Tree symbolic music representation for key finding

David Rizo, José M. Iñesta
Dept. Lenguajes y Sistemas Informáticos
Universidad de Alicante, E-03071 Alicante, Spain
drizo,inesta@dlsi.ua.es

ABSTRACT

This work proposes a polyphonic symbolic music representation that uses trees as coding scheme to solve the problem of key finding

Keywords: tonality, key, symbolic representation, trees

1 Introduction

In music theory, the tonality is defined as the quality by which all tones of a composition are heard in relation to a central tone called the keynote or tonic.

The majority of works that model the tonality of a song use linear sequences of notes (Zhu and Kankanhalli (2004), Temperley (2002)). There are other alternatives such as the *spiral array* presented in Chew (2001), and a different approach, in Rizo et al. (2003) a new tree representation of monophonic music was used to compare the similarity of musical fragments, getting better results with trees than using typical string sequences. In this paper we extend the proposed tree model to represent polyphonic melodies, and use it to find the key of a melodic segment.

The paper is organized as follows: first the monophonic tree representation of music is reviewed, next the extension to polyphonic music is introduced. After that, the preprocess of trees is explained before describing the algorithm to calculate the key of the song. We expose the experiments that have been performed and give the obtained results. We end with some conclusions and planned future works.

2 Tree representation for music sequences

A melody has two main dimensions: time (duration) and pitch. In linear representations, note durations are coded by explicit symbols, but trees are able to implicitly represent this dimension, making use of the fact that note durations are multiples of basic time units, mainly in a binary (sometimes ternary) structure. This way, trees are less sensitive to the codes used to represent melodies, since there are less degrees of freedom for coding.

In this section we review shortly the tree construction method that was introduced in Rizo et al. (2003) for representing a monophonic segment of music, defining the terms needed to build the model.

2.1 Tree construction for each measure

The tree representation approach is based on the fact that the duration of the music notation is designed according to a logarithmic scale: a *whole* note lasts double than a *half* note, that is two times longer than a *quarter* note, etc. (see Fig. 1).

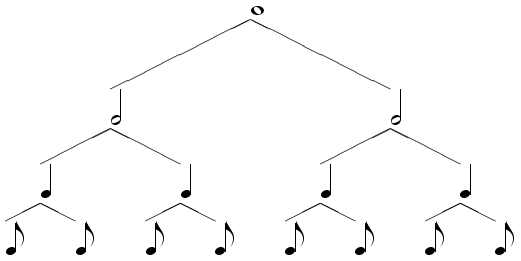

Figure 1: Duration hierarchy for different note figures. From top to bottom: whole (4 beats), half (2 beats), quarter (1 beat), and eighth (1/2 beat) notes.

Each melody measure is represented by a tree, τ. Each note or rest will be a leaf node. The left to right ordering of the leaves keeps the time order of the notes in the melody. The level of each leaf in the tree determines the duration of the note it represents, as displayed in figure 1: the root (level 1) represents the duration of the whole measure (a *whole* note), each of the two nodes at level 2 represents the duration of a *half* note. In general, nodes at level i represent the duration of a $1/2^{i-1}$ of a measure.

During the tree construction, internal nodes are created when needed to reach the appropriate leaf level. Initially, only the leaf nodes will contain a label value, but then, a bottom-up propagation of these labels is performed to fully label the tree nodes. The rules for this propagation will be described later, in section 2.4.

Labels are codes representing any information relating to pitch. In this work labels are the pitch of the note without octave information, also named *folded pitch*, defined by the MIDI note number modulo 12, ranging from 0 to 11. Then, a *C* will be represented as a 0, a *C sharp* as a 1, and *B* as a 11. Rests are coded with a special symbol 's'.

An example of this scheme is presented in Fig. 2. In the tree, the left child of the root has been splitted into two subtrees to reach the level 3, that corresponds to the first note (a quarter note, duration of a $1/2^2$ of the measure,

pitch B (**11**). In order to represent the durations (both are 1/8 of the measure) of the rest and note G (**7**), a new subtree is needed for the right child in level 3, providing two new leaves for representing the rest (**s**) and the note G (**7**). The half note C (**0**) onsets at the third beat of the measure, and it is represented in level 2, according to its duration.

