Automata and Languages

Prof. Mohamed Hamada

Software Engineering Lab.
The University of Aizu
Japan

Today's Topics

- Chomsky Normal Form (CNF)
- Context free grammar to CNF
- Griebach Normal Form (GNF)
- Context free grammar to GNF
- Context Sensitive Grammar
- Relationship between Grammars
- Grammar Applications

Normal Forms

Chomsky Normal Form

Griebach Normal Form

Chomsky Normal Form CNF

Even though we can’t get every grammar into right-linear form, or *in general* even get rid of ambiguity, there is an especially simple form that general CFG’s can be converted into:

Chomsky Normal Form CNF

Chomsky Normal Form

Definition: A CFG is in **Chomsky normal form** if and only if all production rules are of the form

$$A \rightarrow BC$$

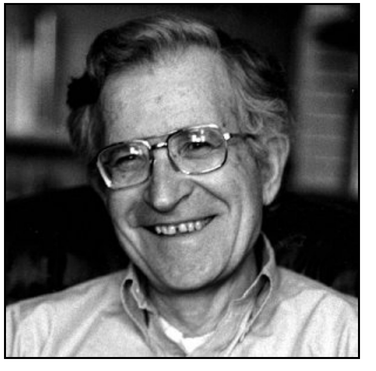
or $\quad A \rightarrow x$

with variables $A,B,C \in V$ and $x \in T$.

(Sometimes rule $S \rightarrow \lambda$ is also allowed.)

CFGs in CNF can be parsed in time $O(|w|^3)$.

Named after Noam Chomsky who in the 60s made seminal contributions to the field of theoretical linguistics. (cf. Chomsky hierarchy of languages).



Chomsky Normal Form CNF

Noam Chomsky came up with an especially simple type of context free grammars which is able to capture all context free languages.

Chomsky's grammatical form is particularly useful when one wants to prove certain facts about context free languages. This is because assuming a much more restrictive kind of grammar can often make it easier to prove that the generated language has whatever property you are interested in.

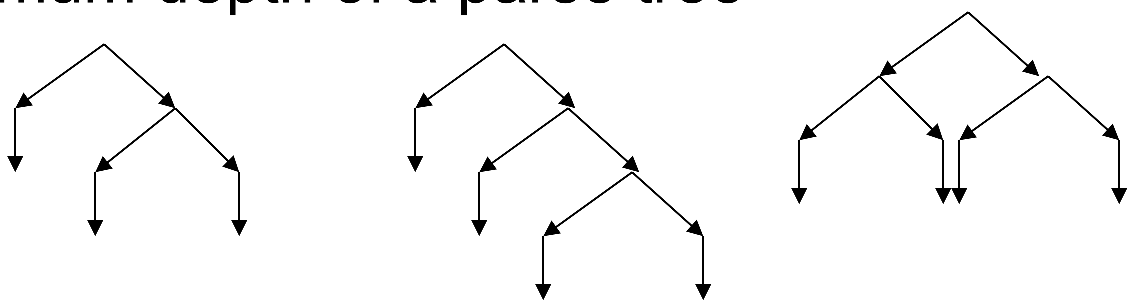
Chomsky Normal Form CNF

Significance of CNF

- Length of derivation of a string of length n in CNF = $(2n\text{-}1)$

  (*Cf*. Number of nodes of a strictly binary tree with n-leaves)

- Maximum depth of a parse tree = n $\lceil \log_2 n \rceil + 1$
- Minimum depth of a parse tree =



Chomsky Normal Form CNF

A CFG is said to be in ***Chomsky Normal Form*** if every rule in the grammar has one of the following forms:

$A \rightarrow BC$ (dyadic variable productions)

$A \rightarrow a$ (unit terminal productions)

$S \rightarrow \lambda$ (λ for empty string sake only)

where $B, C \in V - \{S\}$

Where S is the start variable, A,B,C are variables and a is a terminal. Thus empty string λ may only appear on the right hand side of the start symbol and other RHS are either 2 variables or a single terminal.

Chomsky Normal Form CNF

CFG→ CNF

- *Theorem:* There is an algorithm to construct a grammar G' in CNF that is *equivalent* to a CFG G.

