Computability and Complexity
COSC 4200

Context-Free Languages in Polynomial-Time

Definition

A context-free grammar is in *Chomsky normal form* if every rule is of the form

$$A \rightarrow BC$$

or

$$A \rightarrow a$$

where a is any terminal and A, B, and C are any variable – except B and C may not be the start variable. In addition, the rule $S \rightarrow \epsilon$ is permitted, where S is the start variable.

Theorem

Any context-free language is generated by a context-free grammar in Chomsky normal form.

To prove this, we show how to convert any CFG into Chomsky normal form:

Theorem

Any context-free language is generated by a context-free grammar in Chomsky normal form.

To prove this, we show how to convert any CFG into Chomsky normal form:

- We add a new start variable.

Theorem

Any context-free language is generated by a context-free grammar in Chomsky normal form.

To prove this, we show how to convert any CFG into Chomsky normal form:

- We add a new start variable.
- We eliminate all ϵ-rules of the form $A \to \epsilon$.

Theorem

Any context-free language is generated by a context-free grammar in Chomsky normal form.

To prove this, we show how to convert any CFG into Chomsky normal form:

- We add a new start variable.
- We eliminate all ϵ-rules of the form $A \to \epsilon$.
- We also eliminate all unit rules of the form $A \to B$.

Theorem

Any context-free language is generated by a context-free grammar in Chomsky normal form.

To prove this, we show how to convert any CFG into Chomsky normal form:

- We add a new start variable.
- We eliminate all ϵ-rules of the form $A \rightarrow \epsilon$.
- We also eliminate all unit rules of the form $A \rightarrow B$.
- We convert all remaining rules to the proper form by breaking into multiple rules.

Proof. First, add a new start variable S_0 and the rule $S_0 \rightarrow S$, where S was the original start variable. This ensures the start variable is never on the right-hand side of a rule.

Proof. First, add a new start variable S_0 and the rule $S_0 \to S$, where S was the original start variable. This ensures the start variable is never on the right-hand side of a rule.

Second, we take care of ϵ-rules. Let $A \to \epsilon$ be an ϵ-rule in the grammer.

Proof. First, add a new start variable S_0 and the rule $S_0 \to S$, where S was the original start variable. This ensures the start variable is never on the right-hand side of a rule.

Second, we take care of ϵ-rules. Let $A \to \epsilon$ be an ϵ-rule in the grammer.

- Remove the rule $A \to \epsilon$.

Proof. First, add a new start variable S_0 and the rule $S_0 \to S$, where S was the original start variable. This ensures the start variable is never on the right-hand side of a rule.

Second, we take care of ϵ-rules. Let $A \to \epsilon$ be an ϵ-rule in the grammer.

- Remove the rule $A \to \epsilon$.
- For any rule of the form $R \to uAv$, we add the rule $R \to uv$ with A removed.

Proof. First, add a new start variable S_0 and the rule $S_0 \to S$, where S was the original start variable. This ensures the start variable is never on the right-hand side of a rule.

Second, we take care of ϵ-rules. Let $A \to \epsilon$ be an ϵ-rule in the grammer.

- Remove the rule $A \to \epsilon$.
- For any rule of the form $R \to uAv$, we add the rule $R \to uv$ with A removed.
- If there are multiple occurrences of A in a rule, for example $R \to uAvAw$, then add the rule $R \to uvw$ with all occurrences of A removed.

Repeat this until all ϵ-rules are removed.

Proof. First, add a new start variable S_0 and the rule $S_0 \to S$, where S was the original start variable. This ensures the start variable is never on the right-hand side of a rule.

Second, we take care of ϵ-rules. Let $A \to \epsilon$ be an ϵ-rule in the grammer.

- Remove the rule $A \to \epsilon$.
- For any rule of the form $R \to uAv$, we add the rule $R \to uv$ with A removed.
- If there are multiple occurrences of A in a rule, for example $R \to uAvAw$, then add the rule $R \to uvw$ with all occurrences of A removed.

Repeat this until all ϵ-rules are removed.

