PITCHES IN BACH

Andrea Agostini
Conservatory of Turin
andreaagostini@bachproject.net

Daniele Ghisi
STMS Lab (IRCAM, CNRS, UPMC)
Conservatory of Genoa
danieleghisi@bachproject.net

ABSTRACT

Traditionally, most computer-aided composition environments represent a pitch via a number (typically a MIDI note number or its value in midicents), flattening the enharmonic information onto a single real-valued parameter. Although this choice is convenient in many applications, it can be very limiting in any context where diatonicism, to some degree, matters.

The latest release of *bach*, a library for Max dedicated to musical representation and computer-aided composition, introduces a new 'pitch' data type, designed to overcome this limitation by representing both diatonic pitches and intervals and supporting standard arithmetic operations. In this article we motivate and detail its implementation and its syntax.

As an application, we introduce a new respelling algorithm, also implemented in *bach*, designed to provide an easy-to-read spelling of notes. Differently from most existing pitch spelling algorithms, tailored on the tonal repertoire, our algorithm is targeted to produce a musician-friendly representation of non-tonal music.

1. INTRODUCTION

1.1 The problem

Virtually every software system capable of dealing with symbolic musical information has some kind of representation of pitch. Some tools for computer-aided composition, including OpenMusic [1] and PWGL [2], as well as versions of *bach* [3] prior to 0.8, employ MIDI note numbers or midicents, thus not providing a direct way to express enharmonic information: of course, even in these cases it is always possible to set up custom representations, but manipulating them would require the effort of constructing all the necessary tools. On the other hand, other software systems, such as Abjad [4] and Music21 [5], embed enharmonic information in their basic representation of pitches.

Both choices have their own advantages and disadvantages. Reducing pitch to a basic numeric type by eschewing enharmonic information simplifies the system: at the very least, it avoids the need for specific constructors and methods. In some regards, it can also make life easier for users, who do not need to become acquainted with a specific syntax and set of operations.

On the other hand, it is a very limiting choice, for a number of reasons. For one thing, the tuning of pitches depends on the chosen temperament—yet, we shall consider, in this article, only the case of equal temperaments. Even in this case, the choice of dropping enharmonic information is still inadequate, from at least two points of view: a technical one, because there may be good reasons (such as readability) for preferring one kind of enharmonic spelling to another; and a more strictly musical one, because such a representation is strongly connected to a non-hierarchical conception of musical pitches and the networks of significance they form within the musical discourse. After all, in a typical piece of music by Pierre Boulez, Franco Donatoni or even Anton von Webern, the choice of representing a given musical pitch as an F♯ rather than a G♭ is mostly irrelevant, to the point that several composers, Donatoni included, have made very limited use of accidentals other than the sharp. It is not by chance that the three aforementioned composers have a strong relationship with dodecaphony and serialism. On the contrary, a page by Bach or Mozart would be substantially wrong if typeset with all the F♯'s and G♭'s swapped. Moreover, although in most cases this information can be reconstructed, there are instances in which the enharmonic spelling chosen by the composer carries meaning useful to shed light on how a particular chord or passage should be interpreted [1]. A notable example is Richard Wagner's famous Tristan chord, which has been the subject of debate since more than a century: the analytical tools involved are meaningful only if they take enharmonic spelling into account, and the insight they provide is highly relevant to the understanding of late-19th century and early-20th century tonal music.

Several works and sub-genres of contemporary music fall somewhere between these two categories. Whereas music strictly adhering to the tonal system, as found in works by composers from the 18th and early 19th centuries, is now almost solely composed in the context of school exercises, the same cannot be said for music closer, or belonging, to the harmonic traditions of jazz, rock and pop [2]. On the other hand, the tonal syntax of concert music from the 19th and early 20th centuries still forms the harmonic basis for a wide array of contemporary, non-strictly-concert music,

[1] http://repmus.ircam.fr/openmusic/home
[2] http://www2.siba.fi/PWGL/
[3] www.bachproject.net
[4] http://abjad.mbrsi.org
[5] http://web.mit.edu/music21/

