Computing Theory

COMP 147

Chomsky Normal Form

- Convenient to have grammars in a simplified form

- A CFG is said to be in Chomsky Normal Form if every production is of one of these two forms:
  1. A -> BC (body is two variables).
  2. A -> a (body is a single terminal).

Chomsky Normal Form

- Motivation:
  - Every string of length n can be derived in 2n - 1 steps
  - Makes proof of pumping lemma for CFL easier
  - Makes proof that every Machine has an equivalent PDA easier

Cleaning up Grammars- Eliminating Useless Symbols

Variables That Derive Nothing

- Consider: S -> AB, A -> aA | a, B -> AB
- Although A derives all strings of a's, B derives no terminal strings.
- Thus, S derives nothing, and the language is empty.

Example

In this CFG, which variable(s) (upper-case letters) do(es) not derive any terminal string?

S→ABC; A→ab; C→Bb; C→aAb; B→AB

1. S
2. B
3. B, S
4. A, B, S

Algorithm to Eliminate Variables That Derive Nothing

1. Discover all variables that derive terminal strings.

2. For all other variables, remove all productions in which they appear in either the head or body.

Example: Eliminate Variables

S -> AB | C, A -> aA | a, B -> bB, C -> c

- Basis: A and C are discovered because of A -> a and C -> c.
- Induction: S is discovered because of S -> C.
- Nothing else can be discovered.
- Result: S -> C, A -> aA | a, C -> c

Unreachable Symbols

- Another way a terminal or variable deserves to be eliminated is if it cannot appear in any derivation from the start symbol.

- Basis: We can reach S (the start symbol).

- Induction: if we can reach A, and there is a production A -> α, then we can reach all symbols of α.

Unreachable Symbols – (2)

- Algorithm: Remove from the grammar all symbols not discovered reachable from S and all productions that involve these symbols.

Example

In this CFG, S is the start symbol. Which symbols are unreachable?

S → AB; A → bc; B → aa; C → De; D → aa

1. e, C, D
2. a, e, C, D
3. C, D
4. e, D

Eliminating Useless Symbols

- A symbol is useful if it appears in some derivation of some terminal string from the start symbol.

- Otherwise, it is useless.
  Eliminate all useless symbols by:
  - Eliminate symbols that derive no terminal string.
  - Eliminate unreachable symbols.

Epsilon Productions

- We can almost avoid using productions of the form A -> **ε** (called **ε**-productions).
  - The problem is that **ε** cannot be in the language of any grammar that has no **ε**–productions.

- Theorem: If L is a CFL, then L-{**ε**} has a CFG with no **ε**-productions.

Nullable Symbols

- To eliminate **ε**-productions, we first need to discover the nullable symbols = variables A such that A =>* **ε**.

- Basis: If there is a production A -> **ε**, then A is nullable.

- Induction: If there is a production A -> α, and all symbols of α are nullable, then A is nullable.

Example: Nullable Symbols

S -> AB, A -> aA | **ε**, B -> bB | A

- Basis: A is nullable because of A -> **ε**.
- Induction: B is nullable because of B -> A.
- Then, S is nullable because of S -> AB.

Eliminating ε-Productions

- Key idea: turn each production
  A -> $X_1 \ldots X_n$ into a family of productions.

- For each subset of nullable X's, there is one production with those eliminated from the right side "in advance."
  - Except, if all X's are nullable (or the body was empty to begin with), do not make a production with **ε** as the right side.

Example: Eliminating ε-Productions

- S -> ABC, A -> aA | ε, B -> bB | ε, C -> ε

- A, B, C, and S are all nullable.
- New grammar:
- S -> ~~ABC~~ | AB |~~AC~~ |~~BC~~ | A | B |~~C~~
- A -> aA | a
- B -> bB | b

Note: C is now useless.

Unit Productions

- A unit production is one whose body consists of exactly one variable.

- These productions can be eliminated.

- Key idea: If A =>* B by a series of unit productions, and B -> α is a non-unit-production, then add production A -> α.

- Then, drop all unit productions.

Unit Productions – (2)

- Find all pairs (A, B) such that A =>* B by a sequence of unit productions only.

- Basis: Surely (A, A).

- Induction: If we have found (A, B), and B -> C is a unit production, then add (A, C).

Example

For the following CFG, find all pairs (A,B) such that $A \Rightarrow * B$ by a sequence of unit productions only:

$S \rightarrow AB|Aa;$

$A \rightarrow cD;$

$B \rightarrow aCb|C|A;$

$C \rightarrow D;$

$D \rightarrow a|b|c$

1)(B,C) (B,A) (C,D)
2) (B,C) (B,A) (B,D) (C,D)
3) (B,C) (B,A) (B,D) (C,D) (A,D)
4) (B,C) (B,A) (B,D) (C,D) (A,D)

Example

For the following CFG, find all pairs (A,B) such that $A \Rightarrow * B$ by a sequence of unit productions only:

$S \rightarrow AB|Aa$

$A \rightarrow cD$

$B \rightarrow aCb|C|A$

$C \rightarrow D$

$D \rightarrow a|b|c$

1)(B,C) (B,A) (C,D)

2) (B,C) (B,A) (B,D) (C,D)

3) (B,C) (B,A) (B,D) (C,D) (A,D)

4) (B,C) (B,A) (B,D) (C,D) (A,D)

$S \rightarrow AB|Aa$

$A \rightarrow cD$

$B \rightarrow aCb|cD|a|b|c$

$C \rightarrow a|b|c$

$D \rightarrow a|b|c$

Cleaning Up a Grammar

- Theorem: if L is a CFL, then there is a CFG for L – {**ε**} that has:
  - No useless symbols.
  - No **ε**-productions.
  - No unit productions.