Third, we take care of all unit rules. Let $A \to B$ be a unit rule.

Proof. First, add a new start variable S_0 and the rule $S_0 \to S$, where S was the original start variable. This ensures the start variable is never on the right-hand side of a rule.

Second, we take care of ϵ-rules. Let $A \to \epsilon$ be an ϵ-rule in the grammer.

- Remove the rule $A \to \epsilon$.
- For any rule of the form $R \to uAv$, we add the rule $R \to uv$ with A removed.
- If there are multiple occurrences of A in a rule, for example $R \to uAvAw$, then add the rule $R \to uvw$ with all occurrences of A removed.

Repeat this until all ϵ-rules are removed.

Third, we take care of all unit rules. Let $A \to B$ be a unit rule.

- Remove the rule $A \to B$.
- For any rule $B \to u$, we add the rule $A \to u$.

Repeat this until all unit rules are removed.

Finally, we convert all remaining rules into proper form. We replace each rule

$$A \to u_1 u_2 \cdots u_k,$$

where $k \geq 3$ and each u_i is a variable or a terminal

Finally, we convert all remaining rules into proper form. We replace each rule

$$A \to u_1 u_2 \cdots u_k,$$

where $k \geq 3$ and each u_i is a variable or a terminal, with the rules

$$\begin{array}{rcl} A & \to & u_1 A_1 \\ A_1 & \to & u_2 A_2 \\ A_2 & \to & u_3 A_3 \\ & \vdots & \\ A_{k-2} & \to & u_{k-1} u_k. \end{array}$$

Here $A_1, A_2, \ldots, A_{k-2}$ are new variables.

Finally, we convert all remaining rules into proper form. We replace each rule

$$A \to u_1 u_2 \cdots u_k,$$

where $k \geq 3$ and each u_i is a variable or a terminal, with the rules

$$\begin{array}{rcl} A & \to & u_1 A_1 \\ A_1 & \to & u_2 A_2 \\ A_2 & \to & u_3 A_3 \\ & \vdots & \\ A_{k-2} & \to & u_{k-1} u_k. \end{array}$$

Here $A_1, A_2, \ldots, A_{k-2}$ are new variables. We replace any terminal u_i in these rules with a new variable U_i and add the rule

$$U_i \to u_i.$$

This completes the construction. $\square$

Example - Chomsky Normal Form

Example. $B = \{a^n b^m c^l \mid n = m \text{ or } m = l\}$

$$\begin{array}{rcl} S & \rightarrow & LC \mid AR \\ L & \rightarrow & aLb \mid \epsilon \\ R & \rightarrow & bRc \mid \epsilon \\ A & \rightarrow & Aa \mid \epsilon \\ C & \rightarrow & Cc \mid \epsilon \end{array}$$

Convert this grammar to Chomsky normal form.

Example - Chomsky Normal Form

Example. $B = \{a^n b^m c^l \mid n = m \text{ or } m = l\}$

$$\begin{array}{rcl} S & \rightarrow & LC \mid AR \\ L & \rightarrow & aLb \mid \epsilon \\ R & \rightarrow & bRc \mid \epsilon \\ A & \rightarrow & Aa \mid \epsilon \\ C & \rightarrow & Cc \mid \epsilon \end{array}$$

Convert this grammar to Chomsky normal form.

Add new start variable:

$$S_0 \rightarrow S$$

Example - Chomsky Normal Form

Example. $B = \{a^n b^m c^l \mid n = m \text{ or } m = l\}$

$$
\begin{array}{rcl}
S & \rightarrow & LC \mid AR \\
L & \rightarrow & aLb \mid \epsilon \\
R & \rightarrow & bRc \mid \epsilon \\
A & \rightarrow & Aa \mid \epsilon \\
C & \rightarrow & Cc \mid \epsilon
\end{array}
$$

Convert this grammar to Chomsky normal form.

