CROWDED STAVES — RULES FOR SEMANTICS AND STYLE OF CONVENTIONAL MULTIPLE-VOICES-PER-STAFF MUSICAL NOTATION

Markus Lepper
semantics GmbH Berlin
post@markuslepper.eu

Michael Oehler
Osnabrück University
michael.oehler@uni-osnabrueck.de

Hartmuth Kinzler
Osnabrück University
hkinzler@uni-osnabrueck.de

Baltasar Trancón
semantics GmbH Berlin
baltasar@trancon.de

ABSTRACT

In many variants of Common Western Notation (CWN) more than two voices can be notated together in one staff. Reading this kind of multi-voice notation implies complicated parsing decisions, taken by the trained musician's brain in most cases non-knowingly. This article makes them explicit, supposing a theoretical maximum of information retrieval and formal consistency.

1. INTRODUCTION

When talking about musical notation, very different viewpoints can be sensible. One of these is "notation as a precise encoding", which means that the graphic components of the notated piece of music can be translated by a mathematically well-defined process into the elements of a semantic model, and vice versa. While this approach is obviously not *fully* appropriate to most empirical situations, it can never be *totally absent*, because it represents an abstract and idealistic "basic idea of notation" as a perfect encoding system.

The method of *mathematical re-modeling* as applied in our SemPart project creates compound mathematical models, intended to mimic by a precise mathematical operation the empirically and historically given processes of encoding and decoding music notation. Thus these models *define the semantics* of notation, and the grid spanned by their variants offers precise nomenclature for *semantic and stylistic differences*.

We concentrate on the *Common Western Notation (CWN)*, which has evolved since the 17th century, has been successfully adapted since then in many variants to new developments in composition, practice and theory of music, and is nowadays still in vivid development.

The created models basically consist of (a) a comparatively simple mathematical structure representing the graphical appearance (= "external representation" = "syntax sphere"), namely the graphical components' classes and the syntactical rules for their combination, (b) a collection of arbitrarily complex mathematical objects (values, vectors, sets, relations, functions, constraints) representing the intended musical parameters (= "semantic sphere"), (c) a mathematical function mapping syntax to semantics (= "parsing function") and/or (d) a function mapping semantics to syntax (= "writing function").

For all the technical details and philosophical implications of this approach see [1], where we applied this method to the notation of musical dynamics. From these principles are relevant for this article:

(A) In no case there is "one single true semantic model". Instead, we always found a multitude of similar models (or to say: one model which can be widely parameterized) which represent different usages from contemporary or historical practice. These differences range from slightly different flavors to opposite and incompatible approaches. The intention of the SemPart project is to present the different models to the domain specialists of historic or systematic musicology, music pedagogy, music performance practice, editing, printing, computer programming, etc., to let them select according to their needs, and so offer to all of them a nomenclature for talking about common features and differences in a more precise way.

(B) While the historic evolution and the social processes are the main forces which have defined the syntax and semantics of music notation, this is completely excluded from the modeling. The SemPart method is an *ahistoric* one, which refers to examples from history only for inspiration and for an a posteriori verification by application.

(C) There are important *empirical* results about the reading process of music notation, e.g. [2], [3], [4] and [5]. The meta-study [6] evaluates 15 studies on eye movement "to support the crafting of more well-founded research hypotheses and the more systematic design of experiments". Applying methods from psychology and neurology, all these are completely complementary to our approach, which extracts by abstract methods an idealistic information contents which can only serve as the theoretical maximum of retrievable information. One could say we define a

proposal for the "Abstract Internal Representation", as it is graphically symbolized by the question mark in Figure 1 of [4], and which is explicitly *not* part of their work. E.g., we define a parsing algorithm only to define the abstract information contents, not as a concrete model of the empirical process of reading.

(D) The re-modeling approach presented in this paper also differs fundamentally from the well-known task of *voice separation*, see [7], [8] (both also for surveys), [9], [10] and many others. Those papers cover a concrete data-processing task, required for automated music notation, analysis and retrieval, and the different solutions take into account very different aspects of perception, precision, historic rules, technical representation, etc. In contrast, the SemPart project addresses a more basic level and simply tries to concretize (the different variants of) the mental and cultural *definitions* which are applied when notation is read or written by human actors. This concretization is also given in form of an *operational semantics*, i.e. an algorithm which must be executed to yield results, but this algorithm is only the means for the analyses, not their purpose.

2. PROCESSING PIPELINE FOR PARSING MULTI-VOICE STAVES

2.1 Outline of the problem

This paper applies the fundamental modeling technique of the SemPart programme to the task of recognizing more than two voices notated in the same staff. The idiosyncrasies of the historical evolution of CWN made this parsing process much more complicated and thus mathematically interesting than it appears at the first glance: In daily practice it is nearly always executed by any musician without further difficulties, but also without realizing the complicated intellectual processes required. The following algorithms do nothing more than making these processes visible.

For the following discussion, a *voice* is a sequence of *adjacent notated events* in time. Each event can be a *sounding event (=sound)* or a *pause* and has one particular *duration*, measured in some musical time system. [1]

Even the oldest known example of Western Notation writes two voices in one staff (*Musica Enchiriadis*, using text syllables for note heads, see page 57r of [11].) Much later, in *Ars Nova*, stem direction was used for voice indication [12].

Later more than two note heads simultaneously appeared in course of notating instruments which are capable of (sporadic) chord playing. This case is *not* covered in this article.

Here we look only at more than two parallel, independent and contiguous voices sharing one staff. In practice, these appear (a) in orchestra scores, when more than two "voices" in the sense of "note text for one melody instrument" are comprehended in one staff, and (b) in notes for keyboard instruments, where "voices" in the sense of abstract "voice leading" can spontaneously come into existence and vanish again. (Case (b) comprises also some string instruments like guitar, harp, hackbrett etc.)

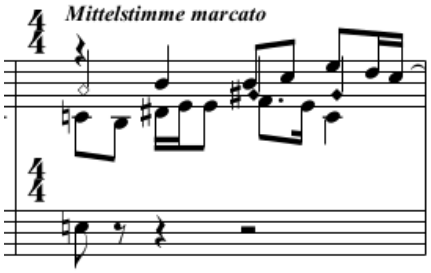


Figure 1. Different note heads for filtering voices

Esp. in piano literature, starting with the pre-classics like C.Ph.E. Bach and D. Scarlatti, complicated combinations of these "keyboard voices" have been realized. Here we do not discuss what these voices are intended to represent, whether they are meant to be heard by the listener, how they should be played by the musician or whether they live only on a pure conceptual level, etc.

Here we are interested only in the mere graphical notation and how this is decoded or "parsed" into a data set which identifies the *relation from events to voices(= to voice names)*. For this we feed one single staff with its contents, called *original staff*, through a pipeline of processing steps.

2.2 Vertical filter

The first step separates the information from different layers in the original staff if they are immediately distinguished by a graphical attribute. In course of the more and more complicated voice textures, especially in piano literature, some authors used such attributes for a first clarification of the voice leading. The most common means are to use a *smaller size of note heads* for marking one particular voice. Also different *note head forms* are possible, like cross or diamond. Figure 1 shows an example from a 20th century piano sonata.

In modern scores, for instance if presented on a digital display, also *different colors* are easily possible.

If such a graphical attribution is present, this step separates the contents of the original staff into separate homogeneous staves according to these attributes, called *filtered staves*. In case that horizontal gaps do result, these have to be filled by additional pauses. The next steps are applied to each of these filtered staves separately in basically the same way as to a homogeneous original staff.

[1] For this article we assume without loss of generality this time system to be metrical duration, with the conventional rational numbers as duration values, and with the conventional graphical duration symbols. Nevertheless, the algorithms presented in the following would work also on arbitrarily different bases.

2.3 Horizontal segmentation

In an orchestra score, when several melody instruments share a staff, the identity of the instruments is established by some *explicit voice order declaration*. It lists the names of the instruments in one particular sequential order, which shall be mapped by the reader to the vertical order of the simultaneously executed events.

In many cases this explicit declaration is not constant over the whole duration of the score but changes by *explicit (voice order) re-declaration*. For our algorithm, a staff can be cut at such a point and the resulting segments treated separately, if this declaration mentions all existing voices. Otherwise its evaluation is much more complicated and part of the parsing process, as explained below.

