

12 enero 2017

Gramática que no se puede modificar:

$S \rightarrow A + B; C$

$A \rightarrow xA | C$

$B \rightarrow By | \lambda$

$C \rightarrow (A) | \lambda$

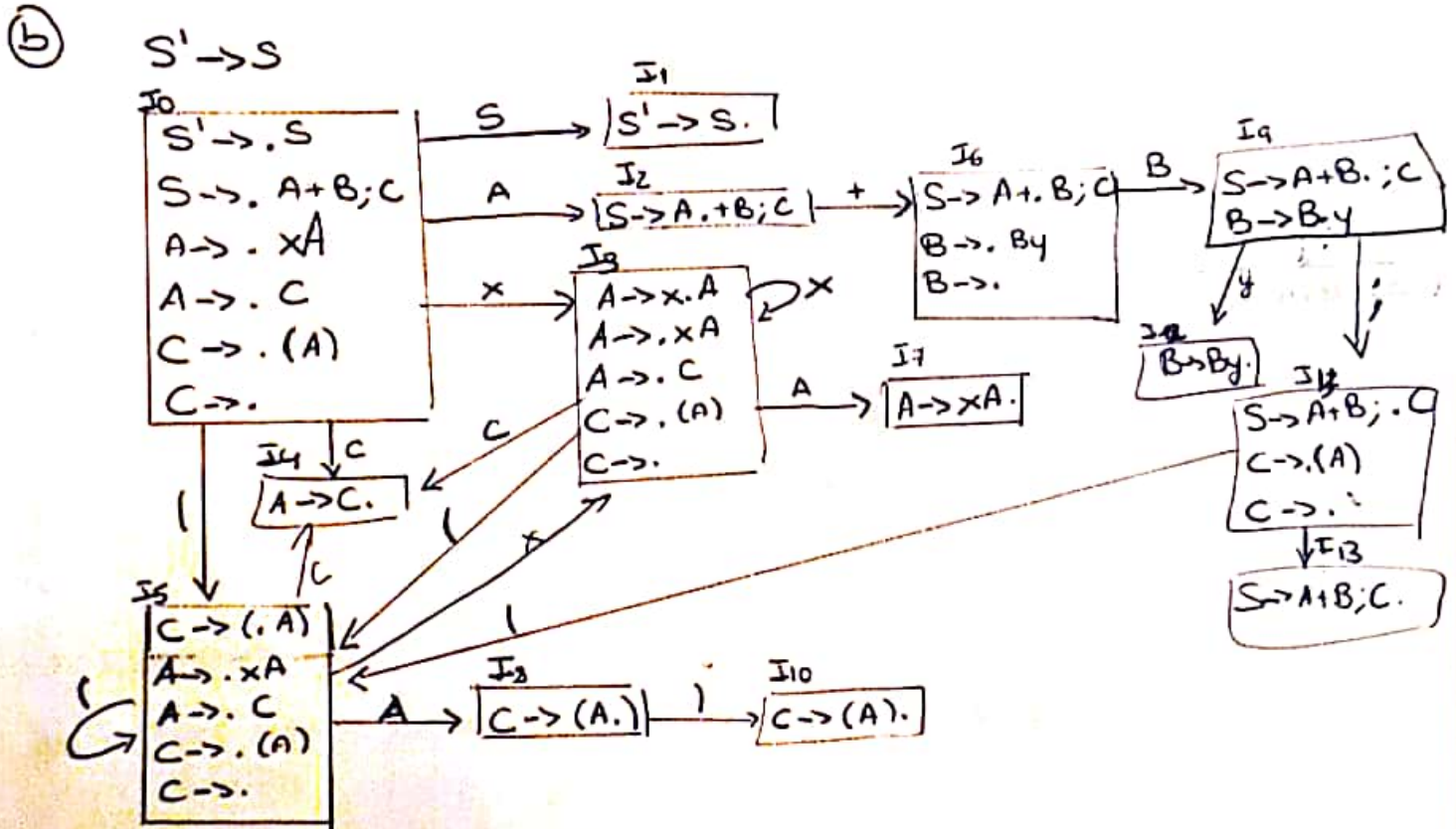
a) Determinar razonadamente qué tipo de Analizador Sintáctico puede ser válido para esta gramática.

b) Construir el tablo de dicho Analizador

Sintáctico

① Un analizador sintáctico descendente no sería válido para esta gramática, porque es recursiva por la izquierda, así que hay que optar por el analizador ascendente.

