

# Parsing

## Part II

## Writing Grammars

When writing a grammar (or RE) for some language, the following must be true:

1. All strings generated are in the language.
2. Your grammar produces all strings in the language.

Example:

$$S \rightarrow (S) S \mid \varepsilon$$

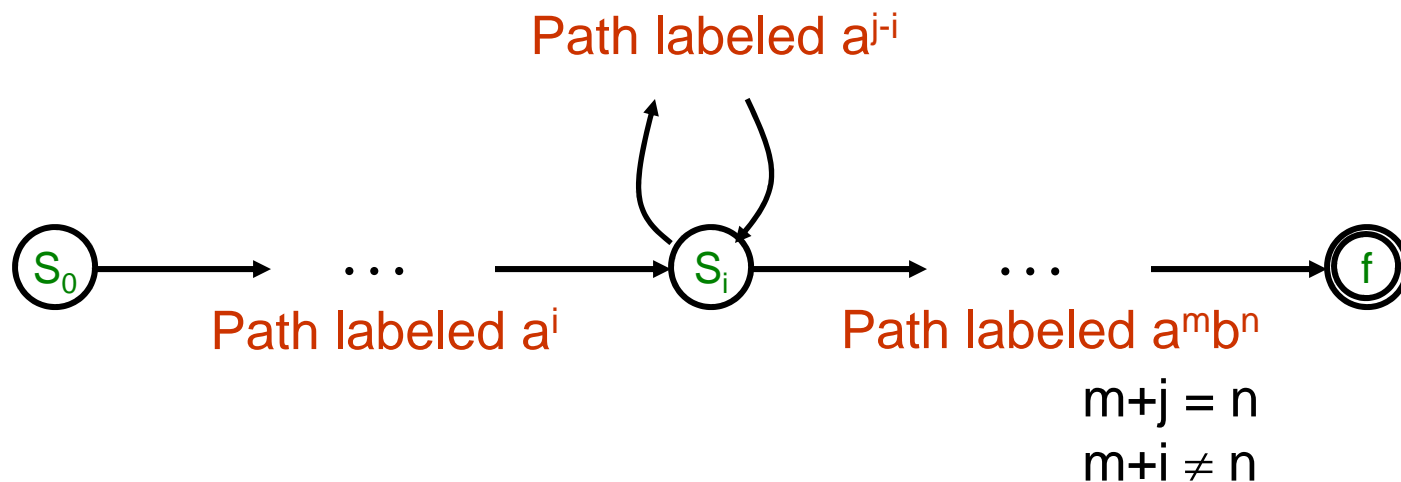
Generates all strings of balanced parentheses

Using induction show that

Every sentence derivable from S is balanced

Every balanced string is derivable from S

- $L = \{ a^n b^n \mid n \geq 1 \}$
- Show that  $L$  can be described by a grammar not by a regular expression
- Construct a DFA  $D$  with  $k$  states to accept  $L$
- For  $a^n b^n$  ( $n > k$ ) some state ( $s_i$ ) of  $D$  must be entered twice



# Elimination of Ambiguity

## Ambiguous Grammar

- A Grammar is ambiguous if there are multiple parse trees for the same sentence
- For the most parsers, the grammar must be unambiguous

## Unambiguous grammar

unique selection of the parse tree for a sentence

## Disambiguation

- Express Preference for one parse tree over others
  - Add disambiguating rule into the grammar

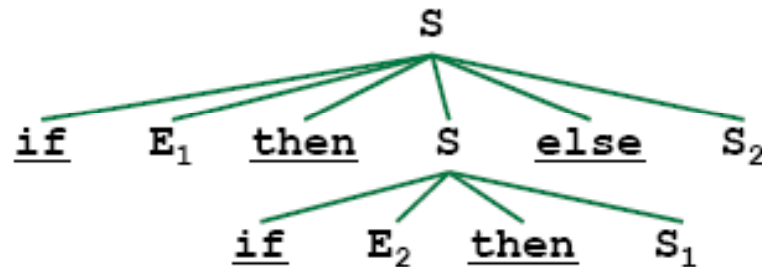
# Dangling-else grammar

This grammar is ambiguous!

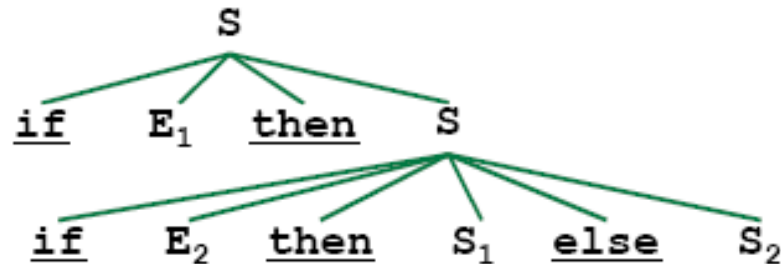
Stmt  $\rightarrow$  if Expr then Stmt  
 $\rightarrow$  if Expr then Stmt else Stmt  
 $\rightarrow$  ...Other Stmt Forms...

