examenes-an.pdf

Anónimo

Procesadores de Lenguajes

4º Doble Grado en Ingeniería Informática y Administración y Dirección de Empresas

Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

12 enero 2017

Gramática que no se puede modificar:

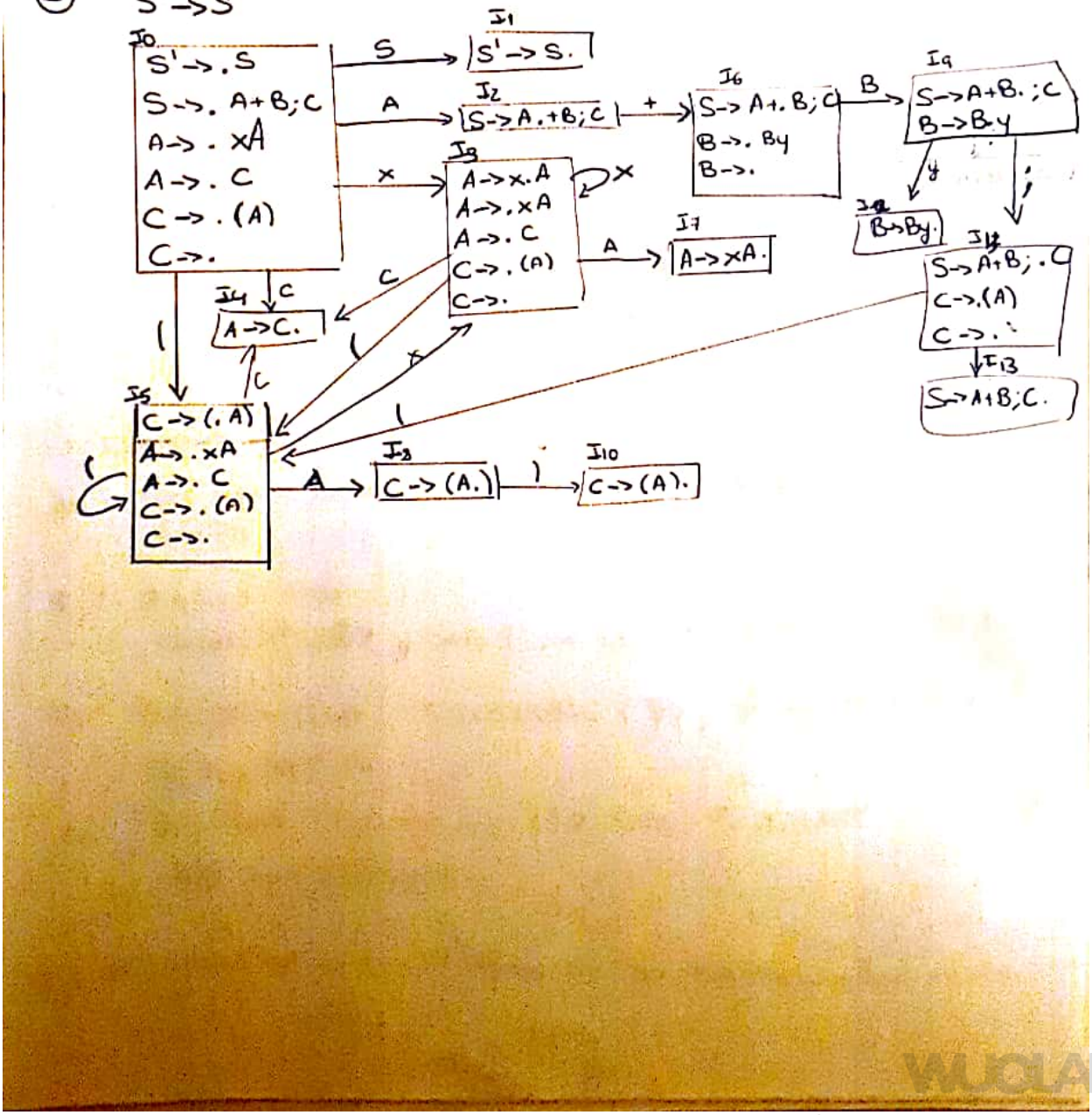
S -> A+B;C
A -> xA | C
B -> By | λ
C -> (A) | λ

a) Determinar razonadamente qué tipo de analizador sintáctico puede ser válido para esta gramática.

