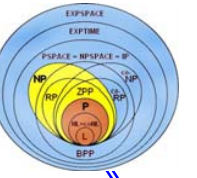
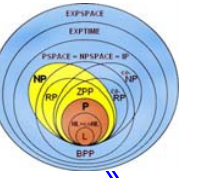


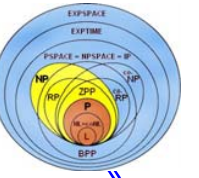
Session 10

- Normal Forms for Context-Free Grammars



Normal Forms

for Context-Free Grammars

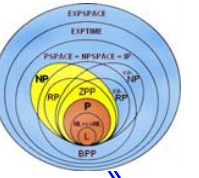


Simplifying CFG's makes life easier, since we can claim that if a language is context-free, then it has some grammar of special forms.

We want to show that every CFL (without ϵ) is generated by a CFG where all productions are of the form

$$A \to BC \quad \text{or} \quad A \to a$$

where A, B and C are variables, and a is a terminal. This is called Chomsky Normal Form (CNF).

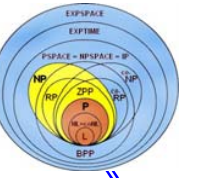


There is another interesting normal form for grammars that we shall not prove.

Every nonempty CFL without ϵ is $L(G)$ for some CFG each of whose productions are of the form

$$A \to a\,\alpha$$

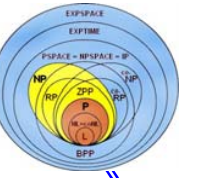
where a is a terminal and α is a string of zero or more variables. This is called Greibach Normal Form (GNF).



Simplification of CFG's

To get CNF or GNF of a CFG we have to

1. Eliminate useless symbols, those that do not appear in any derivation $S \overset{*}{\Rightarrow} w$, for start symbol S and terminal w.
2. Eliminate ϵ-productions, that is, production of the form $A \to \epsilon$.
3. Eliminate unit productions, that is, productions of the form $A \to B$, where A and B are variables.



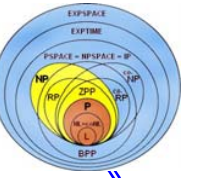
Eliminating Useless Symbols

- A symbol X is useful for a grammar $G = (V, T, P, S)$, if there is a derivation

$$S \overset{*}{\underset{G}{\Rightarrow}} \alpha X \beta \overset{*}{\underset{G}{\Rightarrow}} w$$

for a terminal string w. X may be in either V and T. Symbols that are not useful are called useless.

Evidently, omitting useless symbols from a grammar will not change the language generated.

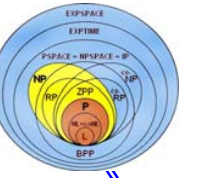


Our approach to eliminating useless symbols begins by identifying the two things a symbol has to be able to do be useful:

- A symbol X is generating if $X \overset{*}{\underset{G}{\Rightarrow}} w$, for some $w \in T^*$.
- A symbol X is reachable if $S \overset{*}{\underset{G}{\Rightarrow}} \alpha X \beta$, for some $\{\alpha, \beta\} \subseteq (V \cup T)^*$.

Note that, every terminal is generating; the start variable is reachable.

Surely, a symbol that is useful will be both generating and reachable.



It turns out that if we eliminate non-generating symbols first, and then non-reachable ones, we will left with only useful symbols.

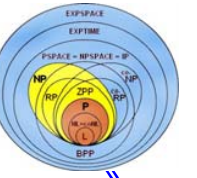
Example Let G be

$$S \to AB \,|\, a,\ A \to b.$$

S and A as well as a, b are generating, B is not.

If we eliminate B we have to eliminate $S \to AB$, leaving grammar $S \to a,\ A \to b$.

Now only S and a is reachable. Eliminating A and b leaves us with $S \to a$ with language $\{a\}$.



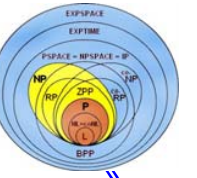
Notice that we have to order the two steps properly or the result might have useless symbols.

In above example, if we start checking for reachability first, we find that all symbols are reachable.

From

$$S \to AB \,|\, a,\ A \to b$$

we then eliminate B as non-generating, and left with $S \to a,\ A \to b$ that still contains useless symbols.