$\hookrightarrow B \rightarrow By$



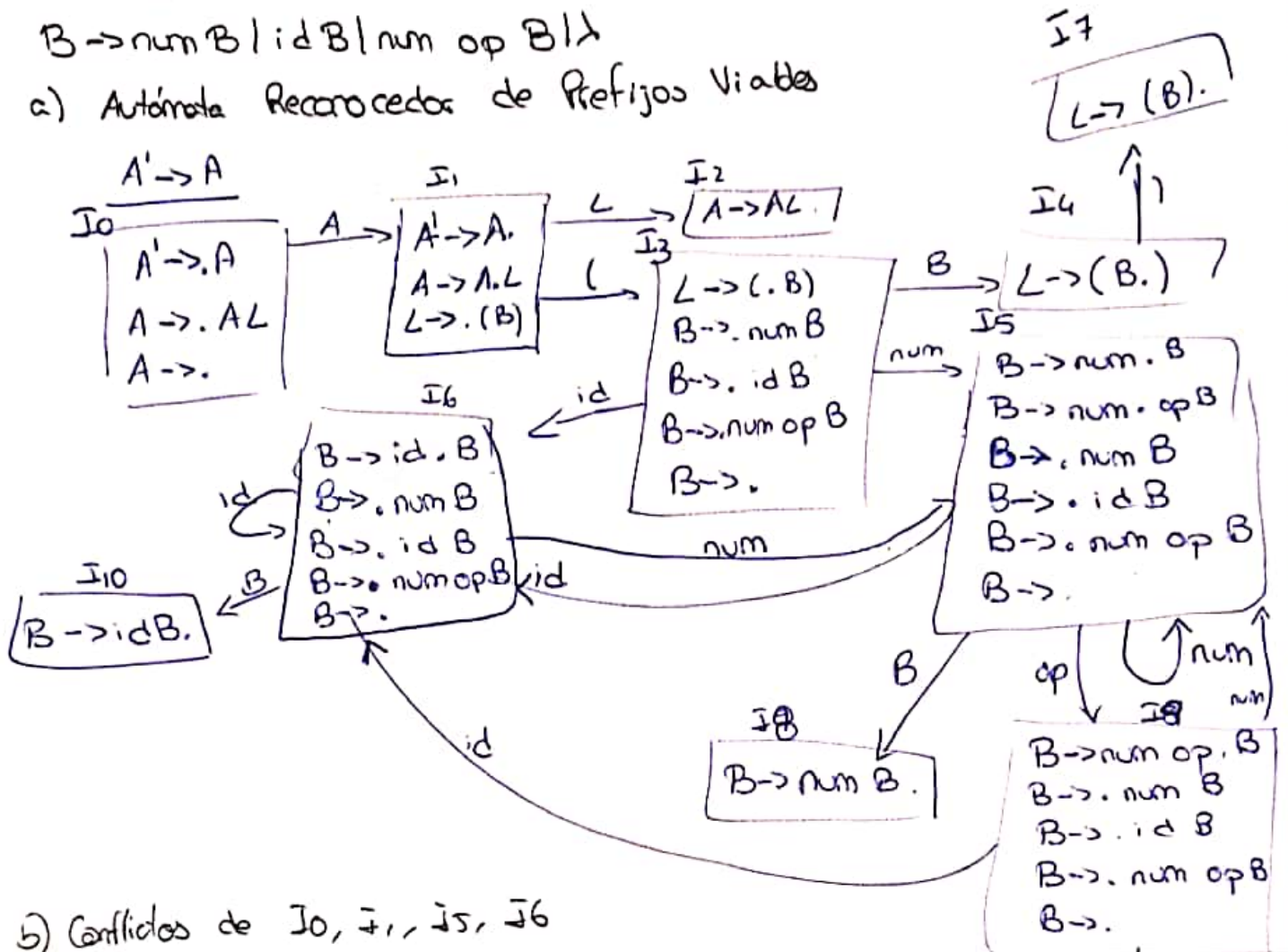
13 diciembre 2010

$A \rightarrow AL \mid \lambda$

$L \rightarrow (B)$

$B \rightarrow \text{num } B \mid \text{id } B \mid \text{num op } B \mid \lambda$

a) Autómata Reconocedor de Prefijos Viabes



b) Conflictos de I_0, I_1, I_5, I_6

I_0 : Red-Desp: como no hay desplazamiento no hay conflicto

I_1 : Reducción-Desp:

$\text{Follow}(A') = \{ \$ \}$ y $\text{Desp} = \{ ' \} \rightarrow$ NO CONFLICTO

I_5 : Reducción-Desp: $\text{Follow}(B) = \{) \}$, $\text{Desp} = \{ \text{num}, \text{id} \}$

NO hay conflicto

I_6 : Red-Desp: $\text{Follow}(B) = \{) \}$, $\text{Desp} = \{ \text{id}, \text{num} \}$

NO hay conflicto

I_0 : Rec-Ped: no hay conflicto xq no hay reducción ocurrencia de $A \rightarrow$.

c) Justificar si la gramática es $LL(1)$. Si no lo es, modificarla y demostrar que la nueva gramática cumple la condición LL .

$A \rightarrow AL | \lambda$

• Recursiva por la izquierda (A)

$L \rightarrow (B)$

• No factorizable (B)

$B \rightarrow \text{num } B | \text{id } B | \text{num op } B | \lambda$

$A \rightarrow L A$

$\text{FIRST}(L A) \cap \text{FOLLOW}(A) = \{ (\cap \{ \$ \} = \emptyset$

$A \rightarrow \lambda$

$L \rightarrow (B)$

$\text{FIRST}(\text{num } C) \cap \text{FIRST}(\text{id } B) = \{ \text{num } \cap \{ \text{id } \} = \emptyset$

$B \rightarrow \text{num } C$

$\text{FIRST}(\text{num } C) \cap \text{FOLLOW}(B) = \{ \text{num } \cap \{) \} = \emptyset$

$B \rightarrow \text{id } B$

$\text{FIRST}(\text{id } B) \cap \text{FOLLOW}(B) = \{ \text{id } \cap \{) \} = \emptyset$

$B \rightarrow \lambda$

$\text{FIRST}(B) \cap \text{FIRST}(\text{op } B) = \{ \text{num, id, } \lambda \cap \{ \text{op } \} = \emptyset$

$C \rightarrow B$

$C \rightarrow \text{op } B$

La gramática es $LL(1)$.

10 enero 2018

$S \rightarrow x y z$

$x \rightarrow \lambda | x^*$

$y \rightarrow z y | +$

$z \rightarrow -$

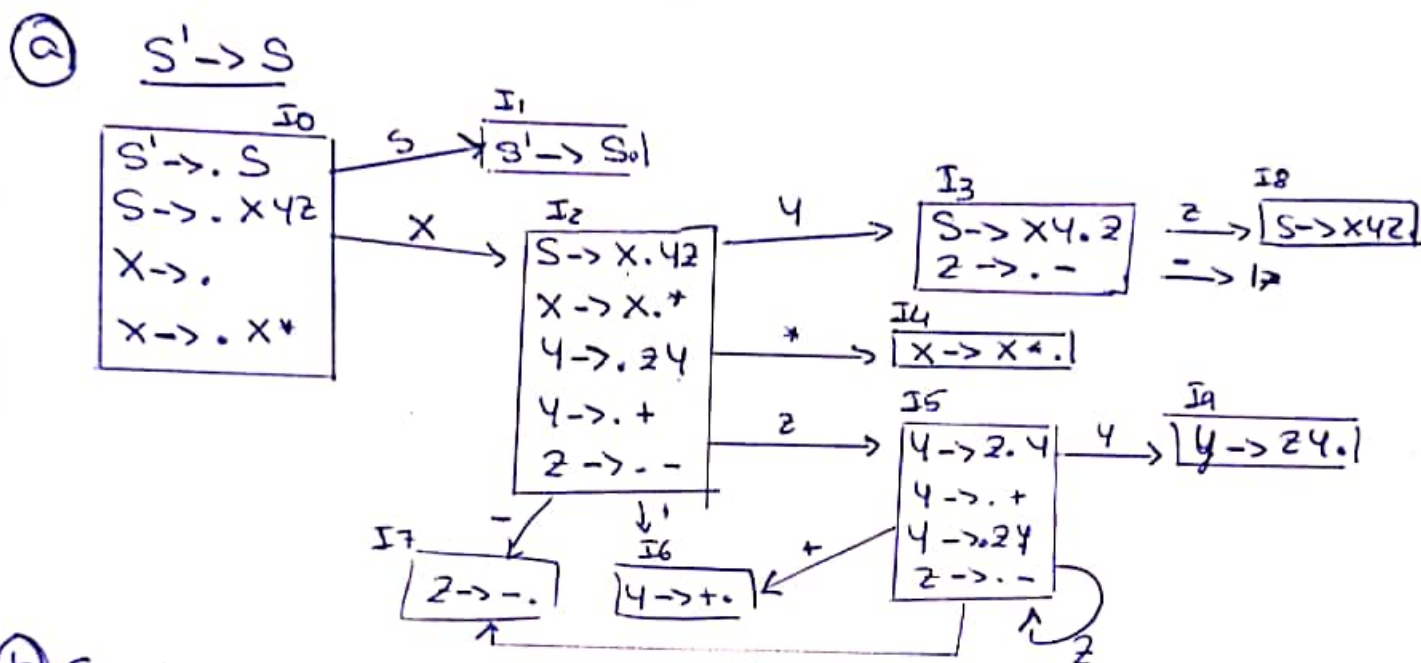
a) Construir autómata reconocedor de Prefijos