Chomsky Normal Form CNF

CFG→ CNF: Construction

- Obtain an equivalent grammar that does not contain λ-rules, chain rules, and useless variables.
- Apply following conversion on rules of the form: $A \to bBcC$

$$A \to PQ \qquad P \to b$$

$$Q \to BR \qquad R \to WC$$

$$W \to c$$

Chomsky Normal Form CNF

CFG→ CNF: Construction

Converting a general grammar into Chomsky Normal Form works in four steps:

1. Ensure that the start variable doesn't appear on the right hand side of any rule.
2. Remove all λ-rules productions, except from start variable.
3. Remove unit variable productions of the form $A \rightarrow B$ where A and B are variables.
4. Add variables and dyadic variable rules to replace any longer non-dyadic or non-variable productions

Chomsky Normal Form CNF

CFG→ CNF: Example 1

Let’s see how this works on the following example grammar:

S→λ | a | b | aSa | bSb

Chomsky Normal Form CNF

CFG→ CNF: Example 1

1. Start Variable

Ensure that start variable doesn't appear on the right hand side of any rule.

S'→S

S→λ | a | b | aSa | bSb

Chomsky Normal Form CNF

CFG→ CNF: Example 1

2. Remove λ-rules

Remove all λ productions, except from start variable.

S'→S | λ

S→λ | a | b | aSa | bSb | aa | bb

Chomsky Normal Form CNF

CFG→ CNF: Example 1

3. Remove variable units

Remove unit variable productions of the form $A \rightarrow B$.

S'→S | λ | a | b | aSa | bSb | aa | bb

S→λ | a | b | aSa | bSb | aa | bb

Chomsky Normal Form CNF

CFG→ CNF: Example 1

4. Longer production rules

Add variables and dyadic variable rules to replace any longer productions.

S’→ λ | a | b | aSa | bSb | aa | bb **AB|CD|AA|CC**

S→a | b | aSa | bSb | aa | bb **AB|CD|AA|CC**

A→ a

B→ SA

C→ b

D→SC

Chomsky Normal Form CNF

CFG→ CNF: Example 1

5. Result

CFG

$S \to \lambda \mid a \mid b \mid aSa \mid bSb$

CNF

$S' \to \lambda \mid a \mid b \mid AB \mid CD \mid AA \mid CC$

$S \to a \mid b \mid AB \mid CD \mid AA \mid CC$

$A \to a$

$B \to SA$

$C \to b$

$D \to SC$

Chomsky Normal Form CNF

Exercise

- Write into Chomsky Normal Form the CFG:

$S \to aA|aBB$
$A \to aaA|\lambda$
$B \to bC|bbC$
$C \to B$

Chomsky Normal Form CNF

Answer

$S \rightarrow aA|aBB$
$A \rightarrow aaA|\lambda$
$B \rightarrow bC|bbC$
$C \rightarrow B$

- (1): First you remove the λ-productions ($A \Rightarrow \lambda$):

$S \rightarrow aA|aBB|a$
$A \rightarrow aaA|aa$
$B \rightarrow bC|bbC$
$C \rightarrow B$

Chomsky Normal Form CNF

Answer

•(2): Next you remove the unit-productions from:

$S \rightarrow aA|aBB|a$

$A \rightarrow aaA|aa$

$B \rightarrow bC|bbC$

$C \rightarrow B$

•Removing $C \rightarrow B$, we have to include the $C \Rightarrow^* B$ possibility, which can be done by substitution and gives:

$S \rightarrow aA|aBB|a$

$A \rightarrow aaA|aa$

$B \rightarrow bC|bbC$

$C \rightarrow bC|bbC$

Chomsky Normal Form CNF

Answer

(3): Next, we determine the useless variables in

$S \rightarrow aA|aBB|a$

$A \rightarrow aaA|aa$

$B \rightarrow bC|bbC$

$C \rightarrow bC|bbC$

The variables B and C can not terminate and are therefore useless. So, removing B and C gives:

$S \rightarrow aA|a$

$A \rightarrow aaA|aa$

Chomsky Normal Form CNF

Answer

(4): To make the CFG in Chomsky normal form, we have to introduce terminal producing variables for

$S \rightarrow aA|a$

$A \rightarrow aaA|aa,$

- which gives

$S \rightarrow X_a A|a$

$A \rightarrow X_a X_a A|X_a X_a$

$X_a \rightarrow a.$

Chomsky Normal Form CNF

Answer

(5): Finally, we have to 'chain' the variables in

$S \rightarrow X_a A | a$

$A \rightarrow X_a X_a A | X_a X_a$

$X_a \rightarrow a,$

- which gives

$S \rightarrow X_a A | a$

$A \rightarrow X_a A_2 | X_a X_a$

$A_2 \rightarrow X_a A$

$X_a \rightarrow a.$

Griebach Normal Form GNF

- A CFG is in *Griebach Normal Form* if each rule is of the form

$$A \rightarrow aA_1A_2\ldots A_n$$

$$A \rightarrow a$$

$$S \rightarrow \lambda$$

$$\text{where } A_i \in V - \{S\}$$

Griebach Normal Form GNF

- The size of the equivalent GNF can be large compared to the original grammar.
  - Next Example CFG has 5 rules, but the corresponding GNF has 24 rules!!

