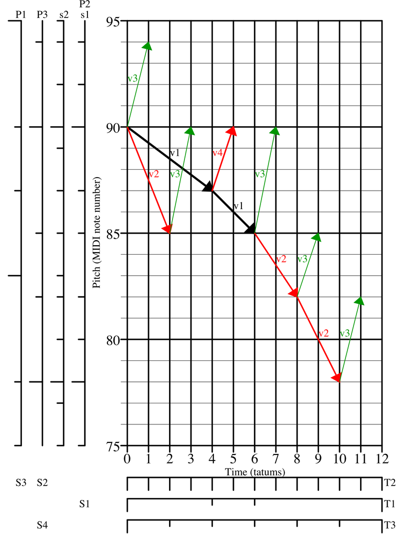


A Geometric Language for Representing Structure in Polyphonic Music

David Meredith
dave@create.aau.dk

Assumptions, Goals and Claims

- A **minimal-length description** of a musical object is a representation of one of the **simplest explanations** for its structure (when considered in isolation).
- The goals of music analysis and music perception are to find minimal-length descriptions of musical objects (particularly musical corpora).
- The goal here is to design an **encoding language** capable of expressing minimal-length descriptions of musical objects.
- This encoding language must be capable of expressing the types of **equivalence relations** that occur in music, since descriptions can be shortened by recognizing equivalences between parts of an object.
- The most important type of equivalence in music is **translational equivalence** within **pitch-time space**.
- Musical translation is different from Euclidean geometric translation because pitch-time space can be transformed by **pitch alphabets** and **rhythms**.
- Pitch alphabets and rhythms can be represented by periodic **masks**, organised into hierarchical **mask sequences**.

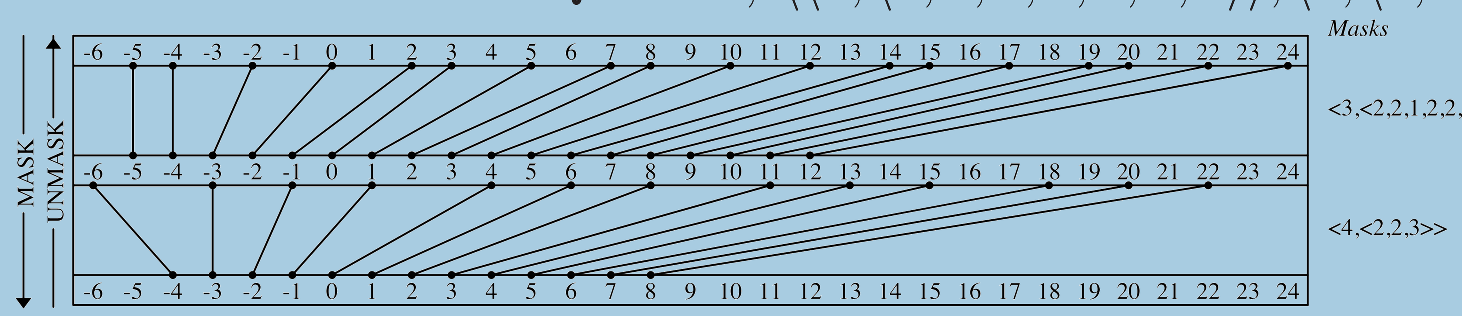
MEL: A Music Encoding Language

`note(t,p)` In MEL, a musical object is represented as a set of **notes**. Each note has an onset time, t, in tatums and a pitch, p, in terms of MIDI note number. A note is a point in **note space**.

`vector(t,p,Mt,Mp)` A **vector** in MEL can be used to translate a note. A vector has a time component, t, a pitch component, p, and two **mask sequences**, Mt and Mp, that define the **space** in which the vector is defined.

`mask(o,s)` A **mask** defines a periodic repeating pattern on the integers. The mask has an **offset**, o, and a **structure**, s, which is a sequence of integers called **intervals**. A mask maps a subset of the integers onto the complete set of integers, as shown below.

Applying the mask sequence, $\langle\langle 3, \langle 2, 2, 1, 2, 2, 2, 1\rangle\rangle, \langle 4, \langle 2, 2, 3\rangle\rangle\rangle$



`maskSequence(m1,m2,...)` A **mask sequence** is a sequence of masks. The output of one mask can be given as the input to another, as shown below left. Mask sequences can be used to define hierarchically-related pitch alphabets or metrical structures or rhythms.

`space(Mt,Mp)` A **space** is defined by two mask sequences, Mt and Mp, which are applied to the time and pitch dimensions, respectively.

`vectorSum(v1,v2,...)` Represents the sum of two or more vectors that may not be in the same space. A vector in a masked space is not in general equal to a unique vector in note space. A sum of two or more vectors is therefore not necessarily equal to a unique vector in any space. It therefore has to be expressed explicitly as a **vector sum**.

`product(X1,X2,...)` Returns the Cartesian product of its arguments. Each argument must be a collection of vectors or vector sums or a sequence of such collections. Corresponds to Deutsch and Feroe's "prime" operator.

`translate(N,V)` Translates the note or note set, N, by the collection of vectors or vector sums, V.

