



**DEPARTAMENTO  
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

# Trabajo Práctico 2

Miércoles 29 de julio de 2015

Teoría de Lenguajes

Integrante	LU	Correo electrónico
Aleman, Damián Eliel	377/10	damian_8591@hotmail.com
Gauna, Claudio Andrés	733/06	gauna_claudio@yahoo.com.ar



**Facultad de Ciencias Exactas y Naturales**  
Universidad de Buenos Aires  
Ciudad Universitaria - (Pabellon I/Planta Baja)  
Intendente Güiraldes 2160 - C1428EGA  
Ciudad Autonoma de Buenos Aires - Rep. Argentina  
Tel/Fax: (54 11) 4576-3359  
<http://www.fcen.uba.ar>

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Gramática</b>	<b>2</b>
2.1. Lexer . . . . .	2
2.2. Parser . . . . .	3
2.3. Atributos . . . . .	3
<b>3. Modo de uso</b>	<b>4</b>
<b>4. Tests</b>	<b>4</b>
<b>5. Conclusion</b>	<b>4</b>
<b>6. Apendice</b>	<b>4</b>
6.1. Código Lexer . . . . .	4
6.2. Código Parser . . . . .	4
6.3. Código Midcomp . . . . .	5

# 1. Introducción

El objetivo del trabajo practico es implementar un parser para un lenguaje orientado a la composicion de piezas musicales, llamado Musileng, que luego sera transformado al formato MIDI 1 para su reproduccion.

Los pasos que seguimos para realizar el trabajo practico fueron:

- Generar la gramatica adecuada, que sintetice el lenguaje orientado a la composicion de las piezas musicales
- Escribir los terminales del lenguaje y las reglas del lexer
- Escribir los no terminales del lenguaje y las reglas del parser.
- Agregarle semantica para que pueda imprimir al lenguaje intermedio que pueda ser leído por el programa midcomp
- Luego de finalizada la traduccion, pueda transformarse a MIDI (.mid) por medio del programa midicomp

Utilizamos para el trabajo la herramienta ANTLR para generar el parser (y el lexer) necesitado.

# 2. Grámatica

Primero mostramos la gramatica del lexer que lee el archivo como secuencia de caracteres y los mapea a una secuencia de simbolos terminales.

El conjunto de terminales es: {NUMERAL, TEMPO, DURACION, NUM, COMPAS, SLASH, CONST, NOMBRE, IGUAL, VOZ, LPAREN, RPAREN, LBRACE, RBRACE, REPETIR, SILENCIO, PUNTILLO, ALTURA, OCTAVA, COMA, PUNTOYCOMA }

A continuación definimos la gramática que generamos para sintetizar el lenguaje: Las terminales de la gramática son todas las cadenas que están entre comillas simples y los que se derivan a partir de una regla ( de los tokens en mayúscula).

## 2.1. Lexer

A la gramatica del lexer, le dimos reglas para que pueda reconocer los tokens:

```
TEMPO → 'tempo';
COMPAS → 'compas';
CONST → 'const';
IGUAL → '=';
VOZ → 'voz';
LPAREN → '(';
RPAREN → ')';
LBRACE → '{';
RBRACE → '}';
NUMERAL → '#';
COMA → ',';
PUNTOYCOMA → '.';
SLASH → '/';
REPETIR → 'repetir';
SILENCIO → 'silencio';
NOTA → 'nota';
PUNTILLO → '.';
DURACION → ('redonda' | 'blanca' | 'negra' | 'corchea' | 'semicorchea' | 'fusa' | 'semifusa');
ALTURA → ('do' | 're' | 'mi' | 'fa' | 'sol' | 'la' | 'si' | 'do+' | 're+' | 'mi+' | 'fa+' | 'sol+' | 'la+' | 'si+' | 'do-' | 're-' | 'mi-' | 'fa-' | 'sol-' | 'la-' | 'si-');
NUM → [0-9]+;
NOMBRE → [a-zA-Z_]+;
OCTAVA → [1-9];
COMENTARIOS → '\\\' ~[\\r\\ n]* → skip;
WS → [ \\t\\r\\n] + → skip;
```

## 2.2. Parser

A continuación mostramos la gramática del parser:

$S \rightarrow \text{tempos elcompas constantes melodia}$   
 $\text{tempos} \rightarrow \text{NUMERAL TEMPO DURACION NUM}$   
 $\text{elcompas} \rightarrow \text{NUMERAL COMPAS NUM SLASH NUM}$   
 $\text{constantes} \rightarrow \text{constante}^*$   
 $\text{constante} \rightarrow \text{CONST NOMBRE IGUAL NUM}$   
 $\text{melodia} \rightarrow (\text{VOZ LPAREN LPAREN NUM RPAREN } \{ \text{compases} \} )+ \\ | (\text{VOZ LPAREN texto RPAREN } \{ \text{compases} \} )+$   
 $\text{compases} \rightarrow \text{compas compases} \mid \text{repeticion compases}$   
 $\text{repeticion} \rightarrow \text{REPETIR LPAREN NUM RPAREN LBRACE compas RBRACE}$   
 $\text{compas} \rightarrow (\text{COMPAS LBRACE nota RBRACE})+ \mid (\text{COMPAS LPAREN silencio RPAREN})+$   
 $\text{silencio} \rightarrow \text{SILENCIO LPAREN DURACION PUNTILLO? RPAREN}$   
 $\text{nota} \rightarrow \text{NOTA LPAREN ALTURA COMA octava, DURACION PUNTILLO? RPAREN}$   
 $\text{octava} \rightarrow \text{OCTAVA} \mid \text{NOMBRE}$

Notar que los símbolos en mayúsculas son terminales, mientras los que están en minúscula son no terminales. El símbolo distinguido es  $S$ .

El conjunto de no terminales es:

$\{S, \text{tempos}, \text{elcompas}, \text{constantes}, \text{constante}, \text{melodia}, \text{compases}, \text{repeticion}, \text{compas}, \text{silencio}, \text{nota}, \text{octava}\}$

## 2.3. Atributos

Asignamos atributos para verificar las restricciones que tenemos que hacer de modo tal que la gramática genere el lenguaje que necesitamos. Los atributos también los usaremos para realizar la traducción al lenguaje intermedio para que sea legible por el programa `midcomp`.

Los atributos sintetizados son:  $\{\text{partitura}, \text{tempo}, \text{indicacion}, \text{listaCompases}, \text{voces}, \text{repeticiones}, \text{compasObj}, \text{silencioObj}, \text{notaObj}, \text{valor}\}$

Los atributos heredados son:  $\{\text{indicacion}\}$

Ahora haremos una breve explicación de cada atributo:

- *partitura*: Es el objeto raíz, almacena el tempo, la indicación y la lista de voces.
- *tempo*: Este atributo guarda la información de la duración de la figura y la cantidad de veces que entra esa figura en un minuto.
- *indicacion*: Almacena el numerador y el denominador definidos del compás.
- *voces*: La lista de voces de la melodía.
- *listaCompases*: La lista de los compásObj y las repeticionesObj.
- *compasObj*: Tiene la lista de notas de cada compás.
- *repeticiones*: Almacena la cantidad de repeticiones.
- *notaObj*: Almacena la altura, de qué octava es, la duración, la alteración y si tiene puntillo.
- *silencioObj*: Tiene la lista de silencios de cada compás.
- *valor*: El valor de la octava.
- *listaCompases*: Tiene todos los compases definidos.

### 3. Modo de uso

### 4. Tests

Realizamos una serie de tests para verificar que el parser funciona correctamente. Para ello le dimos como entrada al parser, archivos invalidos:

- octava 10, es decir fuera del rango del 0 al 9.
- Repetir 0 veces una lista de compases.
- voces:La lista de voces de la melodia
- Distinta duracion de dos compases.
- Instrumento fuera de rango del 1 al 127.
- dos declaraciones distintas de una misma constante.
- Sin voz.

### 5. Conclusion

### 6. Apendice

#### 6.1. Codigo Lexer

```
lexer grammar lexerGrammar;  
  
TEMPO: 'tempo';  
COMPAS : 'compas';  
CONST: 'const';  
IGUAL: '=';  
VOZ: 'voz';  
LPAREN: '(' ;  
RPAREN: ')' ;  
LBRACE: '{' ;  
RBRACE: '}' ;  
NUMERAL: '#';  
COMA: ',' ;  
PUNTOYCOMA: ';' ;  
SLASH: '/';  
REPETIR: 'repetir';  
SILENCIO: 'silencio';  
NOTA: 'nota';  
PUNTILLO : '.' ;  
DURACION : ('redonda'|'blanca'|'negra'|'corchea'|'semicorchea'|'fusa'|'semifusa');  
ALTURA : ('do'|'re'|'mi'|'fa'|'sol'|'la'|'si'|  
           'do+'|'re+'|'mi+'|'fa+'|'sol+'|'la+'|'si+'|  
           'do-'|'re-'|'mi-'|'fa-'|'sol-'|'la-'|'si-');  
  
NUM: [0-9]+;  
NOMBRE: [a-zA-Z_]+;  
OCTAVA: [1-9] ;  
COMENTARIOS: '//' ~[\r\n]* -> skip ;  
//para escapar espacios, tabs, y saltos de linea  
WS : [ \t\r\n]+ -> skip ;
```