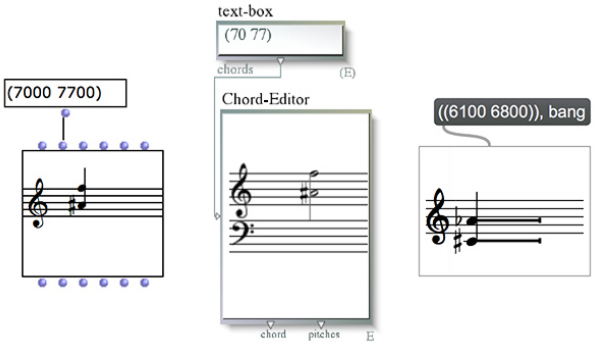

Figure 1. Display of MIDI notes corresponding to perfect fifths in some of the most common computer-aided composition environments (from left to right, OpenMusic, PWGL and *bach*). Each environment has somehow its own 'wolf fifth'.

most notably—but not exclusively—film music. Moreover, although self-described 'art music', roughly over the last century, has distanced itself from the received, historically connoted syntax of tonal language, by no means it has consistently renounced all forms of hierarchical syntax of pitches. This observation refers, in the first place, to various branches of so-called 'neomodal' music, a category that may be applied to works by composers as diverse as Terry Riley, Arvo Pärt and Louis Andriessen, or—more recently—Yannis Kyriakydes, Andrew Hamilton and Nico Mulhy. On the other hand, other sub-genres and individual works in the field of contemporary art music may be described as featuring hierarchical (albeit not modal) pitch structures, including works by composers influenced to various degrees by spectralism, such as Gérard Grisey and Kaija Saariaho, or works explicitly referencing other musical idioms, be they popular, folkloric or historical, such as *Sinfonia* by Luciano Berio, *Professor Bad Trip* by Fausto Romitelli or *Cognitive Consonance* by Christopher Trapani. In all these contexts, the question "Is this an F♯ or a G♭?" is not an idle one, because it alludes to the functional roles that pitches carry within the musical discourse. And anyway, even in more strictly serial or post-serial contexts, there is some sort of consensus on the 'correct' representation of intervals: for instance, it is uncommon to come across diminished sixths where perfect fifths could be used—something composers working with computer-aided composition tools have unfortunately been trained to tolerate (see Figure 1). Effective software tools for musical formalization should take all this into account. Therefore, our aim is to provide a formalization and an arithmetic of pitches in equal temperaments, and implement it in *bach*.

1.2 A proposed solution for Max and *bach*

Max has a very limited focus on symbolic musical representation, and objects that need to represent pitch do it according to the MIDI standard. The *bach* package for Max, conceived specifically to augment Max with advanced capabilities of representation and treatment of musical data [3] has been using midicents as its native way of representing pitches, too, coherently with its main original references (namely, OpenMusic and PWGL). This was also a convenient choice for easing the communication between *bach* objects and native Max objects, as the only conversion tool required was a division or a multiplication by 100, respectively for converting midicents into MIDI pitches, or viceversa. In the latest major version of *bach* (0.8), on the other hand, we felt that this simplistic representation was not adequate to the scope we envisioned. For this reason, we decided to implement in the *bach* system a new data type, aptly called a *pitch*, representing musical pitches and meant to be operated upon through both standard mathematical operators and new, specific tools.

2. REPRESENTATION OF PITCHES

The mathematics of pitch representation is a well-studied field, especially in the context of equal temperament. Although most techniques, influenced by the musical set theory, tend to flatten pitches onto their MIDI note numbers (to the point that nowadays the term 'pitch-class' commonly refers to MIDI note classes rather than diatonic pitch classes), there exist at least two families of approaches that preserve enharmonic information. Models in the the first family represent pitches as belonging to geometrical structures in space (such as the line of fifths, the Tonnetz [4], or the spiral array [6] [6]). Models in the the second family essentially represent pitches as couples $(c, d) \in \mathbf{Q} \times \mathbf{Z}$, where c is the number of chromatic steps or semitones from a reference note, such as middle C, and d is the number of diatonic steps from the same reference note [7, 8]. As an example, the F♯ just above middle C would be represented as $(6, 3)$, while its enharmonic equivalent, G♭, would be $(6, 4)$. Several variants of this representation exist (e.g. using midicents instead of semitones, or choosing C0 as reference note); we will refer to similar encodings as 'chromatic-diatonic couples'.

Both these families of representations have the advantage to make standard operations such as transposition or enharmonic respelling arithmetically trivial—at the expense of making other properties less readable. For example, it is not straightforward to infer the accidental of a pitch from either a spatial position inside a geometrical structure or a chromatic-diatonic couple.

