International Journal of Foundations of Computer Science
Vol. 19, No. 1 (2008) 103–124


World Scientific
www.worldscientific.com

# ON SOME COMBINATORIAL PROBLEMS CONCERNING THE HARMONIC STRUCTURE OF MUSICAL CHORD SEQUENCES

DOMENICO CANTONE*
SALVATORE CRISTOFARO†
and
SIMONE FARO‡
*Dipartimento di Matematica e Informatica, Università di Catania*
*Viale Andrea Doria 6, I-95125 Catania, Italy*
** cantone@dmi.unict.it*
*† cristofaro@dmi.unict.it*
*‡ faro@dmi.unict.it*

Received 5 November 2007
Accepted 14 February 2007
Communicated by J. Holub

We present some combinatorial problems which arise in the fields of music representation and music processing, in particular in the context of analysis of the harmonic structure of chord sequences. We are mainly concerned with problems related to chord sequences which exhibit a certain kind of *regular harmonic structure*, and provide also algorithms to solve some of them. In particular, we present an $\mathcal{O}(n+m)$-time algorithm, based on bit-parallelism, to check whether a given chord progression of length $n$ is regular, where $m$ is the size of the chords in the progression.

*Keywords:* Music processing; harmonic structure analysis; chord sequences.

## 1. Introduction

Musical chord sequences, or chord progressions, possess a very rich and complex combinatorial structure, which requires efficient computational methods to be fully understood and analyzed.

By using a convenient symbolic representation of musical notes and chords, it is possible to apply suitable mathematical methods to discover that kind of regularity in the harmonic structure which many chord sequences seem to exhibit [9, 10].

Musical notes can be coded in various ways. A typical example is provided by the standard MIDI representation, where notes are coded by integers [11]. Once a particular coding of the notes is fixed, a chord can be conveniently represented by the collection of the symbols corresponding to the notes in it. Notice that in codings like the standard MIDI representation, notes that differ by one or more

octaves are represented by distinct symbols. Such kind of codings are particularly appropriate in the context of Music Information Retrieval, where the representation of (monophonic) musical sequences by strings of integers gives the possibility of applying powerful string matching techniques to discover musical pattern repetitions and melodic similarities [3, 6, 7, 5].

However, in many cases, especially in those in which one is interested in the interval content of chords, it is more convenient to assume octave equivalence of notes, i.e., to regard as equal any two notes which are one or more octaves apart [8]. In such a case, only 12 symbols are needed to represent the notes, at least in the equal temperament system of western music. For example, let us consider the two simple chord sequences $S_1$ and $S_2$ represented in Fig. 1.

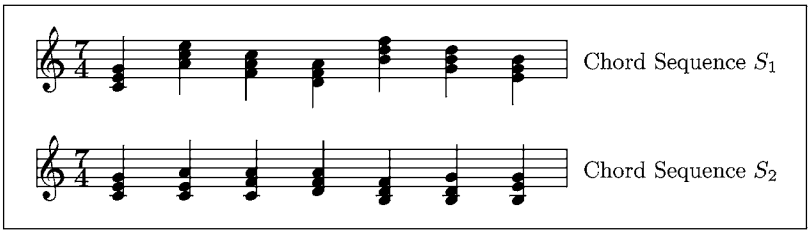


Fig. 1. Two chord sequences related by octave equivalence. Chord sequence $S_1$ exhibits a regular harmonic structure.

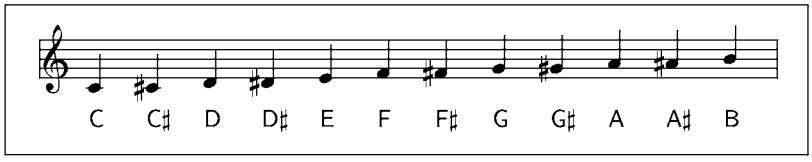


Fig. 2. The traditional naming of the musical notes with the symbols $\mathsf{C}, \mathsf{C\sharp}, \mathsf{D}, \mathsf{D\sharp}, \mathsf{E}, \mathsf{F}, \mathsf{F\sharp}, \mathsf{G}, \mathsf{G\sharp}, \mathsf{A}, \mathsf{A\sharp}, \mathsf{B}$.

If we assume octave equivalence, and use the traditional naming of notes with the symbols $\mathsf{C}, \mathsf{C\sharp}, \mathsf{D}, \mathsf{D\sharp}, \mathsf{E}, \mathsf{F}, \mathsf{F\sharp}, \mathsf{G}, \mathsf{G\sharp}, \mathsf{A}, \mathsf{A\sharp}, \mathsf{B}$ as shown in Figure 2, then we may represent both sequences as the following list of sets

$$\{\mathsf{C},\mathsf{E},\mathsf{G}\},\{\mathsf{C},\mathsf{E},\mathsf{A}\},\{\mathsf{C},\mathsf{F},\mathsf{A}\},\{\mathsf{D},\mathsf{F},\mathsf{A}\},\{\mathsf{D},\mathsf{F},\mathsf{B}\},\{\mathsf{D},\mathsf{G},\mathsf{B}\},\{\mathsf{E},\mathsf{G},\mathsf{B}\}.$$

In fact, the corresponding chords of the sequences are made up of the same notes but in different octaves, i.e., they differ only in the *voicings*. Thus the two chord sequences can be really considered as two distinct variants, or *voice leadings*,[1] of a same chord sequence (assuming octave equivalence). However, if we regard the

[1]Notice that in music, the usual meaning of voice leading concerns the horizontal motion of the notes, or voices, of the chords inside a chord sequence, where a chord sequence is regarded as the superimposition of two or more melodies played simultaneously. For us, a voice leading is simply a sequence of chord-voicings. But from a formal point of view, the two notions are equivalent, up to minor details (see Section 2 and [1]).

chords as ordered sets of notes, we may discover some regularities in the structure of the sequence $S_1$. Indeed, if we look at the voicings of the chords, namely the strings obtained by ordering the notes in each chord from lowest to highest, the chord sequence $S_1$ can be conveniently represented by a matrix $\mathcal{M}_1$ whose columns correspond to such strings:

$$\mathcal{M}_1 = \begin{bmatrix} \mathsf{G} & \mathsf{E} & \mathsf{C} & \mathsf{A} & \mathsf{F} & \mathsf{D} & \mathsf{B} \\ \mathsf{E} & \mathsf{C} & \mathsf{A} & \mathsf{F} & \mathsf{D} & \mathsf{B} & \mathsf{G} \\ \mathsf{C} & \mathsf{A} & \mathsf{F} & \mathsf{D} & \mathsf{B} & \mathsf{G} & \mathsf{E} \end{bmatrix}.$$

A simple inspection of the matrix $\mathcal{M}_1$ reveals the regular structure of the chord sequence. Simply look at the secondary diagonals of each square submatrix of $\mathcal{M}_1$. Also, observe that any two consecutive chords of the sequence share two notes, and any three consecutive chords share one note, so that the chords are connected in such a way that the "transition" from a chord to the one which follows it is gradually achieved by a series of smooth chord-passages: from a perceptual point of view, this translates into a pleasant sensation when the chord sequence is heard.

Notice also that if we "glue" at the left (or right) end of the matrix $\mathcal{M}_1$ a copy of itself, we get a matrix with the very same structure of $\mathcal{M}_1$. Musically speaking, this property can be interpreted by saying that the chord sequence has a kind of "circular" harmonic structure which, when heard, tends to resolve on itself, i.e., after listening to the chord sequence for the first time, one expects that it will be played again. This is strictly related to the phenomenon of musical expectation, which plays a fundamental role in music composition. Thus, the ability of creating and discovering harmonic structures similar to that of the chord sequence $S_1$ in Figure 1 may have various applications in the fields of automatic music composition and automatic music analysis.

Chord sequences having a harmonic structure of the kind described above will be referred to as *regular chord progressions.*

In this paper we report some results concerning various combinatorial questions about regular chord progressions and voicings and raise some open problems.

### 1.1. *Paper's organization*

The paper is organized as follows. In Section 2 we introduce some basic notions and give a formal definition of regular chord progression. Subsequently, in Section 3 we present algorithms, based on the bit-parallelism technique, for some problems concerning combinatorial aspects of regular chord progressions, and also we discuss other related questions. Then, in Section 4 we draw our conclusions. Finally, an appendix containing some technical results concludes the paper.

## 2. Basic Definitions and Properties

Before entering into details, we need a bit of notations and terminology. A string $X$ of length $m \geq 0$ is represented as a finite array $X[0\,..\,m-1]$. For $m = 0$ we obtain the empty string $\varepsilon$. The length of $X$ is denoted by $|X|$. By $X[i]$ we denote the $(i+1)$-st symbol of $X$, for $0 \leq i < |X|$. Likewise, by $X[i\,..\,j]$ we denote the

substring of $X$ contained between the $(i+1)$-st symbol and $(j+1)$-st symbol of $X$, for $0 \leq i \leq j < |X|$. Moreover, for any $i, j \in \mathbb{Z}$, we put

$$X[i\,..\,j] = \begin{cases} X[\max(i,\,0)\,..\,\min(j,\,|X|-1)] & \text{if } \max(i,0) \leq \min(j,\,|X|-1) \\ \varepsilon & \text{otherwise.} \end{cases}$$

For a string $X$, we denote by $Set(X)$ the collection of the symbols occurring in $X$. If $A$ is any finite set of distinct symbols (e.g., an alphabet), we write $|A|$ for the cardinality of $A$. Thus, for any string $X$, we plainly have $|Set(X)| \leq |X|$.[2]

For any two strings $X$ and $Y$, we denote by $X \centerdot Y$ the concatenation or juxtaposition of $X$ with $Y$.

