APUNTES DÍA 12 DE MAYO

COMPILADORES E INTÉRPRETES

DANIEL CORDERO
2014089399

Recordatorios y notas importantes

- Se devolvieron los quices 8 y 9
- Se entregó la nota de proyecto 2
- El parcial 2 será el viernes de semana 16
- El parcial largo será la semana siguiente al parcial 2
- Se habló de comer un queque al final del curso, el profe dijo que traería donas

Análisis sintáctico

Pre proceso gramática

Estos cambios en gramática no alteran el lenguaje

Construcción de la tabla de parsing

Ciertos algoritmos pueden facilitar la creación de esta tabla, pero se requiere cierto pre proceso de la gramática, antes de la creación de la tabla.

Una vez se aplica el pre proceso el lenguaje asociado no cambia, este genera las mismas hileras.

Recursividad a la izquierda

Una CFG es recursiva por la izquierda si se da al menos una regla del siguiente tipo

$$A \rightarrow A\alpha$$

$$A \Rightarrow A\alpha \Rightarrow A\alpha\alpha \Rightarrow A\alpha\alpha\alpha$$

Para corregir este problema debe haber una regla diferente que la aterrice.

$$A \rightarrow \beta$$

Problemas de la recursividad a la izquierda

Al aplicar el algoritmo predictivo se puede dar el siguiente problema

A		$A \rightarrow A\alpha$	

Para corregir este problema se debe crear una regla para que aterrice como se dijo anteriormente, por ejemplo, teniendo:

$$A \rightarrow A\alpha$$
$$A \rightarrow \beta$$

Se debe crear la regla mencionada, la cual llamaremos A’:

$$A \rightarrow \beta A$$
$$A' \rightarrow \alpha A'$$
$$A' \rightarrow \varepsilon$$

Ejemplo 1

Paso 1

$$E \rightarrow E+T$$
$$E \rightarrow T$$

Este ejemplo se puede ver más complejo que el anterior, pero la manera de resolverlo es la misma, primero se debe encontrar al alfa y beta de este problema.

Pasó 2

$$E \rightarrow E\overbrace{+T}^{\alpha}$$
$$E \rightarrow \underbrace{T}_{\beta}$$

Luego de encontrar a alfa y beta se deben seguir los mismos pasos del ejemplo anterior.

Pasó 3

$$E \to TE'$$
$$E' \to +TE'$$
$$E' \to \varepsilon$$

Para resolver todos los problemas de este tipo simplemente se debe identificar alfa y beta y resolverlo siguiendo el mismo algoritmo.

Ejemplo 2

$$E \to E\ OP\ T \mid T$$
$$OP \to + \mid -$$
$$T \to T\ M\ F \mid F$$
$$M \to *$$
$$F \to (E)$$
$$F \to \#$$

Sin importar la gramática se deben seguir los mismos pasos, primero se identifica donde existe la recursión a la izquierda, en este caso se da en la regla 1 y 3.

$$E \to E\ \overbrace{OP\ T}^{\alpha} \mid \overbrace{T}^{\beta}$$
$$OP \to + \mid -$$
$$T \to T\ \overbrace{M\ F}^{\alpha} \mid \overbrace{F}^{\beta}$$
$$M \to *$$
$$F \to (E)$$
$$F \to \#$$

Luego de esto se debe encontrar alfa y beta para cada regla y aplicar el algoritmo visto en el ejemplo inicial.

$$
\begin{aligned}
E &\rightarrow TE' \\
E' &\rightarrow OP\ T\ E' \\
E' &\rightarrow \varepsilon \\
OP &\rightarrow + \\
OP &\rightarrow - \\
T &\rightarrow FT' \\
T' &\rightarrow MFT' \\
T' &\rightarrow \varepsilon \\
M &\rightarrow * \\
F &\rightarrow (E) \\
F &\rightarrow \#
\end{aligned}
$$

Este es el resultado final luego de crear las nuevas reglas

Factorización a la izquierda

Otro problema que se debe corregir es la factorización a la izquierda, la cual se da si hay dos o más reglas de la forma:

$$
\begin{aligned}
A &\rightarrow \boxed{\alpha}\,\beta \\
A &\rightarrow \boxed{\alpha}\,y
\end{aligned}
$$

El problema se da debido a que comparten el lado izquierdo de su lado derecho, la hilera α es lo más grande posible. Para solucionar el problema se debe cambiar por:

$$
\begin{aligned}
A &\rightarrow \alpha\ A' \\
A' &\rightarrow \beta \mid y
\end{aligned}
$$

En la tabla de parsing se puede dar el siguiente error.

