Especificação da Linguagem

Grupo 12, LFA
Universidade de Aveiro

António Santos
Beatriz Borges
Gustavo Inácio
Filipe Vale
Tomás Freitas

Especificação da Linguagem

Grupo 12, LFA

Junho 2018

Conteúdo

1 Estruturas de dados (e tipos de variáveis)

1.1 Sequências

Uma sequência de notas e silêncios pode ser definida, através da palavra chave `sequence`, como:

```
sequence melody = [C C G G A A G R F F E E D D C R]; // os espacos
    sao opcionais
```

1.1.1 Notas musicais

A notas são identificadas pela sua letra:

- `C` para Dó,
- `D` para Ré,
- `E` para Mi,
- `F` para Fá,
- `G` para Sol,
- `A` para Lá,
- `B` para Si.

A letra `R` é usada para silêncios (derivada de *Rests*).

Um cardinal (`#`) a seguir à letra sobe a respetiva nota por meio tom; um b minúsculo (`b`) a seguir à letra desce a respetiva nota por meio tom. Mais que um `#` ou `b` podem ser aplicados a uma nota (por exemplo, `C##`).

Pode, depois do tom, aparecer um número, entre 0 e 8. Este especifica a oitava a usar, sendo 0 a mais grave e 8 a mais aguda. Por omissão, a quarta oitava é usada.

Assim, a sequência já apresentada podia reescrita da seguinte forma:

```
sequence melody = [C4 B# G G4 A4 A G R F E#4 Fb4 E D D4 C4 R];
```

Finalmente, uma sequência pode ser definida a partir doutra sequência:

```
sequence melody = [C C G G A A G R F F E E D D C R];

sequence r = [E G E melody D E D C]; // so e possivel utilizar esta
    nomenclatura se 'melody' for uma sequence
// equivalente a:
// sequence r = [EGE CCGGAAGR FFEEDDCR DEDC];
```

1.1.2 Duração duma nota

Por omissão, uma nota demora um tempo[1]. A sua duração depende do *tempo*, ou `BPM`[2], da música. Esta configuração é explorada na secção 5.

Uma nota pode, no entanto, tocar mais ou menos tempo. Há duas formas de especificar a duração duma dada nota:

1. **Por extensão.** A duração da nota é especificada através de chavetas ({ e }), relativamente à duração unitária. Por exemplo, `C{4}` demora o quádruplo do tempo de `C`, e `C{0.75}` demora três quartos do tempo de `C`.

2. **Notação simplificada.** Utilizam-se apóstrofos para reduzir a duração duma nota em metade, havendo uma correspondência direta com a duração das notas musicais convencionadas [3]. Por exemplo, `C'` demora metade do tempo de `C`, e `C''` apenas um quarto.

1.1.3 Acordes

O símbolo `|` é usado para tocar várias notas em simultâneo, numa só sequência. Para tocar o acorde de Dó maior (C, E e G), na quarta oitava, podíamos então escrever:

```
1 sequence intro = [C|E|G];
```

1.2 Performances

A associação duma sequência musical com um instrumento representa um terceiro tipo de dados, uma `performance`:

```
1 sequence twinkle = [CC GG AA G{2} FF EE DD C{2}];
2 performance p = twinkle on guitar;
3
4 // ou, alternativamente, definindo a sequencia implicitamente
5 performance p = [CC GG AA G{2} FF EE DD C{2}] on guitar;
```

1.3 Números

Números (inteiros ou reais) são também suportados:

```
1 number num = 4;
```

Números podem também ser usados para referenciar um dado momento entre o início e o fim, inclusive, da peça musical. O início da peça é representado pelo número 0.

[1] Em termos musicais, uma semínima.
[2] *Beats Per Minute*
[3] Semínima (duração de 1), colcheia (duração de 1/2), semicolcheia (duração de 1/4), etc.

