



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA



Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Trabajo Práctico 2

Compositor Musical

Teoría de Lenguajes

Primer Cuatrimestre de 2015

Grupo: Autores del Autómata Automático Autodestructivo

Apellido y Nombre	LU	E-mail
Matayoshi, Leandro	79/11	leandro.matayoshi@gmail.com
Panarello, Bernabé	FALTA LU	bpanarello@gmail.com
Vega, Leandro	698/11	leandrogvega@gmail.com

Índice

1. Introducción del problema a resolver	2
1.1. Paso a paso introductorio de la resolución	2
2. Descripción del problema resuelto	3
2.1. Errores surgidos durante la implementación	3
2.2. Lexer	3
2.3. Parser	3
2.4. Midi	3
3. Gramática	4
3.1. Gramática deducida	4
3.1.1. Tupla	4
3.1.2. Conjunto finito de terminales (V_t)	4
3.1.3. Conjunto finito de no terminales (V_n)	4
3.1.4. Producciones (P)	4
3.2. Tokens	5
4. Tests	6
4.1. Tests con fallas	6
4.1.1. Test 1: Tiene compases con distinta duración	6
4.1.2. Test 2: Tiene voces con compases de distinta duración	6
4.1.3. Test 3: Constante que apunta a una constante no definida	7
4.1.4. Test 4: Constante definida circularmente	7
4.2. Tests correctos	8
4.2.1. Test 1: Simple	8
4.2.2. Test 2: Con varias voces	8
4.2.3. Test 3: Con repeticiones	9
5. Manual del programa	11
5.1. Modo de uso	11
5.2. Opciones que acepta	11
5.3. Requerimientos necesarios para ejecutar	11
6. Conclusiones	12

1. Introducción del problema a resolver

El objetivo de nuestro tp es, dado un archivo de entrada, parsearlo para poder tener un archivo de salida, respetando restricciones solicitadas para el correcto funcionamiento.

1.1. Paso a paso introductorio de la resolución

- En primera instancia definiremos una gramática necesaria para interpretar, de manera correcta, nuestro archivo de entrada.
- Luego definiremos cada expresion regular como tokens en el lexer_rules, de esta manera cargamos en la computadora nuestras expresiones regulares definidas en la gramática.
- El siguiente paso será definir las producciones correspondiente, las cuales gracias a los tokens definidos en el paso anterior, podremos construir, diferenciarlas una de las otras y filtrar aquellas que no sean válidas.
- Cada producción llamará a su función interna, formando el árbol decado desde las hojas hasta su raiz, usando en cada una atributos sintetizados o heredados para poder intercambiar valores de una rama a la otra y poder validar las condiciones especificadas en nuestro lenguaje.
- Una vez corroborado y habiendo obtenido con éxito todo lo anterior, procederemos a escribir el archivo MIDI cumpliendo las normas del mismo sin dificultades.

2. Descripción del problema resuelto

En esta sección explicaremos cada parte implementada para realizar el tp. Contaremos dudas, errores que fueron surgiendo y explicaremos las decisiones tomadas. Para eso vamos a dividirlo en cuatro secciones que detallamos a continuación.

2.1. Errores surgidos durante la implementación

2.2. Lexer

2.3. Parser

2.4. Midi

3. Gramática

3.1. Gramática deducida

3.1.1. Tupla

$$G = (V_t, V_n, P, H)$$

3.1.2. Conjunto finito de terminales (V_t)

{ #tempo, #compas, /, ', ', ', ', =, (,), {, }, '. ', +, -, const, voz, compas, repetir, nota, silencio, blanca, negra, redonda, semicorchea, corchea, fusa, semifusa, do, re, mi, fa, sol, la, si }

3.1.3. Conjunto finito de no terminales (V_n)

{ H, TEMPO, COMPASHEADER, CONSTINIT, CONSTLIST, CONST, VOICELIST, VOICE, VOICECONTENT, COMPASLOOP, COMPASLIST, COMPAS, NOTELIST, NOTE, SILENCE, VALUE, SHAPE, NUM, CNAME, NOTENAME, ALTER }