For convenience, we do not distinguish between a symbol $s$ and the one-character string "$s$" which consists only in that symbol. Thus, for any symbol $s$ and any string $X$, we write $s \centerdot X$ for the string $Z$ of length $|X|+1$ such that $Z[0] = s$ and $Z[1\,..\,|X|] = X$. Similarly for the string $X \centerdot s$.

A CHORD is a finite set $C$ of two or more distinct symbols. A chord $C$ is said to be a CHORD OVER AN ALPHABET $\Sigma$, if $C \subseteq \Sigma$.

A CHORD PROGRESSION is a sequence $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ of chords, with $n > 0$, such that $|C_i| = |C_{i+1}|$, for $i = 0, 1, \ldots, n-1$. The length of a chord progression $\mathcal{C}$, denoted length($\mathcal{C}$), is the number of chords in $\mathcal{C}$. A chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ is said to be a CHORD PROGRESSION OVER AN ALPHABET $\Sigma$, if $C_i$ is a chord over $\Sigma$, for $i = 0, 1, \ldots, n$.

A VOICING is a string $V$ such that $|V| \geq 2$ and $V[i] \neq V[j]$, for all distinct $i, j \in \{0, 1, \ldots, |V|-1\}$. The BASE CHORD of a voicing $V$ is the collection $Set(V)$ of the symbols occurring in $V$. A voicing $V$ is a VOICING OVER AN ALPHABET $\Sigma$, if $Set(V) \subseteq \Sigma$.

A VOICE LEADING is a sequence $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ of voicings, with $n > 0$, such that $|V_i| = |V_{i+1}|$, for $i = 0, 1, \ldots, n-1$. The length of a voice leading is the number of voicings it contains. A voice leading $\langle V_0, V_1, \ldots, V_n \rangle$ is a VOICE LEADING OVER AN ALPHABET $\Sigma$, if $V_i$ is a voicing over $\Sigma$, for $i = 0, 1, \ldots, n$.

A voicing $V$ is said to be a VOICING OF A CHORD $C$ if $Set(V) = C$.[3] A voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ is a VOICE LEADING OF A CHORD PROGRESSION $\mathcal{C} = \langle C_0, C_1, \ldots, C_m \rangle$, if $n = m$ and $V_i$ is a voicing of the chord $C_i$, for $i = 0, 1, \ldots, n$.

Let $V$ and $W$ be voicings. We say that $V$ is (IMMEDIATELY) CONNECTED to $W$, and write $V \Longrightarrow W$, if $V[0\,..\,|V|-2] = W[1\,..\,|W|-1]$. Plainly, when $V \Longrightarrow W$, the voicings $V$ and $W$ must have the same length. Moreover, if $\Sigma$ is an alphabet, we write $V \underset{\Sigma}{\Longrightarrow} W$ to mean that $V \Longrightarrow W$ and $V$ and $W$ are voicings over $\Sigma$.

A voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ is CONNECTED if $V_i \Longrightarrow V_{i+1}$, for $i = 0, 1, \ldots, n-1$; $\mathcal{V}$ is CIRCULARLY CONNECTED if it is connected and in addition $V_n \Longrightarrow V_0$.[4]

[2] The meaning of the notation $|\cdot|$ as cardinality of a set or length of a string will always be clear from the context.

[3] Thus, there are $m!$ distinct voicings of any chord $C$ of size $m$.

[4] Observe that if a voice leading $\langle V_0, V_1, \ldots, V_n \rangle$ is circularly connected, then $n \geq |V_0| - 1$.

A voicing $V$ is CONNECTABLE TO A VOICING $W$ WITH RESPECT TO AN ALPHABET $\Sigma$, in symbols $V \underset{\Sigma}{\Longrightarrow}^{+} W$, if there is a connected voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ over $\Sigma$ such that $V_0 = V$ and $V_n = W$. In such a case, we say that the voice leading $\mathcal{V}$ CONNECTS $V$ to $W$ with respect to $\Sigma$, or also that $\mathcal{V}$ is a CONNECTED VOICE LEADING over $\Sigma$ from $V$ to $W$. For any integer $n > 0$ and any voicings $V$ and $W$, we write $V \underset{\Sigma}{\Longrightarrow}^{n} W$ to mean that there is a connected voice leading over $\Sigma$ from $V$ to $W$ of length $n + 1$. Moreover, for any voicing $V$ over $\Sigma$, we put, by definition, $V \underset{\Sigma}{\Longrightarrow}^{0} V$. Notice that if $V \underset{\Sigma}{\Longrightarrow}^{h} W$ and $W \underset{\Sigma}{\Longrightarrow}^{k} Z$, then $V \underset{\Sigma}{\Longrightarrow}^{h+k} Z$, for any voicings $V$, $W$, and $Z$, and any nonnegative integers $h$ and $k$.[5]

A voicing $V$ is CONNECTABLE TO A VOICING $W$ if $V \underset{\Sigma}{\Longrightarrow}^{+} W$, for some alphabet $\Sigma$.

A chord $C$ is (IMMEDIATELY) CONNECTED TO A CHORD $D$, in symbols $C \longrightarrow D$, if $V \Longrightarrow W$, for some voicings $V$ of $C$ and $W$ of $D$. A chord progression $\mathcal{C}$ is CONNECTED (resp., CIRCULARLY CONNECTED) if it has a connected (resp., circularly connected) voice leading. A chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ is REGULAR if it is circularly connected and, in addition, $C_i \neq C_{i+1}$ for $i = 0, 1, \ldots, n-1$, and $C_0 \neq C_n$. It can easily be verified that the chord progression $S_1$ in Figure 1 is regular.

We conclude the section with some examples.

**Example 1** *Given the alphabet* $\Sigma = \{a, b, c, d, e\}$, *the following strings*

$$V_1 = abcd\,, \quad V_2 = dabc\,, \quad V_3 = edab\,, \quad V_4 = edabc$$

*are voicings over* $\Sigma$. *The voicings* $V_1$, $V_2$, *and* $V_3$ *have length* 4, *whereas* $V_4$ *has length* 5. *Moreover, the voice leading* $\mathcal{V} = \langle V_1, V_2, V_3 \rangle$ *is connected, so that the voicing* $V_1$ *is connectable to voicing* $V_3$ *with respect to* $\Sigma$.
*The string*

$$Z = abcdf$$

*is a voicing, but not a voicing over* $\Sigma$, *since* $Z[4] = f$ *is not a symbol of* $\Sigma$ *(however,* $Z$ *is a voicing over the extended alphabet* $\Sigma \cup \{f\}$*).*
*Finally, the strings*

$$X = abca\,, \quad Y = deece$$

*are not voicings (over any alphabet).*

**Example 2** *The following chord progression*

$$\mathcal{C} = \langle \{f, a, c\}, \{a, b, c\}, \{c, e, b\}, \{c, e, b\}, \{f, c, e\}, \{f, a, c\} \rangle$$

*has the circularly connected voice leading*

$$\mathcal{V} = \langle caf, bca, ebc, ceb, fce, afc \rangle$$

*and therefore it is circularly connected. However,* $\mathcal{C}$ *is not regular since its first and last chords are equal.*

[5] In the appendix, we will prove some interesting properties of the connectivity relations $\underset{\Sigma}{\Longrightarrow}^{+}$ and $\underset{\Sigma}{\Longrightarrow}^{n}$.

*Notice that the above voice leading $\mathcal{V}$ can also be represented by the matrix*

$$\mathcal{M} = \begin{bmatrix} f & a & c & b & e & c \\ a & c & b & e & c & f \\ c & b & e & c & f & a \end{bmatrix},$$

*whose columns correspond, from left to right, to the voicings of $\mathcal{V}$, oriented from bottom to top (as are the notes in the staff). Observe that the secondary diagonal elements in each square submatrix of $\mathcal{M}$ are equal.*

**Example 3** *The chord progression*

$$\langle \{a, b, c\}, \{a, b, f\}, \{a, d, f\}, \{b, d, f\} \rangle$$

*is connected but not circularly connected, as can be easily verified by trying out all of its possible connected voice leadings.*

## 3. Discovering Regular Structures: Some Algorithms

In this section we discuss some problems concerning combinatorial properties of regular chord progressions, and provide also algorithms to solve some of them. In particular, we will address in details the problem of determining whether a chord progression is regular (see Problem 2 below), which, as already remarked in the introduction (cf. Section 1), has practical applications in music composition and music analysis.

We begin by considering first a problem on chord voicings, Problem 1 below, whose solution leads to an efficient solution to the problem of checking whether a chord progression is regular. In addition, Problem 1 may have applications in musical situations like those in which a composer is faced with the problem of finding an "efficient" way to voice the chords of a given sequence (of chords), so as to obtain a smooth chord progression which leads from a given starting chord-voicing to a given target chord-voicing.[6]

**Problem 1** *Given a chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$, a voicing $V$ of $C_0$, and a voicing $W$ of $C_n$, construct, if it exists, a voice leading of $\mathcal{C}$ connecting $V$ to $W$, i.e., a connected voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ such that $V_0 = V$, $V_n = W$, and $Set(V_i) = C_i$, for $i = 0, 1, \ldots, n$.*

We can solve Problem 1 as follows. We show how to construct the desired connected voice leading $\mathcal{V}$ of $\mathcal{C}$, or determine that such a voice leading does not exist, by a sequence of $n + 1$ stages. Let $m$ be the length of $V$. We start by setting $V_0 = V$ (this is the initial stage 0). Next, let us suppose that at the end of stage $i$ we have constructed a connected voice leading $\mathcal{V}_i = \langle V_0, V_1, \ldots, V_i \rangle$ of $\mathcal{C}_i = \langle C_0, C_1, \ldots, C_i \rangle$, where $0 \leq i < n$. Then, we form the set $S_i = Set(V_i[0\,..\,m-2])$ and check whether $S_i \subseteq C_{i+1}$. If this is the case, we prolongate the voice leading $\mathcal{V}_i$ with the new voicing $V_{i+1}$ defined by

$$V_{i+1} = c \bullet V_i[0\,..\,m-2]\,,$$

[6] By *voicing a chord $C$* we simply mean to pick any of the voicings of $C$.