1.3.1 Operador de duração

O operador $|x|$, aceita um parâmetro (x), do tipo sequence ou performance, e devolve um número igual à sua duração.

```
sequence intro = [R{4} C{4} G{4} C5{3.5} E|G|C5{.5} Eb|G|C5{8} C{4}
    G{4}]; // Strauss - Also Sprach Zarathustra - Intro (https://
    www.8notes.com/scores/7213.asp)
number endIntro = |intro|;
```

1.4 Instrumentos

Para tocar sequência musical é, naturalmente, necessário especificar que instrumento deve ser utilizado. Assim, para tocar *Twinkle, Twinkle, Little Star* com uma piano, teríamos:

```
// definir a sequencia
sequence twinkle =  [CC GG AA G{2} FF EE DD C{2}];

// utilizando performances
performance p = twinkle on piano;
```

Vários instrumentos podem ser usados para tocar uma dada sequência. Os instrumentos disponíveis são os seguintes:

- `piano`;
- `guitar`;
- `violin`;
- `cello`;
- `bass`;
- `drums`.

A criação de novos instrumentos é suportada, num ficheiro externo de tipo auxiliar, estando detalhada na secção 5. No ficheiro de tipo principal, não é possível definir ou redefinir novos instrumentos, sendo apenas possível usá-los como constantes pré-definidas.

1.5 Arrays

Um array é uma coleção de várias instâncias da mesma estrutura de dados. Suportam-se arrays de sequências, instrumentos, performances e números.

Um array pode ser definido de duas formas distintas:

```
// criar um array com 4 instrumentos - e necessario usar a palavra
    chave instrument para indicar que e um array de instrumentos
instrument[] band = [piano, guitar, bass, drums];

// outra forma de definir o mesmo array
instrument[] band = piano and guitar and bass and drums;
```

```
6
7  // criar um array com 3 performances
8  performance[] p = [
9      [D{1.5} D{0.5}   E  D G F#{2}] on piano,
10     [D{1.5} D{0.5}   E  D A G{2}] on bass,
11     [D{1.5} D{0.5}   D5 B G F# E] on guitar];
```

Pode usar-se um array para criar outro. A palavra chave `and` anexa ao fim do array já existente o elemento ou elementos dados.

```
1  instrument[] band = [piano, guitar, bass, drums];
2
3  // definir arrrays a partir de outros arrays
4  instrument[] new_band = band and violin;
```

Para arrays de números, o operador `a->b`, que devolve um array de inteiros de tipo `[a, a+1, ..., b-1, b]` pode também ser utilizado para gerar um array.

```
1  start_times = 0->3;
2  // equivalente a:
3  //    start_times = [0, 1, 2, 3];
```

Para se aceder a um instrumento, a notação de parênteses retos, começando a contar no 0, é usada. A notação de intervalo, com dois pontos (`:`) é também suportada.

```
1  performance[] p = [
2      [D{1.5} D{0.5}   E  D G F#{2}] on piano,
3      [D{1.5} D{0.5}   E  D A G{2}] on bass,
4      [D{1.5} D{0.5}   D5 B G F# E] on guitar];
5
6  // aceder ao primeiro elemento
7  performance intro = p[0];
8
9  // exemplo da notacao de intervalo
10 performance[] q = p[0:1] and [C5{1.5} C5{0.5} B G A G{3}] on violin
       and p[2:];
```

2 Geração de aúdio

2.1 Reprodução

Para reproduzir uma performance, utiliza-se a palavra chave `play`:

```
1  // definir uma performance
2  sequence twinkle =  [CC GG AA G{2} FF EE DD C{2}];
3  performance p = twinkle on piano;
4
5  // reproduzir a performance
6  play p;
7
8  // alternativamente, definir a performance implicitamente
9  play twinkle on piano;
10
11 // ou definir a performance e a sequencia implicitamente
12 play [CC GG AA G{2} FF EE DD C{2}] on piano;
```

2.2 Modos de reprodução

No exemplo anterior, não foi especificado quando começar a tocar a sequência. Por omissão, a sequência começa a ser tocada no início da peça (por outras palavras, no tempo 0).

Averiguemos os diferentes modos de reprodução:

2.2.1 Simultâneo

Por omissão, todas as sequências são tocadas começando no tempo 0. Se há mais que uma sequência a ser tocada, todas as sequências são tocadas em paralelo.

```
play [CC GG AA G{2} FF EE DD C{2}] on piano;

// e equivalente, em termos do som produzido no ficheiro final, a
play [CC RR RR G{2} FR RE DR C{2}] on piano;
play [RR GG AA RR RF ER RD RR] on piano;
```

