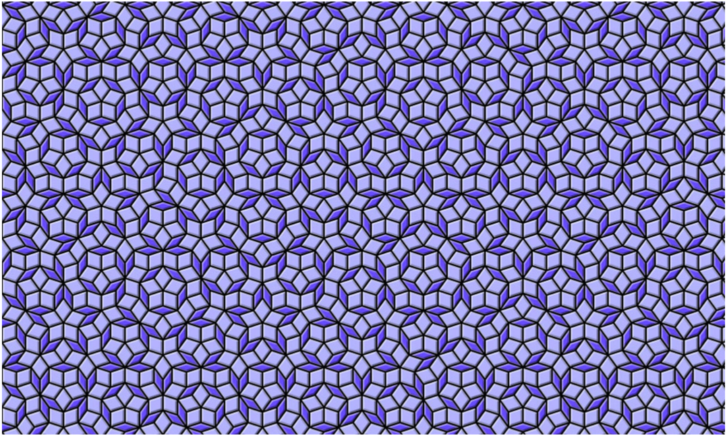
Automata, Languages and Computation

Chapter 7 : Properties of Context-Free Languages
Part I

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture based on material originally developed by :
Gösta Grahne, Concordia University

Properties of context-free languages



1. Eliminating useless symbols : we can delete symbols that do not contribute to the derivation process
2. Eliminating ϵ-productions : we can eliminate all derivations generating the empty string
3. Eliminating unary productions : we can eliminate chains of productions that do not change the length of the sentential forms
4. CFG simplification : combine all presented elimination techniques
5. Chomsky normal form : every CFL has a CFG in special form

CFG semplification

Let G be some CFG. We can eliminate some grammatical symbols and some productions **preserving** the generated language

The motivation is to make the grammar easier to process

We investigate the following techniques :

- elimination of variable and terminal symbols that do not appear in any derivation for strings in the language
- elimination of ϵ-productions, that is, productions of the form $A \rightarrow \epsilon$
- elimination of unary productions, that is, productions of the form $A \rightarrow B$

Useless symbols

Assume a CFG $G = (V, T, P, S)$. Symbol $X \in V \cup T$ is called

- **reachable** if there exists a derivation $S \overset{*}{\Rightarrow} \alpha X \beta$ for some $\alpha, \beta \in (V \cup T)^*$
- **generating** if there exists a derivation $X \overset{*}{\Rightarrow} w$ for some $w \in T^*$
- **useful** if it is reachable and generating; otherwise, X is called **useless**

Example

Consider the CFG G with the following productions

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

S, A, a, b are generating, B is not generating

In order to eliminate B we need to eliminate the production $S \to AB$, resulting in the new grammar

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

Now only S and a are reachable

After eliminating A and b, the resulting grammar has the only production $S \to a$

Example

Note :

- If we start by checking the reachable symbols, we find that no production of the initial grammar must be eliminated
- If we subsequently check for the generating symbols, we have to eliminate B, resulting in a grammar that has unreachable symbols

Removal of non-generating symbols might break reachability relation

Elimination of useless symbols

Let us assume we already have algorithms for computing the sets of generating and reachable symbols of a CFG

We present these algorithms in the next slides

Algorithm Given as input a CFG $G = (V, T, P, S)$ with $L(G) \neq \emptyset$

- we build $G_1 = (V_1, T_1, P_1, S)$ by eliminating from G all non-generating symbols (in G) and all productions in which they appear ($S \in V_1$ since $L(G) \neq \emptyset$)
- we build $G_2 = (V_2, T_2, P_2, S)$ by eliminating from G_1 all non-reachable symbols (in G_1) and all productions in which they appear

Algorithm for generating symbols

Let $G = (V, T, P, S)$. We compute the set $g(G)$ of all generating symbols of G by means of the following inductive algorithm

Base $g(G) \leftarrow T$

Induction if $(A \rightarrow X_1 X_2 \cdots X_n) \in P$ and $X_i \in g(G)$ for each i with $1 \leqslant i \leqslant n$, then

$$g(G) \leftarrow g(G) \cup \{A\}$$

This algorithm is bottom-up, since information is transferred from the right-hand side of a production to its left-hand side

Example

Consider the CFG G with productions

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

At the base step we have $g(G) = \{a, b\}$

From $S \rightarrow a$ we add S to $g(G)$; from $A \rightarrow b$ we add A to $g(G)$. No other production can contribute to set $g(G)$

We thus have $g(G) = \{S, A, a, b\}$

Algorithm for reachable symbols

Let $G = (V, T, P, S)$. We can compute the set $r(G)$ of all reachable symbols of G using the following inductive algorithm

