SIMPLIFICATION of CFG and NORMAL FORMS

Chap. 6

Summary

- The issues of membership and parsing for CFL.
  - The exhaustive parsing is always possible, but inefficient and impractical – A need of more efficient methods in the real application, e.g.) compiler
- Cause: the unrestricted form of the right side of a production in CFG:

  $A \rightarrow x$, where $A \in V$ and $x \in (V \cup T)^*$.

→ restrict the right side without reducing the power of the grammar ?

Resolution: Let's show how we need not worry about certain types of productions.

- For a production with λ on the right in CFG, find an equivalent grammar without λ-*productions*.
- Remove *unit-productions* that have only a single variable on the right.
- Remove *useless productions* that cannot ever occur in the derivation of a string.

- *Normal Forms*
  - Grammatical forms that are very restricted.
  - But, any CFG has an equivalent in normal form.
  - One can define many kinds of normal forms; two of the most useful ones
    - Chomsky normal form and Greibach normal form.

Learning Objectives

- Simplify a Context Free Grammar (CFG) by removing *useless productions*.
- Simplify a CFG by removing *λ-productions*.
- Simplify a CFG by removing *unit-productions*.
- Determine whether or not a CFG is in *Chomsky Normal Form* (CNF).
- Transform a CFG into an equivalent grammar in Chomsky Normal Form (CNF).
- Determine whether or not a CFG is in *Greibach Normal Form* (GNF).
- Transform a CFG into an equivalent grammar in Greibach Normal Form (GNF).

Methods for Transforming Grammars

- The definition of a CFG imposes no restrictions on the right side of a production.
  - $A \rightarrow x$, where $A \in V$ and $x \in (V \cup T)^*$.
- In some cases, it is convenient to restrict the form of the right side of all productions.
- Simplifying a grammar involves *eliminating certain types of productions* while producing an equivalent grammar, but does not necessarily result in a reduction of the total number of productions.
- For simplicity, we focus on languages that do not include the empty string.
  - For a CFG G = (V, T, S, P), a new CFG G' for L(G') = L(G) – { λ }.
  - In G', V' = V $\cup$ {S_0} with P' = P $\cup$ {$S_0 \rightarrow S \mid \lambda$ }.

A Useful Substitution Rule

- Theorem 6.1: Let G=(V, T, S, P) be a CFG.
  P contains a production of the form $A \to x_1Bx_2$.
  Assume that A $\neq$ B and that $B \to y_1|y_2|\cdots|y_n$
  is the set of all productions in P that have B as the left side.
  Let G' = (V, T, S, P') be the grammar in which P' is constructed by deleting $A \to x_1Bx_2$ from P, and adding to it
  $$A \to x_1y_1x_2 \mid x_1y_2x_2 \mid\cdots\mid x_1y_nx_2.$$
  Then, $L(G') = L(G)$.

Proof)

- If A and B are distinct variables, a production of the form $A \to uBv$ can be replaced by a set of productions in which B is substituted by *all strings B* derives in one step.

A Useful Substitution Rule (cont.)

$L(G') = L(G)$.

Proof) $\leftarrow$) Suppose $w \in L(G)$, so $S \overset{*}{\Rightarrow}_G w$.

If $S \overset{*}{\Rightarrow}_G w$ doesn't involve a production $A \to x_1Bx_2$, then $S \overset{*}{\Rightarrow}_{G'} w$.

If it does, then look at the derivation the first time $A \to x_1Bx_2$ is used.

$S \overset{*}{\Rightarrow}_G u_1Au_2 \Rightarrow_G u_1x_1Bx_2u_2 \Rightarrow_G u_1x_1y_jx_2u_2$.

But with G', we get $S \overset{*}{\Rightarrow}_{G'} u_1Au_2 \Rightarrow_{G'} u_1x_1y_jx_2u_2$.

So, we reach the same sentential from with G and G'.

If $A \to x_1Bx_2$ is used again, we can repeat the argument.

So, by Math. induction on the *number of times the production is applied*, $S \overset{*}{\Rightarrow}_{G'} w$. Thus, if $w \in L(G)$, then $w \in L(G')$.

$\to$) Similarly, we can show that if $w \in L(G')$, then $w \in L(G)$.