Um outro exemplo de reprodução simultânea é o seguinte, que separa melodia e harmonia em duas performances diferentes:

```
// definir sequencias
sequence twinkle = [CC GG AA G{2} FF EE DD C{2}];
sequence twinkle_bass = [C|E|G{2} F|A|C{2} C|E|G{4} F|A|C C|E|G G|B
    |D C|E|G];

// tocar performances
play twinkle on guitar;
play twinkle_bass on guitar;
```

2.2.2 Usando Números

A palavra chave `at` indica um número específico no qual a performance deve começar, independentemente de haver outras performances a decorrer nesse momento.

```
performance verse = [CC E{2} GG B{2} C5C5 GG C{4}] on violin;
performance chorus = [GAGA ABBA GEGE EBBE] on violin;

number chorusStart = |verse|;

// tocar performances
play verse;
at chorusStart play chorus;
```

Uma performance pode ser tocada em mais que um momento.

```
performance verse = [CC E{2} GG B{2} C5C5 GG C{4}] on violin;
performance chorus = [GAGA ABBA GEGE EBBE] on violin;

number chorusStart = 2*|verse|;
number otherTimeChorusStarts = 3.14*|verse|;

// tocar performances (chorus e tocada 2 vezes)
play verse;
at chorusStart play chorus;
at otherTimeChorusStarts play chorus;
```

2.2.3 Repetição

Há duas palavras chaves que permitem a repetição: o uso da palavra chave `times` permite repetir uma performance 0 ou mais vezes. `loop` permite repetir uma performance até ao fim da duração atual da peça (determinada pelo fim mais tardio de qualquer outra sequência até aí adicionada).

```
performance verse = [CC E{2} GG B{2} C5C5 GG C{4}] on violin;
performance chorus = [GAGA ABBA GEGE EBBE] on violin;
performance bass = [G|B|D{4} A|D|F#{4} G|B|D{4} A|D|F#{4}] on bass;

number chorusStart = 2*|verse|;

// tocar performances
play verse on piano 2 times;
at chorusStart play chorus;

loop bass; // repete ate ao fim da musica
```

2.3 Uso de arrays

Um array pode ser reproduzido de forma simultânea (forma utilizada por omissão) ou sequencialmente (através do uso da palavra chave `sequentially`).

```
// exemplo com sequencias
sequence[] melody_lines = [
    [D{1.5} D{0.5}   E  D G F#{2}],
    [D{1.5} D{0.5}   E  D A G{2}],
    [D{1.5} D{0.5}   D5 B G F# E],
    [C5{1.5} C5{0.5} B  G A G{3}]];

play melody_lines /*sequentially*/ on piano; // descomentar
    sequentially para tocar melody_lines de forma sequencial
```

```
// exemplo com instrumentos
instrument[] band = [piano, guitar, bass, drums];

play melody_lines[0] /*sequentially*/ on band;
```

```
// exemplo com performances
performance[] perfors = [
    [D{1.5} D{0.5}   E  D G F#{2}] on piano,
    [D{1.5} D{0.5}   E  D A G{2}] on bass,
    [D{1.5} D{0.5}   D5 B G F# E] on guitar];

play perfors /*sequentially*/;
```

No entanto, não é possível reproduzir diretamente um array de sequências num array de instrumentos. Isto é, por exemplo, não é possível fazer o seguinte:

```
// definir array de sequencias
sequence[] melody_lines = [
    [D{1.5} D{0.5}   E  D G F#{2}],
    [D{1.5} D{0.5}   E  D A G{2}],
    [D{1.5} D{0.5}   D5 B G F# E],
    [C5{1.5} C5{0.5} B  G A G{3}]];
```

```
7
8  // definir array de instrumentos
9  instrument[] band = [piano, guitar, bass, drums];
10
11 play melody_lines /*sequentially*/ on band; // ilegal!!
```

Para se obter este tipo de resultados, têm que ser utilizadas instruções de repetição (ver secção 4).

2.4 Modulações

Podem obter-se versões modificadas de sequências ou performances através dos operadores de modulação. Estes operadores devolvem uma nova sequência ou performance, alterada em algum aspeto (tom ou tempo) em relação a uma dada sequência ou performance original, respetivamente.