It can be seen in figure 2 how the order in time of the notes in the score is preserved when traversing the tree from left to right. Note how onset times and durations are implicitly represented in the tree, compared to the explicit encoding of time needed by strings. This representation is invariant against changes in tempo, or different meter representations of the same melody (e.g. 2/2, 4/4, or 8/8).

For a deeper explanation of how to deal with dotted notes, ternary subdivisions, grace notes and more elaborated examples see the full method in Rizo and Iñesta (2002).

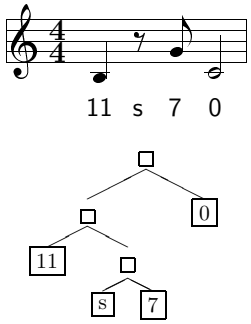

Figure 2: Simple example of tree construction

2.2 Complete melody representation

The method described above is able to represent a single measure as a tree, τ. A measure is the basic unit of rhythm in music, but a melody is composed of a series of M measures. In Rizo et al. (2003) it was proposed to build a tree with a root for the whole melody, being each measure sub-tree a child of that root. This can be considered as a forest of sub-trees, but linked to a common root node that represents the whole. Figure 3 displays an example of a simple melody, composed of three measures and how it is represented by a tree composed of three sub-trees, one per measure, rooted to the same parent node. The level 0 will be assigned to this common root.

2.3 Polyphony

The method to represent polyphonic music is straight forward. All notes are placed in the same tree following the rules of the monophonic music representation, then, two notes with the same onset time will be put in the same node. If a node already existed when a new note is put in the tree, the pitch of this note is added to the current node label. If the label in the node was a rest, it is substituted by the note pitch. The figure 4 contains a melody with a chord as an example. Before propagating (section 2.4), only leaves are labelled.

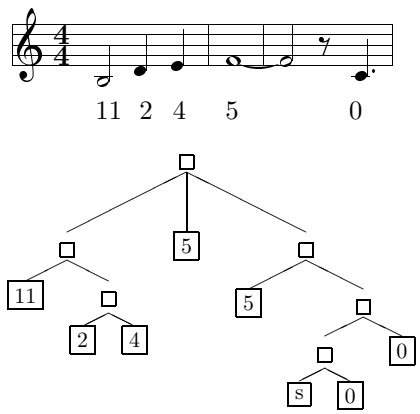

Figure 3: An example of the tree representation of a complete melody. The root of this tree links all the measure sub-trees.

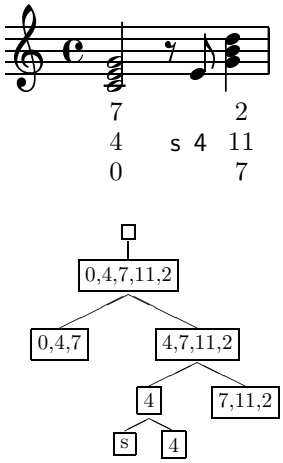

Figure 4: An example of a polyphonic melody, internal nodes are labelled during the propagation

2.4 Bottom-up propagation of labels

Once the tree is constructed, a propagation step is performed. The propagation rules are different from those proposed in Rizo et al. (2003), where the target was the similarity search. Now the presence of all the notes is emphasized since every note and chord are tips to find the key of the song segment.

The propagation process is performed recursively in a post-order traversal of the tree. Labels are propagated using the set algebra. Let $L(\tau)$ be the set of folded pitches label of the root node of the subtree τ. When the label of the node is a rest, the label set is empty: $L(\tau) = \emptyset$. Then, given a subtree τ with children c_i, the upwards propagation of labels is performed as $L(\tau) = \bigcup_i L(c_i)$.

In figure 4 we can see how the *E* quaver note (folded pitch 4) that shared a parent with the rest is promoted ($\emptyset \cup \{4\} = \{4\}$), and merging this 8th note ($\{4\}$) and the chord next ($\{7,11,2\}$) results a parent label: ($\{4\} \cup \{7, 11, 2\} = \{4, 7, 11, 2\}$). Similarly occurs for the root of the measure.