b) Construir la tabla de dicho Analizador Sintáctico

(a) 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.
↳ B -> By

(b) S' -> S



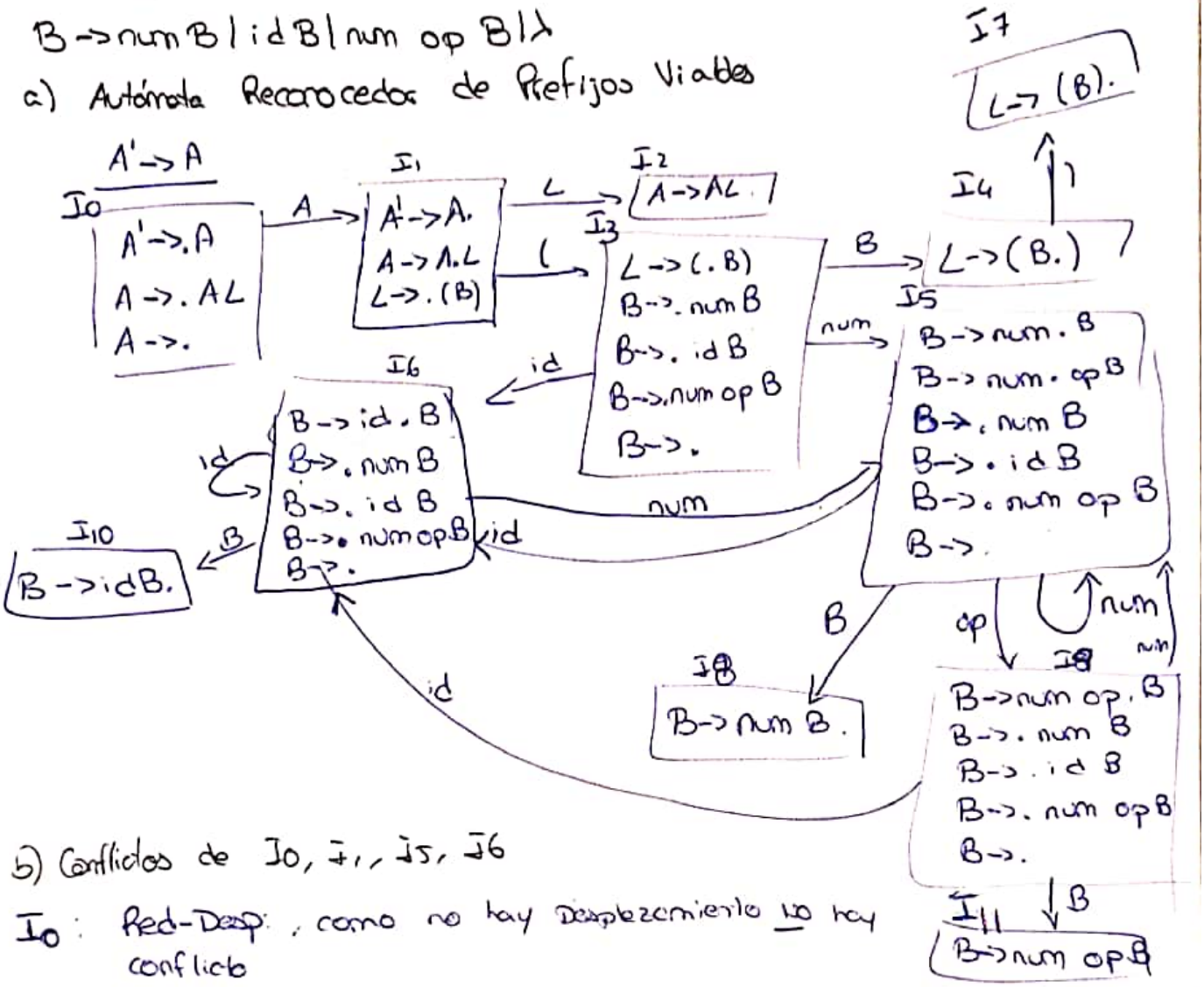
13 diciembre 2010

A -> AL | λ

L -> (B)

B -> num B | id B | num op B | λ

a) Autómata Reconocedor de Prefijos Viables



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:

Follow(A') = {$} y Desp "(" → NO CONFLICTO

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

NO hay conflicto

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

NO hay conflicto

I_0. Rec - Red: no hay conflicto xq no hay reducción - ocerrón de A ->.

