**First e Follow:**

es: il simbolo iniziale S produce, in zero o più passi:

S →\* βAγ

e l'input da controllare è βbδ (con δ diverso da γ).

ci sono 2 possibilità:

1) b appartiene ad un'espansione di A, tale che A → α e b è il primo terminale di α.

In questo caso, b € First(A)

2) A si annulla e b appartiene ad un'espansione di γ.

In questo caso, b € Follow(A) e epsilon € First(A).

**First:**

**Definizione:**

First(X) = { b | X → \* bα} ∪ {ε | X → \* ε}

**Calcolare i First:**

1. First(b) = { b }

2. per ogni produzione X → A1 ... An:

Add First(A 1 ) – {ε} to First(X). Stop if ε ∉ First(A 1 )

Add First(A 2 ) – {ε} to First(X). Stop if ε ∉ First(A 2 )

...

Add First(A n ) – {ε} to First(X). Stop if ε ∉ First(A n )

Add ε to First(X)

3. ripeti lo step 2 finché l'insieme dei First non cresce più

**Follow:**

**Definizione:**

Follow(X) = { b | S → \* β X b δ }

**Calcolare i Follow:**

1. Add $ to Follow(S) (if S is the start non-terminal)

2. For all productions Y → ... X A1 ... An

Add to Follow(X) First(A1 ... An ) – {ε}

(i … prima della X dicono che devi scorrere l'intero corpo della produzione, e fare quest'operazione per ogni fetta)

and for all productions Y → αX or Y → αXβ with ε ∈ First(β)

Add to Follow(X) the Follow(Y)

3. ripeti lo step 2 finché l'insieme dei Follow non cresce più

**Esempio: calcola i First della seguente grammatica fattorizzata a sinistra:**



1) ogni terminale è il first di se stesso => First(b) = {b}

-----------------------

First ( ( ) = { ( }

First ( ) ) = { ) }

First ( + ) = { + }

First ( \* ) = { \* }

First ( int ) = { int }

-----------------------------

2) Y → A1 A2 … An

add to First(Y) il First(A1) - {epsilon}, se epsilon non appartiene a First(A1) termina

add to First(Y) il First(A2) - {epsilon}, se epsilon non appartiene a First(A2) termina

…

add to First(Y) il First(An) - {epsilon}, se epsilon non appartiene a First(An) termina

add to First(Y) epsilon

--------------------------

Considero la produzione:

E → TX, First ( E ) = First( T ) = { (, int}

X → +E, First( X ) = { + }

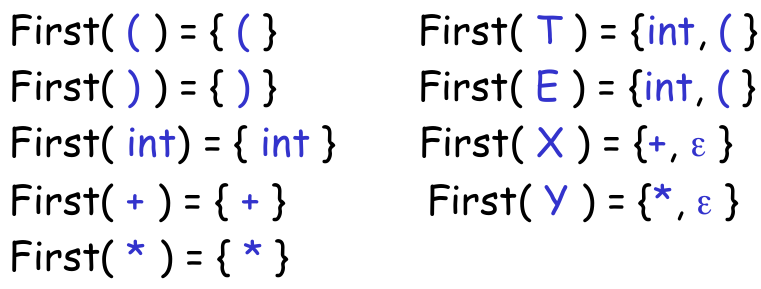
Y → \*T, First( Y ) = { \* }

--------------------------------

3) ripeti il punto 2 finché l'insieme dei First(Y) non cresce più

Non c'è bisogno.

Quindi l'insieme di tutti i first è:



**Esempio: calcola i First della seguente grammatica fattorizzata a sinistra:**



1) E è il simbolo iniziale, quindi:

-------------

Follow( E ) = { $ }

---------------

2a) Y → … X A1 A2 … An, add to Follow(X) i First(A1 A2 … An) - {epsilon}

-------------------

Considero la produzione:

E → TX, add to Follow( T ) i First( X ) = { + }

T → (E), add to Follow( ( ) i First( E) ) = { (, int }

add to Follow( E ) i First( ) ) = { $ } + { ) } = { ), $ }

T → intY, add to Follow( int ) i First( Y ) = { \* }

X → +E, add to Follow( + ) i First( E ) = { (, int }

Y → \*T, add to Follow( \* ) i First( T ) = { (, int }

-------------------

2b) Y → AX OR Y → AXB, con epsilon che appartiene ai First(B):

add to Follow(X) i Follow(Y)

