Generation of musical patterns through operads

Samuele Giraudo

LIGM, Université Gustave Eiffel, CNRS, ESIEE Paris, F-77454 Marne-la-Vallée, France.

Music box model

Main objectives

- ♯ Define a notion of pattern, **simple abstraction for musical phrases**, containing information about used degrees and rhythm.
- ♯ Endow the set of all patterns with operations to form **algebraic structures**.
- ♯ Use this algebraic framework to perform computations on patterns and to **randomly generate new patterns from smaller ones**.
- ♯ Develop a small **programming language** to implement these ideas.

Patterns

A *pattern* is a word on the alphabet $\{\square\} \cup \mathbb{Z}$.

For any pattern $\mathbf{p}$,

- ♯ the *arity* $|\mathbf{p}|$ of $\mathbf{p}$ is its number of integer letters;
- ♯ the *length* $\ell(\mathbf{p})$ of $\mathbf{p}$ is its length as a word.

For instance,

$$\mathbf{p} := \begin{bmatrix} 0 & \square & 1 & 2 & \bar{1} & \square & 0 & 1 & \bar{2} & \square & \square \end{bmatrix}$$

is a pattern satisfying $|\mathbf{p}| = 7$ and $\ell(\mathbf{p}) = 11$ (any $\bar{a}$ stands for $-a$).

Multi-patterns

For any $m \geqslant 1$, an *m-multi-pattern* is a sequence of m patterns having the same length.

For any m-multi-pattern $\mathbf{m}$,

- ♯ the *multiplicity* of $\mathbf{m}$ is m;
- ♯ the *arity* $|\mathbf{m}|$ of $\mathbf{m}$ is the minimal arity among its patterns;
- ♯ the *length* $\ell(\mathbf{m})$ of $\mathbf{m}$ is the common length of its patterns.

For instance,

$$\mathbf{m} := \begin{bmatrix} \bar{1} & 1 & \square & 0 & \square & 0 & \square & \square \\ 1 & \square & \square & 3 & 4 & 3 & \square & \square \\ \square & \square & 0 & 2 & \square & \square & \bar{1} & \square \end{bmatrix}$$

is an m-multi-pattern of multiplicity 3 satisfying $|\mathbf{m}| = 3$ and $\ell(\mathbf{m}) = 8$.

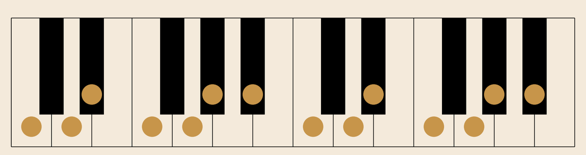
The music box model

The *music box model* is a model to represent musical phrases by m-multi-patterns.

Together with a scale and a root note, an m-multi-pattern denotes a musical phrase:

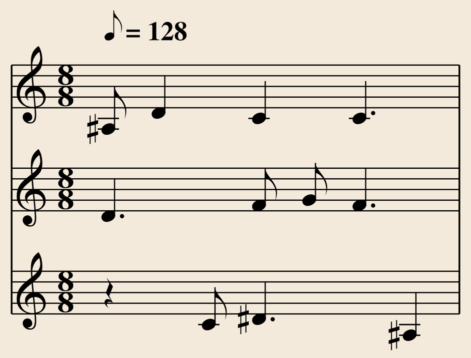
- ♯ each pattern of $\mathbf{m}$ denotes a monophonic phrase;
- ♯ each integer in $\mathbf{m}$ denotes a scale degree lasting one unit of time;
- ♯ each $\square$ in $\mathbf{m}$ extends the duration of a note for one unit of time.

For instance, by considering the natural minor scale and the middle C as root note, one obtains the correspondence

…  …

$\dots\ \bar{7}\ \bar{6}\bar{5}\quad \bar{4}\ \bar{3}\bar{2}\ \bar{1}\quad \mathbf{0}\ 12\quad 3\ 45\ 6\ \dots$

between degrees and notes.