2.4.1 de Tom

Pode mudar-se o tom de uma dada sequência ou performance através dos operadores `+` e `-`. A sequência ou performance devolvida será a sequência ou performance original com todas as suas notas subidas ou descidas n meios-tons[4].

```
1  // sequencia original
2  sequence s = [D{1.5} D{0.5} E D G F#{2}];
3
4  sequence s1 = s - 36; // diminuir a oitava por 3 (36 = 3*12)
5  // equivalente a dizer:
6  //    sequence s1 =  [D1{1.5} D1{0.5} E1 D1 G1 F#1{2}];
7
8  // mudar oitava da sequencia
9  sequence s5 = s + 12; // aumentar a oitava por 1 (12 = 1*12)
10 // equivalente a dizer:
11 //    sequence s5 =  [D5{1.5} D5{0.5} E5 D5 G5 F#5{2}];
```

```
1  // performance original
2  performance p = [D{1.5} D{0.5} E D G F#{2}] on bass;
3
4  play p+5;
5  // equivalente a:
6  //    play  [F#{1.5} F#{0.5} G# F# B A#{2}] on bass;
```

2.4.2 de Tempo

Pode mudar-se o tempo (ou seja, a velocidade) de um dada sequência ou performance através dos operadores `*` e `/`. A sequência ou performance devolvida será a sequência ou performance original com todas as suas notas aceleradas ou atrasadas n vezes.[5]

[4] Doze meios-tons constituem uma oitava.

[5] A nova velocidade é dada por `velocidade original * fator`, ou por `velocidade original / fator`, conforme os operadores `*` ou `/` são usados, respetivamente.

```
// performance original
performance p = [D{1.5} D{0.5} E D G F#{2}] on bass;

play p * 5; // toca a sequencia 5x mais rapido (cada nota dura 1/5
    do seu tempo original)
// equivalente a:
//    play  [D{.3} D{0.1} E{.2} D{.2} G{.2} F#{.4}] on bass;
```

```
// performance original
performance p = [D{1.5} D{0.5} E D G F#{2}] on bass;

play (p + 5) * 5; // toca a sequencia 5x mais rapido, com todas as
    notas 5 meios-tons mais agudas
// equivalente a:
//    play  [F#{.3} F#{0.1} G#{.2} F#{.2} B{.2} A#{.4}] on bass;
```

3 Interação com o exterior

3.1 Estruturas de dados auxiliares

3.1.1 Strings

Strings são sequências de caracteres, números e símbolos delimitadas por aspas (").

Não existe um tipo de dados String explícito, sendo este usado apenas como parâmetro opcional para funções de I/O.

3.2 getInt(string?)

`getInt()` permite obter um inteiro através do *Standard In*.

Opcionalmente, pode ser passada uma String, que será impressa antes de aguardar a resposta do utilizador (uma String de *prompt*).

4 Controlo de fluxo

4.1 Instruções condicionais

4.1.1 if

As palavras chave `if`, `else if` e `else` permitem testar condições. Os operadores suportados numa condição são os de igualdade (`==`), desigualdade (`!=`), menor (`<`), maior (`>`), menor ou igual (`<=`), e maior ou igual (`>=`).

```
sequence s = getSequence("Enter a sequence: ");

if (|s| > 5) {
    play s on piano;
} else if (|s| > 2) {
    play s on cello;
} else {
    play s on guitar;
```

```
}
```

4.2 Intruções de repetição

4.2.1 for

As palavras chave `for` e `in` permitem definir instruções de repetição, ou seja, permitem que um dado código seja executado múltiplas vezes, iterando sobre todos os elementos de um dado array.

```
// instrumentos
instrument[] band = [piano, guitar, bass, drums];
for instrument inst in band {
    play [ABCDCBA] on inst;
}

// sequencias
sequence[] sequences = [
    [D{1.5} D{0.5}   E  D G F#{2}],
    [D{1.5} D{0.5}   E  D A G{2}],
    [D{1.5} D{0.5}   D5 B G F# E],
    [C5{1.5} C5{0.5} B  G A G{3}]];

for sequence seq in sequences {
    play seq on piano;
}

// performances
performance[] perfors = [
    [D{1.5} D{0.5}   E  D G F#{2}] on piano,
    [D{1.5} D{0.5}   E  D A G{2}] on bass,
    [D{1.5} D{0.5}   D5 B G F# E] on guitar];

number t = 0;
for performance perfor in perfors {
    at t play perfor;
    t = t + |perfor|;
}

// numeros
start_times = [0, 1, 3, 7];
for number t in start_times {
    at t play [C1 E1 G1 E1] on piano;
}
```

