

Trabajo Práctico

Teoría de Lenguajes

Segundo cuatrimestre 2017

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).

Para simplificar el trabajo utilizaremos una tipografía monoespaciada. Esto permitirá resolver el problema sin necesidad de contar con información sobre la métrica de los distintos caracteres de la tipografía.

2 Lenguaje de entrada

La siguiente gramática ambigua describe la sintaxis que deben seguir las expresiones que acepte el programa:

E	\rightarrow	$E E$	(concatenación)
		$E \sim E$	(superíndice)
		$E _ E$	(subíndice)
		$E \sim E _ E$	(super y subíndice)
		$E _ E \sim E$	(sub y superíndice)
		E / E	(división)
		(E)	(encerrar entre paréntesis)
		$\{ E \}$	(agrupar)
		l	(cualquier caracter salvo \sim , $_$, $/$, $\{$, $\}$, $($ y $)$)

Para eliminar la ambigüedad en la interpretación, se debe tener en cuenta que la división es la de menor precedencia, seguida de la concatenación. Ambas son asociativas a izquierda. El superíndice y subíndice son no asociativos (no es lícito escribir $a \sim b \sim c$ ni $a _ b _ c$) y tienen más precedencia que la concatenación. Por ejemplo, la cadena $(A \sim B C \sim D / E \sim F _ G + H) - I$ es equivalente a $\{(\{A \sim B\}\{C \sim D\})/\{\{E \sim F _ G\}+H\})-I$, y el resultado se podría ver así:

$$\left(\frac{A^B C^D}{E_G + H} \right) - I$$

3 Descripción de la salida

Scalable Vector Graphics (SVG) es un formato estándar de imágenes vectoriales¹ basado en XML² creado por el W3C³. Describiremos algunas características del formato SVG, pero se pueden encontrar más detalles en la documentación oficial⁴.

En SVG un elemento `text` escribirá un texto en las posiciones establecidas por las coordenadas x e y . La coordenada y indica la línea de base para el texto. Una vez que se presenta el primer carácter en la posición especificada, se actualiza la *posición actual* sumándole el valor de *avance* del carácter escrito para escribir a continuación el carácter siguiente.

En el sistema de coordenadas de SVG, la posición $(x, y) = (0, 0)$ es la correspondiente al extremo superior izquierdo.

Existen distintas convenciones para especificar tamaños de letra. Una de las más usadas es la de indicar el interlineado para el cual está diseñada la tipografía. Cuando se habla de una tipografía de 10 unidades usando esta convención, se quiere decir que la tipografía está diseñada para que la distancia vertical entre las líneas de base de los caracteres sea de 10 unidades cuando se usa interlineado normal. En la tipografía monoespaciada Courier, la separación horizontal entre cada carácter (el *avance*) es el mismo para todos: el 60% del interlineado. En SVG el tamaño del texto se establece mediante el atributo *font-size*.

Por ejemplo, los elementos:

```
<text x="10" y="50" font-family="Courier" font-size="10">
The quick brown fox
</text>
<text x="10" y="60" font-family="Courier" font-size="10">
jumps over the lazy dog
</text>
```

producen el siguiente resultado:

The quick brown fox
jumps over the lazy dog

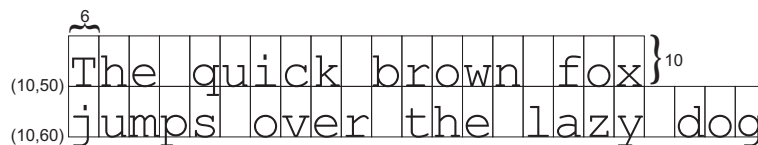
al que le agregamos una grilla y algunas anotaciones para que se vea cuáles son las dimensiones y coordenadas resultantes:

¹Las imágenes vectoriales pueden ser reducidas o agrandadas a cualquier tamaño sin perder datos ni pixelarse.