#### 6.2. Codigo Parser

### 6.3. Codigo Midcomp

```
package tleng.tp2;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

import org antlr.v4.runtime.ANTLRFileStream;
import org antlr.v4.runtime.CharStream;
import org antlr.v4.runtime.CommonTokenStream;
import org antlr.v4.runtime.TokenStream;

import tleng.tp2.MusilengParser.Compas;
import tleng.tp2.MusilengParser.IndicacionCompas;
import tleng.tp2.MusilengParser.Nota;
import tleng.tp2.MusilengParser.NotaEnum;
import tleng.tp2.MusilengParser.Partitura;
import tleng.tp2.MusilengParser.Voz;
import static tleng.tp2.MusilengParser.clicksPorFigura;

public class midicomp {

    private static final String USAGE = "Usage : ./midicomp -c entrada.txt salida.midi";

    public static void main(String[] args) throws IOException {

        if (args.length != 3) {
            System.out.println(USAGE);
        } else {
            if (!args[0].equals("-c")) {
                System.out.println(USAGE);
            } else {
                String inFile = args[1];
                String midiFile = args[2];
                CharStream charStream = new ANTLRFileStream(inFile);
                MusilengLexer lexer = new MusilengLexer(charStream);
                TokenStream tokenStream = new CommonTokenStream(lexer);
                MusilengParser parser = new MusilengParser(tokenStream);
                Partitura partitura = parser.s().partitura;

                PrintWriter writer = new PrintWriter(new BufferedWriter(new FileWriter(midiFile)));

                //Encabezado
                int NTRACKS = partitura.voces.size()+1;
                double midi_tempo = 1000000 * 60 * NotaEnum.valueOf(partitura.tempo.duracion).getDuracion()
                / (4 * partitura.tempo.cantidad);
                String midi_bar = String.format("%s/%s", partitura.indicacion.tiempos,
                partitura.indicacion.tipoNota);
                writer.println(String.format("MFile 1 %d 384", NTRACKS));
                writer.println(String.format("MTrk"));
                writer.println(String.format("000:00:000 Tempo %.0f",midi_tempo));
                writer.println(String.format("000:00:000 TimeSig %s 24 8",midi_bar));
                writer.println("000:00:000 Meta TrkEnd");
                writer.println("TrkEnd");

                int i = 1;
                for (Voz voz : partitura.voces) {
                    generarTrack(voz, i, partitura.indicacion, 384, writer);
                    i = i + 1;
                }

                writer.flush();
                writer.close();
            }
        }
    }
}
```

```

    }
}

}

private static void generarTrack(Voz voz, int voice_number,
    IndicacionCompas indicacion, int clicks_per_beat, PrintWriter writer) {
    writer.println("MTrk");
    writer.println(String.format("000:00:000 Meta TrkName \"Voz %s\"",voice_number));
    writer.println(String.format("000:00:000 ProgCh ch=%d prog=%d",voice_number,voz.instrumento));
    int bar_num = 0;
    int beat_num = 0;
    int click_num = 0;

    for(Compas compas : voz.compases) {
        for(Nota nota : compas.notas) {
            if(!nota.isSilencio()) {
                writer.println(String.format("%03d:%02d:%03d On ch=%s note=%s vol=70",
                    bar_num,beat_num, click_num,
                    voice_number, nota.notacionAmericana()));
                int click_figure = clicksPorFigura(nota.duracion, indicacion);
                int temp_click = click_num + click_figure;
                click_num = temp_click % clicks_per_beat;
                int temp_beat = beat_num + (temp_click / clicks_per_beat);
                beat_num = temp_beat % indicacion.tiempos;
                bar_num = bar_num + (temp_beat / indicacion.tiempos);
                writer.println(String.format("%03d:%02d:%03d Off ch=%d note=%s vol=0",
                    bar_num, beat_num,click_num,
                    voice_number, nota.notacionAmericana()));
            } else {
                int click_figure = clicksPorFigura(nota.duracion, indicacion);
                int temp_click = click_num + click_figure;
                click_num = temp_click % clicks_per_beat;
                int temp_beat = beat_num + (temp_click / clicks_per_beat);
                beat_num = temp_beat % indicacion.tiempos;
                bar_num = bar_num + (temp_beat / indicacion.tiempos);
            }
        }
    }
    writer.println(String.format("%03d:%02d:%03d Meta TrkEnd",bar_num, beat_num, click_num));
    writer.println("TrkEnd");
}

}

```