The *bach* library takes advantage of both of these representations (the first one is used, for instance, in pitch respelling algorithms, whereas the second one is used to facilitate some arithmetic operations). However, we have decided to use internally a container whose fields mirror more directly the way we usually think of notes, that is, a degree, an alteration and an octave.

Several models for tridimensional representations of pitches have been proposed. Most of them involve quotienting by an operation of octave transposition, hence disentangling the octave number from a two-dimensional representation of a diatonic pitch-class.

Brinkman's 'binomial representation' [9], represents such diatonic pitch-classes as a combination of a 'MIDI pitch-

[6] The spiral array should not be confused with Shepard's helix [5], which does not distinguish enharmonic pitches.

class' (0 to 11) and 'letter-class' (0 to 6). Brinkman's representation is however not equivalent to chromatic-diatonic couples; namely, as the author recognizes, it has the inelegant disadvantage of allowing ambiguities when more than five accidentals are involved: a MIDI pitch-class of 6 together with a letter-class of 0 may correspond both to C sextuple sharp and to C sextuple flat.

Clement lays in [10] important groundwork concerning the relationship between pitches and intervals, namely asserting that all intervals, and hence all pitches, can be generated via combinations of a chromatic half-step and a diatonic half-step. However, Clement chooses to eventually flatten the pitch parameter onto a single integer, managing to distinguish quite well the most common enharmonic representations, yet still leaving room for ambiguities when larger alterations are involved. Also, Clement uses different names and grammars for pitches and intervals — a distinction that trained musicians usually take for granted but which, in our own view, is unnecessary (as the next section will detail).

Drawing from all these researches and considerations, we have decided to implement our own encoding of pitches in *bach*, as we explain in section 4.

3. ARITHMETIC

3.1 Pitches and intervals

When we say something seemingly trivial like "this is an E♭ at octave 4", we are superposing two kinds of reasoning: on the one hand, the general concept of 'E♭' is a shared, albeit slippery, one, and there is at least partial consensus about what 'octave 4' means.[7] On the other hand, without a reference pitch and tuning system, it is in principle impossible to assign an exact frequency (that is, an exact meaning with respect to sound) to 'E♭ at octave 4'. In this sense, we can say that the name of any musical pitch represents, strictly speaking, an interval with respect to a fixed reference within a certain tuning system. So, according to one of the most widespread practices, 'E♭ at octave 4' means "a tempered augmented fourth below the A4, the frequency of the latter being 440 Hz". In a context of purely symbolic computation, the relation to the exact frequency of a reference pitch may be irrelevant, but the substantial identification of absolute pitches and intervals is an elegant conceptual tool for simplifying the expression of transpositions and other operations.

[7] There are many possible specific definitions and interpretations of 'E♭', both formal (for example, the set of all the notes that can be obtained by stacking three descending fifths starting from a C) and informal (for example, a referral to the embodied cognition of the production of a generic E♭ on a musical instrument, sometimes coinciding with its enharmonic D♯), but most of them share enough common traits to allow both musicians and non-musician to talk practically about E♭'s without worrying about substantial misunderstandings. Octave numbering is usually a more technical matter, and in fact there are several conventions for distinguishing between different E♭'s in the audible range (and, potentially, beyond it). The arabic numeral after the note name is especially used in electronic instruments and music software. The most widespread convention appears to be the one setting C4 as the middle C, written with one ledger line below a treble clef staff and typically corresponding to a frequency of roughly 261.5 Hz (the case with transposing instruments requiring further specification). As we shall discuss below, we chose to adopt a different convention in this regard.

Another way to see this possibly confusing identification is that, on the one hand, we see the musical interval as an essentially spatial measure, and, as such, we typically use it in a relative way (we cannot say that Montreal is located *at 3000 km*, but rather that it is *3000 km away from Albuquerque*). On the other hand, we are somehow used to treat the nomenclature of pitches as just a set of names, not unlike what we do with colors, albeit a very formally defined one. What we are proposing here is that, considering the unambiguousness of pitch names and the trivial and biunivocal relation between absolute pitches and the interval of each pitch from C0, we can actually merge the two concepts and use a single naming scheme for both. This is not too different from what we do when we use Celsius degrees for both measuring the temperature difference between two bodies and expressing absolute temperatures as the distance between a body's temperature and the arbitrarily chosen reference of the water's melting point.