Example String: if  $E_1$  then if  $E_2$  then  $S_1$  else  $S_2$

Interpretation #1: if  $E_1$  then (if  $E_2$  then  $S_1$ ) else  $S_2$



Interpretation #2: if  $E_1$  then (if  $E_2$  then  $S_1$  else  $S_2$ )



# Dangling-else grammar

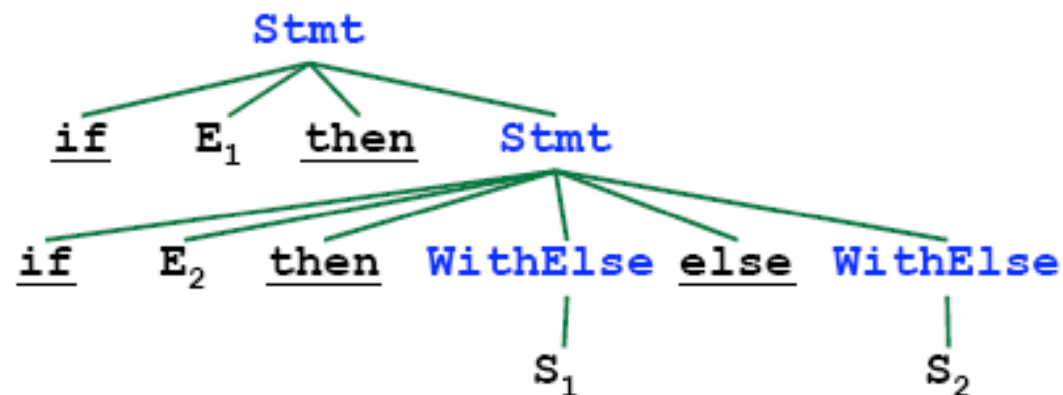
**Goal:** “Match else-clause to the closest if without an else-clause already.”

**Solution:**

Stmt → if Expr then Stmt  
→ if Expr then WithElse else Stmt  
→ ...Other Stmt Forms...  
WithElse → if Expr then WithElse else WithElse  
→ ...Other Stmt Forms...

Any Stmt occurring between then and else must have an else.  
i.e., the Stmt must not end with “then Stmt”.

**Interpretation #2:** if E<sub>1</sub> then (if E<sub>2</sub> then S<sub>1</sub> else S<sub>2</sub>)



## Dangling-else grammar

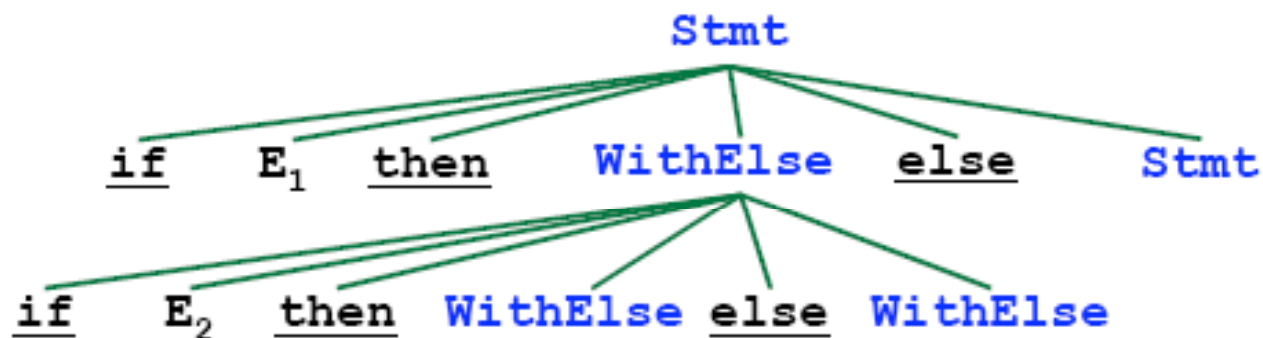
**Goal:** “Match else-clause to the closest if without an else-clause already.”

**Solution:**

Stmt      → if Expr then Stmt  
            → if Expr then WithElse else Stmt  
            → ...Other Stmt Forms...  
WithElse → if Expr then WithElse else WithElse  
            → ...Other Stmt Forms...

Any Stmt occurring between then and else must have an else.  
i.e., the Stmt must not end with “then Stmt”.

