R)$. ◄



The CFL's are not closed under intersection. Here is a simple example.

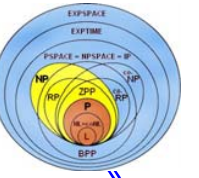
Let $L_1 = \{0^n 1^n 2^i \,|\, n \geq 1,\ i \geq 1\}$. The L_1 is a CFL with grammar

$$S \to AB,\ A \to 0A1 \,|\, 01,\ B \to 2B \,|\, 2$$

Also, $L_2 = \{0^i 1^n 2^n \,|\, n \geq 1,\ i \geq 1\}$ is a CFL with grammar

$$S \to AB,\ A \to 0A \,|\, 0,\ B \to 1B2 \,|\, 12$$

However $L_1 \cap L_2 = \{0^n 1^n 2^n \,|\, n \geq 1\}$ which is not a CFL.



Theorem 7.14 *If L_1 is CFL, and L_2 regular, then $L_1 \cap L_2$ is a CFL.*

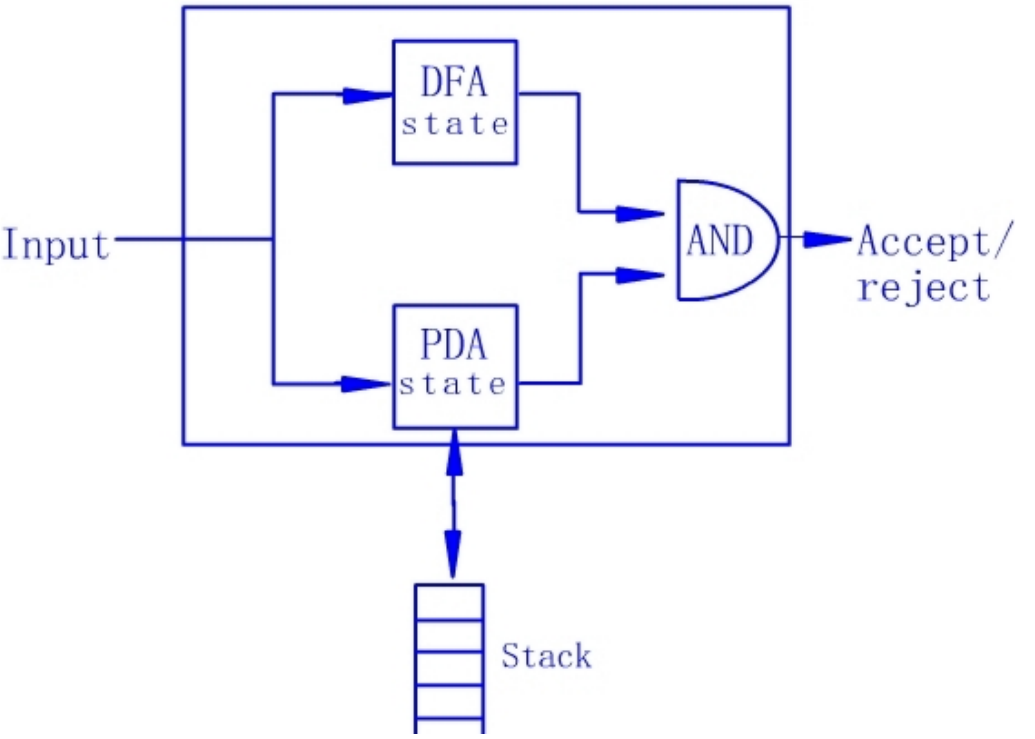
Proof Let L_1 be a accepted by PDA

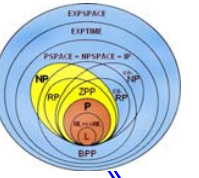
$$P = (Q_P, \Sigma, \delta_P, q_{0P}, Z_0, F_P)$$

by final state, i.e. $L_1 = L(P)$, and let $L_2 = L(A)$, where A is a DFA:

$$A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$$

We'll construct a PDA P' for $L_1 \cap L_2$ according to the picture showed in next slide.