Add new start variable:

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid AR \\
L & \rightarrow & aLb \mid \epsilon \\
R & \rightarrow & bRc \mid \epsilon \\
A & \rightarrow & Aa \mid \epsilon \\
C & \rightarrow & Cc \mid \epsilon
\end{array}
$$

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid AR \\
L & \rightarrow & aLb \mid \epsilon \\
R & \rightarrow & bRc \mid \epsilon \\
A & \rightarrow & Aa \mid \epsilon \\
C & \rightarrow & Cc \mid \epsilon
\end{array}
$$

Remove ϵ-rules (first pass):

Example - Chomsky Normal Form

$$\begin{array}{rcl} S_0 & \rightarrow & S \\ S & \rightarrow & LC \mid AR \\ L & \rightarrow & aLb \mid \epsilon \\ R & \rightarrow & bRc \mid \epsilon \\ A & \rightarrow & Aa \mid \epsilon \\ C & \rightarrow & Cc \mid \epsilon \end{array}$$

Remove ϵ-rules (first pass):

$$\begin{array}{rcl} S_0 & \rightarrow & S \\ S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \end{array}$$

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid AR \\
L & \rightarrow & aLb \mid \epsilon \\
R & \rightarrow & bRc \mid \epsilon \\
A & \rightarrow & Aa \mid \epsilon \\
C & \rightarrow & Cc \mid \epsilon
\end{array}
$$

Remove ϵ-rules (first pass):

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab
\end{array}
$$

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid AR \\
L & \rightarrow & aLb \mid \epsilon \\
R & \rightarrow & bRc \mid \epsilon \\
A & \rightarrow & Aa \mid \epsilon \\
C & \rightarrow & Cc \mid \epsilon
\end{array}
$$

Remove ϵ-rules (first pass):

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc
\end{array}
$$

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid AR \\
L & \rightarrow & aLb \mid \epsilon \\
R & \rightarrow & bRc \mid \epsilon \\
A & \rightarrow & Aa \mid \epsilon \\
C & \rightarrow & Cc \mid \epsilon
\end{array}
$$

Remove ϵ-rules (first pass):

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a
\end{array}
$$

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid AR \\
L & \rightarrow & aLb \mid \epsilon \\
R & \rightarrow & bRc \mid \epsilon \\
A & \rightarrow & Aa \mid \epsilon \\
C & \rightarrow & Cc \mid \epsilon
\end{array}
$$

Remove ϵ-rules (first pass):

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Remove ϵ-rules (second pass):

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Remove ϵ-rules (second pass):

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \mid \epsilon \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R
\end{array}
$$

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Remove ϵ-rules (second pass):

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \mid \epsilon \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \mid \epsilon \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Remove unit rules ($S \rightarrow S_0$):

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & S \mid \epsilon \\
S & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Remove unit rules ($S \rightarrow S_0$):

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

$$
\begin{aligned}
S_0 &\rightarrow LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L &\rightarrow aLb \mid ab \\
R &\rightarrow bRc \mid bc \\
A &\rightarrow Aa \mid a \\
C &\rightarrow Cc \mid c
\end{aligned}
$$

Remove unit rules ($S_0 \rightarrow L \mid C \mid A \mid R$):

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Remove unit rules ($S_0 \rightarrow L \mid C \mid A \mid R$):

$$
S_0 \quad \rightarrow \quad LC \mid aLb \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bRc \mid bc \mid \epsilon
$$

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid L \mid C \mid AR \mid A \mid R \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Remove unit rules ($S_0 \rightarrow L \mid C \mid A \mid R$):

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid aLb \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bRc \mid bc \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid aLb \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bRc \mid bc \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Convert remaining rules ($S_0 \rightarrow aLb \mid bRC$; $L \rightarrow aLb$; $R \rightarrow bRc$):

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid aLb \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bRc \mid bc \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Convert remaining rules ($S_0 \rightarrow aLb \mid bRC; L \rightarrow aLb; R \rightarrow bRc$):

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bc \mid \epsilon \\
S_0 & \rightarrow & aA_1 \\
A_1 & \rightarrow & Lb
\end{array}
$$

