TEC | Tecnológico de Costa Rica

Compiladores e Intérpretes

Profesor:
Francisco Torres

Grupo 40

Elaborado por:
Luis Diego Vargas Arroyo

Apuntes Viernes 13 de mayo del 2017

INFORMACIÓN IMPORTANTE

- Se entregaron los **quices 08 y 09.**
- Se entregó la hoja de evaluación del proyecto 02(el profe fue benevolente e hizo una curva en la nota).
- El II Examen queda para el viernes de semana 16 **(viernes 03 de junio)**
- El III Parcial en que entra toda la materia del semestre es una semana después del II parcial (**10 de junio**)
- La entrega de la revisión de la tare programada 03 queda para el viernes de semana 15 (27 de mayo). Esta fecha puede variar.

Y con eso se acaba el semestre. Suerte a todos.

Contenido

WARNING: Graphic Content

The following images and/or content may be disturbing to some viewers.
Viewer discretion is strongly advised.

Queda totalmente prohibida la lectura para Samantha y para Izcar de estos apuntes.

Análisis Sintáctico

Nosotros si debemos construir la tabla de símbolos. Pero para esto debemos primero hacerle un pre proceso a la gramática para luego generar la tabla de Símbolos.

Construcción de tabla de Parsing

Hay algoritmos para construir la tabla de parsing. Cuando se le realiza el pre proceso a la gramática.

- El lenguaje asociado no cambia
- Genera las mismas hileras.
- No genera ni una hilera más ni una menos.

Recursividad por la Izquierda.

Un CFG es recursivo por la izquierda, si hay al menos contiene una regla de la forma:

A-> Aα

Debido a que esto genera lo siguiente:

A => Aα => Aαα => Aααα => Aααααα => Aαααααα

Por lo tanto, debe de haber otra regla que lo aterrice:

A->β

Las hileras generadas serian de la forma:

Bαααα……α

¿Qué problemas tiene que una gramática sea recursiva por la izquierda?

A		A-> Aα	

No puede haber reglas recursivas por la izquierda

Eliminar Recursividad por la Izquierda

Si tenemos las reglas:

- A-> Aα
- **β**

Si se cambian por:

- A-> βA'
- A'-> αA'
- A'->ε

las hileras generadas no cambian.

Nota: En quices futuros o tareas antes de hacer la tabla hay que tomar en cuenta como paso 0 eliminar la recursividad por la izquierda (Sergio se le va a olvidar el paso 0)

Ejemplo 1

E -> E +T

E -> T

Siguiendo con el algoritmo de arriba tendríamos lo siguiente:

- E= A
- α =+T
- β = T

por lo tanto, si queremos eliminar la recursividad por la izquierda quedaría de la siguiente manera:

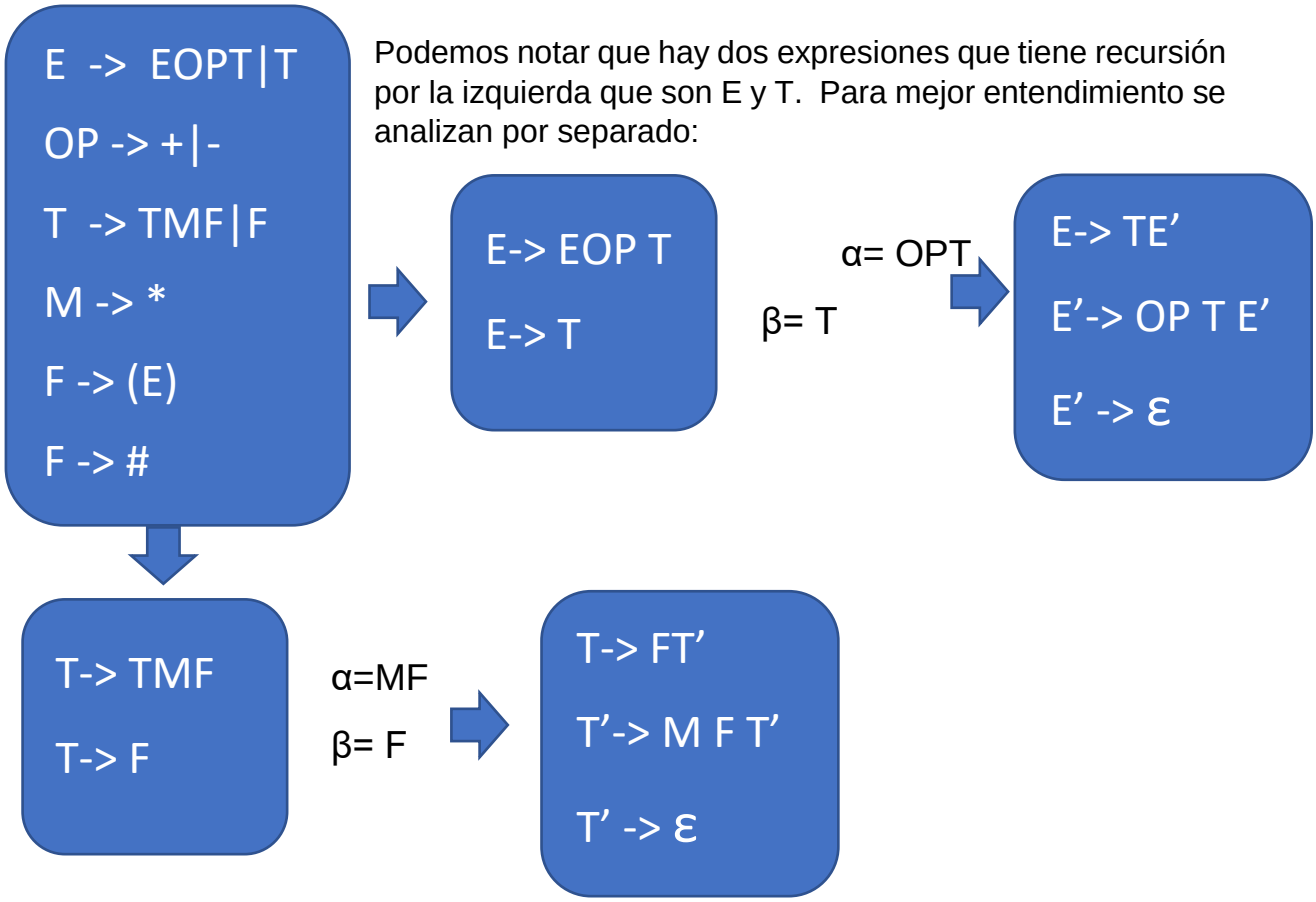
E -> TE'

E' -> +T E'

E' -> ε

Ya con esto eliminamos la recursividad por izquierda.

Ejemplo 2



Y esto da como resultado:

E-> TE'

E'-> OP T E'

E' -> ε

OP -> +|-

T-> FT'

T'-> M F T'

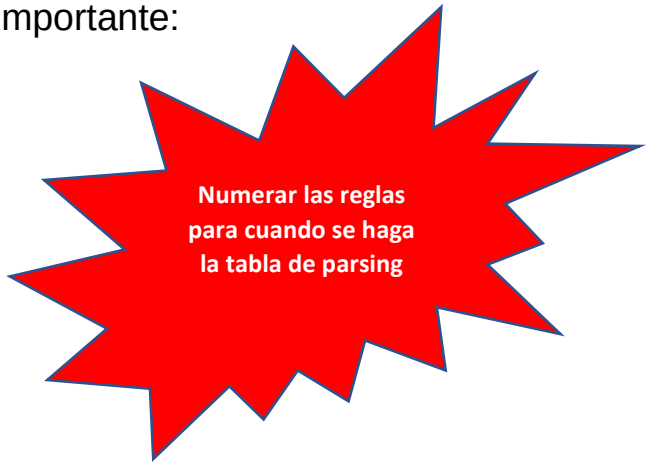
T' -> ε

M -> *

F -> (E)

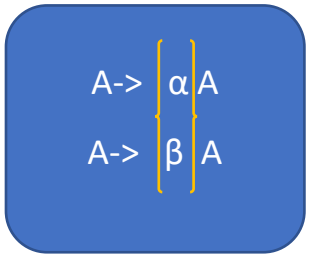
Sergio pregunto cómo hacemos esto cuando estamos con parsing. El profe le dijo que si lo hacen en el parsing es muy tarde. Esto se hace cuando se está con la gramática.