A Useful Substitution Rule

- Example 6.1:

  Consider the grammar G = (V, T, A, P) where

  V = { *A, B* }, T = { *a, b, c* }, and productions

  A → *a* | *aaA* | *abBc*, B → *abbA* | *b* .

  By replacing *A* → *abBc* with two productions that replace *B* (in red), we get an equivalent grammar G' with productions *P'*

  A → *a* | *aaA* | *ababbAc* | *abbc*, *B* → *abbA* | *b* .

  The new grammar G' ≡ G.

  For w = *aaabbc*, A $\Rightarrow_G$ *aaA* $\Rightarrow_G$ *aaabBc* $\Rightarrow_G$ *aaabbc* *in G,*

  while A $\Rightarrow_{G'}$ *aaA* $\Rightarrow_{G'}$ *aaabbc* in *G'.*

Useless Productions

- Definition 6.1: Let G=(V, T, S, P) be a CFG.

  A variable A∈V is *useful* iff there is at least one $w \in L(G)$ s.t. $S \Rightarrow^* xAy \Rightarrow^* w$, with $x, y \in (V \cup T)^*$.

  i.e. it occurs in the derivation of at least one derivation.
- Otherwise, the variable and any productions in which it appears is considered *useless*.
- A variable is *useless* if:
  - *No terminal strings* can be *derived* from the variable.
  - The variable symbol can *not be reached* from S.
- Example 6.2: In the grammar below, *B* can never be reached from the start symbol *S* and is therefore considered useless and so is a production *B* → *bA.*

  S → *A*, *A* → *aA* | λ, *B* → *bA.*

Removing Useless Productions

Theorem 6.2: Let G=(V, T, S, P) be a CFG.

Then, there exists an equivalent grammar G'=(V', T', S, P') *without any useless symbol and production.*

Proof) Step 1: Construct an intermediate $G_1=(V_1, T_1, S, P_1)$ with the useful variables only.

1. Let V_1 be the set of *useful variables*: initially, $V_1 = \{S\}$.
2. Repeat for every $A \in V$,

   Add a variable A to V_1 if there is a production of the form $A \to x_1x_2..x_n, \quad \forall x_i \in V_1 \cup T$

   Until nothing else can be added to V_1.
3. Take P_1 by eliminating any productions from P containing variables not in V_1.

Removing Useless Productions (cont.)

Theorem 6.2: Let G=(V, T, S, P) be a CFG.

Then, there exists an equivalent grammar G'=(V', T', S, P') *without any useless symbol and production.*

Proof (cont.)) Step 2: Get the *final G'* from G_1.

4. Using a dependency graph from G_1,
   a) Identify and eliminate the variables that are *unreachable* from S. -- the final V'.
   b) Eliminate the productions involving those variables in a). – the final P'.
   c) Eliminate any terminal that doesn't occur in any production of P'. -- the final T'.

   So, G' doesn't contain any useless symbols or productions.

Removing Useless Productions (cont.)

Theorem 6.2: Let G=(V, T, S, P) be a CFG.

Then, there exists an equivalent grammar G'=(V', T', S, P') *without any useless symbol and production.*

Proof (cont.)) Step 3: Show that G are G' are equivalent, L(G)=L(G').

→) For each $w \in L(G)$, there is a derivation: $S \Rightarrow^* xAy \Rightarrow^* w$.

Since the construction of G' retains A and all associated productions, *P'* of *G'* makes the derivation $S \Rightarrow^*_{G'} xAy \Rightarrow^*_{G'} w$.

Thus, L(G) $\subseteq$ L(G').

←)

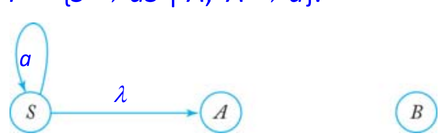
Since G' is constructed from G by the removal of productions, P' $\subseteq$ P.

Consequently, L(G') $\subseteq$ L(G). Therefore, L(G') = L(G).

Thus, G are G' are equivalent: G $\equiv$ G'. Q.E.D.