Example - Chomsky Normal Form

$$\begin{array}{rcl} S_0 & \rightarrow & LC \mid aLb \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bRc \mid bc \mid \epsilon \\ L & \rightarrow & aLb \mid ab \\ R & \rightarrow & bRc \mid bc \\ A & \rightarrow & Aa \mid a \\ C & \rightarrow & Cc \mid c \end{array}$$

Convert remaining rules ($S_0 \rightarrow aLb \mid bRC; L \rightarrow aLb; R \rightarrow bRc$):

$$\begin{array}{rcl} S_0 & \rightarrow & LC \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bc \mid \epsilon \\ S_0 & \rightarrow & aA_1 \\ A_1 & \rightarrow & Lb \\ S_0 & \rightarrow & bA_2 \\ A_2 & \rightarrow & Rc \end{array}$$

Example - Chomsky Normal Form

$$
\begin{aligned}
S_0 &\rightarrow LC \mid aLb \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bRc \mid bc \mid \epsilon \\
L &\rightarrow aLb \mid ab \\
R &\rightarrow bRc \mid bc \\
A &\rightarrow Aa \mid a \\
C &\rightarrow Cc \mid c
\end{aligned}
$$

Convert remaining rules ($S_0 \rightarrow aLb \mid bRC; L \rightarrow aLb; R \rightarrow bRc$):

$$
\begin{aligned}
S_0 &\rightarrow LC \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bc \mid \epsilon \\
S_0 &\rightarrow aA_1 \\
A_1 &\rightarrow Lb \\
S_0 &\rightarrow bA_2 \\
A_2 &\rightarrow Rc \\
L &\rightarrow aA_1 \mid ab
\end{aligned}
$$

Example - Chomsky Normal Form

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid aLb \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bRc \mid bc \mid \epsilon \\
L & \rightarrow & aLb \mid ab \\
R & \rightarrow & bRc \mid bc \\
A & \rightarrow & Aa \mid a \\
C & \rightarrow & Cc \mid c
\end{array}
$$

Convert remaining rules ($S_0 \rightarrow aLb \mid bRC; L \rightarrow aLb; R \rightarrow bRc$):

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bc \mid \epsilon \\
S_0 & \rightarrow & aA_1 \\
A_1 & \rightarrow & Lb \\
S_0 & \rightarrow & bA_2 \\
A_2 & \rightarrow & Rc \\
L & \rightarrow & aA_1 \mid ab \\
R & \rightarrow & bA_2 \mid bc
\end{array}
$$

Example - Chomsky Normal Form

$$
\begin{aligned}
S_0 &\rightarrow LC \mid aLb \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bRc \mid bc \mid \epsilon \\
L &\rightarrow aLb \mid ab \\
R &\rightarrow bRc \mid bc \\
A &\rightarrow Aa \mid a \\
C &\rightarrow Cc \mid c
\end{aligned}
$$

Convert remaining rules ($S_0 \rightarrow aLb \mid bRC; L \rightarrow aLb; R \rightarrow bRc$):

$$
\begin{aligned}
S_0 &\rightarrow LC \mid ab \mid Cc \mid c \mid AR \mid Aa \mid a \mid bc \mid \epsilon \\
S_0 &\rightarrow aA_1 \\
A_1 &\rightarrow Lb \\
S_0 &\rightarrow bA_2 \\
A_2 &\rightarrow Rc \\
L &\rightarrow aA_1 \mid ab \\
R &\rightarrow bA_2 \mid bc \\
A &\rightarrow Aa \mid a \\
C &\rightarrow Cc \mid c
\end{aligned}
$$

Example - Chomsky Normal Form

Replace terminals with rules:

$$
\begin{array}{rcl}
S_0 & \rightarrow & LC \mid U_1U_2 \mid CU_3 \mid c \mid AR \mid AU_1 \mid a \mid U_2U_3 \mid \epsilon \\
S_0 & \rightarrow & U_1A_1 \\
A_1 & \rightarrow & LU_2 \\
S_0 & \rightarrow & U_2A_2 \\
A_2 & \rightarrow & RU_3 \\
L & \rightarrow & U_1A_1 \mid U_1U_2 \\
R & \rightarrow & U_2A_2 \mid U_2U_3 \\
A & \rightarrow & AU_1 \mid a \\
C & \rightarrow & CU_3 \mid c \\
U_1 & \rightarrow & a \\
U_2 & \rightarrow & b \\
U_3 & \rightarrow & c
\end{array}
$$

This is now in Chomsky normal form.