The previous m-multi-pattern $\mathbf{m}$ — seen in the context of the natural minor scale, with C as root note, 128 as tempo, and where each beat lasts one eighth note— denotes the phrase

Multi-patterns as operations

Operads

A *nonsymmetric operad*, or an *operad* for short, is a triple $(\mathcal{O}, \circ_i, \mathbf{1})$ such that

$\mathcal{O}$ is a set decomposing as a disjoint union

$$\mathcal{O} = \bigsqcup_{n \geqslant 0} \mathcal{O}(n),$$

$\circ_i$ is a map

$$\circ_i : \mathcal{O}(n) \times \mathcal{O}(m) \to \mathcal{O}(n + m - 1), \qquad 1 \leqslant i \leqslant n,$$

called *partial composition* map, and $\mathbf{1}$ is an element of $\mathcal{O}(1)$, called *unit*.

This data has to satisfy, for any $x, y, z \in \mathcal{O}$, the three relations

$$(x \circ_i y) \circ_{i+j-1} z = x \circ_i (y \circ_j z), \qquad 1 \leqslant i \leqslant |x|, \quad 1 \leqslant j \leqslant |y|,$$

$$(x \circ_i y) \circ_{j+|y|-1} z = (x \circ_j z) \circ_i y, \qquad 1 \leqslant i < j \leqslant |x|,$$

$$\mathbf{1} \circ_1 x = x = x \circ_i \mathbf{1}, \qquad 1 \leqslant i \leqslant |x|.$$

Operads are algebraic structures wherein elements are n-ary operations which can be composed to form bigger operations.

Operad of multi-patterns

For any $m \geqslant 1$, let

$$\mathbf{MP}_m := \bigsqcup_{n \geqslant 0} \mathbf{MP}_m(n)$$

where $\mathbf{MP}(n)$ is the set of all m-multi-patterns of arity n.

For any m-multi-patterns $\mathbf{m}$ and $\mathbf{m}'$, let $\mathbf{m} \circ_i \mathbf{m}'$ be the m-multi-pattern obtained by replacing each i-th degree in each pattern of $\mathbf{m}$ by the corresponding pattern of $\mathbf{m}'$ obtained by incrementing each of its degrees by the corresponding i-th degree of $\mathbf{m}$.

Set also the unit $\mathbf{1}$ as the m-multi-pattern of arity 1 and length 1 consisting only in degrees 0.

Theorem

For any $m \geqslant 1$, the triple $(\mathbf{MP}_m, \circ_i, \mathbf{1})$ is an operad.

We call $\mathbf{MP}_m$ the *m-music box operad.*

For instance, in $\mathbf{MP}_1$,

$$\begin{bmatrix} 2 & \square & 1 & \square & \mathbf{4} & \square & \square & 0 \end{bmatrix} \circ_3 \begin{bmatrix} 0 & \square & 2 & \square & 4 & \square \end{bmatrix} = \begin{bmatrix} 2 & \square & 1 & \square & 4 & \square & 6 & \square & 8 & \square & \square & \square & 0 \end{bmatrix},$$

and in $\mathbf{MP}_2$,

$$\begin{bmatrix} \square & 1 & \square & \mathbf{4} & 2 & \square \\ 0 & \mathbf{1} & \square & \square & \square & \square \end{bmatrix} \circ_2 \begin{bmatrix} \bar{3} & \square & 0 \\ 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} \square & 1 & \square & 1 & \square & 4 & 2 & \square \\ 0 & 2 & 3 & 4 & \square & \square & \square & \square \end{bmatrix}.$$

Multi-patterns are operations on patterns

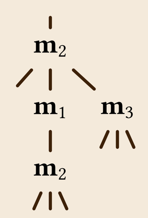
Thanks to the operad structure of $\mathbf{MP}_m$, any m-multi-pattern can be seen as an **operator** acting itself on other m-multi-patterns.

For this reason, we can express a m-multi-pattern by a syntax tree and build new bigger m-multi-patterns for smaller ones.

For instance, by setting

$$\mathbf{m}_1 := \begin{bmatrix} 0 & \square \\ \square & 0 \end{bmatrix}, \quad \mathbf{m}_2 := \begin{bmatrix} 1 & 0 & 1 \\ \bar{7} & 0 & 0 \end{bmatrix}, \quad \mathbf{m}_3 := \begin{bmatrix} 1 & 2 & \square & 3 \\ \bar{1} & 0 & \square & 1 \end{bmatrix},$$

here is a syntax tree involving $\mathbf{m}_1$, $\mathbf{m}_2$, and $\mathbf{m}_3$, a corresponding encoded expression in $\mathbf{MP}_2$, and the new 2-multi-pattern thus obtained:



$$= \quad ((\mathbf{m}_2 \circ_2 \mathbf{m}_1) \circ_2 \mathbf{m}_2) \circ_5 \mathbf{m}_3 \quad = \quad \begin{bmatrix} 1 & 1 & 0 & 1 & \square & 2 & 3 & \square & 3 \\ \bar{7} & \square & \bar{7} & 0 & 0 & \bar{1} & 0 & \square & 1 \end{bmatrix}.$$

Bud generating systems

Colored operads

A $\mathfrak{C}$*-colored operad* is an enriched operad wherein any element x has a color $\mathrm{out}(x)$ and each i-th input of x has a color $\mathrm{in}_i(x)$, all from a set $\mathfrak{C}$.