DFA
state
Input
AND
Accept/
reject
PDA
state
Stack



Formally, define

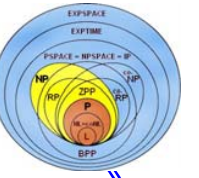
$$P' = (Q_P \times Q_A, \Sigma, \Gamma, \delta, (q_{0P}, q_{0A}), Z_0, F_P \times F_A)$$

where

$$\delta((q, p), a, X) = \{((r, s), \gamma) \mid s = \delta_A(p, a),\ (r, \gamma) \in \delta_P(q, a, X)\}$$

That is, for each move of P, we can make the same move in P', and in addition, we carry along the state of the A in a second component of state of P'.

Note that a may be a symbol of Σ, or $a = \epsilon$. A does not changes state while P makes moves on ϵ input.



We'll prove $L_1 \cap L_2 = L(P')$.

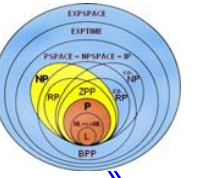
In fact, by an induction on the numbers of moves $\overset{*}{\vdash}$, we can prove, for any $q_P \in Q_P$ and $q_A \in Q_A$,

$$(q_P, w, X) \overset{*}{\underset{P}{\vdash}} (q, \epsilon, \gamma) \quad \text{and} \quad p = \hat{\delta}_A(q_A, w)$$

if and only if

$$((q_P, q_A), w, X) \overset{*}{\underset{P'}{\vdash}} ((q, p), \epsilon, \gamma)$$

If we do, then $w \in L_1 \cap L_2 = L(P) \cap L(A)$, i.e. from the start states q_{0P} and q_{0A}, and $X = Z_0$, $q \in F_P$ and $p \in F_A$, or $(q, p) \in F_P \times F_A$ is equivalent to $w \in L(P')$.



Basis step: One move. That is, $w = a$ (a may be ϵ).

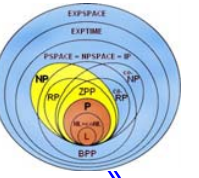
Since $(q_P, a, X) \vdash (q, \epsilon, \gamma)$, i.e. $(q, \gamma) \in \delta_P(q_P, a, X)$, and $p = \delta_A(q_A, a)$, then

$$((q, p), \gamma) \in \delta((q_P, q_A), a, X)$$

Thus $((q_P, q_A), a, X) \vdash ((p, q), \epsilon, \gamma)$. And vice verse.

Inductive step: $n + 1$ moves. Suppose the induction hypothesis holds for n moves.

Let $w = ax$. Then the first move in $(q_P, w, X) \vdash (r, x, \beta) \overset{*}{\vdash} (q, \epsilon, \gamma)$ means $(r, \beta) \in \delta_P(q_P, a, X)$.



On the other hand, $p = \hat{\delta}_A(q_A, w) = \hat{\delta}_A(s, x)$ where $s = \delta_A(q_A, a)$.

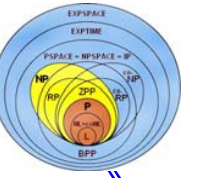
By the definition of δ, we have $((r, s), \beta) \in \delta((q_P, q_A), a, \gamma)$. That is

$$((q_P, q_A), a, \gamma) \vdash ((r, s), \epsilon, \beta), \quad \text{or} \quad ((q_P, q_A), ax, \gamma) \vdash ((r, s), x, \beta)$$

By the induction hypothesis, $((r, s), x, \beta) \overset{*}{\vdash} ((q, p), \epsilon, \gamma)$. Thus,

$$((q_P, q_A), ax, \gamma) \overset{*}{\vdash} ((q, p), \epsilon, \gamma).$$

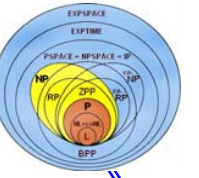
And vice verse. ◄



Theorem 7.15 *Let L, L_1, L_2 be CFL's and R regular. Then*

1. *$L - R$ is a CFL.*
2. *$\overline{L}$ is not necessarily CFL.*
3. *$L_1 - L_2$ is not necessarily CFL.*

Proof 1. Note that $L - R = L \cap \overline{R}$. If R is regular, so is $\overline{R}$ regular. Then $L - R$ is a CFL by Theorem 7.14.