Context-Free Languages are Decidable in Polynomial Time

We will use *dynamic programming* to show that every CFL is decidable in polynomial time.

Let P be the class of problems decidable in polynomial time.

Theorem

$\mathrm{CFL} \subseteq \mathrm{P}$.

This is called the Cocke-Kasami-Younger (CKY) Algorithm.

Dynamic Programming

Recursion is top-down: we start with the full problem, divide it into subproblems, and recurse until we reach base cases.

Dynamic programming is bottom-up: we start with the base cases and build up from there to solve larger and larger subproblems, until we have solved the full problem.

Standard examples of dynamic programming problems are longest common subsequence and edit distance.

Subproblems

Let G be a CFG in Chomsky normal form. Let $w = w_1 \cdots w_n$ be an input string. We wish to determine whether G generates w.

Subproblems

Let G be a CFG in Chomsky normal form. Let $w = w_1 \cdots w_n$ be an input string. We wish to determine whether G generates w.

The idea is to determine for each $i \leq j$, whether the substring

$$w[i,j] = w_i w_{i+1} \cdots w_j$$

is generated by G.

Dynamic Programming Table

We will have a table $table(\cdot, \cdot)$ where $table(i, j)$ includes all variables from which $w[i, j]$ can be derived. We start with the smallest subproblems and work our way up.

Dynamic Programming Table

We will have a table $table(\cdot, \cdot)$ where $table(i, j)$ includes all variables from which $w[i, j]$ can be derived. We start with the smallest subproblems and work our way up.

- The base cases are when $i = j$. For any rule $A \to w_i$, we put A in $table(i, j)$.

Dynamic Programming Table

We will have a table $table(\cdot, \cdot)$ where $table(i, j)$ includes all variables from which $w[i, j]$ can be derived. We start with the smallest subproblems and work our way up.

- The base cases are when $i = j$. For any rule $A \to w_i$, we put A in $table(i, j)$.
- Now let $i < j$. Note $A \stackrel{*}{\Rightarrow} w[i, j]$ if and only if for some rule $A \to BC$ there is a splitting position $k, i \leq k < j$, such that

  $$B \in table(i, k) \text{ and } C \in table(k + 1, j).$$

  In this case we put A in $table(i, j)$.

Dynamic Programming Table

We will have a table $table(\cdot, \cdot)$ where $table(i, j)$ includes all variables from which $w[i, j]$ can be derived. We start with the smallest subproblems and work our way up.

- The base cases are when $i = j$. For any rule $A \to w_i$, we put A in $table(i, j)$.
- Now let $i < j$. Note $A \stackrel{*}{\Rightarrow} w[i, j]$ if and only if for some rule $A \to BC$ there is a splitting position $k, i \le k < j$, such that

  $$B \in table(i, k) \text{ and } C \in table(k + 1, j).$$

  In this case we put A in $table(i, j)$.

At the end, we just need to check whether the start variable S is in $table(1, n)$.

Cocke-Kasami-Younger (CKY) Algorithm

On input $w = w_1 \cdots w_n$:

  if ($w = \epsilon$ and $S \to \epsilon$) is a rule, accept

  else, reject

Cocke-Kasami-Younger (CKY) Algorithm

On input $w = w_1 \cdots w_n$:

    if ($w = \epsilon$ and $S \to \epsilon$) is a rule, accept
    else, reject

    for $i = 1$ to n
        for each variable A
            if $A \to w_i$ is a rule
                put A in $table(i, i)$

