Analizador Léxico – Analizador Sintáctico:

¿Cómo se comunican el Analizador Léxico y el Analizador Sintáctico?

El Analizador Sintáctico va pidiendo al Analizador Léxico los tokens uno a uno y el Léxico lee el fichero fuente para devolver el siguiente token.

Lenguaje sin declaración:

Para un lenguaje de programación sin declaración previa de variables, se ha creado el siguiente conjunto de atributos de la Tabla de Símbolos. ¿Es correcto (aunque no sea completo)?:

Lexema, Tipo, Declarada
Falso. Declarada no es.

Lexema, Tipo, Valor
Falso. Valor no es.

LexemaDeclarado, LexemaNoDeclarado, Tipo
Falso. Separar los lexemas de las variables declaradas de las que no lo han sido no aporta ninguna información que útil ni necesaria.

Lexema, TipoDeclarado, TipoNoDeclarado
Falso. No hay razón para separar los tipos de las variables declaradas de las que no lo han sido.

Nº de variables declaradas, Tipo, Dirección (o Desplazamiento)
Falso. Nº de variables declaradas no es.

Tabla de Símbolos:

¿Cuáles de las siguientes afirmaciones son ciertas en relación con la Tabla de Símbolos?

- El Analizador Léxico guarda el lexema de los identificadores en la Tabla de Símbolos.
  Explicación: Es el Analizador Semántico el que puede detectar una declaración de un identificador y, por tanto, guardar su tipo.

- Se utiliza para almacenar información acerca de los identificadores que aparecen en el programa fuente.
  Explicación: La Tabla de Símbolos contiene la información de los identificadores.

- El Analizador Semántico introduce el tipo de los identificadores en la Tabla de Símbolos.
  Explicación: Todo identificador tiene que tener el atributo tipo para saber el significado del identificador.

- Cada entrada de la Tabla de Símbolos puede tener atributos distintos.

- La Tabla de Símbolos contiene los lexemas de los identificadores que hay en el programa fuente y son introducidos por el Analizador Léxico

FALSAS:
En la Tabla de Símbolos se guardan todas las constantes del programa fuente.

Tabla de Símbolos en memoria:

La Tabla de Símbolos se mantiene en memoria...

...hasta que se termine de analizar el ámbito en el que están declarados los identificadores que se han introducido en la Tabla de Símbolos correspondiente a dicho ámbito.

...durante todo el tiempo en que esté funcionando el Procesador del Lenguaje.

Atributos de la Tabla de Símbolos:

¿Qué atributos necesita una Tabla de Símbolos de un lenguaje de programación habitual (del tipo de los vistos en clase) y qué módulo del Procesador del Lenguaje es el encargado de introducir esta información?

- Atributo lexema, introducido por el analizador léxico
- Atributo posición de memoria (o desplazamiento) de una variable, introducido por el analizador semántico
- Atributo número de parámetros de un identificador que es una función, introducido por el analizador semántico
- Atributo dimensión de un identificador que es una variable de tipo vector, introducido por el analizador semántico
- Atributo etiqueta asociada a una función, introducido por el analizador semántico
- Atributo tipo de un parámetro de una función, introducido por el analizador semántico.
- Atributo tipo, introducido por analizador semántico cuando el lenguaje es tipado con declaraciones obligatorias
- Atributo tipo de retorno de una función, introducido por el analizador semántico

Entradas en la Tabla de Símbolos:

¿Cuántas entradas nuevas se crearán en la Tabla de Símbolos durante el análisis del siguiente fragmento de programa fuente?

int x;
float xx;
function float copia (int x, float y) {y= xx;}

Se crearán 5 entradas: para x, xx, copia, x, y.

Diseño de Tabla de Símbolos:

A la hora de diseñar la Tabla de símbolos para un Lenguaje de Programación, indica cuál de las siguientes afirmaciones es correcta

- Si el lenguaje permite la definición de funciones anidadas se podría implementar una pila de Tablas de Símbolos, donde se van insertando y extrayendo las Tablas de Símbolos que se crean y destruyen.

- Si el lenguaje no permite la definición de funciones anidadas, es suficiente con tener una Tabla de Símbolos Global y una Local para representar todos los identificadores.

Funciones de la Tabla de Símbolos:

¿Cuáles de las siguientes son operaciones que debe permitir realizar un Tipo Abstracto de Datos que represente una Tabla de Símbolos?

- Buscar una determinada entrada
- Acceder a la información de un atributo de una entrada
- Rellenar información sobre un atributo de una entrada
- Crear una Tabla de Símbolos vacía
- Crear una entrada nueva
- Destruir una Tabla de Símbolos

Condiciones Descendente:

¿Cuáles de las siguientes gramáticas se sabe con certeza que no son válidas para construir un Analizador Sintáctico Descendente?