In keyboard music the voices are (normally) not named or declared explicitly and can come into existence and vanish again spontaneously. Here we must cut between segments with different numbers of voices and treat them separately. [2] The naming of the voices (which is needed by the algorithm) is thus implicit and synthetic, like "`voice1`", "`voice2`", etc.

Contrarily, in keyboard fugues like "Das Wohltemperierte Klavier" the number of overall voices is given by the title like "Fuga à 5", which can be expanded to a canonical initial voice declaration "`v1, v2, v3, v4, v5`". [3]

2.4 Input data of the algorithms

Finally we apply parsing to each staff data which results from vertical filtering and horizontal segmentation. The task is to map each note head to a voice name, given the sequence of note heads and an initial voice order declaration.

We model the notation text as a sequence of sets of note heads. The type K in module *perLineas* in Table 1 [4] represents one note head as a tuple of its duration (as a rational number), pitch (natural number for the vertical position in the staff), "fine x" (the horizontal order of the note heads which belong to the same "chord" / time point), the direction of the stem, a flag whether it is a pause, and optionally the start coordinate of an incoming voice leading line. Such a coordinate from Ptx is a tuple of score position (as defined below), vertical position (=pitch) and fine x value and thus identifies a note head unambiguously.

The complete input data to the parsing algorithm is of type N. It includes a sequence of sets of note heads from $\mathbb{P}K$. All note heads in such a set are played synchronously. The sets follow with non-zero delay in temporal order; the index into this sequence is called *score position*, is taken from $\mathbb{N}_1$ and corresponds to the sequence of passing musical time points. [5]

At each combination of score position and voice, there is exactly one note head (not tied to its predecessor) or pause symbol, iff the preceding event in this voice ends at the time point represented by this position. These voices are called *(currently) active (at this score position)*; the other voices are *overlapping*. Thus the main task is to create at any given score position a bijective map between its note heads and its active voices.

Beside these sets, each score position can carry two lists of voice names from V for voice re-declaration, see section 4.1 below. So N represents the above-mentioned *syntax sphere* of the translation model. The *semantic sphere* is represented by the data type R which represents the result of the parsing algorithm, see section 4.3 below.

Since traditionally the "first oboe" sounds higher than the "second", the declaration sequence of voice names from left to right is mapped from top to bottom to the note heads, which corresponds to the physical pitch of the sounding events. This results in the *current (physical) voice order (=CVO)*. The CVO is a status data which reigns the assignment of note heads to voice names. It can be altered by subsequent *explicit re-declarations*, but also by voice leading lines and by other necessities resulting from the parsing process. Therefore it must be passed through all functions of our parsing algorithm as an explicitly modeled datum.

3. ISO-RHYTHMIC VOICES

First we analyze the situation in iso-rhythmic settings, i.e. without overlapping voices. The results of voice crossing analysis can easily be transferred to the more complex case of hetero-rhythmic voices, but doing so for pause sharing is left to future work.

[2] A change in the visible number of voices can also result from pause sharing and should be treated accordingly, see section 3.1.

[3] In practice, two different and disjoint declarations for both staves of the keyboard staff must be generated. This problem is not treated in this article, – all algorithms here treat one single staff only. Furthermore, they may appear more voices than mentioned by the title, typically in a coda section.

[4] The mathematical notation is based on the well-proven Z notation [13], with only few extensions for brevity. See the appendix for details.

[5] In the Z formalism, which is the basis for our modeling, indices of sequences start with the number One(1).[13] Thus the encoding of the notes in Figure 2 is a sequence of sets of tuples. starting with
$\langle\{(1/4, e'', 1, \bot, \texttt{true}, \bot), (1/4, e', 1, \bot, \texttt{true}, \bot)\},$
$\{(1/4, e'', 1, \uparrow, \texttt{false}, \bot), (1/4, e', 1, \bot, \texttt{true}, \bot)\},$
$\{(1/4, e'', 1, \uparrow, \texttt{false}, \bot), (1/4, e', 1, \downarrow, \texttt{false}, \bot)\},$
$\ldots\rangle$
and the top line of Figure 2 holds the data
$\langle\{(3/8, g'', 1, \uparrow, \texttt{false}, \bot), (3/4, e'', 2, \uparrow, \texttt{false}, \bot),$
$(3/4, c'', 2, \uparrow, \texttt{false}, \bot), (7/8, c', 1, \downarrow, \texttt{false}, \bot)\},$
$\{(1/8, g', 1, \uparrow, \texttt{false}, \bot)\},$
$\{(1/2, g', 1, \uparrow, \texttt{false}, \bot)\},$
$\{(1/4, d'', 1, \uparrow, \texttt{false}, \bot), (3/4, h', 1, \uparrow, \texttt{false}, \bot)\},$
$\{(1/8, d', 1, \downarrow, \texttt{false}, \bot)\},$
$\{(1/2, d'', 1, \uparrow, \texttt{false}, \bot), (1/2, d'', 1, \uparrow, \texttt{false}, \bot),$
$(1/2, g', 1, \uparrow, \texttt{false}, \bot), (1/2, e', 1, \downarrow, \texttt{false}, \bot)\},$
$\ldots\rangle$
The right part in the second row of Figure 5 holds
$\langle\{(1/8, c'', 1, \uparrow, \texttt{false}, \bot), (1/4, a', 2, \uparrow, \texttt{false}, \bot)\},$
$\{(1/8, h', 1, \uparrow, \texttt{false}, \bot)\},$
$\{(1/4, a', 1, \uparrow, \texttt{false}, \bot), (1/8, e', 2, \uparrow, \texttt{false}, (2, h', 1))\},$
$\{(1/8, d', 1, \uparrow, \texttt{false}, \bot)\},$
$\ldots\rangle$

module perLineas
$C = \{\uparrow, \downarrow, \perp\}$
// stem direction: up, down, not present
$\mathrm{Ptx} = \mathbb{N}_1 \times \mathbb{N}_1 \times \mathbb{N}_1$
// coordinates: score position, pitch, fine x
V *// given Type: Voice names*
$K = \mathbb{Q} \times \mathbb{N}_1 \times \mathbb{N}_1 \times C \times \mathsf{bool} \times \mathsf{optPtx}$
// note head = duration, pitch, fine x,
// stem, is pause, start of incoming lead line
$N = \mathsf{seq}(\mathbb{P}K \times \mathsf{iseq}V \times \mathsf{iseq}V)$
// score data = note heads and voice decls.
$R = V \nrightarrow \mathsf{iseqPtx}$
// result, maps voice names to the
// coordinates of their note heads

$\mathrm{lexSquash} : (A \nrightarrow \mathsf{seq}\mathbb{Z}) \to \mathsf{iseq}A$
// sort the key elements by the mapped values,
// lexicographically.
$\mathrm{sortH_X}, \mathrm{sortH_Y} : \mathbb{P}K \to \mathsf{iseq}\ K$
$\mathrm{sortH_X}(k) = \mathrm{lexSquash}(k \triangleleft \pi_3)$
$\mathrm{sortH_Y}(k) = \mathrm{lexSquash}(\lambda x : k \bullet (-1 * \pi_2(x), \pi_3(x)))$

module *perLineas.pausae
import perLineas
$d : \mathbb{N}_1$ *// number of voices active in* $\widehat{k}$
$\widehat{k} : \mathsf{iseq}K$
$\forall i, j \bullet i < j \implies \pi_2(\widehat{k}(i)) \geq \pi_2(\widehat{k}(j))$
$\mathrm{pausae}(\widehat{k}) = \{i : \mathbb{N}_1 \mid \pi_5(\widehat{k}(i)) = \mathit{true}\}$
$\mathrm{pSequ}(\widehat{k}) = \mathrm{pausae}(\widehat{k}) \setminus \mathit{succ}(\mathrm{pausae}(\widehat{k}))$
$\mathrm{pAdj}(\widehat{k}) \iff \#\mathrm{pausae}(\widehat{k}) > \#\mathrm{pSequ}(\widehat{k})$
$\mathrm{nonCommunes}(\widehat{k}) \iff d = \#\widehat{k}$
$\mathrm{maxCommunes}(\widehat{k}) \iff \neg\mathrm{pAdj}(\widehat{k})$
$\mathrm{defectis}(\widehat{k}) \iff d > \#\widehat{k} \wedge \mathrm{pAdj}(\widehat{k})$
$\iff \neg(\mathrm{nonCommunes}(\widehat{k}) \vee \mathrm{maxCommunes}(\widehat{k}))$
$\mathrm{ambig}(\widehat{k}) \iff \#d > \#\widehat{k} \wedge \#\mathrm{pSequ}(\widehat{k}) > 1$

Table 1. Note heads and sharing of pauses.

