On Some Combinatorial Problems Concerning the Harmonic Structure of Musical Chord Sequences

Domenico Cantone, Salvatore Cristofaro, and Simone Faro

Università di Catania, Dipartimento di Matematica e Informatica
Viale Andrea Doria 6, I-95125 Catania, Italy
{cantone | cristofaro | faro}@dmi.unict.it

Abstract. We present some combinatorial problems which arise in the fields of music representation and music processing, especially in contexts such as the analysis of the harmonic structure of chord sequences. We concern ourselves with those chord sequences which exhibit a certain kind of *regular harmonic structure*, and discuss some problems related to them. We provide also algorithms to solve some of these problems.

Keywords: music processing, harmonic structure analysis, chord sequences.

1 Introduction

Musical chord sequences, or chord progressions, possess a combinatorial structure very rich and complex, which require 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 [7, 8].

Musical notes can be coded in various ways. A typical example is provided by the standard MIDI representation, where notes are coded by integers [9]. 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 the chord. 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 especially 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 similarity [2, 4, 5, 3].

However, in many cases, especially when 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 [6]. In such a case, only 12 symbols are needed to represent 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 Figure 1. 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

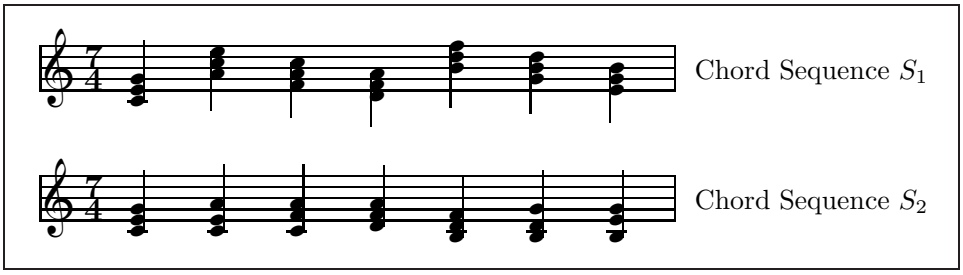

Figure 1. Two chord sequences related by octave equivalence. Chord sequence S_1 exhibits a regular harmonic structure.

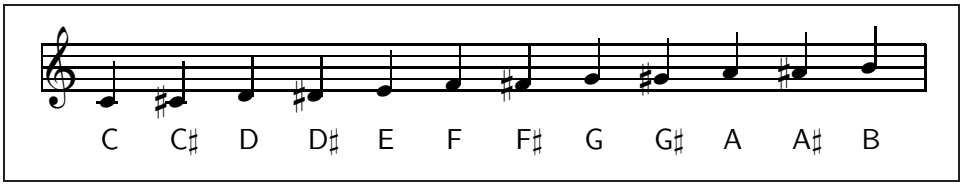

Figure 2. The traditional naming of notes with symbols $\mathsf{C}, \mathsf{C}\sharp, \mathsf{D}, \mathsf{D}\sharp, \mathsf{E}, \mathsf{F}, \mathsf{F}\sharp, \mathsf{G}, \mathsf{G}\sharp, \mathsf{A}, \mathsf{A}\sharp, \mathsf{B}$

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 chords as ordered sets of notes, we may discover some regularities in the structure of the sequence S_1. Indeed, if we order the notes of each chord from the lowest to the highest one, i.e., if we look at the voicings of the chords, we get a representation of each chord as a string of symbols, and the chord sequence can be conveniently represented by a matrix $\mathcal{M}$ whose columns correspond to such strings:

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

A simple inspection of the matrix $\mathcal{M}$ reveals the regular structure of the chord sequence. Simply look at the secondary diagonal elements of each square submatrix of $\mathcal{M}$. Also, observe that any two consecutive chords of the sequence share at least two notes, and any three consecutive chords share at least one note, so that the chords are connected in such a way that the "transition" from a chord to a next one 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 $\mathcal{M}$ a copy of itself, we get a matrix with the very same structure of $\mathcal{M}$. 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., when the chord sequence is heard for the first time, one expects that it will be played again. This is strictly related to the phenomenon of musical expectation, which plays

[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 [10]).

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 important 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 preliminary results concerning an ongoing investigation on various combinatorial questions about regular chord progressions and voicings.

