FIRST

FIRST(X), donde ($X \in \{T \cup N\}$), o
FIRST(α), donde ($\alpha \in \{T \cup N\}^*$)

Conjunto formado por los Terminales que pueden aparecer como **primer símbolo terminal** en las cadenas derivadas a partir de *X (*o a partir de α).

Además de los **terminales (los tokens)** mencionados, puede contener el **elemento nulo** (λ)

FOLLOW

FOLLOW(A), donde *($A \in N$)*

Conjunto formado por los Terminales que pueden aparecer **inmediatamente a continuación** de *A* en alguna forma sentencial.

Además de los **terminales (los tokens)** mencionados, puede contener el **delimitador de final de cadena** ($)

FIRST

Ejemplo.

G:

S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

FIRST de un símbolo terminal

FIRST(a) = { a }
FIRST(b) = { b }
FIRST(h) = { h }
FIRST(c) = { c }
FIRST(d) = { d }

FIRST

Ejemplo.

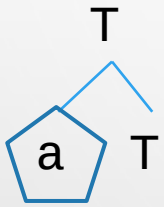
G:

S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

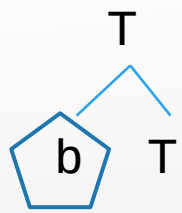
FIRST de un símbolo No terminal

FIRST(T) = { a, b, h }

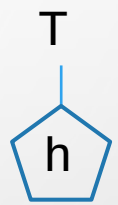
T → a T



T → b T



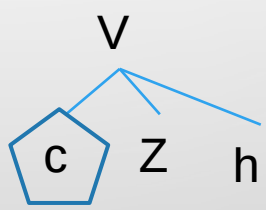
T → h



FIRST(V) = { λ, c }

V → λ

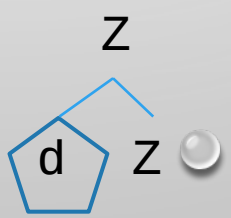
V → c Z h



FIRST(Z) = { λ, d }

Z → λ

Z → d Z



λ representa la cadena vacía

FIRST

Ejemplo.

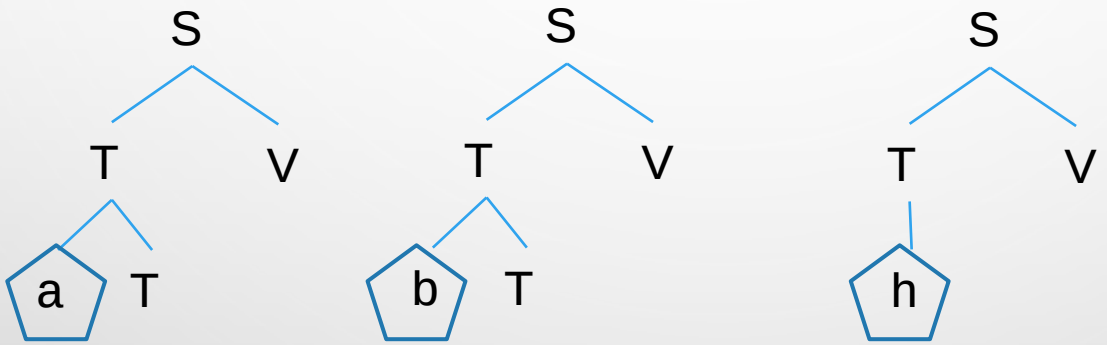
G:

S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

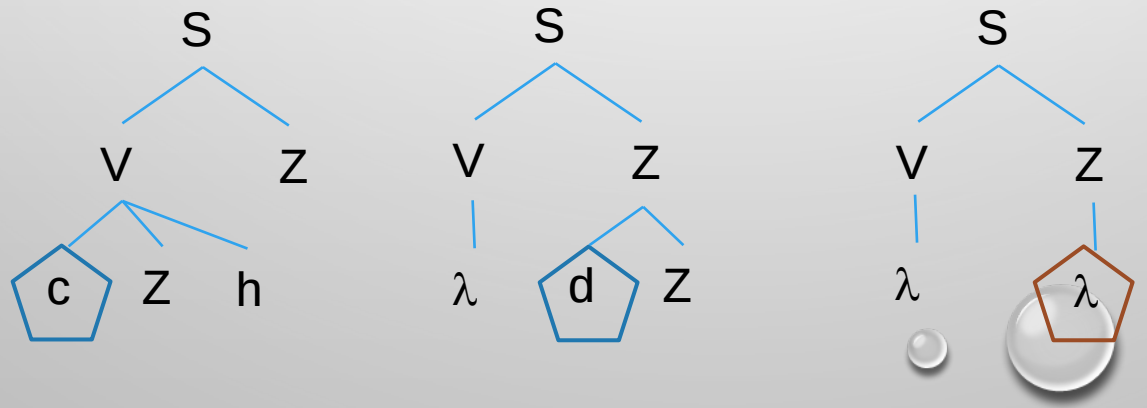
FIRST de un símbolo No terminal

FIRST(S) = { a, b, h, c, d, λ }

S → T V



S → V Z



FIRST

FIRST (X) ($X \in \{T \cup N\}$). El **conjunto de los símbolos terminales que pueden aparecer como primer símbolo terminal** en las cadenas derivadas a partir de X.

- Si X es un terminal ($X = t$, con $t \in T$) entonces $FIRST(X) = \{t\}$.
- Si X es un no terminal ($X \in N$) entonces hay que estudiar cada una de las reglas de X. Se ejecutarán los siguientes pasos hasta que no se puedan añadir más elementos al conjunto *FIRST*:
  1. Si existe la regla $X \rightarrow \lambda$, entonces añadir λ a $FIRST(X)$
  2. Para cada una de las restantes reglas $X \rightarrow Y_1 Y_2 \ldots Y_k$, (donde $Y_i \in \{T \cup N\}$), se aplica el siguiente algoritmo hasta que no se pueda añadir nada nuevo al conjunto $FIRST(X)$:

```
Calcular FIRST(Y1). Todos los elementos no nulos de FIRST(Y1) se añaden a FIRST(X)
if λ ∉ FIRST(Y1) then return FIRST(X)
else Calcular FIRST(Y2). Todos los elementos no nulos de FIRST(Y2) se añaden a FIRST(X)
    if λ ∉ FIRST(Y2) then return FIRST(X)
    else Calcular FIRST(Y3). Todos los elementos no nulos de FIRST(Y3) se añaden a FIRST(X)
        … y así sucesivamente …
            if λ ∉ FIRST(Yk-1) then return FIRST(X)
            else Calcular FIRST(Yk). Todos los elementos no nulos de FIRST(Yk) se añaden a FIRST(X)
                    if λ ∉ FIRST(Yk) then return FIRST(X)
                    else añadir λ a FIRST(X) y return FIRST(X)
```

FIRST

FIRST de un símbolo

FIRST de una cadena de símbolos α ($\alpha = Y_1 Y_2 \ldots Y_k$)

FIRST

FIRST (X) ($X \in \{T \cup N\}$). El **conjunto de los símbolos terminales que pueden aparecer como primer símbolo terminal** en las cadenas derivadas a partir de X.

FIRST(t) = { t } — FIRST de un terminal

FIRST de un no terminal

FIRST (N)

Para cada regla de N

$N \rightarrow Y_1\ Y_2\ Y_3 \ldots Y_n$ $(Y_i \in \{T \cup N\})$

FIRST (Y_1) - λ — Si Y_1 no puede derivar λ, termino.
En caso contrario se pasa a Y_2 y se hace lo mismo:

FIRST (Y_2) - λ — Si Y_2 no puede derivar λ, termino.
En caso contrario se pasa a Y_3

…

Solo si llegamos a FIRST(Y_n) y este **también** contiene λ, se añadirá λ al FIRST(N)

NOTA: La regla $N \rightarrow \lambda$, si existe, evidentemente indica que λ está en FIRST(N)