3.1 Sharing of pauses

The historical development process of CWN brought up many rules for special cases. These make the parsing process non-orthogonal and complicated. Two of them are

nota.perLineas.pausae.communes: Multiple pauses of the same duration in adjacent voices *may* be represented by one single graphical symbol.

nota.perLineas.pausae.maxCommunes: Multiple pauses of the same duration in adjacent voices *must* be represented by one single graphical symbol.

The first rule is implied in the second. Module *perLineas.pausae* in Table 1 shows the definitions of the most important respective properties, when $\widehat{k}$ is an injective list of note heads and pause symbols, sorted by vertical height.

To recognize (=parse) sharing of pauses, the total number of intended voices must be known by some preceding explicit voice declaration. This is modeled by d in that Table.

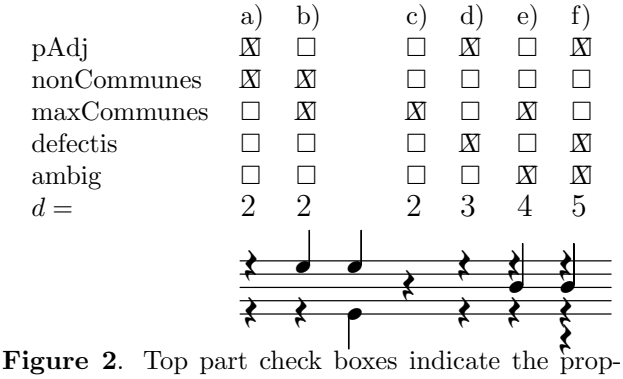


Figure 2. Top part check boxes indicate the properties of the note example below, given a particular voice count d.

The set pausae contains all those indices where a pause stands; the set $\mathrm{pSequ}(\widehat{k})$ all those where a maximal group of adjacent pauses starts, e.g. $\{1,2,4\} \setminus \{2,3,5\} = \{1,4\}$. The property $\mathrm{pAdj}(\widehat{k})$ indicates that there are adjacent pause symbols. Figure 2 shows minimal examples for characteristic combinations:

nota.perLineas.pausae.maxCommunes: All occurring maximal groups of adjacent pauses have been replaced by one single pause symbol.

nota.perLineas.pausae.nonCommunes: No pair of adjacent pauses has been replaced by one single pause symbol.

nota.perLineas.pausae.defectis: At least one pair of adjacent pauses has been replaced by one single pause symbol but another pair has not.

nota.perLineas.pausae.ambig: There are multiple pause symbols which can be expanded to the eliminated pause symbols.

The minimal voice number for ambiguity is indeed four: One non-pause voice is required to separate the two pause symbols; one of these stands for two voices. To combine this with *defectis* requires a further voice with a non-replaced pause symbol.

These attributes play an important role in practice. The piano reduction contained in Wolfgang Graeser's edition of Bach's "Die Kunst der Fuge" BWV 1080 [14] consequently follows nonCommunes, which implies ¬ambig and thus makes voice leading as clear as possible. [6] In contrast, Walter Denhard's edition of "Das wohltemperierte Klavier I" [15] changes frequently: Fuga I à 4 starts with ambig, but Fuga III à 3 with ¬maxCommunes ∧ nonCommunes. Fuga IV à 5 starts with maxCommunes ∧ ¬ambig. [7] The second half of measure 7 starts a new declaration segment of only three voices (see section 2.3 above), because only these are notated without any accompanying pause. Similar in measure 21, when the pauses of voice 2 suddenly

[6] For these remarks we consider the two staves of the keyboard notation as one single staff. The formally correct treatment of this case is complicated and out of scope in this paper.

[7] Reading both staves into one additionally yields defectis, which is an artefact.

disappear while those of voice 5 are visible. Contrarily, measure 35 switches to ¬maxCommunes, showing adjacent note symbols. Starting with measure 77, even ledger lines are employed for these.

3.2 Crossings of iso-rhythmic voices

One of the main concerns in many contexts is a lightweight but unambiguous way to indicate *temporal voice crossings*. This means that the physical top-down order of the sounding notes of the voices contradicts the declared voice name order (= CVO), but only for a short time and without the need for a full voice order re-declaration.

According to the historic evolution, all voices with *stem up* notes are read as a multiplication (or splitting) of one original "upper voice" or "first voice", and all *stem down* notes correspond to a "second voice". Therefore a convenient but limited means for voice crossing is the *stem direction*. Given a certain explicit voice declaration $\langle v_1, v_2, \ldots, v_n \rangle$, and a set of n note heads in which $u < n$ are stem-up, then the first voice names $\langle v_1, \ldots, v_u \rangle$ are assigned to the stem-up note heads, from top to bottom, and the stem-down note heads are mapped to $\langle v_{u+1}, \ldots, v_n \rangle$, again from top to bottom.

In an iso-rhythmic setting, voice crossings can be expressed between the lowest voices with up-stem and the highest voices with down-stem, but not among voices with the same stem direction. Figure 3 shows all possible situations for four-voice chords.

The number of all expressible voice crossings is calculated as follows: Given a chord of n note heads, all with stems and with heads on different heights (= different pitches), and u of them shall be stem-up. Then there are $\binom{n}{u}$ possibilities to assign stem directions. $u = 0$ and $u = n$ do not allow any voice crossing, so we get

$$\sum_{u=1}^{n-1} \binom{n}{u}$$

for the number of possible assignments. For each u, there is one combination which is not a crossing, namely the first in each group in Figure 3. For the number of all possible permutations (original order plus all crossings) we must thus subtract $n - 2$. All other permutations are indeed different, what can be shown as follows: for any given u, the voices u and $u + 1$ (counted from the top) have different stem directions and thus are crossed in all combinations but the very first (see Figure 3). For all $m \neq u$ the voices m and $m + 1$ have the same stem direction and thus never cross. So all permutations are different.

Further normalization yields

$$\sum_{u=1}^{n-1} \binom{n}{u} - n + 2 = \sum_{u=1}^{n-1} \binom{n}{u} - n + \binom{n}{0} + \binom{n}{n}$$

$$= \sum_{u=0}^{n} \binom{n}{u} - n = 2^n - n$$

The table in Figure 3 shows the absolute and relative numbers of expressible voice crossings in the iso-rhythmic case.

4. PARSING OF FREE RHYTHMIC VOICES

In hetero-rhythmic settings more voice crossings are possible: At each score position only the active voices (as defined above) get a new note head. [8] So arbitrary voice crossings can be notated between an active and an overlapping voice, see Figures 4 and 7. Additionally, for more flexible voice crossings, it may be sensible to allow *changes of the stem direction*, which makes parsing significantly more complicated.

Voice crossings between active voices can be indicated by explicit *voice leading lines*. These connect visibly the latest preceding note head of a particular active voice with a note head at the current score position and declares graphically that both note heads belong to the same voice, see Figure 5.

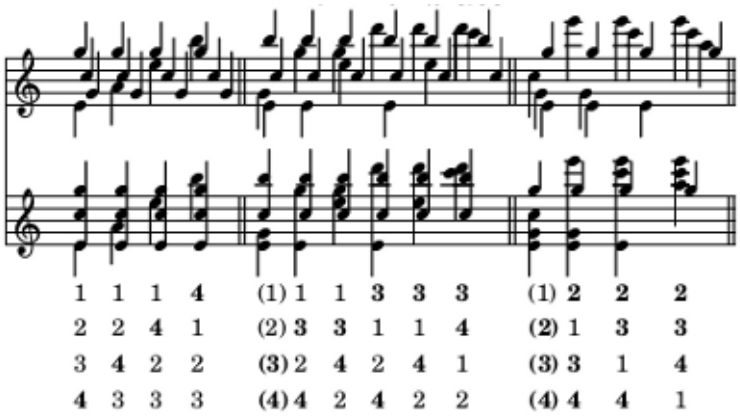
4.1 Voice (re-)declarations

Initial voice declarations (at the beginning of a staff or a segment) must mention all present voices. They can come in two formats: (a) as one list of voice names for all voices, or (b) as two separate lists for up- and down-stem voices, written above and below the staff. In any case, the respective numbers of names and voices must match. In case of the single list, the leading names are matched to the up-stem voices and the trailing names to the down-stem voices. The version with two lists can contain slightly more information when the staff begins with stemless notes, because it declares the voices in the second list as "nominally down-stemmed", see Section 4.7 below.