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 some other related questions. Then, in Section 4 we draw our conclusions. Finally, an appendix containing some theoretical results concludes the paper.

2 Basic definitions and properties

Before entering into details, we need a bit of notations and terminology. Let Σ be a finite alphabet. 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 ε. The length of X is denoted by $|X|$. By $X[i]$ we denote the $(i+1)$-th symbol of X, for $0 \leq i < |X|$. Likewise, by $X[i\,..\,j]$ we denote the substring of X contained between the $(i+1)$-th symbol and $(j+1)$-th symbol of X, for $0 \leq i \leq j < |X|$.

For convenience, we do not distinguish between a symbol s and the one-character string "s". Thus, for any two strings X and Y and any symbol s, we write $X{\bullet}s{\bullet}Y$ for the string Z of length $|X| + |Y| + 1$ such that

- $Z[0\,..\,|X|-1] = X$,
- $Z[|X|] = s$, and
- $Z[|X|+1\,..\,|X|+|Y|] = Y$.

A CHORD over Σ is a nonempty set C of two or more symbols of Σ. The SIZE of a chord C, denoted by $size(C)$ or by $|C|$, is the number of symbols in C. A VOICING over Σ is a string V of symbols of Σ 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 $Set(V)$ of a voicing V is the collection of the symbols occurring in V. A voicing V is said to be a VOICING OF A CHORD C if $Set(V) = C$.[2] A CHORD PROGRESSION is a sequence $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ of chords with the same size.

A VOICE LEADING over Σ is a sequence $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ of voicings over Σ of the same length. A voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ is a VOICE LEADING OF A CHORD PROGRESSION $\mathcal{C} = \langle C_0, C_1, \ldots, C_m \rangle$, provided that $n = m$ and V_i is a voicing of the chord C_i, for $i = 0, 1, \ldots, n$.

Let V and W be voicings over the alphabet Σ. We say that V is (IMMEDIATELY) CONNECTED to W, and write $V \longrightarrow W$, if $W = s{\bullet}V[0\,..\,|V|-2]$, for some symbol $s \in \Sigma$. Plainly, when $V \longrightarrow W$, the voicings V and W must have the same length. A

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

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 $V_n \longrightarrow V_0$.[3]

A voicing V is CONNECTABLE to a voicing W with respect to an alphabet Σ, in symbols $V \Longrightarrow W$, if there is a connected voice leading $\mathcal{V} = \langle V_0, V_1, \ldots, V_n \rangle$ over Σ, with $n \geq 1$, 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 Σ).

The *connectivity relation* "$\Longrightarrow$" is an equivalence relation, as shown in the following lemma.

Lemma 1. *Let V, W, and Z be voicings over an alphabet Σ. Then*

(1) $V \Longrightarrow V$;
(2) if $V \longrightarrow W$ then $W \Longrightarrow V$;
(3) if $V \Longrightarrow W$ then $W \Longrightarrow V$;
(4) if $V \Longrightarrow W$ and $W \Longrightarrow Z$, then $V \Longrightarrow Z$.

Therefore the relation $\Longrightarrow$ 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 "$\Longrightarrow$" and (3) follows from (2) and (4).

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

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

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

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 $V[m-1] = W[0]$, 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

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

for $i = 0, 1, \ldots, m-2$.
On the other hand, if $V[m-1] \neq W[0]$, then W is connected to V by the voice leading $\langle V_0, V_1, \ldots, V_m \rangle$, where $V_0 = W$ and

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

for $i = 0, 1, \ldots, m-1$. □

A chord C is CONNECTED to a chord D, written $C \longrightarrow D$, if $V \longrightarrow W$, for some voicings V of C and W of D.[4] A chord progression $\mathcal{C}$ is CONNECTED (resp., CIRCULARLY CONNECTED) if it has a connected (resp., circularly connected) voice leading. A chord progression $\mathcal{C} = \langle \mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n \rangle$ is REGULAR if it is circularly connected and, in addition, $\mathcal{C}_i \neq \mathcal{C}_{(i+1) \bmod (n+1)}$, for $i = 0, 1, \ldots, n$. It can easily be verified that the chord progression S_1 in Figure 1 is regular.

We conclude the section with some examples.

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

[4] From the context it will always be clear whether the symbol "$\longrightarrow$" denotes the connectivity relation between voicings or between chords.

