Chapter 7

Properties of Context-Free Languages

- We first simplify CFGs
- Then we prove "pumping lemma" for CFLs
- Then closure properties and decision properties are considered.

Chomsky Normal Form (CNF)

- Every CFL (without ϵ) is generated by a CFG in which all productions are of the form
  $A \rightarrow BC \quad \text{or} \quad A \rightarrow a$

But, this requires preprocessing of the CFG …

- We must eliminate *useless* symbols,
- We must eliminate $\epsilon-productions$, and
- We must eliminate *unit* productions.

Eliminating useless symbols

We say a symbol X is *useful* for a grammar $G = (V, T, P, S)$ if there is some derivation of the form $S \stackrel{*}{\Rightarrow} \alpha X \beta \stackrel{*}{\Rightarrow} w$, where w is in T^*.

- Note that X may be either a variable or a terminal symbol.
- If X is not useful, we say it is *useless*.
- This useless symbol elimination should not change the CFL. We see such a one.
  - The language is in tact, but useless are removed.

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

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

1. We say X is *generating* if $X \stackrel{*}{\Rightarrow} w$ for some terminal string w. Note that every terminal is generating, since w can be that terminal itself, which is derived by zero steps.

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

1. We say X is *generating* if $X \stackrel{*}{\Rightarrow} w$ for some terminal string w. Note that every terminal is generating, since w can be that terminal itself, which is derived by zero steps.

2. We say X is *reachable* if there is a derivation $S \stackrel{*}{\Rightarrow} \alpha X \beta$ for some α and β.

Order is important

1. Eliminate nongenerating symbols *first*,
2. *then*, from the remaining eliminate unreachable symbols.

Order is important

1. Eliminate nongenerating symbols *first*,
2. *then*, from the remaining eliminate unreachable symbols.

- Whatever left are only useful ones.

Example 7.1: Consider the grammar:

$$
\begin{aligned}
&S \to AB \mid a \\
&A \to b
\end{aligned}
$$

Example 7.1: Consider the grammar:

$$\begin{aligned} &S \to AB \mid a \\ &A \to b \end{aligned}$$

- Find non-generating symbols.
- How to find this?

Example 7.1: Consider the grammar:

$$
\begin{aligned}
S &\to AB \mid a \\
A &\to b
\end{aligned}
$$

- Find non-generating symbols.
- How to find this?
- We can find generating symbols, inductively.

Example 7.1: Consider the grammar:

$$S \to AB \mid a$$
$$A \to b$$

- Find non-generating symbols.
- How to find this?
- We can find generating symbols, inductively.
- Basis: Every symbol of T is generating. (Why?)
- Induction: if every symbol on RHS of a production $A \to \alpha$ is generating, then A is generating.

Example 7.1: Consider the grammar:

$$
\begin{aligned}
S &\rightarrow AB \mid a \\
A &\rightarrow b
\end{aligned}
$$

- What are the generating symbols?

Example 7.1: Consider the grammar:

$$
\begin{aligned}
S &\to AB \mid a \\
A &\to b
\end{aligned}
$$

- What are the generating symbols?
- {a,b,A,S}

Example 7.1: Consider the grammar:

$$S \to AB \mid a$$
$$A \to b$$

- What are the generating symbols?
- {a,b,A,S}
- So the non-generating symbol is B.

Example 7.1: Consider the grammar:

$$
\begin{aligned}
&S \to AB \mid a \\
&A \to b
\end{aligned}
$$

- What are the generating symbols?
- {a,b,A,S}
- So the non-generating symbol is B.
- So removing B we are left with $S \to a,\ A \to b$

Theorem 7.4: The algorithm above finds all and only the generating symbols of G.

- Proof is skipped.

Finding reachable symbols

- By induction, again.
- Basis. S is reachable.
- Induction. A is reachable, and $A \to \alpha$, then all symbols in α are reachable.

Example 7.1: Consider the grammar:

$$\begin{aligned} S &\to AB \mid a \\ A &\to b \end{aligned}$$

- Find reachable.

Example 7.1: Consider the grammar:

$$
\begin{aligned}
&S \to AB \mid a \\
&A \to b
\end{aligned}
$$

- Find reachable.

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

As per order, to remove useless

$$S \to AB \mid a$$
$$A \to b$$

remove non-generating

$$S \to a$$
$$A \to b$$

remove unreachable

$$S \to a$$

As per order, to remove useless

$$S \to AB \mid a$$
$$A \to b$$

remove non-generating

$$S \to a$$
$$A \to b$$

remove unreachable

$$S \to a$$

Wrong order

$$S \to AB \mid a$$
$$A \to b$$

remove unreachable

$$S \to AB \mid a$$
$$A \to b$$

remove non-generating

A is useless, but remains

$$S \to a$$
$$A \to b$$

Eliminating ϵ-productions

- $A \rightarrow \epsilon$ is an ϵ-production

➢ Let L be a CFL.