Importante:

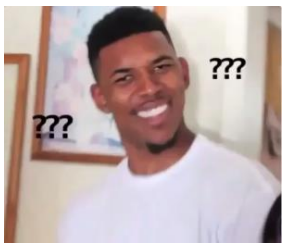


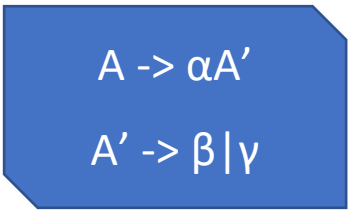
Factorización Izquierda

Si hay 2 o más reglas de la siguiente forma:



- Comparte la parte izquierda de su lado derecho.
- La hilera α es lo más larga posible.
- **No se puede ir a dos lugares.**
- Entonces lo cambiamos por:





Ejemplo 1

SQ -> S;SQ

SQ-> S

S-> Statement

α= S

β= ;SQ

γ= ε

SQ -> SSQ'

SQ'-> ;SQ| ε

S-> Statement

Ejemplo 2

IFS-> If(E)S

IFS> IF(E)S else S

α= If(E)S

β= elseS

γ= ε

IFS -> IF(E) IFS'

IFS'-> else S| ε

Calculo de First()

Debemos de estar bien concentrados con esto

Sea G una CFG

- La Función **First(x)** regresa un conjunto de terminales y posiblemente ε que indican todos los "inicios" factibles de x bajo la gramática G

El argumento x puede ser:

- Un Terminal
- Un No Terminal
- Una hilera de no Terminales y no Terminales (puede ser ε)

El cálculo de First es iterativo hasta que no haya más cambios.

Casos Triviales de First(x)

- Si X es ε se regresa el conjunto { ε }
- Si X es un terminal se regresa el conjunto que contenga este terminal.
  - FIRST (+) = {+}
  - FIRST(id) = {id}

First(x) de un no terminal

X es un no terminal

- Por cada regla de la gramatica X -> X_1 X_2 X_3 …. X_n
  - Una(FIRST(X_1)- {ε}) a FIRST(X)
  - Si FIRST(X_1) contiene a ε:
    - Una(FIRST(X_2)- {ε}) a FIRST(X)
    - Si FIRST(X_2) contiene a ε.
      - Una(FIRST(X_3)- {ε}) a FIRST(X)
      - Si FIRST(X_3) contiene a ε.
        - Una …….

Si ε pertenece a FIRST(X_1) FIRST(X_2) FIRST(X_3) …….. FIRST(X_n) entonces una { ε } pertenece a FIRST(x).

Los conjuntos según la niña Marlene no tiene repetidos. Por lo cual si se repite no se agrega

First(x) de una Hilera

X es una Hilera $X_1X_2X_3$…..X_n

- Haga:
  - Una(FIRST(X_1)- {ε}) a FIRST(X)
  - Si FIRST(X_1) contiene a ε:
    - Una(FIRST(X_2)- {ε}) a FIRST(X)
    - Si FIRST(X_2) contiene a ε.
      - Una(FIRST(X_3)- {ε}) a FIRST(X)
      - Si FIRST(X_3) contiene a ε.

Si ε pertenece a FIRST(X_1) FIRST(X_2) FIRST(X_3) …….. FIRST(X_n) entonces una {ε} pertenece a FIRST(x).

Ejemplo 1 de FIRST (x)

E -> E OP T

E -> T

OP -> +

OP -> -

T -> T M F

T -> F

M -> *

F -> (E)

F -> #

	First()		
No Terminal		Pasada 1	Pasada 2
E	∅		
OP	∅		
T	∅		
M	∅		
F	∅		

Para iniciar todo contienen a vacío. Primero analizamos a **E.**

E-> E OP T

- Hay que echar todo lo que este en el First(E) en el First(E) por lo tanto solo ∅

	First()		
No Terminal		Pasada 1	Pasada 2
E	∅	∅	
OP	∅		
T	∅		
M	∅		
F	∅		

- Luego como no hay épsilon se pasa la siguiente no terminal.