Viablos para esta gramática.

b) Analizar los posibles conflictos de generar un Analizador LR

c) Justificar si la gramática es LL. En caso negativo, transformarla para que lo sea (cambiando solo las reglas del no terminal implicado)

d) Construir la tabla del An. Sintético Descendente para la gramática LL



b) En el autómata no aparece ningún estado con dos reducciones o con reducción y desplazamiento, así que no hay posibilidad de conflicto.

c) x tiene recursividad por la izquierda en la regla $x \rightarrow x^*$. Cambiamos las reglas de x para eliminar la recursividad: $x \rightarrow \lambda | * x$ (reglas explícitas)

Para las reglas de x

$FIRST(x) = FOLLOW(x) = \{+, -\}$

$FIRST(*x) = \{*\}$

$FOLLOW(x) \cap FIRST(*x) = \{+, -\} \cap \{*\} = \emptyset$

Para las reglas de y

$FIRST(zy) \cap FIRST(+) = \{-\} \cap \{+\} = \emptyset$

$FIRST(S) = \{x, y, +\}$

$FIRST(*x) = \{*\}$

$FOLLOW(x) = \{+, -\}$

d)

$S \rightarrow x y z$
 $x \rightarrow \lambda | * x$
 $y \rightarrow z y | +$
 $z \rightarrow -$

	+	-	*	\$
S	$S \rightarrow x y z \cdot$	$S \rightarrow x y z$	$S \rightarrow x y z$	
x	$x \rightarrow \lambda$	$x \rightarrow \lambda$	$x \rightarrow * x$	
y	$y \rightarrow +$	$y \rightarrow z y$		
z		$z \rightarrow -$		

Febrero 2008

$S \rightarrow OAAO \mid \lambda A \lambda \mid \lambda$
 $A \rightarrow B \mid C$
 $B \rightarrow O \lambda \mid \lambda O$
 $C \rightarrow OO \mid \lambda \lambda$

- a) Comprobar condiciones LL1. Si no es LL1 transformar la gramática para que lo sea
- b) Construir An. Sintáctico Descendente Predictivo Recursivo

a) $FIRST(S) = \{O, \lambda, \lambda\}$ $FIRST(B) \cap FIRST(C) = \{O, \lambda\} \cap \{O, \lambda\} \neq \emptyset$
 $FOLLOW(S) = \{\$ \}$
 $FIRST(A) = \{O, \lambda\}$
 $FOLLOW(A) = \{\lambda\}$
 $FIRST(B) = \{O, \lambda\}$
 $FOLLOW(B) = \{\lambda\}$
 $FIRST(C) = \{O, \lambda\}$
 $FOLLOW(C) = \{\lambda\}$

La gramática NO es LL1 porque la intersección de las reglas de A no es el conjunto vacío.

Si se sustituyen los no terminales B y C por sus reglas, quedaría la gramática:

$S \rightarrow OAAO \mid \lambda A \lambda \mid \lambda$

$A \rightarrow O \lambda \mid \lambda O \mid OO \mid \lambda \lambda$

Factorizar la nueva gramática:

- 1 $S \rightarrow OAAO$
- 2 $S \rightarrow \lambda A \lambda$
- 3 $S \rightarrow \lambda$
- 4 $A \rightarrow OD$
- 5 $A \rightarrow \lambda D$
- 6 $D \rightarrow O$
- 7 $D \rightarrow \lambda$

Comprobar que la nueva gramática es LL(1)

- Reglas de S:
 $FIRST(OAAO) = \{O\}$
 $FIRST(\lambda A \lambda) = \{\lambda\}$
 $FOLLOW(S) = \{\$ \}$

- Reglas de A:
 $FIRST(OD) = \{O\}$
 $FIRST(\lambda D) = \{\lambda\}$

- Reglas de D:
 $FIRST(O) = \{O\}$
 $FIRST(\lambda) = \{\lambda\}$

b) Function S(X)

```

if (token == 'O') {
    print(1);
    equipare('O');
    A();
    equipare('O');
} else if (token == 'λ') {
    print(2);
    equipare('λ');
    A();
    equipare('λ');
} else if (token == 'λ') {
    print(3);
} else {
    error();
}

```

Function A(X)

```

if (token == 'O') {
    print(4);
    equipare('O');
    D();
} else if (token == 'λ') {
    print(5);
    equipare('λ');
    D();
} else {
    error();
}

```

Function D(X)

```

if (token == 'O') {
    print(6);
    equipare('O');
} else if (token == 'λ') {
    print(7);
    equipare('λ');
} else {
    error();
}

```

30 junio 2017

$P \rightarrow DP \mid IP \mid \lambda$

$D \rightarrow Tid$

$T \rightarrow int \mid real$

$I \rightarrow inicia \ id \ V$

$V \rightarrow cte-real \ V \mid cte-int \ V \mid id$

a) Demostrar si la gramática es LL(1).