3.1.4. Producciones (P)

$H \rightarrow \{ \text{TEMPO} \} \{ \text{COMPASHEADER} \} \{ \text{CONSTINIT} \} \{ \text{VOICELIST} \}$
 $H \rightarrow \{ \text{TEMPO} \} \{ \text{COMPASHEADER} \} \{ \text{VOICELIST} \}$
 $\text{TEMPO} \rightarrow \{ \text{tempobegin} \} \{ \text{shape} \} \{ \text{num} \}$
 $\text{COMPASHEADER} \rightarrow \{ \text{compasheaderbegin} \} \{ \text{num} \} \{ \text{slash} \} \{ \text{num} \}$
 $\text{CONSTINIT} \rightarrow \{ \text{CONSTLIST} \}$
 $\text{CONSTLIST} \rightarrow \{ \text{CONST} \}$
 $\text{CONSTLIST} \rightarrow \{ \text{CONSTLIST} \} \{ \text{CONST} \}$
 $\text{CONST} \rightarrow \{ \text{const} \} \{ \text{VALUE} \} \{ \text{equals} \} \{ \text{VALUE} \} \{ \text{semicolon} \}$
 $\text{VOICELIST} \rightarrow \{ \text{VOICE} \}$
 $\text{VOICELIST} \rightarrow \{ \text{VOICELIST} \} \{ \text{VOICE} \}$
 $\text{VOICE} \rightarrow \{ \text{voicebegin} \} \{ \text{leftpar} \} \{ \text{VALUE} \} \{ \text{rightpar} \} \{ \text{leftcurl} \} \{ \text{VOICECONTENT} \} \{ \text{rightcurl} \}$
 $\text{VOICECONTENT} \rightarrow \{ \text{COMPAS} \}$
 $\text{VOICECONTENT} \rightarrow \{ \text{COMPASLOOP} \}$
 $\text{VOICECONTENT} \rightarrow \{ \text{VOICECONTENT} \} \{ \text{COMPAS} \}$
 $\text{VOICECONTENT} \rightarrow \{ \text{VOICECONTENT} \} \{ \text{COMPASLOOP} \}$
 $\text{COMPASLOOP} \rightarrow \{ \text{loopbegin} \} \{ \text{leftpar} \} \{ \text{VALUE} \} \{ \text{rightpar} \} \{ \text{leftcurl} \} \{ \text{COMPASLIST} \} \{ \text{rightcurl} \}$
 $\text{COMPASLIST} \rightarrow \{ \text{COMPAS} \}$
 $\text{COMPASLIST} \rightarrow \{ \text{COMPASLIST} \} \{ \text{COMPAS} \}$
 $\text{COMPAS} \rightarrow \{ \text{compasbegin} \} \{ \text{leftcurl} \} \{ \text{NOTELIST} \} \{ \text{rightcurl} \}$
 $\text{NOTELIST} \rightarrow \{ \text{NOTE} \}$
 $\text{NOTELIST} \rightarrow \{ \text{SILENCE} \}$
 $\text{NOTELIST} \rightarrow \{ \text{NOTELIST} \} \{ \text{NOTE} \}$
 $\text{NOTELIST} \rightarrow \{ \text{NOTELIST} \} \{ \text{SILENCE} \}$
 $\text{NOTE} \rightarrow \{ \text{notebegin} \} \{ \text{leftpar} \} \{ \text{notename} \} \{ \text{alter} \} \{ \text{comma} \} \{ \text{VALUE} \} \{ \text{comma} \} \{ \text{shape} \} \{ \text{punto} \} \{ \text{rightpar} \} \{ \text{semicolon} \}$
 $\text{NOTE} \rightarrow \{ \text{notebegin} \} \{ \text{leftpar} \} \{ \text{notename} \} \{ \text{alter} \} \{ \text{comma} \} \{ \text{VALUE} \} \{ \text{comma} \} \{ \text{shape} \} \{ \text{rightpar} \} \{ \text{semicolon} \}$
 $\text{NOTE} \rightarrow \{ \text{notebegin} \} \{ \text{leftpar} \} \{ \text{notename} \} \{ \text{comma} \} \{ \text{VALUE} \} \{ \text{comma} \} \{ \text{shape} \} \{ \text{punto} \} \{ \text{rightpar} \} \{ \text{semicolon} \}$
 $\text{NOTE} \rightarrow \{ \text{notebegin} \} \{ \text{leftpar} \} \{ \text{notename} \} \{ \text{comma} \} \{ \text{VALUE} \} \{ \text{comma} \} \{ \text{shape} \} \{ \text{rightpar} \} \{ \text{semicolon} \}$
 $\text{SILENCE} \rightarrow \{ \text{silencebegin} \} \{ \text{leftpar} \} \{ \text{shape} \} \{ \text{rightpar} \} \{ \text{semicolon} \}$
 $\text{SILENCE} \rightarrow \{ \text{silencebegin} \} \{ \text{leftpar} \} \{ \text{shape} \} \{ \text{punto} \} \{ \text{rightpar} \} \{ \text{semicolon} \}$
 $\text{VALUE} \rightarrow \{ \text{cname} \}$
 $\text{VALUE} \rightarrow \{ \text{num} \}$