E->T

Se debe echar todo lo que este en el FIRST(T) en el FIRST(E). En el FIRST(T) solo esta ∅.

- Luego debemos echar en el FIRST(OP) lo que está en el FIRST(+)

First()			
No Terminal		Pasada 1	Pasada 2
E	ø	ø	
OP	ø	{+}	
T	ø		
M	ø		
F	ø		

- Luego debemos echar en el FIRST(OP) lo que esta en el FIRST(-)

First()			
No Terminal		Pasada 1	Pasada 2
E	ø	ø	
OP	ø	{+,-}	
T	ø		
M	ø		
F	ø		

- Luego con T

T -> T M F

Primero debemos echar en el FIRST(T) todo lo que hay en el FIRST(T) como solo hay ø queda igual.

First()			
No Terminal		Pasada 1	Pasada 2
E	ø	ø	
OP	ø	{+,-}	
T	ø	ø	
M	ø		
F	ø		

- Luego como no hay épsilon se pasa a la siguiente regla:

T -> F

Debemos echar en el **FIRST(T)** todo lo que hay en el **FIRST(F)** como solo hay ø queda igual.

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	
OP	∅	{+,-}	
T	∅	∅	
M	∅		
F	∅		

- Luego seguimos con M

  M -> *

Debemos echar en el **FIRST(M)** todo lo que está en el **FIRST (*)**

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	
OP	∅	{+,-}	
T	∅	∅	
M	∅	{*}	
F	∅		

Pasamos a la siguiente regla.

- **F -> (E)**
  Ahora debemos echar en el **FIRST(F)** todo lo que este en el **FIRST (()**

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	
OP	∅	{+,-}	
T	∅	∅	
M	∅	{*}	
F	∅	{ (}	

¿Por qué solo metemos el (? Debido a que no hay épsilon y no se debe seguir.

- Seguimos con la siguiente regla.

  F-> #

Debemos echar en el **FIRST(F)** todo lo que este en el **FIRST(#)**

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	
OP	∅	{+,-}	
T	∅	∅	
M	∅	{*}	
F	∅	{ (,# }	

Como hubo varios cambios se debe de realizar otra pasada. **Se deben realizar pasadas hasta cuando ya no haya cambios.**

- Empezamos de nuevo con

E -> E OP T

Debemos echar lo que este en el **First(E)** en el **FIRST(E).** Como está vacío queda igual.

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	∅
OP	∅	{+,-}	
T	∅	∅	
M	∅	{*}	
F	∅	{ (,# }	

Como no hay épsilon pasamos a la siguiente regla.

- La siguiente regla es:

E->T

Entonces debemos echar en el **FIRST(E)** lo que hay en el **FIRST(T).** Como no hay nada queda igual. Queda igual a la tabla anterior.

- Luego con OP debemos echar en el **FIRST (OP)** lo que hay en el **FIRST (+).** Como ya está agregado al conjunto, no se agrega de nuevo.

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	∅
OP	∅	{+,-}	{+,-}
T	∅	∅	
M	∅	{*}	
F	∅	{ (, # }	

Igual sucede con OP -> - ya el FIRST (-) se encuentra en el FIRST(OP)

- Luego Pasamos a:

T -> T M F

Debemos echar en el FIRST(T) lo que hay en el FIRST(T) como solo esta ∅ queda igual.

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	∅
OP	∅	{+,-}	{+,-}
T	∅	∅	∅
M	∅	{*}	
F	∅	{ (, # }	

- Pasamos a la siguiente regla:

T->F

Debemos Agregar en el FIRS(T) todo lo que hay en el FIRST(F). El FIRST(F) contiene {(, #} por lo cual se agregan al FIRS(T)

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	∅
OP	∅	{+,-}	{+,-}
T	∅	∅	{ (, #}
M	∅	{*}	
F	∅	{ (, # }	

Como no hay épsilon continuamos a la siguiente regla.

- La siguiente regla es:

M-> *

Debemos echar en el FIRST(M) lo que hay en el FIRST(*) pero como ya está agregado entonces queda igual.