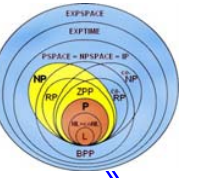


2. Suppose that $\overline{L}$ was always context-free when L is. It would follows that

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

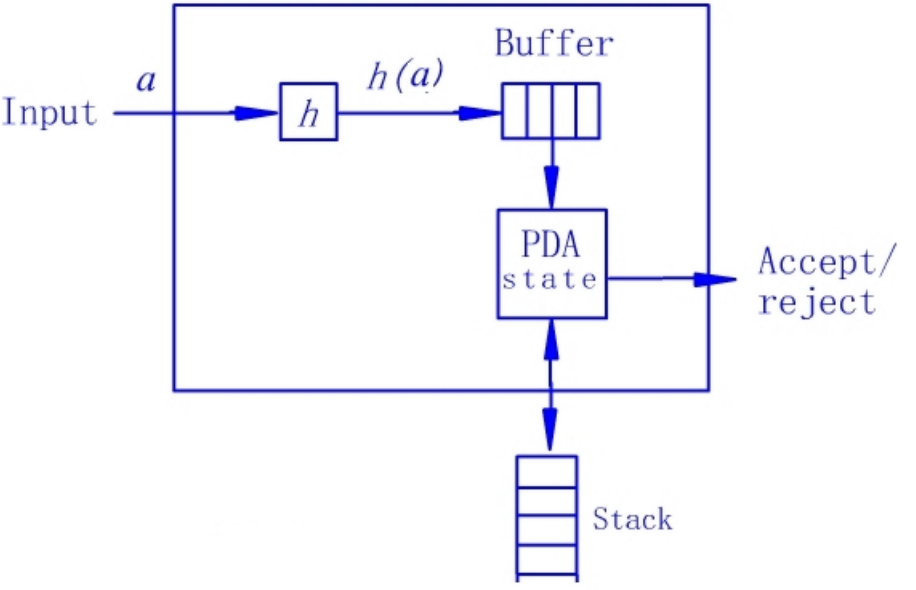
always would be CFL. But we know that CFL's are not closed under intersection.

3. Note that Σ^* is a CFL for every alphabet Σ, so if $L_1 - L_2$ was always CFL when L_1 and L_2 are, then it would follow that $\Sigma^* - L = \overline{L}$ was always CFL when L is. Thus, we would contradict (2). ◄

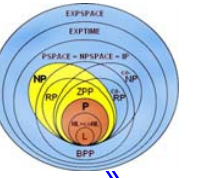


Theorem 7.16 *Let L be a CFL, and h a homomorphism. Then $h^{-1}(L)$ is a CFL.*

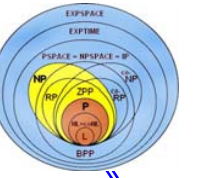
Proof We omit formal proof. The sketch is




BREAK FOR 15 MINUTES



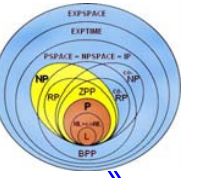
Decision Properties of Context-Free Grammars



Complexity of Conversions

Theorem 7.17 *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 7.18 *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 $O(n^2)$.*



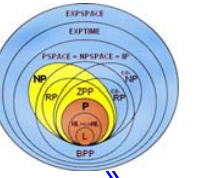
Testing Emptiness of CFL'S

Given a CFG G for the language L, using following algorithm we can decide whether the start symbol S of G is generating, i.e., whether S derives at least one string.

Basis step: $g(G) == T$.

Inductive step: If there is a $X \rightarrow \alpha \in P$ and every symbol of α is in $g(G)$, then $g(G) == g(G) \cup \{X\}$.

L is empty if and only if S is not generating.

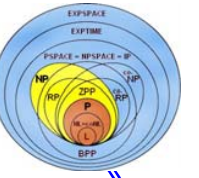


How much time it takes to find all the generating symbols of a grammar G?