²<http://www.w3.org/XML/>

³World Wide Web Consortium

⁴<http://www.w3.org/TR/SVG/>



Para el super y subíndice se suele reducir el tamaño de letra a un 70 % del original y desplazar la línea de base hacia arriba o hacia abajo respectivamente. Para las divisiones, la línea de división se suele alinear con la línea horizontal de los signos más y menos (aproximadamente 28 % del interlineado sobre la línea de base). El numerador y el divisor se ubican centrados horizontalmente con respecto a la línea de división.

En el cuadro 1 se ve una posible salida para la cadena de entrada $(A^BC^D/E^F_G+H)-I$, cuya imagen mostramos en la sección anterior. En este ejemplo se utiliza un elemento `text` para cada caracter de salida especificando su posición y tamaño. La excepción son los paréntesis, a los que se les aplica una transformación para que resulten estirados verticalmente.

Cuadro 1: Ejemplo de salida para la cadena de entrada $(A^BC^D/E^F_G+H)-I$

```
<?xml version="1.0"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
<g transform="scale(40) translate(0, .62)" font-family=
"Courier">
  <text x="0" y="0" font-size="1" transform=
  "translate(0, 1.36875) scale(1,2.475)"></text>
  <text x=".69" y=".53" font-size="1">A</text>
  <text x="1.29" y=".08" font-size=".7">B</text>
  <text x="1.71" y=".53" font-size="1">C</text>
  <text x="2.31" y=".08" font-size=".7">D</text>
  <line x1="0.6" y1="0.72" x2="2.82" y2=".72"
    stroke-width="0.03" stroke="black"/>
  <text x="0.6" y="1.68" font-size="1">E</text>
  <text x="1.2" y="1.93" font-size=".7">G</text>
  <text x="1.2" y="1.23" font-size=".7">F</text>
  <text x="1.62" y="1.68" font-size="1">+</text>
  <text x="2.22" y="1.68" font-size="1">H</text>
  <text x="0" y="0" font-size="1" transform=
  "translate(2.82, 1.36875) scale(1,2.475)"></text>
  <text x="3.42" y="1" font-size="1">-</text>
  <text x="4.02" y="1" font-size="1">I</text>
</g>
</svg>
```

4 Modo de uso

El programa debe poder ser invocado por línea de comandos, recibiendo como parámetro la especificación de la fórmula y escribiendo el resultado en su salida standard o en un archivo. En caso de que lo recibido como parámetro no sea una fórmula válida se lo debe informar en la salida de error standard.

Alternativamente, si se usa una herramienta que genere código JavaScript el programa puede residir en una página web que muestre el resultado en forma interactiva, ya que todos los navegadores modernos pueden mostrar SVG.

5 Implementación

Hay dos grupos de herramientas que se pueden usar para generar los analizadores léxicos y sintácticos, y agregar las acciones requeridas para generar el documento de salida:

- Uno utiliza expresiones regulares y autómatas finitos para el análisis lexicografico, y la técnica LALR para el análisis sintáctico. Ejemplos de esto son flex y bison, que generan código C o C++, JLex y CUP, que generan código Java, y ply, que genera código Python.
- El otro grupo utiliza la técnica ELL(k) tanto para el análisis léxico como para el sintáctico, generando parsers descendentes iterativos recursivos. Ejemplos son JavaCC, y ANTLR, que están escritos en Java. JavaCC puede generar código Java o C++. ANTLR puede generar Java, C++, Python y JavaScript, entre otros.

Si quieren usar otra herramienta, consúltenlo con alguno de los docentes.

6 Detalles de la entrega

Se deberá enviar el código a la dirección de e-mail tptleng@gmail.com.

La entrega debe incluir:

- Un programa que cumpla con lo solicitado
- El código fuente del programa.
- Informe conteniendo:
 - el código de la solución. Si se usaron herramientas generadoras de código, imprimir la fuente ingresada a la herramienta, no el código generado.

- Las modificaciones a la gramática o indicaciones adicionales que hayan sido necesarias para construir el parser.
- Descripción de cómo se implementó la solución.
- Información y requerimientos de software para ejecutar y recompilar el tp (versiones de compiladores, herramientas, plataforma, parámetros, etc).
- Casos de prueba con expresiones sintácticamente correctas e incorrectas, resultados obtenidos y conclusiones.