Universidad Simón Bolívar
Traductores e Interpretadores (CI-3725)
Enero - Marzo 2016
Profesor: Ricardo Monascal

Guillermo Betancourt, carnet 11-10103
Gabriel Giménez, carnet 12-11006

Proyecto – BOT

Revisión teórico-práctica, tercera entrega

1. ***Sea*** $G1_i$ ***la gramática recursiva izquierda*** $(\{S\},\{a\},\{S \rightarrow Sa, S \rightarrow \lambda\},S)$ ***y sea*** $G1_d$ ***la gramática recursiva derecha*** $(\{S\},\{a\},\{S \rightarrow aS, S \rightarrow \lambda\},S)$ ***. Ambas generan el lenguaje denotado por la expresión regular*** $a*$ ***, i.e. el lenguaje*** $L(a*)$

   a) Muestre que ambas gramáticas son $LR(1)$ ***y construya sus analizadores sintácticos:***

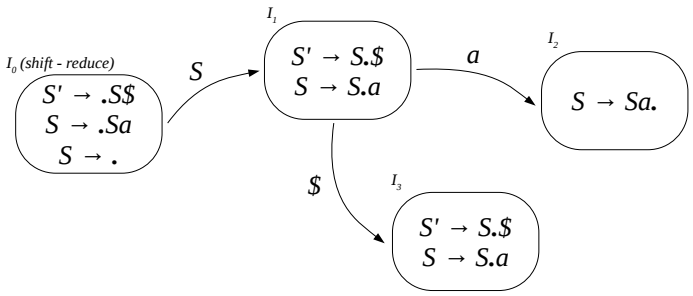
   Para construir el analizador sintáctico de ambas gramáticas, procedemos a crear el autómata de prefijos viables de cada una de ellas. Comencemos por $G1_i$.

   Puesto que el símbolo inicial está al lado derecho de alguna producción, aumentamos la gramática con un nuevo símbolo S' , insertamos un símbolo $\$$ como fin de archivo en la nueva producción y calculamos sus ítems.

   $S' \rightarrow S\$$ *(i)*
   $S \rightarrow Sa$ *(ii)*
   $\mid \lambda$ *(iii)*

   Le damos una numeración a cada producción para simplificar en referencias posteriores.

   Luego, calculando los ítems, el autómata obtenido es el siguiente:



   Procedemos a realizar la tabla de *first* y *follow* , con la cual intentaremos resolver el problema de shift – reduce en I_0 .

Símbolo no terminal	*FIRST*	*FOLLOW*
S'	λ	$\$$
S	λ	$\$$ a

Ahora, construimos la tabla del analizador sintáctico.

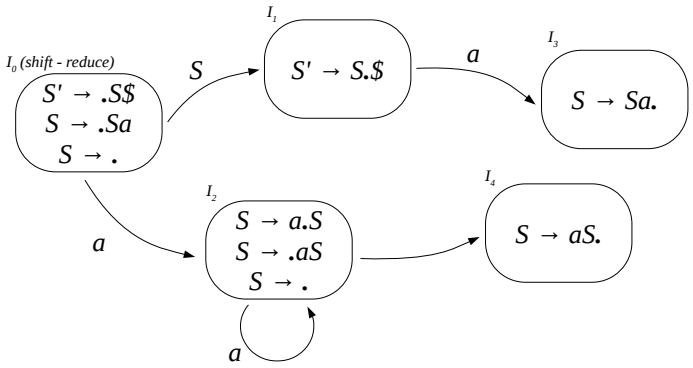
	Acciones		*Go to*	
Ítems	*a*	$	*S'*	*S*
I_0	*reducir(iii)*			I_1
I_1	*avanzar*(I_2)	*avanzar*(I_3)		
I_2	*reducir(ii)*			
I_3		*aceptar*		

Hacemos un procedimiento análogo para construir el analizador sintáctico de $G1_d$.

Puesto que el símbolo inicial está al lado derecho de alguna producción, aumentamos la gramática con un nuevo símbolo S' , insertamos un símbolo $\$$ como fin de archivo en la nueva producción y calculamos sus ítems.

$S' \rightarrow S\$$ *(i)*
$S \rightarrow aS$ *(ii)*
$\mid \lambda$ *(iii)*

Luego, calculando los ítems, el autómata obtenido es el siguiente:



Procedemos a realizar la tabla de *first* y *follow* , con la cual resolveremos el problema de shift – reduce en I_0 .