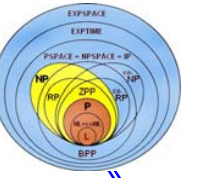


Theorem 7.1 *If $G = (V, T, P, S)$ be a CFG s.t. $L(G) \neq \emptyset$. Let $G_1 = (V_1, T_1, P_1, S)$ be the grammar obtained by*

1. *Eliminating all nongenerating symbols and the productions they occur in. Let the new grammar be $G_2 = (V_2, T_2, P_2, S)$.*
2. *Eliminating from G_2 all nonreachable symbols and the productions they occur in.*

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

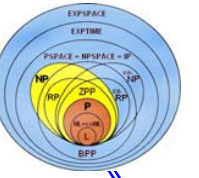
Proof We first prove that G_1 has no useless symbols.



Let X remain in $V_1 \cup T_1$. Since X was not eliminated in step 1, thus $X \overset{*}{\Rightarrow} w$ in G, for some $w \in T^*$. Moreover, every symbol used in this derivation is also generating. Thus $X \overset{*}{\Rightarrow} w$ in G_2 also.

Since X was not eliminated in step 2, there are α and β, such that $S \overset{*}{\Rightarrow} \alpha X \beta$ in G_2. Furthermore, every symbol used in this derivation is also reachable, so $S \overset{*}{\Rightarrow} \alpha X \beta$ in G_1.

Now every symbol in $\alpha X \beta$ is reachable and in $V_2 \cup T_2$ ($\supseteq V_1 \cup T_1$), so each of them is generating in G_2. The terminal derivation $\alpha X \beta \overset{*}{\Rightarrow} xwy$ in G_2 involves only symbols that are reachable from S, because they are reached by symbols in $\alpha X \beta$.



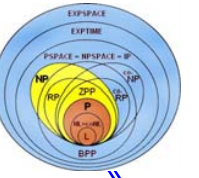
Thus the terminal derivation is also a derivation of G_1, i.e.

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} xwy$$

in G_1.

We then show that $L(G_1) = L(G)$. Since $P_1 \subseteq P$, we have $L(G_1) \subseteq L(G)$.

Then, let $w \in L(G)$. Thus $S \overset{*}{\underset{G}{\Rightarrow}} w$. Each symbol in this derivation is evidently both reachable and generating, so this is also a derivation of G_1. Thus $w \in L(G_1)$. ◄



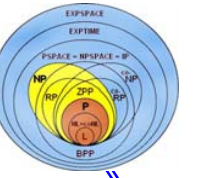
Computing the Generating and Reachable Symbols

We have to give algorithms to compute the generating and reachable symbols of $G = (V, T, P, S)$.

The set of generating symbols $g(G)$ is computed by the following closure algorithm:

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

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



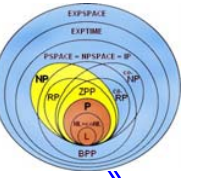
Example Let G be

$$S \to AB \,|\, a,\ A \to b.$$

Then first $g(G) == \{a, b\}$. Since $S \to a$ we put S in $g(G)$, and because $A \to b$ we add A also, and that's it.

Theorem 7.2 *At saturation, g(G) contains all and only the generating symbols of G.*

Proof It is an easy induction on the stage in which a symbol X is added to $g(G)$ than X is indeed generating.



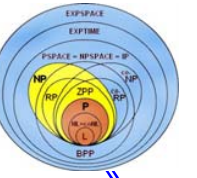
Then, suppose that X is generating. Thus $X \overset{*}{\underset{G}{\Rightarrow}} w$, for some $w \in T^*$. We prove by induction on this derivation that $X \in g(G)$.

Basis step: Zero steps. Then X is added in the basis of closure algorithm.

Inductive step: The derivation takes $n > 0$ steps. Let the first production used be $X \to \alpha$. Then

$$X \Rightarrow \alpha \overset{*}{\Rightarrow} w$$

and $\alpha \overset{*}{\Rightarrow} w$ in less than n steps and by the induction hypothesis $\alpha \in g(G)$. From the inductive part of the closure algorithm it follows that $X \in g(G)$. ◄



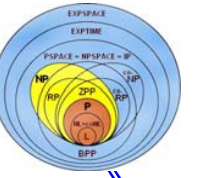
The set of reachable symbols $r(G)$ of $G = (V, T, P, S)$ is computed by the following closure algorithm:

Basis step: $r(G) == \{S\}$.

Inductive step: If $A \in r(G)$ and $A \to \alpha \in P$, then add all symbol in α to $r(G)$.

Example Let G be $S \to AB \mid a$, $A \to b$. Then first $r(G) == \{S\}$. Based on the first and second production we add $\{A, B, a\}$ and $\{b\}$ to $r(G)$, respectively, and that's it.