Suppose the length of G is n. Then there could be on the order of n variables, and each pass of the inductive discovery of generating variables could take $O(n)$ time to examine all the productions of G.

If only one new generating variable is discovered on each pass, then there could be $O(n)$ passes. Thus, a native implementation of the generating-symbols test is $O(n^2)$.

☞ There is a more careful algorithm that sets up a data structure in advance to make our discovery of generating symbols take $O(n)$ time only.

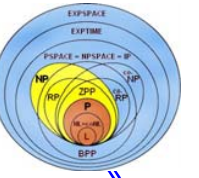


Testing Membership in a CFL

We can also decide membership of a string w in a CFL L. There are several inefficient ways to make the test; they take time that is exponential in $|w|$, assuming a grammar or PDF for the language L is given and its size is treated as a constant.

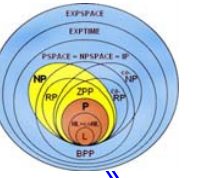
However, there is a much more efficient technique, which may also be known as a "tabulation" or CYK Algorithm.

This algorithm starts with a CNF grammar $G = (V, T, P, S)$ for L. The input is $w = a_1 a_2 \cdots a_n$ in T^*. In $O(n^3)$ time, it construct a table that tells whether w is in L.



In the CYK algorithm, we construct a triangular table as follows. The horizontal axis corresponds to the positions of the string $w = a_1 a_2 \cdots a_n$. The table entry X_{ij} is the *set of variables* A such that $A \overset{*}{\Rightarrow} a_i a_{i+1} \cdots a_j$.

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

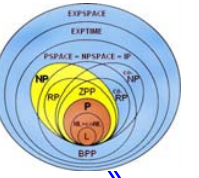


Note that we are interested in whether S is in the set X_{1n}, since that is the same as saying $S \stackrel{*}{\Rightarrow} w$, i.e., $w \in L$.

To fill the table, we work row-by-row, upwards. It takes $O(n)$ time to compute any one entry of the table, by a method we shall discuss next. Since there are $n(n+1)/2$ table entries, the whole table-construction process takes $O(n^3)$ time.

The algorithm for computing the X_{ij}'s (remember our G is in CNF):

Basis step: X_{ii} is the set of variables A such that $A \to a_i$ is a production of G.

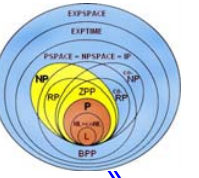


Inductive step: Suppose we want to compute X_{ij}, which is in row $j - i + 1$, and we have computed all the X's in the rows below.

In order for A to be in X_{ij}, we must find variables B and C, and integer k such that:

$$1.\ i \leq k < j \quad 2.\ B \in X_{ik} \quad 3.\ C \in X_{(k+1)j} \quad 4.\ A \to BC \in P$$

Finding such variables A requires us to compare at most n pairs of previously computed sets: $(X_{ii}, X_{(i+1)j})$, $(X_{i(i+1)}, X_{(i+2)j})$, and so on, until $(X_{i(j-1)}, X_{jj})$.



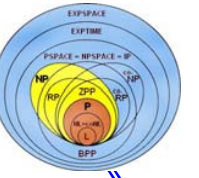
We summarize the algorithm above as a theorem.

Theorem 7.19 *The CYK algorithm correctly computes X_{ij} for all i and j; thus w is in $L(G)$ if and only if S is in X_{1n}. Moreover, the running time of the algorithm is $O(n^3)$.*

Example The following are the productions of a CNF grammar G:

$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$

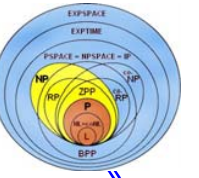
We shall test for membership in $L(G)$ the string *baaba*.



Now we'll show the table filled in for this string.