Example 2. 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 Σ. The voicings V_1, V_2, and V_3 have length 4, whereas V_4 has length 5. On the other hand, the strings

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

are not voicings over Σ. Indeed, $X[0] = X[3] = a$, $Y[1] = Y[2] = Y[4] = e$, and $Z[4] = f$ is not a symbol of Σ (however, Z is a voicing over the extended alphabet $\Sigma \cup \{f\}$). 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 Σ. □

Example 3. The following chord progression over the alphabet $\Sigma = \{a, b, c, d, e, f\}$

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

is circularly connected since it has the circularly connected voice leading

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

However, $\mathcal{C}$ is not regular since the first and the last chord coincides.
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 4. The chord progression

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

over the alphabet $\Sigma = \{a, b, c, d, e, f\}$ 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 them.
We begin by addressing the following question.

Problem 5. Given a chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ over an alphabet Σ, 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 5 as follows. Let m be the length of V. 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. 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 $\mathcal{C}_i = \langle C_0, C_1, \ldots, C_i \rangle$, with $0 \leq i < n$. Then, we form the set $S_i =_{\text{Def}} Set(V_i[0\,..\,m-2])$ 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

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

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, 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 6. 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 8 of ALGO1 we add an instruction **OUTPUT**(X), we get as by-product the sequence $V_1, V_2, \ldots, V_k$, 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 linear arrays, we get $f(m) = \mathcal{O}(m^2)$, yielding an overall running time of $\mathcal{O}(nm^2)$.

However, if the alphabet Σ is sufficiently small to fit into a computer word, we can conveniently use the bit-parallelism technique [1] to reduce the running time to $\mathcal{O}(n+m)$. Indeed, let us assume that $\sigma = |\Sigma| \leq \omega$, where ω is the number of bits in a computer word. Then, any subset of Σ can be represented by a bit mask of length σ, which fits into a computer word. By using such a representation, the set operations of union, intersection, and complement, as well as the set containment 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 Σ, 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 σ), where

$$b_j = \begin{cases} \mathsf{1} & \text{if } j = \sigma - 1 - i \\ \mathsf{0} & \text{otherwise}\,, \end{cases}$$

ALGO1($\boldsymbol{\mathcal{C}}$, V, W)
–It is assumed that $\boldsymbol{\mathcal{C}}$ *is a chord progression,* $\mathbf{V}$ *is a voicing of*
–the first chord of $\boldsymbol{\mathcal{C}}$*, and* $\mathbf{W}$ *is a voicing of the last chord of* $\boldsymbol{\mathcal{C}}$*.*

```
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.          X := z•X[0 .. m − 2]
9.          S := (S \ {X[m − 2]}) ∪ {z}
10.     else
11.         return false
12. if X ≠ W then
13.     return false
14. return true
```

Figure 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

for $j = 0, 1, \ldots, \sigma - 1$;[5]
– a nonempty subset $A = \{s_{i_0}, s_{i_1}, \ldots, s_{i_k}\}$ of Σ 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 Σ is represented by the bit mask $\mathsf{0}^\sigma$, i.e., the string consisting of σ 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 $i = 0, 1, \ldots, n$;
– a voicing V is represented as an array $\mathbf{V}[0 \,..\, m - 1]$ of m bit masks, where $\mathbf{V}[i] = \mathsf{B}(V[i])$, 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

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

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 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 $S_i = \mathit{Set}(V_i[0 \,..\, m-2])$ can be conveniently maintained in a bit mask 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:

[5] 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.

– 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, Z contains the bit mask $\mathsf{B}(z)$);
– retrieve the first bit mask D in $\mathcal{Q}$ by executing the operation *dequeue*$(\mathcal{Q})$;
– enqueue 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, S will be set to $(\mathsf{S} \wedge \sim \mathsf{D}) \vee \mathsf{Z}$, so as to represent the set S_{i+1}.

Remark 7. 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 the array $\mathbf{Q}$ 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 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 a $\mathcal{O}(n+m)$-running time.

