**Documentación Externa**
**TP3**
**TI-3404 Lenguajes de Programación.**

# *SML en Python.*

*Ambiente Estático y dinámico.*


Andrés Maroto 201128337
David Murillo 201108648
Adrián Siles 201136841
Ariel Arias. 201136814

Profesor: Andrei Fuentes L.

7 Noviembre 2013.

# Tabla de Contenidos

# Descripción del Problema

La idea es hacer un programa que lea programas del lenguaje del paradigma funcional sml, y recopile la información del ambiente estático y el ambiente dinámico.

Para la tabla del ambiente estático, se deberá almacenar la información de los tipos de datos de cada una de las expresiones.

Para la tabla del ambiente dinámico, se deberá almacenar la información de los valores de cada una de las expresiones.

Cabe destacar que el alcance del proyecto no contempla implementar el manejo de funciones, se pueden implementar solamente asociaciones de variables, expresiones let, if-else-then, y valores.

Con respecto a los tipos de datos, se tiene que manejar como mínimo integers, booleans, listas y tuplas.

Una vez que hayan leído toda la información del programa, deberán mostrar la información del ambiente estático y dinámico en forma de tabla mediante una lista.

Una vez desplegado en pantalla la tabla, el programa habrá finalizado correctamente.

# Diseño del Programa

El programa se compone de un menú donde se pregunta al usuario el nombre del archivo de texto que contiene el código de sml para que proceda a la lectura del mismo y despliega las tablas tanto del ambiente estático como dinámico. El archivo a leer tiene que estar ubicado en el mismo lugar donde se encuentra el programa.

A la hora de la lectura del archivo se tiene como supuesto que el código venga sin errores semánticos, léxicos y sintácticos, además de que los elementos de las expresiones estén separados por un espacio. Por ejemplo val x = 3, en vez de valx=3 o val x=3.

El proceso de la apertura del archivo se da mediante la función o el método predeterminado open(), salvándose en una variable que posteriormente se usaría con el método read() para guardar el contenido del archivo en un string. A este string se le eliminan los saltos de página y otros datos que no tienen que ver con el código en sí, y se reemplazan por espacios. Se procede a aplicar el método split() al string utilizando de separador de elementos el espacio, convirtiéndose en una lista que contiene los elementos de todas las expresiones.

Por ejemplo, si en el código de sml viene val x = 3, este se transforma a [val,x,=,3], todo esto con el objetivo de poder realizar el análisis del código recorriendo los elementos de sus expresiones.

Como el código en sml se transformara a una lista este se recorre de una forma más sencilla, de forma que cuando se se inicie el proceso de análisis del código se encontrarán las expresiones que inician por un let, o por un val y así se podrán identificar y definir las asociaciones que se hacen cuando se lee el código. Analizando espacio por espacio la lista que contiene el código de forma que cuando se encuentra determinada expresión, ya sea un let o val, se continuará analizando la lista que contiene el código dependiendo del tipo de expresión que se hubiese encontrado con anterioridad. Primero se identifica la variable y luego el

operador de asignación, luego de esto se analizará el valor que se le asignará a la variable y si este se encuentra condicionado por algún if y su respectiva expresión que lo condiciona, de forma que si no cumple con el if, a la variable se le asignará el valor que delimitó el else correspondiente con el dicho if que se encontraba condicionando la asignación de la variable. luego de la identificación de la variable y del valor respectivo, se procede a almacenar el identificador de la variable y el valor que tiene asignada la misma, dentro de la tabla de ambiente dinámico, de la misma forma se almacenará el identificador de la variable y el tipo de dato correspondiente al valor que tiene asignado la variable, en la tabla de ambiente estático. Luego de que se añadieron la variable, su valor y su tipo a las tablas de ambientes, se procede a seguir analizando la lista que contiene el código de forma recursiva hasta que se recorra por completo, al final de esto se despliegan las tablas de ambiente dinámico y ambiente estático.

# Librerías Usadas

No fue necesario el uso de librerías externas de Python, por lo que esta parte queda automáticamente vacía.

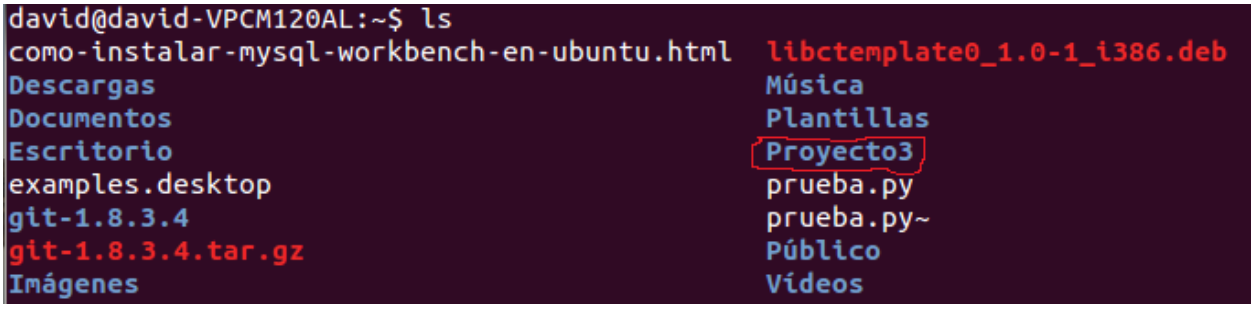
# Análisis de Resultados

Completamos exitosamente los requerimientos para la TP3:

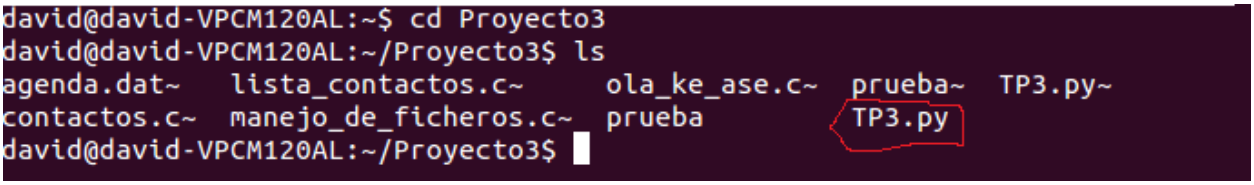
- Leer correctamente un archivo escrito en SML.
- Generar la tabla del ambiente estático que maneja los tipos de las variables.
- Generar la tabla del ambiente dinámico que maneja los valores .
- Desplegar en pantalla el resultado.

# Manual de Usuario

En el siguiente se mostrará los pasos básicos que debe seguir el usuario para abrir y utilizar el programa. Para iniciar el programa, se comienza por abrir la terminal del sistema operativo. Lo primero que se debe hacer, es localizar el archivo donde esta ubicado el programa, para ello utilizamos el comando “ls” y este nos dará la lista de los archivos y directorios que contiene el directorio actual:




Con el comando “cd”, seguido del nombre del directorio donde se encuentra ubicado el programa, nos llevará a ese directorio. Volvemos a utilizar el comando “ls” para obtener el nuevo menú del directorio en el que nos encontramos. En este caso, el programa que se debe de encontrar es TP3.py y este se encuentra en la carpeta “Proyecto 3”.




Para ejecutar el programa, se debe de realizar mediante el uso del siguiente comando “python ”, seguido del nombre de TP3.py con esto iniciará el programa.

```
david@david-VPCM120AL:~/Proyecto3$ python TP3.py
```

Al iniciar el programa, por defecto el menú de acciones será mostrado, preguntando al usuario el nombre del archivo que desea hacer lectura del código sml.

```
   Analizador de ambientes Estaticos y Dinamicos

Escriba el nombre del archivo que desea analizar: prueba
```

A continuación el programa automáticamente realizará el análisis del código y desplegará la tabla del ambiente dinámico y del ambiente estático. Dando fin al programa de manera correcta

```
   TABLA DINAMICA
   y        5
   z        4
   w        [True,False,True]

   TABLA ESTATICA
   y        int
   z        int
   w        bool list
```

# Conclusión Personal

Para el desarrollo de esta tercera tarea programada, decidimos trabajar con el lenguaje de programación Python, las razones fueron muchas, pero principalmente nos decidimos por este ya que teníamos buenas bases de conocimiento sobre como funcionaba ya que en el curso de Intro a la Programación utilizamos este lenguaje y en la segunda tarea programada también lo utilizamos por lo que la curva de aprendizaje era mínima y era cuestión de recordar detalles para utilizarlo correctamente y sacarle provecho a la manera tan sencilla en que funciona este lenguaje interpretado.

Además como otra herramienta de trabajo resultó sumamente provechoso seguir utilizando Github que facilita el proceso a la hora de programar grupalmente. Y brinda opciones para realizar el “branching” de diferentes versiones del proyecto. Gracias a este control de versiones es que facilita la programación en un grupo porque al almacenar todas las versiones del código que se han incluido en el repositorio, resulta sencillo poder observar y comprender los cambios que realiza algún compañero y de ser necesario se pueden encontrar las versiones anteriores a un cambio. Esto sumandole que nuestro grupo de trabajo es de 4 personas y nos ayudó a organizarnos de la mejor manera.

Para desarrollar el programa en sí y entender realmente cómo funciona un lenguaje funcional y sus asociaciones  para poder programarlo, nos basamos en las presentaciones que sube el profesor al Tec Digital y en los conocimientos aprendidos en clase, por lo que no tuvimos que buscar tanto material en la web como en proyectos anteriores.

Gracias al desarrollo de esta tarea logramos ampliar y mejorar nuestros destrezas en el lenguaje de Python; adquirimos conocimientos acerca de la lógica detrás de los  lenguajes funcionales y su proceso de inferencia de tipos y los niveles del scope y su shadowing.

También se pudo analizar y comprender las formas de resolver el problema que propone cada compañero, y con esto poder dar con la mejor combinación de ideas y obtener un programa completo.

# Bibliografía Consultada


1. Python Documentation Index (s. f.). Recuperado el 04 de Noviembre del 2013, de http://www.python.org/doc/21.

2. ML of New Jersey (1998). *SML/NJ FAQ* Recuperado el 04 de Noviembre del 2013, de http://www.smlnj.org/doc/FAQ/index.html