FIRST

Ejemplo de aplicación del algoritmo de cálculo de FIRST a un No terminal

G:

S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

FIRST(S) = { a, b, h, c, d, λ }

S → T V
S → $Y_1 Y_2$

FIRST(T)={a, b, h}

S → V Z
S → $Y_1 Y_2$

FIRST(V)={λ, c}

FIRST(Z)={λ, d}

Como λ está en el FIRST de todos los Y_i (V y Z), se añade a FIRST (S)

FIRST

FIRST(X), donde ($X \in \{T \cup N\}$), o
FIRST(α), donde ($\alpha \in \{T \cup N\}^*$)

Conjunto formado por los Terminales que pueden aparecer como **primer símbolo terminal** en las cadenas derivadas a partir de *X* (o a partir de α).

Además de los **terminales (los tokens)** mencionados, puede contener el **elemento nulo** (λ)

FOLLOW

FOLLOW(A), donde *($A \in N$)*

Conjunto formado por los Terminales que pueden aparecer **inmediatamente a continuación** de *A* en alguna forma sentencial.

Además de los **terminales (los tokens)** mencionados, puede contener el **delimitador de final de cadena** ($)

FOLLOW

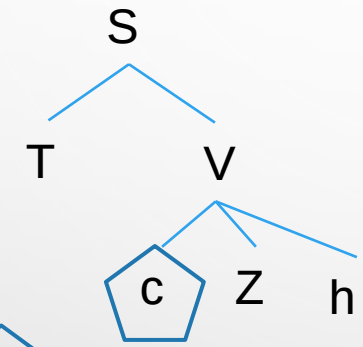
Ejemplo.

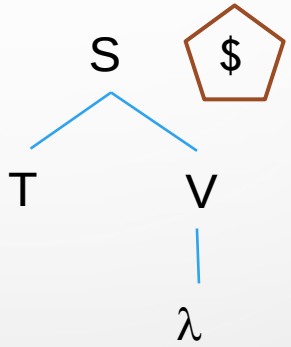
G:

S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

FOLLOW(T) = {$, c }

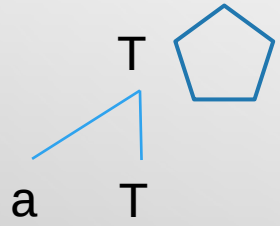
S → T V



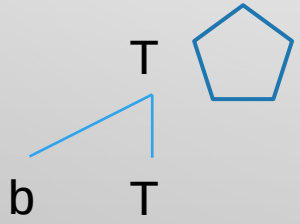


$ es el símbolo de final de cadena (o fin de fichero).
S es el axioma.

T → a T



T → b T



FOLLOW

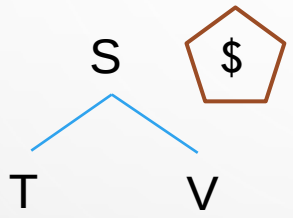
Ejemplo.

G:

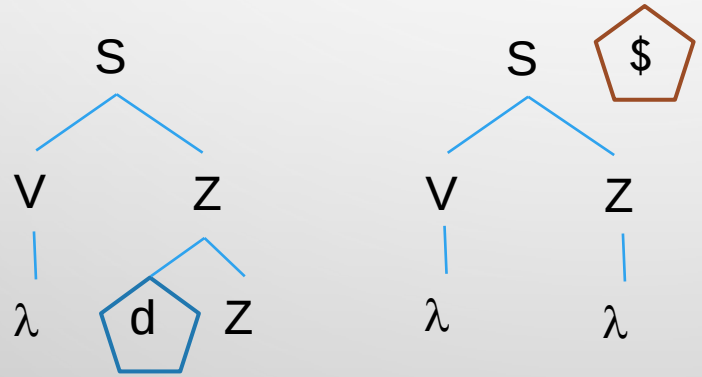
S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

FOLLOW(V) = { $, d }

S → T V



S → V Z



$ es el símbolo de final de cadena (o fin de fichero).
S es el axioma.