3.2. Tokens

- TEMPOBEGIN: # tempo
- CONST: const
- EQUALS: =
- SEMICOLON: ;
- VOICEBEGIN: voz
- LEFTPAR: (
- RIGHTPAR:)
- LEFTCURL: {
- RIGHTCURL: }
- COMPASHEADERBEGIN: # compas
- COMPASBEGIN: compas
- LOOPBEGIN: repetir
- SLASH: /
- NOTEBEGIN: nota
- SILENCEBEGIN: silencio
- PUNTO: .
- ALTER: +|−
- SHAPE: *blanca|negra|redonda|semicorchea|corchea|semifusa|fusa*
- NOTENAME: *do|re|mi|fa|sol|la|si*
- COMMA: ,
- CNAME: $(([a - z][A - Z])([0 - 9][a - z][A - Z])^*)$
- NUM: $([0][1 - 9][0 - 9]^*)$

4. Tests

4.1. Tests con fallas

4.1.1. Test 1: Tiene compases con distinta duración

```
1 #tempo negra 30
2 #compas 3/4
3
4 const oct1 = 2;
5 const oct2 = 6;
6 const oct3 = 1;
7
8 // Instrumentos
9 const flauta = 51;
10
11 voz ( flauta )
12 {
13   compas
14   {
15     nota(do, oct3, blanca.);
16     nota(re, oct1, redonda);
17   }
18
19   compas
20   {
21     nota(mi, oct2, blanca);
22     nota(la, oct1, negra);
23   }
24 }
```

4.1.2. Test 2: Tiene voces con compases de distinta duración

```
1 #tempo negra 120
2 #compas 2/2
3
4 const oct1 = 10;
5 const oct2 = 2;
6 const oct3 = 4;
7
8 // Instrumentos
9 const violin = 20;
10 const guitarra = 12;
11
12 voz ( violin )
13 {
14   compas
15   {
16     nota(do, oct3, blanca.);
17     nota(re, oct1, negra);
18   }
19
20   compas
21   {
22     nota(mi, oct2, blanca.);
23     nota(la, oct1, negra);
24   }
25 }
26
27 voz ( guitarra )
28 {
```

```
29 compas
30 {
31     nota(do, oct3, fusa);
32     nota(re, oct1, semifusa.);
33 }
34
35 compas
36 {
37     nota(mi, oct2, fusa);
38     nota(la, oct1, semifusa.);
39 }
40 }
```

4.1.3. Test 3: Constante que apunta a una constante no definida

```
1 #tempo negra 60
2 #compas 1/1
3
4 const oct1 = 3;
5 const oct2 = ConstanteTrucha;
6
7 // Instrumentos
8 const bajo = 20;
9
10 voz (bajo)
11 {
12     compas
13     {
14         nota(do, oct1, blanca.);
15         nota(re, oct1, negra);
16     }
17
18     compas
19     {
20         nota(mi, oct2, blanca.);
21         nota(la, oct2, negra);
22     }
23 }
```

4.1.4. Test 4: Constante definida circularmente

```
1 #tempo negra 60
2 #compas 2/8
3
4 const oct1 = 3;
5 const oct2 = 5;
6
7 // Instrumentos
8 const bajo = 20;
9 const malPensado = malPensado;
10
11 voz (bajo)
12 {
13     compas
14     {
15         nota(do, oct1, corchea.);
16         nota(re, oct1, semicorchea);
17     }
18
19     compas
20     {
```



```
21     nota(mi, oct2, semicorchea);
22     nota(la, oct2, corchea.);
23 }
24 }
```

