Universidad Simón Bolívar
Jormar Martín Arellano Gonzalez 0537840
Manuel Gómez Chacon 0535456

Proyecto III

Sea G_{1i} la gramatica recursiva-izquierda ({S}, {a}, {S → Sa, S → λ }, S) y sea G_{1d} la gramatica recursiva-derecha ({S}, {a}, {S → aS, S → λ }, S). Ambas generan el lenguaje denotado por la expresion regular a.*

a) Muestre que ambas gramaticas son LR (1) y construya sus analizadores sintácticos segun el metodo de tabla SLR (1).

G_{1i} (gramatica recursiva-izquierda):

Calculamos: FOLLOW(S) = {$, a}, necesarios para construir la tabla de acciones.
Aumentamos la gramática con un nuevo símbolo S', y procedemos a enumerar las producciones, tal y como indica el algoritmo.

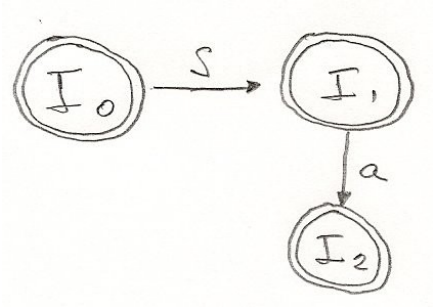
(0) S' → S $
(1) S → Sa
(2) S → λ

Construimos el autómata de prefijos viables con los conjuntos de ítems LR(0) de la gramatica:

I_0: S' → ·S$
S → ·Sa
S → ·$

I_1: S' → S·$
S → S·a

I_2: S → Sa·



Ahora podemos construir la tabla de parsing SLR(1):

	A	$	S
0	*Reduce2*	*Reduce2*	*1*
1	*Shift2*	*Accept*	
2	*Reduce1*	*Reduce1*	

G_{1d} (gramatica recursiva-derecha):
Calculamos: FOLLOW(S) = {$}, necesarios para construir la tabla de acciones.
Aumentamos la gramática con un nuevo símbolo S', y procedemos a enumerar las producciones, tal y como indica el algoritmo.

(3) S' → S $
(4) S → aS
(5) S → λ

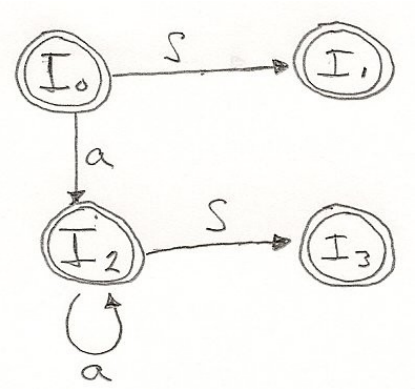
Construimos el autómata de prefijos viables con los conjuntos de ítems LR(0) de la gramatica:

I_0: S' → ·S$
S → ·aS
S → ·$

I_1: S' → S·$

I_2: S → a·S
S → ·aS
S → ·$

I_3: S → aS·



Ahora podemos construir la tabla de parsing SLR(1):

	A	$	S
0	*Shift2*	*Reduce2*	*1*
1		*Accept*	
2	*Shift2*	*Reduce2*	*3*
3		*Reduce1*	

Como fue posible aplicar satisfactoriamente el algoritmo SLR(1) en ambas gramáticas, entonces ambas gramáticas son LR(1).

*b) Compare la eficiencia de ambos analizadores en términos de espacio , i.e. Los tamaños de sus tablas y la cantidad de pila utilizada para reconocer cada frase de a , y de tiempo , i.e. la cantidad de movimientos realizados por el autómata de pila para reconocer cada frase de a**

Primero, hay que notar que en la tabla SLR(1) de recursión derecha, en la columna donde la *'a'* SOLO hay instrucciones shift, por lo que, cada vez que venga una 'a', hay que empilar un nuevo estado al autómata. Por consiguiente, la cantidad de estados que se necesita almacenar es proporcional a la longitud de la palabra; si la longitud de la palabra es n, entonces, la cantidad de memoria usada por la pila es de orden **O(n)**.

Eso no pasa en la recursión izquierda, en donde la pila se vacia (es decir, se realizan *reduce*) cada vez que se puede, en lugar de seguir layendo una 'a'. Para esta recursión, la cantidad de memoria usada es de orden O(1)

Sea G2 la gramatica ({Instr}, {i, ;}, {Instr → i, Instr → Instr ; Instr }, Instr).
a) Muestre que G2 no es una gramática LR (1), intentando construir un analizador para la gramatica y consiguiendo que tiene un conflicto. Especifique el tipo de conflicto encontrado identificándolo en la tabla.

Calculamos: FOLLOW(I) = {; , $}, necesarios para construir la tabla de acciones.
Aumentamos la gramática con un nuevo símbolo S', y procedemos a enumerar las producciones, tal y como indica el algoritmo.

