the Globbian Language

2022

Tamir Globus

Abstract

It may be easier to read letters of a language with unique symbols than a language such as musical notes which all look more or less the same. Languages such as English, Hebrew and French use different sets of symbols, which we refer to as ‘letters’ to instruct the reader how to pronounce the word or in which group the word exists. In other words, the reader does not need to read the whole word to understand it. Researchers have found that some of us can even complete the word just by seeing the first and last letters of it.

Since they all look similar, musical notes require a great amount of effort to read, and there is a good reason for that. If you’ve ever looked at a piano closely, you saw it has 88 keys. If we had 88 letters in the English alphabet, we would have had much harder time to recognize each letter in our mind, and much more to recognize a whole word - since the number of permutations of a vocabulary with that size is enormous. For the same reason, the language in which music players read notes couldn’t be with 88 unique symbols. As a result, musicians have developed a cycle based technique called ‘octave’, which allows us to refer to each note more easily.

The following paper isn’t a mathematical/statistical optimization paper where we attempt to optimize several parameters, and get close to the best language for the human mind. Rather than that, in this paper, we’ll design an explicit new language that is based on mathematics, and is probably easier to understand than the current one.

“In order to define it, you must understand how it works.”

- **v** = octa**v**e
  The coefficient of v is the number of octaves the note is located relative to the middle-C octave. Relative to the middle-C octave, higher pitches will have positive **v** and negative ones will have negative **v**.

- **t** = **t**ime
  If positive, the coefficient of **t** describes the number of counts one needs to play note. If negative, it tells you that no music should be played.

- **p** = ty**p**e
  C, D, E, F, G, A, B are all of the existing types a note could be. Any sharps and flats should be written as C#, E# or Cb, Ab or Bb. There is no sign for cancellation of the Sharpened or flattened note. Therefore, sharpening one note does not change the pitch of all the notes of the same type for the same musical box, as in regular music theory.

- **Note Variable** - Any note or a group of notes (a note-matrix) can be assigned to a variable.

- **Musical Box Variable** - Any Musical Box is assigned to a number, such that the first is 1, the second is 2 and so on. If “3 - 10” was written, then the musical player should play all the musical boxes from the 3rd note, up to the 10th one, including the 3rd and the 10th.

- **Calculations First** - All musical boxes are played only after all calculations were made. All additions/substitutions to/from the **octave** row will not be different from the elements of the following group: {1, -1, 0.5, -0.5}.

- **Sharps & Flats Notations** - Sharpening/Flattening a note **p1** is equivalent to adding/subtracting to/from its half tone. Expressions

such as **p1 + #**, and **p1 + b** are equivalent to **p1#** and **p1b** correspondingly.

- **Zero Row Deletion** - Since we use **0** as the default value of all of the matrix elements, a row in a matrix that all of its elements are **0**, could be ignored, although the matrix will stay the same. If the **octave** row was removed from the matrix, then the time row must be removed too.

- **Chords Representation** - Any letter from the familiar letters of regular music sheets may be written without the **c** sign and without the matrix explicit form, if no special customizations are required.

- **Operations** - All additions may be substituted. For operations such as sharpening a note, there are counter operations such as flattening the note. For operations such as transforming a note to another note, there are counter operations too.

- **Representation** - A note is usually written as a three dimensional vector that its top component is the **type** of that note, the middle is the **octave** of the note and the lower is the **time** of the note, as follows:

$$\begin{bmatrix} F \\ 2 \\ 1/2 \end{bmatrix}$$

Time Matrix - If several notes share the same value such as type, time or octave, then they could be written as a matrix. The **first** note (first column) is played **first**, the **second** (second column) is played **second** and so on. The following matrix column-items share a common number of time counts, which is **+2**. therefore, it could be written as follows (pay attention that the last column note has +1 more time counts than the others, which means it will be played for +1 count longer):

$$\begin{bmatrix} F & G & G \\ 2 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} F+0 & G+0 & G+0 \\ 2+0 & 1+0 & 1+0 \\ 0+2 & 0+2 & 1+2 \end{bmatrix} = \begin{bmatrix} F & G & G \\ 2 & 1 & 1 \\ 2 & 2 & 3 \end{bmatrix}$$