**Interpretation #1:** if E<sub>1</sub> then (if E<sub>2</sub> then S<sub>1</sub>) else S<sub>2</sub>



# Left Recursion

Whenever

$$A \Rightarrow^+ A\alpha$$

Simplest Case: Immediate Left Recursion

Given:

$$A \rightarrow A\alpha \mid \beta$$

Transform into:

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \varepsilon$$

where  $A'$  is a new nonterminal

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

- Grammar

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow ( E ) \mid \mathbf{id}$$

- **Left recursion Eliminated**

$$E \rightarrow T E'$$
$$E' \rightarrow + T E' \mid \varepsilon$$
$$T \rightarrow F T'$$
$$T' \rightarrow * F T' \mid \varepsilon$$
$$F \rightarrow ( E ) \mid \mathbf{id}$$

## Left Recursion in More Than One Step

### 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.  $S \Rightarrow A\underline{f} \Rightarrow S\underline{d}\underline{f}$

### 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.

$A \rightarrow S\underline{d}$

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

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

The modified grammar:

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

$A \rightarrow A\underline{c} \mid A\underline{f}\underline{d} \mid \underline{b}\underline{d} \mid \underline{e}$

Now eliminate immediate left recursion involving  $A$ .

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

$A \rightarrow \underline{b}\underline{d}A' \mid \underline{e}A'$

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

## Left Recursion in More Than One Step

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:

$$S \rightarrow A\underline{f} \mid \underline{b}$$
$$A \rightarrow \underline{b}\underline{d}A' \mid \textcolor{red}{B}\underline{e}A'$$
$$A' \rightarrow \underline{c}A' \mid \underline{f}\underline{d}A' \mid \epsilon$$

# Left Recursion in More Than One Step

## 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

*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

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:

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

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

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

$B \rightarrow \underline{Ag} \mid \underline{Afh} \mid \underline{bh} \mid \underline{k}$

Does any righthand side  
start with “A”?

# Left Recursion in More Than One Step

The Original Grammar:

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

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

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

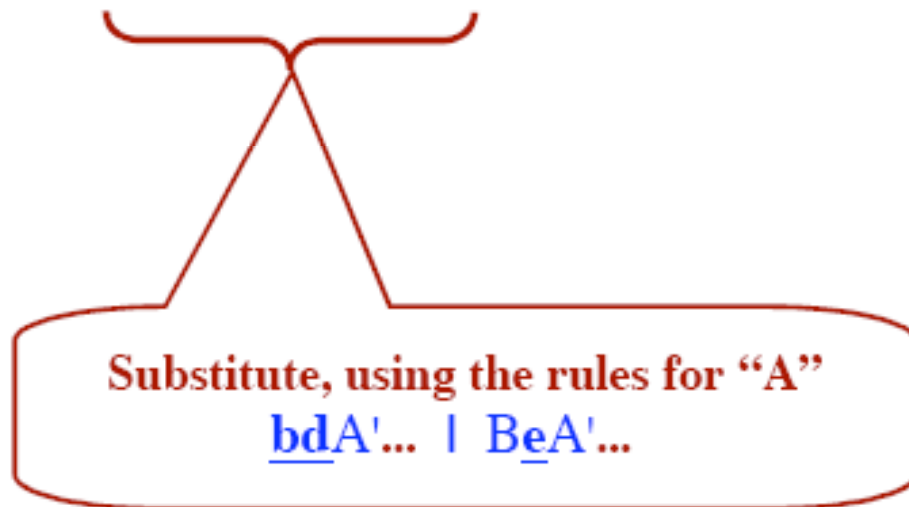
So Far:

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

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

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

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



# Left Recursion in More Than One Step

## The Original Grammar:

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

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

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

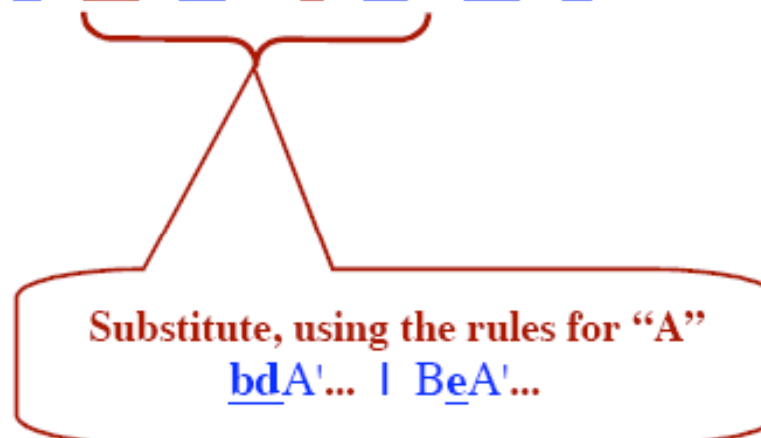
## So Far:

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

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

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

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





# Left Recursion in More Than One Step

## 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:

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

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

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

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

Finally, eliminate any immediate  
Left recursion involving “B”

## Next Form

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

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

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

$B \rightarrow \underline{b}dA'gB' \mid \underline{b}dA'fhB' \mid \underline{b}hB' \mid \underline{k}B'$