➢ For $L - \{\epsilon\}$ there is a CFG which is without ϵ-productions

Eliminating ϵ-productions

- Discover *nullable* variables.
  - A variable A is nullable if $A \overset{*}{\Rightarrow} \epsilon$
  - In this case, replace $B \rightarrow CAD$ by $B \rightarrow CAD|CD$

Eliminating ϵ-productions

- Discover *nullable* variables.
  - A variable A is nullable if $A \overset{*}{\Rightarrow} \epsilon$
  - In this case, replace $B \rightarrow CAD$ by $B \rightarrow CAD|CD$
- Why this correction is needed??

Finding the nullable

Let $G = (V, T, P, S)$ be a CFG. We can find all the nullable symbols of G by the following iterative algorithm.

Finding the nullable

Let $G = (V, T, P, S)$ be a CFG. We can find all the nullable symbols of G by the following iterative algorithm.

BASIS: If $A \to \epsilon$ is a production of G, then A is nullable.

Finding the nullable

Let $G = (V, T, P, S)$ be a CFG. We can find all the nullable symbols of G by the following iterative algorithm.

BASIS: If $A \to \epsilon$ is a production of G, then A is nullable.

INDUCTION: If there is a production $B \to C_1C_2 \cdots C_k$, where each C_i is nullable, then B is nullable. Note that each C_i must be a variable to be nullable, so we only have to consider productions with all-variable bodies.

Finding the nullable

Let $G = (V, T, P, S)$ be a CFG. We can find all the nullable symbols of G by the following iterative algorithm.

BASIS: If $A \to \epsilon$ is a production of G, then A is nullable.

INDUCTION: If there is a production $B \to C_1 C_2 \cdots C_k$, where each C_i is nullable, then B is nullable. Note that each C_i must be a variable to be nullable, so we only have to consider productions with all-variable bodies.

Theorem 7.7: In any grammar G, the only nullable symbols are the variables found by the algorithm above.

Example 7.8: Consider the grammar

$$\begin{aligned} S &\to AB \\ A &\to aAA \mid \epsilon \\ B &\to bBB \mid \epsilon \end{aligned}$$

- Find the nullable symbols.

Example 7.8: Consider the grammar

$$\begin{aligned} S &\to AB \\ A &\to aAA \mid \epsilon \\ B &\to bBB \mid \epsilon \end{aligned}$$

- Find the nullable symbols.
- {A,B,S}

Example 7.8: Consider the grammar

$$\begin{aligned} S &\to AB \\ A &\to aAA \mid \epsilon \\ B &\to bBB \mid \epsilon \end{aligned}$$

- Find the nullable symbols.
- {A,B,S}
- Apply the method.

Example 7.8: Consider the grammar

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

- Find the nullable symbols.
- {A,B,S}
- Apply the method.

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

Example 7.8: Consider the grammar

$$
\begin{aligned}
S &\rightarrow AB \\
A &\rightarrow aAA \mid \epsilon \\
B &\rightarrow bBB \mid \epsilon
\end{aligned}
$$

- Find the nullable symbols.
- {A,B,S}
- Apply the method.

$$
\begin{aligned}
S &\rightarrow AB \mid A \mid B \\
A &\rightarrow aAA \mid aA \mid a \\
B &\rightarrow bBB \mid bB \mid b
\end{aligned}
$$

- Any thing missing??

Example 7.8: Consider the grammar

$$\begin{aligned} S &\to AB \\ A &\to aAA \mid \epsilon \\ B &\to bBB \mid \epsilon \end{aligned}$$

- Find the nullable symbols.
- {A,B,S}
- Apply the method.

$$\begin{aligned} S &\to AB \mid A \mid B \\ A &\to aAA \mid aA \mid a \\ B &\to bBB \mid bB \mid b \end{aligned}$$

- Any thing missing?? ϵ is not in the new language.

Theorem 7.9: If the grammar G_1 is constructed from G by the above construction for eliminating ϵ-productions, then $L(G_1) = L(G) - \{\epsilon\}$.

Eliminating Unit Productions

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

Eliminating Unit Productions

- A unit production is of the form $A \to B$, where A and B are variables.
- They may be useful, but we can construct an equivalent grammar without them.
- This simplifies the CFG.
  - Unit productions introduce extra steps into derivations that technically need not be there.
  - This complicates proving certain facts.