Caso se pretenda iterar sobre algum código um dado número de vezes, sem haver correspondência direta entre esse número e o conteúdo de um dado array, pode usar-se números num dado intervalo, usando o operador `a->b` (que devolve um array de inteiros de tipo `[a, a+1, ..., b-1, b]`):

```
instrument[] band = [piano, guitar, bass, drums];
sequence[] sequences = [
    [D{1.5} D{0.5}   E  D G F#{2}],
    [D{1.5} D{0.5}   E  D A G{2}],
    [D{1.5} D{0.5}   D5 B G F# E],
    [C5{1.5} C5{0.5} B  G A G{3}]];
```

```
7
8  number t = 0;
9  for number i in 0->3 {
10     at t play sequences[i] on band[i];
11     t = t + |sequences[i]|;
12 }
```

5 Configurações (ficheiro auxiliar)

Um ficheiro do tipo principal (ficheiros com extensão `.mux`) suporta um todas as operações descritas até este ponto. Várias configurações podem ser feitas no ficheiro auxiliar (ficheiros com extensão `.aux`). Nesta secção, vamos abordar as diferentes configurações que podem ser definidas através do ficheiro auxiliar.

5.1 BPM (Beats Per Minute)

`BPM` é uma palavra reservada[6] usada para configurar o *tempo* da música. A configuração deve ser feita no ficheiro de configuração, mas é possível reescrevê-la no ficheiro principal (o que define a música a gerar).

```
1 BPM = 160;
```

5.2 Especificação de (novos) instrumentos

O formato *midi* suporta 128 instrumentos diferentes[7].

É possível criar novos instrumentos associando a uma palavra um número, através da palavra chave `instrument`. O nome dado ao instrumento deixa de poder ser usado como nome de variável.

```
1 // definir dois novos instrumentos
2 instrument strings: 49;
3 instrument synth1: 81;
```

Os instrumentos já definidos estão associados aos seguintes códigos: 1(`piano`), 25(`guitar`), 41(`violin`), 43(`cello`), 44(`bass`), e 119(`drums`).

É também possível definir novos instrumentos à custa de instrumentos já existentes, e associar nomes a tons.

```
1 // duplicar um instrumento ja existente
2 instrument percussion: drums;
3
4 // mapear tons (associar nomes a tons)
5 acousticBassDrum = B0;
6 bassDrum1 = C1;
7 sideStick = C#1;
8 acousticSnare = D1;
9 handClap = D#1;
```

[6] BPM não pode ser usado como nome de uma variável.

[7] Para obter mais informação sobre os diferentes instrumentos disponíveis, ver `https://www.midi.org/specifications/item/gm-level-1-sound-set`.

```
electricSnare = E1;
lowFloorTom = F1;
closedHiHat = F#1;
highFloorTom = G1;
pedalHiHat = G#1;
lowTom = A1;
openHiHat = A#1;
lowMidTom = B1;
hiMidTom = C2;

// a um tom podem ser associados varios nomes
// (mas a um nome nao podem ser associadas mais que um tom)
LongWhistle = C4;
MiddleC = C4;
```

É ainda possível criar um instrumento juntando vários instrumentos já existentes. Associa-se a uma nota (ou a um conjunto de notas, sendo o intervalo representado por `NotaInicial - NotaFinal`, incluindo os extremos) o instrumento que deve ser utilizado para a tocar, utilizando o operador `->`.

Se uma nota for definida mais que uma vez, a última definição será a usada. Se uma nota não for definida, gerará um erro se for reproduzida.

```
// definir um novo instrumento a partir de instrumentos existentes
instrument thing: B0–C4 -> drums,
                  G4 -> guitar,
                  A4–G5 -> guitar,
                  C5–A8 -> piano; // notas C5–G5 redifinidas
```

6 Especificação da linguagem destino

6.1 Linguagem destino

A linguagem destino do compilador foi Python 3, porque permite um maior foco na linguagem e no compilador, uma vez que facilita a manipulação de ficheiros do tipo MIDI através da biblioteca MIDIUtil.

