Computer Science 513

Chomsky Hierarchy IV: Normal Forms and Parsing

Instructor: Wayne Eberly

Department of Computer Science
University of Calgary

Lecture #24

Goals for Today

Goals for Today:

- Description of a process to convert an arbitrary context-free grammar into a grammar, in Chomsky normal form, with the same language
- This will be continued in the lecture presentation, which will concern testing membership, and parsing, in regular and context-free grammars.

Chomsky Normal Form

Recall, from Lecture #22, that a context-free grammar $G = (V, \Sigma, \Pi, S)$ is in ***Chomsky Normal Form*** if every rule in Π is in one of the forms

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

where $a \in \Sigma$ and $A, B, C \in V$— except that neither B nor C can be the start variable S.

One more rule is allowed: If the empty string is in the language of the grammar then the rule

$$S \to \lambda$$

can (and, indeed, *must*) also be included.

Chomsky Normal Form

- Claim #3, from Lecture #22, asserts that if

$$G = (V, \Sigma, \Pi, S)$$

is a context-free grammar, then there exists a context-free grammar

$$\widehat{G} = (\widehat{V}, \Sigma, \widehat{\Pi}, \widehat{S})$$

in Chomsky normal form such that $L(\widehat{G}) = L(G)$.

- These notes describe ***construction*** that can be used to obtain such a grammar, $\widehat{G}$, from a given context-free grammar G.
- The supplemental document provides proofs of claims needed to establish the correctness of this construction.

An Ongoing Example

Consider a context-free grammar $G = (V, \Sigma, \Pi, S)$ where $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, $V = \{S, A, B\}$, S is the start variable, and Π includes the rules

$$\begin{array}{lcl} S & \to & ASA \mid \mathtt{a}B \\ A & \to & S \mid B \\ B & \to & \mathtt{b} \mid \lambda \end{array}$$

As the steps in the construction (whose correctness proves the claim) are described, they will be applied to this example grammar.

Addition of a New Start Variable

- One of the conditions in the definition of "Chomsky Normal Form" is that the start variable does not appear as part of the right hand side of any rule.
- This condition can be achieved by adding a new start variable, S_0 (that is not already included in the set V of variables), adding this to V, and adding the single new rule

$$S_0 \to S$$

where S was the start variable in the original grammar.

Why This Works

- It follows by the definition of the new grammar that the start variable, S_0, does not appear in the right hand side of any rule.
- Any derivation of a string (in $\Sigma^\star$) in the original grammar can be turned into a derivation in the new grammar by inserting a use of the rule $S_0 \to S$ at the beginning.
- Any derivation of a string in $\Sigma^\star$ in the new grammar must have positive length and start with the use of the rule $S_0 \to S$. The variable S_0 never reappears, and deleting the application of the first rule produces a derivation of the string from the start variable in the original grammar.
- Thus the new grammar has the same language as the original one.

Application to the Example

When this construction is applied to the example, we get a grammar ($G = V, \Sigma, \Pi, S_0$) where $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ as before, $V = \{S_0, S, A, B\}$, S_0 is now the start variable, and Π includes the rules

$$\begin{array}{lcl} S_0 & \rightarrow & S \\ S & \rightarrow & ASA \mid \mathtt{a}B \\ A & \rightarrow & S \mid B \\ B & \rightarrow & \mathtt{b} \mid \lambda \end{array}$$

Ensuring That Rules are Short

The grammar might still include rules whose right hand sides are strings of symbols over the alphabet $V \cup \Sigma$ with length at least three.

- Each rule

$$A \to \mu_1 \mu_2 \dots \mu_k$$

where $A \in V$, $k \geq 3$, and $\mu_1, \mu_2, \dots, \mu_k \in (V \cup \Sigma)$, should be replaced by $k - 1$ rules

$$A \to \mu_1 C_1,\ C_1 \to \mu_2 C_2, \dots, C_{k-3} \to \mu_{k-2} C_{k-2},\ C_{k-2} \to \mu_{k-1} \mu_k$$

where $C_1, C_2, \dots, C_{k-2}$ are ***brand new*** variables that are not used anywhere else — we have a different set of such new variables for each of these long rules that must be replaced.