Example 6.3: Removing Useless Productions

- Consider the CFG G = (V, T, S, P) where V={S, A, B, C}, T={*a,b*}, and
  $P = \{\ S \to aS \mid A \mid C,\ \ A \to a,\ \ B \to aa,\ \ C \to aCb\ \}$.
- In step 2: Add variables A, B and S to V_1, so $V_1 = \{S, A, B\}$
  -- the set of variables that can lead to terminal string
- Step 3: Since *C* is useless, any production containing C is eliminated from P, so $P_1 = \{\ S \to aS \mid A,\ \ A \to a,\ \ B \to aa\ \}$.
- Step 4.(a): B is unreachable from S, so B is useless: $V_1 = \{S, A\} = V'$
- Step 4.(b): Any production containing B is eliminated from P_1.
  $P_1 = \{\ S \to aS \mid A,\ \ A \to a\} = P'$.
- Step 4.(c): Since the terminal *b* doesn't occur in any *P'*, eliminate it from T_1.
  Thus, the final equivalent G' = (V', T', S, P') is with
  $V' = \{S, A\},\ T' = \{a\},\ P' = \{S \to aS \mid A,\ A \to a\}$.



λ-Productions

- Definition 6.2:

  A production of a CFG of the form $A \rightarrow \lambda$ is called a *λ-production.*
  A variable A is called *nullable* if there is a sequence of derivations that produces λ from A, i.e. $A \Rightarrow^* \lambda$.
- If a grammar generates a language not containing λ, any λ-production can be removed.
- Example 6.4: In the grammar G below, S_1 is nullable:

  $S \rightarrow aS_1b, \quad S_1 \rightarrow aS_1b \mid \lambda.$

  Since the language $L(G) = \{a^nb^n \mid n \geq 1\}$ is λ-free, the λ-production $S_1 \rightarrow \lambda$ can be removed *after adding new productions* by substituting λ for S_1 where it occurs on the right. Thus,

  $S \rightarrow aS_1b \mid ab, \quad S_1 \rightarrow aS_1b \mid ab.$

Removing λ-Productions

Theorem 6.3: Let G be any CFG with $\lambda \notin L(G)$.
Then, there exists an equivalent CFG G'=(V, T, S, P') *without* λ-productions.

Proof) Step 1: Find V_N of all *nullable variables*.

1. Let V_N be the set of *nullable variables*: initially, $V_N = \varnothing$.
2. Repeat for all productions
   Add a variable A to V_N if there is a production of the forms: $A \rightarrow \lambda$ or
   $A \rightarrow A_1A_2 \ldots A_n$ where $A_i \in V_N$
   Until no further variables can be added to V_N
3. Eliminate λ-productions from P.
4. Add the new productions in which *nullable variables* are *replaced by* λ in all possible combinations. -- the new P'.

Step 2: Show that G are G' are equivalent. -- similar to Th^m. 6.2
The final G'=(V, T, S, P') is equivalent to G.

Example: Removing λ-Productions

- Example 6.5: Consider the CFG G with productions
  $S \to ABaC,\ A \to BC,\ B \to b \mid \lambda,\ C \to D \mid \lambda,\ D \to d.$
- In step 2: The nullable variables B, C, and A (in that order) are added to V_N. So, $V_N = \{B, C, A\}$.
- In step 3: λ-productions, $B \to \lambda$, $C \to \lambda$ are removed from P.
  $\{S \to ABaC,\ A \to BC,\ B \to b,\ C \to D,\ D \to d\}.$
- In step 4: the new productions replacing nullable symbols with λ in all possible combinations,
  $P' = \{ S \to ABaC \mid BaC \mid AaC \mid ABa \mid aC \mid Aa \mid Ba \mid a$
  $A \to BC \mid B \mid C,$
  $B \to b,$
  $C \to D,$
  $D \to d \ \}$ with $V_N = \{A, B, C\}$.

Unit-Productions

- Definition 6.3: A production of a CFG of the form
  $A \to B$ where $A, B \in V$ is called a *unit-production.*
- Unit-productions *increase* the *unnecessary complexity* to a grammar and can usually be removed by simple substitution.
- Theorem 6.4: Any CFG *without λ-productions* has an *equivalent* CFG without unit-productions.
- The procedure for eliminating unit-productions assumes that all λ-productions have been previously removed.