Na geração de código utiliza-se, para cada performance, uma lista bastante detalhada, que permite a inserção modular de qualquer sequência de notas musicais ou acordes, suportando um número arbitrário de repetições e uma grande flexibilidade no tempo de inicio de reprodução da performance. Esta lista é parâmetro de entrada da função *addnotes* que permite a inserção das notas, com os instrumentos certos no sítio certo da timeline. Uma lista exemplo é a seguinte:

```
toadd = [[1,10,15,25,36], [(64,0.5,2,3),(62,0.25,25,4)], 3];
```

O primeiro elemento da lista é um *Array* com os start times da performance. No exemplo acima, o array implica que a performance comece a tocar nos segundos 1, 10, 15, 25 e 36. Ainda no mesmo exemplo, o segundo elemento representa a descrição das notas que são `tuplos` constituídos da seguinte maneira: (nota, duração, instrumento, tempo de entrada relativo ao start time). O último ele-

mento é o número de vezes que a performance é repetida em cada momento de inserção (com 5 start times, a performance será reproduzida 15 vezes).

7 Exemplos

7.1 Parabéns

bday.aux

```
// set default BPM setting
BPM = 200;
```

bday.mux

```
// get user input
number repeat_times = getInt("Number of repetitions: ");

// define melody sequences
sequence[] melody_lines = [
    [D{1.5} D{0.5}   E  D G F#{2}],
    [D{1.5} D{0.5}   E  D A G{2}],
    [D{1.5} D{0.5}   D5 B G F# E],
    [C5{1.5} C5{0.5} B  G A G{3}]];

// perform
number offset = 0;
for number n in 1->repeat_times {
  at offset play melody_lines + 12*(n-1) on piano sequentially;
  for sequence melody in melody_lines {
    offset = offset + |melody|;
  }
}
```

7.2 Stromae - Alors on Danse

rap.mux

```
sequence alors = [G#2{.5}R{.5} C#3{.5}R{.5} G#3{1.2}G#3{1.2}G#3{.8}
     E3{.5}R{.5} A3{.5}R{.5} Eb3{1.2}Eb3{1.2}Eb3{.8}] + 12;
sequence alors2 = [E3 R{.33} E3 R{.33} E3 R{.33} E3 R{.33} E3 R
    {.33} E3{.5} F#3{.5} R{.1} Eb3{.5}R{.5} Eb3{.5}R{.5}Eb3{.5}]+
    24;

performance p_alors = alors on piano;

//play alors on piano;

at 2* |alors| play alors2 on piano;
at 3 * |alors| play alors2 on piano;
loop p_alors;
```

O compilador é robusto e flexível, suportando a ausência do ficheiro auxiliar. Nesse caso, o compilador assume uma configuração o equivalente ao seguinte ficheiro auxiliar:

default.aux

```
BPM = 120;
```

7.3 Stranger Things Theme

StrangerThings.aux

```
BPM = 160;

//definicao dos instrumentos a usar pela track
instrument vibrofone    : 12;
instrument synth        : 39;
instrument synthPiano   : 3;
instrument slapBass     : 37;
instrument harpsiChord  : 7;
```