b) Diseñar la tabla del Analizador Sintáctico LL(1).

c) Construir el estado inicial del Automata reconocedor de prefijos viables de un Analizador Sintáctico Ascendente LR, así como todos los demás estados que deriven de él en una sola transición. Determinar si existen conflictos entre estos estados y en su caso, cuáles?

d) En el supuesto en el que se introdujera al lenguaje generado por esta gramática la restricción de que en la lista de valores (V) de la semántica inicial (I) siempre debe existir un mayor número de constantes reales que enteras, ¿qué cambios habría que introducir al An. Sintáctico de b) y c)?

a) Reglas de P

$$FIRST(DP) \cap FIRST(IP) = \{int, real\} \cap \{inicia\} = \emptyset$$

$$FIRST(DP) \cap FOLLOW(P) = \{int, real\} \cap \{\$ \} = \emptyset$$

$$FIRST(IP) \cap FOLLOW(P) = \{inicia\} \cap \{\$ \} = \emptyset$$

Reglas de T

$$FIRST(int) \cap FIRST(real) = \{int\} \cap \{real\} = \emptyset$$

Reglas de V

$$FIRST(cte-real \ V) \cap FIRST(cte-int \ V) = \{cte-real\} \cap \{cte-int\} = \emptyset$$

$$FIRST(cte-real \ V) \cap FIRST(id) = \{cte-real\} \cap \{id\} = \emptyset$$

$$FIRST(cte-int \ V) \cap FIRST(id) = \{cte-int\} \cap \{id\} = \emptyset$$

b)

	id	int	real	inicia	cte-real	cte-int	\$
P		$P \rightarrow DP$	$P \rightarrow DP$	$P \rightarrow IP$			$P \rightarrow \lambda$
D		$D \rightarrow Tid$	$D \rightarrow Tid$				
T		$T \rightarrow int$	$T \rightarrow real$				
I				$I \rightarrow inicia \ id \ V$			
V	$V \rightarrow id$				$V \rightarrow cte-real \ V$	$V \rightarrow cte-int \ V$	

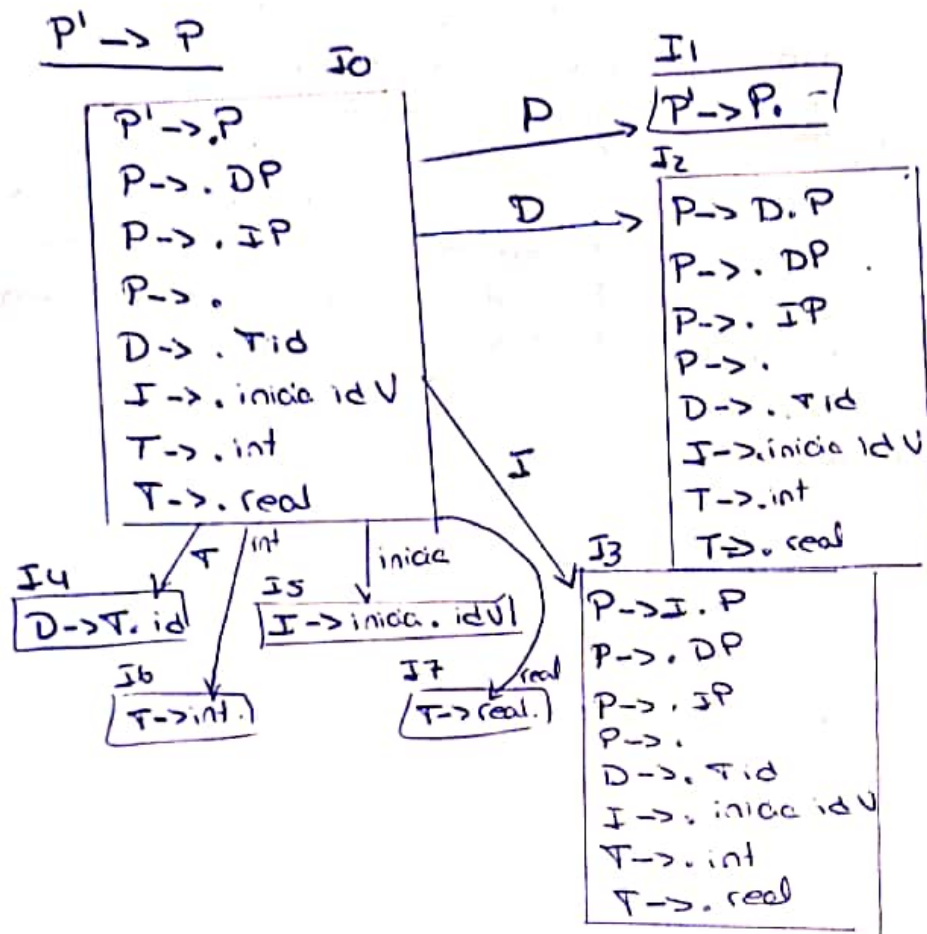
$$FIRST(D) = \{int, real\}$$

$$FIRST(I) = \{inicia\}$$

$$FOLLOW(P) = \{\$ \}$$

$$FIRST(V) = \{int, real\}$$

© $P \rightarrow DP \mid IP \mid \lambda$
 $D \rightarrow T \mid id$
 $T \rightarrow int \mid real$
 $I \rightarrow inicia \ id \ V$
 $V \rightarrow cle-real \ V \mid cle-int \ V \mid id$



Los estados en los que puede haber conflicto son I0, I2, I3 por tener reducción $P \rightarrow \lambda$ y varios desplazamientos. Los posibles conflictos serían para $Follow(P) = \{ \$ \}$ y como no hay desplazamiento posible para "\$", se deduce que no hay conflictos en esos estados.