Removing Unit-Productions

1. Draw a dependency graph with an edge from A to B corresponding to every A $\rightarrow$ B production in the grammar.
2. Construct a new grammar that includes all the productions from the original grammar, except for the unit-productions.
3. Whenever there is a path from A to B in the dependency graph, replace B using the substitution rule from Theorem 6.1, but using only the productions in the new grammar.

Example: Removing Unit-Productions

- Example 6.6: Consider the grammar:

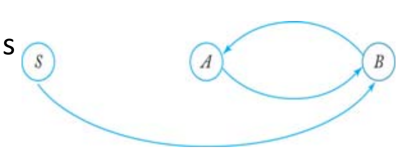
$S \rightarrow Aa \mid B$
$A \rightarrow a \mid bc \mid B$
$B \rightarrow A \mid bb$

The dependency graph contains paths from S to A, S to B, B to A, and A to B

- After removing unit-productions and adding the new productions (in red), the resulting grammar is

$S \rightarrow Aa \mid a \mid bc \mid bb$
$A \rightarrow a \mid bc \mid bb$
$B \rightarrow a \mid bc \mid bb$.

- The removal of the unit-productions has made *B* and the associated productions useless.



Simplification of Grammars

- Theorem 6.5: For any CFL that *does not include λ*, there exists a CFG *without useless, λ-, or unit-productions.*
- Since the removal of one type of production may introduce productions of another type, undesirable productions should be removed in the following order:
  1. Remove λ-productions.
  2. Remove unit-productions.
  3. Remove useless productions.

Chomsky Normal Form (CNF)

- In Chomsky Normal Form (CNF), the number of symbols on the right side of a production is strictly limited.
- Definition 6.4: A CFG is in *Chomsky Normal Form (CNF)* if all of its productions are of the form
  - $A \to BC$ *or*
  - $A \to a$ where $A, B, C \in V$, $a \in T$.
- Example 6.7: The grammar below

  $S \to AS \mid a$

  $A \to SA \mid b$ is in Chomsky Normal Form.
- But, $S \to AS|AAS$, $A \to SA|aa$ is not in CNF, since both $S \to AAS$ and $A \to aa$ violate the conditions

Transforming a CFG into Chomsky Normal Form (CNF)

For any CFG that does *not* generate λ, it is possible to find an equivalent grammar in CNF:

1. Copy any productions of the form $A \rightarrow a$.
2. For other productions containing a terminal symbol x on the right side, replace x with a variable X and add the production $X \rightarrow x$.
3. Introduce additional variables to reduce the lengths of the right sides of productions as necessary, replacing long productions with productions of the form $W \rightarrow YZ$ (W, Y, Z are variables).

Example: Conversion to CNF

- Example 6.8: Consider the CFG which is clearly not in Chomsky Normal Form

  $S \rightarrow ABa$

  $A \rightarrow aab$

  $B \rightarrow Ac$

- After replacing terminal symbols with new variables and adding new productions (in red), the resulting grammar is

  $S \rightarrow AC$, $\quad C \rightarrow BX$, $\quad X \rightarrow a$

  $A \rightarrow XD$, $\quad D \rightarrow XY$, $\quad Y \rightarrow b$

  $B \rightarrow AZ$, $\quad Z \rightarrow c$.

Greibach Normal Form (GNF)

- In Greibach Normal Form, there are restrictions on the positions of terminal and variable symbols
- Definition 6.5: A CFG is in *Greibach Normal Form (GNF)* if all productions have the form $A \to ax$ where $a \in T$, $x \in V^*$. i.e. the right side of any production consists of *single terminal followed by any number of variables.*
- Example 6.9: The grammar

  $S \to aAB \mid bBB \mid bB$
  $A \to aA \mid bB \mid b$
  $B \to b$ is in Greibach Normal Form.

Transforming a Grammar into GNF

- Theorem 6.7: For any CFG with $\lambda \notin L(G)$, it is possible to find an equivalent grammar in Greibach normal form.
- Example 6.10: Consider the grammar which is clearly not in GNF, $S \to abSb \mid aa$.
- After replacing terminal symbols with new variables and adding new productions (in red), the resulting grammar is

  $S \to aBSB \mid aA$
  $A \to a$
  $B \to b$