3 Key finding algorithm

Suppose we have a local segment of music with two notes *C* and *G* (first bar in score at figure 5) and want to know which one of the 24 different keys (12 major, 12 minor) describes this segment the best in terms of tonality. The answer is not a unique key: maybe it is in *C major*, or in *B minor*, *A minor*, etc... The second measure is probably in *C major*, *A minor*, but not in *B minor*. The third local segment is the third measure, it is surely in *C major*, and with less probability in *A minor* because the chords that appear are the subdominant, dominant and tonic of *C major*. If we combine the possible keys of the three bars, the most probable answer is *C major*.

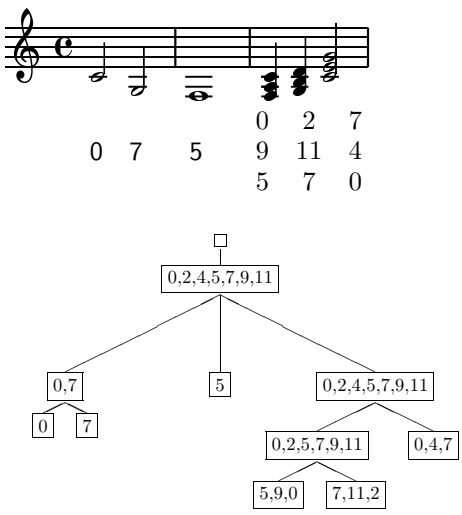

Figure 5: An example of key detection

This means that each local segment can give a clue of the possible keys in which it is written in, but the combination of many local clues can reduce the possible keys leaving at the end the correct tonality.

In our tree scheme, each node is a local segment that contains one or more folded pitches, each node may be in several possible keys. If the possible keys are combined from the leaves to the root following a post-order traversal of the tree, finally the root will give us the correct key. Following the tree of figure 5, the leftmost node with a $\{0\}$ label can be in any key that has not the C with a sharp in its key signature. Its sibling node, the one with a label $\{7\}$ can be in any key that has not the G sharp or flat in its key signature. Their parent node, the one with label $\{0,7\}$ contains a chord (actually it should be interpreted as an arpeggio), this chord gives more probability to some keys (e.g. *C major*) than others that are also possible (e.g. *F major*), furthermore, if the probabilities are combined for these three nodes in a convenient way, the possibilities are reduced.

The computing of the possible keys for each node is performed in two steps. First a *rate* is obtained for each one of the 24 keys following a rules algorithm that will be detailed later. Then, the keys are ordered according to the rate obtained giving as a result a *rank*, which have the more possible keys first. Once each individual node has a rank with the keys ordered, a post-order combination of these ranks yields a final rank at the root node of the tree that will give us the probability of each key to be the central key of the whole song.

The algorithm 1 performs the key finding over the polyphonic tree.

Algorithm 1 Key find on tree τ

```
if arity(τ) = 0 then
    calculate keys ranks for τ root node (see section 3.2)
else
    for all child(τ) ∈ children(τ) do
        Algorithm 1 for tree child(τ)
    end for
    Combine the results for root node of τ and every child into the root node of τ (see section 3.3)
end if
```

3.1 Scales, degrees and chords

3.1.1 Scales

The definition 3.1 specifies the scales we have worked with represented as a vector indexed by the interval from the tonic note of the key (from 0 to 11), being *M* the major scale, and *m* the minor scale. The values $M[i] \neq 0$ are the degrees in the scale represented in roman numbers, being the zero values those notes that do not belong to the scale. In the minor scale the natural, harmonic and melodic minor scales have been represented.

Definition 3.1 *Diatonic scales*

Major scale M = [I, 0, II, 0, III, IV, 0, V, 0, VI, 0, VII]

Minor scale m = [I, 0, II, III, 0, IV, 0, V, VI, VI, VII, VII]

3.1.2 Degrees

Let a key be divided into its key note and its mode, *major* or *minor*, defined by the corresponding scale S. Then, given a folded pitch p and a key k, the $degree$ of the note folded pitch is defined as:

$$degree(p, k, S) = S[(((p + 12) - k)\ mod\ 12)] \quad (1)$$

The +12 is used to avoid a negative modulo.