$B' \rightarrow \underline{e}A'gB' \mid \underline{e}A'fhB' \mid \epsilon$

## Left Recursion in More Than One Step

### 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:

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

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

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

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

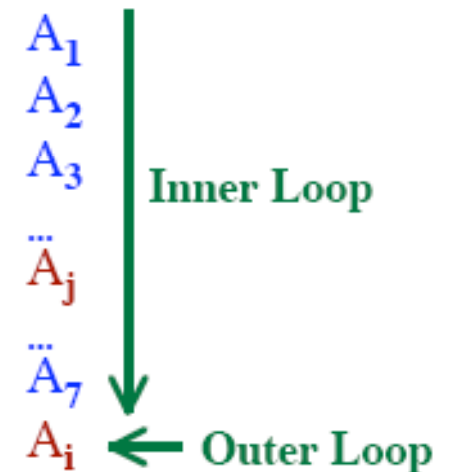
$B' \rightarrow \underline{e}A'\underline{g}B' \mid \underline{e}A'\underline{f}hB' \mid \epsilon$

# 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

## Problem:

Stmt  $\rightarrow$  if Expr then Stmt else Stmt  
 $\rightarrow$  if Expr then Stmt  
 $\rightarrow$  OtherStmt

With predictive parsing, we need to know which rule to use!  
(While looking at just the next token)

## Solution:

Stmt  $\rightarrow$  if Expr then Stmt ElsePart  
 $\rightarrow$  OtherStmt

ElsePart  $\rightarrow$  else Stmt  $\mid \epsilon$

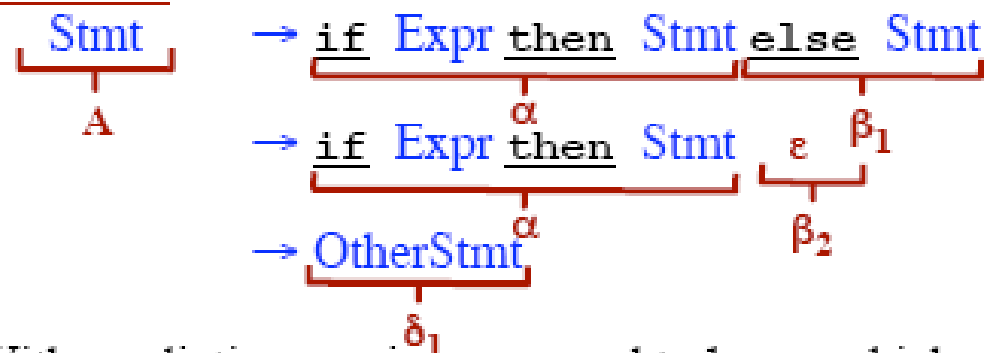
## General Approach:

Before: A  $\rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid \dots \mid \delta_1 \mid \delta_2 \mid \delta_3 \mid \dots$

After: A  $\rightarrow \alpha C \mid \delta_1 \mid \delta_2 \mid \delta_3 \mid \dots$   
C  $\rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots$

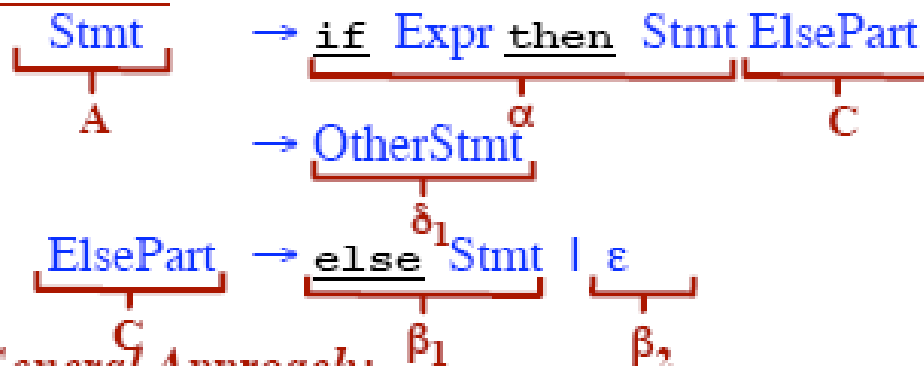
# Left Factoring

Problem:



With predictive parsing, we need to know which rule to use!  
(While looking at just the next token)

Solution:



General Approach:

Before:  $A \rightarrow \alpha\beta_1 \mid \alpha\beta_2 \mid \alpha\beta_3 \mid \dots \mid \delta_1 \mid \delta_2 \mid \delta_3 \mid \dots$

After:  $A \rightarrow \alpha C \mid \delta_1 \mid \delta_2 \mid \delta_3 \mid \dots$

$C \rightarrow \beta_1 \mid \beta_2 \mid \beta_3 \mid \dots$