Elimination of left recursion

LEFT RECURSION. Let G be a context-free grammar. A production of G is said left recursive if it has the form

$$A \longmapsto A\alpha \tag{20}$$

where A is a nonterminal and α is a string of grammar symbols. A nonterminal A of G is said left recursive if there exists a string of grammar symbols α such that we have

$$A \stackrel{*}{\Longrightarrow} A\alpha \tag{21}$$

Such derivation is also said left recursive. The grammar G is left recursive if it has at least one left recursive nonterminal.

Remark 4 Top-down parsing is one of the methods that we will study for generating parse trees. This method cannot handle left recursive grammars. We present now an algorithm for transforming a left recursive grammar G into a grammar G' which is not left recursive and which generates the same language as G.

THE BASIC TRICK is to replace the production

$$A \longmapsto A\alpha \mid \beta \quad \text{by} \quad \left\{ \begin{array}{lll} A & \longmapsto & \beta A' \\ A' & \longmapsto & \alpha A' \mid \varepsilon \end{array} \right. \tag{22}$$

where $\alpha \neq \varepsilon$ is assumed. Observe that this trick eliminates left recursive productions. Applying this trick is not enough to remove all derivations of the form $A \stackrel{*}{\Longrightarrow} A\alpha$. However this trick is sufficient for many grammars.

Example 10 The following grammar which generates arithmetic expressions

$$\begin{array}{l} E \longmapsto E + T \mid T \\ T \longmapsto T*F \mid F \\ F \longmapsto (E) \mid \mathbf{id} \end{array} \tag{23}$$

has two left recursive productions. Applying the above trick leads to

$$\begin{aligned} E &\longmapsto TE' \\ E' &\longmapsto +TE' \mid \varepsilon \\ T &\longmapsto FT' \\ T' &\longmapsto *FT' \mid \varepsilon \\ F &\longmapsto (E) \mid \mathbf{id} \end{aligned} \tag{24}$$

<u>THE CASE OF SEVERAL LEFT RECURSIVE A-PRODUCTIONS.</u> Assume that the set of all A-productions has the form

$$A \longmapsto A\alpha_1 \mid A\alpha_2 \mid \cdots A\alpha_m \mid \beta_1 \mid \beta_2 \mid \cdots \beta_n \tag{25}$$

where no β_i begins with an A and where $\alpha_i \neq \varepsilon$ holds for every i = 1 ⋯ m. Then we repalce these A-productions by

$$\begin{aligned} A &\longmapsto \beta_1 A' \mid \beta_2 A' \mid \cdots \beta_n A' \\ A' &\longmapsto \alpha_1 A' \mid \alpha_2 A' \mid \cdots \alpha_m A' \mid \varepsilon \end{aligned} \tag{26}$$

Remark 5 Let us consider the following grammar.

$$\begin{aligned} S &\longmapsto Aa \mid b \\ A &\longmapsto Ac \mid Sd \mid \varepsilon \end{aligned} \tag{27}$$

The nonterminal S is left recursive since we have

$$S \Longrightarrow Aa \Longrightarrow Sda \tag{28}$$

However there is no left recursive S-productions. We show now how to deal with these left recursive derivations.

<u>REMOVING ALL LEFT RECURSIVE DERIVATIONS.</u> Let us assume that non-terminals are numbered: A_1, A_2,..., A_n. The goal is to remove any possible circuit for all $1 \leq j < i \leq n$

$$A_i \stackrel{*}{\Rightarrow} A_j \delta_j \stackrel{*}{\Rightarrow} A_i \delta_i \,. \qquad (29)$$

The idea is for i = 1 ⋯ n

- To make sure that every A_i-production does not have a form $A_i \longmapsto A_j \delta$ for some j < i.
- To remove any left recursive A_i-production.

The method in more detail:

- remove all left recursive A_1-productions (by the above trick)
- remove A_1 from the right-hand side of each A_2-production of the form $A_2 \longmapsto A_1 \delta$ (by applying all A_1-productions)
- remove all left recursive A_2-productions
- remove A_j from the right-hand side of each A_3-production of the form $A_3 \longmapsto A_j \delta$ for j = 1, 2.
- remove all left recursive A_3-productions
- ...
- remove A_j from the right-hand side of each A_i-production of the form $A_i \longmapsto A_j \delta$ (for every j such that $1 \leq j < i \leq n$)
- remove all left recursive A_i-productions
- ...

