# COMP2012/G52LAC Languages and Computation Lecture 13

*Recursive-Descent Parsing: Elimination of Left Recursion*

Henrik Nilsson

University of Nottingham, UK



## This Lecture

- The problem of recursive-descent parsing and left recursive grammars.
- Elimination of left recursion.



## Left Recursion

Consider: $A \rightarrow Aa \mid \epsilon$

Parsing function:

```
parseA :: [Token] -> Maybe [Token]
parseA ts =
    case parseA ts of
        Just ('a' : ts') -> Just ts'
        _                -> Just ts
```

Any problem?

Would ***loop***! Recursive-descent parsers ***cannot*** (easily) deal with ***left-recursive*** grammars.



## Elimination of Left Recursion (1)

- A grammar is ***left-recursive*** if there is some non-terminal $A$ such that $A \stackrel{+}{\Rightarrow} A\alpha$.
- Certain parsing methods ***cannot*** handle left-recursive grammars.
- If we want to use such a parsing method for parsing a language $L = L(G)$ given by a left-recursive grammar $G$, then the grammar first has to be transformed into an ***equivalent*** grammar $G'$ that is ***not*** left-recursive.



## Recap: Equivalence of Grammars

Two grammars $G_1$ and $G_2$ are ***equivalent*** iff $L(G_1) = L(G_2)$.

Example:

$$G_1: \begin{array}{lcl} S & \rightarrow & \epsilon \mid A \\ A & \rightarrow & a \mid aA \end{array} \qquad G_2: \begin{array}{lcl} S & \rightarrow & A \\ A & \rightarrow & \epsilon \mid Aa \end{array}$$

$L(G_1) = \{a\}^* = L(G_2)$

(The equivalence of CFGs is in general ***undecidable***.)



## Elimination of Left Recursion (2)

- We will first consider ***immediate*** left recursion; i.e., productions of the form
$$A \rightarrow A\alpha$$
We will further assume that $\alpha$ cannot derive $\epsilon$.
- Key idea: $A \rightarrow \beta \mid A\alpha$ and $A \rightarrow \beta(\alpha)^*$ are equivalent.
- The latter can be expressed as:
$$\begin{array}{lcl} A & \rightarrow & \beta A' \\ A' & \rightarrow & \alpha A' \mid \epsilon \end{array}$$
where $A'$ is a new nonterminal (name arbitrary).



## Exercise

- The following grammar $G_1$ is immediately left-recursive:
$$A \rightarrow b \mid Aa$$
Draw the derivation tree for $baa$ using $G_1$.
- The following is a non-left-recursive grammar $G_1'$ equivalent to $G_1$:
$$\begin{array}{lcl} A & \rightarrow & bA' \\ A' & \rightarrow & aA' \mid \epsilon \end{array}$$
Draw the derivation tree for $baa$ using $G_1'$.



## Elimination of Left Recursion (3)

For each nonterminal $A$ defined by some left-recursive production, group the productions for $A$

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

such that no $\beta_i$ begins with an $A$.

Then replace the $A$ productions by

$$\begin{array}{lcl} A & \rightarrow & \beta_1 A' \mid \beta_2 A' \mid \ldots \mid \beta_n A' \\ A' & \rightarrow & \alpha_1 A' \mid \alpha_2 A' \mid \ldots \mid \alpha_m A' \mid \epsilon \end{array}$$

Assumption: no $\alpha_i$ derives $\epsilon$.



## Elimination of Left Recursion (4)

Consider the (immediately) left-recursive grammar:

$$\begin{array}{lcl} S & \rightarrow & A \mid B \\ A & \rightarrow & ABc \mid AAdd \mid a \mid aa \\ B & \rightarrow & Bee \mid b \end{array}$$

Terminal strings derivable from $B$ include:

$b$, $bee$, $beeee$, $beeeeee$

Terminal strings derivable from $A$ include:

$a$, $aa$, $aadd$, $aaadd$, $aaadddd$, $abc$, $aabc$, $abeec$, $aabeec$, $abeecbeec$, $aabeeeecddbeec$

## Elimination of Left Recursion (5)

Let us do a leftmost derivation of $aabeeeecddbeec$:

$$\begin{aligned} S &\Rightarrow A \\ &\Rightarrow ABc \\ &\Rightarrow AAddBc \\ &\Rightarrow aAddBc \\ &\Rightarrow aABcddBc \\ &\Rightarrow aaBcddBc \\ &\Rightarrow aaBeecddBc \\ &\Rightarrow aaBeeeecddBc \\ &\Rightarrow aabeeeecddBc \\ &\Rightarrow aabeeeecddBeec \\ &\Rightarrow aabeeeecddbeec \end{aligned}$$



