TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES

PRÁCTICA PUNTUABLE – ANÁLISIS SINTÁCTICO

CARLOS BENITO JAREÑO

Apartado 1: Minimizar la gramática G

1. Eliminar variables que no lleven a palabras, así como reglas donde aparezcan.
   Iteraciones:
   a. N' = {S, A, B, D}
      La única variable que falta en N' es **C**, pero su única regla es **C → aCb**, lo cual no cumple el requisito de que todas las variables de la regla estén en N'.

   Por tanto, N' no cambia más, así que eliminamos la variable **C** y todas las reglas que la contengan; es decir:
   - C
   - C → aCb
   - S → SCS

2. Eliminar símbolos inalcanzables desde **S**, así como reglas donde aparezcan.
   Iteraciones:
   a. J = {S} N' = {S} T = Ø
      Extraemos S…
   b. J = {A, B} N' = {S, A, B} T = {a, b, c, d}
      Extraemos A…
   c. J = {B} N' = {S, A, B} T = {a, b, c, d, ε}
      Extraemos B…
   d. J = Ø N' = {S, A, B} T = {a, b, c, d, ε}

   Nos hemos quedado sin variables en J; por tanto, eliminamos la variable **D,** el símbolo terminal **e** y todas las reglas que los contengan:
   - D
   - e
   - D → aAb
   - D → ace

   Finalmente, la gramática **G2** queda de la siguiente manera:

   S → aAc | Ba | db
   A → aA | ε
   B → Sd | a

Apartado 2: Modificar G2 para intentar convertirla en LL(1)

1. Eliminar recursividades por la izquierda y factorizar
    a. Recursividades generales
    Variables: S A B
    - Bucle exterior: A Bucle interior: S
      No hay reglas del estilo A → Sβ.
    - Bucle exterior: B Bucle interior: S
      B → Sd se convierte en…
        - B → aAcd | Bad | dbd
    - Bucle exterior: B Bucle interior: A
      No hay reglas del estilo B → Aβ

    Tras eliminar recursividades generales por la izquierda, obtenemos G2':

    S → aAc | Ba | db
    A → aA | ε
    B → aAcd | Bad | dbd | a

    b. Recursividades directas
    B → Bad posee recursividad directa por la izquierda, así que las reglas **B → Bad | aAcd | dbd | a** se convierten en…
        - B → aAcdB' | dbdB' | aB'
        - B' → adB' | ε

    Por tanto, obtenemos G2'':

    S → aAc | Ba | db
    A → aA | ε
    B → aAcdB' | dbdB' | aB'
    B' → adB' | ε

    c. Factorizar
    Solo podemos factorizar dos reglas de B, las cuales empiezan por 'a':
    B → aC | dbdB'
    C → AcdB' | B'
    Por tanto, la gramática **G3** quedaría así (renombrada por comodidad):

    S → aAc | Ba | db
    A → aA | ε
    B → aC | dbdD
    C → AcdD | D
    D → adD | ε

2. Generar la tabla LL(1)
    Primero, necesitamos construir los conjuntos First y Follow.
    - First(S) = {a, d} ∪ First(B) = {a, d}
    - First(A) = {a, ε}
    - First(B) = {a, d}
    - First(C) = First(A) ∪ {c} ∪ First(D) = {a, c, ε}
    - First(D) = {a, ε}

- Follow(S) = {$}
- Follow(A) = {c} ∪ Follow(A) = {c}
- Follow(B) = {a}
- Follow(C) = Follow(B) = {a}
- Follow(D) = Follow(B) ∪ Follow(C) ∪ Follow(D) = {a}

Con dichos conjuntos, podemos construir la tabla LL(1).

	a	**b**	**c**	**d**	**$**
S	S → aAc S → Ba			S → Ba S → db	
A	A → aA		A → ε		
B	B → aC			B → dbdD	
C	C → AcdD C → D		C → AcdD		
D	D → adD D → ε				