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

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

$A \to AL \mid \lambda$
$L \to (B)$
$B \to num\ B \mid id\ B \mid num\ op\ B \mid \lambda$

- Recursiva por la izq (A)
- No factorizada (B)

$A \to LA$
$A \to \lambda$
$L \to (B)$
$B \to num\ C$
$B \to id\ B$
$B \to \lambda$
$C \to B$
$C \to op\ B$

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

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

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

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

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

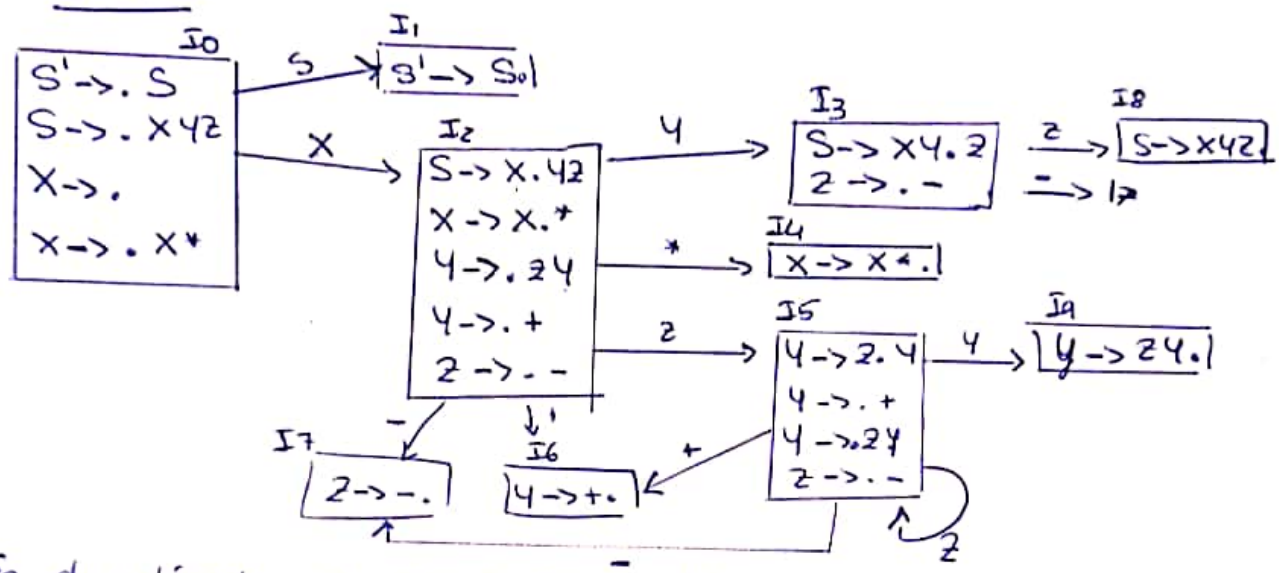
La gramática es LL(1).

10 enero 2018

S -> X4Z
X -> λ | X*
4 -> Z4 | +
Z -> -

a) Construir autómata reconocedor de Prefijos Viables 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. Sintáctico Descendente para la gramática LL

(a) S' -> S



(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 -> X*. Cambiamos las reglas de X para eliminar la recursividad: X -> λ | * X (reglas equivalentes)

Para las reglas de X

FIRST(λ) = FOLLOW(X) = {+, -4
FIRST(*X) = {*4
FOLLOW(X) ∩ FIRST(*X) = {+, -4 ∩ {*4 = ∅

Para las reglas de 4

FIRST(Z4) ∩ FIRST(+) = {-4 ∩ {+4 = ∅

FIRST(S) = {*, +4
FIRST(*X) = {*4
FOLLOW(X) = {+, -4

(d)

S -> X4Z
X -> λ | *X
4 -> Z4 | +
Z -> -

	+	-	*	$
S	S->X4Z	S->X4Z	S->X4Z	
X	X->λ	X->λ	X->*X	
4	4->+	4->Z4		
Z		Z->-		

Febrero 2008

$S \to OAAO \mid 1A1 \mid \lambda$
$A \to BIC$
$B \to 01 \mid 10$
$C \to 00 \mid 11$

a) Comprobar condiciones LL(1). Si no es LL(1) transformar la gramática para que lo sea
b) Construir An. Sintáctico Descendente Predictivo Recursivo

a) $FIRST(S) = \{0, 1, \lambda\}$
$FOLLOW(S) = \{\$\}$
$FIRST(A) = \{0, 1\}$
$FOLLOW(A) = \{1\}$
$FIRST(B) = \{0, 1\}$
$FOLLOW(B) = \{1\}$
$FIRST(C) = \{0, 1\}$
$FOLLOW(C) = \{1\}$

(1) • $FIRST(B) \cap FIRST(C) = \{0,1\} \cap \{0,1\} \neq \emptyset$
La gramática NO es LL(1) porque la intersección de las reglas de A no da el conjunto vacío.