Theorem 7.3 *At saturation, r(G) contains all and only the reachable symbols of G.* ◀

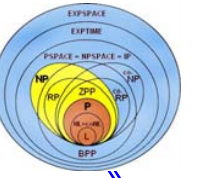


Eliminating ϵ-Productions

We shall prove that if L is context-free, then $L/\{\epsilon\}$ has a grammar without ϵ-productions.

- Variable A is said to be nullable if $A \stackrel{*}{\Rightarrow} \epsilon$.

Let A be nullable. We'll the replace a rule like $A \rightarrow BAD$ with two productions $A \rightarrow BAD$, $A \rightarrow BD$ and delete any rules with body ϵ.



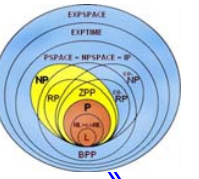
We'll compute $n(G)$, the set of nullable symbols of a grammar $G = (V, T, P, S)$ as follows:

Basis step: $n(G) == \{A \mid A \to \epsilon \in P\}$.

Inductive step: If $\{C_1, C_2, \cdots, C_k\} \subseteq n(G)$ and $A \to C_1 C_2 \cdots C_k \in P$, then $n(G) == n(G) \cup \{A\}$.

Theorem 7.4 *At saturation, n(G) contains all and only the nullable symbols of G.*

Proof Easy induction in both directions. ◄



Once we know the nullable symbols, we can transform G into G_1 as follows:

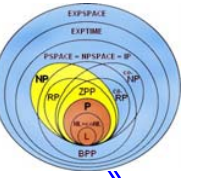
- For each $A \to X_1X_2\cdots X_k \in P$ with $m \le k$ nullable symbols X_i's, replace it by 2^m rules, one with each sublist of the nullable symbols absent.

  Exception: If $m = k$ we don't delete all m nullable symbols.

- Delete all rules of the form $A \to \epsilon$.

Example Eliminating ϵ-productions in G:

$$S \to AB,\ A \to aAA \mid \epsilon,\ B \to bBB \mid \epsilon$$



Now $n(G) = \{A, B, S\}$. The first rule will become

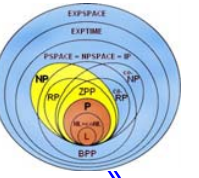
$$S \to AB \,|\, A \,|\, B$$

the second and third, respectively

$$A \to aAA \,|\, aA \,|\, aA \,|\, a, \quad B \to bBB \,|\, bB \,|\, bB \,|\, b$$

We then delete rules with ϵ-bodies, and end up with grammar G_1:

$$S \to AB \,|\, A \,|\, B,\ A \to aAA \,|\, aA \,|\, a,\ B \to bBB \,|\, bB \,|\, b$$



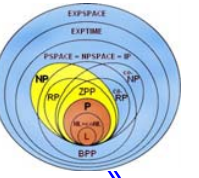
Theorem 7.5 *If the grammar G_1 is constructed from G by the above construction for eliminating ϵ-productions, then $L(G_1) = L(G)/\{\epsilon\}$.*

Proof We'll prove the stronger statement:

$$A \overset{*}{\Rightarrow} w \text{ in } G_1 \quad \text{iff} \quad w \neq \epsilon \text{ and } A \overset{*}{\Rightarrow} w \text{ in } G$$

The theorem follows by choosing $A = S$.

(Only if) Suppose $A \overset{*}{\Rightarrow} w$ in G_1. Then clearly $w \neq \epsilon$. We'll show by an induction on the length of the derivation that $A \overset{*}{\Rightarrow} w$ in G also.

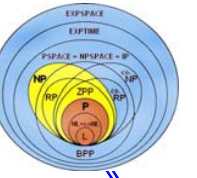


Basis step: One step. Then there exists $A \to w$ in G_1. From the construction of G_1 it follows that there exists $A \to \alpha$ in G, where α is w plus some nullable variables interspersed. Then $A \Rightarrow \alpha \overset{*}{\Rightarrow} w$ in G.

Inductive step: Derivation takes $n > 1$ steps. Then

$$A \Rightarrow X_1 X_2 \cdots X_k \overset{*}{\Rightarrow} w$$

in G_1 and the first derivation is based on production $A \to Y_1 Y_2 \cdots Y_m$, where $m \geq k$, some Y_i's are X_j's and the other are nullable symbols of G.