These considerations have informed two fundamental choices at the basis of the pitch representation system in *bach*: first, as hinted above, the same format and data type used for expressing pitches is also used for intervals with respect to a reference pitch of C0. Thus, E♭0 denotes both a very low E flat and an ascending minor third, whereas -F0 (and the equivalent form G-1) denotes a descending perfect fourth. A possibly more rigorous way to see this is considering the system from the point of view of intervals: E♭0 has "minor third" as its first meaning, and we can use it to denote an absolute pitch located a minor third above the C0 reference. This also explains the -F0 = G-1 identity: -F0, considered as an interval, denotes a downward perfect fourth; and a perfect fourth below the C0 reference is G-1. As a side note, *bach* accepts the two representations indifferently, but (since very low values are more often used to express intervals than absolute pitches) returns the 'interval' format, the one with an optional leading minus sign but only non-negative octaves, as the preferred format for textual representation; two objects, bach.write and bach.textout, provide options for returning the other format, potentially with negative octaves. The fact that C0 is the reference for absolute pitches as well as for intervals leads to the second consideration: because we want our system to retain backwards compatibility with *bach*'s previous, midicent-based system of representation of pitches, we now need transposition of pitches to behave consistently with transposition of midicents. This means that transposing a pitch by a minor third must be compatible with summing 300 midicents, which implies that E♭0 must be 300 midicents, C0 (the perfect unison, and the identity element for transposition) be 0 midicents, and C5 be 6000 midicents (that is, middle C). This is a different standard from the two most widely used (placing middle C at the beginning of octave 3 or 4), but there is at least one precedent in Cakewalk Sonar, and there used to be an additional one in older versions of Reaper.

The simplicity and elegance of this architecture have been the two important factors leading to our choice of C5 as middle C in *bach*. On the other hand, it is always possible to express pitch literals according to a different standard,

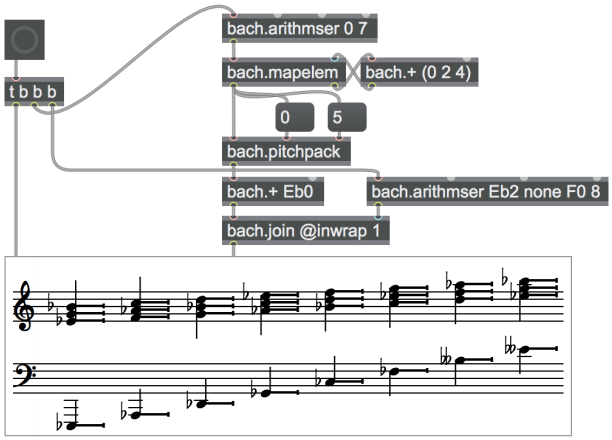

Figure 2. Pitch arithmetic addresses a whole area of diatonic, modal and tonal musical processes.

and applying to them a transposition of one or two octaves. We will hence assume throughout the rest of the article that C0 is MIDI note 0 (and hence C5 is middle C).

3.2 Operations

Algebraic sums and multiplications are meaningful on intervals: for example, a minor third plus a major third is a perfect fifth, and a perfect fifth minus a minor third (that is, plus a descending minor third) is a major third. Within our convention, we may write E♭0 + E0 = G0. This amounts to transposing any of the two pitches by the interval represented by the other one; for instance, summing any pitch to E♭0 will result in a transposition by a minor third (see Figure 2). C0 (unison) is the identity element for the sum.

We have not been able to assign a musical meaning to the multiplication of two intervals in the pitch domain. [8] However, there is a natural external multiplication of an interval by an integer, which can simply be seen as a sequence of sums, with a sign depending on the signs of the factors. As an example, $12 \cdot$ G0 = B♯6. A multiplication by -1 inverts the interval; as previously stated, pitches lower than C0 can be expressed either with a negative octave or with a negative interval (for example, $-1 \cdot$ E♭0 = −E♭0 = A-1).