(2) • Si se sustituyen los NO terminales B y C por sus reglas, quedaría la gramática:
$S \to OAAO \mid 1A1 \mid \lambda$
$A \to 01 \mid 10 \mid 00 \mid 11$

(3) • Factorizar la nueva gramática:
1 $S \to OAAO$
2 $S \to 1A1$
3 $S \to \lambda$
4 $A \to OD$
5 $A \to 1D$
6 $D \to 0$
7 $D \to 1$

(4) • Comprobar que la nueva gramática es LL(1)

- Reglas de S
$FIRST(OAAO) = \{0\}$
$FIRST(1A1) = \{1\}$
$FOLLOW(S) = \{\$\}$

- Reglas de A:
$FIRST(OD) = \{0\}$
$FIRST(1D) = \{1\}$

- Reglas de D
$FIRST(0) = \{0\}$
$FIRST(1) = \{1\}$

b)
```
Function S(){
  if(token=='0'){
    print(1);
    equipara('0');
    A();
    A();
    equipara('0');
  }else if(token=='1'){
    print(2);
    equipara('1');
    A();
    equipara('1');
  }else if(token=='1');
    print(3);
  }else{
    error();
  }
}
```

```
Function A(){
  if(token=='0'){
    print(4);
    equipara('0');
    D();
  }else if(token=='1'){
    print(5);
    equipara('1');
    D();
  }else{
    error();
  }
}
```

```
Function D(){
  if(token=='0'){
    print(6);
    equipara('0');
  }else if(token=='1'){
    print(7);
    equipara('1');
  }else{
    error();
  }
}
```

30 junio 2017

P -> DP | IP | λ

D -> T id

T -> int | real

I -> inicia id V

V -> cte-real V | cte-int V | id

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

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

c) Construir el estado inicial del Autómata reconocedor de prefijos viables de un Analizador Sintáctico Ascendente LR, así como todos los demás estados que derivan 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 sentencia 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) ∩ FIRST(IP) = {int, real} ∩ {inicia} = ∅

FIRST(DP) ∩ FOLLOW(P) = {int, real} ∩ {$} = ∅

FIRST(IP) ∩ FOLLOW(P) = {inicia} ∩ {$} = ∅

Reglas de T

FIRST(int) ∩ FIRST(real) = {int} ∩ {real} = ∅

Reglas de V

FIRST(cte-real V) ∩ FIRST(cte-int V) = {cte-real} ∩ {cte-int} = ∅

FIRST(cte-real V) ∩ FIRST(id) = {cte-real} ∩ {id} = ∅

FIRST(cte-int V) ∩ FIRST(id) = {cte-int} ∩ {id} = ∅

b)

	id	int	real	inicia	cte-real	cte-int	$
P		P->DP	P->DP	P->IP			P->λ
D		D->Tid	D->Tid				
T		T->int	T->real				
I				I->inicia idV			
V	V->id				V->cte-real V	V->cte-int V	