d) Ninguno. Es una cuestión que debe resolver el An. Semántico

1

14 noviembre 2017

$F \rightarrow \text{fun } AC$

$A \rightarrow T \text{ id } A \mid \text{ref}$

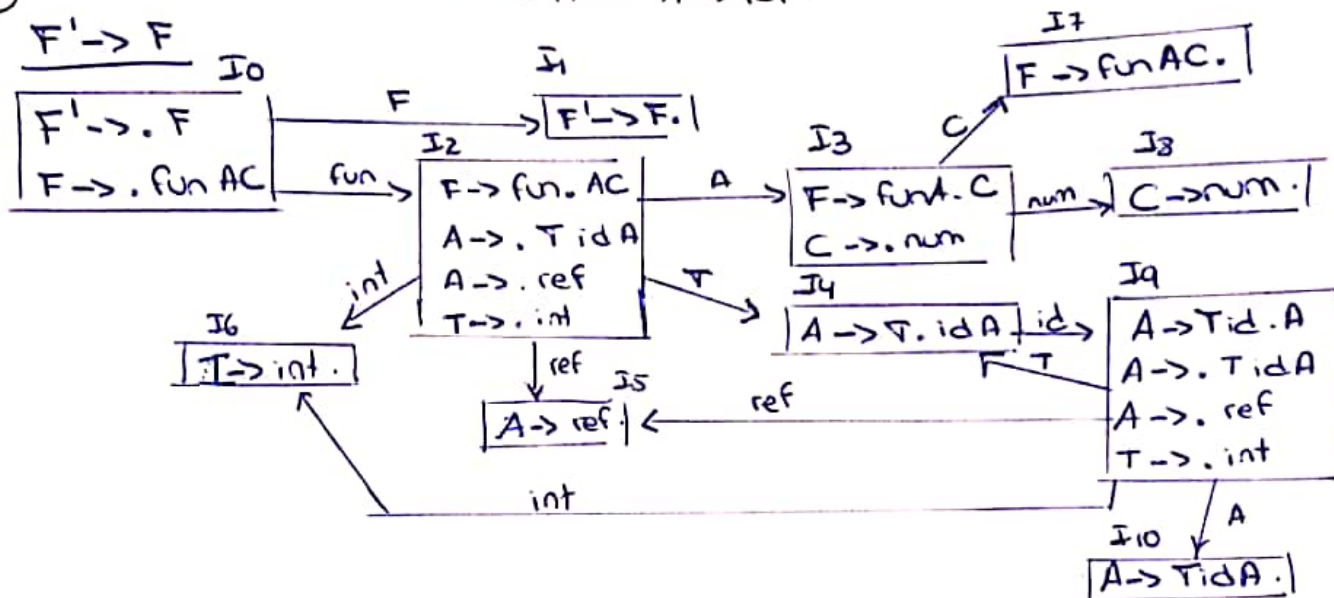
$T \rightarrow \text{int}$

$C \rightarrow \text{num}$

a) En relación con el método de Análisis Sintáctico Ascendente (LR(1)), construye el Autómata Reconocedor de prefijos viables.

b) Detalla los flujos de los estados LR de todos los estados que contienen al ítem $A \rightarrow T \text{ id } A$ o el ítem $A \rightarrow \text{ref}$.

a)



b)

$A \rightarrow T \text{id } A$

I9

I5

Acción

GOTO

	fun	id	ref	int	num	\$	F	A	T	C
I9			d5	d6				10	4	
I5			R3 $A \rightarrow \text{ref}$							

14 noviembre 2017

②

$F \rightarrow \text{Fun id A C val: T}$

$A \rightarrow T \text{ id A } | \text{ ref T id A } | \lambda$

$T \rightarrow \text{int} | \text{real}$

$C \rightarrow \text{id} = \text{num} | \lambda$

a) Calcular FIRST y FOLLOW

b) Compruebo si los reglas de A cumplen la condición LL1.

c) Diseñar el procedimiento del método Descendente Recursivo correspondiente al símbolo A. Se puede usar el método EGT(t): equipara token

a)

$\text{FIRST}(F) = \{ \text{fun} \}$

$\text{FIRST}(A) = \{ \text{int}, \text{real}, \text{ref}, \lambda \}$

$\text{FIRST}(T) = \{ \text{int}, \text{real} \}$

$\text{FIRST}(C) = \{ \text{id}, \lambda \}$

$\text{FOLLOW}(F) = \{ \$ \}$

$\text{FOLLOW}(A) = \{ \text{id}, \text{val} \}$

$\text{FOLLOW}(T) = \{ \text{id}, \$ \}$

$\text{FOLLOW}(C) = \{ \text{val} \}$

$\rightarrow \text{FIRST}(C)$, como tiene λ pasa al siguiente en la regla que es $\text{FIRST}(\text{val}) = \text{val}$

b) Reglas de A

$\text{FIRST}(T \text{ id A}) \cap \text{FIRST}(\text{ref T id A}) = \{ \text{int}, \text{real} \} \cap \{ \text{ref} \} = \emptyset$

$\text{FIRST}(T \text{ id A}) \cap \text{FOLLOW}(A) = \{ \text{int}, \text{real} \} \cap \{ \text{id}, \text{val} \} = \emptyset$

$\text{FIRST}(\text{ref T id A}) \cap \text{FOLLOW}(A) = \{ \text{ref} \} \cap \{ \text{id}, \text{val} \} = \emptyset$

Reglas de C

$\text{FIRST}(\text{int}) \cap \text{FIRST}(\text{real}) = \{ \text{int} \} \cap \{ \text{real} \} = \emptyset$

$\text{FIRST}(\text{id} = \text{num}) \cap \text{FOLLOW}(C) = \{ \text{id} \} \cap \{ \text{val} \} = \emptyset$

• la intersección de los FIRST de los elementos no terminales con los de una regla es \emptyset

• la gramática no es recursiva por lo 129

• la gramática SI está factorizada

$\rightarrow \text{ES LL(1)}$

c)

Function A()

if (sig_token == 'int' || sig_token == 'real') {

T();

EGT(id);

A();

} else if (sig_token == 'ref') {

EGT(ref);

T();

EGT(id);

A();

} else if (sig_token == 'id' || sig_token == 'val') {

↓

else {

error();

↓

↓

d) Señala en la tabla los errores u omisiones presentes (método de Análisis Sintáctico Descendente LCA).

	fun	val	:	id	ref	int	real	
F	$F \rightarrow \text{fun id A val : T}$							\$, no
A		$A \rightarrow \lambda$		$A \rightarrow \lambda$	$A \rightarrow \text{ref T id A}$	$A \rightarrow \text{T id A}$	$A \rightarrow \text{T id A}$	NO
T								$A \rightarrow \lambda$
C		$C \rightarrow \lambda$		$A \rightarrow \text{ref T id A}$	$T \rightarrow \text{int}$	$T \rightarrow \text{real}$		NO
			$C \rightarrow \text{id = num}$		NO			$E \rightarrow \lambda$