Símbolo no terminal	*FIRST*	*FOLLOW*
S'	*a* λ	$
S	*a* λ	$

Ahora, construimos la tabla del analizador sintáctico.

	Acciones		*Go to*	
Ítems	*a*	$	*S'*	*S*
I_0	*avanzar*(I_2)	*reducir(iii)*		I_1
I_1		*avanzar*(I_3)		
I_2	*avanzar*(I_2)	*reducir(iii)*		I_4
I_3		*aceptar*		
I_4	*reducir(ii)*			

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 $L(a*)$ ***, y de tiempo, i.e. cantidad de movimientos realizados por el autómata de pila para reconocer cada frase de*** $L(a*)$ ***.***

En términos de complejidad, el parser de la gramática $G1_i$ realiza $O(n)$ operaciones de reducción y $O(n)$ operaciones de avance, lo que da como resultado una complejidad de $O(n)$ para el reconocimiento de una frase a^n . La cantidad de pila utilizada para reconocer cualquier frase en el lenguaje dado es de máximo tres elementos, por lo que para toda frase $f \in L(a*)$, la cantidad de pila utilizada para saber si la frase pretenece o no al lenguaje es de $O(1)$.

El parser de la gramática $G1_d$ realiza $O(n)$ operaciones de reducción y $O(n)$ operaciones de avance, lo que da como resultado una complejidad de $O(n)$ para el reconocimiento de una frase a^n . La cantidad de pila utilizada para reconocer cualquier frase en el lenguaje dado es proporcional al tamaño de la frase de entrada, por lo que el orden de complejidad de elementos utilizados en la pila es de $O(n)$.

2. *Sea* $G2$ ***la gramática*** $(\{Instr\},\{;, IS\}, P2, Instr)$ ***con*** $P2$ ***compuesto por:***

Instr → *Instr* , *Instr*
| *IS*

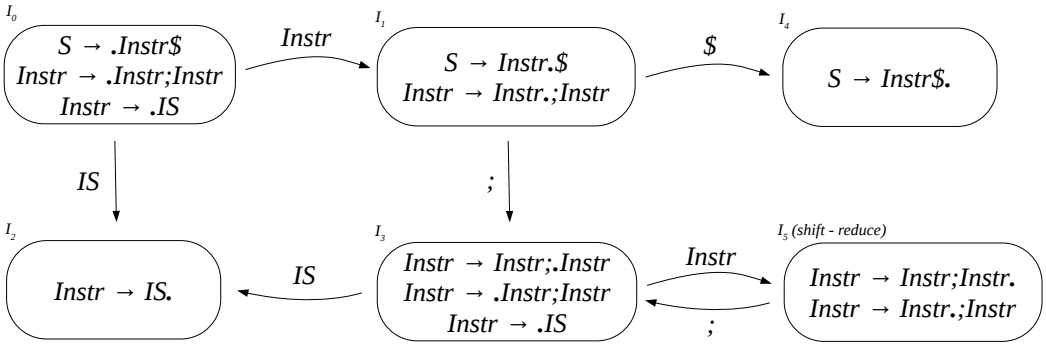
a) Muestre que $G2$ ***no es una gramática*** $LR(1)$ ***, intentando construir un analizador sintáctico*** $LR(1)$ ***para ella y consiguiendo que tal analizador tendrá un conflicto***

Para construir el analizador sintáctico de la gramática, procedemos a crear el autómata de prefijos viables de la misma.

Puesto que el símbolo inicial está al lado derecho de alguna producción, aumentamos la gramática con un nuevo símbolo S , insertamos un símbolo $\$$ como fin de archivo en la nueva producción y calculamos sus ítems.

S → *Instr* $ *(i)*
Instr → *Instr* , *Instr* *(ii)*
| *IS* *(iii)*

Luego, calculando los ítems, el autómata obtenido es el siguiente:



Observamos que en ítem I_5 existe en problema de shift – reduce.

b) A pesar de que la gramática $G2$ no es $LR(1)$, se puede construir un analizador sintáctico $LR(1)$ con conflictos para ella (lo cual corresponde a un autómata de pila no-determinístico). Construya tal analizador sintáctico, especificando cuál es el conflicto y de qué tipo (i.e. shift-reduce o reduce-reduce) es.