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

- In this $E \rightarrow T$ is a unit production.
- How to remove this?
- $E \rightarrow F \mid T * F \mid E + T$
- Still $E \rightarrow F$ is problematic
- Finally…

$$E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \mid (E) \mid T * F \mid E + T$$

$$A \Rightarrow B_1 \Rightarrow B_2 \Rightarrow \cdots \Rightarrow B_n \Rightarrow \alpha$$

can be replaced by

$$A \rightarrow \alpha.$$

How to do this systematically?

- Unit pairs are identified.
- Along with this, the CFG is used to produce a new CFG which is without any unit production.

Unit pair

- (A, B) is a unit pair if $A \overset{*}{\Rightarrow} B$

- Note, if the CFG have $A \rightarrow BC$ and $C \rightarrow \epsilon$
- then, (A, B) is a unit pair

BASIS: (A, A) is a unit pair for any variable A. That is, $A \stackrel{*}{\Rightarrow} A$ by zero steps.

INDUCTION: Suppose we have determined that (A, B) is a unit pair, and $B \to C$ is a production, where C is a variable. Then (A, C) is a unit pair.

$$\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}$$

The basis gives us the unit pairs (E, E), (T, T), (F, F), and (I, I). For the inductive step, we can make the following inferences:

1. (E, E) and the production $E \rightarrow T$ gives us unit pair (E, T).
2. (E, T) and the production $T \rightarrow F$ gives us unit pair (E, F).
3. (E, F) and the production $F \rightarrow I$ gives us unit pair (E, I).
4. (T, T) and the production $T \rightarrow F$ gives us unit pair (T, F).
5. (T, F) and the production $F \rightarrow I$ gives us unit pair (T, I).
6. (F, F) and the production $F \rightarrow I$ gives us unit pair (F, I).

There are no more pairs that can be inferred, and in fact these ten pairs represent all the derivations that use nothing but unit productions. $\square$

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; in that way, P_1 contains all the nonunit productions in P.

Given CFG:

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

Non-unit productions are

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & (E) \\
T & \rightarrow & T * F \\
E & \rightarrow & E + T
\end{array}
$$

Given CFG:

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

Non-unit productions are

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & (E) \\
T & \rightarrow & T * F \\
E & \rightarrow & E + T
\end{array}
$$

Unit Pairs

Pair
(E, E)
(E, T)
(E, F)
(E, I)
(T, T)
(T, F)
(T, I)
(F, F)
(F, I)
(I, I)

Given CFG:

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

Non-unit productions are

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & (E) \\
T & \rightarrow & T * F \\
E & \rightarrow & E + T
\end{array}
$$

Unit Pairs

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

Given CFG:

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

Non-unit productions are

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & (E) \\
T & \rightarrow & T * F \\
E & \rightarrow & E + T
\end{array}
$$

Unit Pairs

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

CFG without unit productions

$$
\begin{array}{l}
E \rightarrow E + T \mid T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
T \rightarrow T * F \mid (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F \rightarrow (E) \mid a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1
\end{array}
$$

The order in which these preprocessing steps can occur

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

* **Exercise 7.1.1:** Find a grammar equivalent to

$$
\begin{array}{lcl}
S & \rightarrow & AB \mid CA \\
A & \rightarrow & a \\
B & \rightarrow & BC \mid AB \\
C & \rightarrow & aB \mid b
\end{array}
$$

with no useless symbols.

* **Exercise 7.1.1:** Find a grammar equivalent to

$$\begin{array}{lcl} S & \rightarrow & AB \mid CA \\ A & \rightarrow & a \\ B & \rightarrow & BC \mid AB \\ C & \rightarrow & aB \mid b \end{array}$$

with no useless symbols.

- Eliminate non-generating, first.

* **Exercise 7.1.1:** Find a grammar equivalent to

$$
\begin{array}{lcl}
S & \rightarrow & AB \mid CA \\
A & \rightarrow & a \\
B & \rightarrow & BC \mid AB \\
C & \rightarrow & aB \mid b
\end{array}
$$

with no useless symbols.

- Eliminate non-generating, first.
- Generating symbols = $\{a, b, A, C, S\}$
- Non-generating symbol is B.

* **Exercise 7.1.1:** Find a grammar equivalent to

$$\begin{array}{lcl} S & \rightarrow & AB \mid CA \\ A & \rightarrow & a \\ B & \rightarrow & BC \mid AB \\ C & \rightarrow & aB \mid b \end{array}$$

with no useless symbols.