**ALGO1($\mathcal{C}$, $\boldsymbol{V}$, $\boldsymbol{W}$)**
*– It is assumed that $\mathcal{C}$ is a chord progression, $\boldsymbol{V}$ is a voicing of*
*– the first chord of $\mathcal{C}$, and $\boldsymbol{W}$ is a voicing of the last chord of $\mathcal{C}$.*
*– Moreover, $\mathcal{C}[i]$ denotes the $(i+1)$-st chord of $\mathcal{C}$.*

```
1.  m := |V|
2.  n := length(C) − 1
3.  S := {V[0], ..., V[m − 2]}
4.  X := V
5.  for i := 1 to n do
6.      if S ⊆ C[i] then
7.          - let z be such that C[i] = S ∪ {z}
8.          S := (S \ {X[m − 2]}) ∪ {z}
9.          X := z . X[0 .. m − 2]
10.     else
11.         return false
12. if X ≠ W then
13.     return false
14. return true
```

Fig. 3. Pseudo-code of the algorithm ALGO1 for determining whether a chord progression $\mathcal{C}$ has a voice leading which connects a voicing $V$ to a voicing $W$.

where $c \in C_{i+1} \setminus S_i$, and proceed to the next stage. Otherwise, we stop the process and announce that there is no connected voice leading of $\mathcal{C}$ from $V$ to $W$. If all stages are completed successfully and, in addition, if the last voicing of $\mathcal{V}_n$ equals $W$, then it is immediate to check that $\mathcal{V}_n$ is a voice leading of $\mathcal{C}$ which connects $V$ to $W$. The correctness of the above procedure follows immediately from the observation that if a voice leading of $\mathcal{C}$ connecting $V$ to $W$ does exist, then it is unique.

In Figure 3 we show the pseudo-code of an algorithm, named ALGO1, which implements the above construction process.

**Remark 1** *Notice that during the execution of the algorithm ALGO1, the string-variable $X$ contains the voicings $V_i$, which form a connected voice leading of $\mathcal{C}$ from $V$ to $W$, provided that it exist. Therefore, if immediately after line 9 of ALGO1 we add an instruction **OUTPUT**$(X)$, we get as by-product the sequence $V_1, V_2, \ldots, V_k$ of voicings, where $k$ is the largest index less than or equal to $n$ such that the chord progression $\mathcal{C}_k = \langle C_0, C_1, \ldots, C_k \rangle$ has a connected voice leading starting at $V$. In fact, $\mathcal{V}_k = \langle V, V_1, \ldots, V_k \rangle$ turns out to be a voice leading of $\mathcal{C}_k$.*

Concerning the complexity of the algorithm ALGO1, we notice that in the worst case we need to compute all the partial voice leadings $\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_n$; thus the total time spent in the whole process is $\mathcal{O}(n \cdot f(m))$, where $f(m)$ is an upper bound to the time needed to check whether $S_i \subseteq C_{i+1}$ and to construct the voicing $V_{i+1}$, for $i = 0, 1, \ldots, n-1$.

If we represent sets by ordered linear arrays, we get $f(m) = \mathcal{O}(m \log m)$, yielding an overall running time of $\mathcal{O}(nm \log m)$.[7]

However, if the size $\sigma$ of the alphabet $\Sigma = \bigcup_{i=0,1,\ldots,n} C_i$ is sufficiently small, we can conveniently use the bit-parallelism technique [2] to reduce the running time to $\mathcal{O}(n+m)$. Indeed, let us assume that $\sigma \leq \omega$, where $\omega$ is the number of bits in a computer word. Then, any subset of $\Sigma$ can be represented by a bit mask of length $\sigma$, which fits into a computer word. By using such a representation, the set operations of union, intersection, and complement, as well as the set inclusion test, can be executed in constant time by suitable combinations of the bitwise operations "OR", "AND", and "NOT" (denoted by the symbols "$\vee$", "$\wedge$", and "$\sim$", respectively).

More precisely, after fixing an (arbitrary) ordering

$$s_0, s_1, \ldots, s_{\sigma-1}$$

of the symbols of $\Sigma$, we use the following representations:

- a singleton $\{s_i\} \subseteq \Sigma$ is represented as the bit mask $\mathsf{B}(s_i) = b_0 b_1 \cdots b_{\sigma-1}$ (of length $\sigma$), where
$$b_j = \begin{cases} 1 & \text{if } j = \sigma - 1 - i \\ 0 & \text{otherwise}, \end{cases}$$
for $j = 0, 1, \ldots, \sigma - 1$;[8]

- a nonempty subset $A = \{s_{i_0}, s_{i_1}, \ldots, s_{i_k}\}$ of $\Sigma$ is represented as the bit mask
$$\mathsf{B}(A) =_{\text{Def}} \mathsf{B}(s_{i_0}) \vee \mathsf{B}(s_{i_1}) \vee \cdots \vee \mathsf{B}(s_{i_k});$$

- the empty subset of $\Sigma$ is represented by the bit mask $\mathsf{0}^\sigma$, i.e., the string consisting of $\sigma$ copies of the bit $\mathsf{0}$;

- a chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ is represented as an array $\boldsymbol{\mathcal{C}}[0\,..\,n]$ of $n+1$ bit masks, where $\boldsymbol{\mathcal{C}}[i] = \mathsf{B}(C_i)$ for $i = 0, 1, \ldots, n$;[9]

- a voicing $V$, of length $m$, is represented as an array $\boldsymbol{V}[0\,..\,m-1]$ of $m$ bit masks, where $\boldsymbol{V}[i] = \mathsf{B}(V[i])$, for $i = 0, 1, \ldots, m-1$ (this amounts to represent a voicing $V = v_0 v_1 \cdots v_{m-1}$ as the ordered tuple of the bit masks corresponding to the singletons $\{v_0\}, \{v_1\}, \ldots, \{v_{m-1}\}$).

It is convenient to use a queue $\mathcal{Q}$ to store the first $m-1$ symbols (represented as singleton bit masks) of the voicings $V_0, V_1, \ldots, V_n$, as they are generated during the construction process. More precisely, at stage $i$ of the construction, the queue $\mathcal{Q}$ will have the following configuration

$$\mathsf{B}(V_i[0])\,,\ \mathsf{B}(V_i[1])\,,\ \ldots\ ,\ \mathsf{B}(V_i[m-2])\,,$$

[7] Notice that, in practical applications, the value of $m$ could be considered as a constant. For instance, in western music any chord is made up of at most 12 notes (assuming octave equivalence).

[8] Notice that this amounts to representing the singleton $\{s_i\}$ by the "machine integer" $(1 \ll i)$, where "$\ll$" denotes the bitwise operation of left-shifting.

[9] We mention here that a similar representation of chords as bit masks has been used in [4], for a different problem on chord sequences processing.

with the head pointing to the rightmost bit mask, $\mathsf{B}(V_i[m-2])$, and the tail pointing to the leftmost one, $\mathsf{B}(V_i[0])$. Notice that there is no need to store the last symbol of the voicing $V_i$, as the subsequent voicing $V_{i+1}$ is completely determined by the partial voicing $V_i[0\,..\,m-2]$ and by the chord $C_{i+1}$. The collection $S_i = Set(V_i[0\,..\,m-2])$ can be conveniently maintained in a bit mask $\mathsf{S}$, so that the test $S_i \subseteq C_{i+1}$ becomes $\mathsf{S} \wedge \boldsymbol{\mathcal{C}}[i+1] = \mathsf{S}$. Then the construction of the voicing $V_{i+1}$ can be accomplished by the following sequence of steps:

- retrieve the unique element $z$ in $C_{i+1} \setminus S_i$, which will be the first symbol of $V_{i+1}$, by setting $\mathsf{Z} := \boldsymbol{\mathcal{C}}[i+1] \wedge \sim \mathsf{S}$ (plainly, $\mathsf{Z}$ contains the bit mask $\mathsf{B}(z)$);
- retrieve the first bit mask $\mathsf{D}$ in $\mathcal{Q}$ by executing the operation $dequeue(\mathcal{Q})$;
- enqueue $\mathsf{Z}$ in $\mathcal{Q}$.

After these steps, $\mathcal{Q}$ will have the following configuration

$$\mathsf{Z}\,,\ \mathsf{B}(V_i[0])\,,\ \mathsf{B}(V_i[1])\,,\ \ldots\,,\ \mathsf{B}(V_i[m-3])$$

and $V_{i+1}[0\,..\,m-2]$ will be correctly stored in $\mathcal{Q}$. As a final step, $\mathsf{S}$ will be set to $(\mathsf{S} \wedge \sim \mathsf{D}) \vee \mathsf{Z}$, so as to represent the set $S_{i+1}$.

**Remark 2** *Since each dequeue operation on $\mathcal{Q}$ is always followed by an enqueue operation, the queue $\mathcal{Q}$ may be conveniently implemented as an array $\mathbf{Q}[0\,..\,m-2]$ of bit masks with a pointer $h$, which at stage $i$ stores the partial voicing $V_i[0\,..\,m-2]$ into $\mathbf{Q}$ in a circular manner, starting at position $h$. Then a $dequeue(\mathcal{Q})$ operation is just performed by retrieving the element $\mathbf{Q}[h]$ and the subsequent operation $enqueue(\mathcal{Q}, \mathsf{Z})$ is simply performed by setting $\mathbf{Q}[h]$ to $\mathsf{Z}$ and then shifting circularly the pointer $h$ one position to the right.*

The complete algorithm, named ALGO2, is presented in details in Figure 4. By inspection, it is immediate to see that ALGO2 has an $\mathcal{O}(n+m)$-running time, provided that $\sigma \leq \omega$, where $\sigma$ is the size of the alphabet and $\omega$ is the length of a computer word. However, if $\sigma > \omega$, then we need $\lceil \sigma/w \rceil$ computer words to represent a subset of $\Sigma$, and in this case the running time of the algorithm ALGO2 increases to $\mathcal{O}((n+m)\lceil \sigma/w \rceil)$.