StrangerThings.mux

```
//Parte 1

sequence intro1 = [B4{0.25} E5{0.25} G5{0.25} B5{0.25} C6{0.25} B5
    {0.25} G5{0.25} E5{0.25}];
sequence intro2 = [R{0.25} B4{0.25} B4{0.25} B4{0.25} B4|C5{0.25}
    B4{0.25} B4{0.25}];
sequence intro3 = [R{0.5} G4{0.25} R{0.25}];
sequence intro4 = [E4{0.25}];

play intro1 on vibrofone 4 times;
play intro2 on synth 4 times;
play intro3 on synth 8 times;
play intro4 on synth 32 times;

//Parte 2
number parte2 = 4*|intro1|;

sequence bouncer = [C4''R'' E4''R'' G4''R'' B4''R'' C5''R'' B4''R''
     G4''R'' E4''R''];
sequence ticker1  = [C3''R''C3''R{1.25}];
sequence ticker2  = [E3''R''E3''R{1.25}];

at parte2 play bouncer on slapBass 30 times;
at parte2 play bouncer on synthPiano 30 times;
at parte2 play ticker1 on slapBass 8 times;

number t1 = parte2 + 8*|ticker1|;
at t1 play ticker2 on slapBass 8 times;

sequence danglingNotes1 = [C5''R{1.75} G5''];
number t2 = t1 + 8*|ticker2|;
at t2 play danglingNotes1 on harpsiChord;
at t2 play ticker1 on slapBass 7 times;

sequence ticker3 = [D3''R''D3''R{1.25}];
sequence danglingNotes2 = [D4''R''D4''R{1.25} E4''R{3.75} E3''R
    {9.75}G4''R''G4''R{1.25}C4''];
number t3 = t2 + 7*|ticker1|;

at t3 play danglingNotes2 on harpsiChord;
at t3 play ticker3 on slapBass;
```

```
38
39 number t4 = t2 + 8*|ticker1|;
40 number t5 = t4 + 7*|ticker1|;
41
42 at t4 play ticker2 on slapBass 7 times;
43 at t5 play [G3''R''G3''] on slapBass;
44
45
46 sequence ticker10 = [C3''R''C3''R''];
47 number t6 = t4 + 8*|ticker1|;
48 at t6 play ticker1 on slapBass 6 times;
49 at t6 play ticker10 on synth 12 times;
50
51 number t7 = t6 + 12*|ticker10|;
52 sequence ticker4 = [D3''R'' D3''R'' R{0.5}C3''R'' C3''R''C3''R'' R
       {0.5}B2''R''];
53 sequence ticker5 = [D3''R'' D3''R'' C3''R''C3''R''C3''R''C3''R'' B2
       ''R''B2''R''];
54 at t7 play ticker4 on slapBass;
55 at t7 play ticker5 on synth;
56
57
58 sequence ticker6 = [B2''R''B2''R''R];
59 sequence ticker7 = [B2''R''B2''R''];
60 number t8 = t7 + |ticker4|;
61 at t8 play ticker6 on slapBass 12 times;
62 at t8 play ticker7 on synth 24 times;
63
64
65 number t9 = t8 + 12*|ticker6|;
66 at t9 play ticker1 on slapBass 8 times;
67 at t9 play ticker1 on synth 8 times;
68 at t9 play [C5''R{1.75}G5''R{1.75}C5''R{7.75}D5''R{1.25}C5''R{1.25}
       B4|E4''] on harpsiChord;
69
70 number t10 = t9 + |ticker1|;
71 at t10 play [G3''R'']on synth 26 times;
```

7.4 John Chambers - Beethoven's Favourite Waltz

waltz.aux

```
BPM = 320;

instrument flute: 74;
instrument strings: 45;
```

waltz.mux

```
// beethoven's favourite waltz

sequence waltz = [
  B C5

  D5 B G{2} B C5
  D5 B G{2} G5 F5
  E5{2} E5{2} E5{2}
  E5{3} D5 C5 B

  A{2} A{2} A{2}
  D5{3} C5  C5 A
  G F G B A F
  G4 D5 F5

  G5{2} G5{2} A5{2}
  G5{2} G5{2} A5{2}

  G5{2} A5{2} B5{2}
  A5 F5 D5{2} E5 F5

  G5 F5 G5 D5 F5 D5
  G5 F5 G5 D5 F5 D5
  G5{2} D5 C5 B A
  G4 G B

  D5{2} D#5 C5 D5 C5
  D5 B E5 D5 C5 B
  A{2} A{2} D5{2}
  B{2} R{2} G B

  D5{2} D5 C#5 D5 C5

  D5 BE5 D5 C5 B
  A{2} A G A B
  G4 D5{2}

  G5{2} G5{2} B5 G5
  D5{2} D5 G5 D5 B
  G{2} G F G B
  B{2} A{2} D5 F5

  G5{2} G5{2} B5 G5
  D5{2} D5{2} G5 D5
  B D5 G B A F
```

```
  G{4}
];

at 2 play waltz + 12 on piano;
at 1 play waltz on flute;
play waltz - 24 on bass;

loop [G0 B0 D1] on strings; // simple bassline
```

7.5 Alexandre Desplat - Courtyard Apocalypse

court.aux

```
BPM = 240;

instrument taiko : 117;

instrument synth : piano;
instrument synth : C2-G#2 -> taiko;

beat = F2;
```