In a colored operad $\mathcal{C}$, the **partial composition becomes a partial map**: for any $x, y \in \mathcal{C}$, $x \circ_i y$ is defined only if the output color of y is the same as the color of the i-th input of x.

Bud operads and colored multi-patterns

If $\mathcal{O}$ is an operad, we construct a $\mathfrak{C}$-colored operad by setting

$$\mathrm{B}_{\mathfrak{C}}(\mathcal{O}) := \left\{(a, x, u) : x \in \mathcal{O}, a \in \mathfrak{C}, u \in \mathfrak{C}^{|u|}\right\}.$$

In other words, the elements of $\mathrm{B}_{\mathfrak{C}}(\mathcal{O})$ are the ones of $\mathcal{O}$ augmented by an output color and by input colors.

The *pruned* $\mathrm{pr}((a, x, u))$ of any $(a, x, u) \in \mathrm{B}_{\mathfrak{C}}(\mathcal{O})$ is the element x of $\mathcal{O}$.

The colored operad $\mathrm{B}_{\mathfrak{C}}(\mathcal{O})$ is the $\mathfrak{C}$*-bud operad* of $\mathcal{O}$ and it inherits from the partial composition of $\mathcal{O}$.

For instance, in $\mathrm{B}_{\mathfrak{C}}(\mathbf{MP}_2)$ with $\mathfrak{C} := \{\mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3\}$,

$$\left(\mathtt{b}_3, \begin{bmatrix} 0 & \mathbf{1} & \square \\ \bar{1} & \square & \mathbf{0} \end{bmatrix}, \mathtt{b}_2\mathtt{b}_\mathbf{1}\right) \circ_2 \left(\mathtt{b}_1, \begin{bmatrix} 1 & \square \\ 2 & \bar{1} \end{bmatrix}, \mathtt{b}_3\mathtt{b}_2\right) = \left(\mathtt{b}_3, \begin{bmatrix} 0 & 2 & \square & \square \\ \bar{1} & \square & 2 & \bar{1} \end{bmatrix}, \mathtt{b}_2\mathtt{b}_3\mathtt{b}_2\right).$$

Bud generating systems

A bud generating system is a **generalization of context-tree grammars**, intended to generate sets of elements of operads (and not only sets of words or trees).

More precisely, a *bud generating system* is a tuple $(\mathcal{O}, \mathfrak{C}, \mathcal{R}, \mathtt{b})$ where

- ♩ $(\mathcal{O}, \circ_i, \mathbf{1})$ is an operad, called *ground operad*;
- ♪ $\mathfrak{C}$ is a finite set of colors;
- ♪ $\mathcal{R}$ is a finite subset of $\mathrm{B}_{\mathfrak{C}}(\mathcal{O})$, called *set of rules*;
- ♪ $\mathtt{b}$ is a color of $\mathfrak{C}$, called *initial color*.

Colors play the role of nonterminal symbols, and each rule $(\mathtt{b}_j, x, u)$ can be seen as a **production rule** allowing us to replace an input having $\mathtt{b}_j$ as color by x and by its attached input colors u.

For any color $a \in \mathfrak{C}$, we shall denote by $\mathcal{R}_a$ the set of all rules of $\mathcal{R}$ having a as output color.

Random generation

Given a bud generating system $(\mathcal{O}, \mathfrak{C}, \mathcal{R}, \mathtt{b})$, it is possible to generate at random an element of $\mathcal{O}$ by means of the following algorithm.