A re-declaration stands at a particular score position in the midst of a segment. It comes in similar formats: (a) one list for both stem directions or (b) two lists for up-stemmed and down-stemmed each, or (c) one list only for up-stemmed or (d) one list only for down-stemmed heads. These formats must be recognizable by the position(s) and length(s) of the list(s). Furthermore, the declaration can address different targets: (A) All voices appearing in the segment, or (B) only the voices with a note head at this score position, or (C) only the active voices (=only the heads not tied to its predecessors).

Re-declarations either (α) change the CVO, or (β) they redundantly only confirm it and show it to the reader for convenience. These are **two fundamental different semantics**, because the latter can be erased from the notation without changing its meaning. This results in twenty-four(24) combinations for the types of voice re-declaration, all of which can be found in practice. Figure 4 shows some examples.

[8] In the following we write "note head" for shortness, also when "note head or pause symbol" is meant.



n	$n!$	$2^n - n$	expressible/total
2	2	2	1
3	6	5	0.833
4	24	12	0.5
5	120	27	0.225
6	720	58	0.0805
7	5040	121	0.0224

Figure 3. Voice crossings expressible by stem directions

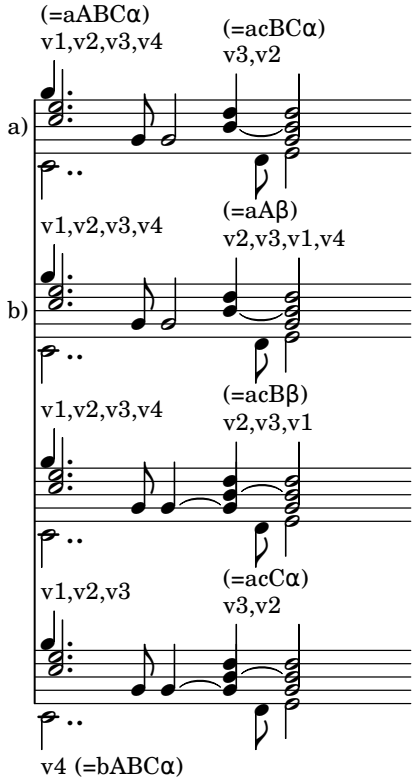


Figure 4. Voice (re-)declarations and crossings

Ambiguities can only arise between cases cA vs. aB, cA vs. aC, and cB vs. aC, and can in most cases be resolved by the nominal stem direction of the named voices.

4.2 Preparatory steps

To apply our algorithm to concrete examples from practice, some *preparatory transformations* must be applied. These model the extraction of *additional information* for voice parsing (e.g. from beams and slurs) which is done by the trained musicians brain unknowingly. We realize it by adding explicit voice lead lines, see Figure 5. Additionally all *ties* are removed by replacing e.g. the combination by a

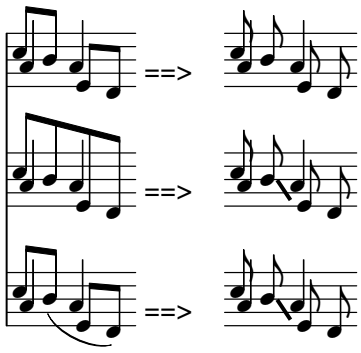


Figure 5. Preparatory adding of voice lead lines from beams and slurs (not formalized in this paper)

single (artificial) note head with a duration value of "5/8". [9]

4.3 Parsing

The following tables show different versions of a parsing algorithm. This is done in a *modular* way of specification. Tables 2 and 3 show the fundamental and auxiliary functions, common to all versions. Table 4 gives the more simple variant which treats filtered segments containing only one stem direction; Table 5 shows the more complex version in which each voice may arbitrarily change its stem direction. Both algorithms come in two different flavors, *suonaeProximae* and *solumExplicitum*, as explained below. Result is R, which gives for each voice the sequence of the coordinates of all note head contained in that voice. [10]

The parsing function calc($n : N$) calls the real computation function step(..) [11] with this input data, the

[9] Indeed, the problem of "ties vs. slur" has most intricate cases, philosophically and mathematically most complex. This is left out in this article. E.g., the function assign_D(..) in Table 2, which treats explicit re-declarations, does not treat those of type "B" but only "C", so it does not need to know about incoming ties.

[10] Since coordinates represent the score position order internally, the modeling as sequences per voice is redundant, but practical; semantically sets would suffice.

[11] We write "f(..)" to refer to a function by its name, while omitting the full type signature.

module *perLineas.extrahere[H] // H = *auxiliary data = running status, parameterizable = CVO*
import perLineas
read : $N \times \mathrm{Ptx} \nrightarrow K$ // *extract note head at given coordinate from score data*
store : $R \times (V \times \mathbb{N}_1 \times K) \to R$ // *assign voice at score position to note head*
calc : $N \nrightarrow R$ // *main function = analyze score data and initial voice declaration*
$*\mathrm{initH} : N \nrightarrow H$ // *initialize running status; defined by importing module*
step : $N \times \mathbb{N}_1 \times (V \nrightarrow \mathbb{Q}) \times R \times H \nrightarrow R \times H$ // *score data, score position, voice ends, result, running status*
$\mathrm{assign_W} : \mathbb{N}_1 \times \mathbb{P}V \times \mathbb{P}K \times \mathbb{P}K \times \mathsf{seq}V \times \mathbb{P}V \times R \times R \times H \nrightarrow R \times H$
// score pos., active voices, heads with/without lead, re-declaration, active voices, old/current result, aux state
$\mathrm{assign_F} : \mathbb{N}_1 \times \mathsf{seq}V \times \mathsf{seq}K \times R \times H \times \mathbb{P}V \times R \nrightarrow R \times H$
// score position, active voices, note heads, result, running status, old active voices, old result
$\mathrm{assign_D} : \mathbb{N}_1 \times \mathbb{P}V \times \mathbb{P}V \times \mathbb{P}V \times \mathbb{P}V \times \mathsf{seq}K \times \mathsf{seq}V \times R \times H \times \mathbb{P}V \times R \nrightarrow R \times H$
// score pos., overlap/tied/led/active voices, sorted heads, voice decls, result, old CVO, up-stem voices, old result
$*\mathbf{phase2} : \mathbb{N}_1 \times \mathbb{P}V \times \mathbb{P}K \times \mathsf{seq}V \times \mathbb{P}V \times R \times R \times H \nrightarrow R \times H$
// score pos., active voices, heads without lead, re-declaration, active voices, old/current result, aux state
$*\mathbf{finalize} : \mathbb{N}_1 \times R \times H \times \mathbb{P}V \times R \nrightarrow R \times H$
// score position, result, running status, old active voices, old result
lastOrd : $R \times \mathbb{P}V \to \mathsf{iseq}\ V$ // *sort the set of voices according to their latest occurrence in the result*
$*\mathrm{aboveVK}_{\_,\_}, *\mathrm{equalVK}_{\_,\_} : R \times \mathbb{P}V \to (V \leftrightarrow K)$
// result × stem dir → relation whether a overlapping voice is above/on equal height with a note head,

$$\mathrm{read}(p,(m,t,x)) = k \iff k \in \pi_1(p(m)) \ \wedge k = (\_,t,x,\_,\_,\_)$$

$$\mathrm{store}(r,(v,m,(\_,t,x,\_,\_,\_))) = r \oplus \{v \mapsto r(v) \frown \langle (m,t,x)\rangle\}$$

$$\frac{\mathrm{initH}(p) = (v,u)}{\mathrm{calc}(p) = \pi_1(\ \mathrm{step}(p,1,(\mathsf{ran}v)\times\{0\},(\mathsf{ran}v)\times\{\langle\rangle\},(v,u)))}$$