court.mux

```
//synth parte 1
sequence s1 = [C4C4A#3A#3C4C4F4F4];
sequence s2 = [D#4D#4D4D4D#4D#4G4G4];
sequence s3 = [F4F4D#4D#4F4F4];
sequence s4 = [G#4G#4G4G4G#4G#4D#5D#5];
sequence s5 = [G#4G#4G4G4G#4G#4C5C5];
sequence s6 = [G#4G#4G4G4G#4G#4A#4A#4];
sequence s7 = [G4G4F4F4G4G4C5C5];
sequence s8 = [G4G4F4F4G4G4G#4G#4G#4G#4G4G4G#4G#4F5F5G#4G#4G4G4G#4G
    #4F5F5G#4G#4G#4G#4G#4G#4G#4G#4G#4G#4G#4G#4G#4G#4];

play s1 on synth 4 times;
at 4*|s1| play s2 on synth 2 times;
at 4*|s1| + 2*|s2| play s3 on synth;

number t1 = 4*|s1| + 2*|s2| + |s3|;
at t1 play [A#4A#4] on synth;

performance ps1 = [A#4A#4] on synth;

number t2 = t1 + |ps1|;
at t2 play s3 on synth;

number t3 = t2 + |s3|;
at t3 play [G4G4] on synth;


performance ps2 = [G4G4] on synth;

number t4 = t3 +|ps2|;
at t4 play s5 on synth 2 times;
number t5 = t4 + 2*|s5|;
at t5 play s4 on synth;
number t6 = t5 + |s4|;
at t6 play s6 on synth;
number t7 = t6 + |s6|;
at t7 play s7 on synth;
number t8 = t7 + |s7|;
at t8 play s8 on synth;


```

```
41 //synth parte 2
42
43 number tl1 = 62;
44 sequence pl1 = [G3G3G#3G#3G#3G#3G#3G#3G#3G#3G#3G#3G#3G#3];
45 at tl1 play pl1 on synth 2 times;
46 number tl2 = tl1 + 2*|pl1|;
47 sequence pl2 = [A#3A#3G3G3G3G3G3G3G3G3G3G3G3G3G3G3G3];
48 at tl2 play pl2 on synth;
49 number tl3 = tl2 + |pl2|;
50 sequence pl3 = [D#3D#3F3F3F3F3F3F3F3F3F3F3F3F3F3F3F3];
51 at tl3 play pl3 on synth;
52 number tl4 = tl3 + |pl3|;
53 sequence pl4 = [C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4C4];
54 at tl4 play pl4 on synth;
55
56 //synth de C2-G#2 -> taiko
57
58 performance p1 = [beat RR beat beat RRR beat RR beat beat RRR] on
       synth;
59 performance p2 = p1 + 3;
60 performance p3 = p1 - 5;
61 performance p4 = p1 - 4;
62 number n1 = |p1|;
63 number n2 = n1 + |p2|;
64 number n3 = n2 + |p3|;
65 number n4 = n3 + |p4|;
66 number n5 = n4 + |p1|;
67 number n6 = n5 + |p2|;
68 number n7 = n6 + |p3|;
69
70 play p1;
71 at n1 play p2;
72 at n2 play p3;
73 at n3 play p4;
74 at n4 play p1;
75 at n5 play p2;
76 at n6 play p3;
77 at n7 play p1;
```

7.6 Mais exemplos

No repositório do projeto, podem ser encontrados mais exemplos, na pasta `Visitor/examples/`.

Destaca-se o exemplo da música Despacito (`despacito.mux` e `despacito.aux`), que faz uso de 15 instrumentos, incluindo uma grande amostra de diferentes sons de percussão. Abaixo encontra-se uma imagem de um excerto do ficheiro MIDI gerado:

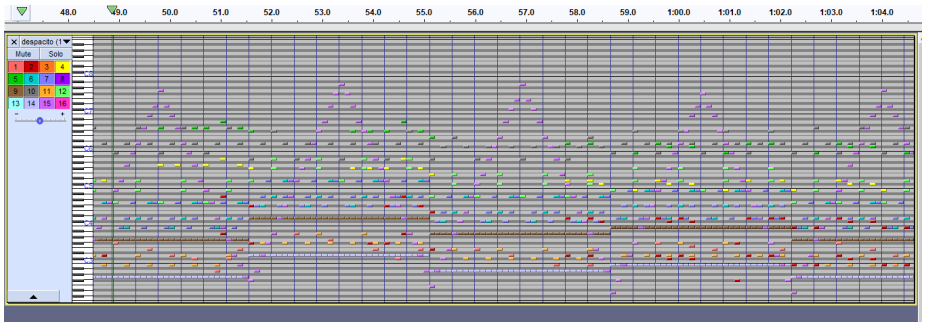

Figura 1: Excerto do ficheiro gerado por `despacito.mux` e `despacito.aux`