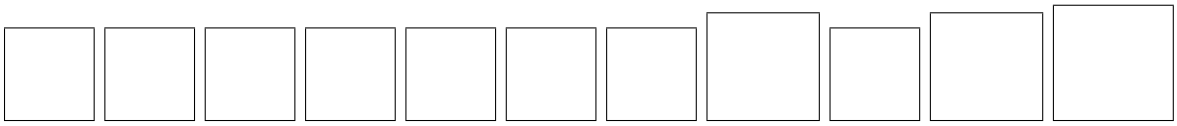
Universidad de Valladolid

Departamento de Informática

Teoría de autómatas y lenguajes formales . 2º I.T.Informática. Gestión.

Examen ordinario. 7 de junio de 2004. Cincuentenario de la muerte de Turing

Apellidos, Nombre.....



1. Todos los alumnos deberán entregar esta hoja, grapada con las soluciones de los ejercicios 1 a 5.
2. El ejercicio 6 se entregará grapado aparte.
3. Entregar las respuestas en el orden propuesto.
4. Cualquier apartado puede ser resuelto sin haber resuelto los anteriores. Cualquier apartado o resultado del enunciado puede ser usado en otro aunque no haya sido resuelto.

---

1 **(2 p.)** **Demostrar**[1] o **refutar** las siguientes igualdades entre lenguajes regulares:

1. $L^* - \varepsilon = L^+$
2. $(aa^*b)^* - \varepsilon = (ab \mid a)^*ab$
3. $(\alpha\alpha^*\beta)^* - \varepsilon = (\alpha\beta \mid \alpha)^*\alpha\beta \quad$ con $\quad \alpha = (\varepsilon \mid a), \beta = \varepsilon$

2 **(1 p.)** Demostrar que $\{a^p \ / \ p$ primo $\}$ no es independiente de contexto.

3 **(2 p.)** 1. Justifíquese la siguiente afirmación: si L es reconocible **por estado final** por un AP **determinista** y R es regular, entonces $L \cap R$ también es reconocible **por estado final** por un AP **determinista**.

2. ¿Es el lenguaje $\{wcw^R \,/\, w \in (a \mid b)^* \wedge |w|_a$ es par $\}$ independiente de contexto?. Justifíquese la respuesta y, si lo es, hállese una gramática independiente de contexto que lo genere o indíquese un algoritmo para obtenerla.

4 **(2 p.)** Considérese el conjunto de gramáticas sobre $\{a, b\}$ con auxiliares $\{S, A, B\}$, suponiendo que S es siempre el símbolo inicial.

Cada una de ellas puede ser codificada como una cadena sobre el alfabeto $\Sigma = \{S, A, B, a, b, e, f, c\}$ donde e, f y c representan respectivamente a la cadena vacía, la flecha y la coma; por ejemplo,

$\{S \to AS, SA \to \varepsilon\}$ se codifica como la cadena $SfAScSAfe$.

1. Describir el conjunto de cadenas sobre Σ que representan gramáticas independientes de contexto.
2. Razonar por qué el lenguaje formado por las codificaciones de gramáticas independientes de contexto ambiguas es recursivamente numerable.

5 **(1'5 p.)** Para calcular la TASP para una determinada gramática, un estudiante calcula los primeros de cada auxiliar, y coloca las reglas no anulables en los lugares adecuados de la tabla.

En lugar de continuar con el algoritmo conocido, (calculando siguientes, etc.), coloca las reglas $N \to \varepsilon$ que encuentra en la gramática en todas las casillas de la fila N de la TASP que quedaron vacías después del proceso anterior. Construye a partir del resultado un analizador sintáctico predictivo.

1. Supongamos que la gramática era $LL(1)$ y sólo tenía un símbolo anulable N, que no era el inicial. ¿Qué lenguaje reconoce su analizador? (¿$L(G)$, un subconjunto de $L(G)$ o un conjunto que contiene a $L(G)$?).

[1] usando reconocedores finitos o equivalencias conocidas de expresiones regulares

2. Considérese la gramática

$$G_1 : \begin{cases} S \rightarrow & aSb \mid N \\ N \rightarrow & bN \mid \varepsilon \end{cases}$$

   Determínese el lenguaje generado. Realícese el proceso de nuestro estudiante con esta gramática y simúlese el resultado sobre ε, ab , abb y aab. ¿Qué cadenas reconoce el analizador construido?

3. Constrúyase la TASP correcta.

4. Descríbanse las formas sentenciales derechas para G. ¿Será G $LALR(1)$?

6 **(1'5 p.)** Usando la herramienta Yacc, construir un analizador sintáctico para un determinado tipo de sentencias de asignación en Pascal: las que están formadas por una variable en la parte izquierda y una expresión de constantes en la parte derecha.

El analizador léxico necesario para resolver el problema se construirá utilizando la herramienta Lex.

Las variables Pascal podrán ser de cualquiera de estos cuatro tipos : enteras (cualquier identificador que empiece por la letra `i`), reales (cualquier identificador que empiece por la letra `r`), string (cualquier identificador que empiece por la letra `s`) o boolean (cualquier identificador que empiece por la letra `b`).

Las operaciones permitidas en la expresión dependerán del tipo. Con el tipo entero y el real se permiten la suma (+), resta (-), multiplicación(*), y cambio de signo (-). Con el tipo string, la concatenación de cadenas (+). Con el tipo boolean, AND, OR y NOT. El uso de paréntesis debe permitirse para cualquier tipo de datos.

El programa resultante deberá devolver el resultado de realizar las operaciones indicadas en la expresión de la entrada, sólo en el caso de que ésta sea correcta.

Por ejemplo:

Entrada	Salida esperada
ivar1 := 5 + 8 * 5;	45
bvar2 := 5 + 3;	error de tipo
bvar3 := TRUE or TRUE and FALSE;	TRUE
svar4 := 'hola' + 'mundo';	'holamundo'