$$\frac{\begin{array}{c} q_0 = \mathsf{min}(\mathsf{ran}\,q) \quad \widehat{v} = q^{-1}(\!|\{q_0\}|\!) \quad \widehat{k} = \pi_1(p(m)) \quad d = \pi_2(p(m)) \\ \#\widehat{v} \neq \#\widehat{k} \implies \mathtt{error}(\text{“Numbers of expected and notated note heads differ”},m) \\ k_\mathrm{W} = \{k \in \widehat{k} \mid \pi_6(k) \neq \bot\} \quad k_\mathrm{F} = \widehat{k} \setminus k_\mathrm{W} \\ \#k_\mathrm{W} = 1 \ \wedge \ \#k_\mathrm{F} = 0 \implies \mathtt{warning}(\text{”Redundant voice leading line”},m) \\ (r',h') = \mathrm{assign_W}(m,\widehat{v},k_\mathrm{W},k_\mathrm{F},d,\widehat{v},r,r,h) \\ q' = q \oplus \lambda v : \widehat{v} \bullet q_0 + \pi_1[\mathrm{read}(p,\mathit{last}(r'(v)))] \end{array}}{\mathrm{step}(p,m,q,r,h) = \mathrm{step}(p,m+1,q',r',h')}$$

$$m > \#p \implies \mathrm{step}(p,m,\_,r,h) = (r,h)$$

$$\frac{\begin{array}{c} k_\mathrm{W} = \pi_6(k) : \mathrm{Ptx} \quad \exists v_\mathrm{W} \bullet \mathit{last}(r(v_\mathrm{W})) = k_\mathrm{W} \\ v_\mathrm{W} \notin v \implies \mathtt{error}(\text{“Leading line comes from non-active voice”},\ k) \\ r' = \mathrm{store}(r,(v_\mathrm{W},m,k)) \quad v' = v \setminus \{v_\mathrm{W}\} \end{array}}{\mathrm{assign_W}(m,v,\{k\} \cup k_\mathrm{W},k_\mathrm{F},d,v_0,r_0,r,h) = \mathrm{assign_W}(m,v',k_\mathrm{W},k_\mathrm{F},d,v_0,r_0,r',h)}$$

$$\mathrm{assign_W}(m,v,\{\},k_\mathrm{F},d,v_0,r_0,r,h) = \mathbf{phase2}(m,v,k_\mathrm{F},d,v_0,r_0,r,h)$$

Table 2. Common and auxiliary functions for parsing two or more voices per staff

next score position to process (initially $m = 1$), a map q of type $V \nrightarrow \mathbb{Q}$ which gives the next time point for each voice (initially all point to 0), and the result accumulator, which is initially empty.

The data type H is a parameter specific for the variants of the algorithm. It threads additional auxiliary status data through all function calls. Its initial value is delivered by initH(..), which must be defined accordingly in the importing module. [12] For brevity of the algorithm, the score must carry a complete voice order declaration of Type aAα with the very first event, and (for the double stemmed case) the nominal stem direction must be visible from the very first note heads.

[12] Therefore the head of the module is marked with an asterisk, meaning “not a complete schema in the sense of Z”.

The function step(..) calculates the next time point to process ($= q_0$) as the lowest value in the map q, and the set $\widehat{v}$ of all voices active there. $\widehat{k}$ are the note heads at the next score position m, and d the voice name re-declaration there, which at most score positions will be empty. The set $\widehat{k}$ is divided into those note heads at which an explicit voice leading line arrives ($= k_\mathrm{W}$, from German “Stimm*W*eiser”) and the *F*ree rest ($= k_\mathrm{F}$). First $\mathrm{assign_W}(..)$ is called to process k_W. Afterwards the two variants of **phase2**(..) (in Tables 4 and 5) call the common functions $\mathrm{assign_D}(..)$ if there is an explicit voice re-declaration, or $\mathrm{assign_F}(..)$ if not. Afterwards, the specific function **finalize**(..) updates the running status H.

$$\text{assign}_\text{F}(m, \langle v\rangle \frown \alpha, \langle k\rangle \frown \beta, r, h, v_0, r_0) = \text{assign}_\text{F}(m, \alpha, \beta, \text{store}(r, (v, m, k)), h, v_0, r_0)$$

$$\text{assign}_\text{F}(m, \langle\rangle, \langle\rangle, r, h, v_0, r_0) = \textbf{finalize}(m, r, h, v_0, r_0)$$

$$\frac{A = \lambda v \in \text{dom} r \cap \widehat{v} \mid k = \mathit{last}(r(v)) \bullet (-1 * \pi_2(k), \pi_1(k), \pi_3(k))}{\text{lastOrd}(r, \widehat{v}) = \text{lexSquash}(A)}$$

$$r' = \begin{cases} \textbf{if } \#d = \#v3 + \#v4 & \textbf{then } \text{zip}(m, v_4, \text{lastOrd}(r, v_3), \widehat{k}, d, r, U) \\ \textbf{else if } \#d = \#v2 + \#v3 + \#v4 & \textbf{then } \text{zip}(m, v_4, \text{lastOrd}(r, v_2 \cup v_3), \widehat{k}, d, r, U) \\ \textbf{else if } \#d = \#v1 + \#v2 + \#v3 + \#v4 & \textbf{then } \text{zip}(m, v_4, \text{lastOrd}(r, v_1 \cup v_2 \cup v_3), \widehat{k}, d, r, U) \\ \textbf{else} & \texttt{error}(\textit{“wrong length of voice declaration list”}) \end{cases}$$

$$\text{assign}_\text{D}(m, v_1, v_2, v_3, v_4, \widehat{k}, d, r, h, U, r_0) = \textbf{finalize}(m, r', h, v_3 \cup v_4, r_0)$$

$\text{zip}(m, \widehat{v}, \langle v\rangle \frown \alpha, \langle k\rangle \frown \kappa, \langle d\rangle \frown \delta, r, U)$
// score pos., voices to allocate, fixed voices, heads to allocate, declarations, result, nominal upstem voices

$$= \begin{cases} \textbf{if } (v \text{ aboveVK}_{r,U}\ k) \textbf{ then } \begin{cases} \textbf{if } v = d & \textbf{then } \text{zip}(m, \widehat{v}, \alpha, \langle k\rangle \frown \kappa, \delta, r, U) \\ \textbf{else} & \texttt{error}(\textit{“declaration contradicts overlap/lead”}, d, v) \end{cases} \\ \textbf{else } (v \text{ equalVK}_{r,U}\ k) \wedge v = d & \textbf{then } \text{zip}(m, \widehat{v}, \alpha, \langle k\rangle \frown \kappa, \delta, r, U) \\ \textbf{else if } d \in \widehat{v} & \textbf{then } \text{zip}(m, \widehat{v} \setminus \{d\}, \langle v\rangle \frown \alpha, \kappa, \delta, \text{store}(r, (d, m, k)), U\ ) \\ \textbf{else} & \texttt{error}(\textit{“Voice for note head is not active”}, d, k) \end{cases}$$

$$\text{zip}(m, \{\}, \langle v\rangle \frown \alpha, \langle\rangle, \langle d\rangle \frown \delta, r, \_) = \begin{cases} \textbf{if } v = d & \textbf{then } \text{zip}(m, \{\}, \alpha, \langle\rangle, \delta, r, \_) \\ \textbf{else} & \texttt{error}(\textit{“declaration contradicts overlap/lead”}, \end{cases}$$

$$\text{zip}(m, \{\}, \langle\rangle, \langle\rangle, \langle\rangle, r, \_) = r$$

Table 3. Common and auxiliary functions, continued.

module perLineas.extrahere. suonaeProximae / solumExplicitum . X / Y
import perLineas.extrahere[**iseq***V*] *// parameter H used to pass through the CVO*

$$\frac{v = \pi_2(p(1))}{\text{initH}(p) = \textbf{if } (\#v = \#\pi_1(p(1))) \textbf{ then } v \textbf{ else } \texttt{error}(\textit{“Error in initial voice declaration”})}$$

$\textbf{phase2}(m, v, k_\text{F}, d, v_0, r_0, r, h)$ *// $\#k_\text{F} = \#v$*

$$= \begin{cases} \textbf{if } d = \langle\rangle & \textbf{then } \text{assign}_\text{F}(m, \mathit{squash}(h \triangleright v), \text{sortH}_{\boxed{\text{X}/\text{Y}}}(k_\text{F}), r, h, v_0, r_0) \\ \textbf{else} & \text{assign}_\text{D}(m, \text{ran}h \setminus v_0, \{\}, v_0 \setminus v, v, \text{sortH}_{\boxed{\text{X}/\text{Y}}}(k_\text{F}), d, r, h, \{\}, r_0) \end{cases}$$

$\textbf{finalize}(m, r, h, v_0, r_0) = (r, \boxed{\text{lastOrd}(r, \mathbb{N}_1) / h \fatsemi (\textbf{ID}_{\mathbb{N}_1} \oplus (\text{lastOrd}(r_0, v_0)^{-1} \fatsemi \text{lastOrd}(r, v_0)))}\ )$

$v \text{ aboveVK}_{r,\_}\ k \iff \pi_2(\mathit{last}(r(v))) > \pi_2(k)$
$v \text{ equalVK}_{r,\_}\ k \iff \pi_2(\mathit{last}(r(v))) = \pi_2(k)$

Table 4. Parsing multiple voices with all the same stem direction

On return from $\text{assign}_\text{W}(..)$, the function step(..) calculates for all active voices their next time points in the map q': $\mathit{last}(r'(v))$ is the latest score coordinate recognized as part of v; $\text{read}(p, \mathit{last}(r'(v)))$ reads the event at this position of the input score; $\pi_1(..)$ extracts the duration. Then step(..) calls itself recursively, until the input data from N is exhausted.

