Teoría de Lenguajes - Trabajo Práctico
Dibu: Gráficos vectoriales para niños

Versión 1.0

2do cuatrimestre 2016

Fecha de entrega: jueves 1 de diciembre

1. Introducción

Como parte de la popularización de la programación en la enseñanza inicial, se desea desarrollar un lenguaje para crear gráficos vectoriales simples. Los scripts de este lenguaje (que llamaremos *Dibu*) consistirán de una serie de instrucciones que describan las formas geométricas que componen el gráfico, como se aprecia en la Figura 1.

```
size height=200, width=200
rectangle upper_left=(0,0), size=(50, 50), fill="red"
rectangle upper_left=(100,0), size=(50, 50)
rectangle upper_left=(50,50), size=(50, 50)
rectangle upper_left=(150,50), size=(50, 50)
rectangle upper_left=(0,100), size=(50, 50)
rectangle upper_left=(100,100), size=(50, 50)
rectangle upper_left=(50,150), size=(50, 50)
rectangle upper_left=(150,150), size=(50, 50)
```

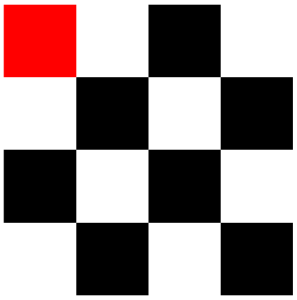

Figura 1: Ejemplo de gráfico con *Dibu*, y su representación gráfica

En este primer prototipo del lenguaje, sólo nos concentraremos en describir programas que consten de sentencias sin ninguna estructura de control ni variables. Aún más, utilizaremos por debajo el formato SVG[1] para poder aprovechar las herramientas gráficas existentes.

2. Descripción del lenguaje de entrada

Los scripts de nuestro lenguaje consistirán en una serie de instrucciones de la forma

```
IDENTIFICADOR PARAM1=V1, PARAM2=V2, ..., PARAMN=VN
```

donde `IDENTIFICADOR` es el nombre de la función y `PARAM_I=V_I` enumera los argumentos que recibe la función. Los argumentos de la función no están ordenados con lo cual las siguientes llamadas son equivalentes:

```
rectangle upper_left=(0, 0), size=(50, 50)
rectangle size=(50, 50), upper_left=(0, 0)
```

Todos los parámetros de una función están nombrados, para mayor entendimiento de los usuarios del lenguaje.

2.1. Literales

Los literales que forman parte del lenguaje son

- Números: enteros o de punto flotante
- Strings: tiras de caracteres encerrados entre comillas dobles
- Pares/Puntos: elementos de la forma (v_1, v_2), con v_1, v_2 otros literales
- Arreglos: elementos de la forma `[ v_1, v_2, ... , v_n]` con `v_i` otros literales

[1] http://commons.oreilly.com/wiki/index.php/SVG_Essentials

2.2. Funciones a implementar

Las funciones que debe implementar el lenguaje son:

size

Configura las dimensiones del lienzo(*canvas*).

- `height` (Numérico): alto del lienzo
- `width` (Numérico): ancho del lienzo

Esta función sólo puede llamarse en, a lo sumo, una ocasión, sin importar dónde. De llamarse más de una vez, se debe marcar error.

rectangle

Dibuja un rectángulo con esquina izquierda `upper_left`, ancho `width`, y altura `height`

- `upper_left` (Punto): esquina superior izquierda del rectángulo
- `height` (Numérico): alto del lienzo
- `width` (Numérico): ancho del lienzo

line

Dibuja una línea desde `from` hasta `to`.

- `from` (Punto): Punto de salida de la línea
- `to` (Punto): Punto de llegada de la línea

circle

Dibuja una circunferencia centrada en `center`

- `center` (Punto): Centro de la circunferencia
- `radius` (Numérico): Radio de la circunferencia

ellipse

Dibuja un elipse centrado en `center` y semiejes `rx` y `ry`

- `center` (Punto): Centro de la circunferencia
- `rx` (Numérico): Semiradio horizontal del elipse
- `ry` (Numérico): Semiradio vertical del elipse

polyline

Dibuja una sucesión de líneas que conectan los puntos sucesivos de `points`

- `points` (Arreglo de puntos): puntos que forman parte de la polilínea

polygon

Igual que `polyline` pero conecta el primer y el último punto

- `points` (Arreglo de puntos): puntos que forman parte de la polilínea

text

Muestra la leyenda `t` en la posición `at`

- `t` (String): Contenido del texto
- `at` (Punto): Coordenadas donde se grafica el texto

Argumentos opcionales

- `font-family` (String): Familia de la fuente
- `font-size` (String): Tamaño de la fuente (según estándar SVG)

Como puede observarse, todas las funciones de *Dibu*se corresponden con alguna figura geométrica del formato SVG.