**Remark 3** *Analogously to the observation in Remark 1 relative to the algorithm ALGO1, also algorithm ALGO2 can be adapted so as to produce as output the longest connected voice leading starting at $V$ (of an initial segment) of $\mathcal{C}$, by using an additional string-variable $X$ and adding the following lines of code between lines 11 and 12:*

```
X := decode(Z)
for j := 0 to m − 2 do
    X := X • decode(Q[(h + m − 2 − j) mod (m − 1)])
OUTPUT(X)
```

*The one-argument function "decode" yields the symbol $s_i$, when applied to the bit mask $\mathsf{B}(s_i)$ which represents the singleton $\{s_i\}$, for $s_i \in \Sigma$. It admits the following simple implementation. To begin with, we represent the alphabet $\Sigma$ as an array $\Sigma[0\,..\,\sigma-1]$, such that $\Sigma[i] = s_i$, for $i = 0, 1, \ldots, \sigma-1$. Then, if $x = \mathsf{B}(s_i)$, we*

```
ALGO2(C, V, W, σ)
 1.  m := |V|
 2.  n := length(C) − 1
 3.  for h := m − 2 down to 0 do
 4.      Q[h] := V[m − 2 − h]
 5.  S := 0^σ
 6.  for i := 0 to m − 2 do
 7.      S := S ∨ V[i]
 8.  h := 0
 9.  for i := 1 to n do
10.      if (C[i] ∧ S) = S then
11.          Z := (C[i] ∧ ∼ S)
12.          D := Q[h]
13.          Q[h] := Z
14.          h := (h + 1) mod (m − 1)
15.          S := (S ∧ ∼ D) ∨ Z
16.      else
17.          return false
18.  for j := 0 to m − 2 do
19.      if Q[(h + j) mod (m − 1)] ≠ W[m − 2 − j] then
20.          return false
21.  return true
```

Fig. 4. An optimized variant with bit-parallelism of the algorithm ALGO1. The parameter $\sigma$ stands for the size of the alphabet.

*have immediately* $s_i = \Sigma[\lceil \log_2 x \rceil]$, *for* $i = 0, 1, \ldots, \sigma - 1$, *so that* $decode(x) = \Sigma[\lceil \log_2 x \rceil]$. *If we further assume that the symbols of the alphabet* $\Sigma$ *are the first* $\sigma$ *nonnegative integers, i.e.,* $s_i = i$, *for* $0 \leq i \leq \sigma - 1$, *then we have more simply* $decode(x) =_{Def} \lceil \log_2 x \rceil$. *Notice also that under such an assumption, we have* $\mathsf{B}(s) = (1 \ll s)$, *where* $\ll$ *denotes the bitwise operation of left-shifting, and therefore the coding of a singleton* $\{s\}$ *as the bit mask* $\mathsf{B}(s)$ *can be performed in constant time too, for any symbol* $s \in \Sigma$.

*Notice, however, that if we modify the algorithm ALGO2 so as to output a voice leading as described above, its running time increases to* $\mathcal{O}(nm)$, *provided that* $\log_2 x$ *can be computed in constant time.*

Next, we turn our attention to the main problem of the section, namely the problem of determining whether a chord progression is regular.

**Problem 2** *Given a chord progression* $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$, *check whether* $\mathcal{C}$ *is regular.*

A natural but inefficient solution to Problem 2 is the following one. Let $m$ be the size of the chords $C_0, C_1, \ldots, C_n$. We start by checking that $C_i \neq C_{i+1}$ holds, for $i = 0, 1, \ldots, n - 1$. Then we form all possible voicings of the first chord $C_0$, and for each such voicing $V$ we run the algorithm ALGO1 to search for a connected

voice leading of $\mathcal{C}$ from $V$ to the voicing $W = V[1\,..\,m-1]\bullet w$, where $w$ is the only symbol of $C_n$ not contained in $C_0$ (if, indeed, $C_n = C_0$ or $C_n \setminus C_0$ contains more than one symbol, then, certainly, $\mathcal{C}$ would not be regular). But since there are $m!$ possible voicings of $C_0$, such an approach is not of any practical interest.

However, we note some facts. First of all, given the two distinct chords $C_i$ and $C_{i+1}$, we have that $C_i \longrightarrow C_{i+1}$ if and only if $C_i$ and $C_{i+1}$ share exactly $m-1$ symbols. Thus we can check easily if $C_0 \longrightarrow C_1 \longrightarrow \cdots \longrightarrow C_n \longrightarrow C_0$, which is a necessary condition for the chord progression $\mathcal{C}$ to be regular. Thence, if we find a pair of chords $C_i$ and $C_{i+1}$ such that $C_i \longrightarrow C_{i+1}$ does not hold, we conclude immediately that $\mathcal{C}$ is not regular. But unfortunately the condition $C_0 \longrightarrow C_1 \longrightarrow \cdots \longrightarrow C_n \longrightarrow C_0$ is not, in general, a sufficient condition for $\mathcal{C}$ to be regular. For instance, let us consider the chords $C' = \{a, b, c\}$, $C'' = \{a, b, x\}$ and $C''' = \{a, b, d\}$. Although $C' \longrightarrow C'' \longrightarrow C''' \longrightarrow C'$ and $C' \neq C'' \neq C''' \neq C'$, the chord progression $\langle C', C'', C'''\rangle$ is not regular.

We observe, however, that if the chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n\rangle$ is regular, and we set

$$X_k = \bigcap_{i=0}^{m-1-k} C_i\,, \quad \text{for } k = 0, 1, \ldots, m-1,$$ [10]

then each of the first $m-1$ sets in the sequence $X_0, X_1, \ldots, X_{m-1}$ must be a nonempty proper subset of the set which immediately follows it, i.e., there must exist $m-1$ distinct symbols $c_0, c_1, \ldots, c_{m-2}$ of $C_0$ such that

$$X_0 = \{c_0\}\,, \quad X_1 = \{c_0, c_1\}\,, \quad \ldots\,, \quad X_{m-2} = \{c_0, c_1, \ldots, c_{m-2}\}.$$

Additionally, if $c_{m-1}$ is the symbol of $C_0$ distinct from $c_0, c_1, \ldots, c_{m-2}$, then a circularly connected voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n\rangle$ of $\mathcal{C}$ must begin necessarily with the voicing $c_0c_1\cdots c_{m-2}c_{m-1}$, i.e. $V_0[i] = c_i$, for $i = 0, 1, \ldots, m-1$. These considerations are immediate consequences of Theorem A.3 in the appendix. Furthermore, we must also have that $C_n = \{c_1, \ldots, c_{m-1}, w\}$, for some symbol $w$ distinct from $c_0$, because $C_n \neq C_0$ and $V_n \Longrightarrow V_0$, with $V_n$ a voicing of $C_n$.

Then, given the chord progression $\mathcal{C}$, in order to check whether $\mathcal{C}$ is regular, we can proceed as follows. We begin by forming the sets $X_0, X_1, \ldots, X_{m-1}$, and check whether $|X_{k+1} \setminus X_k| = 1$, for $k = 0, 1, \ldots, m-2$. If this is not the case, we conclude immediately that $\mathcal{C}$ is not regular. Otherwise, we extract the symbols $c_0, c_1, \ldots, c_{m-1}$ such that $c_{k+1} \in X_{k+1} \setminus X_k$, for $0 \leq k \leq m-2$, and $c_0 \in X_0$, and then check whether $C_n = \{c_1, \ldots, c_{m-1}, w\}$, for some symbol $w$ distinct from $c_0$. If this is not the case, we conclude again that $\mathcal{C}$ is not regular; otherwise we form the voicings $V = c_0c_1\cdots c_{m-1}$ and $W = c_1c_2\cdots c_{m-1}w$, and run the algorithm ALGO1 with inputs $\mathcal{C}$, $V$, and $W$ to search for a connected voice leading of $\mathcal{C}$ from $V$ to $W$. The resulting algorithm, named ALGO3, is presented in Figure 5. Plainly, the time complexity of ALGO3 is $\mathcal{O}(nm\log m + m^2)$ (at least in the case in which sets are represented as ordered linear arrays.)

[10] Notice that $\mathcal{C}$ cannot be regular unless $n \geq m-1$.

```
ALGO3(C)
 1.  m := |C[0]|
 2.  n := length(C) − 1
 3.  for i := 0 to n − 1 do
 4.      if C[i] = C[i + 1] then
 5.          return false
 6.  X_{m−1} := C[0]
 7.  for k := m − 2 down to 0 do
 8.      X_k := X_{k+1} ∩ C[m − k − 1]
 9.      if |X_{k+1} \ X_k| = 1 then
10.          - let z be such that X_{k+1} = X_k ∪ {z}
11.          V[k + 1] := W[k] := z
12.      else
13.          return false
14.  - let c be such that X_0 = {c}
15.  V[0] := c
16.  if c ∉ C[n] and |C[n] ∩ C[0]| = m − 1 then
17.      - let w be such that C[n] \ C[0] = {w}
18.      W[m − 1] := w
19.      return ALGO1(C, V, W)
20.  else
21.      return false
```

Fig. 5. Pseudo-code of the algorithm ALGO3 for checking whether a given chord progression $\mathcal{C}$ is regular.