Chord Matrix - If the **c** sign is on the bottom right of the matrix, then the user will have to play the notes simultaneously. However, each note in the chord can have a different time and octave, which is one of the reasons that this language is so powerful. Pay attention that some notes are played longer, but all are starting at the same time. The right matrix calculation is the explicit form of the left side calculation:

$$\begin{bmatrix} F & G & G \\ 2 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}_c + 2t = \begin{bmatrix} F & G & G \\ 2 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}_c + \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

From the last example we understand that **t**, **v** and **p** are all 3 dimensional vectors that could be multiplied by scalar and could be added/substituted to/from a matrix:

$$v = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, t = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

In order to define the third vector **p**, we must define the group of all the note types that exist, and then define the addition/substitution methods when acting over this group. As you can see in the example below, we first define the known notes from basic music theory, and then define their group in the mathematical form that provides us with the freedom to add/sub from certain elements of the group and get others:

$$\begin{aligned} types &= \{C, D, E, F, G, A, B\} \\ &= \{C,\ C+1,\ C+2, C+3, C+4, C+5, C+6\} \end{aligned}$$

Such that...

$$p = \begin{bmatrix} C \\ 0 \\ 0 \end{bmatrix}$$

Chord Inverse - The first inverse of the following chord is written using the **-1** scalar, at the top right of the matrix, as in mathematics. As you can see, all of its notes are played for the same period of time but possibly located on different octaves. Additionally, the first note of the chord was raised by one octave, as would have happened in regular notes syntax:

$$\begin{bmatrix} C & E & G \\ v_1 & v_2 & v_3 \\ t & t & t \end{bmatrix}_c^{-1} = \begin{bmatrix} C & E & G \\ v_1 + 1 & v_2 & v_3 \\ t & t & t \end{bmatrix}_c$$

The second inverse of the chord is defined as follows:

$$\begin{bmatrix} C & E & G \\ v_1 & v_2 & v_3 \\ t & t & t \end{bmatrix}_c^{-2} = \begin{bmatrix} C & E & G \\ v_1 + 1 & v_2 + 1 & v_3 \\ t & t & t \end{bmatrix}_c$$

The **k**th inverse of **any** chord is defined as follows:

$$\begin{bmatrix} p_1 & p_2 & p_3 \\ v_1 & v_2 & v_3 \\ t_1 & t_2 & t_3 \end{bmatrix}_c^{-k} = \begin{bmatrix} p_1 & p_2 & p_3 \\ v_1 + floor\left(1 + \frac{k}{4}\right) & v_2 + floor\left(\frac{x+1}{3}\right) & v_3 + floor\left(\frac{k}{3}\right) \\ t_1 & t_2 & t_3 \end{bmatrix}_c$$

Sharps & Flats - In the last definition of the group **types**, we didn't include the sharps and flats, which are known as the black keys on the piano. In order to do so, we'll define the **types** group one last time:

$$\begin{aligned} types &= \{C, C\#, D, D\#, E, F, F\#, G, G\#, A, A\#, B\} \\ &= \left\{C, C + 1 \cdot \frac{1}{2}, C + 2 \cdot \frac{1}{2}, C + 3 \cdot \frac{1}{2}, \ldots, C + 11 \cdot \frac{1}{2}\right\} \end{aligned}$$

In the last definition of **types** we defined the sharps as an additional half tone to the last element, except where we add **E** half tone and get a white key. Since, all additions may be substituted, we can define the two vectors for the sharpening and flattening operations:

$$sharp = \begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \# \\ 0 \\ 0 \end{bmatrix} \qquad flat = -\begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.5 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \flat \\ 0 \\ 0 \end{bmatrix}$$

Calculations First - In order to apply a transformation on certain notes or values of the chord specifically, we must add to it a matrix with the same dimensions. Since we don't know what the notes of the chord would be transformed to, me must calculate the expression, and then play the chord:

$$\begin{bmatrix} A & C\# & E \\ 0 & 1 & 1 \\ 2 & 1 & 0 \end{bmatrix}_c + \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} A & C\# & E \\ 2 & 1 & 1 \\ 3 & 2 & 1 \end{bmatrix}_c$$

Since we didn't actually see the Globbian Language in work, I want you to see an example from the beginning of "Turkish March" by Mozart:

$$\overbrace{\begin{bmatrix} B & A & G\# & A \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{1}{16}t}^{1} \rightarrow \overbrace{\begin{bmatrix} C & D & C & B & C \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \frac{1}{16}t}^{2} \rightarrow \overbrace{\begin{bmatrix} E & F & E & D\# & E \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + v + \frac{1}{16}t}^{3}$$

In the example above we started by calculating the first operation and got a matrix that tells us to initially play **B** for 1/16 counts, then **A** for 1/16 and so on. Most of the notes of the second matrix are located one octave above the middle **C** octave, and this is the reason they have **1** in the middle row - which tells us how many octaves above/below middle C octave we have to move in order to play the note.