- I.e., every body is either a single terminal or has length ≥ 2.

Cleaning Up a Grammar

- Proof: Start with a CFG for L.
- Perform the following steps in order:
  - Eliminate **ε**-productions.
  - Eliminate unit productions.
  - Eliminate variables that derive no terminal string.
  - Eliminate variables not reached from the start symbol.

Must be first. Can create unit productions or useless variables.

Chomsky Normal Form

- A CFG is said to be in Chomsky Normal Form if every production is of one of these two forms:
  - A -> BC (body is two variables).
  - A -> a (body is a single terminal).

- Theorem: If L is a CFL, then L – {**ε**} has a CFG in CNF.

CNF Theorem
Every CFG -> CNF

- Step 1: Add a new rule S_0 -> S (this will ensure the start variable does not appear on the RHS of any rule)
- Step 2: “Clean” the grammar, so every body is either a single terminal or of length at least 2.
  - Eliminate **ε**-productions.
  - Eliminate unit productions.
  - Eliminate variables that derive no terminal string.
  - Eliminate variables not reached from the start symbol.
- Step 3: For each body ≠a single terminal, make the right side all variables.
  - For each terminal a create new variable A_a and production A_a -> a.
  - Replace a by A_a in bodies of length ≥ 2.

(Note if **ε** is part of the language then we can add with S_0 -> S| **ε** instead)

Example: Step 3

- Consider production A -> BcDe.
- We need variables A_c and A_e. with productions A_c -> c and A_e -> e.
  - Note: you create at most one variable for each terminal, and use it everywhere it is needed.
- Replace A -> BcDe by A -> BA_cDA_e.

CNF Proof – Continued

- Step 3: Break right sides longer than 2 into a chain of productions with right sides of two variables.

- Example: A -> BCDE is replaced by
  A -> BF, F -> CG, and G -> DE.
  - F and G must be used nowhere else.

Example of Step 3 – Continued

- Recall A -> BCDE is replaced by
  A -> BF, F -> CG, and G -> DE.

- In the new grammar, A => BF => BCG => BCDE.

- More importantly: Once we choose to replace A by BF, we must continue to BCG and BCDE.
  - Because F and G have only one production.

Example - Putting it all together

S → ASA |aB
A→B|S
B→b|ε

1) Add a new start variable
S_0 → S
S → ASA |aB
A→B|S
B→b|ε

2a) Identify nullable symbols
B and A
eliminate ε-productions and fix productions

Resulting grammar
S_0 → S
S → ASA |aB | AS | SA | S | a
A→B|S
B→b

2a) S -> S can be removed

Resulting grammar
S_0 → S
S → ASA |aB | AS | SA | a
A→B|S
B→b

Example - Putting it all together

2b) Identify unit productions
(A,B), (A,S), (S_0,S)

eliminate unit production (A,B)
Resulting grammar
$S_0 \rightarrow S$
$S \rightarrow ASA \mid aB \mid AS \mid SA \mid a$
$A \rightarrow b \mid S$
$B \rightarrow b$

2b) Identify unit productions
(A,B), (A,S), (S_0,S)

eliminate unit production (A,S)
Resulting grammar
$S_0 \rightarrow S$
$S \rightarrow ASA \mid aB \mid AS \mid SA \mid a$
$A \rightarrow b \mid ASA \mid aB \mid AS \mid SA \mid a$
$B \rightarrow b$

2b) Identify unit productions
(A,B), (A,S), (S_0,S)

eliminate unit production (S_0,S)
Resulting grammar
$S_0 \rightarrow ASA \mid aB \mid AS \mid SA \mid a$
$S \rightarrow ASA \mid aB \mid AS \mid SA \mid a$
$A \rightarrow b \mid ASA \mid aB \mid AS \mid SA \mid a$
$B \rightarrow b$

2c and 2d: Check for variables that derive nothing and unreachable symbols (none)

Grammar is unchanged
$S_0 \rightarrow ASA \mid aB \mid AS \mid SA \mid a$
$S \rightarrow ASA \mid aB \mid AS \mid SA \mid a$
$A \rightarrow b \mid ASA \mid aB \mid AS \mid SA \mid a$
$B \rightarrow b$

Example - Putting it all together

Step3: Convert remaining rules by adding variables and rules

Grammar
$S_0 \rightarrow$ ASA |aB | AS | SA| a
S → ASA |aB | AS | SA | a
A→ b|ASA |aB | AS | SA | a
B→ b

Grammar
$S_0 \rightarrow$ DA |CB | AS | SA | a
S → DA | CB | AS | SA | a
A→ b|DA |CB | AS | SA | a
B→ b
D→ AS
C→ a