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	∅
OP	∅	{+,-}	{+,-}
T	∅	∅	{ (, #}
M	∅	{*}	{*}
F	∅	{ (, # }	

Luego como no hay épsilon pasamos a la siguiente regla.

- Este **FIRST(F)** queda igual al anterior debido a que se deben agregar el
  FIRST (() y el **FIRST (#)** pero estos ya se encuentran en el conjunto

First()			
No Terminal		Pasada 1	Pasada 2
E	∅	∅	∅
OP	∅	{+,-}	{+,-}
T	∅	∅	{ (, #}
M	∅	{*}	{*}
F	∅	{ (, # }	{ (, # }

Como hubo algunos cambios en algunos FIRST() se debe realizar otra pasada desde el inicio

First()				
No Terminal		Pasada 1	Pasada 2	Pasada 3
E	∅	∅	∅	
OP	∅	{+,-}	{+,-}	
T	∅	∅	{ (, #}	
M	∅	{*}	{*}	
F	∅	{ (, # }	{ (, # }	

- Se inicia con **E -> EOPT** se debe poner en el FIRST(E) lo que está en el FIRST(E) como solo esta ∅, no se ingresa.

First()				
No Terminal		Pasada 1	Pasada 2	Pasada 3
E	∅	∅	∅	∅
OP	∅	{+,-}	{+,-}	
T	∅	∅	{ (, #}	
M	∅	{*}	{*}	
F	∅	{ (, # }	{ (, # }	

- Luego se pasa a la siguiente regla **E-> T** se debe ingresar al FIRST(E) lo que está en el FIRST(T)

First()				
No Terminal		Pasada 1	Pasada 2	Pasada 3
E	∅	∅	∅	{(, #}
OP	∅	{+,-}	{+,-}	
T	∅	∅	{ (, #}	
M	∅	{*}	{*}	
F	∅	{ (, # }	{ (, # }	

- Seguimos con la siguiente regla.

OP->+

Como esta regla ya se encuentra en el FIRST(OP). No varia nada. Pasamos a la siguiente regla. OP -> -, como esta regla también ya se encuentra en el FIRST(OP), no se modifica nada.

First()				
No Terminal		Pasada 1	Pasada 2	Pasada 3
E	∅	∅	∅	{(, #}
OP	∅	{+,-}	{+,-}	{+,-}
T	∅	∅	{ (, #}	
M	∅	{*}	{*}	
F	∅	{ (, # }	{ (, # }	

- Pasamos a la siguiente regla **T -> TMF** hay que echar en el FIRS(T) lo que hay en el FIRST(T), no se modifica el conjunto de T

First()				
No Terminal		Pasada 1	Pasada 2	Pasada 3
E	∅	∅	∅	{(, #}
OP	∅	{+,-}	{+,-}	{+,-}
T	∅	∅	{ (, #}	{ (, #}
M	∅	{*}	{*}	
F	∅	{ (, # }	{ (, # }	

Luego Pasamos a la regla **T-> F,** esta tampoco modifica el conjunto del FIRS(T) por lo cual queda igual.

- Pasamos a la siguiente regla M -> {*} el FIRST(*) ya está en el conjunto del FIRST(M)

First()				
No Terminal		Pasada 1	Pasada 2	Pasada 3
E	∅	∅	∅	{(, #}
OP	∅	{+,-}	{+,-}	{+,-}
T	∅	∅	{ (, #}	{ (, #}
M	∅	{*}	{*}	{*}
F	∅	{ (, # }	{ (, # }	

- Pasamos a la siguiente regla **F -> (E),** debemos ingresar en el FIRST(F) lo que hay en el FIRST ((), pero los elementos ya se encuentran en el conjunto del FIRST(F).
- La siguiente regla es **F -> #**, este conjunto también ya se encuentra en el FIRST(F).

First()				
No Terminal		Pasada 1	Pasada 2	Pasada 3
E	∅	∅	∅	{(, #}
OP	∅	{+,-}	{+,-}	{+,-}
T	∅	∅	{ (, #}	{ (, #}
M	∅	{*}	{*}	{*}
F	∅	{ (, # }	{ (, # }	{ (, # }

Ya está terminado nuestro primer Calculo FIRST ()

IMPORTANTE: Se debe corroborar que no se genere ningún otro cambio en la tabla. Si hubo algún cambio se debe realizar otra pasada.

