Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Trabajo Práctico 1

Teoria de Lenguajes

Segundo Cuatrimestre de 2015

Apellido y Nombre	LU	E-mail
Rodriguez Pedro	197/12	pedro3110.jim@gmail.om
Matias Pizzagali	257/12	matipizza@gmail.com

Índice

1. Introducción

El objetivo de este trabajo práctico es desarrollar un compositor de fórmulas matemáticas. El mismo tomará como entrada la descripción de una fórmula en una versión muy simplificada del lenguaje utilizado por LATEX y producirá como salida un archivo SVG (Scalable Vector Graphics).

2. Desarrollo

Para poder realizar el TP, utilizamos la librería PLY para Python, la cual implementa Lex y Yacc enteramente en Python. Este analizador léxico y sintáctico nos permitirá aceptar o no aceptar cadenas de entrada de acuerdo a la gramática que definamos.

Definimos los siguientes tokens, con sus respectivas expresiones regulares:

- ID (para cualquier caracter distinto a _, ^, (,), { y })
- SUBINDEX (para la ER " / ")
- SUPERINDEX (para la ER " ^ ")
- DIVIDE (para la ER " /")
- LPAREN (para la ER " (")
- RPAREN (para la ER ") ")
- LBRACKET (para la ER " { ")
- RBRACKET (para la ER " } ")

2.1. Desambiguación de la gramática

$$
\begin{array}{lcl}
E & \to & E\ E \\
 & | & E\ /\ E \\
 & | & E\ \hat{}\ E \\
 & | & E\ \_\ E \\
 & | & E\ \hat{}\ E\ \_\ E \\
 & | & E\ \_\ E\ \hat{}\ E \\
 & | & (\ E\ ) \\
 & | & \{\ E\ \} \\
 & | & l
\end{array}
$$

$$
\begin{array}{lcl}
S & \to & E \\
E & \to & E\ /\ A \\
 & | & A \\
A & \to & A\ \ B \\
 & | & B \\
B & \to & C \\
 & | & C\ \hat{}\ C \\
 & | & C\ \_\ C \\
 & | & C\ \hat{}\ C\ \_\ C \\
 & | & C\ \_\ C\ \hat{}\ C \\
C & \to & (\ E\ ) \\
 & | & \{\ E\ \} \\
 & | & l
\end{array}
$$

La gramática propuesta inicialmente fue la de la izquierda. Sin embargo, al utilizar esta gramática para hacer el parsing, se generan conflictos Shift/Reduce. Esto es debido a que la gramática es ambigua (para una misma cadena pueden haber más de un árbol de derivación posibles). Luego, para evitar conflictos Shift/Reduce, generamos la otra gramática expuesta, equivalente a la anterior pero con la diferencia de que no es ambigua.

Tuvimos en cuenta que la división es la operación de menor precedencia, seguida de la concatenación, luego de los subíndice y superíndice y luego por lo paréntesis y corchetes. También que división y concatenación son asociativas a izquierda y que subíndice y superíndice no son asociativos.

2.2. TDS y Atributos

Para cada uno de los símbolos A,B,C,E, definimos los siguientes atributos heredados de tipo entero:

- x : para guardar el comienzo de cada subexpresión sobre el eje x
- y : para guardar el comienzo de cada subexpresión sobre el eje y
- tam : para guardar la altura total de cada subexpresión

Y también los siguientes atributos sintetizados, también de tipo entero.

- $ancho$: para guardar el ancho de cada subexpresión
- h_up : contiene la distancia que ocupa una expresion entre la linea sobre la que se escriben los caracteres y el techo de la expresión, vista como un rectángulo.
- h_down : contiene la distancia que ocupa una expresion entre la linea sobre la que se escriben los caracteres y la línea de base de la expresión, vista como un rectángulo.
- $altura$: contiene la altura total de la expresión. Es la suma de h_up y h_down

A partir de estos atributos, generamos la siguiente TDS:

- $S \to \{E.tam = 1, E.x = 0, E.y = 0\}E$
- $E \to \{A.tam = E.tam, A.x = E.x, A.y = E.y\}A$
- $E_1 \to \{E_2.tam = E_1.tam, E_2.x = E_1.x + E_1.ancho/2 - E_2.ancho/2, E_2.y = E_1.y - E_2.h\_down - 0{,}2 * E_1.tam\}E_2/$
  $\{A.tam = E_1.tam, A.x = E_1.x + E_1.ancho/2 - A.ancho/2, A.y = E_1.y + A.h\_up + 0{,}1 * E_1.tam\}A$
  $\{E_1.h\_up = E_2.altura + 0{,}35 * E_1.tam, E_1.h\_down = A.altura + 0{,}3 * E_1.tam, E_1.ancho = max(E_2.ancho, A.ancho), E_1.altura = E_1.h\_up + E_1.h\_down\}$
- $A \to \{B.tam = 1, B.x = A.x, B.y = A.y\}B$
  $\{A.h\_up = B.h\_up, A.h\_down = B.h\_down, A.altura = B.altura, A.ancho = B.ancho\}$
- $A_1 \to \{A_2.tam = A_1.tam, A_2.x = A_1.x, A_2.y = A_1.y\}A_2$
  $\{B.tam = 1, B.x = A_2.x + A_2.ancho, B.y = A_1.y\}B$
  $\{A_1.h\_up = max(A_2.h\_up, B.h\_up), A_1.h\_down = max(A_2.h\_down, B.h\_down), A_1.ancho = A_2.ancho + B.ancho, A_1.altura = A_1.h\_up + A_1.h\_down\}$
- $B \to \{C.tam = B.tam, C.x = B.x, C.y = B.y\}C$
  $\{B.ancho = C.ancho, B.h\_up = C.h\_up, B.h\_down = C.h\_down\}$
- $B \to ID\{B.h\_up = 0{,}8 * B.tam, B.h\_down = 0{,}2 * B.tam, B.altura = B.\_up + B.h\_down, B.ancho = 0{,}6 * B.tam\}$
- $B \to \mathbf{\{}\{E.x = B.x, E.y = B.y, E.tam = B.tam\}E\mathbf{\}}ds$
  $\{B.h\_up = E.h\_up, B.h\_down = E.h\_down, B.ancho = E.ancho\}$
- $B \to \{C_1.tam = B.tam, C_1.x = B.x, C_1.y = B.y\}C_1\hat{}$
  $\{C_2.tam = C_1.tam * 0{,}7, C_2.x = C_1.x + C_1.ancho, C_2.y = C_1.y - C_2.h\_down - 0{,}5 * C_1.tam\}C_2$

$\{B.ancho = C_1.ancho + C_2.ancho, B.h\_up = C_1.h\_up + C_2.altura * 0{,}7, B.h\_down = C_1.h\_down B.altura = B.h\_down + B.h\_up\}$

- $B \rightarrow \{C_1.tam = B.tam, C_1.x = B.x, C_1.y = B.y\} C_1\_$
  $\{C_2.tam = C_1.tam * 0{,}7, C_2.x = C_1.x + C_1.ancho, C_2.y = C_1.y + C_2.h\_up\} C_2$
  $\{B.ancho = C_1.ancho + C_2.ancho, B.h\_up = C_1.h\_up, B.h\_down = C_1.h\_down + 0{,}7 * C_2.altura, B.altura = B.h\_up + B.h\_down\}$

- $B \rightarrow \{C_1.tam = B.tam, C_1.x = B.x, C_1.y = B.y\} C_1\hat{}$
  $\{C_2.tam = C_1.tam * 0{,}7, C_2.x = C_1.x + C_1.ancho, C_2.y = C_1.y - C_2.h\_down - 0{,}5 * C_1.tam\} C_2\_$
  $\{C_3.tam = C_1.tam * 0{,}7, C_3.x = C_1.x + C_1.ancho, C_3.y = C_1.y + C_3.h\_up\} C_3$
  $\{B.ancho = C_1.ancho + max(C_2.ancho, C_2.ancho), B.h\_up = C_1.h\_up + C_2.altura * 0{,}7, B.h\_down = C_1.h\_down + C_3.altura * 0{,}7, B.altura = B.h\_up + B.h\_down\}$

- $B \rightarrow \{C_1.tam = B.tam, C_1.x = B.x, C_1.y = B.y\} C_1\_$
  $\{C_2.tam = C_1.tam * 0{,}7, C_2.x = C_1.x + C_1.ancho, C_2.y = C_1.y + C_2.h\_up\} C_2\hat{}$
  $\{C_3.tam = C_1.tam * 0{,}7, C_3.x = C_1.x + C_1.ancho, C_3.y = C_1.y - C_3.h\_down - 0{,}5 * C_1.tam\} C_3$
  $\{B.ancho = C_1.ancho + max(C_2.ancho, C_2.ancho), B.h\_up = C_1.h\_up + C_3.altura * 0{,}7, B.h\_down = C_1.h\_down + C_2.altura * 0{,}7, B.altura = B.h\_up + B.h\_down\}$