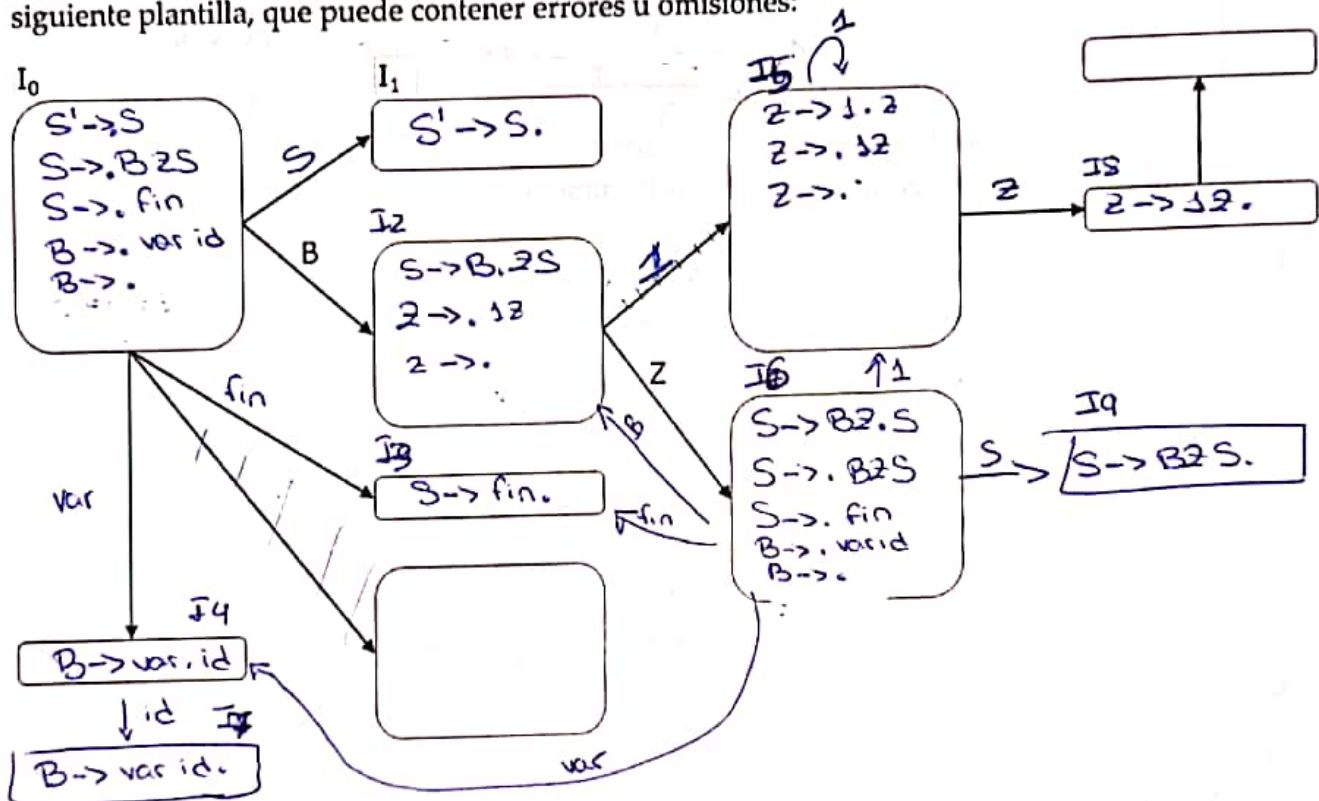
PROCESADORES DE LENGUAJES

Plantilla de Respuesta - Análisis Sintáctico. 27 de junio de 2018

a. En relación con el método de Análisis Sintáctico Ascendente (LR), dada la gramática:

$$\begin{array}{l|l} S \rightarrow B Z S & \text{fin} \\ B \rightarrow \text{var id} & \lambda \\ Z \rightarrow 1 Z & \lambda \end{array}$$

Se pide construir el **Autómata Reconocedor de Prefijos Viabiles** para esta gramática, usando la siguiente plantilla, que puede contener errores u omisiones:



b. Se pide realizar el análisis de todos los posibles conflictos en el Autómata resultante del apartado a.

$\text{FOLLOW}(B) = \{ 1, \text{var}, \text{fin} \}$

$\text{FOLLOW}(Z) = \{ \text{var}, \text{fin} \}$

I0: Reducción - desplazamiento: reducción por $\text{FOLLOW}(B)$ y desplazamiento por $\{ \text{var}, \text{fin} \} \rightarrow$ Hay conflicto

I2: Reducción - desplazamiento: reducción por $\text{FOLLOW}(Z)$ y desplazamiento por $\{ 1 \} \rightarrow$ Hay conflicto

I6: Reducción - desplazamiento: reducción por $\text{FOLLOW}(B)$ y desplazamiento por $\{ \text{var}, \text{fin} \} \rightarrow$ Hay conflicto

I5: Reducción - desplazamiento: reducción por $\text{FOLLOW}(Z)$ y desplazamiento por $\{ 1 \} \rightarrow$ Hay conflicto

c. En relación con el método de Análisis Sintáctico Descendente LL(1), dada la gramática:

$A \rightarrow 1 B \mid 2 C \mid D C$
 $B \rightarrow 1 C$
 $C \rightarrow 3 D \mid 4 C$
 $D \rightarrow 5 A \mid 6 \mid \lambda$

$FIRST(A) = \{1, 2, 5, 6, 3, 4\}$
 $FIRST(B) = \{1\}$
 $FIRST(C) = \{3, 4\}$
 $FIRST(D) = \{5, 6, \lambda\}$
 $FOLLOW(D) = \{3, 4, \$\}$

Se pide corregir los errores u omisiones de la siguiente tabla LL(1):

	1	2	3	4	5	6	\$	λ
A	$A \rightarrow 1 B$	$A \rightarrow 2 C$	$A \rightarrow D C$	$A \rightarrow D C$	$A \rightarrow D C$	$A \rightarrow D C$	$A \rightarrow D C$	
B	$B \rightarrow 1 C$							
C			$C \rightarrow 3 D$	$C \rightarrow 4 C$				
D			$D \rightarrow \lambda$	$D \rightarrow \lambda$	$D \rightarrow 5 A$	$D \rightarrow 6$	$D \rightarrow \lambda$	$D \rightarrow \lambda$

$FIRST(C)$

$FOLLOW(C)$

d. Se pide escribir los procedimientos correspondientes a los símbolos B y D pertenecientes a un Analizador Sintáctico Descendente Predictivo Recursivo, dada la gramática del apartado c.

