

Parsing : Part-II

PREDICTIVE PARSING

- The goal is to construct a top-down parser that never backtracks
- Always leftmost derivations
- We must transform a grammar in two ways:
 - eliminate left recursion
 - perform left factoring
- These rules eliminate most common causes for backtracking although they do not guarantee a completely backtrack-free parsing

LEFT RECURSION: INFINITE LOOPING PROBLEM

- ❑ A grammar is left-recursive if it has a non-terminal A, such that there is a derivation :

$$A \xRightarrow{+} A\alpha, \text{ for some string } \alpha.$$

- ❑ Top-Down parsing can't reconcile this type of grammar, **since it could consistently make choice which wouldn't allow termination.**

$$A \Rightarrow A\alpha \Rightarrow A\alpha\alpha \Rightarrow A\alpha\alpha\alpha \dots \text{ etc. } A \rightarrow A\alpha \mid \beta$$

- ❑ So we have to convert our left-recursive grammar into an equivalent grammar which is not left-recursive.
- ❑ The left-recursion may appear in a single step of the derivation (*immediate left-recursion*), or may appear in more than one step of the derivation.

IMMEDIATE LEFT RECURSION

➤ $A \rightarrow A \alpha \mid \beta$ where β does not start with A

\Downarrow eliminate immediate left recursion

$A \rightarrow \beta A'$ where A' is a new nonterminal

$A' \rightarrow \alpha A' \mid \varepsilon$ an equivalent grammar

More General (but still immediate):

$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \dots \mid \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots$

Transform into:

$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \beta_3 A' \mid \dots$

$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \alpha_3 A' \mid \dots \mid \varepsilon$

IMMEDIATE LEFT RECURSION ELIMINATION: EXAMPLE

Our Example :

$$\begin{array}{lcl} \mathbf{E} \rightarrow \mathbf{E} + \mathbf{T} \mid \mathbf{T} & \longrightarrow & \left\{ \begin{array}{l} \mathbf{E} \rightarrow \mathbf{T}\mathbf{E}' \\ \mathbf{E}' \rightarrow + \mathbf{T}\mathbf{E}' \mid \epsilon \end{array} \right. \\ \mathbf{T} \rightarrow \mathbf{T} * \mathbf{F} \mid \mathbf{F} & \longrightarrow & \left\{ \begin{array}{l} \mathbf{T} \rightarrow \mathbf{F}\mathbf{T}' \\ \mathbf{T}' \rightarrow * \mathbf{F}\mathbf{T}' \mid \epsilon \end{array} \right. \\ \mathbf{F} \rightarrow (\mathbf{E}) \mid \mathbf{id} & \longrightarrow & \mathbf{F} \rightarrow (\mathbf{E}) \mid \mathbf{id} \end{array}$$

LEFT RECURSION IN MORE THAN ONE STEP

- A grammar cannot be immediately left-recursive, but it still can be left-recursive.
- By just eliminating the immediate left-recursion, we may not get a grammar which is not left-recursive.

Example:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow A\underline{c} \mid S\underline{d} \mid \underline{e}$

Is A left recursive? Yes.

Is S left recursive? Yes, but not immediate left recursion. $S \Rightarrow A\underline{f} \Rightarrow S\underline{d}f$

LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

Approach:

Look at the rules for S only (ignoring other rules)... No left recursion.

Look at the rules for A ...

Do any of A 's rules start with S ? Yes.

$$A \rightarrow Sd$$

Get rid of the S . Substitute in the righthand sides of S .

$$A \rightarrow Afd \mid \underline{bd}$$

The modified grammar:

$$S \rightarrow Af \mid \underline{b}$$

$$A \rightarrow A\underline{c} \mid Afd \mid \underline{bd} \mid \underline{e}$$

Now eliminate immediate left recursion involving A .

$$S \rightarrow Af \mid \underline{b}$$

$$A \rightarrow \underline{bd}A' \mid \underline{e}A'$$

$$A' \rightarrow \underline{c}A' \mid \underline{fd}A' \mid \underline{\epsilon}$$

LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

The Original Grammar:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow A\underline{c} \mid S\underline{d} \mid B\underline{e}$

$B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

So Far:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow \underline{b}dA' \mid B\underline{e}A'$

$A' \rightarrow \underline{c}A' \mid \underline{f}dA' \mid \epsilon$

LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

The Original Grammar:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow A\underline{c} \mid S\underline{d} \mid B\underline{e}$

$B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

So Far:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow \underline{b}dA' \mid B\underline{e}A'$

$A' \rightarrow \underline{c}A' \mid \underline{f}dA' \mid \epsilon$

$B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

Look at the B rules next;
Does any righthand side
start with "S"?

LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

The Original Grammar:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow A\underline{c} \mid S\underline{d} \mid B\underline{e}$

$B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

So Far:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow \underline{b}dA' \mid B\underline{e}A'$

$A' \rightarrow \underline{c}A' \mid \underline{f}dA' \mid \epsilon$

$B \rightarrow A\underline{g} \mid A\underline{f}h \mid \underline{b}h \mid \underline{k}$

Substitute, using the rules for “S”

$A\underline{f}\dots \mid \underline{b}\dots$

LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

The Original Grammar:

$S \rightarrow Af \mid \underline{b}$

$A \rightarrow A\underline{c} \mid S\underline{d} \mid B\underline{e}$

$B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

So Far:

$S \rightarrow Af \mid \underline{b}$

$A \rightarrow \underline{b}dA' \mid B\underline{e}A'$

$A' \rightarrow \underline{c}A' \mid \underline{f}dA' \mid \epsilon$

$B \rightarrow \underline{A}g \mid \underline{A}f\underline{h} \mid \underline{b}h \mid \underline{k}$

Does any righthand side
start with “A”?

LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

The Original Grammar:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow A\underline{c} \mid S\underline{d} \mid B\underline{e}$

$B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

So Far:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow \underline{b}dA' \mid B\underline{e}A'$

$A' \rightarrow \underline{c}A' \mid \underline{f}dA' \mid \epsilon$

$B \rightarrow \underline{b}dA'g \mid B\underline{e}A'g \mid A\underline{f}h \mid \underline{b}h \mid \underline{k}$



Substitute, using the rules for “A”

$\underline{b}dA'... \mid B\underline{e}A'...$

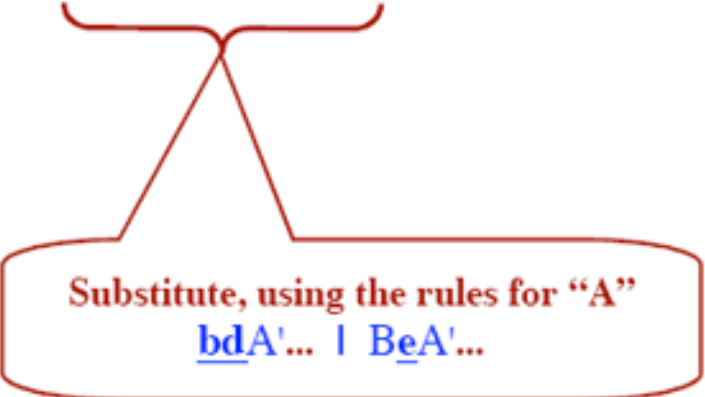
LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

The Original Grammar:

$S \rightarrow A\underline{f} \mid \underline{b}$
 $A \rightarrow A\underline{c} \mid S\underline{d} \mid B\underline{e}$
 $B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

So Far:

$S \rightarrow A\underline{f} \mid \underline{b}$
 $A \rightarrow \underline{b}A' \mid B\underline{e}A'$
 $A' \rightarrow \underline{c}A' \mid \underline{f}A' \mid \epsilon$
 $B \rightarrow \underline{b}A'\underline{g} \mid B\underline{e}A'\underline{g} \mid \underline{b}A'\underline{f}h \mid B\underline{e}A'\underline{f}h \mid \underline{b}h \mid \underline{k}$



Substitute, using the rules for “A”
 $\underline{b}A' \dots \mid B\underline{e}A' \dots$

LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

The Original Grammar:

$S \rightarrow A\underline{f} \mid \underline{b}$
 $A \rightarrow A\underline{c} \mid S\underline{d} \mid B\underline{e}$
 $B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

So Far:

$S \rightarrow A\underline{f} \mid \underline{b}$
 $A \rightarrow \underline{b}dA' \mid B\underline{e}A'$
 $A' \rightarrow \underline{c}A' \mid \underline{f}dA' \mid \epsilon$
 $B \rightarrow \underline{b}dA'g \mid B\underline{e}A'g \mid \underline{b}dA'fh \mid B\underline{e}A'fh \mid \underline{b}h \mid \underline{k}$

Finally, eliminate any immediate
Left recursion involving "B"

Next Form

$S \rightarrow A\underline{f} \mid \underline{b}$
 $A \rightarrow \underline{b}dA' \mid B\underline{e}A'$
 $A' \rightarrow \underline{c}A' \mid \underline{f}dA' \mid \epsilon$
 $B \rightarrow \underline{b}dA'gB' \mid \underline{b}dA'fhB' \mid \underline{b}hB' \mid \underline{k}B'$
 $B' \rightarrow \underline{e}A'gB' \mid \underline{e}A'fhB' \mid \epsilon$

LEFT RECURSION IN MORE THAN ONE STEP: ELIMINATION

The Original Grammar:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow A\underline{c} \mid S\underline{d} \mid B\underline{e} \mid C$

$B \rightarrow A\underline{g} \mid S\underline{h} \mid \underline{k}$

$C \rightarrow B\underline{k}mA \mid AS \mid \underline{j}$

If there is another nonterminal,
then do it next.