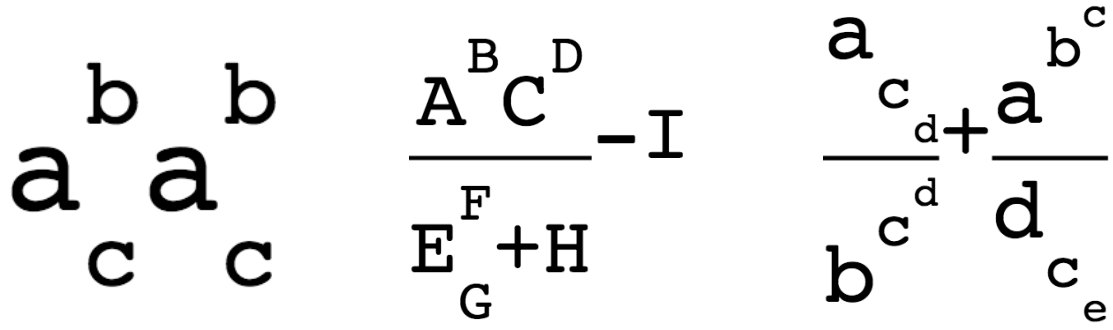
- $B \rightarrow (\{E.x = B.x + B.tam * 0{,}3, E.y = B.y, E.tam = B.tam\} E)$
  $\{B.h\_up = E.h\_up, B.h\_down = E.h\_down, B.ancho = E.ancho + 2 * B.tam * 0{,}3, B.altura = B.h\_down + B.h\_up\}$

Para implementar esta TDS, generamos para cada cadena de entrada un árbol sintáctico en el cual cada hoja está dada por el símbolo terminal l. Para trabajar con el resto de los símbolos (no terminales), definimos los siguientes nodos:

- START: es el nodo raíz. Allí inicializamos el tamaño inicial de la subexpresión, y calculamos la posición y de la expresión para que posteriormente quede centrada en el espacio donde la dibujamos. Tiene 1 hijo.
- (): para caracterizar a la subexpresión contenida entre las ER " (" y ") ". Cuenta con 1 hijo
- CONCAT: para caracterizar la concatenación de dos subexpresiones válidas. Cuenta con 2 hijos
- SUPERINDEX: para caracterizar dos subexpresiones unidas por la ER ^. Cuenta con 2 hijos
- SUBINDEX: para caracterizar dos subexpresiones unidas por la ER _. Cuenta con 2 hijos
- SUBSUPERINDEX: para caracterizar tres subexpresiones unidas primero por un _y luego por un ^. Cuenta con 3 hijos
- SUPERSUBINDEX: para caracterizar tres subexpresiones unidas primero por un ^ y luego por un _. Cuenta con 3 hijos
- DIVISION: para caracterizar dos subexpresiones unidas mediante la ER de división, " / ". Cuenta con 2 hijos

Cada nodo y cada hoja tiene los mismos 7 atributos que tienen los símbolos de nuestra gramática. Para heredar y sintetizar de forma correcta dichos atributos según la TDS ya descripta, recorremos este árbol 3 veces (la primera y última de forma top-down y la segunda bottom-up). Esto lo hacemos en los siguientes 3 recorridos (etapas):

- recorrer1: heredamos desde la raíz hasta las hojas el atributo *tam*



First Duck

Second Duck

Third Duck

(a)

- recorrer2: sintetizamos desde las hojas hasta la raíz los atributos *altura*, *ancho*, *h_up* y *h_down*, utilizando el atributo *tam*, ya calculado.
- recorrer3: heredamos desde la raíz hasta las hojas los atributos x e y, utilizando los atributos ya calculados en los dos recorridos anteriores. Como en esta etapa obtenemos el árbol decorado correctamente con todos los atributos, también aquí generamos el archivo SVG que buscamos, a partir de un nuevo recorrido top-down del árbol sintáctico.

3. Resultados y casos de prueba

4. Manual de usuario

Para correr el TP, es necesario tener instalado Python 2.7 o posterior y la librería PLY, que puede obtenerse en https://pypi.python.org/pypi/ply. Para correr el tp, abrir una terminal en la carpeta src, y ejecutar el comando " python tp.py ". A continuación, introducir la cadena que se desea generar. Se genera la figura deseada con la extensión " .svg " en el archivo " figura.svg ".

5. Codigo

5.1. tp.py