4.2. Tests correctos

4.2.1. Test 1: Simple

```
1 #tempo negra 30
2 #compas 2/4
3
4 const oct1 = 2;
5 const oct2 = 6;
6 const oct3 = 1;
7
8 // Instrumentos
9 const flauta = 51;
10
11 voz ( flauta )
12 {
13     compas
14     {
15         nota(do, oct3, negra);
16         nota(re, oct1, negra);
17     }
18
19     compas
20     {
21         nota(mi, oct2, blanca);
22     }
23 }
```

4.2.2. Test 2: Con varias voces

```
1 #tempo negra 120
2 #compas 3/4
3
4 const oct1 = 2;
5 const oct2 = 6;
6 const oct3 = 1;
7 const oct4 = 3
8
9 // Instrumentos
10 const piano = 65;
11 const violin = 31;
12
13 voz (piano)
14 {
15     compas
16     {
17         nota(sol, oct3, blanca);
18         nota(fa+, oct2, negra);
19     }
20
21     compas
22     {
23         nota(mi, oct2, negra.);
24         nota(fa, oct4, corchea);
25         nota(mi, oct2, negra);
```

```

26 }
27 compas
28 {
29     silencio (negra);
30     nota(sol -, oct1, negra);
31     nota(sol -, oct1, negra);
32 }
33 }
34
35 voz ( violin )
36 {
37     compas
38     {
39         nota(la , oct1, blanca.);
40     }
41
42     compas
43     {
44         nota(mi, oct2, negra.);
45         nota(fa, oct4, corchea);
46         nota(mi, oct2, negra);
47     }
48     compas
49     {
50         silencio (semicorchea);
51         nota(mi, oct1, semicorchea);
52         nota(sol , oct4, corchea);
53         nota(sol , oct4, blanca);
54     }
55 }

```

4.2.3. Test 3: Con repeticiones

```

1 #tempo negra 120
2 #compas 2/4
3
4 const oct1 = 2;
5 const oct2 = 6;
6 const oct3 = 1;
7 const oct4 = 3
8
9 // Instrumentos
10 const flauta = 10;
11 const violin = 31;
12
13 voz ( flauta )
14 {
15     repetir (5)
16     {
17         compas
18         {
19             nota(sol , oct3, negra);
20             nota(fa+, oct2, negra);
21         }
22
23         compas
24         {
25             nota(mi, oct2, negra.);
26             nota(fa, oct4, corchea);
27         }
28         compas
29         {

```

```
30     nota(sol , oct1, blanca);
31   }
32 }
33 }
34
35 voz ( violin )
36 {
37   compas
38   {
39     nota(la , oct1, blanca);
40   }
41
42   compas
43   {
44     nota(mi, oct2, negra.);
45     nota(fa , oct4, corchea);
46     nota(mi, oct2, negra);
47   }
48   compas
49   {
50     silencio (semifusa );
51     nota(si +, oct1, semifusa);
52     nota(si , oct4, semifusa);
53     nota(fa , oct4, semifusa);
54     nota(sol , oct4, semicorchea);
55     nota(mi−, oct4, corchea);
56     nota(re, oct4, negra);
57   }
58 }
```

5. Manual del programa

5.1. Modo de uso

Aportar un archivo de entrada con extensión `mus`, el cual debe contener internamente: (no se si vale la pena detallar lo mencionado en el enunciado, pero si es un manual puede ser, empece a escribirlo y me entró la duda)

5.2. Opciones que acepta

Se aceptan archivos de entrada que contengan:

- Todos los compases deben tener la misma duración al sumar la duración de sus notas y/o silencios.
- Todas las voces deben tener la misma cantidad de compases.
- No deben existir constantes indefinidas o definidas circularmente. Ejemplo constante no definida: `const eval = hola;` (hola jamás se definió). Ejemplo constante definida circularmente: `const eval1 = eval2 ; const eval2 = eval1 ;`.
- La suma de la duración de cada compas debe ser igual a $num1/num2$, donde $num1$ y $num2$ son los números definidos en `#compas num1/num2` en el encabezado.

En caso contrario que no se respete lo mencionado anteriormente, nuestro programa especificará el error cometido para que pueda solucionarlo.

5.3. Requerimientos necesarios para ejecutar

- Programa: Python
- Versión: 2.7
- Línea de ejecución: `./musileng entrada.mus salida.txt`

6. Conclusiones