- Una gramática sin factorizar.
  Explicación: Para que una gramática se pueda utilizar en un Analizador Sintáctico Descendente Predictivo tiene que estar factorizada.

- Una gramática ambigua.
  Explicación: Las gramáticas ambiguas no pueden utilizarse para construir Analizadores Sintácticos Acceder a la información de un atributo de una entrada.

- Todas las reglas del axioma de la gramática estarán colocadas en la fila del axioma.

- Una gramática que tenga las reglas $A \rightarrow a\ b$ y $A \rightarrow a\ b\ c$

- Una gramática que tenga una regla $A \rightarrow A\ b\ c$

Condición LL(1) - 1:

Dada la siguiente gramática, ¿qué afirmaciones son correctas?

$S \rightarrow 0\ A\ A\ 0 \mid 1\ A\ 1 \mid \lambda$
$A \rightarrow B \mid C$
$B \rightarrow 0\ 1 \mid 1\ 0$
$C \rightarrow 0\ 0 \mid 1\ 1$

- Las reglas de C sí cumplen la condición LL(1)
  Explicación: El FIRST(00)={0} y el FIRST(11)={1}, por lo que la intersección de ambos conjuntos es vacía

- La gramática no es válida para construir un Analizador Sintáctico Descendente Recursivo
  Explicación: La gramática no es LL(1) por las reglas de A, por lo que no se puede construir un Analizador Sintáctico Descendente

- La gramática no es válida para construir un Analizador Sintáctico Descendente con tablas
  Explicación: La gramática no es LL(1), por lo que no se puede construir un Analizador Sintáctico Descendente con tablas

- Las reglas de C sí cumplen la condición LL(1)
  Explicación: El FIRST(3)={3} y el FIRST(2B)={2}, por lo que la intersección de ambos conjuntos es vacía

- Para aplicar la condición LL(1) a las reglas de A hay que comprobar que los First no tienen elementos en común considerando 2 de las reglas de cada vez, y también que el Follow (A) no tiene elementos en común con el First de ninguna de las dos primeras reglas
  Explicación: No es suficiente comprobar que no hay ningún elemento común a todas las reglas, sino que hay que comprobarlo 2 a 2 con todas las posibles parejas de reglas

- La gramática no es LL(1) por las reglas de A, por lo que no se puede construir un Analizador Sintáctico Descendente.

- La gramática no es LL(1)

- Las reglas de A no cumplen la condición LL(1) porque las dos reglas pueden derivar en cadenas que empiezan por 1 o por 0.

- Las reglas de A no cumplen la condición LL(1)

Condición LL(1) - 2:

Dada la siguiente gramática, ¿qué afirmaciones son correctas?

S → 0 A B 0 | 1 A 1 | λ
A → 0 C | 1 C
B → 0 1 | 1 0
C → 0 | 1

- Las reglas de A sí cumplen la condición LL(1)
  Explicación: El FIRST(01)={0} y el FIRST(10)={1}, por lo que la intersección de estos conjuntos no es vacía

- La gramática es válida para construir un Analizador Sintáctico Descendente con tablas
  Explicación: La gramática es LL(1), por lo que se puede construir un Analizador Sintáctico Descendente con tablas

- Las reglas de A sí cumplen la condición LL(1) porque el FIRST de una regla es {0} y el FIRST de la otra regla es {1}

- Las reglas de B sí cumplen la condición LL(1)

- Las reglas de S sí cumplen la condición LL(1)

- Las reglas de C sí cumplen la condición LL(1)

- La gramática es LL(1)

Condición LL(1) - 3:

Dada la siguiente gramática, ¿qué afirmaciones son correctas?

S → A B | B C
A → 1 A | 2 A | λ
B → 3 C 4 | λ
C → 3 | 2 B

- Las reglas de A sí cumplen la condición LL(1)
  Explicación: El FIRST(1A)={1}, el FIRST(2A)={2} y el FIRST(λ)={$, 3}, por lo que la intersección de estos conjuntos dos a dos es vacía

- Esta gramática no cumple la condición LL(1) porque la regla B→3C4 se aplicaría cuando el siguiente token es “3” y la regla B→λ se aplicaría para los elemento del Follow(B) que contiene el “3”
  Explicación: Este caso incumple la condición LL(1)

- Para aplicar la condición LL(1) a las reglas de A hay que comprobar que los First no tienen elementos en común considerando 2 de las reglas de cada vez, y también que el Follow (A) no tiene elementos en común con el First de ninguna de las dos primeras reglas
  Explicación: No es suficiente comprobar que no hay ningún elemento común a todas las reglas, sino que hay que comprobarlo 2 a 2 con todas las posibles parejas de reglas

- La gramática no es válida para construir un Analizador Sintáctico Descendente con tablas
  Explicación: La gramática no es LL(1), por lo que no se puede construir un Analizador Sintáctico Descendente con tablas

- Las reglas de C sí cumplen la condición LL(1)
  Explicación: El FIRST(3)={3} y el FIRST(2B)={2}, por lo que la intersección de ambos conjuntos es vacía

Analizador Descendente con Tablas:

En relación con el Analizador Sintáctico Descendente por Tablas (LL(1)), ¿cuáles de las siguientes respuestas son correctas?