In $\text{assign}_\text{W}(..)$, the variable k steps through the note heads k_W; the start coordinate of the voice leading line arriving there ($= \pi_6(k)$) must come from a note head which exactly ends at the current time point. So the voice assigned to it must be active ($v_w \in v$). It is assigned to k by calling the auxiliary function store(..) and is removed from the set v. [13]

The function $\text{assign}_\text{F}(..)$ simply assigns the voices in its second argument to the note heads in its third, in that sequential order provided by the caller.

The function $\text{assign}_\text{D}(..)$ gets in $v_1..v_4$ the sets of voices which are overlapping/tied/with lead-lines/still to allocate, and in d the declaration text. It recognizes cases aA, aB and aC, as defined in Section 4.1, by comparing the cardinalities of the sets. [14] It always

[13] The set of all initially active voices and the old result, before evaluating the voice leading lines, are additionally passed through by the function arguments v_0 and r_0, because they are needed later by one of the algorithm's variants.

[14] For brevity, cases b to d are not supported and case B is not called by the other modules: for brevity, ties are not modeled in our data. The algorithm treats cases α and β uniformly and

holds that $\#v_4 = \#\widehat{k}$. It calls zip(..), which iterates synchronously over three lists and one set. It gets as many declarations in d as voices in $\widehat{v} \cup v'$, and as many un-assigned voices in $\widehat{v}$ as heads to assign in $\widehat{k}$. It can assign the next note head (in the respective sorting order) to the next entry in the declaration, if this voice is in the set $\widehat{v}$ (= still unassigned). Otherwise it checks whether the topmost assigned voice (=overlapping, with lead lines, etc.) is in sync with the declaration. When the topmost note head and the topmost assigned voice are on the same height, both alternatives for their nominal order are considered. The set U of all nominal up-stemmed voices is only needed to compare the height of the current note head and the current assigned voice: in the double-stem case each nominally up-stemmed voice is infinitely higher than any down-stemmed note head.

4.4 Parsing only one stem direction

The first and more simple case is to process only all voices with one particular stem direction, see Table 4. The arguments for $\text{assign}_\text{F}(..)$ are simply the currently active voices in the sequential order of the CVO and the sorted noted heads. ($h \triangleright v$ is the "range restriction", which selects from the sequence h all maplets which point into v, and *squash*(..) compactifies this to a sequence.) The note heads are all those with no arriving lead line, sorted according to the selected method, see next section.

4.5 Two methods of note head sorting

For the sorting of the note heads (see Tables 4 and 5, for calling $\text{assign}_\text{D}(..)$ and $\text{assign}_\text{F}(..)$) there are two different methods: sortH_X sorts by the "fine x" horizontal position only; sortH_Y sorts by pitch = vertical position, and subordinately by fine x position, if necessary. Both functions are defined in Table 1.

These two methods imply **fundamentally different ways to express short-term voice crossings**: The method *perAltitudinemCaputis* uses sortH_Y and assumes that the physically lower voice is mentioned later in the CVO. The horizontal position is used as secondary criterion, only in cases of equal height. This method is widely used and allows sharing of stems and note heads (see below Section 5).

The method *perCaudaeSequentiam* uses sortH_X and defines the sequential order by the x position of the stem only. Therefore it can easily express short-time voice crossings, see Figure 6. It is not found as frequently as the preceding variant, but can also be found in practice.

4.6 Two strategies of changing the CVO

The four modules combined in Table 4 also differ in the strategy the CVO is finally affected by the crossings

can easily be enhanced for detecting and signaling them.

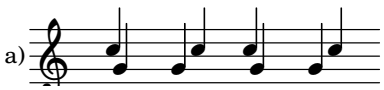
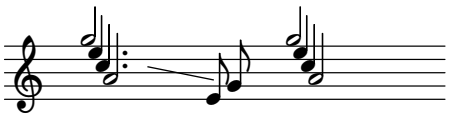


Figure 6. Voice crossing by x position of stems: voice one goes c-g-c-g, vs. voice two with g-c-g-c



suonaeProximae	1	1	1
(=physical)	2	**4**	4
	3	**3**	3
	4	**2**	2
solumExplicitum	1	1	1
(=nominal)	2	**3**	3
	3	**2**	2
	4	4	4

Figure 7. Physical vs. explicit-only changes of the CVO. Voice names below notes, changes in **bold**.

between active and overlapping voices: *perLineas.extrahere.suonaeProximae* takes the complete finally resulting *physical* order (calculated by $\text{lastOrd}(r, \mathbb{N}_1)$, see the left part of the last box in the Table) as the new CVO, which will reign the parsing step at the subsequent score position.

Contrarily, *perLineas.extrahere.solumExplicitum* lets only explicit changes (by voice order re-declaration or by voice lead lines) change the CVO. The calculation is more complicated: $\text{lastOrd}(r_0, v_0)$ is the sequential order of only the free voices, before any processing/assignment at this score position had been started. $\text{lastOrd}(r, v_0)$ is the same after all evaluation. So $\text{lastOrd}(r_0, v_0)^{-1} \fatsemi \text{lastOrd}(r, v_0)$ is the permutation of these voices, a mapping from voice name to voice name. This permutation is now expanded to a mapping which is the identity on all other voices and then applied to the old CVO. (This is the only place where the function parameters v_0 and r_0 are needed.)

The effects of both strategies are illustrated in Figure 7: The last chord has different voice assignments depending on the CVO calculated after processing the next-to-last score position. These are again **two fundamentally different semantics** which must be declared or found out when talking about a given text. [15]

4.7 Parsing both stem directions

Here the parameter H is set to $\mathsf{iseq}V \times \mathbb{N}_0$. The CVO is the sequence of voice names, first the up-stemmed, then the down-stemmed, and the additional value (called u in the following) is the index after which the latter start (=the count of up-stemmed voices).

[15] Of course, *suonaeProximae* is only sensible in combination with *perAltitudinemCaputis*. In *perCaudaeSequentiam* the height of notes does not influence the CVO anyhow. See Section 5 for a survey on the sensible combinations of strategies.

module perLineas.extrahere.duplex. suonaeProximae / solumExplicitum . X / Y
import Lmn.nota.perLineas.extrahere[iseq$V \times \mathbb{N}_0$]

$$\frac{K = \pi_1(p(1)) \quad v = \pi_2(p(1)) \quad k_{\square\in\{\uparrow,\perp,\downarrow\}} = \{k \in K \mid \pi_4(k) = \square\}}{\mathrm{initH}(p,v) = \begin{cases} \textbf{if } \#v \neq \#K & \textbf{then } \texttt{error}(\textit{“Error in initial voice declaration”}) \\ \textbf{else if } \#k_\perp = 0 & \textbf{then } (v, \#k_\uparrow) \\ \textbf{else} & \texttt{error}(\textit{“Nominally up-stemmed voices not visible.”}) \end{cases}}$$

