# CS311 Formal Language and Automata

Tingting Chen
Computer Science
Cal Poly Pomona

# Two questions

- Can we solve the membership problem for context-free languages? That is, can we develop a parsing algorithm for any context-free language?
- If so, can we develop an *efficient* parsing algorithm?
- Since CFG has no restrictions on the right side of a production, it is difficult to come up with generic algorithms.
- In many instances, it is desirable to place restrictions on the grammar.

# S-grammars

**Definition 5.4:** A context-free grammar $G = (V, T, S, P)$ is said to be a simple grammar or s-grammar if all of its productions are of the form

$$A \to ax,$$

where $A \in V$, $a \in T$, $x \in V^*$, and any pair $(A, a)$ occurs at most once in $P$.

Example: The following grammar is an s-grammar:

$$S \to aS \mid bSS \mid c$$

The following grammar is not an s-grammar. Why not?

$$S \to aS \mid bSS \mid aSS \mid c$$

# S-grammars

Let’s consider the grammar expressed by the following production rules: $S \rightarrow aS \mid bSS \mid c$

- Since G is an s-grammar, all rules have the form $A \rightarrow ax$.
- Assume that w = *abcc*.
- Due to the restrictive condition that any pair (A, a) may occur at most once in P, we know immediately the rule $S \rightarrow aS$, generated the *a* in *abcc*.

- Similarly, there is only one way to produce the *b* and the two *c*’s.
- So we can parse w in no more than |w| steps.

# Chomsky Normal Form

- There are other ways to limit the form a grammar can have.
- A context-free grammar in Chomsky Normal Form (CNF) has all of its rules restricted so that on the right-hand side of a production rule, there are no more than two symbols, either one terminal or two variables.
- This seems very restrictive, but actually every context-free grammar can be converted into Chomsky Normal Form.

# Chomsky Normal Form

**Definition 6.4:** A context-free grammar is in Chomsky Normal Form (CNF) if every production is one of these two types:

$A \to BC$

$A \to a$

where *A*, *B*, and *C* are variables and *a* is a terminal symbol.

# Chomsky Normal Form -- Example

Example:

$S \to ABa$

$A \to aab$

$B \to Ac$

Not a Chomsky Normal Form

Conversion to Chomsky Normal From

# Chomsky Normal Form -- Example

Conversion to Chomsky Normal From

1. Introduce variables for terminals $T_a$, $T_b$, $T_c$

from

$$S \to ABa$$
$$A \to aab$$
$$B \to Ac$$

to

$$S \to ABT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

# Chomsky Normal Form -- Example

Conversion to Chomsky Normal From

2. Introduce intermediate variable $V_1$

from

$$S \to ABT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

to

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

# Chomsky Normal Form -- Example

Conversion to Chomsky Normal From

3. Introduce intermediate variable $V_2$

from

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

to

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aV_2$$
$$V_2 \to T_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

# Conversion to Chomsky Normal Form –In general

- From any context-free grammar (which does not produce empty string) not in Chomsky Normal From, we can obtain an equivalent grammar in Chomsky Normal Form.

- First remove Nullable variables and unit productions.
- Then, for every symbol a, add production $T_a \rightarrow a$
  in productions, replace a with new variable $T_a$
- Replace any production $A \rightarrow C_1 C_2 \ldots C_n$ with
  $A \rightarrow C_1 V_1$, $V_1 \rightarrow C_2 V_2$, …, $V_{n-2} \rightarrow C_{n-1} C_n$
  new intermediate variables: $V_1$ $V_2$…, $V_{n-2}$

# Unit Productions $A \rightarrow B$, Nullable Productions $A \rightarrow \lambda$

- If a string w does not belong to L(G), then in the process of making the decision, we hope there is no such production rules in the grammar that $A \rightarrow B$, $A \rightarrow \lambda$.
- When each production rule is applied, it is guaranteed to either increase the length of the sentential form generated or to increase the number of terminals in the sentential form.
- As soon as a sentential form is generated that is longer than our string, w, we can abandon any attempt to generate w from this sentential form.

# Removing $A \to \lambda$ in grammar

- Example Grammar:

  $S \to aMb$

  $M \to aMb$

  $M \to \lambda$

  After removing $M \to \lambda$, equivalent grammar:

  $S \to aMb$

  $S \to ab$

  $M \to aMb$

  $M \to ab$

# Removing $A \to B$ in grammar

- Example Grammar:

  $S \to aA$

  $A \to a$

  $A \to B$

  $B \to A$

  $B \to bb$

  After removing $A \to B$, equivalent grammar:

  $S \to aA|aB$

  $A \to a$

  $B \to A|B$

  $B \to bb$

# Removing A $\rightarrow$ B in grammar

- Example Grammar:

  S $\rightarrow$aA

  A $\rightarrow$a

  A $\rightarrow$B

  B $\rightarrow$A

  B $\rightarrow$bb

  After removing B$\rightarrow$ B, equivalent grammar:

  S $\rightarrow$aA|aB

  A $\rightarrow$ a

  B $\rightarrow$A

  B$\rightarrow$ bb

# Removing $A \to B$ in grammar

- Example Grammar:

  $S \to aA$

  $A \to a$

  $A \to B$

  $B \to A$

  $B \to bb$

  After removing $B \to A$, equivalent grammar:

  $S \to aA|aB$

  $A \to a$

  $B \to bb$

# Greibach Normal Form

**Definition 6.5:** A context-free grammar is said to be in Greibach Normal Form if all productions have the form

$$A \rightarrow ax$$

where $a \in T$ and $x \in V^*$

# Exercise

- Find CNF for this grammar:

  $S \rightarrow 0A0 \mid 1B1 \mid BB$

  $B \rightarrow 1$

  $A \rightarrow S \mid \lambda$

# Greibach Normal Form

Example:

Convert the following grammar into GNF:

$$S \rightarrow abSb \mid aa$$

Introduce new variables A and B to stand for a and b respectively, and substitute:

$$S \rightarrow aBSB \mid aA$$

$$A \rightarrow a$$

$$B \rightarrow b$$