

G52MAL 2012/13: Lecture 14

Recursive-Descent Parsing: Elimination of Left Recursion

Henrik Nilsson

University of Nottingham, UK

G52MAL 2012/13: Lecture 14 – p.1/17

Elimination of Left Recursion (1)

- A grammar is **left-recursive** if there is some non-terminal A such that $A \xrightarrow{+} A\alpha$.
- 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.

G52MAL 2012/13: Lecture 14 – p.4/17

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 :

$$\begin{aligned} A &\rightarrow bA' \\ A' &\rightarrow aA' \mid \epsilon \end{aligned}$$

Draw the derivation tree for baa using G'_1 .

G52MAL 2012/13: Lecture 14 – p.7/17

This Lecture

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

G52MAL 2012/13: Lecture 14 – p.2/17

Recap: Equivalence of Grammars

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

Example:

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

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

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

G52MAL 2012/13: Lecture 14 – p.5/17

Elimination of Left Recursion (3)

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

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

such that no β_i begins with an A .

Then replace the A productions by

$$\begin{aligned} A &\rightarrow \beta_1A' \mid \beta_2A' \mid \dots \mid \beta_nA' \\ A' &\rightarrow \alpha_1A' \mid \alpha_2A' \mid \dots \mid \alpha_mA' \mid \epsilon \end{aligned}$$

Assumption: no α_i derives ϵ .

G52MAL 2012/13: Lecture 14 – p.8/17

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** deal with **left-recursive** grammars.

G52MAL 2012/13: Lecture 14 – p.9/17

Elimination of Left Recursion (2)

- We will first consider **immediate** left recursion; i.e., productions of the form

$$A \rightarrow A\alpha$$

where α cannot derive ϵ .

- Key idea: $A \rightarrow \beta \mid A\alpha$ and $A \rightarrow \beta(\alpha)^*$ are equivalent.
- The latter can be expressed as:

$$\begin{aligned} A &\rightarrow \beta A' \\ A' &\rightarrow \alpha A' \mid \epsilon \end{aligned}$$

G52MAL 2012/13: Lecture 14 – p.6/17

Elimination of Left Recursion (4)

Consider the (immediately) left-recursive grammar:

$$\begin{aligned} S &\rightarrow A \mid B \\ A &\rightarrow ABc \mid AAdd \mid a \mid aa \\ B &\rightarrow Bcc \mid b \end{aligned}$$

Terminal strings derivable from B include:

$$b, bcc, bcccc, bcccccc$$

Terminal strings derivable from A include:

$$a, aa, aadd, aaadd, aaadddd, abc, aabc, abccc, aabccc, abcccbecc, aabcccbeccbecc$$

G52MAL 2012/13: Lecture 14 – p.11/17

Elimination of Left Recursion (5)

Let us do a leftmost derivation of $aabeeeecdbeec$:

$$\begin{aligned}
 S &\Rightarrow A \\
 &\Rightarrow ABc \\
 &\Rightarrow AAddBc \\
 &\Rightarrow aAddBc \\
 &\Rightarrow aABcddBc \\
 &\Rightarrow aaBcddBc \\
 &\Rightarrow aaBeeecddBc \\
 &\Rightarrow aabeeeecdBc \\
 &\Rightarrow aabeeeecdBeeec \\
 &\Rightarrow aabeeeecdbeec
 \end{aligned}$$

OS2MAL 2012/13, Lecture 14 – p.15/17

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.

OS2MAL 2012/13, Lecture 14 – p.15/17

Exercise

Transform the following generally left-recursive grammar

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

into an equivalent immediately left-recursive grammar.

OS2MAL 2012/13, Lecture 14 – p.16/17

Elimination of Left Recursion (6)

Here is the grammar again:

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

An equivalent right-recursive grammar:

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

OS2MAL 2012/13, Lecture 14 – p.16/17

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. Aho, Sethi, and Ullman (1986) for details.)

OS2MAL 2012/13, Lecture 14 – p.17/17

Solution

First:

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

Then:

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

Or, eliminating B completely:

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

OS2MAL 2012/13, Lecture 14 – p.17/17

Elimination of Left Recursion (7)

Derivation of $aabeeeecdbeec$ in the new grammar:

$$\begin{aligned}
 S &\Rightarrow A \Rightarrow aA' \Rightarrow aAddA' \Rightarrow aaA'ddA' \\
 &\Rightarrow aaBcA'ddA' \\
 &\Rightarrow aabB'cA'ddA' \\
 &\Rightarrow aabeeB'cA'ddA' \\
 &\Rightarrow aabeeeeB'cA'ddA' \\
 &\Rightarrow aabeeeecA'ddA' \\
 &\Rightarrow aabeeeecdA' \\
 &\Rightarrow aabeeeecdBcA' \\
 &\Rightarrow aabeeeecdBbB'cA' \\
 &\Rightarrow aabeeeecdBeeB'cA' \\
 &\Rightarrow aabeeeecdBeeecA' \Rightarrow aabeeeecdbeec
 \end{aligned}$$

OS2MAL 2012/13, Lecture 14 – p.12/17

General Left Recursion (2)

For example, the generally left-recursive grammar

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

is first transformed into the immediately left-recursive grammar

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

OS2MAL 2012/13, Lecture 14 – p.15/17