All the above operations are unambiguous with respect to enharmonicity. Partitive division of an interval by an integer, on the other hand, is generally problematic: what does it mean to divide an augmented fourth by two? The difficulty here arises from the fact that we wish our operations to be meaningful with respect to enharmonic spelling. So, although an augmented fourth is 6 semitones wide, and as such dividing it by two would result in 3 semitones, there are theoretically infinite intervals spanning 3 semitones (minor third, augmented second, doubly diminished fourth, etc.), but none of those, if multiplied by two, will return an augmented fourth. For example, a minor third times two is a diminished fifth, and an augmented second times two is a doubly augmented third. On the other hand, by performing an integer division on the 0-based degree and the integer, and subsequently adjusting correctly the accidental and/or alteration, it is possible to obtain a pitch quotient spanning the correct amount of semitones, or fraction thereof. This pitch quotient, if multiplied back by the original divisor, is an interval possibly different from the original dividend, but enharmonic to it. The difference between the divisor and the product of the dividend and the quotient is the remainder of the division, and it always spans 0 semitones—that is, it is always enharmonic to a perfect unison. So, an augmented fourth divided by two is an augmented second, with a remainder of a diminished second (because an augmented second times two is a doubly augmented third, and an augmented fourth minus a doubly augmented third is just a diminished second). In our pitch syntax: F♯0/2 = D♯0 with the remainder of D♭♭0, because $2 \cdot$ D♯0 + D♭♭0 = F♯0.

Quotative division of an interval by an interval is also possible. It involves promotion of the two terms to midicents, and returns an integer or a rational number. If the second term is C0, the division is indeterminate. The remainder of the quotative division is simply defined as the difference between the divisor and the product of the dividend and the quotient: since the dividend is a pitch and the quotient is an integer, their product is also a pitch and the aforementioned difference is also a pitch. For instance, A1/G0 = 3 with no remainder, while E♯2/G0 = 4 with a remainder of C♯0.

3.3 Comparisons

Comparisons among pitches can also be expressed: given two pitches A and B, we say that $A = B$ iff their degrees, alterations and octaves are the same. Thus, C♯5 is different from D♭5, even if their midicents are the same. In this sense, and differently from what happens (not considering the limitations of numerical representation) when promoting an integer to a float, promoting pitches to rationals may change the result of an equality comparison performed upon them. Moreover, the 'less than' comparison operates lexicographically: $A < B$ if the octave of A is less than the octave of B, or, in case they coincide, if the degree of A is less than the degree of B, or, in case they also coincide, if the alteration of A is less than the alteration of B. Again, an inequality comparison performed on pitches can lead to the opposite result of the same inequality performed upon the midicents of those pitches: for example, B♯4 < C♭5 and E♯5 < F♭5.

These choices have been the subject of careful consideration, and have not been taken lightly. The main reason to choose these seemingly incoherent behaviors as the default is to preserve the richness of the pitch semantics (using the midicents ordering as 'less than or equal to' criterion would imply that all enharmonic spellings are equal). After all, it is straightforward to implement the 'other' behavior (the one according to which B♯4 > C♭5 and E♯5 > F♭5) when needed: all it takes is forcing the conversion to midicents, something *bach* provides various simple options for. All this being said, we are well aware that the

[8] For the sake of clarity, it may be worth recalling that multiplying a frequency by an interval as defined in the frequency domain (that is, the ratio between two frequencies) is perfectly meaningful and corresponds to an equal temperament transposition in the frequency domain. This operation is completely distinct from the meaningless multiplication of two intervals in the pitch domain.

answer most musicians would give to the question "Which is higher, B♯4 or C♭5?" would probably be the opposite of what our system gives.

3.4 Chromatic-diatonic representation

All the aforementioned choices are expressed more concisely using the chromatic-diatonic representation of pitches. Let $A = (c_A, d_A)$ and $B = (c_B, d_B)$ be two pitches such that c_i and d_i are respectively the number of semitones and the number of diatonic steps from the reference point C0 (with midicents 0). Then $A + B := (c_A + c_B, d_A + d_B)$ is the transposition operation, $-A := (-c_A, -d_A)$ is the inversion operation, $n \cdot A := (n \cdot c_A, n \cdot d_A)$ is the multiplication of a pitch by a number $n \in \mathbf{Z}$ (the set of pitches is thus a $\mathbf{Z}$-module). Partitive division is $A/n := (c_A/n, \lfloor d_A/n \rfloor)$ with remainder of $(0, d_A - n\lfloor d_A/n \rfloor)$, enharmonic to C0; quotitive division is $A/B := (\lfloor c_A/c_B \rfloor, \lfloor d_A/d_B \rfloor)$ with remainder of $(c_A - c_B\lfloor c_A/c_B \rfloor, d_A - d_B\lfloor d_A/d_B \rfloor)$.

