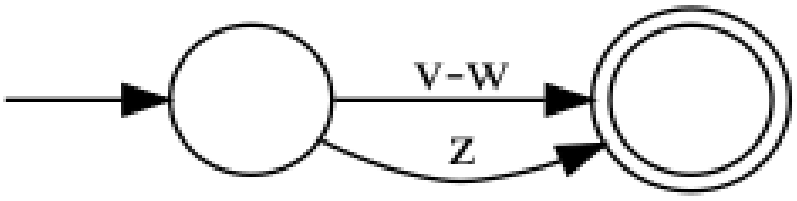
JEFA

Un lenguaje para autómatas, expresiones regulares y más.



Autores:

- *Agustin Lavarello*
- *Julián Palacci*
- *Esteban Kramer*
- *Francisco Pérez Sammartino*

TABLA DE CONTENIDO

IDEA SUBYACENTE

El lenguaje inició como una idea para facilitar el manejo de expresiones regulares y autómatas dada la equivalencia existente entre ambos.

CONSIDERACIONES REALIZADAS

Elegimos que el código de salida del compilador sea Java puesto que disponíamos de una librería externa que permite realizar operaciones sobre autómatas y expresiones regulares fácilmente, con lo que las funcionalidades de nuestro lenguaje podían ser implementadas más fácilmente si el lenguaje de salida era Java.

DESCRIPCIÓN DEL DESARROLLO DEL TP

Para realizar el analizador lexicográfico se utilizó Lex, y para el parser Yacc. Ambos fueron programados en el lenguaje C.

Como lenguaje intermedio se utilizó Java, es decir que el ejecutable resultado de la compilación debe ser ejecutado sobre la JVM.

Se intentó relegar la mayor parte de los problemas de compilación a Java. Los problemas de compilación del compilador de JEFA lanzará errores en aquellos casos donde dado el error se haga imposible generar el código Java.

POTENCIA DEL LENGUAJE

El lenguaje JEFA, como fue mencionado anteriormente, provee una forma sencilla de manejar autómatas y expresiones regulares. Los tipos de datos primarios del lenguaje son: enteros, strings y autómatas. A continuación, se detallan algunas de las operaciones posibles:

- Generación de autómata a partir de expresión regular.
- Generación de autómata a partir de unión/intersección/concatenación/complemento/minimización/ determinación de otro/s autómata/s.
- Lectura de a líneas de entrada estándar.
- Impresión de autómatas en formato GraphViz. Puede visualizarlo de forma online o descargando algún programa como "DOT" que permita la conversión del archivo graphviz a formato imagen.
- Operación booleana "accepts" que determina si un autómata acepta una palabra(string) dada.
- Concatenación de strings
- Impresión de strings/autómatas/números.
- Operaciones binarias and/or/equals entre strings, autómatas, enteros.

El lenguaje admite comentarios. Se deben escribir como "#{comentario}"

DESCRIPCIÓN DE LA GRAMÁTICA

G = < NT, T, S, P >

T = {`REGEXP, REGEXP_T, WHILE, LPAREN, RPAREN, LCURLY, RCURLY, IF, ELSE, SEMICOL, EQASS, OR, AND, EQCOMP, NE, LT, LE, GT, GE, ADD, SUB, MUL, DIV, MOD, OPP, TOREGEXP, STRING_T, INT_T, AUTO_T, AUTO, PRINT, ACC, GRAPH, SCAN, MIN, CONCAT, COMP, DET, TRUE, FALSE`}

NT = {`file, statement, expression, definition, primary, type, assignment, operand, operator, built_in, function, fargs, boperator, uop` }

S = {`file`}

Donde el conjunto P está definido por:

file : statement

statement : statement statement

| WHILE LPAREN expression RPAREN LCURLY statement RCURLY

| SEMICOL

| IF LPAREN expression RPAREN LCURLY statement RCURLY

| definition SEMICOL

| assignment SEMICOL

| function SEMICOL

| expression SEMICOL

| operand SEMICOL

definition :

type assignment

| type ID

type : INT_T

| AUTO_T

| STRING_T

expression : operand boperator operand

| OPP expression

| LPAREN expression RPAREN AND LPAREN expression RPAREN

| LPAREN expression RPAREN OR LPAREN expression RPAREN

| LPAREN expression RPAREN

| TRUE

| FALSE

```
boperator : EQCOMP
        | NE
        | LT
        | GT
        | LE
        | GE
        | ACC
operand :  primary
        | operand operator operand
        | uop primary
        | SUB INT
        | ADD INT
uop : TOREGEXP
        | MIN
        | DET
        | COMP
primary : ID
        | built_in
        | LPAREN operand RPAREN
        | uop primary
built_in : STRING
         | INT
operator : ADD
        | SUB
```

| AND

| OR

| MUL

| DIV

| CONCAT

assignment : ID EQASS operand

function : PRINT fargs

| GRAPH fargs

| SCAN ID

fargs : expression

| operand

DIFICULTADES ENCONTRADAS

- En algunas sentencias del programa, por ejemplo (a equals b) surgía el problema que al convertirlo a java equals no se reemplaza solo en el lugar sino que necesita, en este caso, convertirse a "a.equals(b)" , es decir que hay que agregar un paréntesis al final. Este problema fue resuelto indicándole al padre de la operación, para que él se encargue de imprimir el equals y llame a sus hijos. Es decir que se altera la idea original de que la impresión se realice sólo en las hojas del árbol.

FUTURAS EXTENSIONES

- Definir un autómata en base a transiciones y estados, y no solo a partir de expresiones regulares. Esto tiene una complejidad alta, no en cuanto a la implementación, sino en el diseño de una forma amigable para el usuario que quiere realizarlo.

REFERENCIAS

- dk.brics.automaton
  http://www.brics.dk/automaton/index.html
- GraphViz https://www.graphviz.org/