$$\frac{\begin{array}{c} k_{\square\in\{\uparrow,\perp,\downarrow\}} = \{k \in k_\mathrm{F} \mid \pi_4(k) = \square\} \quad \widehat{k} = \mathsf{anyPerm}(k_\uparrow) \frown \mathrm{sortH}_{\boxed{\mathrm{X}/\mathrm{Y}}}(k_\perp) \frown \mathsf{anyPerm}(k_\downarrow) \\ k_\mathrm{U} = \mathrm{sortH}_{\boxed{\mathrm{X}/\mathrm{Y}}}(\widehat{k}(\!|\{1..u\}|\!)) \quad k_\mathrm{D} = \mathrm{sortH}_{\boxed{\mathrm{X}/\mathrm{Y}}}(k_\mathrm{F} \setminus \mathsf{ran}(k_\mathrm{U})) \end{array}}{\begin{array}{c} \textbf{phase2}(m, v, k_\mathrm{F}, d, v_0, r_0, r, h) \\ = \begin{cases} \textbf{if } d = \langle\rangle & \textbf{then } \mathrm{assign}_\mathrm{F}(m, \mathit{squash}(h \triangleright v), k_\mathrm{U} \frown k_\mathrm{D}, r, (h,u), v_0, r_0) \\ \textbf{else} & \mathrm{assign}_\mathrm{D}(m, \mathsf{ran}h \setminus v_0, \{\}, v_0 \setminus v, v, k_\mathrm{U} \frown k_\mathrm{D}, d, r, h, h(\!|\{0..u\}|\!), r_0) \end{cases} \end{array}}$$

$$\frac{\begin{array}{c} v_\uparrow = \{x \in v_0 \mid \pi_4(\mathit{last}(r(x))) = \uparrow\} \quad v_\downarrow = \{x \in v_0 \mid \pi_4(\mathit{last}(r(x))) = \downarrow\} \\ v_\mathrm{U} = h(\!|\{0..u\}|\!) \quad v_\mathrm{D} = (\mathsf{ran}\, h) \setminus v_\mathrm{U} \quad v'_\mathrm{U} = (v_\mathrm{U} \cup v_\uparrow) \setminus v_\downarrow \quad v'_\mathrm{D} = (v_\mathrm{D} \cup v_\downarrow) \setminus v_\uparrow \quad u' = \#v'_\mathrm{U} \end{array}}{\begin{array}{l} \textbf{finalize}(m, r, (h,u), v_0, r_0) = (r, (\boxed{\mathrm{lastOrd}(r, v'_\mathrm{U}) \frown \mathrm{lastOrd}(r, v'_\mathrm{D})/} \\ \quad \boxed{/\mathrm{rotate}(h, v_0, \#(v_\mathrm{U} \cap v_0), u, u') \fatsemi (\mathbf{ID}_{\mathbb{N}_1} \oplus ((\mathrm{lastOrd}(r_0, v_0)^{-1} \fatsemi \mathrm{lastOrd}(r, v_0))))}, u')) \end{array}}$$

$\mathrm{rotate} : \mathsf{iseq}V \times \mathbb{P}V \times \mathbb{N}_1 \times \mathbb{N}_1 \times \mathbb{N}_1 \to \mathsf{iseq}V$
// CVO, active voices, number of active & orig. up-stem voices, old/new number of up-stem voices
$v_\mathrm{P} = (\mathit{squash}(h \triangleright v))(w + u' - u) \quad p = h^{-1}(v_\mathrm{P})$ *// max index to rotate*
$s = \textbf{if } u' > u \textbf{ then } \{(u+1)..p\} \triangleleft h \textbf{ else } \{p..u\} \triangleleft h$

$$\frac{s' = \begin{cases} \textbf{if } u = u' & \textbf{then } \mathbf{ID}_V \\ \textbf{else if } u' > u & \textbf{then } \mathit{squash}(s)^{-1} \fatsemi (\mathit{squash}(s \triangleright v) \frown \mathit{squash}(s \triangleright\!\!- v)) \\ \textbf{else} & \mathit{squash}(s)^{-1} \fatsemi (\mathit{squash}(s \triangleright\!\!- v) \frown \mathit{squash}(s \triangleright v)) \end{cases}}{\mathrm{rotate}(h, v, w, u, u') = h \fatsemi (\mathbf{ID}_V \oplus s')}$$

$v\ \mathrm{aboveVK}_{r,U}\ k \Longleftrightarrow (v \in U \wedge \pi_3(k) = \downarrow) \vee ((v \in U \vee \pi_3(k) \neq \uparrow) \wedge (\pi_2(\mathit{last}(r(v))) > \pi_2(k)))$
$v\ \mathrm{equalVK}_{r,U}\ k \Longleftrightarrow (\pi_4(k) = \perp \vee (v \in U \Longleftrightarrow \pi_4(k) = \uparrow)) \quad \wedge (\pi_2(\mathit{last}(r(v))) = \pi_2(k))$

Table 5. Parsing more than two voices with changing stem direction

Each voice is always treated either as “nominally up-stemmed” or “nominally down-stemmed”, even if the current note head (or pause symbol) does not carry a visible stem. The membership in these two groups is only altered if this is *unavoidable*, i.e. unambiguously indicated by the graphical input.

The algorithm in the modules *perLineas.extrahere.duplex. suonaeProximae/solumExplicitum* in Table 5 is basically the same as in *perLineas.extrahere.suonaeProximae/solumExplicitum* in Table 4. Main difference is that the criterion whether a note head is higher than a particular voice or higher than another note head is additionally affected by the stem direction: All up-stem voices and up-stem heads are infinitely higher than all down-stem voices and down-stem heads. This rule comes into play when sorting note heads, comparing note heads with voices (see the new definition of $\mathrm{aboveVK}_{\_,\_}$) and calculating the new CVO. For example, the physical variant *suonaeProximae* (see the left part of the last box in Table 5) extracts the physical order of all up-stem voices and appends that of the down-stemmed, so that physical crossings between both groups do not affect the CVO.

4.8 Voices changing the stem direction

The algorithm assigns the note heads to the active voices in the above-mentioned order: all up-stemmed precede all down-stemmed. But the numbers need not match: There can be more or less nominally up-stem voices in the currently active voices than there are up-stem note heads. The algorithm applies the minimal necessary change of direction assignments on the fly.

Only in case *solumExplicitum* special means must be taken by prepending the permutation delivered by rotate(..), see Figure 8: The y-axis means increasing pitch, with up-stem notes infinitely higher than down-stems; all horizontal lines are overlapping voices; the four eighth notes at score position T_1 (1 up and 3 down) shall proceed to the four eighth notes at T_2 (3 up and 1 down). There are explicit voice leading lines for v1 and v8 (see the solid lines), the proceedings of v4 and v6 follow from the algorithm (see the dotted

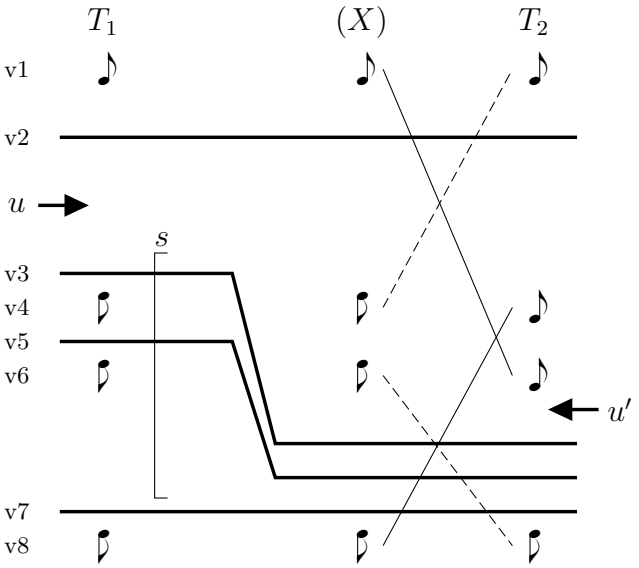


Figure 8. Preparatory rotation of voice names to enable the necessary change of stem directions

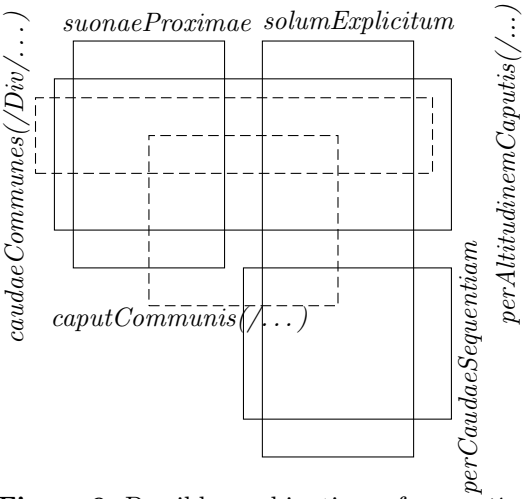


Figure 9. Possible combinations of *semantical* strategies (solid boxes) vs. graphical representations = *stylistic* variants (dashed boxes)

lines). The algorithm *perLineas.extrahere.solumExplicitum* simply applied the resulting permutation of the voice names to the CVO. But here the value u (= the number of up-stemmed voices) changes, and the lowest up-stemmed voice (v1 after the permutation) must precede the highest down-stemmed voice (v3) in the CVO. Therefore the graphically indicated permutation rotate(..) (as defined in Table 5) of all voices in $s = \{v3, v4, v5, v6\}$ is prepended. This leads to the intermediate situation (X). Now u can be replaced by u', changing the number of nominally up-stemmed voices, and the permutation of voice names can be applied as in the simple case.