Function B()

```

if (sig_token == '1') {
    equipara(1);
    C();
}
else {
    error();
}

```

Function D()

```

if (sig_token == '5') {
    equipara(5);
    A();
}
else if (sig_token == '6') {
    equipara(6);
}
else if (sig_token == '3' || sig_token == '4' || sig_token == '4') {
}
else {
    error();
}

```

PROCEDURE B()

BEGIN

IF sig_token = "1" THEN
equipara(1)

C

ELSE THEN
error

END B

10 JULIO 2013

2. Sea la siguiente gramática G:

$P \rightarrow D S$
 $D \rightarrow T : id ; D \mid \lambda$
 $T \rightarrow real \mid int$
 $S \rightarrow if id then S else S ; S \mid forEach id in id S ; S \mid \lambda$
 $E \rightarrow id \mid id [E]$

Se pide:

- Construir una gramática G' equivalente sin recursividad a izquierdas y factorizada.
- Calcular los conjuntos *First* y *Follow* de todos los símbolos no terminales de G' .
- Construir la tabla de un Analizador Sintáctico LL(1) para G' y justificar si la gramática G' es LL(1).
- Diseñar los procedimientos del Analizador Sintáctico Descendente Recursivo correspondientes a los símbolos D y S (puede utilizarse un procedimiento auxiliar para equiparar *tokens*).

10 julio 2013

a) la gramática NO es recursiva por la izquierda

Si hay que factorizar por la regla $E \rightarrow id | id Z Z$

G':

- $P \rightarrow DS$
- $D \rightarrow T: id ; D$
- $D \rightarrow \lambda$
- $T \rightarrow real$
- $T \rightarrow int$
- $S \rightarrow if id then S else S ; S$
- $S \rightarrow foreach id in id S ; S$
- $S \rightarrow \lambda$
- $E \rightarrow id F$
- $F \rightarrow [E]$
- $F \rightarrow \lambda$

b)

- $FIRST(P) = \{real, int, if, foreach, \lambda\}$
- $FIRST(D) = \{real, int, \lambda\}$
- $FIRST(T) = \{real, int\}$
- $FIRST(S) = \{if, foreach, \lambda\}$
- $FIRST(D) = \{real, int, \lambda\}$
- $FIRST(T) = \{real, int\}$
- $FIRST(S) = \{if, foreach, \lambda\}$
- $FIRST(E) = \{id\}$
- $FIRST(F) = \{[, \lambda\}$
- $FIRST(T) = \{real, int\}$

• $FOLLOW(P) = \{ \$ \}$

• $FOLLOW(D) = \{ if, foreach, \$ \}$

$FIRST(S) = \{ if, foreach, \lambda \}$

ej. $FIRST(S)$ tiene λ así que voy al siguiente

$P \rightarrow DS$ no hay más, así que hago $FOLLOW(P) = \{ \$ \}$

• $FOLLOW(T) = \{ ; \}$

• $FOLLOW(S) = \{ \$, else, ; \}$

• $FOLLOW(E) = \{] \}$

• $FOLLOW(F) = \{] \}$

$FOLLOW(E) = \{] \}$

c) Para construir la tabla necesitamos los FIRST del consecuente entero

$$\text{FIRST}(DS) = \{ \text{real}, \text{int}, \text{if}, \text{forEach}, \lambda \}$$

$$\text{FOLLOW}(P) = \{ \$ \}$$

$$\text{FIRST}(T \text{ id} ; D) = \{ \text{real}, \text{int} \}$$

$$\text{FOLLOW}(D) = \{ ;, \text{forEach}, \$ \}$$

$$\text{FIRST}(\text{real}) = \{ \text{real} \}$$

$$\text{FIRST}(\text{int}) = \{ \text{int} \}$$

$$\text{FIRST}(\text{if id then S else S} ; S) = \{ \text{if} \}$$

$$\text{FIRST}(\text{forEach id in id S} ; S) = \{ \text{forEach} \}$$

$$\text{FOLLOW}(S) = \{ \$, \text{else}, ; \}$$

$$\text{FIRST}(\text{id F}) = \{ \text{id} \}$$

$$\text{FIRST}(\text{E}) = \{ \text{E} \}$$

$$\text{FOLLOW}(F) = \{ ; \}$$

	id	;	:	real	int	if	then	else	forEach	in	E	}	\$
P				P → DS	P → DS	P → DS			P → DS				P → E
D				D → T id ; D	D → T id ; D	D → λ			D → λ				D →
T				T → real	T → real								
S			S → λ			S → if id then S else ; S		S → λ	S → forEach id in id S ; S				S →
E	E → id F												
F												F → [E]	F → ;

⊛ Tener columnas vacías no es error

⊛ Caso de error es cuando accedemos a la columna vacía

⊛ Fila vacía → no hay regla para ese no terminal

⊛ Si una columna queda en blanco es porque ese terminal no está en el FIRST de ningún consecuente

Function S() {

```
if (sig_tok == 'if') {  
    print(6);  
    equipara (if);  
    equipara (id);  
    equipara (then);  
    S;  
    equipara (else);  
    S;  
    equipara (; )  
    S;  
}
```

```
else if (sig_tok == 'foreach') {  
    print(7);  
    equipara (foreach);  
    equipara (id);  
    equipara (in);  
    equipara (id);  
    S;  
    equipara (; );  
    S;  
}
```

```
else if (sig_tok == '$' || sig_tok == 'else' || sig_tok == ';') {  
    print(8);  
}
```

```
else {  
    error ();  
}
```

Function D() {

```
if (sig_tok == real || sig_tok == int) {  
    print(2);  
    T;  
    equipara (l);  
    equipara (id);  
    equipara (; );  
    D;  
}
```

```
else if (sig_tok == if || sig_tok == $ ||  
sig_tok == foreach) {  
    print(3);  
}
```

```
else {  
    error ();  
}
```

ANÁLISIS SINTÁCTICO

23 de noviembre de 2016

Observaciones: 1. Las calificaciones se publicarán hacia el 14 de diciembre.
2. La revisión será hacia el 16 de diciembre.
3. En la web se avisará de las fechas exactas.
4. La duración de este examen es de 40 minutos.