- Cada celda de la tabla puede contener una regla o estar vacía
  Explicación: Si hay una regla indica que el reconocimiento puede proseguir y si está vacía indica un error en el reconocimiento

- Una misma regla puede aparecer únicamente en una fila de la tabla, y podría estar varias veces
  Explicación: Cada regla está en la fila de su no terminal y en las columnas correspondientes a su FIRST

- La columna $ puede estar vacía

- La tabla no puede tener una fila vacía

Resultados Conflictos LR

Dado el siguiente estado perteneciente al Autómata reconocedor de Prefijos Viables de un Analizador Sintáctico LR, In={S→A F •, A→B D •, B→3 • A, B→• 3 A, C→3 • 4, B→5 A •} ¿cuál de las siguientes afirmaciones es correcta?

- Si el Follow(A) contiene el terminal “3”, hay un conflicto de Reducción-Desplazamiento
  Explicación: En este estado, se debería desplazar con el token “3” (por B→• 3 A), pero también reducir (por A→B D •), por lo que hay un conflicto

- Si el Follow(A) contiene el terminal “4”, hay un conflicto de Reducción-Desplazamiento
  Explicación: En este estado, se debería desplazar con el token “4” (por C→3 • 4), pero también reducir (por A→B D •), por lo que hay un conflicto

- El estado inicial contiene el ítem B -> •3C4
  Explicación: Como está el ítem axioma(S’ -> ·S) y de ahí surge el ítem S -> •BC, hay que introducir los ítems que surgen a partir de B.

- Si se añade el ítem A→3 •, habría un conflicto de Reducción-Reducción independientemente de cual fuera el Follow (A)
  Explicación: En todo caso habría dos reducciones posibles para cualquier elemento del Follow (A), por A→B D y por A→3

goto

Para la siguiente gramática, indica cuáles de los siguientes cálculos de los conjuntos cierre o goto son correctos, cuando se quiere construir un Analizador LR:

S → A B
A → B C | k C
B → C D | λ
C → + A
D → k B

- I0=cierre({S'→·S})={S'→·S, S→·AB, A→·BC, A→·kC, B→·CD, B→·, C→·+A}
- I1=goto(I0, S)={S'→S·}
- I2=goto(I0, A)={S→A·B, B→·CD, B→·, C→·+A}
- I3=goto(I0, B)={A→B·C, C→·+A}
- I4=goto(I0, k)={A→k·C, C→·+A}
- I5=goto(I0, C)={B→C·D, D→·kB}
- I6=goto(I0, +)={C→+·A, A→·BC, A→·kC, B→·CD, B→·, C→·+A}
- I7=goto(I0, D)={} Esta respuesta es correcta. Explicación: El cálculo es correcto

Gramática aumentada

¿Cuáles de las siguientes afirmaciones son correctas en relación con la Gramática aumentada?

- Si S es el axioma de la gramática original, la gramática aumentada tendrá la regla S'→S, siendo S' un nuevo axioma
  Explicación: Así es como se crea la gramática aumentada

- La gramática aumentada se obtiene a partir de la gramática original, añadiendo un nuevo axioma y una nueva regla que deriva el nuevo axioma en el axioma de la gramática original
  Explicación: De esta manera, se asegura que al reducir por esta regla, se aceptará la cadena de entrada

- La gramática aumentada es necesaria para que un Analizador Sintáctico Ascendente LR sepa cuándo se debe utilizar la acción de Aceptar
  Explicación: Se necesita para poder identificar la acción de Aceptar en un analizador ascendente

- La gramática aumentada es necesaria para construir un Analizador Sintáctico Ascendente LR
  Explicación: Se necesita para poder identificar la acción de Aceptar

- La nueva regla del axioma introducida en la gramática aumentada dará lugar a un ítem en el estado inicial del autómata de prefijos viables que tendrá el punto inmediatamente antes del axioma de la gramática original
  Explicación: Este ítem indica que aún no ha comenzado el análisis de la cadena de entrada

- Si S es el axioma de la gramática original y S' el nuevo axioma de la gramática aumentada, el estado inicial del autómata del LR tendrá el ítem S'→·S
  Explicación: Este ítem indica que aún no ha comenzado el análisis de la cadena de entrada

- La gramática aumentada es necesaria para asegurarse que el axioma nunca aparecerá en el lado derecho de ninguna regla
  Explicación: De esta manera, al reducir por la nueva regla del nuevo axioma, se sabrá que se debe aceptar la cadena