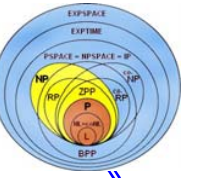
Furthermore, $w = w_1 w_2 \cdots w_k$, and $X_i \overset{*}{\Rightarrow} w_i$ in G_1 in less than n steps. By the induction hypothesis we have $X_i \overset{*}{\Rightarrow} w_i$ in G.

Now we get

$$A \underset{G}{\Rightarrow} Y_1 Y_2 \cdots Y_m \underset{G}{\overset{*}{\Rightarrow}} X_1 X_2 \cdots X_k \underset{G}{\overset{*}{\Rightarrow}} w_1 w_2 \cdots w_k = w.$$

(If) Let $A \overset{*}{\Rightarrow} w$ in G, and $w \neq \epsilon$. we'll show by induction of the length of the derivation that $A \overset{*}{\Rightarrow} w$ in G_1.

Basis step: Length is one. Then $A \to w$ is in G, and since $w \neq \epsilon$ the rule is in G_1 also.



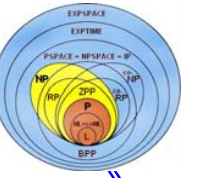
Inductive step: Derivation takes $n > 1$ steps. Then it looks like

$$A \underset{G}{\Rightarrow} Y_1 Y_2 \cdots Y_m \overset{*}{\underset{G}{\Rightarrow}} w.$$

Now $w = w_1 w_2 \cdots w_m$, and $Y_i \overset{*}{\Rightarrow} w_i$ in G in less than n steps. Let $X_1 X_2 \cdots X_k$ be those Y_j's in order, such that $w_j \neq \epsilon$. Then $A \to X_1 X_2 \cdots X_k$ is a rule in G_1.

Now $X_1 X_2 \cdots X_k \overset{*}{\Rightarrow} w$ in G_1. Each $X_j / Y_j \overset{*}{\Rightarrow} w_j$ in G in less than n steps, so by the induction hypothesis we have that if $w_j \neq \epsilon$ then $Y_j \overset{*}{\Rightarrow} w_j$ in G_1. Thus in G_1

$$A \Rightarrow X_1 X_2 \cdots X_k \overset{*}{\Rightarrow} w.$$



Eliminating Unit Production

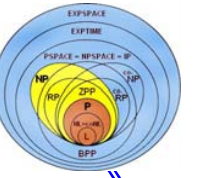
- A unit production is a production of the form $A \to B$, where both A and B are variables.

In CFG's, unit production can be eliminated.

Let's look at grammar

1. $I \to a|b|Ia|Ib|I0|I1$, 2. $F \to I|(E)$, 3. $T \to F|T \times F$, 4. $E \to T|E + T$

It has unit production $F \to I$, $T \to F$, $E \to T$.



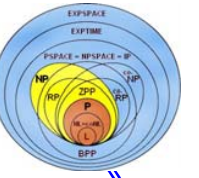
We'll expand rule $E \to T$ and get rules $E \to F \,|\, T \times F \,|\, E + T$.

We then expand $E \to F$ and get $E \to I \,|\, (E) \,|\, T \times F \,|\, E + T$.

Finally we expand $E \to I$ and get

$$E \to a \,|\, b \,|\, Ia \,|\, Ib \,|\, I0 \,|\, I1 \,|\, (E) \,|\, T \times F \,|\, E + T$$

The expansion method works as long as there are no cycles in the rules, as e.g. in $A \to B,\ B \to C,\ C \to A$. The following method based on unit pairs will work for all grammars.



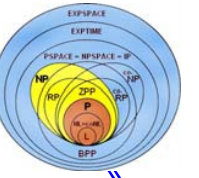
- A pair of variables (A, B) is a unit pair if $A \stackrel{*}{\Rightarrow} B$ using unit productions only.

☞ In $A \to BC$, $C \to \epsilon$ we have $A \stackrel{*}{\Rightarrow} B$, but not using unit productions only.

To compute $u(G)$, the set of all unit pairs of $G = (V, T, P, S)$ we use the following closure algorithm:

Basis step: $u(G) == \{(A, A) \,|\, A \in V\}$.

Inductive step: If $(A, B) \in u(G)$ and $B \to C \in P$, where C is a variable, then add (A, C) to $u(G)$.

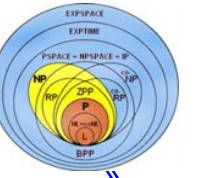


Theorem 7.6 *At saturation, u(G) contains all and only the unit pairs of G.*

Proof In one direction, it is an easy induction on the order in which the pairs are discovered, that if (A, B) is found to be unit pair, then $A \overset{*}{\underset{G}{\Rightarrow}} B$ using only unit productions.