Cocke-Kasami-Younger (CKY) Algorithm

On input $w = w_1 \cdots w_n$:
    if ($w = \epsilon$ and $S \rightarrow \epsilon$) is a rule, accept
    else, reject

    for $i = 1$ to n
        for each variable A
            if $A \rightarrow w_i$ is a rule
                put A in $table(i, i)$

    for $l = 2$ to n // l is the length of the substring

Cocke-Kasami-Younger (CKY) Algorithm

On input $w = w_1 \cdots w_n$:
    if ($w = \epsilon$ and $S \to \epsilon$) is a rule, accept
    else, reject

    for $i = 1$ to n
        for each variable A
            if $A \to w_i$ is a rule
                put A in $table(i, i)$

    for $l = 2$ to n // l is the length of the substring
        for $i = 1$ to $n - l + 1$ // i is the start position of the substring

Cocke-Kasami-Younger (CKY) Algorithm

```
On input w = w_1 ··· w_n:
    if (w = ϵ and S → ϵ) is a rule, accept
    else, reject

    for i = 1 to n
        for each variable A
            if A → w_i is a rule
                put A in table(i, i)

    for l = 2 to n                          // l is the length of the substring
        for i = 1 to n − l + 1              // i is the start position of the substring
            let j = i + l − 1               // j is the end position of the substring
```

Cocke-Kasami-Younger (CKY) Algorithm

```
On input w = w_1 ··· w_n:
    if (w = ϵ and S → ϵ) is a rule, accept
    else, reject

    for i = 1 to n
        for each variable A
            if A → w_i is a rule
                put A in table(i, i)

    for l = 2 to n                          // l is the length of the substring
        for i = 1 to n − l + 1              // i is the start position of the substring
            let j = i + l − 1               // j is the end position of the substring
            for k = i to j − 1              // k is the split position
```

Cocke-Kasami-Younger (CKY) Algorithm

```
On input w = w_1 ··· w_n:
    if (w = ε and S → ε) is a rule, accept
    else, reject

    for i = 1 to n
        for each variable A
            if A → w_i is a rule
                put A in table(i, i)

    for l = 2 to n                              // l is the length of the substring
        for i = 1 to n − l + 1                  // i is the start position of the substring
            let j = i + l − 1                   // j is the end position of the substring
            for k = i to j − 1                  // k is the split position
                for each rule A → BC
```

Cocke-Kasami-Younger (CKY) Algorithm

On input $w = w_1 \cdots w_n$:
    if ($w = \epsilon$ and $S \to \epsilon$) is a rule, accept
    else, reject

    for $i = 1$ to n
        for each variable A
            if $A \to w_i$ is a rule
                put A in $table(i, i)$

    for $l = 2$ to n // l is the length of the substring
        for $i = 1$ to $n - l + 1$ // i is the start position of the substring
            let $j = i + l - 1$ // j is the end position of the substring
            for $k = i$ to $j - 1$ // k is the split position
                for each rule $A \to BC$
                    if [$B \in table(i, k)$ and
                        and $C \in table(k + 1, j)$],

Cocke-Kasami-Younger (CKY) Algorithm

```
On input w = w_1 ··· w_n:
    if (w = ε and S → ε) is a rule, accept
    else, reject

    for i = 1 to n
        for each variable A
            if A → w_i is a rule
                put A in table(i, i)

    for l = 2 to n                                  // l is the length of the substring
        for i = 1 to n − l + 1                      // i is the start position of the substring
            let j = i + l − 1                       // j is the end position of the substring
            for k = i to j − 1                      // k is the split position
                for each rule A → BC
                    if [ B ∈ table(i, k) and
                          and C ∈ table(k + 1, j) ],
                    then put A in table(i, j)
```

Cocke-Kasami-Younger (CKY) Algorithm

```
On input w = w_1 ··· w_n:
    if (w = ε and S → ε) is a rule, accept
    else, reject

    for i = 1 to n
        for each variable A
            if A → w_i is a rule
                put A in table(i, i)

    for l = 2 to n                              // l is the length of the substring
        for i = 1 to n − l + 1                  // i is the start position of the substring
            let j = i + l − 1                   // j is the end position of the substring
            for k = i to j − 1                  // k is the split position
                for each rule A → BC
                    if [ B ∈ table(i, k) and
                        and C ∈ table(k + 1, j) ],
                    then put A in table(i, j)

    if S ∈ table(1, n), accept
    else, reject
```

CKY Algorithm Runtime Analysis

Let

$$\begin{aligned} n &= \text{the length of the input,} \\ v &= \text{the number of variables,} \\ r &= \text{the number of rules.} \end{aligned}$$

Note that for a fixed context-free language, v and r are fixed constants.

