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

Elaboration:

• A grammar is left-recursive if there is some non-terminal $A$ such that $A \rightarrow A\alpha$.

• 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:

$$G_1: \quad S \rightarrow \epsilon | A \\ A \rightarrow a | aA$$

$$G_2: \quad S \rightarrow A \\ A \rightarrow \epsilon | A\alpha$$

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

(The equivalence of CFGs is in general undecidable.)

Exercise

• The following grammar $G_1$ is immediately left-recursive:

$$A \rightarrow b | Aa$$

Draw the derivation tree for $baa$ using $G_1$.

• The following is a non-left-recursive grammar $G_1'$ equivalent to $G_1$:

$$A \rightarrow bA'$$

$$A' \rightarrow aA' | \epsilon$$

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 $A_B$ such that no $\beta$ begins with an $A$.

Then replace the $A$ productions by

$$A \rightarrow \beta_1A' | \beta_2A' | \ldots | \beta_mA'$$

$$A' \rightarrow \alpha_1A' | \alpha_2A' | \ldots | \alpha_mA' | \epsilon$$

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

This Lecture

• The problem of recursive-descent parsing and left recursive grammars.

• Elimination of left recursion.

Left Recursion

Consider: $A \rightarrow Aa | \epsilon$

Parsing function:

```haskell```
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.

Elimination of Left Recursion (4)

Consider the (immediately) left-recursive grammar:

$$S \rightarrow A | B$$

$$A \rightarrow A| A|a | aA$$

$$B \rightarrow B | b$$

Terminal strings derivable from $B$ include:

$$b, bee, beee, \ldots$$

Terminal strings derivable from $A$ include:

$$a, aa, aadd, aaddl, abc, aabc, abecc, aabcecc, aabecceccbdbecc$$
Elimination of Left Recursion (5)

Let us do a leftmost derivation of \( aabeeeecddbeec \):

\[
S \Rightarrow A \\
A \Rightarrow ABc \\
A \Rightarrow AAddBc \\
A \Rightarrow aaBcddBc \\
A \Rightarrow aabBeecddBc \\
A \Rightarrow aabeeeecddBc \\
A \Rightarrow aabeeeecddBeec \\
A \Rightarrow aabeeeecddbeec
\]

Elimination of Left Recursion (6)

Here is the grammar again:

\[
S \rightarrow A | B \\
A \rightarrow ABc | AAdd | a | aa \\
B \rightarrow Bee | b
\]

An equivalent right-recursive grammar:

\[
S \rightarrow A | B \\
A \rightarrow aA' | aaA' \\
A' \rightarrow BcA | AddA | \varepsilon \\
B \rightarrow bB' | \varepsilon
\]

General Left Recursion (1)

To eliminate \textit{general} left recursion:

- first transform the grammar into an \textit{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. Aho, Sethi, and Ullman (1986) for details.)

General Left Recursion (2)

For example, the generally left-recursive grammar

\[
A \rightarrow BaB \\
B \rightarrow Cb | \varepsilon \\
C \rightarrow Ab | Ac
\]

is first transformed into the immediately left-recursive grammar

\[
A \rightarrow A'b \\
A \rightarrow A'c \\
A \rightarrow a
\]

Solution (1)

First:

\[
A \rightarrow BaB \\
B \rightarrow Abb | A'cb | \varepsilon
\]

Then:

\[
A \rightarrow A'baB | A'cbA | aB \\
B \rightarrow Abb | A'cb | \varepsilon
\]

Or, eliminating \( B \) completely:

\[
A \rightarrow A'baB | A'cbA | aB \\
| A'baA | A'cbA | aA' \\
| A' | A'
\]

Solution (2)

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

\[
A \rightarrow A'baB | A'cbA | aB \\
B \rightarrow Abb | A'cb | \varepsilon
\]

Only productions for \( A \) are immediately left-recursive. Applying the elimination transformation:

\[
A \rightarrow aB' \\
A' \rightarrow bbaB' | cbaB' | \varepsilon \\
B \rightarrow Abb | A'cb | \varepsilon
\]

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