



Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3725 - Traductores e Interpretadores
Enero-Marzo 2016

Proyecto 3

Traductores

Samuel Arleo
10-10969

Sergio Terán
11-11020

07 de Marzo, 2016

1. Formulación e Implementación

2. Revisión Teórico-Practica

Pregunta 1

(a)

$$(a.1) \ G_1 = (\{S\}, \{a\}, \{S \rightarrow Sa, S \rightarrow \lambda\}, S)$$

Determinemos si la gramática

$$\begin{array}{l} S \rightarrow Sa \\ S \rightarrow \lambda \end{array}$$

Es LR(1) y construyamos su analizador sintáctico. Comenzamos por aumentar la gramática con el símbolo S' y agregando el símbolo $\$$ al final de la primera entrega. A demás enumeramos las producciones.

$$\begin{array}{lll} (i) & S' & \rightarrow S\$ \\ (ii) & S & \rightarrow Sa \\ (iii) & S & \rightarrow \lambda \end{array}$$

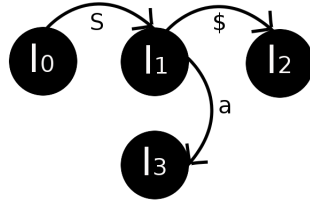
Construimos los conjuntos FIRST y FOLLOW para los simbolos no terminales:

$$\begin{array}{l} \text{FIRST}(S') = \{ \lambda, a \} \\ \text{FIRST}(S) = \{ \lambda, a \} \\ \text{FOLLOW}(S') = \{ \$ \} \\ \text{FOLLOW}(S) = \{ a, \$ \} \end{array}$$

El conjunto de clauduras nos queda:

$$\begin{array}{lll} I_0 : & S' & \rightarrow \cdot S\$ \\ & S & \rightarrow \cdot Sa \\ & S & \rightarrow \cdot \\ I_1 : & S' & \rightarrow S \cdot \$ \\ & S & \rightarrow S \cdot a \\ I_2 : & S' & \rightarrow S\$ \cdot \\ I_3 : & S & \rightarrow Sa \cdot \end{array}$$

Construimos el automata de prefijos viables, que nos queda de la forma:



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

	Acciones		Goto	
	a	\$	S'	S
I_0	r(iii)	r(iii)		1
I_1	s(3)	s(2)		
I_2	acc			
I_3	r(ii)	r(ii)		

$$(a.2) \ G_1 = (\{S\}, \{a\}, \{S \rightarrow aS, S \rightarrow \lambda\}, S)$$

Determinemos si la gramática

$$\begin{aligned} S &\rightarrow aS \\ S &\rightarrow \lambda \end{aligned}$$

Es LR(1) y construyamos su analizador sintáctico. Comenzamos por aumentar la gramática con el símbolo S' y agregando el símbolo $\$$ al final de la primera entrega. A demás enumeramos las producciones.

$$\begin{aligned} (i) \quad S' &\rightarrow S\$ \\ (ii) \quad S &\rightarrow aS \\ (iii) \quad S &\rightarrow \lambda \end{aligned}$$

Construimos los conjuntos FIRST y FOLLOW para los simbolos no terminales:

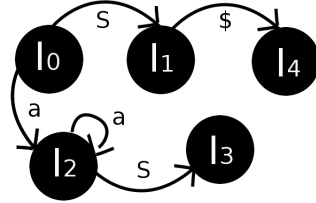
$$\begin{aligned} \text{FIRST}(S') &= \text{FIRST}(S) = \{ \lambda, a \} \\ \text{FOLLOW}(S') &= \text{FOLLOW}(S) = \{ \$ \} \end{aligned}$$

El conjunto de clauduras nos queda:

$$\begin{aligned} I_0 : \quad S' &\rightarrow \cdot S\$ \\ &S \rightarrow \cdot aS \\ &S \rightarrow \cdot \\ I_1 : \quad S' &\rightarrow S \cdot \$ \\ I_2 : \quad S &\rightarrow a \cdot S \\ &S \rightarrow \cdot aS \\ &S \rightarrow \cdot \end{aligned}$$

$$\begin{array}{lcl} I_3 & : & S \longrightarrow aS \cdot \\ I_4 & : & S' \longrightarrow S\$ \cdot \end{array}$$

Construimos el automata de prefijos viables, que nos queda de la forma:



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

	Acciones		Goto	
	a	\$	S'	S
I_0	s(2)	r(iii)		1
I_1		s(4)		
I_2	s(2)	r(iii)		3
I_3		r(ii)		
I_4		acc		

- (b) Comparando las tablas de $G1_i$ y $G1_d$ vemos que la tabla de parsing de $G1_i$ contiene menos filas que $G1_d$.

En la pila, $G1_i$, consume un maximo de 3 items en la pila, en el caso especifico de la frase aaa la pila para $G1_d$ puede llegar a tener hasta 4 items, por lo que podemos ver que $G1_i$ es mas eficiente en cuanto al consumo de memoria por parte de la pila.

a demás vemos que para reconocer las frases, la cantidad de movimientos realizados por ambos es de $2 * n + 3$ po lo que el tiempo de ejecucion es de orden $O(n)$ donde n es la cantidad de 'a' en la frase.