A given scale, S, can be either S = *M* or S = *m*. Given the set of $|P|$ folded pitches $P = \{p_1, p_2, ..., p_{|P|}\}$ in a node, the number of pitches in P that belong to the scale *S* of key *k* is defined as:

$$scaleNotes(P, k, S) = \sum_{i=1}^{|P|}(degree(P_i, k, S) \neq 0) \quad (2)$$

Given the $degree$ for a note in the key, we consider as *tonal* and *modal* degrees:

Tonal degrees TD = {I, IV, V}

Modal degrees MD = {III}

Given the above definitions, the $tonalDegreesNotes$ and $modalDegreesNotes$ functions are defined as:

$$tonalDegreesNotes(P,k,S) = \sum_{i=1}^{|P|}(degree(P_i,k,S) \in TD) \quad (3)$$

$$modalDegreesNotes(P,k,S) = \sum_{i=1}^{|P|}(degree(P_i,k,S) \in MD) \quad (4)$$

3.1.3 Chords

Only the diatonic scale triad chords have been considered. The set of notes contained in the label of a node may constitute either a full triad or a partial one. Given the set *P*, $chordNotes(k,c,P)$ is defined as the number of elements in P that belong to a chord c of key k.

In figure 4, look at the leftmost node in the tree that represents the first chord in the score. $P = \{0,4,7\}$, for $k = C\ major$ and c =I (the tonic triad of *C Major*), then $chordNotes(k,c,P)$=3 because it contains the three pitches of this chord (*C* represented by a 0, *E* as a 4 and *G* whose folded pitch is 7). Let $k = A\ minor$ and c =I again (*A minor* tonic triad composed by the pitches *A*, *C* and *E*), the result would be 2 because only the pitches *C* and *E* are found in the chord.

3.2 Node keys rating

Given the previous set of definitions, the rules in algorithm 2 compute the rate for each key according to the set of pitches in a node. The return values correspond to the rate for each key in the node. These rates, detailed in the table 1, have been established empirically.

This scheme gives preference to the triads that clearly belong to a key, it rates with less important two notes chords and single notes that belong to the key, and leaves with no rate the chords with many notes that may belong to several keys.

After computing the previous rates, the keys are ranked. The objective of this ranking is to avoid the propagation of a high rate for a key in a node that might condition the final result.

To compute the ranking, the keys are ordered. If two keys have the same rate, they are given the same rank position. The best rank is given a 0, 1 for the second, and so on. The function $rank(\tau,k)$ returns the position of key k in the rank.

Constant	Rate
FULL_TRIADS_I_V	16
FULL_TRIADS	15
2NOTES_TRIADS_I_V	9
2NOTES_TRIADS	8
NOTES_CHORDS_I_V	10
2NOTES_CHORDS	9
TONAL_DEGREES	4
MODAL_DEGREES	3
SCALE_NOTES	2

Table 1: Rates

Algorithm 2 Key k rating, for the node pitches P

if ($|P| = 3$) **then**
  if ($chordNotes(k,\mathrm{I},P)$=3 **or** $chordNotes(k,\mathrm{V},P)$=**3) then**
    return (*FULL_TRIADS_I_V*)
  else if ($chordNotes(k,\mathrm{II},P)$=**3** **or** $chordNotes(k,\mathrm{III},P)$=**3** **or** $chordNotes(k,\mathrm{IV},P)$=**3** **or** $chordNotes(k,\mathrm{VI},P)$=**3** **or** $chordNotes(k,\mathrm{VII},P)$=**3) then**
    return (*FULL_TRIADS*)
  else if ($chordNotes(k,\mathrm{I},P)$=**2 or** $chordNotes(k,\mathrm{V},P)$=**2) then**
    return (*2NOTES_TRIADS_I_V*)
  else if ($chordNotes(k,\mathrm{II},P)$=**2** **or** $chordNotes(k,\mathrm{III},P)$=**2** **or** $chordNotes(k,\mathrm{IV},P)$=**2** **or** $chordNotes(k,\mathrm{VI},P)$=**2** **or** $chordNotes(k,\mathrm{VII},P)$=**2) then**
    return (*2NOTES_TRIADS*)
  end if
else if ($|P| = 2$) **then**
  if ($chordNotes(k,\mathrm{I},P)$=2 **or** $chordNotes(k,\mathrm{V},P)$=**2 then**
    return (*2NOTES_CHORDS_I_V*)
  else if $chordNotes(k,\mathrm{II},P)$=**2** **or** $chordNotes(k,\mathrm{III},P)$=**2** **or** $chordNotes(k,\mathrm{IV},P)$=**2** **or** $chordNotes(k,\mathrm{VI},P)$=**2** **or** $chordNotes(k,\mathrm{VII},P)$=**2 then**
    return (*2NOTES_CHORDS*)
  end if
else if ($|P| - scaleNotes(P,k,S) > 2$) **then**
  if ($tonalDegreesNotes(P,k,S) > 0$) **then**
    return (*TONAL_DEGREES*)
  else if ($modalDegreesNotes(P,k,S) > 0$) **then**
    return (*MODAL_DEGREES*)
  else if ($scaleNotes(P,k,S) > 0$) **then**
    return (*SCALE_NOTES*)
  end if
end if