Why This Works

- It is still true that $A \Rightarrow^\star \mu_1\mu_2\ldots\mu_k$, since the sequence of new rules can be used to produce a derivation of $\mu_1\mu_2\ldots\mu_k$ from A.
- Since the variables $C_1, C_2, \ldots, C_{k-2}$ are new and not used anywhere else, the only way to use *any* of them is to use *all* of them, in order (so that C_1 gets introduced, is replaced by C_2, C_3, and so on). Thus these rules cannot be used to derive anything that could not be derived before.

See the supplemental document for a more formal (and proof) of the correctness of this step and, indeed, the correctness of the entire process being described.

Application To the Example

When this construction is applied to the example, we get a grammar ($G = V, \Sigma, \Pi, S_0$) where $\Sigma = \{\mathtt{a}, \mathtt{b}\}$ as before, $V = \{S_0, S, A, B, C_1\}$, S_0 is now the start variable, and Π includes the rules

$$\begin{array}{rcl} S_0 & \rightarrow & S \\ S & \rightarrow & AC_1 \mid \mathtt{a}B \\ C_1 & \rightarrow & SA \\ A & \rightarrow & S \mid B \\ B & \rightarrow & \mathtt{b} \mid \lambda \end{array}$$

Eliminating λ-Rules

- A λ-***rule*** is a rule with the form

$$A \to \lambda$$

- One the conditions in the definition of "Chomsky Normal Form" is that a λ-rule $A \to \lambda$ can *only* appear in the grammar if A is the start variable.
- ***The Tricky Part:*** When we delete any other λ-rule, $A \to \lambda$, we must also modify rules

$$B \to A,\ B \to \mu A,\ B \to A\mu,\ B \to AA$$

— where $\mu \neq A$ — that have at least one occurrence of A on the right hand side.

Since right hand sides of rules have length at most two, there are no other cases to consider.

Eliminating λ-Rules

When eliminating the λ-rule $A \to \lambda \ldots$

- Any rule $B \to A$ (where $B \in V$ and $B \neq A$) should be replaced by
  - A ***pair*** of rules $B \to A$, $B \to \lambda$ if the λ-rule $B \to \lambda$ ***has not*** been eliminated already.
  - A ***single*** rule $B \to A$ otherwise.
- Any rule $B \to \mu A$ (where $\mu \neq A$) should be replaced by a ***pair of rules*** $B \to \mu A$, $B \to \mu$
- Any rule $B \to A\mu$ (where $\mu \neq A$) should be replaced by a ***pair of rules*** $B \to A\mu$, $B \to \mu$
- Any rule $B \to AA$ should be replaced by
  - ***Three*** rules $B \to AA$, $B \to A$, $B \to \lambda$ if the λ-rule $B \to \lambda$ ***has not*** been eliminated already.
  - ***Two*** rules $B \to AA$, $B \to A$, otherwise.

Why This Works

It is possible to prove the following.

- For every variable B, every variable C such that a λ-rule $C \to \lambda$ has already been eliminated, and every symbol $\mu \in V \cup \Sigma$, if either $B \to \mu C$ or $B \to C\mu$ is a rule in G then $B \to \mu$ is also a rule in G.
- For every variable B such that a rule $B \to \lambda$ ***has not*** been eliminated already, and for every variable C such that a rule $C \to \lambda$ ***has*** been eliminated already, if $B \to C$ is a rule in G then $B \to \lambda$ is also a rule in G.

Why This Works

Using the above, the following can also be proved: For each variable C,

- let L_C the the set of strings in $\omega \in \Sigma^\star$ such that $C \Rightarrow^\star \omega$ ***before*** the rule $A \to \lambda$ has been eliminated, and
- let $\widehat{L}_C$ be the set of strings $\omega \in \Sigma^\star$ such that $C \Rightarrow^\star \omega$ ***after*** the rule $A \to \lambda$ has been eliminated.

Then the following is true.

- If $A = C$ or the λ-rule $C \to \lambda$ has already been eliminated then either $\widehat{L}_C = L_C$ or $\lambda \notin \widehat{L}_C$ and $L_C = \widehat{L}_C \cup \{\lambda\}$.
- If $A \neq C$ and the λ-rule $C \to \lambda$ has not been eliminated before this then $\widehat{L}_C = L_C$.

The λ-rule $S_0 \to \lambda$ never gets eliminated if S_0 is the (new) start variable, since this rule is allowed. It follows that the language of the ***grammar*** never gets changed.

Why This Works

Since a λ-rule never gets added back again after it has been eliminated, all the λ-rules except $S_0 \to \lambda$ (if it exists) will be eliminated after at most $|V| - 1$ applications of the above rule — at most once for every variable $A \in V \setminus \{S_0\}$.

