Práctica de Compiladores

A Music Language: Definición del lenguaje y gramática

Mario Fernández Villalba

Juan Miguel de Haro Ruiz

Carlos Roldán Montaner

Grupo 11

Primavera 2016-2017

Contents

Definición del lenguaje

1. Funcionalidades

A Music Lenguage (AML) es un lenguaje que permite la mezcla de notación musical y de un lenguaje imperativo inspirado en C++. Para alcanzar esto AML dispone de dos tipos de instrucciones: imperativas y musicales.

Las instrucciones imperativas son muy similares a las que implementan los principales lenguajes imperativos (C++, C, C, Java, etc.). Encontramos las típicas instrucciones de control de flujo: *if*, *for* y *while*, que operan con enteros, booleanos y strings. También disponemos de llamadas a funciones (que podrán devolver datos), declaraciones (con o sin asignación), declaraciones múltiples, asignaciones, operadores aritméticos (suma, resta, producto, etc.) y lógicos (AND, OR, NOT). Opcionalmente el lenguaje implementará vectores.

Las instrucciones musicales nos permiten la representación de diferentes elementos de la música. Éstos son notas, acordes, compases y canciones, que implementan métodos propios que permiten modificar sus atributos.

Los atributos de una nota son la tonalidad y la figura. La unidad básica de tonalidad es el semitono, y es tratada como un entero. AML reconoce las figuras redonda, blanca, negra, corchera, semi-corchera, fusa y semi-fusa; que son representadas con sus iniciales. Éstas también son representadas con enteros. Sobre una nota podemos realizar operaciones que modifiquen su tonalidad (sostenidos, bemoles, etc.) y su figura rítmica (ligaduras, puntos, etc.). También existe la opción de representar un conjunto de notas.

Un acorde, dada una serie de parámetros (nota base, modo, etc.) crea un conjunto de notas que serán reproducidas a la vez y que representan el acorde descrito. El acorde dispone de métodos que nos permiten modificar sus distintos atributos.

Las canciones representan composiciones musicales formadas por pistas, de las que hablaremos más adelante. Los atributos de una canción son su tempo, su métrica y su tonalidad. El tempo viene dado en *Beats Per Minute* (BPM). La métrica es representada con el formato *n:m*, es decir, en un compás debe haber **n** figuras de duración **m/4** negras. La tonalidad viene dada por el número de sostenidos o bemoles de ésta. Estos atributos pueden ser modificados con instrucciones musicales. Una canción puede ser ejecutada en un momento dado con la función *Play*.

Para realizar todas estas operaciones sobre estos tres tipos de elementos musicales, se utiliza la notación *dato_musical.elemento*. Podemos imaginar estos tipos como si fueran clases, utilizando la notación '.' para acceder a sus métodos y a sus atributos (y modificarlos).

Estos tres elementos (notas, acordes y canciones) pueden ser almacenados en variables, pero aún así existen más elementos que aparecen implícitamente en el código. Éstos son pistas y compases.

Una pista está formada por compases, y es interpretada por un instrumento. Un instrumento puede ser especificado por una string al principio de la pista, que ha de contener un nombre válido de la especificación general MIDI.

Un compás es un conjunto de notas que puede contener tanto instrucciones imperativas como musicales. Dentro de un compás no puede haber otros compases. No obstante, es posible llamar desde dentro de un compás a una función que contenga compases. También es posible

introducir dentro de un compás fragmentos de compases. Un fragmento es una función especial que solo puede contener todo lo que pueda contener un compás. La métrica de un compás sera comprobada automáticamente, y se tendrán en cuenta los fragmentos para esta comprobación. AML da la posibilidad de repetir compases y de finales alternativos usando la variable *Time*, que contiene la iteración en la que se encuentra el compás. Un compás no ha de estar situado necesariamente dentro de una canción.

AML interpreta el hilo principal de ejecución de un programa como una pista. Ésta tendrá un tempo, métrica, tonalidad y instrumento propios que por defecto son 120 bpm, 4:4, Do mayor y "Grand Piano" respectivamente. Toda canción que no especifique alguno de estos atributos lo heredará del hilo principal. Opcionalmente habrá control de volumen y interacción con el usuario mediante la consola.

2. Arquitectura

AML será un lenguaje tanto interpretado como compilado. Se le dará al usuario la opción de reproducir directamente un fichero de código o la opción de escribirlo en un fichero MIDI, que podrá ser reproducido más tarde. Para ello utilizaremos una librería implementada en Java, *javax.sound.midi*.

Ejemplos de uso

1. Instrucciones imperativas

```
void main() {
    int var1, var2;
    bool var3, var4;
    var3 = true;
    int var5 = 2, var6 = 4+5, var7;
    bool var8 = false;

    var1 = 1;
    while (var1 > 0) {
        --var1;
    }

    for (int i = 9; i >= var1; ++i) {
        var3 = true;
     }

    var2 = funcion1(var3, var4);


}

int funcion1(int var1, bool var2) {
    /* Do something */
    if (var2) {
        return var1;
    }
    else {
        return 0;
    }
}
```

Figure 1: jp1.music

2. Instrucciones musicales: notas

```
  void main() {
    Note note = La.n, notes;
    notes = (Do Re Mi Fa Sol).c;

    note.raiseTone(2);
    notes.isQuiet();
    notes.figure = 2 + 3;
    note.tone = 2 + notes.figure;
}
```

Figure 2: jp2.music

3. Instrucciones musicales: acordes

```
void main() {
    Chord C = Chord(Do 7th);
    Chord Dm7 = Chord(Re m 7th);
    C.alterMode("augmented");
}
```

Figure 3: jp3.music

4. Instrucciones musicales: canciones

```
void main() {
    Song song {
        Beat 4:8;
        Speed 60;
        Tone 3&;
        Transport 5;
        Track "Sitar"  || Sol-1.c Sol-2.c Sol-3.c Sol-4.c ||
    }
}
```

Figure 4: jp4.music

5. Instrucciones musicales: pistas

```
void main() {
     Song {
         Beat 4:8;
         Speed 60;
         Tone 3&;
         Transport 5;
         Track "Sitar"  || Sol-1.c Sol-2.c Sol-3.c Sol-4.c ||
         Track  || Sol-1.c Sol-2.c Sol-3.c Sol-4.c ||
         Track "Celesta"  || Sol-1.c Sol-2.c Sol-3.c Sol-4.c ||

     }
 }
```

Figure 5: jp5.music

6. Instrucciones musicales: compases

```
fragment frag () {
    while (true) { Re.n; rr = 42; if (ch) {tt = 3; La} else { Re.n } }
}
void main() {
     3x||: La.c Do Re.c | if (d == 3) { La } :|| La Do.n ||
     Song {
         Beat 4:8;
         Speed 60;
         Transport 5;
         Track "Trumpet" || &Do.c Do.c &Do.c Re.c | if (Times == 2) { La.n }
         else if (Time == 2) { La.n Re.n (Do Si); int ex = 3; } | frag();||
     }
}
```

Figure 6: jp6.music

7. Instrucciones musicales: hilo

```
void main() {

     Beat 4:8;
     3x||: La.c Do Re.c | if (d == 3) { La } :|| La Do.n ||

     Song song {
         Beat 4:8;
         Speed 60;
         Transport 5;
         Track "Trumpet" || &Do.c Do.c Do.c Re.c | if (Times == 2) { La.n }
         else if (Time == 2) { La.n Re.n (Do Si); int ex = 3; }||
     }

     song.play();
}
```

Figure 7: jp7.music