------------------------------------

Considero la produzione:

E → TX, add to Follow( X ) i Follow( E ) => {} + { ), $ } = { ), $ }

dato che epsilon appartiene ai First(X):

add to Follow( T ) i Follow( E ) => { + } + { ), $ } = { +, ), $ }

T → (E), add to Follow( ) ) i Follow( T ) => {} + { +, ), $ } = { +, ), $ }

T → intY, add to Follow( Y ) i Follow( T ) => {} + { +, ), $ } = { +, ), $ }

dato che epsilon appartiene a First(Y):

add to Follow( int ) i Follow( T ) => { \* } + { +, ), $ } = { \*, +, ), $ }

X → +E, add to Follow( E ) i Follow( X ) => sono lo stesso insieme

Y → \*T, add to Follow( T ) i Follow( Y ) => sono lo stesso insieme

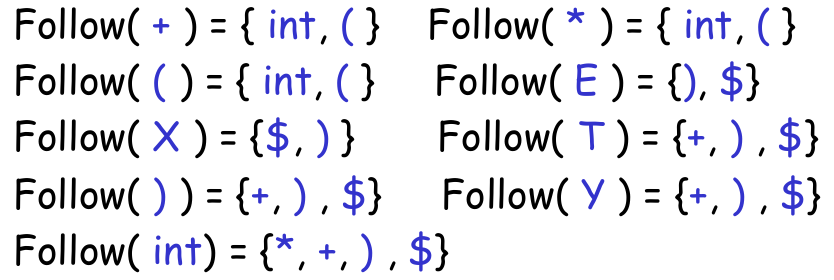
--------------------------

3) ripeti lo step 2b finché l'insieme dei Follow non cresce più

Non c'è ne bisogno.

-------------------

Quindi l'insieme dei Follow finale è:



**LL1 Parsing Table:**

2 dimensioni: variabili e terminali, ogni cella è T[A,b]

per ogni produzione A → α:

- per ogni b € First(α) fai: T[A,b] = α

- se epsilon € First(α), fai: per ogni b € Follow(A) fai: T[A,b] = α

Esempio:

Data la grammatica:



Considero E → TX

First(TX) = { int, ( } => T[ E,int ] = TX, T[ E,( ] = TX

Considero T → (E)

First( (E) ) = { ( } => T[ T,( ] = (E)

Considero T → intY

First( intY ) = {int} => T[ T,int ] = intY

Considero X → +E:

First(+E) = {+} => T[ X,+ ] = +E

Considero X → epsilon

dato che epsilon appartiene ai First di epsilon, passo al caso 2:

Follow(X) = { $,) } => T[ X,$ ] = epsilon, T[ X,) ] = epsilon

Considero Y → \*T

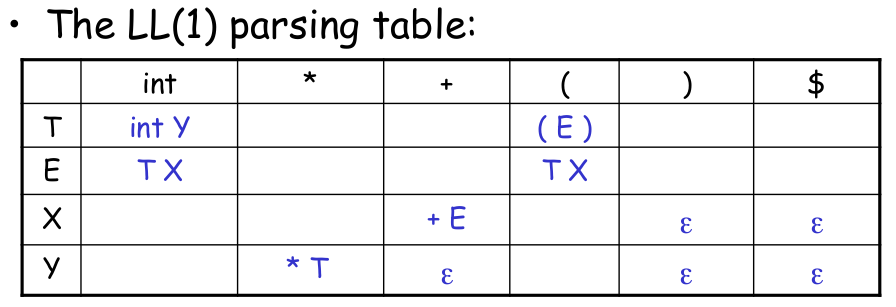
First(\*T) = {\*} => T[ Y,\* ] = \*T

Considero Y → epsilon

First(epsilon) = epsilon, quindi passo al caso 2

Follow(Y) = { +,),$ } => T[ Y,+ ] = epsilon, T[ Y,) ] = epsilon, T[ Y,$ ] = epsilon

Quindi la parsing table finale diventa:



Tutte le celle bianche sono situazioni di errore sintattico.

Se una cella ha una definizione multipla, allora la grammatica G non è LL(1)