Lidando com ambiguidade e recursão à esquerda

$E \to n$
$E \to E \cdot E$
$E \to E \times E$
$E \to (E)$

↝ Um analisador sintático top-down tem várias funções como

```
func parseX():
    x prox ∈ First(α):
        parse "x → α"
    x prox ∈ First(β):
        parse "x → β"
```

↝ Um problema ocorre se duas regras têm um prefixo comum

$X \to ab$
$X \to ac$

⇓ podemos alterar para:

$X \to aY$
$Y \to b$
$Y \to c$

↳ Após a refatoração, o parser para a X:

```
func parseX():
    x prox == "a":
        a = come("a")
        x prox == "b"
            b = come("b")
            return X
                   / \
                  a   b
        x prox = "c"
            c = come("c")
```

return X
/ \
a c

ou

```
func parseX():
    if prox == "a":
        a = come("a")
        return parseY(a)
    else
        ERRO!
```

```
func parseY():
    if prox == "b":
        b = come("b")
        return Y
               / \
              a   b
    if prox == "c":
        c = come("c")
        return Y
               / \
              a   c
```

Recursão à Esquerda

$A \to b$
$A \to Ac$

↳ bcc sempre começe com b

Vamos transformar para ser recursivo à esquerda?

$A \to bR$
$R \to cR$
$R \to \varepsilon$

que vai ao mesmo que:

$A \to b(A|A)$
$(A|A) \to c(A|A)$
$(A|A) \to b$

Vamos construir a árvore de "baixo para cima"

```
func parseA():
    b = come("b")
    true = A
    while próx == "a"
        a = come("a")
        true = A
                / \
               /   \
            true    a
    return true
```

ou recursivamente:

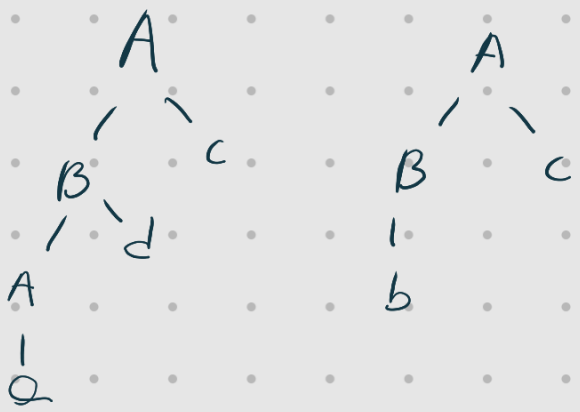
```
func parseA():
    b = come("b")
    return parseR(A
                  |
                  b)

func parseR():
    se próx == "a"
        a = come("a")
        return parseR(A
                      / \
                     /   \
                  true    a)
    se próx == EOF
        return true
    else
        ERRO!
```

Recursão Esquerda Indireta

$A \to Bc$
$A \to a$

$B \to Ad$
$B \to b$

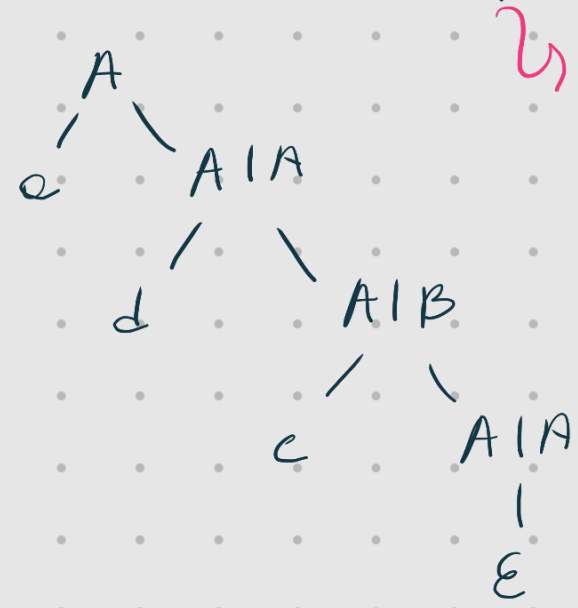


transformando

$A \to a\ (A|A)$
$A \to b\ (A|B)$
$A|B \to c\ (A|A)$

$A|A \to d\ (A|B)$
$A|A \to \varepsilon$
$A|B \to \varepsilon$

Uma gramática transformada



Os parses funcionam de cima para baixo

mas asamos uma bottom-up para devolver a original