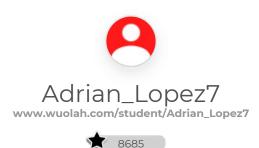
WUOLAH



ExamenPL-Junio2016.pdf

Exámenes

- 3° Procesadores de Lenguajes
- Escuela Politécnica Superior de Córdoba UCO Universidad de Córdoba

MÁSTER EN DIRECCIÓN Y GESTIÓN DE RECURSOS HUMANOS

www.mastersevilla.com









PROCESADORES DE LENGUAJES

Ingeniería Informática Especialidad de Computación Tercer curso, segundo cuatrimestre



Departamento de Informática y Análisis Numérico Escuela Politécnica Superior de Córdoba Universidad de Córdoba Curso académico 2015 - 2016 Examen de la convocatoria de junio

17 de junio de 2016

Teoría

- 1. Generación del código intermedio.
 - Descripción y características

1 punto

- 2. Boostrapping
 - Descripción
 - Ejemplo

1 punto

3. Criterios "generales" para el tratamiento de errores léxicos

1 punto

Problemas

4. Componentes léxicos

- La siguiente expresión regular denota números naturales: $c + n (c + n)^*$ donde
 - o c es el número 0
 - o n es un dígito del 1 al 9
- Utiliza el Algoritmo de Thompson para construir un AFN equivalente a la expresión regular.
- Utiliza el Algoritmo de Construcción de Subconjuntos para obtener un AFD equivalente al AFN obtenido en el apartado "a".
- · Minimiza, si es posible, el AFD obtenido en el apartado anterior.
- Comprueba si el último autómata obtenido reconoce a la siguiente secuencia:

nncc

2 puntos

5. Análisis descendente predictivo

Considérese la siguiente gramática de contexto libre

```
P = {
        1) S \rightarrow SD
        2) S \rightarrow D
        3) D → T identificador (L);

 T → entero

        5) T \rightarrow real
        6) L → ε
        7) L → A
        8) A → A, identificador
       9) A → identificador
```

Esta gramática permite generar declaraciones de prototipos de funciones como, por ejemplo:

real media (entero a, entero b); donde "media", "a" y "b" son identificadores



MASTER DIRECCIÓN Y GESTIÓN DE RECURSOS HUMANOS



Posibilidad de BECAS

escuela de negocios

CÁMARA DE SEVILLA

Ingeniería Informática

Procesadores de Lenguajes

- a) Elimina la recursividad por la izquierda y factoriza la gramática por la izquierda.
- b) A partir de la gramática obtenida en el apartado "a":
 - o Construye los conjuntos "primero" y "siguiente" de los símbolos no terminales.
 - o Construye la tabla de análisis descendente predictivo.
 - o Utiliza el método recuperación de errores de "nivel de frase" para completar la tabla predictiva.
 - o Utiliza la tabla predictiva para realizar un análisis descendente no recursivo de la siguiente declaración errónea:

real (entero entero a, entero b;

2 puntos

6. Análisis sintáctico ascendente SLR

Considérese la siguiente gramática de contexto libre

```
1) S \rightarrow SD
2) S \rightarrow D
3) D \rightarrow Var L: T;

 L → L , identificador

5) L → identificador
6) T → Integer
7) T \rightarrow Real
```

Esta gramática puede generar declaraciones de variables en el lenguaje de programación Pascal, como, por ejemplo:

```
Var a, b: Integer;
```

Var x, y, z: Real; donde "a", "b", "x", "y" y "z" son identificadores.

- a) Construye la colección canónica de LR(0)-elementos del análisis SLR
- b) Dibuja el autómata que reconoce los prefijos viables.
- c) Construye los conjuntos "primero" y "siguiente" de los símbolos no terminales.
- d) Construye la tabla de análisis sintáctico SLR: partes acción e ir_a
- e) Utiliza el método recuperación de errores de "nivel de frase" para completar la tabla de análisis SLR
- Utiliza la tabla SLR para analizar la siguiente declaración errónea:
 - a , b : Integer Integer;

2 puntos

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

7. Diseño de una gramática de contexto libre

- Diseña una gramática que permita generar declaraciones de tipos de registros usando la sintaxis del lenguaje de programación Pascal
- Por ejemplo:

```
Type TipoPersona = Record
         Nombre, Apellidos: string;
         Direccion: array [1..3] of string;
         Telefono: Integer;
       End;
```

1 punto