- Length of the derivation in GNF
  = Length of the string.
- GNF is useful in relating CFGs ("generators") to pushdown automata ("recognizers"/"acceptors").

Griebach Normal Form GNF

CFG→ GNF

- *Theorem:* There is an algorithm to construct a grammar G’ in GNF that is *equivalent* to a CFG G.

Griebach Normal Form GNF

CFG→ GNF: Example

$$A \rightarrow BC$$
$$B \rightarrow CA \mid b$$
$$C \rightarrow AB \mid a$$

$$A \rightarrow BC$$
$$B \rightarrow CA \mid \overleftrightarrow{b}$$
$$C \rightarrow \underline{BC}B \mid a$$

$$C \rightarrow \underline{\underline{CA}CB} \mid \underline{bCB} \mid a$$

$$C \rightarrow (\underline{bCB} \mid \underline{a})R$$
$$\mid \underline{bCB} \mid \underline{a}$$
$$R \rightarrow \underline{\overleftrightarrow{A}CBR} \mid \underline{\overleftrightarrow{A}CB}$$

$$C \rightarrow bCBR \mid aR \mid bCB \mid a$$
$$B \rightarrow bcBRA \mid aRA$$
$$\mid bCBA \mid aA \mid \overleftrightarrow{b}$$
$$A \rightarrow bcBRAC \mid aRAC$$
$$\mid bCBAC \mid aAC \mid bC$$
$$R \rightarrow \overleftrightarrow{(bCBRAC \mid \ldots \mid bC)}(CBR \mid CB)$$

Context Sensitive Grammar

An even more general form of grammars exists. In general, a non-context free grammar is one in which whole mixed variable/terminal substrings are replaced at a time. For example with $\Sigma = \{a,b,c\}$ consider:

$S \rightarrow \lambda \mid ASBC$	$aB \rightarrow ab$
$A \rightarrow a$	$bB \rightarrow bb$
$CB \rightarrow BC$	$bC \rightarrow bc$
	$cC \rightarrow cc$

For technical reasons, when length of LHS always ≤ length of RHS, these general grammars are called ***context sensitive***.

Context Sensitive Grammar (CSG)

Example

Find the language generated by the CSG:

$S \rightarrow \lambda \mid ASBC$

$A \rightarrow a$

$CB \rightarrow BC$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

Context Sensitive Grammar (CSG)

Example

Answer is $\{a^n b^n c^n\}$.

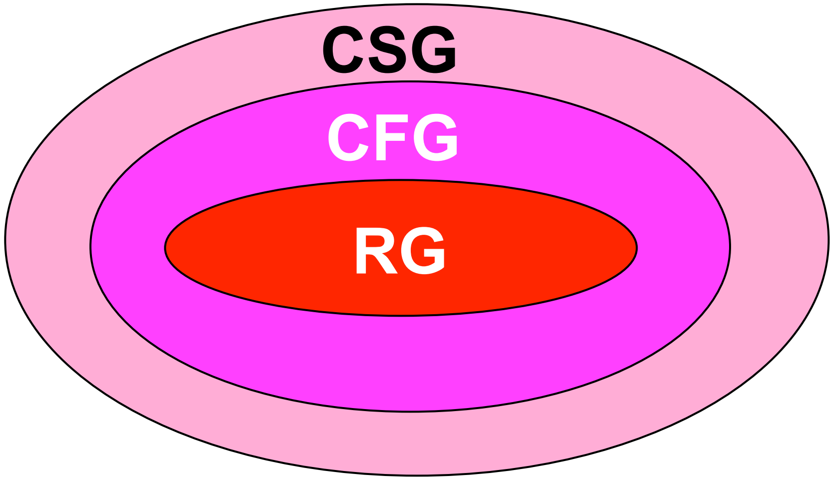
In a future class we'll see that this language is not context free. Thus perturbing context free-ness by allowing context sensitive productions expands the class.

Relations between Grammars

So far we studied 3 grammars:

1. Regular Grammars (RG)
2. Context Free Grammars (CFG)
2. Context Sensitive Grammars (CSG)

The relation between these 3 grammars is as follow:



Grammar Applications

Programming Languages

Programming languages are often defined as Context Free Grammars in **Backus-Naur Form** (**BNF**).