CKY Algorithm Runtime Analysis

Let

$$\begin{aligned} n &= \text{the length of the input,} \\ v &= \text{the number of variables,} \\ r &= \text{the number of rules.} \end{aligned}$$

Note that for a fixed context-free language, v and r are fixed constants.

- The first for loop takes $O(nv) = O(n)$ time.
- The nested for loops take $O(n^3 r) = O(n^3)$ time.

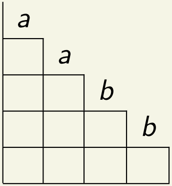
The total run time is $O(n^3)$.

Therefore this is a polynomial-time algorithm and $\mathrm{CFL} \subseteq \mathrm{P}$. $\square$

Chomsky normal form grammar for $\{a^n b^n \mid n \geq 0\}$:

$$
\begin{aligned}
S_0 &\rightarrow AT \mid AB \mid \epsilon \\
S &\rightarrow AT \mid AB \\
T &\rightarrow SB \\
A &\rightarrow a \\
B &\rightarrow b
\end{aligned}
$$

Input: *aabb*



Chomsky normal form grammar for $\{a^n b^n \mid n \geq 0\}$:

$$
\begin{array}{rcl}
S_0 & \rightarrow & AT \mid AB \mid \epsilon \\
S & \rightarrow & AT \mid AB \\
T & \rightarrow & SB \\
A & \rightarrow & a \\
B & \rightarrow & b
\end{array}
$$

Input: *aabb*

a			
A	a		
	A	b	
		B	b
			B

Chomsky normal form grammar for $\{a^n b^n \mid n \geq 0\}$:

$$\begin{array}{rcl} S_0 & \rightarrow & AT \mid AB \mid \epsilon \\ S & \rightarrow & AT \mid AB \\ T & \rightarrow & SB \\ A & \rightarrow & a \\ B & \rightarrow & b \end{array}$$

Input: *aabb*

a			
A	a		
	A	b	
	S_0, S	B	b
			B

Chomsky normal form grammar for $\{a^n b^n \mid n \geq 0\}$:

$$
\begin{array}{rcl}
S_0 & \rightarrow & AT \mid AB \mid \epsilon \\
S & \rightarrow & AT \mid AB \\
T & \rightarrow & SB \\
A & \rightarrow & a \\
B & \rightarrow & b
\end{array}
$$

Input: *aabb*

a			
A	a		
	A	b	
	S_0, S	B	b
	T		B

Chomsky normal form grammar for $\{a^n b^n \mid n \geq 0\}$:

$$
\begin{aligned}
S_0 &\rightarrow AT \mid AB \mid \epsilon \\
S &\rightarrow AT \mid AB \\
T &\rightarrow SB \\
A &\rightarrow a \\
B &\rightarrow b
\end{aligned}
$$

Input: *aabb*

a			
A	a		
	A	b	
	S_0, S	B	b
S_0, S	T		B

Chomsky normal form grammar for $\{a^n b^n \mid n \geq 0\}$:

$$
\begin{array}{rcl}
S_0 & \rightarrow & AT \mid AB \mid \epsilon \\
S & \rightarrow & AT \mid AB \\
T & \rightarrow & SB \\
A & \rightarrow & a \\
B & \rightarrow & b
\end{array}
$$

Input: *aabb*

a			
A	a		
	A	b	
	S_0, S	B	b
S_0, S	T		B

aabb is accepted.

Summary

- Every CFG may be converted into Chomsky normal form.
- Membership in a Chomsky normal form grammar may be decided in $O(n^3)$ time by the CKY algorithm.
- $\mathrm{REG} \subseteq \mathrm{CFL} \subseteq \mathrm{P}$.