CSEN 1003: Compilers

Tutorial 4 - Left Recursion Elimination and Left Factoring

23/2/2020 - 26/2/2020

Top Down Parsing

- A top down parser is a parser that generates the parse tree from the root to the leaves.

Top Down Parsing

- A top down parser is a parser that generates the parse tree from the root to the leaves.
- To build a top down parser, the CFG must be:
  1. Unambiguous.
  2. Non left recursive.
  3. Left factored.

Today's Plan

1. Left Recursion Elimination
2. Left Factoring
3. Recap

Immediate Left Recursion

- A CFG is left recursive if $\exists A \in V$ where $A \overset{+}{\Rightarrow} A\alpha$.

Immediate Left Recursion

- A CFG is left recursive if $\exists A \in V$ where $A \overset{+}{\Rightarrow} A\alpha$.
- A top down parser can get into infinite loops if the grammar is left recursive (why?).

Immediate Left Recursion

- A CFG is left recursive if $\exists A \in V$ where $A \overset{+}{\Rightarrow} A\alpha$.
- A top down parser can get into infinite loops if the grammar is left recursive (why?).
- In General:

$$A \rightarrow A\alpha_1 \mid \dots \mid A\alpha_n \mid \beta_1 \mid \dots \mid \beta_m$$

$$\downarrow$$

$$A \rightarrow \beta_1 A' \mid \dots \mid \beta_m A'$$

$$A' \rightarrow \alpha_1 A' \mid \dots \mid \alpha_n A' \mid \varepsilon$$

$\alpha_i \neq \varepsilon$ and β_i does not start with A.

Immediate Left Recursion Elimination Example

$$A \rightarrow A\alpha_1 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \ldots \mid \beta_m$$

$$\downarrow$$

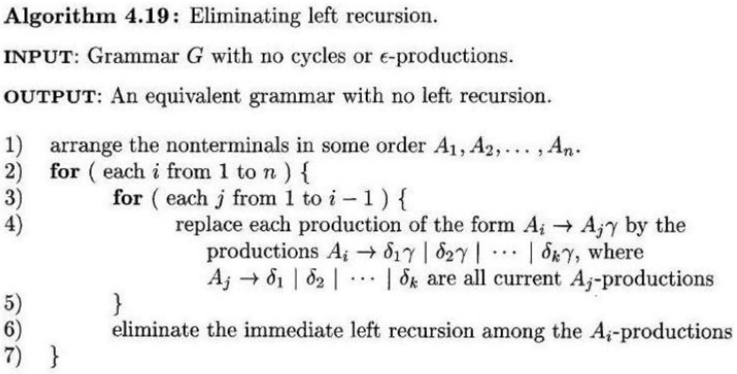
$$A \rightarrow \beta_1 A' \mid \ldots \mid \beta_m A'$$

$$A' \rightarrow \alpha_1 A' \mid \ldots \mid \alpha_n A' \mid \varepsilon$$

Example

1. $S \rightarrow S\mathtt{ab} \mid \mathtt{cd}$
2. $S \rightarrow S \cup S \mid S\ S \mid S* \mid (S) \mid \mathtt{a}$
3. $A \rightarrow 0 \mid T1$
   $T \rightarrow 1 \mid A0$

Eliminating General Left Recursion



Algorithm 4.19: Eliminating left recursion.

INPUT: Grammar G with no cycles or ϵ-productions.

OUTPUT: An equivalent grammar with no left recursion.

```
arrange the nonterminals in some order $A_1, A_2, \ldots, A_n$.
for ( each $i$ from 1 to $n$ ) {
        for ( each $j$ from 1 to $i-1$ ) {
                replace each production of the form $A_i \rightarrow A_j\gamma$ by the
                    productions $A_i \rightarrow \delta_1\gamma \mid \delta_2\gamma \mid \cdots \mid \delta_k\gamma$, where
                    $A_j \rightarrow \delta_1 \mid \delta_2 \mid \cdots \mid \delta_k$ are all current $A_j$-productions
        }
        eliminate the immediate left recursion among the $A_i$-productions
}
```

Eliminating ε-productions

- Why are ε-productions problematic?

$$\begin{aligned} S &\rightarrow XSa \mid b \\ X &\rightarrow \varepsilon \end{aligned}$$

Eliminating ε-productions

- Why are ε-productions problematic?

$$\begin{aligned} S &\rightarrow XSa \mid b \\ X &\rightarrow \varepsilon \end{aligned}$$

- Eliminating ε-productions:
  1. $\forall A \rightarrow \varepsilon$, remove it.
  2. $\forall B \rightarrow \alpha A \beta$, add $B \rightarrow \alpha\beta$. Repeat for all choices of α and β. Make sure you do not reintroduce an already eliminated ε production while susbstitution.

Example

Eliminate the ε productions from the following CFG.

$$\begin{aligned} S &\rightarrow AbB \mid C \\ B &\rightarrow AA \mid AC \\ C &\rightarrow b \mid c \\ A &\rightarrow a \mid \varepsilon \end{aligned}$$

Eliminating Cycles

- **Direct case:** $A \to A \mid Ab \mid b$
- **Indirect case:** $$\begin{array}{lcl} S & \to & X \mid Xb \mid SS \\ X & \to & S \mid a \end{array}$$
- Eliminating cycles:
  1. Make sure ε productions are removed first.
  2. For each unit rule $A \to B \in R$, remove it. Replace it by $A \to \alpha_1 \mid \dots \mid \alpha_n$ where $B \to \alpha_1 \mid \dots \mid \alpha_n$. Make sure you do not reintroduce already eliminated cycles.

Example

Eliminate the cycles from the following CFG.

$$\begin{array}{lcl} A & \to & B \mid a \\ B & \to & A \mid C \mid b \\ C & \to & d \end{array}$$

One More Left Recursion Elimination

Example

Eliminate the cycles from the following CFG.

$$
\begin{array}{lcl}
A & \rightarrow & BC \\
B & \rightarrow & Bb \mid \varepsilon \\
C & \rightarrow & AC \mid a
\end{array}
$$

Today's Plan

1. Left Recursion Elimination
2. Left Factoring
3. Recap

Left Factoring Grammars

- A grammar G is left factored if $\forall\{A \rightarrow \alpha, A \rightarrow \beta\} \subseteq R$, the longest common prefix of α, β is ε.
- $\begin{array}{lcl} S & \rightarrow & cAd \\ A & \rightarrow & ab \mid a \end{array}$ Input: cad
- In General:

$$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid ...\alpha\beta_m \mid \gamma_1|...|\gamma_n$$

$$\downarrow$$

$$A \rightarrow \alpha A' \mid \gamma_1|...|\gamma_n$$

$$A' \rightarrow \beta_1 \mid ... \mid \beta_m$$

$\alpha \neq \varepsilon$ and γ_i does not start with α.

Left Factoring Exercise

Example

Left factor the following CFG.

$$S \quad \rightarrow \quad \texttt{abx} \mid \texttt{aby} \mid \texttt{acx} \mid \texttt{acy}$$

Today's Plan

1. Left Recursion Elimination
2. Left Factoring
3. Recap

Points to Take Home

1. Left Recursion Elimination.
2. Left Factoring.

Next Week: LL(1) Parsers!