However, by using the bit-parallelism technique, namely by representing as before sets as bit masks and voicings as arrays of bit masks, we obtain an efficient variant of the algorithm ALGO3, which we call ALGO4. The algorithm ALGO4, shown in Figure 6, uses the algorithm ALGO2 (the variant of ALGO1 based on bit-parallelism) as a subroutine. It assumes that the input chord progression $\mathcal{C}$ is given as an array $\boldsymbol{\mathcal{C}}$ of bit masks representing the chords in $\mathcal{C}$. By a simple inspection, it is easy to see that when $\sigma \leq \omega$, where $\sigma$ is the size of the alphabet and $\omega$ is the length of a computer word, the algorithm ALGO4 has an $\mathcal{O}(n + m)$-running time and requires only $\mathcal{O}(m)$-extra space, for chord progressions of length $n$ whose chords have size $m$. In the general case, the algorithm ALGO4 has $\mathcal{O}((n + m)\lceil\sigma/\omega\rceil)$-running time and requires $\mathcal{O}(m\lceil\sigma/\omega\rceil)$-space.

### 3.1. *Further questions on the connectivity of chords and voicings*

We discuss next a few further questions concerning the connectivity of chords and voicings. We begin by observing that

**Property 1** *Any two chords of the same size can always be connected by a voice leading.*

Indeed, let $C'$ and $C''$ be two chords of size $m$, and let $V_0$ be any voicing of $C'$. We define a connected voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_m \rangle$, in such a way that

```
ALGO4(C, m, σ)
 1.  n := length(C) − 1
 2.  for i := 0 to n − 1 do
 3.      if C[i] = C[i + 1] then
 4.          return false
 5.  X := C[0]
 6.  for k := m − 2 down to 0 do
 7.      Y := X ∧ C[m − k − 1]
 8.      if Y ≠ 0^σ and Y ≠ X then
 9.          V[k + 1] := W[k] := X∧ ∼ Y
10.          X := Y
11.      else
12.          return false
13.  V[0] := X
14.  if X ∧ C[n] = 0^σ and (C[n] ∧ C[0]) ∨ X = C[0] then
15.      W[m − 1] := C[n]∧ ∼ C[0]
16.      return ALGO2(C, V, W, σ)
17.  else
18.      return false
```

Fig. 6. An optimized variant with bit-parallelism of the algorithm ALGO3. The parameters $m$ and $\sigma$ stand respectively for the size of the chords and the size of the alphabet.

- $V_{i+1}[1\,..\,m-1] = V_i[0\,..\,m-2]$, and
- $V_{i+1}[0]$ is any symbol in $C'' \setminus \mathit{Set}(V_{i+1}[1\,..\,m-1])$,

for $i = 0, 1, \ldots, m-2$.

Since $V_m$ is a voicing of $C''$, it follows that $\mathcal{V}$ is indeed a voice leading connecting $C'$ to $C''$.

Notice that in the above construction the voicing $V_0$ of $C'$ has been selected arbitrarily. Therefore, we can conclude that the following property holds too:

**Property 2** *Any given chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ can always be extended to a connected chord progression $\mathcal{C}' = \langle C'_0, C'_1, \ldots, C'_p \rangle$, in the sense that $C_i = C'_{k_i}$, for some strictly increasing sequence of indices $0 \leq k_i \leq p$, for $i = 0, 1, \ldots, n$.*

An interesting problem is then the following:

**Open Problem 1** *Find a connected chord progression of* minimal *length which extends a given chord progression.*

The connectivity relation between voicings depends on the richness of the alphabet. For instance, let us consider the voicings $V = abcd$ and $W = abdc$ of the same chord $C = \{a, b, c, d\}$. If we try to connect $V$ to $W$ by using only symbols of the alphabet $\Sigma = \{a, b, c, d\}$, then we end up with the "periodic" voice leading

$$\mathcal{V} = \langle V_1, V_2, V_3, V_4,\ V_1, V_2, V_3, V_4,\ V_1, V_2, V_3, V_4, \ldots \rangle\,,$$

where $V_1 = V = abcd$, $V_2 = dabc$, $V_3 = cdab$ and $V_4 = bcda$, proving that $V$ can not be connected to $W$ with respect to the alphabet $\Sigma$.

However, if we are allowed to use a new symbol, say $x$, then it is immediate to see that

$$\langle abcd, xabc, cxab, dcxa, bdcx, abdc \rangle$$

is a voice leading which connects $V$ to $W$ (with respect to the alphabet $\Sigma \cup \{x\}$).

An immediate consequence of Corollary A.1 in the appendix is the following connectability test for voicings:

> Given any two voicings $V$ and $W$ of the same length over an alphabet $\Sigma$, if $Set(V) \neq \Sigma$ or $Set(W) \neq \Sigma$, then $V$ can be connected to $W$ with respect to $\Sigma$, otherwise $V$ can be connected to $W$ if and only if $W$ is a substring of $V_{\bullet}V$.

But despite the simplicity of the above test, the related optimization problem does not seem to possess a simple and efficient algorithmic solution:

**Open Problem 2** *Given two voicings $V$ and $W$ of the same length over an alphabet $\Sigma$, determine a shortest voice leading which connects $V$ to $W$ with respect to $\Sigma$.*

**Remark 4** *Notice that Problems 1 and 2 above may have practical applications in various musical situations, as for instance in the case in which one wants to compose a chord progression by using certain fixed or preferred chords or chord-voicings, possibly interspacing them by some other chords, and the length of the chord progression is constrained so as to fit within a given maximum number of available musical bars.*

Concerning Problem 2, notice that from Theorems A.1 and A.2 in the appendix it follows that if $V$ is connectable to $W$ with respect to $\Sigma$, then $V$ can be connected to $W$ by a voice leading (over $\Sigma$) of length at most $m^2 - m + 2$, where $m$ is the length of $V$ (and $W$). In fact, when the size of the alphabet $\Sigma$ is at least $3m - 2$, then, as shown in the second part of Theorem A.2, the length of a minimal voice leading which connects $V$ to $W$ is bounded above by $2m$.

The above considerations show that in general the problem of minimally-connecting two given voicings depends also on the size of the alphabet $\Sigma$ under consideration (and not only on the length of the voicings). This is also illustrated by the following example. Let us consider the voicings $V = axbcd$ and $W = abcxd$ and the alphabets $\Sigma = \{a, b, c, d, x, y\}$ and $\Omega = \{a, b, c, d, x, y, z\}$. It is not difficult to see that there is a connected voice leading over $\Omega$ from $V$ to $W$ of length 8 and a connected voice leading over $\Sigma$ from $V$ to $W$ of length 11. (Simply consider the strings $S_1 = abcxdyzaxbcd$ and $S_2 = abcxdabcydaxbcd$ and take the substrings of length $m = 5$ of the form $S_1[12 - m - i \,..\, 12 - 1 - i]$ and $S_2[15 - m - j \,..\, 15 - 1 - j]$, for $0 \leq i < 8$ and $0 \leq j < 11$. See also Lemma A.2 in the appendix.) Moreover, it can easily be verified that no connected voice leading over $\Sigma$ from $V$ to $W$ of length less than 11 exists.

Notice also that Problem 2 could be solved, at least in principle, by running any algorithm for the *Single Source (Single Destination) Shortest Path Problem* (e.g.,

Dijkstra's algorithm) on the weighted, directed graph $\mathcal{G}_\Sigma$ whose vertices are the voicings over the alphabet $\Sigma$ of a fixed length $m$, and whose edges are all pairs of voicings $(V, W)$ such that $V \Longrightarrow W$, each of which has weight 1. However, the main problem with such approach is in the practical construction of the graph $\mathcal{G}_\Sigma$, since it has $\frac{\sigma!}{(\sigma-m)!}$ vertices and $\frac{\sigma!(\sigma-m+1)}{(\sigma-m)!}$ edges, where $\sigma$ is the size of $\Sigma$.

Another interesting question related to the connectivity relation between voicings, with applications in music composition, is the following one. When a composer is engaged in assembling a harmonic progression, sometimes he or she has at hand only a limited number of available tones to form the various chords of the progression; this is the case, for instance, when the notes must belong to a particular scale, such as a pentatonic scale, or a diatonic scale, or similar. In this case, the above cited Corollary A.1 has the consequence that if $V$ and $W$ are voicings of two distinct chords, then no additional tone is required to connect $V$ to $W$. However, the corollary does not say anything on the fact that a voice leading $\mathcal{V}$ which connects $V$ to $W$ has to satisfy the additional property that any two or more consecutive voicings of $\mathcal{V}$ must have distinct base chords.

The ability of creating harmonic progressions with a certain degree of "dissimilarity" between consecutive chords is an important issue in order for a harmonic progression not to result too monotonous or uninteresting. From Corollary A.2 in the appendix, it follows that two extra symbols suffice to allow any two voicings $V$ and $W$ of the same length to be connected by a voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ such that $Set(V_i) \neq Set(V_{i+1})$, for $i = 0, 1, \ldots, n-1$. This implies also that any given chord progression $\mathcal{C}$ can always be extended to a regular chord progression by adding at most two new symbols to the alphabet. Moreover, from Lemma A.1(2) it follows that if a chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ is connected and satisfies the property that $C_0 \neq C_1 \neq \ldots \neq C_n$, then $\mathcal{C}$ can be prolonged to a regular chord progression without using any additional symbol. A significant question is then the following optimization problem:

**Open Problem 3** *Given a chord progression $\mathcal{C} = \langle C_0, C_1, \ldots C_n \rangle$ over an alphabet $\Sigma$ and a fixed bound $k > n$, determine the minimum number of new symbols that must be added to $\Sigma$ in order that $\mathcal{C}$ can be extended to a regular chord progression of length at most $k$.*

## 4. Conclusions

We have presented some combinatorial problems on strings which arise in the fields of music processing and music analysis. We have also provided algorithms to solve some of these problems, whereas a few of them have just been raised and left unsolved (at least in the sense that no efficient algorithm has been provided). We plan to address in more detail such problems in the future and to provide also efficient exact or approximated algorithmic solutions to them.

