Removing indirect left recursion

- Pseudocode from Cooper & Torczon:

```
impose an order on the nonterminals, A1, A2, ..., An

for i ← 1 to n do;
    for j ← 1 to i - 1 do;
        if ∃ a production Ai → Aj γ
            then replace Ai → Aj γ with one or more
                 productions that expand Aj
    end;
    rewrite the productions to eliminate
         any direct left recursion on Ai
end;
```

■ **FIGURE 3.6** Removal of Indirect Left Recursion.

- Rather conservative: no need to push A_j into A_i if you know that $A_j \not\Rightarrow \alpha A_i \beta$ for any α, β

Grammar 1

0. S ::= a B | a w
1. B ::= C x | y
2. C ::= ε | z

Grammar 2

0. S ::= a B
1. B ::= C x | y
2. C ::= ε | x

Grammar 3

0. S ::= S B | a | w
1. B ::= C x | y
2. C ::= ε | z

Grammar 4

0. S ::= B w | a B
1. B ::= S | z x

Example pseudocode of a recursive-descent parser for grammar 2:

```
// S ::= aB
void S() {
    match(a)
    B()
}
```

```
// B ::= Cx | y
void B() {
    if next is y {
        match(y)
    } else {
        C()
        match(x)
    }
}
```

```
// C ::= ε | x
// *One* implementation of
// the C() method
void C() {
    if next is x {
        match(x)
    } else if (lookahead is
        not in follow set for C){
        error
    }
    // else, epsilon so do
    // nothing
}
```