$\mathbf{CFL} \subseteq \mathbf{P}$

Lecture 10

Binghui Peng

Today's lecture

Recognize words in context-free languages.

Cocke-Younger-Kasami Algorithm

Sakai (1962), Kasami (1965), Younger (1967), Cocke/Schwartz (1970)

Problem

Let L be a context-free language and let G be a Chomsky Normal Form grammar for L.

$w = \sigma_1 \cdots \sigma_n$.
We want to decide whether $w \in L$.
The goal is an algorithm that halts in polynomial time $p(n)$.
We will obtain $p(n) = O(n^3)$.

Omitting a Case

Assume $\epsilon \notin L$, so the rule

$$S \to \epsilon$$

does not appear in G.
By the definition of Chomsky Normal Form, no other ϵ-rules are allowed.

The proof can be modified to include the case $\epsilon \in L$.

Plan for CFL $\subseteq$ P

$w = \sigma_1 \cdots \sigma_n$.
Define

$$\mathrm{GEN}(\sigma_1 \cdots \sigma_n) = \{A : A \Rightarrow \sigma_1 \cdots \sigma_n\}.$$

Then deciding $w \in L$ is equivalent to checking whether

$$S \in \mathrm{GEN}(\sigma_1 \cdots \sigma_n).$$

For notation, define

$$\mathrm{GEN}[1, n] = \{A : A \Rightarrow \sigma_1 \cdots \sigma_n\}.$$

Hence the decision question is: Is $S \in \mathrm{GEN}[1, n]$?

Easier to Solve a Harder Problem

The final goal is to determine whether $S \in \mathrm{GEN}[1, n]$.

Instead, compute the full set $\mathrm{GEN}[1, n]$.

Easier to Solve an Even Harder Problem

For $i \leq j$, define

$$\mathrm{GEN}[i, j] = \{A : A \Rightarrow \sigma_i \cdots \sigma_j\}.$$

Compute **all** sets $\mathrm{GEN}[i, j]$.
Then $\mathrm{GEN}[1, n]$ is available, and we can decide whether $S \in \mathrm{GEN}[1, n]$.

Why compute all intervals?
Dynamic programming uses previously computed $\mathrm{GEN}[i, j]$ values to obtain larger intervals.

Bottom-Up View

$$\sigma_1 \cdots \sigma_{i-1} \overbrace{\sigma_i}^{A} \sigma_{i+1} \cdots \sigma_n$$

$$\mathrm{GEN}[i, i] = \{A : A \to \sigma_i\}$$

$$\sigma_1 \cdots \sigma_{i-1} \overbrace{\sigma_i}^{B} \overbrace{\sigma_{i+1}}^{C} \sigma_{i+2} \cdots \sigma_n$$

$$\begin{aligned}
&\mathrm{GEN}[i, i+1] \\
&= \{A : A \to BC \ \wedge \ B \to \sigma_i \ \wedge \ C \to \sigma_{i+1}\} \\
&= \{A : A \to BC \ \wedge \ B \in \mathrm{GEN}[i, i] \ \wedge \ C \in \mathrm{GEN}[i+1, i+1]\}
\end{aligned}$$

Bottom-Up View (Continued)

$$\sigma_1 \cdots \sigma_{i-1} \overbrace{\sigma_i}^{B} \overbrace{\sigma_{i+1}\sigma_{i+2}}^{C} \sigma_{i+3} \cdots \sigma_n$$

$$\sigma_1 \cdots \sigma_{i-1} \overbrace{\sigma_i\sigma_{i+1}}^{B} \overbrace{\sigma_{i+2}}^{C} \sigma_{i+3} \cdots \sigma_n$$

$$\begin{aligned}
&\mathrm{GEN}[i, i+2] \\
&= \{A : A \to BC \ \wedge \ ((B \to \sigma_i \ \wedge \ C \Rightarrow \sigma_{i+1}\sigma_{i+2}) \\
&= \{A : A \to BC \ \wedge \ ((B \in \mathrm{GEN}[i, i] \ \wedge \ C \in \mathrm{GEN}[i+1, i+2])
\end{aligned}$$

Bottom-Up View (Continued)

$$\sigma_1 \cdots \sigma_{i-1} \overbrace{\sigma_i}^{B} \overbrace{\sigma_{i+1}\sigma_{i+2}\sigma_{i+3}}^{C} \sigma_{i+4} \cdots \sigma_n$$

$$\sigma_1 \cdots \sigma_{i-1} \overbrace{\sigma_i\sigma_{i+1}}^{B} \overbrace{\sigma_{i+2}\sigma_{i+3}}^{C} \sigma_{i+4} \cdots \sigma_n$$

$$\sigma_1 \cdots \sigma_{i-1} \overbrace{\sigma_i\sigma_{i+1}\sigma_{i+2}}^{B} \overbrace{\sigma_{i+3}}^{C} \sigma_{i+4} \cdots \sigma_n$$

This yields the general computation pattern for $\mathrm{GEN}[i, i+3]$.

Recurrence

$$\mathrm{GEN}[i,j] = \{A : A \Rightarrow \sigma_i \cdots \sigma_j\}$$

$$\sigma_1 \cdots \sigma_{i-1} \overbrace{\sigma_i \sigma_{i+1} \cdots \sigma_k}^{B} \overbrace{\sigma_{k+1} \sigma_{k+2} \cdots \sigma_j}^{C} \sigma_{j+1} \cdots \sigma_n$$

$$\begin{aligned}
&\mathrm{GEN}[i,j] \\
&= \bigcup_{i \le k < j} \{A : A \to BC \ \wedge \ B \Rightarrow \sigma_i \cdots \sigma_k \ \wedge \ C \Rightarrow \sigma_{k+1} \cdots \sigma_j\} \\
&= \bigcup_{i \le k < j} \{A : A \to BC \ \wedge \ B \in \mathrm{GEN}[i,k] \ \wedge \ C \in \mathrm{GEN}[k+1,j]\}
\end{aligned}$$

The Algorithm

```
for i = 1 to n do
    for j = i to n do
        GEN[i,j] ← ∅
for i = 1 to n do
    for all rules A → σ_i do
        GEN[i,i] ← GEN[i,i] with A
for s = 2 to n do
    for i = 1 to n-s+1 do
        j ← i+s-1
        for k = i to j-1 do
            for all rules A → BC
                where B ∈ GEN[i,k] and C ∈ GEN[k+1,j] do
                    GEN[i,j] ← GEN[i,j] with A
```