## Appendix A: Some theoretical results on voicings and connected chord progressions

In this appendix we present in details some theoretical results that have been already referenced in Section 3.

To begin with, we show a few properties of the connectivity relation $\underset{\Sigma}{\Longrightarrow}^{+}$, implying that it is an equivalence relation.

**Lemma A.1** *Let $V$, $W$, and $Z$ be voicings over an alphabet $\Sigma$. Then*

*(1)* $V \underset{\Sigma}{\Longrightarrow}^{+} V$.

*(2) If* $V \Longrightarrow W$ *then* $W \underset{\Sigma}{\Longrightarrow}^{+} V$. *Moreover, if* $Set(V) \neq Set(W)$, *then* $W$ *can be connected to* $V$ *by a voice leading* $\langle V_0, V_1, \ldots, V_m \rangle$ *over* $\Sigma$ *such that* $Set(V_i) \neq Set(V_{i+1})$, *for* $i = 0, 1, \ldots, m-1$.

*(3) If* $V \underset{\Sigma}{\Longrightarrow}^{+} W$ *then* $W \underset{\Sigma}{\Longrightarrow}^{+} V$.

*(4) If* $V \underset{\Sigma}{\Longrightarrow}^{+} W$ *and* $W \underset{\Sigma}{\Longrightarrow}^{+} Z$, *then* $V \underset{\Sigma}{\Longrightarrow}^{+} Z$.

*Therefore the relation* $\underset{\Sigma}{\Longrightarrow}^{+}$ *is an equivalence relation.*

**Proof.** We give only the proof of (1) and (2), since (4) is an immediate consequence of the definition of the relation $\underset{\Sigma}{\Longrightarrow}^{+}$ and (3) follows from (2) and (4).

Let $V$, $W$, and $Z$ be voicings of the same length $m$, over the alphabet $\Sigma$.

Concerning (1), it can easily be verified that $V$ is connected to $V$ by the voice leading $\langle V_0, V_1, \ldots, V_m \rangle$, where $V_0 = V$ and

$$V_{i+1} =_{\text{Def}} V_i[m-1] \centerdot V_i[0 \,..\, m-2]\,,$$

for $i = 0, 1, \ldots, m-1$.

Next, let $V \Longrightarrow W$. In order to verify (2), we distinguish two cases, according to whether $V[m-1] = W[0]$ or $V[m-1] \neq W[0]$. If $V[m-1] = W[0]$ (i.e., if $Set(V) = Set(W)$), then $W$ is connected to $V$ by the voice leading $\langle V_0, V_1, \ldots, V_{m-1} \rangle$, where $V_0 = W$ and

$$V_{i+1} =_{\text{Def}} V_i[m-1] \centerdot V_i[0 \,..\, m-2]\,,$$

for $i = 0, 1, \ldots, m-2$.

On the other hand, if $V[m-1] \neq W[0]$ (i.e., if $Set(V) \neq Set(W)$), then $W$ is connected to $V$ by the voice leading $\langle V_0, V_1, \ldots, V_m \rangle$, where $V_0 = W$ and

$$V_{i+1} =_{\text{Def}} V[m-i-1] \centerdot V_i[0 \,..\, m-2]\,,$$

for $i = 0, 1, \ldots, m-1$.

It can be easily verified that $V_{i+1}[0] \notin Set(V_i)$, for $i = 0, 1, \ldots, m-1$, and thus $Set(V_0) \neq Set(V_1) \neq \ldots \neq Set(V_m)$, as required. □

Before proceeding with further properties, we need to introduce some definitions. Let $X$ and $Y$ be strings. We say that $X$ is a PREFIX (resp., SUFFIX) of $Y$, and write $X \sqsubset Y$ (resp., $X \sqsupset Y$), if $Y = X \centerdot Z$ (resp., $Y = Z \centerdot X$) for some string $Z$. Moreover, we write $X \prec Y$ to mean that $X$ is a substring of $Y$, i.e., $Y = U \centerdot X \centerdot Z$

for some strings $U$ and $Z$. The concatenation of a string $X$ with itself, i.e., the string $X \centerdot X$, will be denoted by $X^{[2]}$. If $s$ is any symbol and $0 \leq k < |X|$, we denote by $X(k:s)$ the string obtained from $X$ by replacing its $(k+1)$-st symbol by $s$, so that the following equality holds

$$X(k:s) = X[0\,..\,k-1] \centerdot s \centerdot X[k+1\,..\,|X|-1]\,.$$

Moreover, we put also $X(k:s) = X$ when $k \geq |X|$.

Given a positive integer $m$, we say that a string $X$ is an $m$-VOICING if any substring of $X$ of length $m$ is a voicing or, more formally, if for every pair of indices $i$ and $j$, with $0 \leq i < j < |X|$ and such that $j - i < m$, one has $X[i] \neq X[j]$. If $X$ is not an $m$-voicing, then any pair of indices $(i,j)$ such that $0 \leq i < j < |X|$, $j - i < m$ and $X[i] = X[j]$ is called an $m$-TIE of $X$.

**Lemma A.2** *Let $V$ and $W$ be voicings of length $m$ over an alphabet $\Sigma$. Then, for each $n > 0$, $V \underset{\Sigma}{\Longrightarrow}^{n} W$ if and only if there is a string $S$ over $\Sigma$, of length $m+n$, such that*

*(1) $W \sqsubset S$,*

*(2) $V \sqsupset S$, and*

*(3) $S$ is an $m$-voicing.*

**Proof.** Let $V$ and $W$ be voicings of length $m$ over an alphabet $\Sigma$.

We show first that the conditions of the lemma are necessary. Thus, let us suppose that $V \underset{\Sigma}{\Longrightarrow}^{n} W$, and let $\mathcal{Z} = \langle Z_0, Z_1, \ldots Z_n \rangle$ be a connected voice leading over $\Sigma$ from $V$ to $W$ of length $n+1$, so that $Z_0 = V$ and $Z_n = W$. We define a sequence $S_0, S_1, \ldots, S_n$ of strings over $\Sigma$ according to the following recursive relations:

$$\begin{array}{lcll} S_0 & = & Z_0, & \\ S_i & = & Z_i[0] \centerdot S_{i-1}, & \text{for } i = 1, 2, \ldots, n. \end{array}$$

We claim that the string $S_n$ has length $m+n$ and satisfies conditions (1), (2), and (3) above. To begin with, notice that $|S_0| = m$ and $|S_i| = 1 + |S_{i-1}|$, for $i = 1, 2, \ldots, n$. Thus $|S_i| = m + i$, for $i = 0, 1, \ldots, n$, and therefore $|S_n| = m + n$. Observe also that $S_0 \sqsupset S_i$, for $i = 0, 1, \ldots, n$, and thus, in particular, $V = Z_0 = S_0 \sqsupset S_n$, showing that condition (2) holds. It remains only to show that conditions (1) and (3) hold too. We do this by showing by induction on $i = 0, 1, \ldots, n$ that $Z_i \sqsubset S_i$ and that $S_i$ is an $m$-voicing. The base case, for $i = 0$, is trivial. For the inductive step, let $0 < i \leq n$, and let us suppose that $Z_{i-1} \sqsubset S_{i-1}$ and that $S_{i-1}$ is an $m$-voicing. Since $Z_{i-1} \sqsubset S_{i-1}$, we have that $S_{i-1} = Z_{i-1} \centerdot U$, for some string $U$ of length $i-1$. Hence $S_i = Z_i[0] \centerdot Z_{i-1} \centerdot U = Z_i[0] \centerdot Z_{i-1}[0\,..\,m-2] \centerdot Z_{i-1}[m-1] \centerdot U$. But since $Z_{i-1} \Longrightarrow Z_i$, we have also that $Z_{i-1}[0\,..\,m-2] = Z_i[1\,..\,m-1]$, and thus $S_i = Z_i[0] \centerdot Z_i[1\,..\,m-1] \centerdot Z_{i-1}[m-1] \centerdot U = Z_i \centerdot Z_{i-1}[m-1] \centerdot U$, from which $Z_i \sqsubset S_i$ follows. Next, in order to show that $S_i$ is an $m$-voicing, let us assume, by way of contradiction, that $S_i$ has an $m$-tie $(a,b)$. Then, since $S_i = Z_i[0] \centerdot S_{i-1}$, and $S_{i-1}$ is an $m$-voicing, we must have $a = 0$, because otherwise $(a,b)$ would be an $m$-tie of $S_{i-1}$. This implies that $(a,b)$ is an $m$-tie of the prefix of $S_i$ of length $m$ (since

$b - a < m$). But we have just shown that the prefix of $S_i$ of length $m$ is exactly the voicing $Z_i$, which by the very definition of voicings is an $m$-voicing, a contradiction. This concludes the proof of the necessity part of the lemma.