Base $r(G) \leftarrow \{S\}$

Induction if $(A \rightarrow X_1 X_2 \cdots X_n) \in P$ and $A \in r(G)$, then for each i with $1 \leqslant i \leqslant n$

$$r(G) \leftarrow r(G) \cup \{X_i\}$$

This algorithm is top-down, since information is transferred from the left-hand side of a production to its right-hand side

Example

Consider the CFG G with productions

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

At the base step we have $r(G) = \{S\}$

From $S \rightarrow AB$ we add A and B to $r(G)$.
From $S \rightarrow a$ we add a to $r(G)$.
From $A \rightarrow b$ we add b to $r(G)$

We thus obtain $r(G) = \{S, A, B, a, b\}$

Elimination of ϵ-productions

Observation : If $\epsilon \in L$ we **cannot** eliminate ϵ-productions preserving the generated language

We prove that if L is a context-free language, then there is a CFG without ϵ-productions that generates $L \setminus \{\epsilon\}$

String ϵ must be processed separately

Elimination of ϵ-productions

Variable A is **nullable** if $A \stackrel{*}{\Rightarrow} \epsilon$

Idea : If A is nullable and there exists a production $B \rightarrow CAD$, then

- we remove productions with right-hand side ϵ
- we construct two alternative versions of the above production

$B \rightarrow CD$	A generates ϵ
$B \rightarrow CAD$	A generates other strings

If also C and D are nullable, we have to remove all possible combinations of C, A and D from production $B \rightarrow CAD$

Algorithm for nullable variables

Let $G = (V, T, P, S)$. We can compute the set $n(G)$ of all nullable variables of G by means of the following inductive algorithm

Base $n(G) \leftarrow \{A \mid (A \rightarrow \epsilon) \in P\}$

Induction If there exists in G a production $A \rightarrow B_1 B_2 \cdots B_k$ such that $B_i \in n(G)$ for each i, $1 \leqslant i \leqslant k$, then

$$n(G) \leftarrow n(G) \cup \{A\}$$

Very similar to the algorithm for generating symbols

Elimination of ϵ-productions

Let $G = (V, T, P, S)$ be some CFG. Given $n(G)$, we can build a new CFG $G_1 = (V, T, P_1, S)$ where P_1 is computed from P as follows

- each production $(A \rightarrow \epsilon) \in P$ is excluded from P_1
- let $p : (A \rightarrow X_1 X_2 \cdots X_k) \in P$ with $k \geqslant 1$; define $\mathcal{N} = \{i_1, i_2, \ldots, i_m\}$ as the set of all **indices** of nullable variables X_i, $m \leqslant k$
- for every possible choice of set $\mathcal{N}' \subseteq \mathcal{N}$, we add to P_1 a production constructed from p by deleting each X_i with $i \in \mathcal{N}'$

Exception : In case $m = k$, we do not add to P_1 the null production $A \rightarrow \epsilon$

Example

Elimination of ϵ-production from CFG G with productions

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

We first compute set $n(G)$

- $A, B \in n(G)$ since $A \rightarrow \epsilon$ and $B \rightarrow \epsilon$
- $S \in n(G)$ since $S \rightarrow AB$, with $A, B \in n(G)$

Example

From $S \to AB$ we construct the new productions $S \to AB \mid A \mid B$

From $A \to aAA$ we construct the new productions
$A \to aAA \mid aA \mid a$

From $B \to bBB$ we construct the new productions
$B \to bBB \mid bB \mid b$

The resulting CFG G_1 has productions

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

and we have $L(G_1) = L(G) \smallsetminus \{\epsilon\}$

Elimination of unary productions

Let $G = (V, T, P, S)$ be some CFG. A **unary** production has the form $A \to B$, where both A and B are variables in V

Note : $A \to a$ and $A \to \epsilon$ are not unary productions

We can eliminate unary productions by expanding the variables in the right-hand side

Example

Our grammar for arithmetic expressions with productions

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

has unary productions $E \rightarrow T$, $T \rightarrow F$ and $F \rightarrow I$

Expanding the right-hand side of production $E \rightarrow T$ results in

$$E \rightarrow F \mid T * F$$

which introduces a new unary production $E \rightarrow F$

Example

If we in turn expand the right-hand side of $E \rightarrow F$ we get

$$E \rightarrow I \mid (E)$$

Finally, if we expand $E \rightarrow I$ we get

$$E \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

The method of successive expansions **does not work** if there is some cycle among unary rules, such as in

$$A \rightarrow B, \;\; B \rightarrow C, \;\; C \rightarrow A$$

Elimination of unary productions

We now present a method based on the notion of unary pairs which eliminates the unary productions in the **general case**