Pregunta 2

- (a)
- (i) $S' \longrightarrow Instr$
 - (ii) $Instr \longrightarrow Instr ; Instr$
 - (iii) $Instr \longrightarrow IS$

$$\text{FIRST}(S') = \text{FIRST}(Instr) = \{ IS \}$$

$$\text{FOLLOW}(S') \{ \$ \}$$

$$\text{FOLLOW}(Instr) \{ ;, \$, IS \}$$

$$\begin{array}{ll}
I_0 : & S' \longrightarrow \cdot Instr \$ \\
& Instr \longrightarrow \cdot Instr ; Instr \\
& Instr \longrightarrow \cdot IS \\
I_1 : & S' \longrightarrow Instr \cdot \$ \\
& Instr \longrightarrow Instr \cdot ; Instr \\
I_2 : & Instr \longrightarrow IS \cdot \\
I_3 : & S' \longrightarrow Instr \$ \cdot \\
I_4 : & Instr \longrightarrow Instr ; \cdot Instr \\
& Instr \longrightarrow \cdot Instr ; Instr \\
& Instr \longrightarrow \cdot IS \\
I_5 : & Instr \longrightarrow Instr ; Instr \cdot \\
& Instr \longrightarrow Instr \cdot ; Instr
\end{array}$$

En la regla I_5 vemos que existe un conflicto

- (b) Este conflicto, del tipo *shift/reduce*, lo intentaremos solucionar usando el algoritmo de SLR(1), apoyandonos con los FIRST y FOLLOW que ya hemos calculado.

	Acciones			Goto	
	;	IS	\$	S'	$Instr$
I_0	s(2)			1	
I_1	s(4)		s(3)		
I_2	r(iii)	r(iii)	r(iii)		
I_3	acc				
I_4	s(2)				
I_4	s(4)/r(ii)	r(ii)	r(ii)		

- (c) Secuencia de reconocimiento para la frase IS;IS;IS, dando prioridad al *shift* en el conflicto *shift/reduce*

Pila	Entrada	Acción	Salida
I_0	IS;IS;IS\$	s(2)	
$I_2 I_0$;IS;IS\$	r(iii)	(iii)
$I_1 I_0$;IS;IS\$	s(4)	(iii)
$I_4 I_1 I_0$	IS;IS\$	s(2)	(iii)
$I_2 I_4 I_1 I_0$;IS\$	r(iii)	(iii), (iii)
$I_5 I_4 I_1 I_0$;IS\$	s(4)	(iii), (iii)
$I_4 I_5 I_4 I_1 I_0$	IS\$	s(2)	(iii), (iii)
$I_2 I_4 I_5 I_4 I_1 I_0$	\$	r(iii)	(iii), (iii), (iii)
$I_5 I_4 I_5 I_4 I_1 I_0$	\$	r(ii)	(ii), (iii), (iii), (iii)
$I_5 I_4 I_1 I_0$	\$	r(ii)	(ii), (ii), (iii), (iii), (iii),
$I_1 I_0$	\$	s(3)	(ii), (ii), (iii), (iii), (iii),
$I_3 I_1 I_0$	\$	acc	(ii), (ii), (iii), (iii), (iii),

Secuencia de reconocimiento para la frase IS;IS;IS, dando prioridad al *reduce* en el conflicto *shift/reduce*

Pila	Entrada	Acción	Salida
I_0	IS;IS;IS\$	s(2)	
$I_2 I_0$;IS;IS\$	r(iii)	(iii)
$I_1 I_0$;IS;IS\$	s(4)	(iii)
$I_4 I_1 I_0$	IS;IS\$	s(2)	(iii)
$I_2 I_4 I_1 I_0$;IS\$	r(iii)	(iii), (iii)
$I_5 I_4 I_1 I_0$;IS\$	r(ii)	(ii), (iii), (iii)
$I_1 I_0$;IS\$	s(4)	(ii), (iii), (iii)
$I_4 I_1 I_0$	IS\$	s(2)	(ii), (iii), (iii)
$I_2 I_4 I_1 I_0$	\$	r(iii)	(iii), (ii), (iii), (iii)
$I_5 I_4 I_1 I_0$	\$	r(ii)	(ii), (iii), (ii), (iii), (iii)
$I_1 I_0$	\$	s(3)	(ii), (iii), (ii), (iii), (iii)
$I_3 I_1 I_0$	\$	acc	(ii), (iii), (ii), (iii), (iii)

Podemos concluir que a pesar de que en ambos casos se genera la misma frase sin asociatividades ‘explicitas’, al priorizar *shift* nos genera una asociatividad ‘implicita’ a la izquierda, mientras que al priorizar *reduce* nos genera una asociatividad ‘implicita’ a la derecha.

- (d) Para comparar la eficiencia de ambas opciones, reconocimos frase de tamaño $IS(;IS)^2$, $IS(;IS)^3$, $IS(;IS)^4$ y $IS(;IS)^5$ para estudiar sus comportamientos.

Comparando el comportamiento de las pilas en ambas alternativas, vemos que dando prioridad al *shift* el tamaño máximo de la pila es $2 * (n + 1)$ para frases de la forma $IS(IS)^n$ mientras que para el

reduce el tamaño máximo de la pila es 4 de manera constante.

Por otro lado vemos que ambas alternativas cuestan $4(n + 1)$ pasos para aceptar la frase, luego, en cuestión de tiempo son de orden $O(n)$.

Podemos concluir que en cuestión de eficiencia la alternativa de *reduce* es más conveniente ya que cuesta la misma cantidad de tiempo, pero utiliza menos memoria en su pila.