- Eliminate non-generating, first.
- Generating symbols = $\{a, b, A, C, S\}$
- Non-generating symbol is B.
- We get $S \rightarrow CA,\ A \rightarrow a,\ C \rightarrow b$

*** Exercise 7.1.1:** Find a grammar equivalent to

$$\begin{array}{lcl} S & \rightarrow & AB \mid CA \\ A & \rightarrow & a \\ B & \rightarrow & BC \mid AB \\ C & \rightarrow & aB \mid b \end{array}$$

with no useless symbols.

- Eliminate non-generating, first.
- Generating symbols = $\{a, b, A, C, S\}$
- Non-generating symbol is B.
- We get $S \rightarrow CA,\ A \rightarrow a,\ C \rightarrow b$
- Remove unreachable.

* **Exercise 7.1.1:** Find a grammar equivalent to

$$\begin{array}{lcl} S & \rightarrow & AB \mid CA \\ A & \rightarrow & a \\ B & \rightarrow & BC \mid AB \\ C & \rightarrow & aB \mid b \end{array}$$

with no useless symbols.

- Eliminate non-generating, first.
- Generating symbols = $\{a, b, A, C, S\}$
- Non-generating symbol is B.
- We get $S \rightarrow CA,\ A \rightarrow a,\ C \rightarrow b$
- Remove unreachable.
- All are reachable. So, $S \rightarrow CA,\ A \rightarrow a,\ C \rightarrow b$ is the answer.

Chomsky Normal Form

- For a CFL without ϵ, where all productions are of the form $A \to BC, A \to a$.
- After preprocessing steps, it is quite easy to get in to CNF.

- $A \to BCDE$ can be replaced by

  $A \to BF, F \to CDE$.

- $A \to BCDE$ can be replaced by

  $A \to BF, F \to CDE$.
- Then $F \to CDE$ can be replaced by $F \to CG$ and $G \to DE$

- $A \rightarrow BCDE$ can be replaced by

  $A \rightarrow BF, F \rightarrow CDE$.
- Then $F \rightarrow CDE$ can be replaced by $F \rightarrow CG$ and $G \rightarrow DE$
- So, $A \rightarrow BCDE$ can be replaced by $A \rightarrow BF$, $F \rightarrow CG$ and $G \rightarrow DE$

- Similarly, $A \to BabC$ can be replaced by $A \to BDEC, D \to a, E \to b$.
- Then $A \to BDEC$ can be replaced by …

* **Exercise 7.1.2:** Begin with the grammar:

$$\begin{array}{lll} S & \rightarrow & ASB \mid \epsilon \\ A & \rightarrow & aAS \mid a \\ B & \rightarrow & SbS \mid A \mid bb \end{array}$$

a) Eliminate ϵ-productions.

b) Eliminate any unit productions in the resulting grammar.

c) Eliminate any useless symbols in the resulting grammar.

d) Put the resulting grammar into Chomsky Normal Form.

- Theorem: Let G be a CFG in CNF form, then for any $w \in L(G), w$ can be derived in exactly $(2|w| - 1)$ steps.

- Theorem: Let G be a CFG in CNF form, then for any $w \in L(G), w$ can be derived in exactly $(2|w| - 1)$ steps.
- Proof: Parse tree when the CFG is in CNF form is a binary tree (except for a small technical issue with $A \to a$)

- Theorem: Let G be a CFG in CNF form, then for any $w \in L(G), w$ can be derived in exactly $(2|w| - 1)$ steps.
- Proof: Parse tree when the CFG is in CNF form is a binary tree (except for a small technical issue with $A \rightarrow a$)
- Let n_0 be nodes with 0 children (leaves)
- Let n_1 be nodes with 1 child
- Let n_2 be nodes with 2 children

- Theorem: Let G be a CFG in CNF form, then for any $w \in L(G), w$ can be derived in exactly $(2|w| - 1)$ steps.
- Proof: Parse tree when the CFG is in CNF form is a binary tree (except for a small technical issue with $A \to a$)
- Let n_0 be nodes with 0 children (leaves)
- Let n_1 be nodes with 1 child
- Let n_2 be nodes with 2 children
- We have $n_0 = n_2 + 1$ (can you prove this?)

- We have, $n_0 = |w|$
- $n_0 = n_1$ (why?)
- Number of steps

$$
\begin{aligned}
&= n_2 + n_1 \\
&= n_0 + n_0 - 1 \\
&= 2n_0 - 1 \\
&= 2|w| - 1
\end{aligned}
$$