## Elimination of Left Recursion (6)

Here is the grammar again:

$$\begin{array}{lcl} S & \rightarrow & A \mid B \\ A & \rightarrow & ABc \mid AAdd \mid a \mid aa \\ B & \rightarrow & Bee \mid b \end{array}$$

An equivalent right-recursive grammar:

$$\begin{array}{lcl} S & \rightarrow & A \mid B \\ A & \rightarrow & aA' \mid aaA' \\ A' & \rightarrow & BcA' \mid AddA' \mid \epsilon \end{array} \qquad \begin{array}{lcl} B & \rightarrow & bB' \\ B' & \rightarrow & eeB' \mid \epsilon \end{array}$$



## Elimination of Left Recursion (7)

Derivation of $aabeeeecddbeec$ in the new grammar:

$$\begin{aligned} S &\Rightarrow A \Rightarrow aA' \Rightarrow aAddA' \Rightarrow aaA'ddA' \\ &\Rightarrow aaBcA'ddA' \\ &\Rightarrow aabB'cA'ddA' \\ &\Rightarrow aabeeB'cA'ddA' \\ &\Rightarrow aabeeeeB'cA'ddA' \\ &\Rightarrow aabeeeecA'ddA' \\ &\Rightarrow aabeeeecddA' \\ &\Rightarrow aabeeeecddBcA' \\ &\Rightarrow aabeeeecddbB'cA' \\ &\Rightarrow aabeeeecddbeeB'cA' \\ &\Rightarrow aabeeeecddbeecA' \Rightarrow aabeeeecddbeec \end{aligned}$$



## General Left Recursion (1)

To eliminate ***general*** left recursion:

- first transform the grammar into an ***immediately*** left-recursive grammar through systematic substitution
- then proceed as before.



## Substitution

- An occurrence of a non-terminal in a right-hand side may be replaced by the right-hand sides of the productions for that non-terminal if done in all possible ways.
- All productions for non-terminals that, as a result, cannot be reached from the start symbol, can be eliminated.

(See e.g. the Typeset Lecture Notes section 8.3, or Aho, Sethi, and Ullman (1986) for details.)



## General Left Recursion (2)

For example, the generally left-recursive grammar

$$\begin{array}{lcl} A & \rightarrow & Ba \\ B & \rightarrow & Ab \mid Ac \mid \epsilon \end{array}$$

is first transformed into the immediately left-recursive grammar

$$\begin{array}{lcl} A & \rightarrow & Aba \\ A & \rightarrow & Aca \\ A & \rightarrow & a \end{array}$$



## Exercise

Transform the following generally left-recursive grammar

$$\begin{array}{lcl} A & \rightarrow & BaB \\ B & \rightarrow & Cb \mid \epsilon \\ C & \rightarrow & Ab \mid Ac \end{array}$$

into an equivalent immediately left-recursive grammar.

Then eliminate the left recursion.



## Solution (1)

First:

$$\begin{array}{lcl} A & \rightarrow & BaB \\ B & \rightarrow & Abb \mid Acb \mid \epsilon \end{array}$$

Then:

$$\begin{array}{lcl} A & \rightarrow & AbbaB \mid AcbaB \mid aB \\ B & \rightarrow & Abb \mid Acb \mid \epsilon \end{array}$$

Or, eliminating $B$ completely:

$$\begin{array}{lcl} A & \rightarrow & AbbaAbb \mid AcbaAbb \mid aAbb \\ & \mid & AbbaAcb \mid AcbaAcb \mid aAcb \\ & \mid & Abba \mid Acba \mid a \end{array}$$



## Solution (2)

Let's go with the smaller version (fewer productions):

$$\begin{array}{lcl} A & \rightarrow & AbbaB \mid AcbaB \mid aB \\ B & \rightarrow & Abb \mid Acb \mid \epsilon \end{array}$$

Only productions for $A$ are immediately left-recursive. Applying the elimination transformation:

$$\begin{array}{lcl} A & \rightarrow & aBA' \\ A' & \rightarrow & bbaBA' \mid cbaBA' \mid \epsilon \\ B & \rightarrow & Abb \mid Acb \mid \epsilon \end{array}$$

Note: $A$ appears to the left in $B$-productions; yet grammar no longer left-recursive. Why?