LL(1) Transforms

**1) Eliminate immediate left recursion:**

F -> F α

-> F β

-> **γ**

-> **δ**

Becomes:

F -> **γ** F’

-> **δ** F’

F’ -> α F’

F’ -> β F’

-> λ

Process:

1. Identify the productions that have immediate left recursion
   1. Mark the rest of the productions (all the but left recursive term) in red
   2. Mark the non-left recursive productions in **blue**
2. Create a new non-terminal (F’)
   1. Productions are

red F’

λ

**2) Eliminate indirect left recursion**

A -> B x y

B -> A m n

B -> C x y

C -> B t u

Productions for A are OK because there is no immediate left recursion, and this is the first NT

First production for B must be fixed because it starts with A (earlier in the list)

B -> A m n

Becomes

B -> **B x y** m n

Fix immediate left recursion:

B -> B’ (there is no **blue**)

B’ -> x y m n B’

-> λ

Need to fix production for C because it starts with B (earlier in list)

C -> B t u

Becomes

C -> B’ t u

-> C x y

Fix immediate left recursion:

C -> B’ t u C’

C’ -> x y C’

-> λ

Final grammar:

A -> B x y

B -> B’

B -> C x y

B’ -> x y m n B’

-> λ

C -> B’ t u C’

C’ -> x y C’

-> λ

**3) Left Factoring**

Term -> **Fact Op** Term

-> **Fact Op** num

Step 1: Factor **Fact Op**. The factored part stays put and is followed by a new Non-Terminal. The new Non-Terminal is the parts after the factored portion.

Term -> **Fact Op** Term’

Term’ -> Term

-> num