5. GRAPHICAL APPEARANCE OF MULTI-VOICE STAVES. POSSIBLE COMBINATIONS OF STRATEGIES

The strategy *perLineas.caudaeCommunes* allows the note heads of different voices with the same score position, the same stem direction and the same head form to share the graphical representation of the stem. Frequently found is also the more lenient variant *perLineas.caudaeCommunes.div* allowing stem sharing for note heads of different forms (i.e. quarters and halfs), and the more restricted *perLineas.caudaeCommunes.idemPuncta* requiring the same head form and the same number of prolongation dots [16, pg.55]. While the parsing method *perAltitudinemCaputis* is used, these transformations are purely graphical and can be introduced or removed without changing the information contents. This is not longer true with *perAltitudinemCaputisVelSequentiam* and *perCaudaeSequentiam*. A further wide-spread restriction is *perLineas.caudaeCommunes.trabsCompleta*: If the stems are connected to a *beam*, than either all or none of the notes of the two voices under this beam share their stems.

Vice versa, *perLineas.caputCommunis* allows two voices with stems in different directions to share a note head, if score position, vertical position, duration (including prolongation dots) and accidentals are the same.

More lenient is *[nota.vox.perLineam.caputCommunis .punctaMixta*, which allows different numbers of prolongation dots. This can be found in practice (Beethoven, piano sonata op14/1, 1.mvmt., m.7pp,[17]) but is not always accepted in text books ([16, pg.55, point(7)]).

Figure 9 shows the possible combinations of these graphical strategies with the different semantical strategies defined in this paper.

6. REFERENCES

[1] M. Lepper, M. Oehler, H. Kinzler, and B. Trancón, "Diminuendo al bottom —- clarifying the semantics of music notation by re-modeling," *plosOne*, 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0224688

[2] M. A. Cara, "Anticipation awareness and visual monitoring in reading contemporary music," *Musicae Scientiae*, vol. 22, no. 3, pp. 322–343, 2018.

[3] R. Kopiez, C. Weihs, U. Ligges, and J. I. Lee, "Classification of high and low achievers in a music sight-reading task," *Psychology of Music*, vol. 34, no. 1, pp. 5–26, 2006.

[4] D. Schön, J. L. Anton, M. Roth, and M. Besson, "An fMRI study of music sight-reading," *Neuroreport*, vol. 13, no. 17, pp. 2285–2289, 2002.

[5] A. Stenberg and I. Cross, "White spaces, music notation and the facilitation of sight-reading," *Scientific reports*, vol. 9, no. 1, p. 5299, 2019.

[6] M. Puurtinen, "Eye on music reading: A methodological review of studies from 1994 to 2017," *Journal of Eye Movement Research*, vol. 11, no. 2, pp. 1–16, 2018.

[7] J. Kilian and H. H. Hoos, "Voice separation - A local optimization approach," in *ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France, October 13-17, 2002, Proceedings*, 2002. [Online]. Available: http://ismir2002.ismir.net/proceedings/02-FP01-6.pdf

[8] N. Guiomard-Kagan, M. Giraud, R. Groult, and F. Levé, "Comparing voice and stream segmentation algorithms," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, M. Müller and F. Wiering, Eds., 2015, pp. 493–499. [Online]. Available: http://ismir2015.uma.es/articles/180_Paper.pdf

[9] E. Chew and X. Wu, "Separating voices in polyphonic music: A contig mapping approach," in *Computer Music Modeling and Retrieval: Second International Symposium, CMMR 2004, Esbjerg, Denmark, May 26-29, 2004, Revised Papers*, ser. Lecture Notes in Computer Science, U. K. Wiil, Ed., vol. 3310. Springer, 2004, pp. 1–20. [Online]. Available: https://doi.org/10.1007/978-3-540-31807-1_1

[10] S. T. Madsen and G. Widmer, "Separating voices in MIDI," in *ISMIR 2006, 7th International Conference on Music Information Retrieval, Victoria, Canada, 8-12 October 2006, Proceedings*, 2006, pp. 57–60.

[11] H. von Werden, *Musica Enchiriadis*. Bamberg: Staatsbibliothek, 1000, vol. Var. 1, https://zendsbb.digitale-sammlungen.de/db/0000/sbb00000078/images/index.html?id=00000078&seite=115&bibl=sbb.

[12] W. Tappolet, *La notation musicale et son influence sur la pratique de la musique du moyen age a nos jours*. Neuchâtel: Ed. de la Baconnière, 1947.

[13] J. M. Spivey, *The Z Notation: A reference manual*, ser. International Series in Computer Science. Prentice Hall, 1988.

[14] J. S. Bach, *Die Kunst der Fuge BWV 1080*. Wiesbaden: Breitkopf und Härtel, 1924, *edt.* W. Graeser.

[15] ——, *Das Wohltemperierte Klavier I BWV 846-869*. Wien: Wiener Urtext Edition, 1977, *edt.* Walter Denhard.

[16] K. Hader, *Aus der Werkstatt eines Notenstechers*. Wien: Waldheim-Eberle, 1948.

[17] L. van Beethoven, *Klaviersonanten I*. München: Henle Verlag, 1952, *edt.* B.A. Wallner.

A. MATHEMATICAL NOTATION

The employed mathematical notation is fairly standard, inspired by the Z notation [13]. For leaner notation, we add some overloading. Important constructs are:

$\mathbb{P}\ A$	Power set, the type of all subsets of the set A, incl. infinites.
$a \setminus b$	The set containing all elements of a which are not in b.
$A \times B$	The product type of two sets A and B, i.e. all pairs $\{c = (a, b) \mid a \in A \wedge b \in B\}$.
π_n	The nth component of a tuple.
$A \to B$	The type of the *total* functions from A to B.
$A \nrightarrow B$	The type of the *partial* functions from A to B.
$A \leftrightarrow B$	The type of the relations between A and B.
$a \mapsto b$	An element of a relation; simply another way to write (a, b)
$\mathsf{dom}\ a, \mathsf{ran}\ a$	Domain and range of a function or relation.
$S \lhd R$	$= R \cap (S \times \mathsf{ran}\ R)$, i.e. domain restriction of a relation.
$R \rhd S$	$= R \cap (\mathsf{dom} R \times S)$, i.e. range restriction of a relation.
$R \rhd\!\!\!\!- S$	$= R \setminus (\mathsf{dom} R \times S)$, i.e. negative range restriction of a relation.
$f\ (\!\!|\ s\ |\!\!)$	The image of set s under function or relation f
r^{-1}	The inverse of a relation
$\mathbf{ID}_A$	$= \{a \in A \bullet (a \mapsto a)\}$, the identity relation.
$r \fatsemi s$	The composition of two relations: the smallest relation s.t. $a\ r\ b \wedge b\ s\ c \Rightarrow a\ (r \fatsemi s)\ c$. (first apply r, then apply s)
$r \oplus s$	Overriding of function or relation r by s. Pairs from r are shadowed by pairs from s: $r \oplus s = \big(r \setminus (\mathsf{dom} s \times \mathsf{ran} r)\big) \cup s$
$\mathsf{seq}\ A$	The type of finite sequences from elements of A i.e. maps $\mathbb{N}_1 \nrightarrow A$ with a contiguous range $\{1..n\}$ as its domain.
$\mathsf{iseq}\ A$	The type of injective finite sequences from elements of A
$squash(a)$	Turns any partial function $\mathbb{N}_1 \nrightarrow A$ into a $\mathsf{seq}\ A$ by compactifying the indices.
$last(a)$	The last element in a sequence
$\langle\rangle$	The empty sequence.
$\alpha \frown \beta$	Concatenation of two lists.
$\#a$	The magnitude of a set (=number of contained elements).

Functions are considered as special relations, i.e. sets of pairs, like in "$f \cup g$".