FIRST(D) = {int, real}

FIRST(I) = {inicia}

FOLLOW(P) = {$}

FIRST(T) = {int, real}

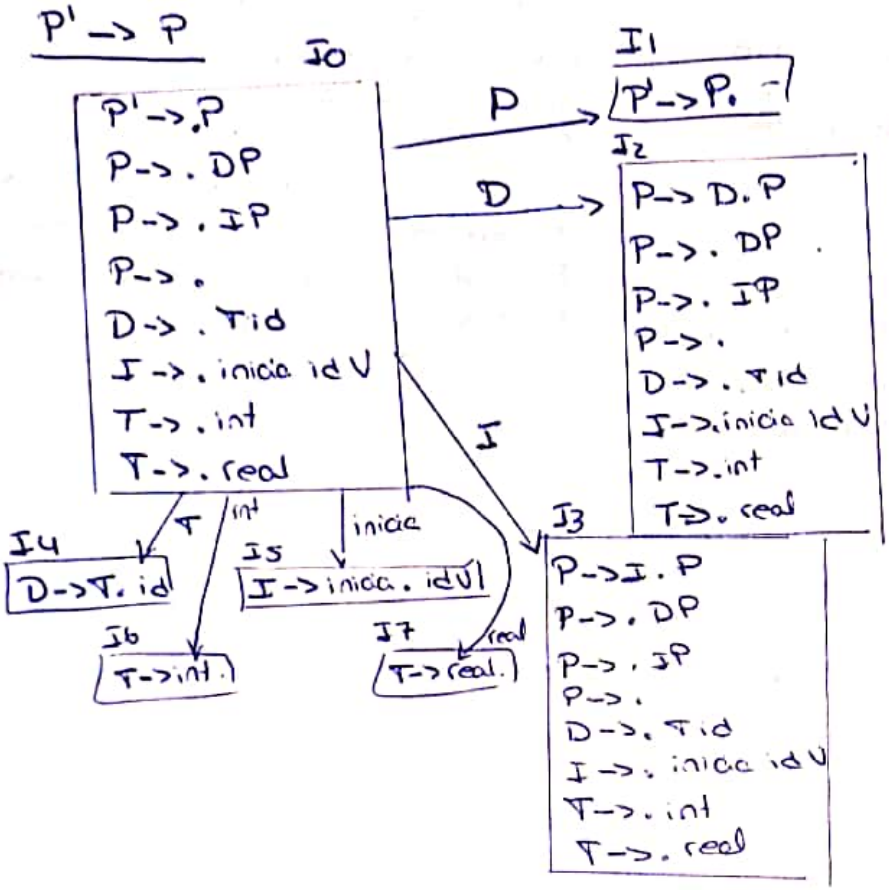
© P -> DP | IP | λ

D -> T id

T -> int | real

I -> inicia id V

V -> cte-real V | cte-int V | id



Los estados en los que puede haber conflicto son I0, I2, I3 por tener reducción P -> λ 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