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}
b	a	a	b	a

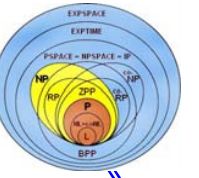
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
$\{B\}$	X_{22}	X_{33}	$\{B\}$	X_{55}
b	a	a	b	a

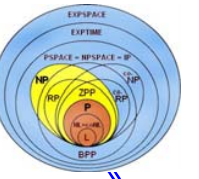
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	a	a	b	a

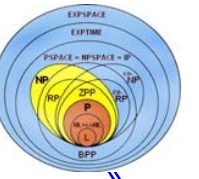
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
$\{S, A\}$	X_{23}	X_{34}	X_{45}	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

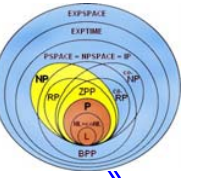
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
$\{S, A\}$	$\{B\}$	X_{34}	X_{45}	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

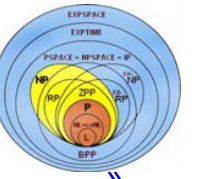
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	X_{45}	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

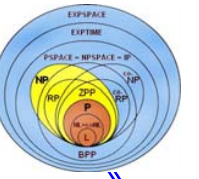
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

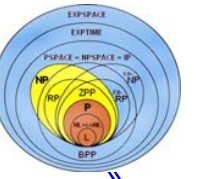
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

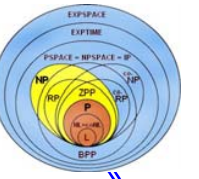
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
$--$	X_{24}	X_{35}		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

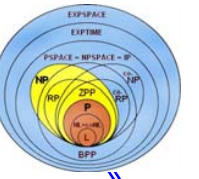
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
$--$	$\{B\}$	X_{35}		
$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$	
$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	a	a	b	a

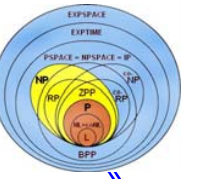
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
X_{14}	X_{25}			
$--$	$\{B\}$	$\{B\}$		
$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$	
$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	a	a	b	a

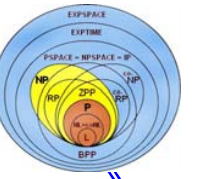
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
$--$	X_{25}			
$--$	$\{B\}$	$\{B\}$		
$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$	
$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	a	a	b	a

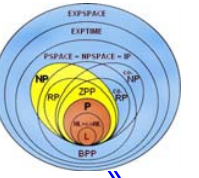
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

X_{15}				
$--$	$\{S,A,C\}$			
$--$	$\{B\}$	$\{B\}$		
$\{S,A\}$	$\{B\}$	$\{S,C\}$	$\{S,A\}$	
$\{B\}$	$\{A,C\}$	$\{A,C\}$	$\{B\}$	$\{A,C\}$
b	a	a	b	a

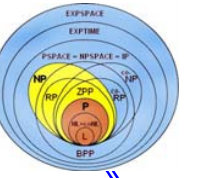
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Now we'll show the table filled in for this string.

$\{S, A, C\}$				
$--$	$\{S, A, C\}$			
$--$	$\{B\}$	$\{B\}$		
$\{S, A\}$	$\{B\}$	$\{S, C\}$	$\{S, A\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$
b	a	a	b	a

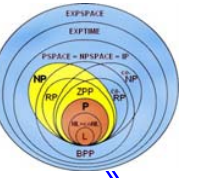
$$S \to AB \,|\, BC, \quad A \to BA \,|\, a, \quad B \to CC \,|\, b, \quad C \to AB \,|\, a$$



Some Undecidable Problems about CFL's

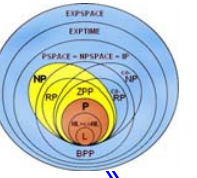
The following significant questions about CFG's and CFL's are undecidable:

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 equal to Σ^*, where Σ is the alphabet of this language?


EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
RP
ZPP
P
NP
RP
L
BPP

Thank you!


華東師範大學
EAST CHINA NORMAL UNIVERSITY


EXPSPACE
EXPTIME
PSPACE = NPSPACE = IP
NP
RP
ZPP
P
L
BPP


華東師範大學
EAST CHINA NORMAL UNIVERSITY