Algorithm

- ♯ Title: **Partial random generation algorithm**
- ♯ Inputs:
  - ♩ A bud generating system $\mathcal{B} := (\mathcal{O}, \mathfrak{C}, \mathcal{R}, \mathtt{b})$;
  - ♪ An integer $k \geqslant 0$.
- ♯ Output: an element of $\mathcal{O}$.
  - ♩ Set x as the element $(\mathtt{b}, \mathbf{1}, \mathtt{b})$;
  - ♪ Repeat k times:
    - ♩ Pick a position $i \in [|x|]$ at random;
    - ♪ If $\mathcal{R}_{\mathrm{in}_i(x)} \neq \emptyset$:
      - ♩ Pick a rule $r \in \mathcal{R}_{\mathrm{in}_i(x)}$ at random;
      - ♪ Set $x := x \circ_i r$;
  - ♪ Returns $\mathrm{pr}(x)$.

Bud music box language

Bud Music Box tool

Bud Music Box is a new small programming language available at

`https://github.com/SamueleGiraudo/Bud-Music-Box`

It allows us to manipulate multi-patterns, compute various operations on these, generate at random some patterns from other ones, and play and write patterns.

Given a `.bmb` file (see examples below), the compiler translates it into an `ABC` file, a ps file containing its score, and a `MIDI` file.

Creating, naming, and playing patterns

```
{Creates a 1-multi-pattern mpat_1.}
multi_pattern mpat_1 0 * 1 2 * -1 0

{Plays the created 1-multi-pattern.  By default, it is interpreted in the harmonic minor scale with the
 middle A as root note, and with 192 as tempo where each beat lasts one eighth note.}
play mpat_1

{Creates and plays a 2-multi-pattern mpat_2.}
multi_pattern mpat_2 0 * * 1 ; 4 0 -1 *
play mpat_2

{Prints all the defined data and status.}
show
```

Concatenating and composing patterns

```
{Creates three 1-multi-patterns and the 1-multi-pattern res_1 as their concatenation.}
multi_pattern mpat_1 0 1 2 * 3
multi_pattern mpat_2 0 * * 2 4
multi_pattern mpat_3 -1 * -3 -5
concatenate res_1 mpat_1 mpat_2 mpat_3

{Compute the partial composition of the second pattern into the first one at position 2.}
partial_compose res_2 mpat_1 2 mpat_2
```

Changing the ambiant scale, tempo, and instruments

```
{Sets the ambiant scale as the Hirajoshi scale by its sequence of consecutive intervals in semitones.}
set_scale 2 1 4 1 4

{Sets the root to be the note having 60 as MIDI code, the middle C.}
set_root 60

{Sets the tempo to 128.}
set_tempo 128

{Sets the MIDI sound of the first voice to the "Kalimba" of code 108 and of the second voice to the "Koto"
 of code 107.}
set_sounds 108 107

{Defines a 2-multi-pattern and plays it in this context.}
multi_pattern mpat_1 0 * * 1 * 4 * * 2 * ; * * * -3 * * 2 * 0 * 0
play mpat_1
```

Creating colored multi-patterns

```
{Defines a 2-multi-pattern of arity 3.}
multi_pattern mpat_1 0 * * 2 1 * 1 ; -5 * * * 0 * 0

{Creates the colored 2-multi-pattern cpat_1 by augmenting the 2-multi-pattern mpat_1 with c1 as output
 color and the sequence c2 c1 c1 of length 3 of input colors.}
colorize cpat_1 mpat_1 c1 c2 c1 c1
```

Complete example

```
{Sets some context information.}
set_scale 2 1 4 1 4
set_root 60
set_tempo 128
set_sounds 108 107

{Defines three 2-multi-patterns.}
multi_pattern mpat_1 1 1 0 0 2 2 1 1 ; -5 * * * 0 * * *
multi_pattern mpat_2 -1 * 0 * 1 * ; * 0 * * 0 *
multi_pattern mpat_3 0 * ; * 0

{Defines four colored 2-multi-patterns from the previous 2-multi-patterns.}
colorize cpat_1 mpat_1 c1 c2 c1
colorize cpat_2 mpat_1 c1 c1 c2
colorize cpat_3 mpat_2 c1 c1 c2
colorize cpat_4 mpat_3 c1 c1

{Creates a new 2-multi-pattern mpat_4 obtained by using the partial random generation algorithm with
 k := 32, c1 as initial color, and cpat_1, cpat_2, cpat_3, and cpat_4 as rules.}
generate mpat_4 partial 32 c1 cpat_1 cpat_2 cpat_3 cpat_4

play mpat_4
```


Email address: samuele.giraudo@u-pem.fr