It follows from this — and an examination of the new rules that get added — that the following is true.

- The language of the grammar is still the language L.
- The start variable does not appear on the right hand side of any rule.
- No rule has a right hand side with length more than two.
- The grammar does not include any λ-rules except, possibly, the rule $S_0 \to \lambda$.

Application To the Example

First the λ-rule $B \to \lambda$ must be eliminated. This produces the grammar

$$\begin{array}{rcl} S_0 & \to & S \\ S & \to & AC_1 \mid \mathrm{a}B \mid \mathrm{a} \\ C_1 & \to & SA \\ A & \to & S \mid B \mid \lambda \\ B & \to & \mathrm{b} \end{array}$$

Application To the Example

Next the (newly added) λ-rule $A \to \lambda$ must be eliminated. This produces the grammar

$$\begin{array}{lcl} S_0 & \to & S \\ S & \to & AC_1 \mid C_1 \mid \mathtt{a}B \mid \mathtt{a} \\ C_1 & \to & SA \mid S \\ A & \to & S \mid B \\ B & \to & \mathtt{b} \end{array}$$

Since this grammar does not include any λ-rules, this part of the process is now complete.

Eliminating Unit Rules

- A ***unit rule*** is a rule with the form

$$A \to B$$

  where A and B are both variables.

- Grammars in Chomsky Normal Form do not have any unit rules.

- ***The Tricky Part:*** Adding new rules, when a unit rule is removed, to ensure that the language of the grammar is not changed — and understanding why this works!

  When eliminating the rule $A \to B$ you must consider existing rules $B \to C$ (where C is a variable), $B \to \sigma$ (where $\sigma \in \Sigma$) or $B \to \mu_1 \mu_2$ (where $\mu_1, \mu_2 \in (V \cup \Sigma)$). No other rules with B on the left hand side are possible.

Eliminating Unit Rules

When eliminating a unit rule $A \to B$: If $A = B$, simply delete the (useless) rule. Otherwise...

- If the grammar also includes a rule $B \to C$ ...
  - Add the rule $A \to C$ if this unit rule ***has not*** been eliminated already and $A \neq C$.
  - Do not add any new rules if the rule $A \to C$ ***has*** been eliminated already or if $A = C$.
- If the grammar also includes a rule $B \to \sigma$, where $\sigma \in \Sigma$, then add the rule $A \to \sigma$.
- If the grammar also includes a rule $B \to \mu_1 \mu_2$, where $\mu_1, \mu_2 \in (V \cup \Sigma)$, then add the rule $A \to \mu_1 \mu_2$.

Repeat this process until no unit rules are left in the grammar.

Why This Works

Understanding why this works is tricky.

- For each variable $C \in V$ let L_C be the set of strings $\omega \in \Sigma^\star$ such that $C \Rightarrow^\star \omega$.
- It is possible to prove, by induction on the number of unit rules that have been eliminated already — and an examination of the way that unit rules get replaced — that L_C is not changed by this process, for any variable $C \in V$.
- This is ***not*** obvious, because *some* derivations $C \Rightarrow^\star \omega$ do not get "replaced" when the grammar is changed: They just disappear, instead

Why This Works

- This can be proved by proving something else, along the way: If $A \in V$, $\omega \in \Sigma^\star$, and $A \Rightarrow^\star \omega$, then there is always a derivation of ω from A that does not include a derivation of C from B, for any variables $B, C \in V$ such that the unit rule $B \to C$ has been deleted before this.
- The supplemental document includes a proof that this is true and uses it to show that the language of the grammar is not changed.

Application To the Example

"What happens next" depends on the order that we choose to use to eliminate unit rules. Let us start by eliminating the unit rule $S_0 \to S$. This produces the grammar

$$\begin{array}{lcl} S_0 & \to & AC_1 \mid C_1 \mid \mathtt{a}B \mid \mathtt{a} \\ S & \to & AC_1 \mid C_1 \mid \mathtt{a}B \mid \mathtt{a} \\ C_1 & \to & SA \mid S \\ A & \to & S \mid B \\ B & \to & \mathtt{b} \end{array}$$

Application To the Example

Let us continue by eliminating the unit rule $S_0 \to C_1$. This produces the grammar

$$\begin{array}{lcl} S_0 & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ S & \to & AC_1 \mid C_1 \mid \mathtt{a}B \mid \mathtt{a} \\ C_1 & \to & SA \mid S \\ A & \to & S \mid B \\ B & \to & \mathtt{b} \end{array}$$

Note: The unit rule $S_0 \to S$ did not get introduced, even though the grammar included the rule $C_1 \to S$, because the unit rule $S_0 \to S$ had already been eliminated.