FOLLOW(S) = { $ }

S no aparece en ningún consecuente

FOLLOW

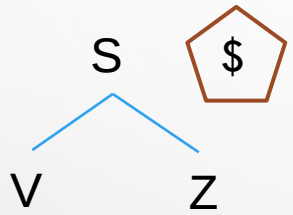
Ejemplo.

G:

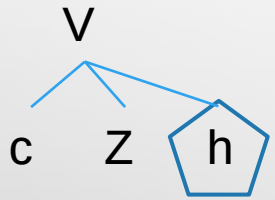
S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

FOLLOW(Z) = {$, h }

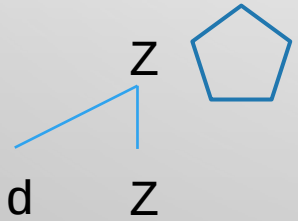
S → V Z



V → c Z h



Z → d Z



$ es el símbolo de final de cadena (o fin de fichero).
S es el axioma.

FOLLOW

FOLLOW (A), ($A \in N$). El **conjunto de los símbolos terminales que pueden aparecer inmediatamente a la derecha de *A*** en alguna forma sentencial, es decir, el conjunto de terminales *t* tales que haya una derivación de la forma $S \overset{*}{\Rightarrow} \alpha A t \beta$ (siendo $\alpha, \beta \in (N \cup T)^*$). Si *A* puede ser el símbolo de más a la derecha en alguna forma sentencial, entonces *$* está en *FOLLOW (A)*.

Para calcular *FOLLOW (A)* se aplican las siguientes reglas hasta que no se pueda añadir nada más al conjunto *FOLLOW (A)*:

1. Si *A* es el axioma de la gramática, añadir *$* a *FOLLOW (A)*.
2. Si existe una regla $B \rightarrow \alpha A \beta$, entonces todos los elementos no nulos de *FIRST (β)* se añaden a *FOLLOW (A)*.
3. Si existe una regla $B \rightarrow \alpha A \beta$ y $\lambda \in$ *FIRST (β)* (es decir, $\beta \overset{*}{\Rightarrow} \lambda$), o bien si existe una regla $B \rightarrow \alpha A$, entonces todo lo que esté en *FOLLOW (B)* se añade a *FOLLOW (A)*.

FOLLOW

FOLLOW de un símbolo No terminal

FOLLOW

FOLLOW (A), ($A \in N$). El **conjunto de los símbolos terminales que pueden aparecer inmediatamente a la derecha de *A*** en alguna forma sentencial, es decir, el conjunto de terminales *t* tales que haya una derivación de la forma $S \Rightarrow^{*} \alpha\, A\, t\, \beta$ (siendo $\alpha, \beta \in (N \cup T)^{*}$). Si *A* puede ser el símbolo de más a la derecha en alguna forma sentencial, entonces *$* está en *FOLLOW (A)*.

Para calcular *FOLLOW (A)* se aplican las siguientes reglas hasta que no se pueda añadir nada más al conjunto *FOLLOW (A)*:

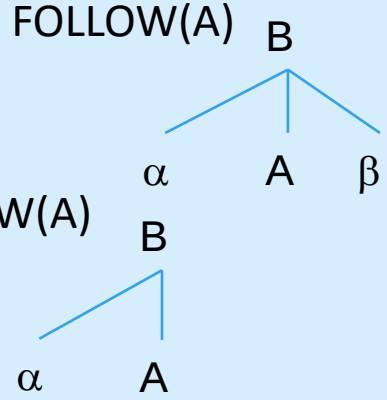
Si A es el axioma, se añade $ a FOLLOW(A)

Para cada regla B→ α **A** β

- Todos los elementos de FIRST(β) **excepto λ**, se añaden a FOLLOW(A)

Si β es λ o se puede hacer λ, entonces

- Todos los elementos de FOLLOW(B) se añaden a FOLLOW(A)