		α	
α		A/B	

Ejemplo 1

$$SQ \rightarrow S;\ SQ$$
$$SQ \rightarrow S$$
$$S \rightarrow \text{<statement>}$$

En este caso el problema está en el no terminal S, el cual está al inicio en dos reglas, primero se pasa a identificar todos los símbolos, mostrados en el ejemplo base.

$$SQ \rightarrow \overbrace{S;}^{\alpha}\ \overbrace{SQ}^{\beta}$$
$$SQ \rightarrow \overbrace{S}^{\alpha}\ \overbrace{\varepsilon}^{\gamma}$$
$$S \rightarrow \text{<statement>}$$

Ahora que ya se identificó cada parte del problema igualmente que con el problema de recursividad a la izquierda, simplemente se sigue el algoritmo del ejemplo base.

$$SQ \rightarrow S;\ SQ'$$
$$SQ' \rightarrow ;SQ \mid \varepsilon$$
$$S \rightarrow \text{<statement>}$$

Una vez aplicado el algoritmo se da este resultado.

Ejemplo 2

$$\begin{array}{l} IFS \rightarrow \text{if (E) S} \\ IFS \rightarrow \text{if (E) S else S} \end{array} \longrightarrow \begin{array}{l} IFS \rightarrow \text{if (E) S } \varepsilon \\ IFS \rightarrow \text{if (E) S else S} \end{array} \longrightarrow \begin{array}{l} IFS \rightarrow \text{if (E) S } IFS' \\ IFS' \rightarrow \text{else S} \mid \varepsilon \end{array}$$

Calculo de first()

Sea G una CFG, la función FIRST(X) regresa un conjunto de terminales, y posiblemente ε que indican todos los "inicios" posibles de X bajo la gramática G.

X puede ser:

- Un terminal.
- Un no terminal.
- Una hilera de terminales y no terminales, puede ser ε.

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

Casos triviales de FIRST(X)

- En caso de que x sea ε se regresa el conjunto que contenga ε = { ε}

- Si X es un terminal se regresa el conjunto que lo contenga, = {X}

Estos casos son un queque.

Caso FIRST(X) de un no terminal

X es un no terminal.

Por cada regla de X -> $X_1, X_2, X_3, \ldots, X_n$ se deber hacer:

1) Se obtiene (FIRST(X_1)-{ ε}) a FIRST(X)
2) Si FIRST(X_1) contiene ε

Si FIRST(X_1) contiene ε se toma el siguiente X, y se vuelven a aplicar los pasos anteriores, hasta que algún FIRST(X_n) no contenga ε o no existan más X para evaluar.

Ejemplo

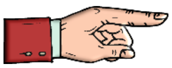
Paso 1	Paso 2	Paso 3
$X \rightarrow X_1\ X_2\ X_3$	$X \rightarrow X_1\ X_2\ X_3$	$X \rightarrow X_1\ X_2\ X_3$
$First(X_1) = \{a,\varepsilon\}$	$First(X_1) = \{a,\varepsilon\}$	$First(X_2) = \{a,b\}$
$First(X_2) = \{a,b\}$	$First(X) = \{a,\varepsilon$	$First(X) = \{a,b,\varepsilon\}$
$First(X_3) = \{c\}$		

- En el paso 1 se muestran los conjuntos resultantes de aplicar FIRST a cada X
- En el paso 2 se inicia con X_1 cuyo conjunto es {a,ε}, como este conjunto contiene ε se une con el conjunto de salida, "se echa en el conjunto X", y se continua con X_2
- En el paso 3 se obtiene el first(X_2) cuyo conjunto es {a,b}, dado que este conjunto no contiene ε solo se une al conjunto de salida, dando como el resultado {a,b,ε}, donde el conjunto anterior representa FIRST(X).
- Nota, en caso de que X_2 y X_3 contengan ε se llega hasta X_3 y el conjunto de salida seria {a,b,c,ε}

Caso FIRST(X) de una hilera

En caso de que sea una hilera se resuelve de la misma manera, la única diferencia es que se debe tener en cuenta que se debe seguir buscando dentro de esta

Ejemplo 1

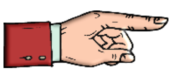


E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{}		
T	{}		
M	{}		
F	{}		

En la primera todos empiezan con conjuntos vacíos.