Application To the Example

Eliminating the until rule $S \to C_1$ produces the following.

$$\begin{array}{rcl} S_0 & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ S & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ C_1 & \to & SA \mid S \\ A & \to & S \mid B \\ B & \to & \mathtt{b} \end{array}$$

Note: The "useless" rule $S \to S$ did not get introduced, even though the grammar included the rule $C_1 \to S$.

Application To the Example

Eliminating the unit rule $C_1 \to S$ produces the following.

$$\begin{array}{rcl} S_0 & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ S & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ C_1 & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ A & \to & S \mid B \\ B & \to & \mathtt{b} \end{array}$$

Note: It was not necessary to add a "second copy" of the rule $C_1 \to SA$ here.

Application To the Example

Eliminating the unit rule $A \to S$ produces the following.

$$
\begin{array}{rcl}
S_0 & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\
S & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\
C_1 & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\
A & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \mid B \\
B & \to & \mathtt{b}
\end{array}
$$

Application To the Example

Finally, eliminating the unit rule $A \to B$ produces the following.

$$\begin{array}{rcl} S_0 & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ S & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ C_1 & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \\ A & \to & AC_1 \mid SA \mid \mathtt{a}B \mid \mathtt{a} \mid \mathtt{b} \\ B & \to & \mathtt{b} \end{array}$$

Since there are no unit rules left, this stage in the process is now complete too.

Making Sure That Terminals are Properly Handled

- At this point, the only rules that the grammar might include, that are not allowed in grammars in Chomsky Normal Form, are rules

$$A \to \mu_1 \mu_2$$

where $A \in V$, $\mu_1, \mu_2 \in (V \cup \Sigma)$, and where one (or both) of μ_1 and μ_2 belongs to Σ.

For every such rule...

  - If $\mu_1 \in \Sigma$ then add a new variable V_{μ_1} (if it has not already been introduced), replace μ_1 with V_{μ_1} in this rule, and add the rule $V_{\mu_1} \to \mu_1$ to the grammar.
  - If $\mu_2 \in \Sigma$ then add a new variable V_{μ_2} (if it has not already been introduced), replace μ_2 with V_{μ_2} in this rule, and add the rule $V_{\mu_2} \to \mu_2$ to the grammar.

Why This Works

- It should not be too hard to see that the set of strings L_A including strings $\omega \in \Sigma^\star$ such that $A \Rightarrow^\star \omega$ is not changed, for any variable A that belonged to the grammar before this step got performed.
- In particular, this is true for the variable S_0 (the new start variable) — so that the resulting grammar still has language L.
- You should also be able to confirm that — at last — the resulting grammar is in Chomsky Normal Form!

Application To the Example

Each of the rules $S_0 \to \mathtt{a}B$, $S \to \mathtt{a}B$, $C_1 \to \mathtt{a}B$, and $A \to \mathtt{a}B$ must be replaced — by adding a new variable V_a, along with rules $S_0 \to V_aB$, $S \to V_aB$, $C_1 \to V_aB$, $A \to V_aB$ — and one more rule, $V_a \to \mathtt{a}$.

The resulting grammar is as follows.

$$
\begin{array}{rcl}
S_0 & \to & AC_1 \mid SA \mid V_aB \mid \mathtt{a} \\
S & \to & AC_1 \mid SA \mid V_aB \mid \mathtt{a} \\
C_1 & \to & AC_1 \mid SA \mid V_aB \mid \mathtt{a} \\
A & \to & AC_1 \mid SA \mid V_aB \mid \mathtt{a} \mid \mathtt{b} \\
B & \to & \mathtt{b} \\
V_a & \to & \mathtt{a}
\end{array}
$$

Concluding Remarks

- Textbooks and other references sometimes give ***very different*** processes to convert a grammar into Chomsky Normal Form — so you should not expect all references to describe the same process.
- The other methods generally replace λ-rules and unit rules in considerably different ways.
- These other methods are generally trickier to ***apply,*** but it is easier to prove that they are ***correct.***