In the other direction, suppose that $A \overset{*}{\underset{G}{\Rightarrow}} B$ using unit production only. We can show by induction on the length of the derivation that the pair (A, B) will be found.

Basis step: Zero steps. Then $A = B$, and the pair (A, B) is added in the basis.

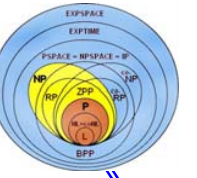


Inductive step: Suppose $A \stackrel{*}{\Rightarrow} B$ using $n+1$ steps, for some $n > 0$, each step being the application of a unit production. Then the derivation looks like

$$A \stackrel{*}{\Rightarrow} C \Rightarrow B$$

The derivation $A \stackrel{*}{\Rightarrow} C$ takes n steps, so by the induction hypothesis, we discover the pair (A, C). Then the inductive part of the algorithm combines the pair (A, C) with the production $C \to B$ to infer the pair (A, B).

◀

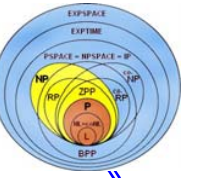


To eliminate unit productions, we proceed as follows. Given a CFG $G = (V, T, P, S)$, construct CFG $G_1 = (V, T, P_1, S)$:

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

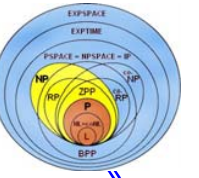
Example Eliminating unit productions from the following grammar

1. $I \to a|b|Ia|Ib|I0|I1$, 2. $F \to I|(E)$, 3. $T \to F|T \times F$, 4. $E \to T|E + T$



By the construction as above, we get

Pair	Productions
(E,E)	$E \to E+T$
(E,T)	$E \to T \times F$
(E,F)	$E \to (E)$
(E,I)	$E \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(T,T)	$T \to T \times F$
(T,F)	$T \to (E)$
(T,I)	$T \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(F,F)	$F \to (E)$
(F,I)	$F \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$
(I,I)	$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

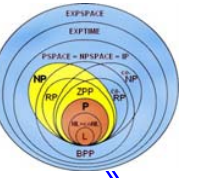


Theorem 7.7 *If the grammar G_1 is constructed from G by the algorithm described above for eliminating unit productions, then $L(G_1) = L(G)$.*

Proof (If) Suppose $S \overset{*}{\Rightarrow} w$ in G_1. Since every production of G_1 is equivalent to a sequence of zero or more unit productions of G followed by a nonunit production of G, we know that $\alpha \Rightarrow \beta$ in G_1 implies $\alpha \overset{*}{\underset{G}{\Rightarrow}} \beta$.

If we put these sequences of steps together, we conclude that $S \overset{*}{\underset{G}{\Rightarrow}} w$.

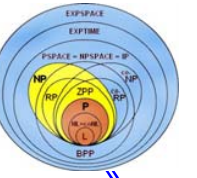
(Only If) Suppose now that $w \in L(G)$. Then $S \overset{*}{\underset{lm}{\Rightarrow}} w$.



Whenever a unit production is used in a leftmost derivation, the variable of the body becomes the leftmost variable, and so immediately replaced.

Thus, the leftmost derivation in grammar G can be broken into a sequence of steps in which zero or more unit productions are followed by a nonunit production. Note that any nonunit production that is not preceded by a unit production is a "step" by itself.

Each of these steps can be performed by one production of G_1, because the construction of G_1 created exactly the productions that reflect zero or more unit productions followed by a nonunit production. Thus, $S \stackrel{*}{\Rightarrow} w$ in G_1. ◄



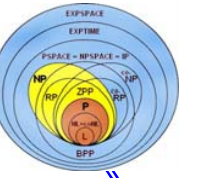
We now summarize the various simplifications described so far.

To "clean up" a context-free grammar, we can

1. Eliminating ϵ-productions.
2. Eliminating unit productions.
3. Eliminating useless symbols.

This is a safe order.