Analizamos la primer regla, En este caso se busca el FIRST(E) cuyo resultado es {}, se continua



E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{}		
T	{}		
M	{}		
F	{}		

Analizamos la segunda regla, en este caso se busca el FIRST(T), dado que T no contiene ningún elemento el resultado es {}

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{+}		
T	{}		
M	{}		
F	{}		

Con la tercera regla ya hay un cambio, ubicados en OP, se busca el FIRTS(+), el cual es {+}, este conjunto se une al conjunto de OP, dando como resultado FIRST(OP) = {+}

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{+,-}		
T	{}		
M	{}		
F	{}		

En la tercera regla continuamos buscando el FIRST (OP) en este caso se debe obtener el FIRST (-) = {-}, este conjunto se une al conjunto de FIRST (OP), dando como resultado {+,-}

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{+,-}		
T	{}		
M	{}		
F	{}		

Ahora se busca el FIRST de (T), el cual es { } por el momento.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{+,-}		
T	{}		
M	{}		
F	{}		

Ahora se busca el FIRST de (T), para esto se obtiene FIRST (F) = {} y se une a FIRST (T) = {}

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{+,-}		
T	{}		
M	{*}		
F	{}		

Ahora se busca el FIRST de (M), para esto se obtiene FIRST (*) = {} y se une a FIRST (M) = {}, dando como resultado {*}

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{+,-}		
T	{}		
M	{*}		
F	{ (}		

Ahora se busca el FIRST de (F), para esto se obtiene FIRST (( ) = { (} y se une a FIRST (F) = {}, dando como resultado { (}

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}		
OP	{+,-}		
T	{}		
M	{*}		
F	{ (,# }		

Ahora se busca el FIRST de (F), para esto se obtiene FIRST (#) = { # } y se une a FIRST (F) = { (}, dando como resultado { (,# }

Pasada 2

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}		
T	{}		
M	{*}		
F	{ (,# }		

Ahora se busca el FIRST de (E), para esto se obtiene FIRST (E) = { } y se une a FIRST (E) = { }, dando como resultado { }, No hay cambios.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}		
T	{}		
M	{*}		
F	{ (,# }		

Ahora se busca el FIRST de (E), para esto se obtiene FIRST (T) = { } y se une a FIRST (E) = { }, dando como resultado { }, aquí tampoco hay cambios.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}		
T	{}		
M	{*}		
F	{ (,# }		

Ahora se busca el FIRST de (OP), para esto se obtiene FIRST (+) = {+} y se une a FIRST (OP) = {+,-}, dando como resultado {+,-}, aquí ya se habían encontrado todos los elementos, no hay cambios.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}		
T	{}		
M	{*}		
F	{ (,# }		

Aquí pasa lo mismo que en caso anterior, por lo cual no hay cambios.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}	{+,-}	
T	{}	{}	
M	{*}		
F	{ (,# }		

Ahora se busca el FIRST de (T), para esto se obtiene FIRST (T) = { } y se une a FIRST (T) = { }, dando como resultado { }, no hay cambios.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}	{+,-}	
T	{ }	{ (, # }	
M	{*}		
F	{ (,# }		

Ahora se busca el FIRST de (T), para esto se obtiene FIRST (T) = { } y se une a FIRST (F) = { (,# }, dando como resultado { (,# }.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}	{+,-}	
T	{ }	{ (, # }	
M	{*}	{*}	
F	{ (,# }		

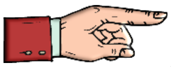
Ahora se busca el FIRST de (M), para esto se obtiene FIRST (*) = { } y se une a FIRST (M) = { * }, dando como resultado { * }.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}	{+,-}	
T	{ }	{ (, # }	
M	{*}	{*}	
F	{ (,# }	{ (,# }	

Ahora se busca el FIRST de (F), para esto se obtiene FIRST (( ) = { (} y se une a FIRST (F) = { (, # }, dando como resultado { (,# }.

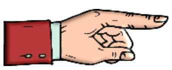
Pasada 3



E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	
OP	{+,-}	{+,-}	
T	{ }	{ (, # }	
M	{*}	{*}	
F	{ (,# }	{ (,# }	

Ahora se busca el FIRST de (E), para esto se obtiene FIRST (E) = { } y se une a FIRST (E) = { } dando como resultado { }. Sigue vacío.



