Teoría de la Computación
Proyecto Grupal

Prof. José Fiestas
UTEC

Instrucciones para desarrollar el proyecto:

- formen grupos de 3 estudiantes
- la entrega será: un informe de proyecto desarrollado en LaTeX
- escoja uno de los 5 ejercicios. La calificación será sobre 20
- La nota de proyecto será igual a $\mathbf{P} = \mathbf{0.5} \cdot \mathbf{D} + \mathbf{0.4} \cdot \mathbf{Pe} + \mathbf{0.1} \cdot \mathbf{Po}$, donde **D** es el desarrollo de códigos de programación, **Pe** es la presentación escrita , y **Po** es la presentación oral
- Entrega: miércoles 6 de Noviembre

1 Ping-Pong

Diseñe autómatas con las siguientes funciones

- un proceso de login: el sistema acepta solo un login a la vez y pide primero el usuario, el cual es aceptado luego de ser chequeado, y continuar; o rechazarlo y regresar al estado inicial. Luego pide la clave y ejecuta el mismo procedimiento. Implemente además un contador de ingresos fallidos (contador de seguridad). Cuando el usuario accede al sistema, se regresa a esperar al siguiente usuario.
- un proceso de comunicación entre dos computadores. Se inicia con un estado de espera de mensaje, seguido de un estado de recibo de mensaje. Una vez realizado, se envía la confirmación del recibo. Luego se ejecuta el requerimiento de información, el cual, si le es denegado, se vuelve a enviar. Si es aceptado, regresa al estado inicial a esperar por un requerimiento de otra computadora
- dos máquinas que permitan un login al sistema independientemente, y realicen el juego del ping-pong, consistente en enviar un número entero del primer computador al segundo computador, el cual duplica el valor del mismo y lo re-envía al primero. Así sucesivamente por 10 veces consecutivas.

Las partes del proyecto son:

1. Construcción de los autómatas de login y comunicacion
2. Construcción del autómata de ping-pong
3. Describir la derivación que conduce al estado de aceptación y obtener su valor (siendo el valor inicial 1)

2 Cifras en alemán

Programe en C++ la siguiente gramática de números cardinales en alemán en forma normal de Chomsky:
$\mathbf{R_1} : \mathbf{Z_2} \rightarrow \mathbf{Z_3}+ \mathbf{zehn}$
$\mathbf{R_2} : \mathbf{Z_7} \rightarrow \mathbf{Z_4}+ \mathbf{zig}$
$\mathbf{R_3} : \mathbf{Z_7} \rightarrow \mathbf{drei} + \mathbf{ssig}$
$\mathbf{R_4} : \mathbf{Z_8} \rightarrow \mathbf{U} + \mathbf{Z_7}$
$\mathbf{R_5} : \mathbf{Z_9} \rightarrow \mathbf{Z_1} + \mathbf{Z_5}$
$\mathbf{R_6} : \mathbf{Z_{10}} \rightarrow \mathbf{Z_2} + \mathbf{Z_5}$
$\mathbf{R_7} : \mathbf{Z_{11}} \rightarrow \mathbf{Z_{5,9}} + \mathbf{Z_{1,2,7,8}}$
$\mathbf{R_8} : \mathbf{Z_{12}} \rightarrow \mathbf{Z_{10}} + \mathbf{Z_{1,2,7,8}}$
$\mathbf{R_9} : \mathbf{Z_{13}} \rightarrow \mathbf{Z_{1,2,7,8,9,11}} + \mathbf{Z_6}$
$\mathbf{R_{10}} : \mathbf{Z_{14}} \rightarrow \mathbf{Z_{6,13}} + \mathbf{Z_{1,2,7,8,9,11}}$
$\mathbf{R_{11}} : \mathbf{U} \rightarrow \mathbf{Z_1}+\mathbf{und}$

Donde:
Z_1 = {ein, zwei, drei, vier, fünf, sechs, sieben, acht, neun}
Z_2 = {zehn, elf, zwölf }
Z_3 = {drei, vier, fünf, sech, sieb, acht, neun }
Z_4 = {zwan, vier, fünf, sech, sieb, acht, neun }
Z_5 = {hundert}
Z_6 = {tausend}
drei = {drei}
zehn = {zehn}
zig = {zig}
ssig = {ssig}
und = {und}

El proyecto, consistirá de las siguientes partes:

1. Dado un número entero, programar el **scanner** correspondiente (i.e. que determina expresiones válidas de acuerdo al alfabeto)
2. Programar el **parser** correspondiente con un estado de aceptación y rechazo
3. Implementar además una regla de identificación de errores (estado de error) que imprima un mensaje de error

4. Implementar las salidas del análisis (aceptado o rechazado)

Probar el código con los siguientes números:

- **fünftausendzweihundertneunundfünfzig**
- **zweitausendneunhundertsechsundsiebzig**
- **zweihundertzweiundzwanzigtausendvierhundertsiebzehn**
- **einszwei**