Ejemplo 2 de FIRST(x)

El profe nos hizo hacer grupos para verlo a él resolverlo en la pizarra. Creo que así entendíamos mejor.

E-> TE'

E'-> OPTE'

E'-> ε

OP -> +

OP -> -

T -> F T'

T' -> MFT'

T' -> ε

M-> *

F -> (E)

F -> #

Este ejemplo siguiente los pasos del FIRST () quedaría de 3

Cada una seria:

FIRST()			
No Terminal	Pasada 1	Pasada 2	Pasada 3
E	∅	∅	{(, #}
E'	{ **ε** }	{+,- ,**ε** }	{+, - **ε** }
OP	{+,-}	{+ , -}	{+, -}
T	∅	{(, #}	{ (, #}
T'	{ **ε** }	{*, **ε** }	{* , **ε** }
M	{*}	{*}	{*}
F	{(,#}	{(, #}	{ (, #}

Ejemplo 3 de FIRST(X)

S-> aSc

S->B

B-> bBc

B-> c

c-> cCc

c-> d

FIRST()			
No Terminal	Pasada 1	Pasada 2	Pasada 3
S	{a}	{a, b}	{a, b, c, d}
B	{b}	{b, c, d }	{b, c, d}
C	{c, d}	{c, d}	{c, d}

Ejemplo 4 de FIRST(X)

S-> ABc

A->a

B-> b

A-> ε

c-> ε

FIRST()			
No Terminal	Pasada 1		
S	∅		
A			
B			

- Se debe echar en el FIRST(S) lo que hay en el FIRST(A) por lo tanto es solo vacío.

FIRST()		
No Terminal	Pasada 1	Pasada 2
S	∅	
A	{a}	
B		

- Luego se Debe echar en el FIRST(A) lo que hay en el FIRST(a).

FIRST()		
No Terminal	Pasada 1	
S	∅	
A	{a}	
B	{b}	

- Luego se debe echar en el FIRST(B) lo que contenga el FIRST(b).

FIRST()		
No Terminal	Pasada 1	
S	∅	
A	{a, ε }	
B	{b}	

- Se pasa a la siguiente regla A -> ε, Se debe agregar en el FIRST(A) lo que hay en el FIRST {ε}.
- Pasa lo mismo con la regla siguiente B-> ε, Se debe agregar en el FIRST(B) lo que hay en el FIRST(ε)

FIRST()		
No Terminal	Pasada 1	
S	∅	
A	{a, ε }	
B	{b, ε }	

Se termina la primera pasada. Se debe realizar otra pasada desde el Inicio.

- Se Inicia con la Regla **S-> A B c**

FIRST()		
No Terminal	Pasada 1	Pasada 2
S	∅	
A	{a, ε }	
B	{b, ε }	

Se debe agregar lo que este el FIRST(A) en el FIRST(S)

FIRST()		
No Terminal	Pasada 1	Pasada 2
S	∅	{a, ε }
A	{a, ε }	
B	{b, ε }	

Este debe continuar debido a que contiene a ε, **S-> A B c** por lo tanto hay que echar en el FIRST(S) lo que contenga el FIRST{B}:

FIRST()		
No Terminal	Pasada 1	Pasada 2
S	∅	{a, b, ε }
A	{a, ε }	
B	{b, ε }	

Este contiene a ε por lo cual hay que seguir con la regla, por lo tanto **S-> A B c**

Hay que echar en el FIRST(S) lo que contenga el FIRST(c):

FIRST()		
No Terminal	Pasada 1	Pasada 2
S	∅	{a, b, c}
A	{a, ε }	
B	{b, ε }	

Y como no hay más épsilon se puede pasar a la siguiente Regla. Al final la tabla debe estar así:

FIRST()		
No Terminal	Pasada 1	Pasada 2
S	∅	{a, b, c}
A	{a, ε }	{a, ε }
B	{b, ε }	{b, ε }