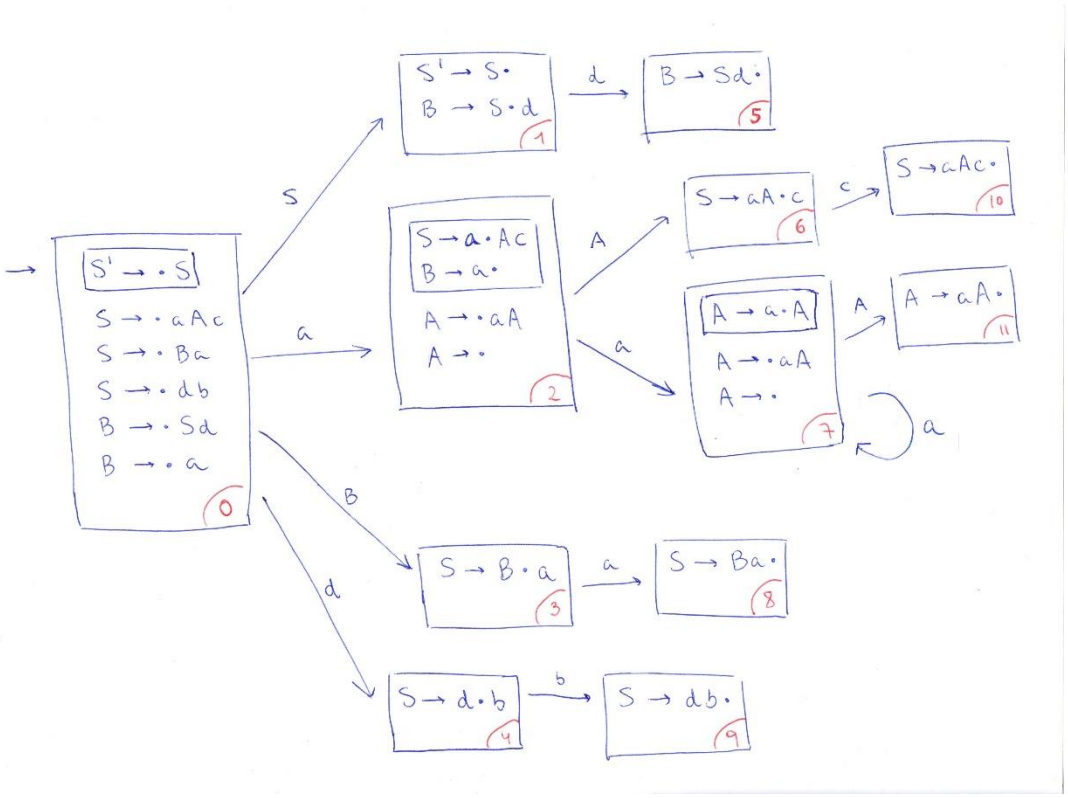
Podemos apreciar que existen celdas con más de una regla; por tanto, podemos concluir que **la gramática G3 no es LL(1)**.

Apartado 3: Generar la tabla SLR(1) para G2 y comprobar si es SLR(1) o no

Primero, modificamos la gramática G2 para añadir la variable **S'**:

S' → S
S → aAc | Ba | db
A → aA | ε
B → Sd | a

A continuación, creamos el autómata con las clausuras de los ítems:



Finalmente, construimos las tablas Action y Goto, ayudándonos de los conjuntos Follow:

Follow(S') = {$}
Follow(S) = Follow(S') ∪ {d} = {d, $}
Follow(A) = {c} ∪ Follow(A) = {c}
Follow(B) = {a}

	ACTION table					GOTO table		
State	a	b	c	d	$	A	B	S
0	Shift 2			Shift 4			Goto 3	Goto 1
1				Shift 5	Accept			
2	Shift 7 R(B→a)		R(A→ε)			Goto 6		
3	Shift 8							
4		Shift 9						
5	R(B→Sd)							
6			Shift 10					
7	Shift 7		R(A→ε)			Goto 11		
8				R(S→Ba)	R(S→Ba)			
9				R(S→db)	R(S→db)			
10				R(S→aAc)	R(S→aAc)			
11			R(A→aA)					

Podemos ver que en la celda [2,a] hay dos posibles opciones: un shift o una reducción. Por lo tanto, **la gramática G2 no es SLR(1)**.

Esto último lo podemos respaldar si implementamos la gramática en SLY: examinando el debugfile, nos encontramos esto:

```
state 4

    (3) S -> a . A c
    (6) B -> a .
    (4) A -> .
    (5) A -> . a A
  ! shift/reduce conflict for a resolved as shift
    c              reduce using rule 4 (A -> .)
    a              shift and go to state 8

    A                           shift and go to state 9
```

El estado 4 del programa detecta un conflicto entre shift y reduce cuando recibe el terminal **a**. Si nos fijamos en las cuatro reglas del estado, podemos ver que coincide exactamente con el estado 2 de nuestro autómata, el cual, según la tabla Action, también posee dos posibles acciones cuando se encuentra el terminal **a**: **shift 7** o **reducir a B→a**; es decir, un conflicto shift/reduce tal como el que reporta SLY.