3 Árbol ancestral

Programe en C++ para traducir un traductor de términos ancestrales
A_1={ mother, father, grandmother, grandfather, greatgrandmother, greatgrandfather, greatgreatgrandmother, ..., greatgreatgreatgreatgrandfather, ...}
y construya una gramática que lo reproduzca.
Traduzca asimismo las expresiones de entrada en el número de generaciones que existen entre los ancestros y la persona. I.e. **mother** y **father** : 1. generación, **grandmother** y **grandfather**: 2. generación, **greatgrandmother** y **greatgrandfather**: 3. generación, etc..
Traduzca las expresiones a un lenguaje intermedio, como:
mother, mo()
father, fa()
grandmother, g(mo())
grandfather, g(fa())
greatgrandmother, g(g(mo()))
greatgrandfather, g(g(fa()))
...
greatgreatgreatgrandmother , g(g(g(g(mo()))))
greatgreatgreatgrandfather, g(g(g(g(fa()))))
...

Almacene las expresiones en una lista de tokens y reconstrúyalas al alemán según el lenguaje
A_2 ={mutter, vater, grossmutter, grossvater, urgrossmutter, urgrossvater, ururgrossmutter, ...} .
E.g. la palabra **greatgreatgrandfather** de A_1 debe traducirse en la lista **string["ur", "ur", "gross", "vater"]**.

El código debe también traducir una expresión del lenguaje $\mathbf{A_3}$ en uno del lenguaje $\mathbf{A_4}$, según la siguiente tabla

A_3	A_4
The mother of Mary	Die mutter von Maria
The father of Mary	Der vater von Maria
The mother of John	Die mutter von Johann
The father of John	Der vater von Johann
The mother of the mother of Mary	Eine grossmutter von Maria
The mother of the father of Mary	Eine grossmutter von Maria
The father of the mother of Mary	Ein grossvater von Maria
The father of the father of Mary	Ein grossvater von Maria
The mother of the mother of John	Eine grossmutter von Johann
The mother of the father of John	Eine grossmutter von Johann
The father of the mother of John	Ein grossvater von Johann
The father of the father of John	Ein grossvater von Johann
...	...
The mother of the mother of the father of the mother of John	Eine urgrossmutter von Johann

El proyecto, consistirá de las siguientes partes:

1. Diseñar la **gramática**
2. Programar el **parser** correspondiente
3. Construir una **representación intermedia** de las expresiones
4. Implementar atributos en el **parser**, que hagan la traducción
5. Implementar las salidas de traducción

4 Clausuras

Resuelva las siguientes demostraciones:

- Siendo L un lenguaje y **half**(L) el conjunto de las primeras mitades de cadenas en L, i.e. L={ w | $\forall x$, $|x| = |w|$, $wx \in L$ }. E.g. si L={ $\epsilon, 0010, 011, 010110$}, **half**(L)= { $\epsilon, 00, 010$}. Observe que cadenas de longitud impar no contribuyen a **half**(L). Demuestre que si L es un lenguaje regular, también lo es **half**(L)

- La **fusion** de los lenguajes L_1 y L_2 está definida como:
fusion(L_1, L_2) = { $w_1v_1w_2v_2...w_mv_m$ | $w_1w_2...w_m \in L_1, v_1v_2...v_m \in L_2, \forall w_iv_i \in \Sigma^*$}. Demuestre que la familia de lenguajes regulares es cerrada bajo la operación **fusion**

- Una **expresión regular generalizada** es una expresión regular con dos operaciones adicionales, de intersección y complemento. E.g. $abb\phi' \cap (\phi' aaa\phi')'$ representa el conjunto de cadenas en $\{a, b\}^*$ que empiezan con abb y no contienen aaa. Demuestre que el subconjunto $\{aba\}^*$ de $\{a, b\}^*$ puede ser expresado por una expresión regular generalizada sin ocurrencias de *

5 Cosmología

Algunos científicos formulan el modelo de evolución del universo como un Autómata Celular (modelo discreto) de la ecuación cósmica de onda (KdV ecuation, por Korteweg-de Vries) $\partial_t u + \partial_x^3 u + \frac{3u\partial_x u}{u_0} = 0$, siendo u_0 una constante, y x,t representando el espacio y tiempo
Una forma discretizada es la propuesta por A.Steeb & Hardy's:
$u_j(t+1) = u_j(t)(u_{j+1}(t) - u_j(t)) + u_{j+2}(t) - u_{j+1}(t) - u_{j-1}(t)$

- Diseñe un autómata para un universo de 4 celdas que resuelva la ecuación discretizada
- Programe en C++ la solución discretizada del sistema de ecuaciones diferenciales para un universo de 1000 celdas. Normalice las constantes a 1.
- Observe el estado final del universo en comparación al estado inicial y comente los resultados