2.2.1. Argumentos opcionales comunes a todas las funciones

Todas las funciones reciben como mínimo los parámetros listados, y también pueden recibir los parámetros descriptos para cada una de las figuras en el estándar SVG:

- `fill`
- `stroke`
- `stroke-width`

Cualquier otro argumento que no sea los obligatorios de una función, o bien estos argumentos comunes debe ser señalado como error.

3. Descripción del lenguaje de salida

La salida generada debe ser un string en formato SVG que represente el gráfico vectorial descripto. Ejemplos de programas en *Dibu*y sus respectivas salidas son provistas en las notebooks de *Jupyter* que acompañan este enunciado.

4. Implementación

Para el lenguaje descripto se solicita crear un analizador léxico, generando los tokens necesarios de acuerdo a la gramática que se haya diseñado, y un analizador sintáctico y semántico,que deberá compilar un programa de *Dibu*a *SVG* en caso de que sea válida. En caso de que no lo sea, será necesario indicar el lugar preciso (línea y posición) en el cual se encuentra el problema detectado.

4.1. Herramientas a utilizar

Se sugiere utilizar *Ply* como generador del analizador sintáctico, y *Jupyter* para graficar los *SVG* resultantes. El repositorio de la materia ofrece el esqueleto del trabajo práctico y los ejemplos de *Dibu*.
De usar otro conjunto de herramientas, se solicita que se incluyan ejemplos donde se grafique por lo menos los ejemplos dados.

Otras herramientas sugeridas son:

- *bison+flex*, parser LALR que genera código $C, C++$ (y algunos otros) Son las implementaciones gratuitas de
- *JLex y Cup*, parser LALR para Java
- *ANTLR*, un parser *ELL(k)*

5. Detalles de la entrega

Se deberá enviar el código a la dirección de e-mail tptleng@gmail.com, satisfaciendo lo siguiente:

- el **asunto de mail** debe ser `[TL-TP]` seguido por el nombre del grupo (e.g., "[TL-TP] Los Turing-computables").
- en el mail deberán estar copiados todos los integrantes del grupo.

La entrega debe incluir lo descripto a continuación:

- un programa que cumpla con lo solicitado,
- el código fuente del mismo, adecuadamente documentado,
- informe enviado por e-mail y entregado impreso, con los siguientes contenidos:
  - carátula con datos de integrantes del grupo y nombre del grupo,
  - breve introducción al problema a resolver,
  - la gramática (incluyendo las expresiones regulares de los tokens) obtenida a partir del enunciado y las transformaciones de la misma, en caso de haberlas, que hubieran sido necesarias para la implementacion (explicándolas y justificándolas),
  - indicación del tipo de la gramática definida, de acuerdo a los vistos en clase,
  - el código de la solución. Si se usaron herramientas generadoras de código, imprimir la fuente ingresada a la herramienta, no el código generado,
  - descripción de cómo se implementó la solución, con decisiones que hayan tenido que tomar y justificación de las mismas,
  - información y requerimientos de software para ejecutar y recompilar el tp (versiones de compiladores, herramientas, plataforma, etc). Sería como un pequeño manual del usuario, que además de los requerimientos, contenga instrucciones para compilarlo, ejectuarlo, información de parámetros y lo que consideren necesario,
  - casos de prueba para el lenguaje, graficando sus salidas
  - un resumen de los resultados obtenidos, y conclusiones del trabajo.

Referencias

[1] Aho, A.V., Lam, M.S., Sethi, R., *Compilers: Principles, Techniques and Tools - Second Edition*, Pearson Education, 2007.

[2] Grune, D., Jacobs, C.J.H, *Parsing Techniques: A Practical Guide - Second Edition*, Springer, 2008.

[3] PLY (Python Lex-Yacc). Documentación. Disponible en http://www.dabeaz.com/ply/ply.html

[4] Goyvaerts, J., Levithan, S., *Regular Expressions Cookbook*, O'Reilly, 2009.

[5] Hopcroft, J.E., Motwani, R., Ullman, J.D., *Introduction to Automata Theory, Languages, and Computation - Third Edition*, Addison Wesley, 2007.

[6] Parr, T., *The Definitive ANTLR 4 Reference*, The Pragmatic Programmers, 2012.