Remark 8. Analogously to the observation in Remark 6 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**
  $X := X_{\bullet}$*decode*$(\mathbf{Q}[(h+m-2-j) \mod (m-1)])$
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 $s_i \in \Sigma$. The function *decode* admits a simple constant-time implementation. To begin with, let us represent the alphabet Σ as an array $\Sigma[0\,..\,\sigma-1]$, so that $\Sigma[i] = s_i$, for $i = 0, 1, \ldots, \sigma-1$. Then, if $x = \mathsf{B}(s_i)$, we have immediately $s_i = \Sigma[\lceil \log_2 x \rceil]$, for $i = 0, 1, \ldots, \sigma-1$. Therefore, we can just put *decode*$(x) =_{\text{Def}} \Sigma[\lceil \log_2 x \rceil]$. If we further assume that the symbols of the alphabet Σ are the first σ nonnegative integers, i.e. $s_i = i$, for $0 \leq i \leq \sigma-1$, the decoding function becomes more simply *decode*$(x) =_{\text{Def}} \lceil \log_2 x \rceil$. Additionally, under such an assumption, we have also that $\mathsf{B}(s) = (1 \ll s)$, where $\ll$ denotes the bitwise operation of left-shifting, implying that also the coding of a singleton $\{s\}$ as the bit mask $\mathsf{B}(s)$ can be performed in constant time, for any symbol $s \in \Sigma$.

Notice, however, that if we modify the ALGO2 algorithm 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. □

A second question we address is the following.

Problem 9. Given a chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$, check whether $\mathcal{C}$ is regular. □

```
ALGO2(𝒞, V, W)
 1.  m := length(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
```

Figure 4. An optimized variant with bit-parallelism of the ALGO1 algorithm

To solve this problem, a natural but inefficient solution could be the following one. Let m be the size of the chords $C_0, C_1, \ldots, C_n$. We start by checking that $C_i \neq C_{i+1}$, 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 ALGO2 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, $|C_n \setminus C_0| \neq 1$, then, certainly, $\mathcal{C}$ would not be regular). Since there are $m!$ possible voicings of C_0, such an approach has a $\mathcal{O}(m!(n+m))$-time complexity.

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 we point out that 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 $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ is regular, and we set

$$X_k = \bigcap_{i=0}^{m-1-k} C_i\,, \quad \text{for } k = 0, 1, \ldots, m-1,^{6}$$

where $X_{m-1} = C_0$, then each of the $m-1$ sets $X_0, X_1, \ldots, X_{m-1}$, except the last one, must be a nonempty proper subset of the set which immediately follows it; i.e., there

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

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

Figure 5. The algorithm ALGO3 checks if a given chord progression $\mathcal{C}$ is regular. The size of the chords in $\mathcal{C}$ is m.

must be $m-1$ 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 indeed immediate consequences of Theorem 19 in Appendix A. In addition, 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 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 $\mathcal{O}(nm^2)$ (at least in case in which sets are represented as linear arrays.)

However, by using the bit-parallelism technique, thus representing as usual sets as bit masks, and voicings as arrays of bit masks, we can obtain an efficient variant of the ALGO3 algorithm, called 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

```
ALGO4(C, m)
 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)
17.  else
18.      return false
```

Figure 6. An optimized variant with bit-parallelism of the ALGO3 algorithm

the ALGO4 algorithm has an $\mathcal{O}(n+m)$-running time and requires only $\mathcal{O}(m)$-extra 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 10. 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

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

We observe 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 11. Any given chord progression $\mathcal{C} = \langle C_0, C_1, \ldots, C_n \rangle$ can always be embedded into 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 12. Find a minimal connected chord progression 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 Σ.

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 Theorem 16 in Appendix A is the following connectability test for voicings:

> Given any two voicings V and W of the same length over an alphabet Σ, if $Set(V) \neq \Sigma$ or $Set(W) \neq \Sigma$, then V can be connected to W with respect to Σ, 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 13. Given two voicings V and W of the same length over an alphabet Σ, determine a shortest voice leading connecting V to W. □

Notice that Open Problems 12 and 13 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, eventually 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 bars.