Since the common coefficient of the times in the first matrix is 1/16, we can remove the zeros since we already take them as default, and do the same for the other matrices:

$$\overbrace{[B \quad A \quad G\# \quad A]}^{1} + \frac{1}{16}t \quad \rightarrow \quad \overbrace{\begin{bmatrix} C & D & C & B & C \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}}^{2} + \frac{1}{16}t \quad \rightarrow \quad \overbrace{[E \quad F \quad E \quad D\# \quad E] + v}^{3} + \frac{1}{16}t$$

Repeat Sign - As you can see, it saved us a lot of data to analyze - but we still need to be carefull. If we remove the second row, which is the **octave** row, we'll have to remove the **time** row too, so that we won't get confused between them and think that the **time** row is the **octave** row. If we want the player to play the three matrices again, as we mentioned in the fourth musical box, then we need to use the curly braces, which imply that the player should play a musical box, and the straight line so the player will understand to play a range:

$$\overbrace{[B \quad A \quad G\# \quad A]}^{1} + \frac{1}{16}t \quad \rightarrow \quad \overbrace{\begin{bmatrix} C & D & C & B & C \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}}^{2} + \frac{1}{16}t \quad \rightarrow \quad \overbrace{[E \quad F \quad E \quad D\# \quad E] + v}^{3} + \frac{1}{16}t \quad \rightarrow \quad \{1\} - \{3\}$$

Directed Matrix - In order to save even more data, we can use the fact that all three first matrices have a common factor of (1/16)t, and add the same value to the three of them over a single operation. Directed Matrix, which is a matrix that contains matrices of notes, or a small piece of music, is the right solution for this challenge. If we look closer, each of the musical boxes we apply the **time** addition on, is an element of the Directed Matrix:

$$\frac{1}{16}t + \left(\overbrace{[B \quad A \quad G\# \quad A]}^{1} \quad \rightarrow \quad \overbrace{\begin{bmatrix} C & D & C & B & C \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}}^{2} \quad \rightarrow \quad \overbrace{[E \quad F \quad E \quad D\# \quad E] + v}^{3} \right) \quad \rightarrow \quad \{1\} - \{3\}$$

In order to show you how powerful this language is, in the following example, the writer used the syntax to tell the player to play the Directed Matrix in higher +1 pitch of the inverse of the matrix. Just for the example let's consider that all of the repeated parts are valid invertible chords:

$$\frac{1}{16}t + \left(\overbrace{[B \quad A \quad G\# \quad A]}^{1} \quad \rightarrow \quad \overbrace{\begin{bmatrix} C & D & C & B & C \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}}^{2} \quad \rightarrow \quad \overbrace{[E \quad F \quad E \quad D\# \quad E] + v}^{3} \right) \quad \rightarrow \quad (\{1\} - \{3\})^{-1} + v$$

Chords Theory - With the Globbian Language, one can know the position and duration a chord should be played, with a very simple syntax. An **E major** chord played for **2 counts** and starts **one octave above** middle C's octave, looks like this and equivalent to the matrix on the right:

$$E + v + 2t = \begin{bmatrix} E & G\# & B \\ 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix}_c$$

There are several benefits to the Globbian Language over the regular music syntax, which haven't been changed for hundreds of years, and some of them are described below:

+ **Mathematical Operations** - Operations such as adding two counts to a group of notes is something that is not possible in regular music theory syntax, and should be written explicitly for each note. Adding one octave, or half tone to a note requires no additional sign with the Globbian Language, rather, a specification of a number.

+ **Coefficients** - If a group of notes share the same amount of time, or share the same octave, then common coefficients could be written by adding a vector of the same common property to the note-matrix.

+ **Explicit Value** - With the Globbian Language, there is no need for effort to understand the time, octave or type of note that should be played such as in regular music sheets, but only to understand the types of notes that exist, what an octave is, and real numbers.