3.3 Subtree key combination and propagation

Once the ranks for the children nodes and the parent node have been calculated, they must be combined to replace all the key ranks in the parent node. This operation is performed in two steps, first the calculation of key rates and then a new ranking is performed.

Given a parent tree node τ, with children $c_1, c_2, ..., c_{arity(\tau)}$, the new rate for each key k is calculated as:

$$rate(\tau,k) = rank(\tau,k) + \sum_{i=1}^{arity(\tau)} rank(c_i,k) \quad (5)$$

With these new rates, a new ranking is required, that

Relation to correct key	Points
Same	1
Perfect fifth	0.5
Relative major/minor	0.3
Parallel major/minor	0.2

Table 2: MIREX 2005 Key finding scorings

is calculated as already described. This scheme captures both the actual chords and the chords that are split in arpeggios.

4 Experiments and results

Our algorithm is evaluated during the *Audio and Symbolic Key Finding* topic of the 2nd Annual Music Information Retrieval Evaluation eXchange (MIREX 2005) contest [1]. A set of 1,252 MIDI files has been used. The evaluation process is the one proposed for the contest using for each file the scoring detailed in table 2.

The success rate of an algorithm is obtained as the achieved points averaged for all the songs in the corpus.

Total Score	982.4
Percentage Score	78.5
Correct Keys	913
Perfect 5th Errors	81
Relative Major/Minor Errors	87
Parallel Major/Minor Errors	14
Other Errors	157
Runtime (s)	631

Table 3: 2005 MIREX Contest Results

The experiment has been performed on a machine running a CentOS operating system, and processor Dual AMD Opteron 64 1.6GHz. Our system uses the Java 1.4.2-38 virtual machine.

The results of the experiment with our algorithm are detailed in table 4. The complete results can be found in the web *http://www.music-ir.org/evaluation/mirex-results/sym-key/index.html*.

5 Conclusions and future work

In this work we have shown a polyphonic music tree representation that has performed as a simple and adequate representation for finding the key of a song. We have seen that with very little harmonic information a good key finding can be achieved. This early system can be improved by the use of a more powerful harmonic model that presumably will report better results. We are working also in the key change finding inside the same song getting some promising results.

[1] http://www.music-ir.org/mirexwiki/index.php/MIREX_2005

6 Acknowledgments

This work was supported by the projects Spanish CICYT TIC2003–08496–C04, partially supported by EU ERDF, and Generalitat Valenciana GV043-541.

References

Elaine Chew. Modeling Tonality: Applications to Music Cognition. In Johanna D. Moore and Keith Stenning, editors, *Proceedings of the 23rd Annual Meeting of the Cognitive Science Society*, pages 206–211, Edinburgh, Scotland, UK, August 1-4 2001. Lawrence Erlbaum Assoc. Pub, Mahwah, NJ/London. URL `http://www.hcrc.ed.ac.uk/cogsci2001`.

D. Rizo, F. Moreno-Seco, and J.M. Iñesta. Tree-structured representation of musical information. *Lecture Notes in Computer Science - Lecture Notes in Artificial Intelligence*, 2652:838–846, 2003.

David Rizo and José M. Iñesta. Tree-structured representation of melodies for comparison and retrieval. In *Proc. of the 2^{nd} Int. Conf. on Pattern Recognition in Information Systems, PRIS 2002*, pages 140–155, Alicante, Spain, 2002.

D. Temperley. A bayesian approach to key-finding. *Lecture Notes in Computer Science*, 2445:195–206, 2002.

Y. Zhu and M. Kankanhalli. Key-based melody segmentation for popular song. *17th International Conference on Pattern Recognition (ICPR'04)*, 3:862–865, 2004.