Another interesting question related to the connectivity relation between voicings, with applications in music composition, is the following. 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 Theorem 16 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 theorem does not say anything on the fact that a voice leading $\mathcal{V}$ which connects V to W have 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 18 in Appendix A, 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 given a chord progression $\mathcal{C}$, we can always extend $\mathcal{C}$ to a regular chord progression by adding at most two new symbols. An interesting question is then the following:

Open Problem 14. Given a chord progression $\mathcal{C} = \langle C_0, C_1, \ldots C_n \rangle$ and a fixed bound $k > n$, determine the minimum number of new symbols we need to add 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 some others have been raised but left unsolved (at least in the sense that no efficient algorithm has been provided). We plan to address in more details such problems in the future and to provide also efficient algorithmic solutions to them.

References

[1] R. Baeza-Yates and G. H. Gonnet: *A new approach to text searching.* Communications of the ACM, 35(10) 1992, pp. 74–82.

[2] E. Cambouropoulos, M. Crochemore, C. S. Iliopoulos, L. Mouchard, and Y. J. Pinzon: *Algorithms for computing approximate repetitions in musical sequences*, in Proc. of the 10th Australasian Workshop on Combinatorial Algorithms, R. Raman and J. Simpson, eds., Perth, WA, Australia, 1999, pp. 129–144.

[3] T. Crawford, C. Iliopoulos, and R. Raman: *String matching techniques for musical similarity and melodic recognition.* Computing in Musicology, 11 1998, pp. 71–100.

[4] M. Crochemore, C. S. Iliopoulos, T. Lecroq, and Y. J. Pinzon: *Approximate string matching in musical sequences*, in Proc. of the Prague Stringology Conference '01, M. Balík and M. Šimánek, eds., Prague, Czech Republic, Annual Report DC–2001–06, 2001, pp. 26–36.

[5] M. Crochemore, C. S. Iliopoulos, Y. J. Pinzon, and G. Navarro: *A bit-parallel suffix automaton approach for (δ, γ)-matching in music retrieval*, in Proc. of the 10th International Symposium on String Processing and Information Retrieval (SPIRE'03), E. S. D. Moura and A. L. Oliveira, eds., vol. 2857 of Lect. Notes in Comp. Sci., Springer-Verlag, 2003, pp. 211–223.

[6] A. Forte: *The Structure of Atonal Music*, Yale University Press, New Heaven, 1973.

[7] D. Lewin: *Generalized Musical Intervals and Transformations*, Yale University Press, New Heaven, 1987.

[8] G. Mazzola: *The Topos of Music*, Basel: Birkhäuser Verlag, 2003.

[9] C. MIDI Manufacturers Association, Los Angeles: *The Complete Detailed MIDI 1.0 Specification*, 1996.

[10] D. Tymoczko: *Scale Theory, Serial Theory, and Voice Leading*, Princeton University, 2006.

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.

We need first some further definitions. Let Σ be an alphabet, and let X and Y be strings over Σ. We say that X is a prefix of Y, and write $X \sqsubset Y$, if $Y = X.Z$ for some string Z. If s is a symbol of Σ and $0 \leq k < |X|$, we denote by $X(k:s)$ the string obtained from X by replacing its $(k+1)$-th symbol by s, so that the following equality holds

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

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

Lemma 15. *Let V be a voicing of length m over the alphabet Σ, and let $s \in \Sigma \setminus Set(V)$. Then, for each $k \geq 0$, $V(k:s)$ is a voicing and $V \Longrightarrow V(k:s)$.*

Proof. Plainly, $V(k:s)$ is a voicing, since $s \notin Set(V)$. To show that $V \Longrightarrow V(k:s)$, we set $S = V(k:s).V$ and $V_i = S[|S| - m - i\,..\,|S| - 1 - i]$, for $i = 0, 1, \ldots, |S| - m$. Then it is easy to verify that $\langle V_0, V_1, \ldots, V_{|S|-m}\rangle$ is a voice leading connecting V to $V(k:s)$. □

Theorem 16. *Any two voicings of the same length m over an alphabet Σ of size larger than m are connectable in Σ.*

Proof. Let V and W be two voicings of length m over an alphabet Σ of size $\sigma > m$. We will show that $V \Longrightarrow W$, by proving by induction that for each $\ell = 0, 1, \ldots, m$ there is a voicing Z of length m such that $V \Longrightarrow Z$ and $W[0\,..\,\ell-1] \sqsubset Z$.

For $\ell = 0$, it is enough to take $Z = V$, since $V \Longrightarrow V$ (by (1) of Lemma 1), and $W[0\,..\,\ell-1] = \varepsilon \sqsubset V$.