Buscar A en los consecuentes de las reglas

Nunca λ puede estar en FOLLOW (porque añadimos $ detrás de la cadena)

FOLLOW

Ejemplo de aplicación del algoritmo de cálculo de FOLLOW

G:

S → T V | V Z
T → a T | b T | h
V → λ | c Z h
Z → λ | d Z

FOLLOW(V) = { $, d }

S → T V

$B \rightarrow \alpha A$

FOLLOW(S) ⊆ FOLLOW(V)
FOLLOW(S) = { $ }

S → V Z

$B \rightarrow \alpha A \beta$

FIRST(Z) – {λ} ⊆ FOLLOW(V)
FIRST(Z) = { λ, d }

FOLLOW(S) ⊆ FOLLOW(V)

Para cada regla $B \rightarrow \alpha$ **A** β

- Todos los elementos de FIRST(β) excepto λ, se añaden a FOLLOW(A)

Si β es λ o se puede hacer λ, entonces

- Todos los elementos de FOLLOW(B) se añaden a FOLLOW(A)

B es el antecendente de la regla

CONDICIÓN LL(1)

Una gramática *G* es LL(1) sii

- Para cada no terminal *A* para el que haya más de una producción en *G*:
  - Para cada par de producciones $A \rightarrow \alpha \mid \beta$ se cumplen las siguientes dos condiciones:
    1. No existe ningún terminal *t* tal que tanto α como β deriven cadenas que empiecen por ese mismo terminal

       $$FIRST(\alpha) \cap FIRST(\beta) = \phi$$

    2. A lo sumo, o α o β pueden derivar la cadena vacía λ, pero no ambos. Supongamos que β puede derivarla, es decir, $\beta \stackrel{*}{=>} \lambda$. En este caso, α no puede derivar ninguna cadena que empiece con un terminal que pertenezca a *FOLLOW(A)*

       $$FIRST(\alpha) \cap FOLLOW(A) = \phi$$

CONDICIÓN LL(1)

Una gramática *G* es LL(1) sii

Para cada no terminal A para el que haya más de una producción ($A \rightarrow \alpha \mid \beta \mid \gamma \mid \ldots$)

/ como máximo uno de esos consecuentes puede ser λ o derivar λ */*

Para cada par de producciones $A \rightarrow \alpha \mid \beta$

1. $FIRST(\alpha) \cap FIRST(\beta) = \phi$
2. Si $\beta \stackrel{*}{\Rightarrow} \lambda$, */* supongamos que β deriva λ */*

   $FIRST(\alpha) \cap FOLLOW(A) = \phi$

Por ello, un analizador sintáctico construido sobre esa gramática LL(1):

aplicará A→ α sii sig_tok $\in$ FIRST(α)

aplicará A→ β sii sig_tok $\in$ FIRST(β) o a FOLLOW(A)

. . .

aplicará A→ γ sii sig_tok $\in$ FIRST(γ)

Ejemplo de comprobación de la condición LL(1)

G:

E → T E'
E' → + T E' | λ
T → F T'
T' → * F T' | λ
F → (E) | id

E' → + T E' | λ

FIRST(+TE') ∩ FIRST(λ) = { + } ∩ { λ } = ∅
FIRST(+TE') ∩ FOLLOW(E') = { + } ∩ {), $ } = ∅

T' → * F T' | λ

FIRST(*FT') ∩ FIRST(λ) = { * } ∩ { λ } = ∅
FIRST(*FT') ∩ FOLLOW(T') = { * } ∩ { +,), $ } = ∅

F → (E) | id

FIRST((E)) ∩ FIRST(id) = { (} ∩ { id } = ∅

Para cada par de producciones $A \rightarrow \alpha \mid \beta$ se ha de cumplir:

1. $FIRST(\alpha) \cap FIRST(\beta) = \phi$
2. Si $\beta \overset{*}{\Rightarrow} \lambda$, $FIRST(\alpha) \cap FOLLOW(A) = \phi$

Por tanto, la G es LL(1)