E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{}
OP	{+,-}	{+,-}	
T	{ }	{ (, # }	
M	{*}	{*}	
F	{ (,# }	{ (,# }	

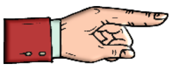
Ahora se busca el FIRST de (E), para esto se obtiene FIRST (E) = { } y se une a FIRST (E) = { } dando como resultado { }. Sigue vacío.

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

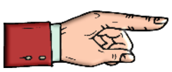
	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
OP	{+,-}	{+,-}	
T	{ }	{ (, # }	
M	{*}	{*}	
F	{ (,# }	{ (,# }	

Continuamos con el FIRST de (E), para esto se obtiene FIRST (T) = { (,# } y se une a FIRST (E) = { } dando como resultado { (,# }.

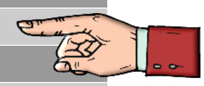
En OP ocurre lo mismo que en pasadas anteriores, por lo cual solo se mostrara el proceso no la explicación

E → E OP T
E → T

OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
OP	{+,-}	{+,-}	{+,-}
T	{ }	{ (, # }	
M	{*}	{*}	
F	{ (,# }	{ (,# }	

E → E OP T
E → T
OP → +

OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

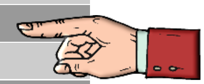
	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
OP	{+,-}	{+,-}	{+,-}
T	{ }	{ (, # }	
M	{*}	{*}	
F	{ (,# }	{ (,# }	



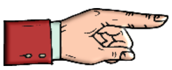
E → E OP T
E → T
OP → +
OP → -

T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
OP	{+,-}	{+,-}	{+,-}
T	{ }	{ (, # }	{ (, # }
M	{*}	{*}	
F	{ (,# }	{ (,# }	



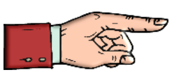
T no sufre cambios, se mantiene igual.



E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

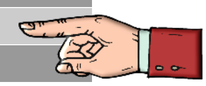
	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
OP	{+,-}	{+,-}	{+,-}
T	{ }	{ (, # }	{ (, # }
M	{*}	{*}	
F	{ (,# }	{ (,# }	

T tampoco sufre cambios aquí



E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
OP	{+,-}	{+,-}	{+,-}
T	{ }	{ (, # }	{ (, # }
M	{*}	{*}	{*}
F	{ (,# }	{ (,# }	



M se mantiene igual

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
OP	{+,-}	{+,-}	{+,-}
T	{ }	{ (, # }	{ (, # }
M	{*}	{*}	{*}
F	{ (,# }	{ (,# }	{ (,# }

F se mantiene igual

E → E OP T
E → T
OP → +
OP → -
T → T M F | F
T → F
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
OP	{+,-}	{+,-}	{+,-}
T	{ }	{ (, # }	{ (, # }
M	{*}	{*}	{*}
F	{ (,# }	{ (,# }	{ (,# }

F se mantiene igual

Con esto se terminó la pasada 3, se debe hacer una pasada extra de comprobación, para asegurarse de que no existan más cambios, pero lo único que hará será pasar por todas las reglas escribiendo lo mismo de la pasada 3 en lo que sería la pasada 4.

Además se debe evitar cometer el error de que al unir dos conjuntos, agregar por error los elementos dos veces.

Por ejemplo

A = {a,b}　　　　A U B != {a,b,b}

B = {b}　　　　A U B == {a,b }

Ejemplo 2

E → TE'
E' → OP T E'
E' → ε
OP → +
OP → -
T → F T'
T' → M F T'
T' → ε
M → *
F → (E)
F → #

	Pasada 1	Pasada 2	Pasada 3
E	{}	{}	{ (,# }
E'	{ε}	{+,-,ε}	{+,-,ε}
OP	{+,-}	{+,-}	{+,-}
T	{}	{ (,# }	{ (,# }
T'	{ε}	{*,ε}	{*,ε}
M	{*}	{*}	{*}
F	{ (,# }	{ (,# }	{ (,# }

Este caso es similar al anterior, pero en este caso se utiliza ε.

Ejemplo 3

S → aSe
S → B
B → bBe
B → C
C → cCe
C → d

	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

S → aBc
A → a
B → b
A → ε
B → ε

	Pasada 1	Pasada 2
S	{a}	{a,b,c}
A	{b}	{a,ε}
B	{c,d}	{b,ε}