Next, for the sufficiency part, let $S$ be an $m$-voicing over $\Sigma$ of length $m+n$ such that $W \sqsubset S$ and $V \sqsupset S$. Let us define a sequence $V_0, V_1, \ldots, V_n$ of strings over $\Sigma$ of length $m$ by putting

$$V_i =_{\text{Def}} S[|S| - m - i \,..\, |S| - 1 - i], \quad \text{for } i = 0, 1, \ldots, n\,.$$

Plainly, $V_0 = V$ and $V_n = W$. Moreover, each $V_i$ is a voicing, since the string $S$ is an $m$-voicing. Then, by construction, we also have that $V_i \underset{\Sigma}{\Longrightarrow} V_{i+1}$, for $i = 1, 2, \ldots, n-1$. Thus $\langle V_0, V_1, \ldots, V_n \rangle$ is a connected voice leading over $\Sigma$ from $V$ to $W$ of length $n+1$, concluding the proof of the sufficiency part of the lemma. □

**Lemma A.3** *Let $X$ be a string of length $m \geq 2$. Then,*

$$X^{[2]}[k + m - 1] \bullet X^{[2]}[k \,..\, k + m - 2] \prec X^{[2]}$$

*for any $k \in \{0, 1, \ldots, m\}$.*

**Proof.** Simply note that

$$X^{[2]}[k + m - 1] \bullet X^{[2]}[k \,..\, k + m - 2] = \begin{cases} X^{[2]}[k - 1 \,..\, k + m - 2], & \text{if } 0 < k \leq m \\ X^{[2]}[m - 1 \,..\, 2m - 2], & \text{if } k = 0. \end{cases}$$

□

**Theorem A.1** *Let $V$ and $W$ be voicings of length $m$ over an alphabet $\Sigma$. The following facts hold:*

*(1) If $|\Sigma| = m$ and $V \underset{\Sigma}{\Longrightarrow}^{+} W$ then $W \prec V^{[2]}$.*

*(2) If $W \prec V^{[2]}$ then $V \underset{\Sigma}{\Longrightarrow}^{n} W$, for some $0 < n \leq m$.*

**Proof.** Let $V$ and $W$ be voicings of length $m$ over a given alphabet $\Sigma$.

Let us first assume that $|\Sigma| = m$ and that $V \underset{\Sigma}{\Longrightarrow}^{+} W$, and let $\langle V_0, V_1, \ldots, V_n \rangle$ be a connected voice leading over $\Sigma$ from $V$ to $W$ of length $n+1$. To show (1), it is enough to prove by induction on $i = 0, 1, \ldots, n$ that $V_i \prec V^{[2]}$. The base case $i = 0$ is immediate, since $V_0 = V \prec V^{[2]}$. For $i > 0$, let us assume that $V_{i-1} \prec V^{[2]}$, i.e., $V_{i-1} = V^{[2]}[k \,..\, k+m-1]$, for some $0 \leq k \leq m$. Then, since $V_{i-1}$ and $V_i$ are voicings of length $m$ over the alphabet $\Sigma$ and we have assumed that $|\Sigma| = m$, we plainly have that $Set(V_{i-1}) = \Sigma = Set(V_i)$. Moreover, $V_{i-1} \underset{\Sigma}{\Longrightarrow} V_i$, and therefore $V_i = V_{i-1}[m-1] \bullet V_{i-1}[0 \,..\, m-2]$, which implies $V_i = V^{[2]}[k+m-1] \bullet V^{[2]}[k \,..\, k+m-2]$. By Lemma A.3 it then follows $V_i \prec V^{[2]}$.

Concerning (2), suppose that $W \prec V^{[2]}$, and let $S = V^{[2]}[\ell \,..\, 2m-1]$, where

$$\ell = \min\{0 \leq i < 2m : W \sqsubset V^{[2]}[i \,..\, 2m-1]\}.$$

Notice that $W \sqsubset S$ trivially holds. We claim that $V \sqsupset S$ and $|S| > m$ hold too. Indeed, since $|W| = m$ and $W \sqsubset S$, we have that the string $S$ is a suffix of $V^{[2]}$ of

length at least $m$. Moreover, since $V$ is by itself a suffix of $V^{[2]}$ of length $m$, we have also that $V \sqsupset S$. Therefore, $W \sqsubset S$ and $V \sqsupset S$. To show that $|S| > m$ holds, let us suppose, by way of contradiction, that $|S| \leq m$. In this case we would plainly have that $V = W$, since $W \sqsubset S$, $V \sqsupset S$, and $|V| = |W| = m$. Therefore $\ell = 0$, which in turn implies $|S| = 2m$. Hence $2m \leq m$, which is a contradiction, since $m > 0$, completing the proof of our claim. To conclude the proof of the theorem, it is now enough to observe that $S$ is an $m$-voicing and apply Lemma A.2. □

**Theorem A.2** *Let $V$ and $W$ be voicings of length $m$ over an alphabet $\Sigma$ such that $|\Sigma| > m$. Then, $V \underset{\Sigma}{\Longrightarrow}^h W$ for some $0 < h \leq m^2 - m + 1$. Moreover, if $|\Sigma| \geq 3m - 2$ then $V \underset{\Sigma}{\Longrightarrow}^h W$, for some $0 < h \leq 2m - 1$.*

**Proof.** Let $V$ and $W$ be voicings of length $m$ over an alphabet $\Sigma$ such that $|\Sigma| > m$.

To begin with, we prove by induction that for each $i = 0, 1, \ldots, m-1$ there is a voicing $Z_i$ over $\Sigma$ of length $m$ such that $V[m-i-1\,..\,m-2] \sqsupset Z_i$ and $Z_i \underset{\Sigma}{\Longrightarrow}^h W$, for some $0 \leq h \leq mi$.

For $i = 0$, it is enough to take $Z_0 = W$, since $V[m-1\,..\,m-2] = \varepsilon \sqsupset W$, and $W \underset{\Sigma}{\Longrightarrow}^0 W$ (by definition).

For the inductive step, let $0 \leq i < m-1$, and let us assume that there is a voicing $Z_i$ over $\Sigma$ of length $m$ such that $V[m-i-1\,..\,m-2] \sqsupset Z_i$ and $Z_i \underset{\Sigma}{\Longrightarrow}^k W$ for some $0 \leq k \leq mi$. Then, our task is to construct a voicing $Z_{i+1}$ over $\Sigma$, of length $m$, such that $V[m-i-2\,..\,m-2] \sqsupset Z_{i+1}$ and $Z_{i+1} \underset{\Sigma}{\Longrightarrow}^h W$, for some $0 \leq h \leq m(i+1)$. First of all, we may assume w.l.o.g. that $Z_i[m-i-1] \neq V[m-i-2]$, since otherwise it would be enough to take $Z_{i+1} = Z_i$. Under such assumption we will construct the voicing $Z_{i+1}$ in such a way that $V[m-i-2\,..\,m-2] \sqsupset Z_{i+1}$ and the string $Z_i \bullet Z_{i+1}$ is an $m$-voicing. We begin by replacing the $(m-i)$-th symbol of $Z_i$ by $V[m-i-2]$, obtaining the string $U = Z_i(m-i-1 : V[m-i-2])$ which plainly satisfies $V[m-i-2\,..\,m-2] \sqsupset U$. If $V[m-i-2] \notin \mathit{Set}(U[0\,..\,m-i-2])$, then we claim that the string $Z_i \bullet U$ is an $m$-voicing, so that we can take $Z_{i+1} = U$. Indeed, if $Z_i \bullet U$ were not an $m$-voicing then we would have $V[m-i-2] = Z_i[j]$, for some $m-i-1 < j < m$ (the only $m$-tie of $Z_i \bullet U$ could be the one ending at position $2m-i-1$, created by the newly inserted symbol $V[m-i-2]$), and thus $V[j-1] = V[m-i-2]$ with $j-1 > m-i-2$. But this is impossible, since $V$ is a voicing. On the other hand, if $V[m-i-2] \in \mathit{Set}(U[0\,..\,m-i-2])$, by choosing $\ell \in \{0, \ldots, m-i-2\}$ in such a way that $U[\ell] = V[m-i-2]$, we put $Z_{i+1} = U(\ell : s)$, where $s$ is any symbol in $\Sigma \setminus \mathit{Set}(Z_i)$ (notice that the set $\Sigma \setminus \mathit{Set}(Z_i)$ is nonempty, since $|Z_i| = m < |\Sigma|$). In any case, the string $Z_{i+1}$ can be formally defined as $Z_{i+1} = U(\ell : s)$, where

- $s \in \Sigma \setminus \mathit{Set}(Z_i)$,
- $U = Z_i(m-i-1, V[m-i-2])$, and
- $\ell = \min(\{0 \leq j < m-i-1 : U[j] = V[m-i-2]\} \cup \{m\})$.

Now, since the string $Z_i \bullet Z_{i+1}$ is an $m$-voicing and has length $2m$, by Lemma A.2 we have that $Z_{i+1} \underset{\Sigma}{\Longrightarrow}^m Z_i$. The latter, together with the assumption that $Z_i \underset{\Sigma}{\Longrightarrow}^k W$ for some $0 \leq k \leq mi$, yields $Z_{i+1} \underset{\Sigma}{\Longrightarrow}^{m+k} W$. To conclude the proof of the inductive step, we have only to observe that $m + k \leq m(i+1)$ plainly holds, since $k \leq mi$.

Summing up, so far we have shown that for each $i = 0, 1, \ldots, m-1$ there is a

voicing $Z_i$ over $\Sigma$ of length $m$ such that $V[m-i-1\,..\,m-2] \sqsupset Z_i$ and $Z_i \underset{\Sigma}{\Longrightarrow}^h W$, for some $0 \leq h \leq mi$.

Coming back to the proof of the theorem, let $Z = Z_{m-1}$. Then, since $V[0\,..\,m-2] \sqsupset Z$, we have that $V \underset{\Sigma}{\Longrightarrow} Z$. And since we have also that $Z \underset{\Sigma}{\Longrightarrow}^h W$, for some $0 \leq h \leq m(m-1)$, it follows that $V \underset{\Sigma}{\Longrightarrow}^{h+1} W$, with $0 < h+1 \leq m^2 - m + 1$, proving the first part of the theorem.