Let $G = (V, T, P, S)$ be some CFG. (A, B) is a **unary pair** if $A \overset{*}{\Rightarrow} B$ using **only** unary productions

Note : For productions $A \rightarrow BC$ and $C \rightarrow \epsilon$ we have $A \overset{*}{\Rightarrow} B$; however, we have not used unary productions only

Algorithm for unary pairs

Let $G = (V, T, P, S)$. We can compute the set $u(G)$ of all unary pairs of G by means of the following inductive algorithm

Base $u(G) \leftarrow \{(A, A) \mid A \in V\}$

Induction If $(A, B) \in u(G)$ and $(B \rightarrow C) \in P$, then

$$u(G) \leftarrow u(G) \cup \{(A, C)\}$$

Compare with the algorithm for reachable symbols

Example

Consider the CFG

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

In the base step we derive the unary pairs (E, E), (T, T), (F, F) e (I, I)

Example

In the inductive step

- from (E, E) and $E \rightarrow T$ we add pair (E, T)
- from (E, T) and $T \rightarrow F$ we add pair (E, F)
- from (E, F) and $F \rightarrow I$ we add pair (E, I)
- from (T, T) and $T \rightarrow F$ we add pair (T, F)
- from (T, F) and $F \rightarrow I$ we add pair (T, I)
- from (F, F) and $F \rightarrow I$ we add pair (F, I)

Eliminating unary productions

Let $G = (V, T, P, S)$ be some CFG. We produce a new CFG $G_1 = (V, T, P_1, S)$, where P_1 is constructed from P as follows

- compute $u(G)$
- for each $(A, B) \in u(G)$ and for each $(B \to \alpha) \in P$ which is not a unary production, add to P_1 the production $A \to \alpha$

Note :

- In the second step, we might have $A = B$; in this way non-unary productions in P are all transferred to P_1
- Unary productions are **filtered**

Example

We eliminate unary productions from CFG

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

We have already computed set $u(G)$ in a previous example

Example

The second step of the algorithm results in the following productions

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)	$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Example

Summing up, after eliminating unary productions from the grammar G with productions

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

we have the CFG G_1 with productions

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

CFG simplification

When simplifying a CFG we need to pay special attention to the **order** in which we apply the previous transformations

The **correct** ordering is

- elimination of ϵ-productions
- elimination of unary productions
- elimination of useless symbols

Chomsky normal form

A CFG is in **Chomsky normal form**, or CNF for short, if its productions have one of the two forms

- $A \rightarrow BC$, with $A, B, C \in V$
- $A \rightarrow a$, with $A \in V$ and $a \in T$

and the grammar does not have useless symbols

We show that every CFL without the empty string ϵ can be generated by CNF grammar

Chomsky normal form

In order to transform a CFG in CNF, we first need to eliminate in the **specified order**

- ϵ-productions
- unary productions
- useless symbols

The resulting grammar has productions of the form

- $A \rightarrow a$
- $A \rightarrow \alpha$, where $\alpha \in (V \cup T)^*$ and $|\alpha| \geqslant 2$

Chomsky normal form

To transform the previous CFG in CNF, we need to perform two further transformations

- right-hand sides of length 2 or larger must only have variables
- right-hand sides of length larger than 2 must be decomposed into **chains** of productions with only two variables in their right-hand side

First transformation

For each production with right-hand side α such that $|\alpha| \geqslant 2$ and for each **occurrence** in α of $a \in T$

- construct a new production $A \rightarrow a$ (A is a fresh variable)
- use A in place of a in α

Second transformation

For each production of the form

$$A \rightarrow B_1 B_2 \cdots B_k, \quad k \geqslant 3$$

- introduce fresh variables $C_1, C_2, \ldots, C_{k-2}$
- replace the production with the chain of new productions

$$\begin{aligned} A &\rightarrow B_1 C_1 \\ C_1 &\rightarrow B_2 C_2 \\ &\vdots \\ C_{k-3} &\rightarrow B_{k-2} C_{k-2} \\ C_{k-2} &\rightarrow B_{k-1} B_k \end{aligned}$$

Example

Consider the CFG from the previous example

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

The first transformation adds productions for the terminal symbols

$$\begin{array}{llll}
A \rightarrow a & B \rightarrow b & Z \rightarrow 0 & O \rightarrow 1 \\
P \rightarrow + & M \rightarrow * & L \rightarrow (& R \rightarrow)
\end{array}$$