Example:

<if_statement> ::= **IF** <expression><then_clause><else_clause>
<expression> ::= <term> | <expression>+<term>
<term> ::= <factor>|<term>*<factor>

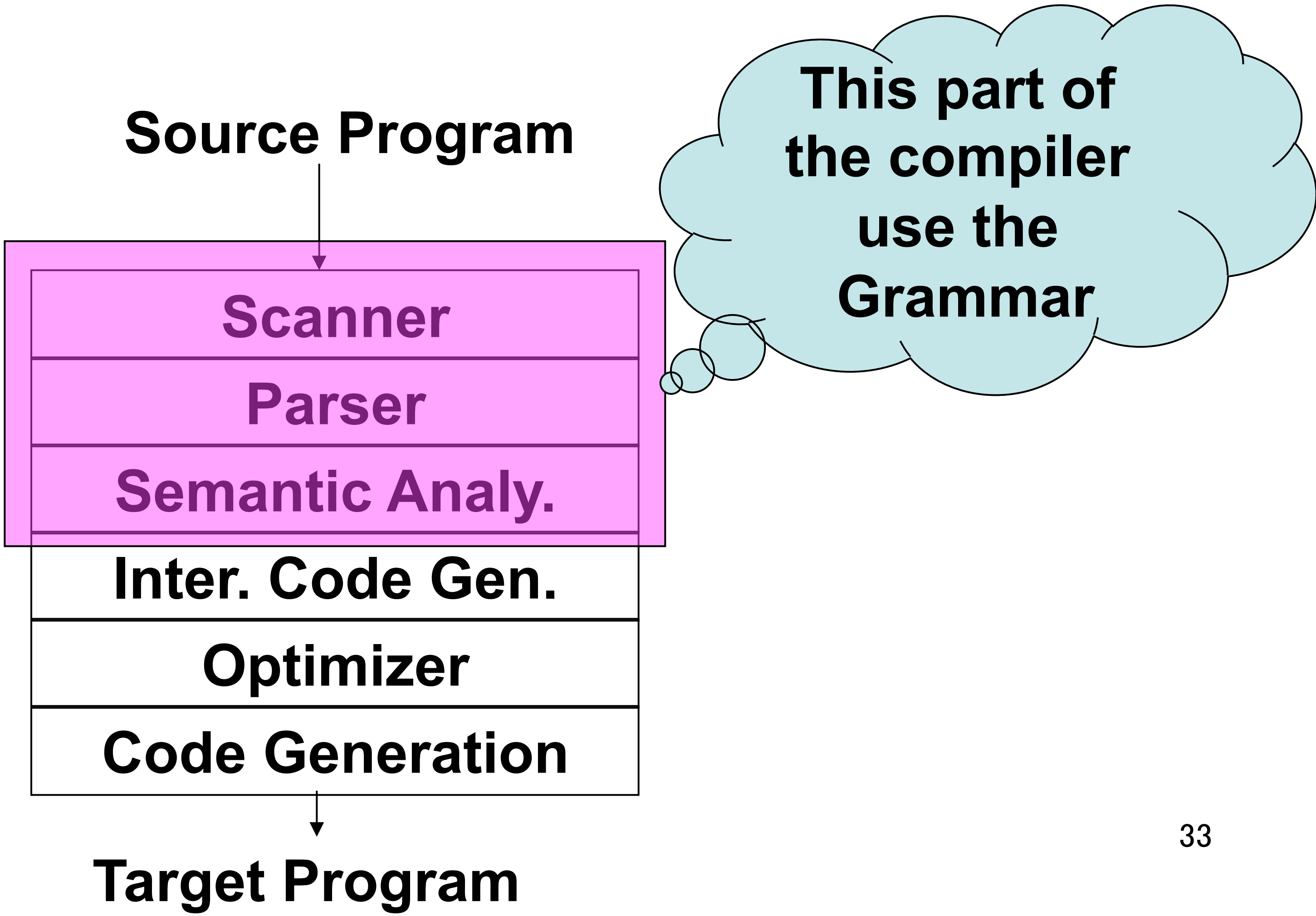
The variables as indicated by <a variable name>
The arrow → is replaces by ::=
Here, **IF**, + and * are terminals.

"Syntax Checking" is checking if a program is an element of the CFG of the programming language.

Grammar Applications

Compiler Syntax Analysis

Compiler:



Applications of CFG

Parsing is where we use the theory of CFGs.

The theory is especially relevant when dealing with **Extensible Markup Language** (**XML**) files and their corresponding **Document Type Definitions** (DTDs).

Document Type Definitions define the grammar that the XML files have to adhere to. Validating XML files equals parsing it against the grammar of the DTD.

The nondeterminism of NPDAs can make parsing slow. What about deterministic PDAs?