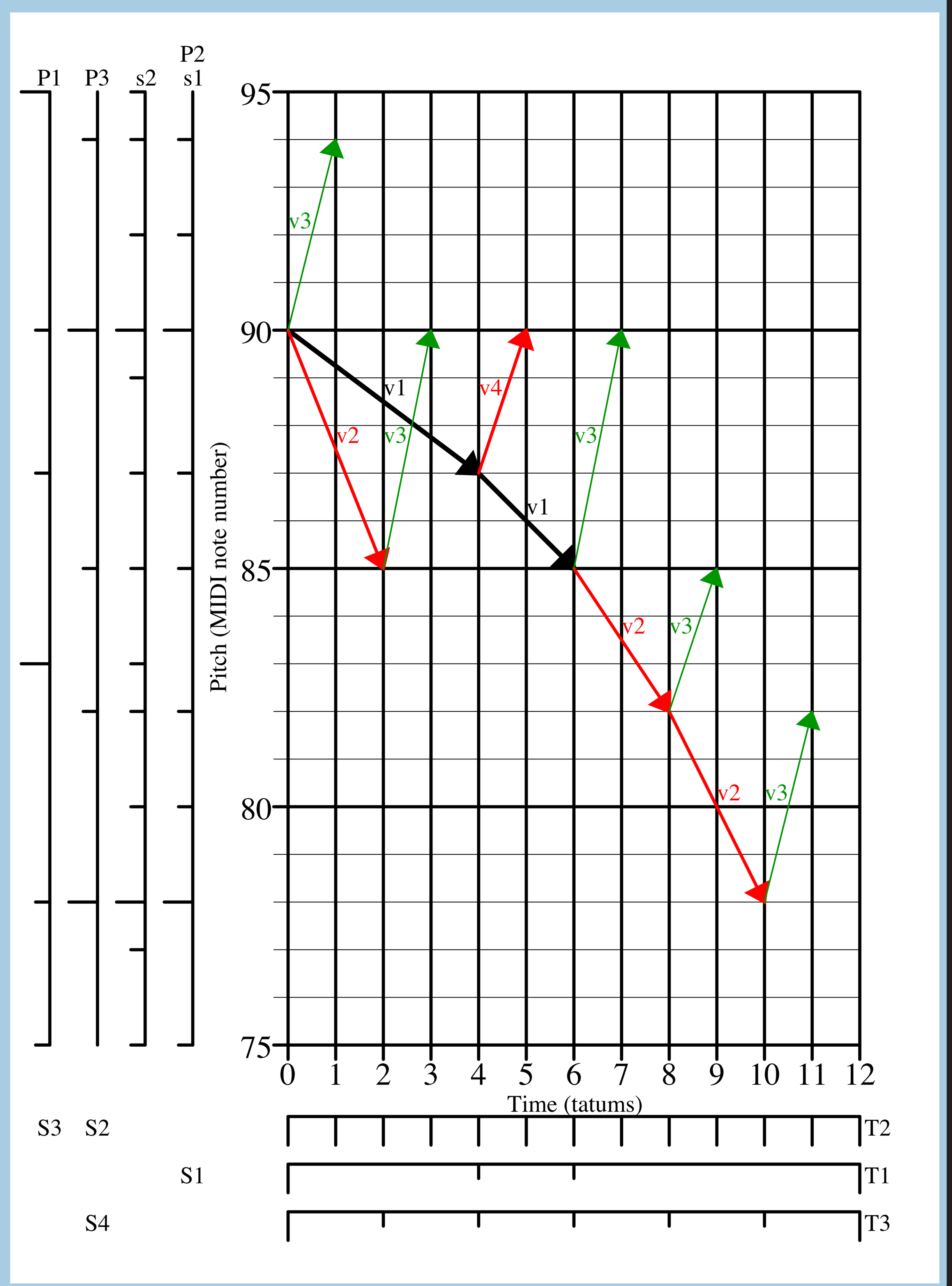
An Example MEL Encoding

Foreground

Middleground

Background

```
MEL25;
n1 = note(0,90);                    //First note
p = coords(1,-1);                   //Corresponds to p ("previous") operator in Deutsch and Feroe
n = coords(1,1);                    //Corresponds to n ("next") operator in Deutsch and Feroe
ms1 = maskStructure(2,2,3);         //Triad mask structure
s1 = mask(6,2,2,3,2,3);             //Background scale: Gb pentatonic
s2 = mask(6,2,2,1,2,2,2,1);         //Gb major scale
T1 = maskSequence(mask(0,4,2,6));   //Background rhythm
T2 = maskSequence(mask(0,1));       //Tatum time mask sequence
T3 = maskSequence(mask(0,2));       //Time mask sequence for alternate semiquavers
P1 = maskSequence(s2,mask(3,ms1));  //Subdominant triad in Gb major
P2 = maskSequence(s1);              //Pitch mask sequence for background (Gb pentatonic)
P3 = maskSequence(s2,mask(0,ms1));  //Tonic triad in Gb major
S1 = space(T1,P2);                  //Background space
S2 = space(T2,P3);                  //Space for first four semiquavers
S3 = space(T2,P1);                  //Space for vector v4
S4 = space(T3,P3);                  //Space for vector v2
v1 = vector(p,S1);                  // \
v2 = vector(p,S4);                  //  | Vectors - see figure ->
v3 = vector(n,S2);                  //  |
v4 = vector(n,S3);                  // /
Q1 = repeat(2,v1);                  //Sequence of 2 v1 vectors in background space
Q2 = repeat(2,v2);                  //Sequence of 2 v2 vectors in middleground space
R1 = product(v2,v3);                //Cartesian product of v2 and v3
R2 = product(Q2,v3);                //Cartesian product of Q2 = <v2,v2> and v3
add(translate(n1,
              product(Q1,                   //<v1,v1>
                      sequence(R1,          //v2 x v3
                               vectorSumSet(v4),  //v4
                               R2))));      //<v2,v2> x v3
print();                            //Prints to the console
draw();                             //Draws a graph in a window
play(100);                          //Plays resulting note set, with tatum = 100ms
```



Reference

Deutsch, D. and Feroe, J. (1981). The internal representation of pitch sequences in tonal music. *Psychological Review*, 88(6):503–522.

Code and further information

MEL Java code at `http://chromamorph.googlecode.com`
Full paper at `http://www.titanmusic.com/papers.php`