Concerning the second part of the theorem, let us suppose that $|\Sigma| \geq 3m-2$. Then, if $|Set(V) \cup Set(W)| = 2m$, it is immediate to verify that the string $W \centerdot V$ is an $m$-voicing, and hence, by Lemma A.2, we have that $V \underset{\Sigma}{\Longrightarrow}^m W$, with $m \leq 2m-1$. On the other hand, if $|Set(V) \cup Set(W)| < 2m$, then $|\Sigma \backslash (Set(V) \cup Set(W))| \geq m-1$, and therefore we can pick $m-1$ distinct symbols $s_0, s_1, \ldots, s_{m-2}$ in $\Sigma \backslash (Set(V) \cup Set(W))$. The string $W \centerdot s_0 s_1 \ldots s_{m-2} \centerdot V$ is therefore an $m$-voicing, and since its length is $3m-1$, by Lemma A.2 we have that $V \underset{\Sigma}{\Longrightarrow}^{2m-1} W$. □

**Remark A.1** *The proof of Theorem A.2 suggests an algorithm for constructing effectively a voice leading $\mathcal{V}$ which connects any two given voicings $V$ and $W$ of the same length $m$, over an alphabet $\Sigma$ of size greater than $m$. Observe, however, that the voice leading $\mathcal{V}$ so constructed is not, in general, the shortest possible. Indeed, the choice of the symbol $s$ in the construction of the voicing $Z_{i+1}$ from the voicing $Z_i$, may influence the length of the resulting voice leading from $V$ to $W$.*

**Corollary A.1** *Let $V$ and $W$ be voicings of length $m$ over an alphabet $\Sigma$. If $|\Sigma| > m$, then $V$ is connectable to $W$ with respect to $\Sigma$. Otherwise, i.e., if $|\Sigma| = m$, then $V$ is connectable to $W$ with respect to $\Sigma$ if and only if $W$ is a substring of $V \centerdot V$.*

**Proof.** Immediate from Theorems A.1 and A.2. □

**Corollary A.2** *Let $V$ and $W$ be voicings of length $m$ over an alphabet $\Sigma$ of size at least $m+2$. Then there is a connected voice leading $\langle V_0, V_1, \ldots, V_n \rangle$, which connects $V$ to $W$ with respect to $\Sigma$, such that $Set(V_i) \neq Set(V_{i+1})$, for $i = 0, 1, \ldots, n-1$.*

**Proof.** Since $|Set(V)| = |Set(W)| < |\Sigma|$, there exist $a, b \in \Sigma$ such that $a \notin Set(V)$ and $b \notin Set(W)$. Let us put $V' = V \centerdot a$ and $W' = W \centerdot b$. Then $V'$ and $W'$ are voicings over $\Sigma$ of length $m+1$, and since $|\Sigma| > m+1$, by Corollary A.1 there exists a connected voice leading $\mathcal{U} = \langle U_0, U_1, \ldots, U_n \rangle$ such that $U_0 = V'$, $U_n = W'$ and $Set(U_i) \subseteq \Sigma$, for $i = 0, 1, \ldots, n$.

Next, we define a voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$, by putting $V_i = U[0\,..\,m-1]$, for $i = 0, 1, \ldots, n$. Then, since $V_0 = V$, $V_n = W$ and $V_{i+1} = U_{i+1}[0] \centerdot V_i[0\,..\,m-2]$, for $i = 0, 1, \ldots, n-1$, it follows that $\mathcal{V}$ is a voice leading which connects $V$ to $W$ with respect to the alphabet $\Sigma$. It only remains to show that $Set(V_i) \neq Set(V_{i+1})$, for $i = 0, 1, \ldots, n-1$, which we do as follows. By way of contradiction, let us assume that $Set(V_j) = Set(V_{j+1})$, for some $j \in \{0, 1, \ldots, n-1\}$. Then we would have $V_j[m-1] = V_{j+1}[0]$, as $V_j \underset{\Sigma}{\Longrightarrow} V_{j+1}$. But $V_j[m-1] = U_j[m-1] = U_{j+1}[m]$, as $U_j \underset{\Sigma}{\Longrightarrow} U_{j+1}$, and $V_{j+1}[0] = U_{j+1}[0]$. Therefore, $U_{j+1}[m] = U_{j+1}[0]$, which is in contradiction with the fact that $U_{j+1}$ is a voicing, completing the proof of the corollary. □

**Theorem A.3** *Let $\mathcal{C} = \langle C_0, C_1, \ldots, C_{m-1} \rangle$ be a connected chord progression such that $C_i \neq C_{i+1}$, for $i = 0, 1, \ldots, m-2$ and where $m = |C_0|$, and let*

*$\langle V_0, V_1, \ldots, V_{m-1}\rangle$ be a connected voice leading of $\mathcal{C}$. Then*

$$\bigcap_{i=0}^{k} C_i = Set(V_0[0\,..\,m-k-1])\,,$$

*for $k = 0, 1, \ldots, m-1$.*

**Proof.** We begin by showing that

$$V_0[0\,..\,m-k-1] = V_i[i\,..\,i+m-k-1]\,, \quad \text{for } 0 \le i \le k\,, \tag{A.1}$$

by induction on $k = 0, 1, \ldots, m-1$.

For $k = 0$, (A.1) simply reduces to $V_0 = V_0$, which is trivially true.

For the inductive step, let us assume that (A.1) holds for some $k$ such that $0 \le k \le m-2$. In order to prove that (A.1) holds also for $k+1$, we just need to verify that $V_0[0\,..\,m-k-2] = V_{k+1}[k+1\,..\,m-1]$.

Since $V_k \underset{\Sigma}{\Longrightarrow} V_{k+1}$, we have

$$V_k[k\,..\,m-2] = V_{k+1}[k+1\,..\,m-1]\,. \tag{A.2}$$

In addition, by inductive hypothesis, we have $V_0[0\,..\,m-k-1] = V_k[k\,..\,m-1]$, which together with (A.2) implies $V_0[0\,..\,m-k-2] = V_{k+1}[k+1\,..\,m-1]$, completing the proof of (A.1).

From (A.1), it follows that

$$Set(V_0[0\,..\,m-k-1]) \;\subseteq\; \bigcap_{i=0}^{k} Set(V_i) \;=\; \bigcap_{i=0}^{k} C_i\,,$$

for each $k = 0, 1, \ldots, m-1$. Hence, to complete the proof of the theorem we only have to prove the converse inclusions

$$\bigcap_{i=0}^{k} C_i \;\subseteq\; Set(V_0[0\,..\,m-k-1])\,, \tag{A.3}$$

for $k = 0, 1, \ldots, m-1$. Thus, let $0 \le k \le m-1$. Since $\bigcap_{i=0}^{k} C_i \subseteq C_0 = Set(V_0)$, to show (A.3) it is enough to prove that

$$Set(V_0[m-k\,..\,m-1]) \cap \bigcap_{i=0}^{k} C_i \;=\; \emptyset\,, \tag{A.4}$$

which we do as follows. Let $m-k \le h \le m-1$. Then $0 < m-h \le k$, so that $\bigcap_{i=0}^{k} C_i \subseteq C_{m-h-1} \cap C_{m-h} = Set(V_{m-h-1}) \cap Set(V_{m-h})$. Since $V_{m-h-1} \underset{\Sigma}{\Longrightarrow} V_{m-h}$, we have $Set(V_{m-h-1}[0\,..\,m-2]) = Set(V_{m-h}[1\,..\,m-1])$. But $C_{m-h-1} \neq C_{m-h}$, hence $V_0[h] = V_{m-h-1}[m-1] \notin C_{m-h}$, which implies $V_0[h] \notin \bigcap_{i=0}^{k} C_i$. Therefore we have (A.4), which in turn implies (A.3), concluding the proof of the theorem. □

## Acknowledgments

The authors are grateful to three anonymous referees for their helpful comments.

## References

1. E. Aldwell and C. Schachter, *Harmony and Voice Leading*, Harcourt Brace, Orlando, FL, 2002.
2. R. Baeza-Yates and G. H. Gonnet, A new approach to text searching. *Communications of the ACM*, 35(10):74–82, 1992.
3. E. Cambouropoulos, M. Crochemore, C. S. Iliopoulos, L. Mouchard, and Y. J. Pinzon, Algorithms for computing approximate repetitions in musical sequences. In R. Raman and J. Simpson, eds., *Proc. of the 10th Australasian Workshop on Combinatorial Algorithms*, Perth, WA, Australia, pp. 129–144, 1999.
4. R. Clifford, R. Groult, C. Iliopoulos, R. Byrd. Music retrieval algorithms for the lead sheet problem. *Journal of New Music Research*, 34(4):311–317, 2005.
5. T. Crawford, C. Iliopoulos, and R. Raman, String matching techniques for musical similarity and melodic recognition. *Computing in Musicology*, 11:71–100, 1998.
6. M. Crochemore, C. S. Iliopoulos, T. Lecroq, and Y. J. Pinzon, Approximate string matching in musical sequences. In M. Balík and M. Šimánek, eds., *Proc. of the Prague Stringology Conference '01*, Prague, Czech Republic, Annual Report DC–2001–06, pp. 26–36, 2001.
7. M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and G. Navarro, A bit-parallel suffix automaton approach for $(\delta, \gamma)$-matching in music retrieval. In E. S. D. Moura and A. L. Oliveira, eds., *Proc. of the 10th International Symposium on String Processing and Information Retrieval (SPIRE'03)*, vol. 2857 of Lect. Notes in Comp. Sci., Springer-Verlag, pp. 211–223, 2003.
8. A. Forte, *The Structure of Atonal Music.* Yale University Press, New Heaven, 1973.
9. D. Lewin, *Generalized Musical Intervals and Transformations.* Yale University Press, New Heaven, 1987.
10. G. Mazzola, *The Topos of Music: Geometric Logic of Concepts, Theory, and Performance.* Birkhäuser, Basel, 2002.
11. C. MIDI Manufacturers Association, Los Angeles, *The Complete Detailed MIDI 1.0 Specification*, 1996.