BREAK FOR 15 MINUTES

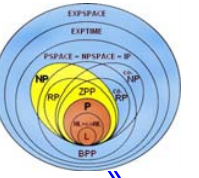


An Example of Simplifying CFG

We have presented various simplifications for a CFG, and achieved at

Theorem 7.8 *If G is a CFG generating a language that contains at least one string other than ϵ, then there is another CFG G' such that $L(G') = L(G)/\{\epsilon\}$, and G' has no ϵ-productions, unit productions, or useless symbols.*

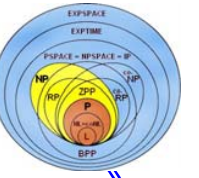
To convert G into G', some care must be taken in the order of application of the constructions. A safe order is eliminating ϵ-productions first, then unit productions, and then useless symbols.



Let PDA $P = (\{p, q\}, \{0, 1\}, \{X, Z_0\}, \delta, q, Z_0)$, where δ is given by

1. $\delta(q, 1, Z_0) = \{(q, XZ_0)\}$
2. $\delta(q, 1, X) = \{(q, XX)\}$
3. $\delta(q, 0, X) = \{(p, X)\}$
4. $\delta(q, \epsilon, Z_0) = \{(q, \epsilon)\}$
5. $\delta(p, 1, X) = \{(p, \epsilon)\}$
6. $\delta(p, 0, Z_0) = \{(q, Z_0)\}$

We have constructed the equivalent CFG G_P such that $L(G_P) = N(P)$. Now we want to "clean up" G_P. $N(P) = \{\epsilon\} \bigcup \{1^{n_1}01^{n_1}01^{n_2}01^{n_2}0 \cdots 1^{n_k}01^{n_k}0 \mid k, n_1, \cdots n_k \geq 1\}$



We get $G_P = (V, \{0, 1\}, R, S)$ where

$$V = \{S, [pXp], [pXq], [pZ_0p], [pZ_0q], [qXp], [qXq], [qZ_0p], [qZ_0q]\}$$

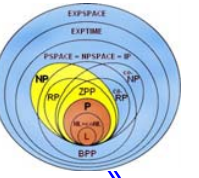
and the productions in R are

$S \to [qZ_0q] \,|\, [qZ_0p], \quad [qZ_0q] \to \epsilon, \; [pXp] \to 1$

$[qZ_0q] \to 1[qXq][qZ_0q] \,|\, 1[qXp][pZ_0q], \;\; [qZ_0p] \to 1[qXq][qZ_0p] \,|\, 1[qXp][pZ_0p]$

$[qXq] \to 1[qXq][qXq] \,|\, 1[qXp][pXq], \;\; [qXp] \to 1[qXq][qXp] \,|\, 1[qXp][pXp]$

$[qXq] \to 0[pXq], \; [qXp] \to 0[pXp], \; [pZ_0q] \to 0[qZ_0q], \; [pZ_0p] \to 0[qZ_0p]$



We may, for convenience, replace the triple $[pXp], [pXq], [pZ_0p], [pZ_0q], [qXp], [qXq], [qZ_0p], [qZ_0q]$ by some less complex symbols, say A, B, C, D, E, F, G, H.

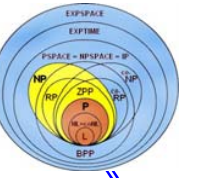
If we do, then the complete grammar consists of the productions:

$S \to H \,|\, G,$

$H \to 1FH \,|\, 1ED,\ G \to 1FG \,|\, 1EC,$

$F \to 1FF \,|\, 1EB \,|\, 0B,\ E \to 1FE \,|\, 1EA \,|\, 0A,$

$H \to \epsilon,\ A \to 1,\ D \to 0H,\ C \to 0G$



1. Eliminating ϵ-productions

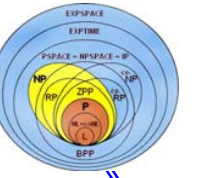
We compute $n(G_P) = \{H\}$, and eliminate nullable symbol H:

$S \to H \,|\, G,$

$H \to 1F \,|\, 1FH \,|\, 1ED,\ G \to 1FG \,|\, 1EC,$

$F \to 1FF \,|\, 1EB \,|\, 0B,\ E \to 1FE \,|\, 1EA \,|\, 0A,$

$A \to 1,\ D \to 0 \,|\, 0H,\ C \to 0G$



2. Eliminating unit productions

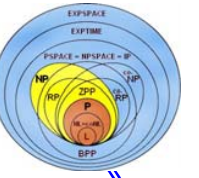
We compute $u(G_P) = \{(A,A),(B,B),(C,C),(D,D),(E,E),(F,F),(G,G),(H,H),(S,S),(S,G),(S,H)\}$, and eliminate unit productions $S \to H$ and $S \to G$:

$S \to 1F \mid 1FH \mid 1ED \mid 1FG \mid 1EC,$

$H \to 1F \mid 1FH \mid 1ED,\ G \to 1FG \mid 1EC,$

$F \to 1FF \mid 1EB \mid 0B,\ E \to 1FE \mid 1EA \mid 0A,$

$A \to 1,\ D \to 0 \mid 0H,\ C \to 0G$



3.1 Eliminating useless symbols – nongenerating symbols

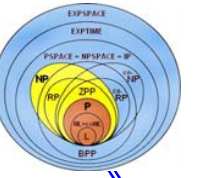
We compute $g(G_P) = \{0, 1, A, D, E, H, S\}$, and eliminate nongenerating symbols G, F, C, B:

$S \to 1ED,$

$H \to 1ED,$

$E \to 1EA \,|\, 0A,$

$A \to 1,\ D \to 0 \,|\, 0H$



3.2 Eliminating useless symbols – nonreachable symbols

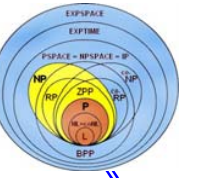
We compute $r(G_P) = \{S, 1, E, D, A, 0, H\}$, all symbols are reachable. So:

$$S \to 1ED,\ H \to 1ED,\ E \to 1EA\,|\,0A,\ A \to 1,\ D \to 0\,|\,0H$$

In fact, if we notice that there is only a single production for variables A and H, respectively, we may write the complete grammar G'_P as

$$S \to 1ED,\ E \to 1E1\,|\,01,\ D \to 0\,|\,01ED$$

Notice that $N(P) = L(G'_P) \cup \{\epsilon\}$.

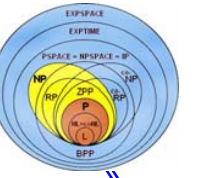


Chomsky Normal Form

We shall show that every nonempty CFL without ϵ has a grammar G without useless symbols, and such that every production is of the form

- $A \to BC$, where $\{A, B, C\} \subseteq V$, or
- $A \to a$, where $A \in V$, and $a \in T$.

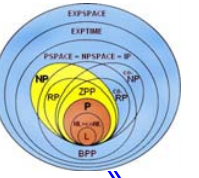
Such a grammar is said to be in Chomsky Normal Form or CNF. One of the uses of CNF is to turn parse trees into binary trees.



To achieve CNF, start with any grammar for CFL, and

1. "Clean up" the grammar.
2. Arrange that all bodies of length 2 or more consists only of variables.
3. Break bodies of length 3 or more into a cascade of two-variable-bodied productions.

For step 2, for every terminal a that appears in a body of length ≥ 2, create a new variable, say A, and replace a by A in all bodies. Then add a new rule $A \to a$.



For step 3, for each rule of the form

$$A \to B_1 B_2 \cdots B_k,$$

($k \geq 3$), introduce new variables $C_1, C_2, \cdots, C_{k-2}$, replace the rule with