14 noviembre 2017

F -> fun AC

A -> T id A | ref

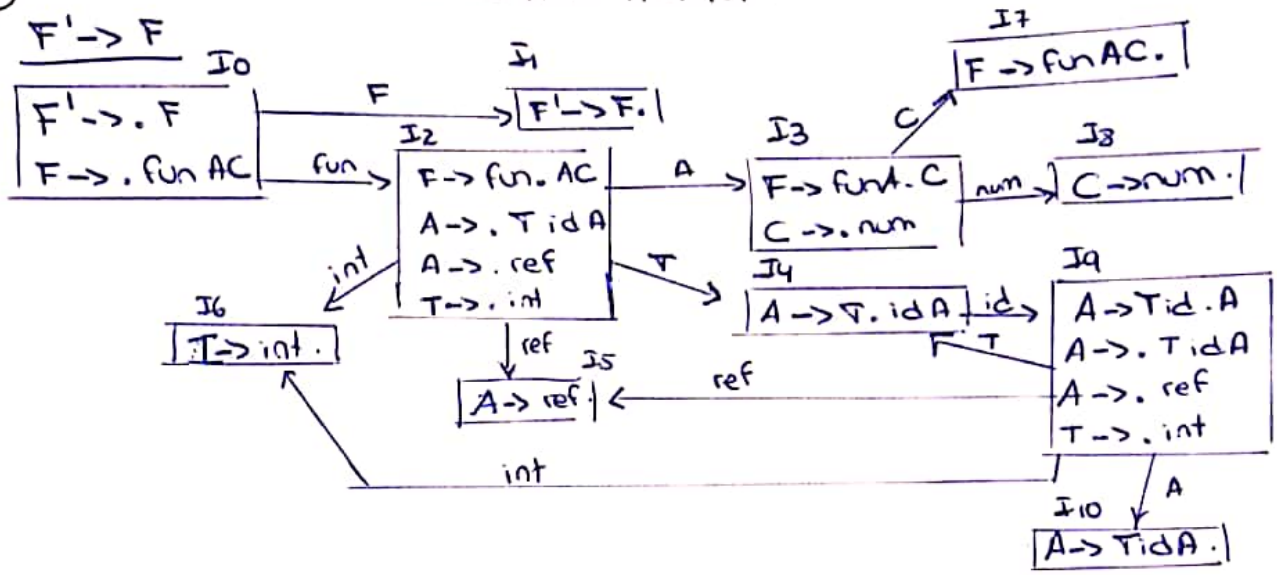
T -> int

C -> num

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

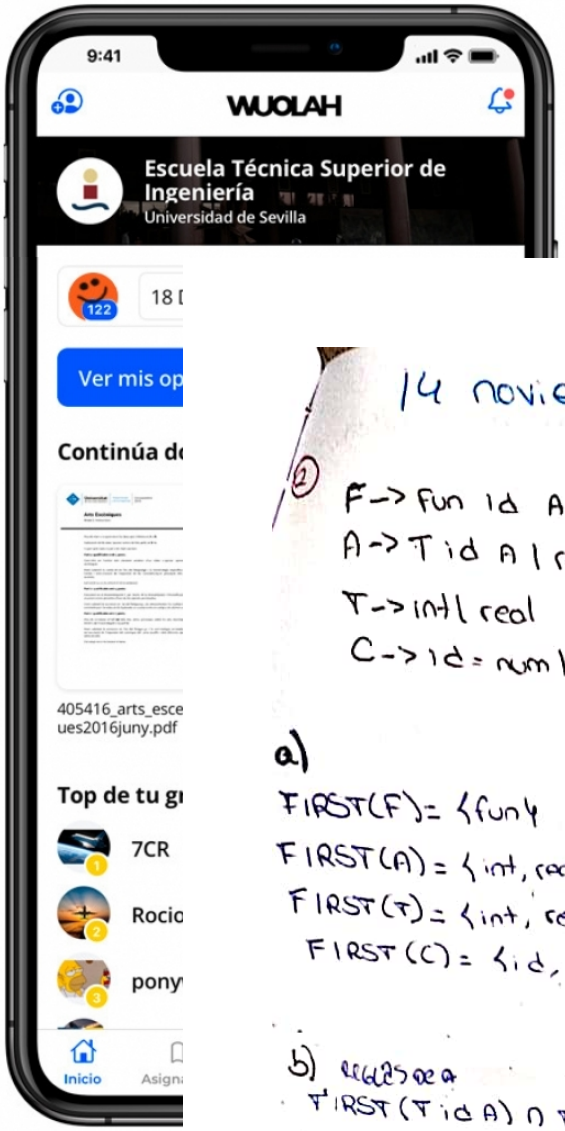
b) Detalla las filas de la tabla LR de todos los estados que contienen al ítem A -> Tid•A o el ítem A -> ref•.

(a)



(b)

	Acción						GOTO			
A->T id A	fun	id	ref	int	num	$	F	A	T	C
I9			d5	d6				10	4	
I5			R3 A->ref							



14 noviembre 2017

② F -> Fun id A C val : T
A -> T id A | ref T id A | λ
T -> int | real
C -> id = num | λ

a) Calcula FIRST y FOLLOW
b) Comprueba si las reglas de A cumplen la condición LL1.
c) Diseña el procedimiento del método Descendente Recursivo correspondiente al símbolo A. Se puede usar el método EqT(t) : equipara token

a)
FIRST(F) = {fun}
FIRST(A) = {int, real, ref, λ}
FIRST(T) = {int, real}
FIRST(C) = {id, λ}

FOLLOW(F) = {$}
FOLLOW(A) = {id} val (FIRST(C) como tiene λ pasa al siguiente en la 2ª regla que es FIRST(val) = val)
FOLLOW(T) = {id, $}
FOLLOW(C) = {val}

b) Reglas de A
FIRST(T id A) ∩ FIRST(ref T id A) = {int, real} ∩ {ref} = ∅
FIRST(T id A) ∩ FOLLOW(A) = {int, real} ∩ {id, val} = ∅
FIRST(ref T id A) ∩ FOLLOW(A) = {ref} ∩ {id, val} = ∅
Reglas de C
FIRST(int) ∩ FIRST(real) = {int} ∩ {real} = ∅
FIRST(id = num) ∩ FOLLOW(C) = {id} ∩ {val} = ∅
- La intersección de los FIRST de los elementos NO terminales con más de una regla es ∅
- La gramática NO es recursiva por la izq
- La gramática SÍ está factorizada

-> ES LL(1)