So Far:

$S \rightarrow A\underline{f} \mid \underline{b}$

$A \rightarrow \underline{b}dA' \mid B\underline{e}A' \mid CA'$

$A' \rightarrow \underline{c}A' \mid \underline{f}dA' \mid \epsilon$

$B \rightarrow \underline{b}dA'\underline{g}B' \mid \underline{b}dA'\underline{f}hB' \mid \underline{b}hB' \mid \underline{k}B' \mid CA'\underline{g}B' \mid CA'\underline{f}hB'$

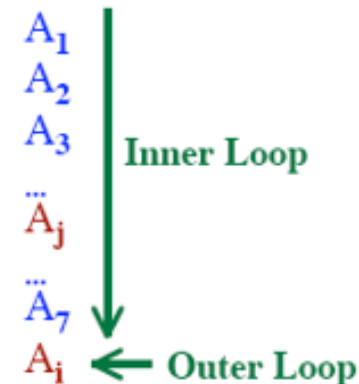
$B' \rightarrow \underline{e}A'\underline{g}B' \mid \underline{e}A'\underline{f}hB' \mid \epsilon$

ALGORITHM FOR ELIMINATING LEFT RECURSION

Assume the nonterminals are ordered A_1, A_2, A_3, \dots

(In the example: S, A, B)

```
for each nonterminal  $A_i$  (for  $i = 1$  to  $N$ ) do  
  for each nonterminal  $A_j$  (for  $j = 1$  to  $i-1$ ) do  
    Let  $A_j \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots \mid \beta_N$  be all the rules for  $A_j$   
    if there is a rule of the form  
       $A_i \rightarrow A_j \alpha$   
    then replace it by  
       $A_i \rightarrow \beta_1 \alpha \mid \beta_2 \alpha \mid \beta_3 \alpha \mid \dots \mid \beta_N \alpha$   
    endIf  
  endFor  
  Eliminate immediate left recursion  
    among the  $A_i$  rules  
endFor
```



Left Factoring: Common Prefix Problem

Problem : Uncertain which of 2 rules to choose:

$stmt \rightarrow \text{if } expr \text{ then } stmt \text{ else } stmt$
 $\quad | \text{if } expr \text{ then } stmt$

When do you know which one is valid ?

What's the general form of $stmt$?

$A \rightarrow \alpha\beta_1 \mid \alpha\beta_2$ $\alpha : \text{if } expr \text{ then } stmt$
 $\beta_1 : \text{else } stmt \quad \beta_2 : \in$

Transform to:

$A \rightarrow \alpha A'$

$A' \rightarrow \beta_1 \mid \beta_2$

EXAMPLE:

$stmt \rightarrow \text{if } expr \text{ then } stmt \text{ rest}$

$rest \rightarrow \text{else } stmt \mid \in$

Left Factoring : Example

$A \rightarrow \underline{ab}B \mid \underline{a}B \mid \underline{cd}g \mid \underline{cde}B \mid \underline{cdf}B$

\Downarrow

$A \rightarrow \underline{a}A' \mid \underline{cd}g \mid \underline{cde}B \mid \underline{cdf}B$

$A' \rightarrow \underline{b}B \mid B$

\Downarrow

$A \rightarrow \underline{a}A' \mid \underline{cd}A''$

$A' \rightarrow \underline{b}B \mid B$

$A'' \rightarrow g \mid \underline{e}B \mid \underline{f}B$

Left Factoring : Example

$A \rightarrow ad \mid a \mid ab \mid abc \mid b$

\Downarrow

$A \rightarrow aA' \mid b$

$A' \rightarrow d \mid \varepsilon \mid b \mid bc$

\Downarrow

$A \rightarrow aA' \mid b$

$A' \rightarrow d \mid \varepsilon \mid bA''$

$A'' \rightarrow \varepsilon \mid c$

Reading Materials

- Chapter -4 of your Text book:
 - Compilers: Principles, Techniques, and Tools

THE END