Dada la Gramática G_1 :

$S \rightarrow A + B ; C$
 $A \rightarrow a A \mid C \mid ; C$
 $C \rightarrow (A) \mid \lambda$
 $B \rightarrow b A b$

Se pide:

- a. Comprobar la Condición LL(1) para la gramática G_1 .
- b. Construir la tabla completa del Analizador Sintáctico Descendente LL(1) para la gramática G_1 .

Dada la Gramática G_2 :

$S \rightarrow A + B ; C$
 $A \rightarrow a A$
 $A \rightarrow C$
 $B \rightarrow a A$
 $B \rightarrow b$
 $C \rightarrow a A B b$

Se pide:

- c. Para la gramática G_2 , construir el estado I_0 del Autómata Reconocedor de Prefijos Viables de un Analizador Sintáctico LR, más todos los estados que parten directamente de él.
- d. Escribir la fila correspondiente al estado I_0 de las tablas ACCIÓN y GOTO de un analizador LR para la gramática G_2 .
- e. Para el fragmento de Autómata construido en el apartado c, indicar qué estados contienen un ítem que indica reducción, por qué regla se produciría la reducción y para qué tokens de la entrada se realizaría dicha reducción.

$S \rightarrow A + B ; C$
 $A \rightarrow a A \mid C \mid ; C$
 $C \rightarrow (A) \mid \lambda$
 $B \rightarrow b A b$

a) Reglas de A

$$\text{FIRST}(aA) \cap \text{FIRST}(C) = \{a\} \cap \{(\lambda) = \emptyset$$

$$\text{FIRST}(aA) \cap \text{FIRST}(;C) = \{a\} \cap \{; \} = \emptyset$$

$$\text{FIRST}(C) \cap \text{FIRST}(;C) = \{(\lambda) \cap \{; \} = \emptyset$$

Reglas de C

$$\text{FIRST}((A)) \cap \text{FOLLOW}(C) = \{(\cap \{b, \lambda, +, b\} = \emptyset$$

• Como todas las intersecciones son disjuntas, la gramática es LLI

• Gramática NO recursiva por la izquierda

• Gramática factorizada

b)

	+	;	a	()	b	\$
S	$S \rightarrow A+B;C$	$S \rightarrow A+B;C$	$S \rightarrow A+B;C$	$S \rightarrow A+B;C$			
A	$A \rightarrow C$	$A \rightarrow ;C$	$A \rightarrow aA$	$A \rightarrow C$	$A \rightarrow C$	$A \rightarrow C$	$A \rightarrow C$
C	$C \rightarrow \lambda$			$C \rightarrow (A)$	$C \rightarrow \lambda$	$C \rightarrow \lambda$	$C \rightarrow \lambda$
B						$B \rightarrow bAb$	

$$\text{FIRST}(S) = \{a, ;, (, +\}$$

$$\text{FIRST}(A) = \{a, ;, (\lambda) \rightarrow \text{FOLLOW}(A) = \{+,), b\}$$

$$\text{FIRST}(C) = \{(, \lambda) \rightarrow \text{FOLLOW}(C) = \{\$, +, b,)\}$$

$$\text{FIRST}(B) = \{b\}$$

$$A \rightarrow aA \rightarrow a$$

$$A \rightarrow C \rightarrow \underbrace{(\,}_{\text{FIRST}(C)} \underbrace{(\$, +,), b)}_{\text{FOLLOW}(C)}$$

$$C \rightarrow (A) \rightarrow ($$

$$C \rightarrow \lambda \rightarrow \text{FOLLOW}(C)$$

$S \rightarrow A + B ; C$

c)

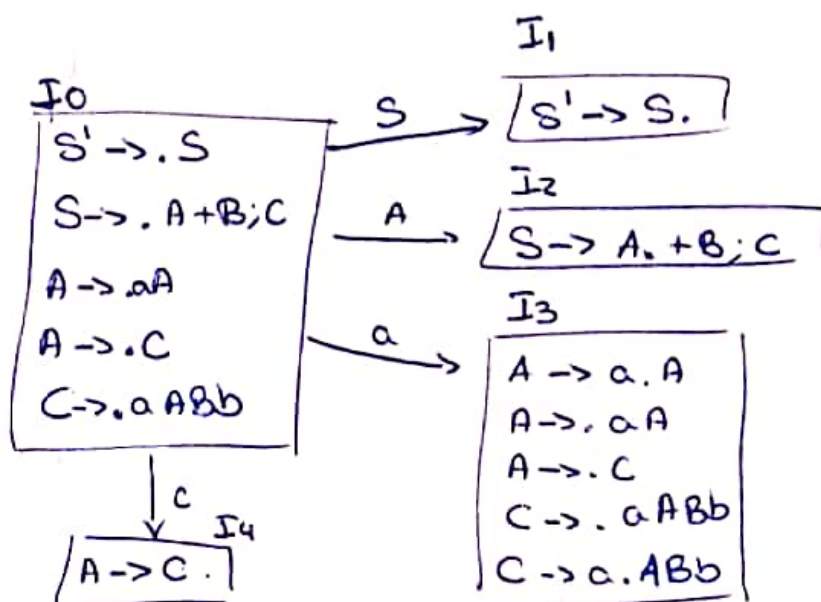
$A \rightarrow o A$

$A \rightarrow C$

$B \rightarrow a A$

$B \rightarrow b$

$C \rightarrow a A B b$



b)

	Accion					GOTO		
	+	;	a	b	\$	S	A	C
0			d3			1	2	4

c)

• Habría una reducción en el estado 4, por la regla $A \rightarrow C$ para los tokens pertenecientes al $\text{Follow}(A) = \{ \underset{\text{F(A)}}{+}, \underset{\text{F(A)}}{b}, \overset{\text{FIRST(B)}}{a}, \underset{\text{Follow(B)}}{;} \}$

• Aunque en el estado 3 también hay un punto en la última posición del consecuente, es el ítem con la que corresponde a la regla con la que se genera la gramática ($S' \rightarrow S$) y, por tanto, da lugar a una acción ACEPTAR y no a una reducción