Autómata LR 1

Dada la siguiente gramática, ¿cuál de las siguientes afirmaciones relativas al Autómata Reconocedor de Prefijos Viables (método de Análisis Sintáctico Ascendente LR(1)) es correcta?
S → A B | B C
A → 1 A | 2 A | λ
B → 3 C 4 | λ
C → 3 | 2 B

- El estado inicial contiene al ítem B→•3C4
  Explicación: Como está el ítem del axioma (S'→•S) y de ahí surge el ítem S→•BC, hay que introducir los ítems que surgen a partir de B

- El estado inicial contiene al ítem A→•2A

FALSA:
El ítem C→•3 pertenece al estado inicial del autómata
Explicación: Al no aparecer en ningún momento la configuración "•C" en la parte derecha de un ítem, las reglas de C no entran en juego para añadir nuevos ítems al estado

Autómata LR 2

Dada la siguiente gramática, ¿cuál de las siguientes afirmaciones relativas al Autómata Reconocedor de Prefijos Viables (método de Análisis Sintáctico Ascendente LR(1)) es correcta?

S → B A | C B
A → 1 A | 2 A | λ
B → 3 C 4 | λ
C → 3 | 2 B

- Desde el estado inicial, hay una transición etiquetada con el símbolo “1” a un estado donde se encuentra el ítem A→•1ª
  Explicación: El ítem A→•1A está presente en el estado inicial y eso produce una transición con “1” correspondiente al Goto (I0,1) en el que al calcular el cierre({A→1•A}) vuelve a salir A→•1ª

- El estado inicial contiene al ítem C→•2B
  Explicación: Como está el ítem del axioma (S’→•S) y de ahí surge el ítem S→•CB, hay que introducir los ítems que surgen a partir de C

FALSA:
El estado inicial contiene al ítem A→•2A
Explicación: Al no aparecer en ningún momento la configuración “•A” en la parte derecha de un ítem, las reglas de A no entran en juego para añadir nuevos ítems al estado

First and Follow – 1:

P → A B C | d
A → a | λ
B → a | b | λ
C → e A D R f | λ
D → B S E
R → int | bool | P
S → if (E) {S} else {S} ; S | id = E ; S | λ
E → (E) | g

¿cuáles de los siguientes conjuntos son correctos para la gramática dada?

FIRST(E) = {(, g}
FIRST(B) = {a, b, λ}
FIRST(A) = {a, λ}
FIRST(P) = {a, b, e, λ, d}
FIRST (R) = {int, bool, a, b, e, λ, d}
FIRST(D) = {a, b, if, id, (, g}

Follow(S) = {}, (, g}
Follow(P) = {$, f}

First and Follow – 2:

P → D P | S P | λ
D → var T id ; D | λ | F; D
F → function id T (id : T L) begin S end
L → ; id : T L | λ
T → integer | boolean
S → if E do S | return E | id := E ; S
E → id (K)
K → E R
R → λ | ; K R
¿cuáles de los siguientes conjuntos son correctos para la gramática dada?

First(E) = {id}
First(F) = {function}
FIRST (R) = {;, λ}
FIRST (K) = {id}
First(SP) = {if, return, id}
First(D) = {var, λ, function}
First(P) = {var, function, λ, if, return, id}
First(DP) = {var, function, λ, if, return, id}
First(var T id ; D) = {var}
First(id := E ; S) = {id}

Follow(R) = {), ;}
Follow (D) = {var, function, $, if, return, id}
Follow (F) = {;}
FOLLOW(S) = {var, function, $, if, return, id, end}
Follow(P) = {$}

Gramática de Contexto Libre:

A → X B c | X A a | λ
X → g | λ
B → z B | z

¿Qué elementos tiene exactamente el conjunto FIRST (A)?
{ g, z, a, λ }

¿Qué elementos tiene exactamente el conjunto FOLLOW(X)?
{ z, g, a }

A → X B c | X A a | λ
X → g | λ
B → z B | z | λ

¿Qué elementos tiene exactamente el conjunto FOLLOW(X)?
{ z, c, g, a }

First y Follow de una regla:

En relación con el First y el Follow, de la regla A → B C (siendo A,B,C símbolos No Terminales), ¿cuál de las siguientes afirmaciones es **siempre** correcta?

First (C) está contenido en Follow (B)
~~Follow (C) – {λ} está contenido en Follow (B)~~
Follow (C) está contenido en Follow (B)
Follow (A) está contenido en Follow (C)
First (B) U First (C) está contenido en First (A)
Respuesta en blanco
First (B) está contenido en First (A)
(First (B) U First (C)) – {λ} está contenido en First (A)
First (B) – {λ} está contenido en First (A)