$$A \to B_1 C_1$$

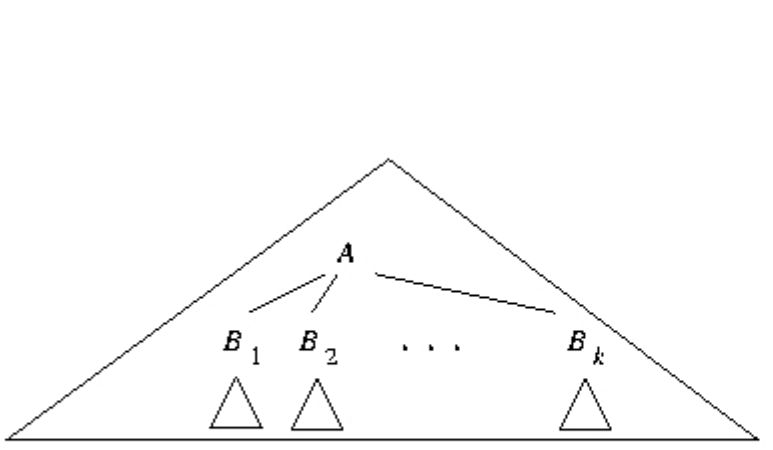
$$C_1 \to B_2 C_2$$

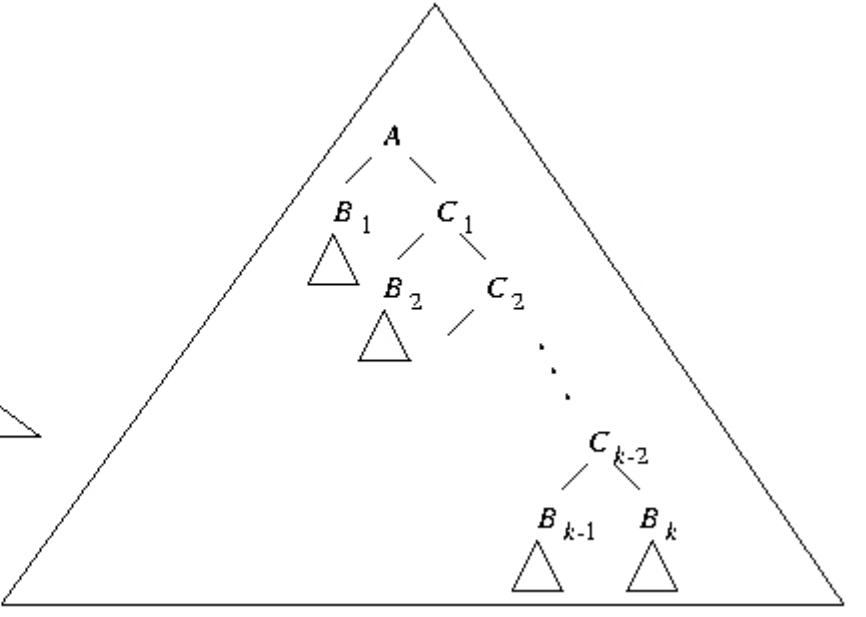
$$\cdots$$

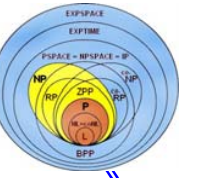
$$C_{k-3} \to B_{k-2} C_{k-2}$$

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

Illustration of the effect of step 3







Example of CFN Conversion

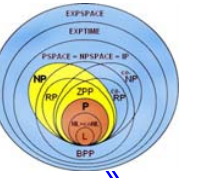
Let's start with the grammar (step 1 already done)

$$E \to E + T \mid T \times F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid, I0 \mid I1$$

$$T \to T \times F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

$$F \to (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

$$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$



For step 2, we need the rules

$$A \to a,\ B \to b,\ Z \to 0,\ O \to 1,\ P \to +,\ M \to \times,\ L \to (,\ R \to)$$

and by replacing we get the grammar

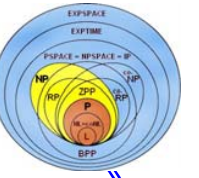
$$E \to EPT \,|\, TMF \,|\, LER \,|\, a \,|\, b \,|\, IA \,|\, IB \,|, IZ \,|\, IO$$

$$T \to TMF \,|\, LER \,|\, a \,|\, b \,|\, IA \,|\, IB \,|\, IZ \,|\, IO$$

$$F \to LER \,|\, a \,|\, b \,|\, IA \,|\, IB \,|\, IZ \,|\, IO$$

$$I \to a \,|\, b \,|\, IA \,|\, IB \,|\, IZ \,|\, IO$$

$$A \to a,\ B \to b,\ Z \to 0,\ O \to 1,\ P \to +,\ M \to \times,\ L \to (,\ R \to)$$



For step 3, we introduce C_1 for EPT with $C_1 \to PT$, C_2 for TMF with $C_2 \to MF$, and C_3 for LER with $C_3 \to ER$. And finally we get the CNF grammar:

$$E \to EC_1 \,|\, TC_2 \,|\, LC_3 \,|\, a \,|\, b \,|\, IA \,|\, IB \,|, IZ \,|\, IO$$

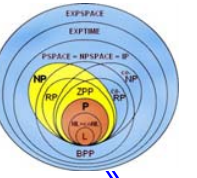
$$T \to TC_2 \,|\, LC_3 \,|\, a \,|\, b \,|\, IA \,|\, IB \,|\, IZ \,|\, IO$$

$$F \to LC_3 \,|\, a \,|\, b \,|\, IA \,|\, IB \,|\, IZ \,|\, IO$$

$$I \to a \,|\, b \,|\, IA \,|\, IB \,|\, IZ \,|\, IO$$

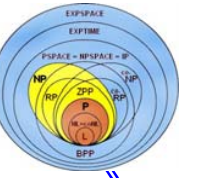
$$C_1 \to PT,\ C_2 \to MF,\ C_3 \to ER$$

$$A \to a,\ B \to b,\ Z \to 0,\ O \to 1,\ P \to +,\ M \to \times,\ L \to (,\ R \to)$$


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