Example

The first transformation results in the CFG

$$E \rightarrow EPT \mid TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$$
$$T \rightarrow TMF \mid LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$$
$$F \rightarrow LER \mid a \mid b \mid IA \mid IB \mid IZ \mid IO$$
$$I \rightarrow a \mid b \mid IA \mid IB \mid IZ \mid IO$$
$$A \rightarrow a, \quad B \rightarrow b, \quad Z \rightarrow 0, \quad O \rightarrow 1$$
$$P \rightarrow +, \quad M \rightarrow *, \quad L \rightarrow (, \quad R \rightarrow)$$

Example

The second transformations performs the following replacements

- $E \rightarrow EPT$ replaced by
  $E \rightarrow EC_1, C_1 \rightarrow PT$
- $E \rightarrow TMF, T \rightarrow TMF$ replaced by
  $E \rightarrow TC_2, T \rightarrow TC_2, C_2 \rightarrow MF$
- $E \rightarrow LER, T \rightarrow LER, F \rightarrow LER$ replaced by
  $E \rightarrow LC_3, T \rightarrow LC_3, F \rightarrow LC_3, C_3 \rightarrow ER$

Some variable optimization has been used

Example

The second transformation results in the final CFG in CNF

$$\begin{aligned}
E &\rightarrow EC_1 \mid TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
T &\rightarrow TC_2 \mid LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
F &\rightarrow LC_3 \mid a \mid b \mid IA \mid IB \mid IZ \mid IO \\
I &\rightarrow a \mid b \mid IA \mid IB \mid IZ \mid IO \\
C_1 &\rightarrow PT, \quad C_2 \rightarrow MF, \quad C_3 \rightarrow ER \\
A &\rightarrow a, \quad B \rightarrow b, \quad Z \rightarrow 0, \quad O \rightarrow 1 \\
P &\rightarrow +, \quad M \rightarrow *, \quad L \rightarrow (, \quad R \rightarrow)
\end{aligned}$$

Exercise

Cast into CNF the CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$ with production set P

$$S \rightarrow bA \mid aB$$
$$A \rightarrow bAA \mid aS \mid a$$
$$B \rightarrow aBB \mid bS \mid b$$

There are no ϵ-productions, unary productions, or useless symbols. Therefore we apply the two transformations for the construction of the CNF

Exercise

The first transformation performs he following replacements

- $S \rightarrow bA$ replaced by $C_b \rightarrow b$ and $S \rightarrow C_bA$
- $S \rightarrow aB$ replaced by $C_a \rightarrow a$ and $S \rightarrow C_aB$
- $A \rightarrow bAA$ replaced by $A \rightarrow C_bAA$
- $A \rightarrow aS$ replaced by $A \rightarrow C_aS$
- $B \rightarrow aBB$ replaced by $B \rightarrow C_aBB$
- $B \rightarrow bS$ replaced by $B \rightarrow C_bS$

Exercise

The second transformation performs he following replacements

- $A \rightarrow C_b AA$ replaced by $A \rightarrow C_b D_1$ and $D_1 \rightarrow AA$
- $B \rightarrow C_a BB$ replaced by $B \rightarrow C_a D_2$ and $D_2 \rightarrow BB$

Exercise

The resulting CFG is

$$G_1 = (\{S, A, B, C_a, C_b, D_1, D_2\}, \{a, b\}, P', S)$$

where P' consists of the following productions

$$\begin{aligned}
S &\rightarrow C_b A \mid C_a B \\
A &\rightarrow C_a S \mid C_b D_1 \mid a \\
B &\rightarrow C_b S \mid C_a D_2 \mid b \\
D_1 &\rightarrow AA \\
D_2 &\rightarrow BB \\
C_a &\rightarrow a \\
C_b &\rightarrow b
\end{aligned}$$

Exercise

Given a CFG in CNF, how many steps are needed in order to generate a sentential form of length 9 having 2 variables and 7 terminal symbols? Discuss your answer

Solution : In CNF every production has one of the forms $A \rightarrow BC$, $A \rightarrow b$

To generate a sentential form of length $n \geqslant 1$ entirely composed by variables, we need $n - 1$ derivation steps (proof by induction on n)

Exercise

Thus 8 steps are needed for a sentential form of length 9 with 9 variables

In addition, 7 variables must become terminal symbols by means of productions of the form $A \rightarrow a$. Thus we need seven more steps in the derivation

Overall, we need 8+7=15 derivation steps

Greibach normal form

A CFG is in **Greibach normal form** (GNF) if every production has the form

$$A \rightarrow a\alpha$$

with $a \in T$ and $\alpha \in V^*$

Important properties of GNF :

- every nonempty CFL with non-empty strings only has a GNF grammar
- a grammar in GNF generates a string of length n in exactly n steps