c)
```
Function A(){
  if (sig_token == 'int' || sig_token == 'real'){
      T();
      EqT(id);
      A();
  } else if (sig_token == 'ref') {
      EqT(ref);
      T();
      EqT(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 LL(1)).

	fun	val	:	id	ref	int	real	$, no_t
F	F->fun id A val:T							$
A		A->λ		A->λ	A->ref T id A	A->T id A	A->T id A	NO A->λ
T								
C		C->λ			A->ref T id A	T->int	T->real	NO ε->λ
				C->id=num	NO			

APELLIDOS:.....**NOMBRE:**.....

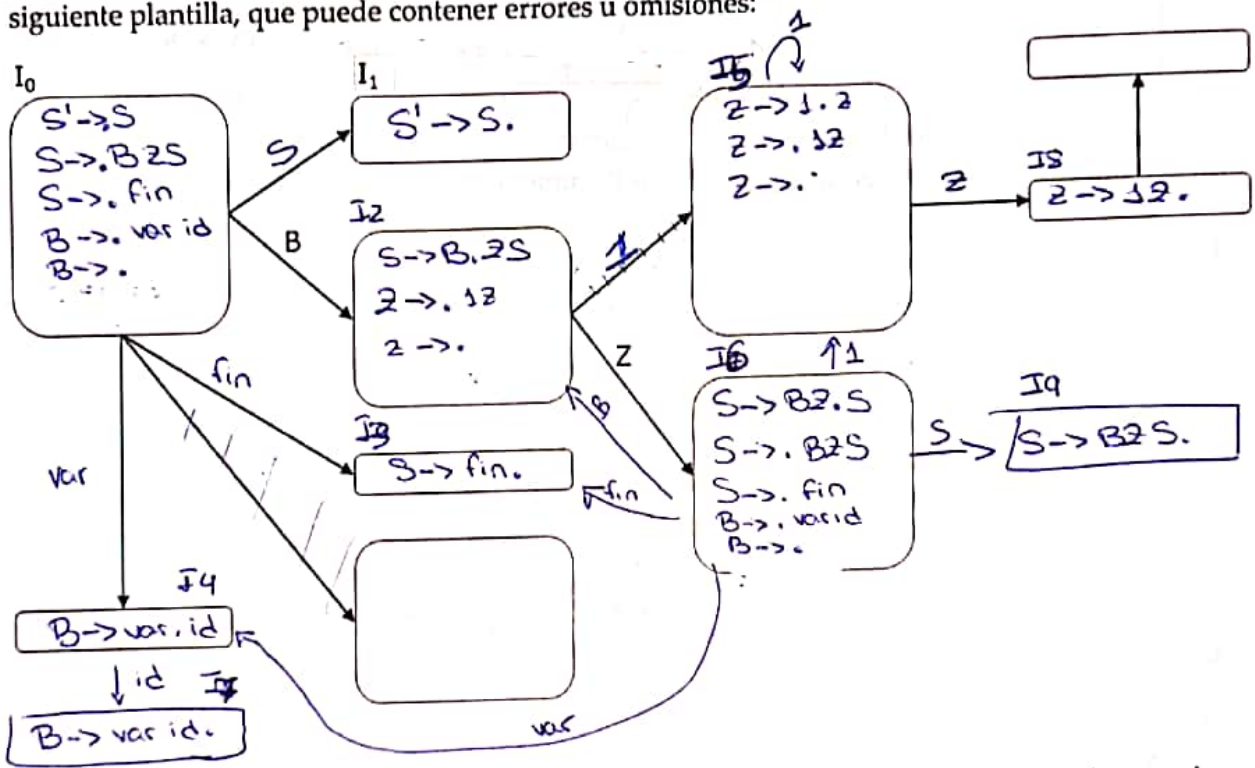
PROCESADORES DE LENGUAJES

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

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

S → B Z S | fin
B → var id | λ
Z → 1 Z | λ

Se pide construir el **Autómata Reconocedor de Prefijos Viables** 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.**

FOLLOW(B) = { 1, var, fin } FOLLOW(Z) = { var, fin }

I_0: Reducción - desplazamiento: reducción por FOLLOW(B) y desplazamiento por {var}, {fin} → Hay conflicto

I_2: Reducción - desplazamiento: reducción por FOLLOW(Z) y desplazamiento por {1} → Hay conflicto

I_6: Reducción - desplazamiento: reducción por FOLLOW(B) y desplazamiento por {var, fin} → Hay conflicto

I_5: Reducción - desplazamiento: reducción por FOLLOW(Z) y desplazamiento por {1} → Hay conflicto

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

A → 1 B | 2 C | D C
B → 1 C
C → 3 D | 4 C
D → 5 A | 6 | λ

FIRST(A) = {1, 2, 5, 6, 3, 4}
FIRST(B) = {1}
FIRST(C) = {3, 4}
FIRST(D) = {5, 6, λ}
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→ 1 B	A→ 2 C	A->DC	A->DC	A→ D C	A→ D C	A→ D C	
B	B→ 1 C							
C			C->3D	C->4C				
D			D->λ	D->λ	D→ 5 A	D→ 6	D-> λ	D→ λ

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 == '$'){

    }
    else {
        error();
    }
```

```
PROCEDURE B()
BEGIN
    IF sigle_token = "1" THEN
        equipara(1)
        C
    ELSE THEN
        ERROR
END B
```

10 JULIO 2013

2. Sea la siguiente gramática G:

P → D S
D → T : id ; D | λ
T → real | int
S → if id then S else S ; S | forEach id in id S ; S | λ
E → id | id [E]

Se pide:

a. Construir una gramática G' equivalente sin recursividad a izquierdas y factorizada.
b. Calcular los conjuntos *First* y *Follow* de todos los símbolos no terminales de G'.
c. Construir la tabla de un **Analizador Sintáctico LL(1)** para G' y justificar si la gramática G' es LL(1).
d. 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

Sí hay que factorizar por la regla $\bar{E} \to id \mid id[\bar{E}]$

G':

P -> DS

D -> T:id ; D

D -> λ

T -> real

T -> int

S -> if id then S else S; S

S -> foreach id in id S; S

S -> λ

$\bar{E}$ -> id F

F -> [E]

F -> λ

b)