For the inductive step, let $1 \leq \ell \leq m-1$, and let us assume that there is a voicing U (of length m) such that $V \Longrightarrow U$ and $W[0\,..\,\ell-1] \sqsubset U$. Since $|U| < \sigma$, the set $\Sigma \setminus Set(U)$ is nonempty, so we can pick an $s \in \Sigma \setminus Set(U)$. Let

$$\begin{aligned} k &= \mathbf{min}(\{0 \leq i < m : U[i] = W[\ell]\} \cup \{m\})\,, \\ \widehat{U} &= U(k:s)\,, \\ Z &= \widehat{U}(\ell : W[\ell])\,. \end{aligned}$$

Then, by Lemma 15, the strings $\widehat{U}$ and Z are voicings of length m and, additionally, $U \Longrightarrow \widehat{U}$ and $\widehat{U} \Longrightarrow Z$ hold. By the transitivity of the connectivity relation "$\Longrightarrow$" (cf. Lemma 1) we have $V \Longrightarrow Z$. To conclude the proof of the inductive step, we have only to observe that $W[0\,..\,\ell] \sqsubset Z$ plainly holds. □

Remark 17. The proof of Theorem 16 suggests an algorithm to effectively construct a voice leading $\mathcal{V}$ which connects any two given voicings V and W of the same length m, over an alphabet Σ of size larger than m. Observe, however, that the voice leading $\mathcal{V}$ so constructed is not, in general, the smallest possible. □

Corollary 18. *Let V and W be voicings of length m over an alphabet Σ 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 Σ,such that $Set(V_i) \neq Set(V_{i+1})$, for $i = 0, 1, \ldots, n-1$.*

Proof. As usual, let $\sigma = |\Sigma|$. 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 $V' = V_{\bullet}a$ and $W' = W_{\bullet}b$. Then V' and W' are voicings of length $m+1$, and since $\sigma > m+1$, by Theorem 16, there exists a connected voice leading $\mathcal{U} = \langle U_0, U_1, \ldots, U_n \rangle$ such that $U_0 = V'$ and $U_n = W'$.

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 $i = 0, 1, \ldots, n$. To begin with, notice that $V_0 = V$ and $V_n = W$. Moreover, since $V_{i+1} = U_{i+1}[0]_{\bullet}V_i[0\,..\,m-2]$, 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 Σ. 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 $V_j \longrightarrow V_{j+1}$. But $V_j[m-1] = U_j[m-1] = U_{j+1}[m]$, as $U_j \longrightarrow U_{j+1}$, and $V_{j+1}[0] = U_{j+1}[0]$. Therefore, $U_{j+1}[m] = U_{j+1}[0]$, which is a contradiction, since U_{j+1} is a voicing. □

Theorem 19. *Let $\mathcal{C} = \langle C_0, C_1, \ldots, C_{m-1} \rangle$ be a connected chord progression, with $m = size(C_0) \geq 2$, such that $C_i \neq C_{i+1}$, for $i = 0, 1, \ldots, m-2$, 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

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

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

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

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

Since $V_k \longrightarrow V_{k+1}$, we have $V_k[k\,..\,m-2] = V_{k+1}[k+1\,..\,m-1]$. In addition, by the inductive hypothesis, we have $V_0[0\,..\,m-k-1] = V_k[k\,..\,m-1]$, which plainly implies $V_0[0\,..\,m-k-2] = V_k[k\,..\,m-2]$, by dropping the last symbol in both voicings. Therefore we have $V_0[0\,..\,m-k-2] = V_{k+1}[k+1\,..\,m-1]$, completing the proof of (1).

From (1), it follows that

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

so that we are only left with proving the converse inclusion

$$\bigcap_{i=0}^{k} C_i \;\subseteq\; Set(V_0[0\,..\,m-k-1])\,. \tag{2}$$

Since $\bigcap_{i=0}^{k} C_i \subseteq C_0 = Set(V_0)$, to show (2) it is enough to prove that

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

which we do as follows. Let $m - k \leq h \leq m - 1$. Then $0 < m - h \leq k$, 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 $V_{m-h-1} \longrightarrow V_{m-h}$, 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 (3) holds, which in turn implies (2), concluding the proof of the theorem. □