Observations:

- To remove A_j from the right-hand side of the A_i-production $A_i \longmapsto A_j \delta$ (where $1 \leq j < i \leq n$) replace

  $$A_i \longmapsto A_j \delta \text{ by } A_i \longmapsto \gamma_1 \delta \mid \cdots \mid \gamma_k \delta \text{ where } A_j \longmapsto \gamma_1 \mid \cdots \mid \gamma_k \qquad (30)$$

  are all A_j-productions and $\gamma_i \neq \varepsilon$ for i = 1 ⋯ k.

- This construction may not work if G constains initially a production of the form A $\longmapsto \varepsilon$. See Example 11. However we can always remove such productions. Explain how!
- Also, this construction may fail if the grammar G has a cycle, that is, if there exists a nonterminal A such that A $\stackrel{*}{\Rightarrow}$ A. This can happen with the following grammar fragment

$$
\begin{array}{l}
A_1 \longmapsto A_2 \mid \gamma_1 \\
A_2 \longmapsto A_1 \mid \gamma_2
\end{array}
\tag{31}
$$

  where

  - γ_1 is a string of grammar symbols not starting with A_1 (what we can assume by application of the basic trick given in Relation 22)
  - γ_2 is a string of grammar symbols not starting with A_1 (what we can assume by application of the A_1-productions).

  Then we can replace the above fragment by

$$
\begin{array}{l}
A_1 \longmapsto \gamma_2 \mid \gamma_1 \\
A_2 \longmapsto \gamma_1 \mid \gamma_2
\end{array}
\tag{32}
$$

  without changing the generated language. Unfortunately a cycle may be hidden in a more subtle manner. Indeed γ_2 may write γ_{21} A_2 γ_{22} and γ_1 may write γ_{11} A_1 γ_{12} with $\gamma_{21} \stackrel{*}{\Rightarrow} \varepsilon$, $\gamma_{22} \stackrel{*}{\Rightarrow} \varepsilon$, $\gamma_{11} \stackrel{*}{\Rightarrow} \varepsilon$ and $\gamma_{12} \stackrel{*}{\Rightarrow} \varepsilon$. However this may only happen if there are productions of the form A $\longmapsto \varepsilon$.
- In conclusion, the above construction for removing all left recursive derivations should be performed after

  - removing all productions of the form A $\longmapsto \varepsilon$,
  - removing cycles, that is derivations of the form A $\stackrel{*}{\Rightarrow}$ A.

  Now observe that the basic trick given in Relation 22 introduces productions of the form A $\longmapsto \varepsilon$. However one can show that the above construction cannot reintroduce cycles.

Example 11 Consider the following grammar.

$$A_3 \qquad A_2A_1 \tag{33}$$

$$\longmapsto$$

$$A_2 \longmapsto \varepsilon$$

$$A_1 \longmapsto A_2A_1$$

Applying the previous algorithm for removing all possible left-recursive derivations fail on this grammar, because of the ε -production.

Example 12   Consider again the following grammar.

$$\begin{aligned} S &\longmapsto Aa \mid b \\ A &\longmapsto Ac \mid Sd \mid \varepsilon \end{aligned} \tag{34}$$

We order the nonterminals: S < A. There is no left recursive S-productions. So the next step is to remove S from the right-hand side of the A-productions of the form A $\longmapsto$ S α. Hence we obtain

$$\begin{aligned} S &\longmapsto Aa \mid b \\ A &\longmapsto Ac \mid Aad \mid bd \mid \varepsilon \end{aligned} \tag{35}$$

Then we remove the left recursive A-productions.

$$\begin{aligned} S &\longmapsto Aa \mid b \\ A &\longmapsto bdA' \mid A' \\ A' &\longmapsto cA' \mid adA' \mid \varepsilon \end{aligned} \tag{36}$$

Marc Moreno Maza
2004-12-02