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

- FOLLOW(P) = {$}
- FOLLOW(D) = {if, foreach, $}
  - FIRST(S) = {if, foreach, λ}

(* Ya FIRST(S) tiene λ, así que hay el siguiente P -> DS ? no hay nada, así que hago FOLLOW(P) = {$})

- FOLLOW(T) = {:}
- FOLLOW(S) = {$, else, ;}
- FOLLOW($\bar{E}$) = {]}
- FOLLOW(F) = {]}
  - FOLLOW($\bar{E}$) = {]}

c)

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

FIRST(DS) = { real, int, if, foreach, λ }

FOLLOW(P) = { $ }

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

FOLLOW(D) = { if, foreach, $ }

FIRST(real) = { real }

FIRST(int) = { int }

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

FIRST(foreach id in id S ; S) = { foreach }

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

FIRST(id F) = { id }

FIRST([E]) = { [}

FOLLOW(F) = {] }

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

① Tener columnas vacías NO es error

② Caso de error es querer acceder a la columna vacía

③ Fila vacía → no hay regla para ese no terminal.

④ 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(:);
    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 -> A + B ; C

A -> a A | C | ; C

C -> (A) | λ

B -> b A b

a) Reglas de A

FIRST(aA) ∩ FIRST(C) = {a} ∩ {(, λ} = ∅

FIRST(aA) ∩ FIRST(; C) = {a} ∩ {;} = ∅

FIRST(C) ∩ FIRST(; C) = {(, λ} ∩ {;} = ∅

Reglas de C

FIRST((A)) ∩ FOLLOW(C) = {(} ∩ {$,), +, b} = ∅ (FOLLOW(A))

- Como todas las intersecciones son disjuntas, la gramática es LL1
- Gramática NO recursiva por la izquierda
- Gramática factorizada

b)

	+	;	a	(	)	b	$
S	S->A+B;C	S->A+B;C	S->A+B;C	S->A+B;C			
A	A->C	A->;C	A->aA	A->C	A->C	A->C	A->C
C	C->λ			C->(A)	C->λ	C->λ	C->λ
B						B->bAb	

FIRST(S) = {a, ;, (, +}

FIRST(A) = {a, ;, (, λ} → FOLLOW(A) = {+,), b}

FIRST(C) = {(, λ} → FOLLOW(C) = {$, +, b,)}

FIRST(B) = {b}

A -> a A → a

A -> C → (, {$, +,), b}

FIRST(C) FOLLOW(C) (FOLLOW(A))

C -> (A) → (

C -> λ → FOLLOW(C)

$S \to A + B ; C$

$A \to a\,A$

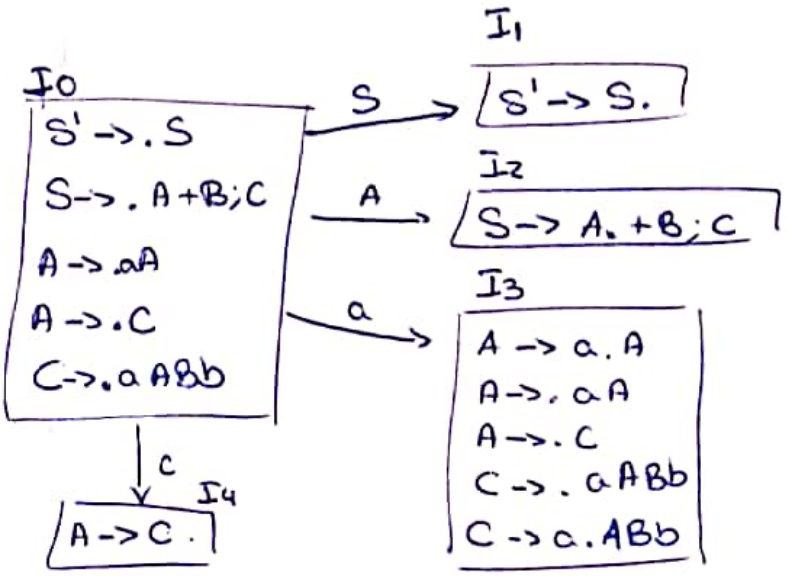
$A \to C$

$B \to a\,A$

$B \to b$

$C \to a\,A\,B\,b$

c)



b)

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

c)

- Habría una reducción en el estado 4, por la regla $A \to C$ para los cobens pertenecientes al $FOLLOW(A) = \{+, b, a, ;\}$ (a: FIRST(B); $+$: F(A); b: FIRST(B) y FOLLOW(B); $;$: FOLLOW(B))

- Aunque en el estado 1 también hay un punto en la última posición del consecuente, es el ítem que corresponde a la regla con la que se aumenta la gramática ($S' \to S$) y, por tanto, da lugar a una acción ACEPTAR y no a una reducción