Universidad Tecnológica Nacional
Facultad Regional Delta
Sintaxis y Semántica de los Lenguajes
12 de Mayo de 2025

Trabajo Práctico de Laboratorio N° 1

Objetivo

Diseñar e Implementar un Analizador Lexicográfico (Lexer) para el lenguaje de programación TINY, cuya gramática se específica posteriormente.

Descripción

El analizador lexicográfico se implementará en el lenguaje de programación Python, mediante un programa principal que utilice los autómatas finitos determinísticos(AFD) que se construirán para cada uno de los tokens del lenguaje de programación TINY.
El software que resulte de la implementación del lexer, deberá aceptar como entrada una cadena que representa código escrito en el lenguaje TINY, y deberá convertir este código, interpretado como una cadena de caracteres ASCII o UTF-8, a una lista de tokens correspondiente a la gramática provista, donde cada token estará representado por un par $(tipo\_token, lexema)$, donde *tipo_token* representa, para cada subcadena del código de entrada, el tipo de token correspondiente, y *lexema* es el valor asociado al token que generó dicha clasificación.

Tareas a Desarrollar

Para completar el trabajo, y poder desarrollar el analizador lexicográfico, se deben realizar los siguientes ítems:

1) Identificar los tokens a utilizar en el analizador lexicográfico.
2) Por cada uno de los tokens identificados en el ítem anterior, dar su especificación mediante un expresión regular.
3) Dar el autómata finito determinístico asociado a cada una de las expresiones regulares del ítem anterior.
4) Implementar cada uno de los autómatas en Python.
5) Diseñar e implementar en Python el programa principal que utilice los autómatas del ítem anterior, para completar el lexer.
6) Diseñar como mínimo 10 pruebas, consistentes en código fuente en TINY.
7) Mostrar la salida de su lexer para cada una de esas pruebas.

   Nota: *estas pruebas deben cubrir el uso de todos los tokens, así como también casos de error.*

Observaciones

Aquí enumeramos algunas consideraciones a tener en cuenta a la hora de desarrollar el trabajo, a modo de recordatorio de lo sugerido en clase:

- La implementación del analizador lexicográfico se podra realizar en grupos de 4 integrantes como máximo.
- Ya sea durante las entregas intermedias de corrección, como en la entrega final, ***no utilizar funciones del tipo input o cualquier interactividad que requiera la intervención del usuario***, por ejemplo tener que tipear la cadena de entrada, más allá de correr el programa, pues esto dificulta el desarrollo para los alumnos y la evaluación por parte del profesor.
- Enviar todos los archivos necesarios para correr el programa comprimidos en un único archivo con formato zip, rar o similares.
- El archivo debe llamarse: apellido o nombre del grupo-tp1-version, donde cada nueva ronda de correcciones tiene que tener una nueva versión (1, 2, 3, etc). Por ejemplo: ***gonzalez-lopez-tp1-v3***
- El archivo zip debe contener el código fuente de su trabajo y el desarrollo de todos los ítems en un informe en formato pdf.

Gramática de TINY

Si bien se especifica a continuación la gramática en la notación usual $G =< VN, VT, P, S >$, detallando cada una de los 4 elementos de la gramática, se hacen las siguientes aclaraciones para que puedan leer la gramática más fácilmente desde las producciones si lo desean:

- El símbolo distinguido es *Program*.
- Los terminales comienzan con minúsculas y se hallan en negrita.
- Los no terminales comienzan en mayúsculas, pudiendo contener mayúsculas intermedias para aclarar su significado, y se escriben en itálica.
- Terminales y No Terminales se hallan separados por espacios en blanco para claridad de la gramática.

$$VN = \{TCode,\ Body,\ DecVarList,\ DecVar,\ DecVarBody,\ Statement, StatementList,\ StatementBody,\ Goto,\ Assingment,\ Op,\ MatOp,\ BoolOp,\ Lvalue,\ Rvalue,\ Conditional,\ CompExpr,\ CompOp\}$$

$$VT = \{id,\ num,\ program,\ var,\ .\ ,\ ;,\ =,\ :=,\ int,\ bool,\ true,\ false,\ begin,\ end,\ if,\ else,\ not,\ <,\ >,\ <>,\ <=,\ >=,\ +,\ -,\ *,\ ==,\ (,\ ),\ \cdots,\ and,\ or\}$$

$$S = TCode$$

$$
\begin{aligned}
P = \{ TCode &\longrightarrow \textbf{program id .}\ Body \\
Body &\longrightarrow \textbf{begin}\ StatementList\ \textbf{end} \\
&\mid \textbf{var}\ DecVar\ DecVarList\ \textbf{begin}\ StatementList\ \textbf{end} \\
DecVarList &\longrightarrow DecVarList\ DecVar \\
&\mid \lambda \\
DecVar &\longrightarrow \textbf{id :}\ DecVarBody \\
DecVarBody &\longrightarrow \textbf{int (num} \cdots \textbf{num) = num ;} \\
&\mid \textbf{bool = true ;} \\
&\mid \textbf{bool = false ;} \\
StatementList &\longrightarrow StatementList\ Statement \\
&\mid \lambda \\
Statement &\longrightarrow \textbf{id :}\ StatementBody \\
&\mid StatementBody \\
StatementBody &\longrightarrow Assignment \\
&\mid Conditional \\
&\mid Goto \\
Goto &\longrightarrow \textbf{goto id ;} \\
Assignment &\longrightarrow \textbf{let}\ Lvalue = Rvalue\ \textbf{;} \\
&\mid \textbf{let}\ Lvalue = Rvalue\ Op\ Rvalue\ \textbf{;} \\
&\mid \textbf{let}\ Lvalue = \textbf{not}\ Rvalue\ \textbf{;} \\
Op &\longrightarrow MatOp \\
&\mid BoolOp \\
MatOp &\longrightarrow + \\
&\mid \textbf{-} \\
&\mid \textbf{*} \\
BoolOp &\longrightarrow \textbf{or} \\
&\mid \textbf{and} \\
Lvalue &\longrightarrow \textbf{id} \\
Rvalue &\longrightarrow \textbf{id} \\
&\mid \textbf{num} \\
&\mid \textbf{true} \\
&\mid \textbf{false} \\
Conditional &\longrightarrow \textbf{if}\ CompExpr\ \textbf{goto id ; else goto id ;} \\
&\mid \textbf{if}\ CompExpr\ \textbf{goto id ;} \\
CompExpre &\longrightarrow Rvalue\ CompOp\ Rvalue \\
CompOp &\longrightarrow == \\
&\mid <> \\
&\mid < \\
&\mid > \\
&\mid <= \\
&\mid >= \\
&\}
\end{aligned}
$$