(0) S' → I $
(1) I → i
(2) I → I ; I

Construimos el autómata de prefijos viables con los conjuntos de ítems LR(0) de la gramatica:

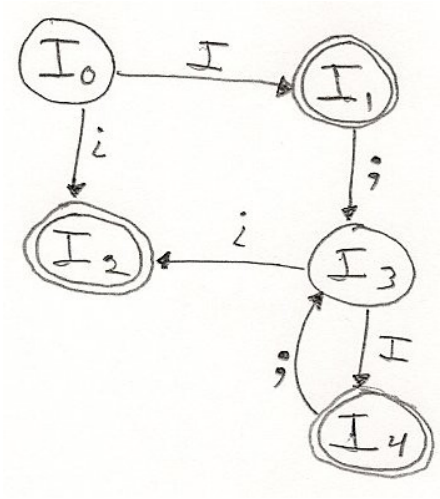
I_0: S' → ·I $
I → · i
I → · I ; I

I_1: S' → I ·
I → I · ; I

I_2: I → i ·

I_3: I → I ; · I
I → · i
I → · I ; I

I_4: I → I ; I ·
I → I · ; I



Ahora podemos construir la tabla de parsing SLR(1):

	i	;	$	I
0	*Shift2*		*Reduce2*	*1*
1		*Shift3*	*Accept*	
2		*Reduce1*	*Reduce1*	
3	*Shift2*			*4*
4		*Shift3 / Reduce2*	*Reduce2*	

Existe un conflicto shift-reduce en la ultima fila, columna del ';'. Por lo tanto, esta gramática no es LR(1).

b) Considere las dos posibilidades de resolucion del conflicto (i.e. en favor del shift o en favor del reduce si se tratase de un shift-reduce , o en favor de una y otra de las producciones si se tratase de un reduce-reduce). Muestre, para ambas alternativas de resolución del conflicto, la secuencia de reconocimiento para la frase:

i ; i ; i

dando como salida la secuencia de producciones reducidas. Si consideramos al separador de instrucciones (;) como un operador asociativo, a qué sentido de asociatividad corresponde cada una de las alternativas de resolución?

Si le damos prioridad al shift, tenemos que:

Pila	Entrada	Acción	Salida
I_0	i;i;i $	Shift 2	
I_2, I_0	;i;i $	Reduce 1	I → i
I_1, I_0	;i;i $	Shitf 3	
I_3, I_1, I_0	i;i $	Shift 2	
I_2, I_3, I_1, I_0	;i $	Reduce 1	I → i
I_4, I_3, I_1, I_0	;i $	Shift 3	
I_3, I_4, I_3, I_1, I_0	i $	Shift 2	
$I_2, I_3, I_4, I_3, I_1, I_0$	$	Reduce 1	I → i
$I_4, I_3, I_4, I_3, I_1, I_0$	$	Reduce 2	I → I ; I
I_4, I_3, I_1, I_0	$	Reduce 2	I → I ; I
I_1, I_0	$	accept	S' → I

Correspondiente a la derivación: S' → I → I ; I → I ; I ; I → I ; I ; i → I ; i ; i → i ; i ; i

Correspondiente a una asociatividad derecha

Si le damos prioridad al shift, tenemos que:

Pila	Entrada	Acción	Salida
I_0	i;i;i $	Shift 2	
I_2, I_0	;i;i $	Reduce 1	I → i
I_1, I_0	;i;i $	Shitf 3	
I_3, I_1, I_0	i;i $	Shift 2	
I_2, I_3, I_1, I_0	;i $	Reduce 1	I → i
I_4, I_3, I_1, I_0	;i $	Reduce 2	I → I ; I
I_1, I_0	;i $	Shift 3	
I_3, I_1, I_0	i $	Shift 2	
I_2, I_3, I_1, I_0	$	Recuce 1	I → i
I_4, I_3, I_1, I_0	$	Reduce 2	I → I ; I
I_1, I_0	$	Accept	S' → I

Correspondiente a la derivación: S' → I → I ; I → I ; i → I ; I ; i → I ; i ; i → i ; i ; i

Correspondiente a una asociatividad izquierda

c) Compare la eciencia de ambos analizadores en terminos de espacio y de tiempo , para reconocer frases de la forma i (; i)^n.

Al darle prioridad al shift, obtuvimos una gramatica con asociatividad derecha. Por lo que, siempre leeremos el siguiente caracter (prioridad shift) en lugar de reducirlo. El análisis es análogo al problema anterior (con recursión derecha), resultando que la cantidad de memoria usada por la pila es de orden **O(n)**.

Al darle prioridad al reduce, obtuvimos gramática con asociatividad izquierda. Por lo que, preferimos reducir la pila antes que seguir leyendo. El análisis es análogo al problema anterior (con recursión izquierda), resultando que la cantidad de memoria usada por la pila es de orden **O(1)**.