	Acciones			*Go to*	
Ítems	*IS*	*;*	*$*	*S*	*Instr*
I_0	*avanzar(I_2)*				I_1
I_1		*avanzar(I_3)*	*avanzar(I_4)*		
I_2	*reducir(iii)*				
I_3	*avanzar(I_2)*				I_5
I_4		*aceptar*			
I_5	*reducir(ii)*	*avanzar(I_3) o reducir(ii)*	*reducir(ii)*		

Como se mencionó en el apartado anterior, I_5 presenta un problema de shift – reduce al momento de reconocer el símbolo $;$, puesto que puede avanzar al estado I_3 con dicho símbolo o bien puede reducir por la producción *(ii)*.

c) Considere las dos alternativas de eliminación del conflicto (i.e. en favor del shift o en favor del reduce en caso de un conflicto shift-reduce, o en favor de una producción o de otra en caso de un conflicto reduce-reduce). Muestre, para ambas alternativas de eliminación del conflicto, la secuencia de reconocimiento de la frase $IS;IS;IS$ dando como salida la secuencia de producciones reducidas. ¿A qué corresponde cada una de las alternativas: a asociar el operador de secuenciación hacia la izquierda o hacia la derecha?

Con la alternativa en favor del shift, la tabla del analizador sintáctico resultaría de la siguiente manera:

	Acciones			Go to	
Ítems	IS	;	$	S	Instr
I_0	avanzar(I_2)				I_1
I_1		avanzar(I_3)	avanzar(I_4)		
I_2	reducir(iii)				
I_3	avanzar(I_2)				I_5
I_4		aceptar			
I_5	reducir(ii)	avanzar(I_3)	reducir(ii)		

Luego, con la alternativa en favor del reduce, la tabla del analizador sintáctico resultaría de la siguiente manera:

	Acciones			Go to	
Ítems	IS	;	$	S	Instr
I_0	avanzar(I_2)				I_1
I_1		avanzar(I_3)	avanzar(I_4)		
I_2	reducir(iii)				
I_3	avanzar(I_2)				I_5
I_4		aceptar			
I_5	reducir(ii)				

Ahora, reconocemos la frase $IS\,;\,IS\,;\,IS$ con ambos analizadores sintácticos.

En favor del shift

Pila	Entrada	Acción
I_0	IS;IS;IS$	avanzar(I_2)
I_2I_0	;IS;IS$	reducir(iii)
I_1I_0	;IS;IS$	avanzar(I_3)
$I_3I_1I_0$	IS;IS$	avanzar(I_2)
$I_2I_3I_1I_0$	;IS$	reducir(iii)
$I_5I_3I_1I_0$	;IS$	avanzar(I_3)
$I_3I_5I_3I_1I_0$	IS$	avanzar(I_2)
$I_2I_3I_5I_3I_1I_0$	$	reducir(iii)
$I_5I_3I_5I_3I_1I_0$	$	reducir(ii)
$I_5I_3I_1I_0$	$	reducir(ii)
I_1I_0	$	avanzar(I_4)
$I_4I_1I_0$	$	aceptar

En favor del reduce

Pila	Entrada	Acción
I_0	IS;IS;IS$	avanzar(I_2)
I_2I_0	;IS;IS$	reducir(iii)
I_1I_0	;IS;IS$	avanzar(I_3)
$I_3I_1I_0$	;IS;IS$	avanzar(I_2)
$I_2I_3I_1I_0$	IS;IS$	reducir(iii)
$I_5I_3I_1I_0$	;IS$	reducir(ii)
I_1I_0	;IS$	avanzar(I_3)
$I_3I_1I_0$	IS$	avanzar(I_2)
$I_2I_3I_1I_0$	$	reducir(iii)
$I_5I_3I_1I_0$	$	reducir(ii)
I_1I_0	$	avanzar(I_4)
$I_4I_1I_0$	$	aceptar

d) En la Etapa II se concluyó que era indiferente resolver esta ambigüedad hacia la izquierda o hacia la derecha. Compare ahora la eficiencia de ambas alternativas, en términos de la cantidad de pila y del tiempo que se utiliza para reconocer frases de la forma $IS;(IS)^n$ ***con n un número natural. ¿Cuál alternativa conviene entonces utilizar?***

En términos de complejidad, ambos parsers realizan $O(n)$ operaciones de reducción y $O(n)$ operaciones de avance, lo que da como resultado una complejidad de $O(n)$ para el reconocimiento de una frase $IS(;IS)^n$. La cantidad de pila utilizada para reconocer cualquier frase en el lenguaje dado es, en favor del shift, de orden $O(n)$, mientras que en favor del reduce, la cantidad de pila utilizada es de máximo cuatro elementos, es decir $O(1)$. La alternativa en favor del reduce es más conveniente en términos de espacio pues utiliza menor cantidad de pila.