The standard lexicographical order is defined on pitches: $A \leq B \Leftrightarrow d_A < d_B \vee (d_A = d_B \wedge c_A \leq c_B)$. Any $(c_A, d_A + k), \forall k \in \mathbf{Z}$ is an enharmonic respelling of A.

4. THE BACH IMPLEMENTATION

A pitch in *bach* is a triplet (g, a, o), where $g \in \mathbf{Z}/7\mathbf{Z}$ is the degree, $a \in \mathbf{Q}$ is the alteration (in fraction of tone) and $o \in \mathbf{Z}$ is the octave. In the internal representation, the degree is a number from 0 to 6, representing white keys names from C to B; the alteration is a rational number [9]; and the octave is an integer, with octave 5 starting with middle C (corresponding to the MIDI pitch 60), and subsequently octave 0 starting with MIDI pitch 0.

Conversions between this chromatic-diatonic representation (c, d) and *bach*'s encoding of pitches as triplets (g, a, o) of degree, alteration and octave are straightforward:

$$\begin{cases} c &= \text{deg2chr}(g) + 2a + 12o \\ d &= g + 7o \end{cases}$$

and

$$\begin{cases} g &= [d]_7 \\ o &= \lfloor d/7 \rfloor \\ a &= \dfrac{c - 12o - \text{deg2chr}([d]_7)}{2} \end{cases}$$

with deg2chr: $\mathbf{Z}/7\mathbf{Z} \to \mathbf{Z}$ mapping $[0]_7 \mapsto 0, [1]_7 \mapsto 2, [2]_7 \mapsto 4, [3]_7 \mapsto 5, [4]_7 \mapsto 7, [5]_7 \mapsto 9, [6]_7 \mapsto 11$.

A pitch, according to the above definition, is stored in a double word, according to the computer architecture in use. Under a 32-bit architecture, a pitch is stored in 8 bytes (64 bits): 2 bytes for the degree (which of course is overkill, since its value is limited to the 0-6 range), 2 bytes for the octave (hence limited to the enormous range -32768 to 32767), and 4 bytes for the alteration (2 bytes for the numerator and 2 for the denominator, allowing for an extremely precise representation). Under a 64-bit architecture, a pitch is stored in 16 bytes (128 bits), thus doubling the size of all its fields with respect to the above.

[9] Rational numbers and arithmetic operations upon them are introduced in Max by *bach*.

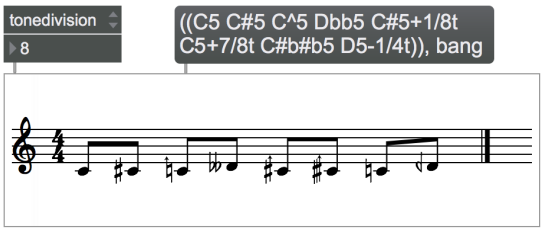

Figure 3. Some examples of pitch syntax in *bach*.

There is no explicit concept of a 'pitch constructor' in *bach*: the simplest way to construct a pitch is just typing its textual representation into a message object and passing it to a *bach* object. The textual syntax of a pitch is structured as follows (brackets delimit optional elements):

$$\big(\pm\big)\langle\text{degree}\rangle\big[\langle\text{accidental}\rangle\big]\langle\text{octave}\rangle\big[\pm\langle\text{alteration}\rangle\text{t}\big]$$

where the degree is a letter corresponding to an Anglo-saxon note name (from A to G); the accidental is a combination of the characters # (sharp), b (flat), x (double sharp), q (quartertone sharp), d (quartertone flat), ^ (eighth-tone sharp), v (eighth-tone flat), whose values are summed together; the octave is a positive or negative integer; and the alteration is a signed integer or rational number expressing a deviation in tones (or a fraction thereof) from the pitch as defined by the degree / accidental / octave representation. Both the accidental and the alteration are optional, but the degree and the octave must always be present (for instance C is not a pitch). The leading unary minus or plus is also optional: the plus sign has no effect, whereas the meaning of the unary minus flips the interval direction, as explained above. Examples of properly formatted pitches, as typed into a message box, are: C0, D#3, E-1, Fbbb6 (an F triple-flat at octave 6), Abv5 (an A flat minus an eighth tone at octave 5), B5-1/2t (a B minus a half tone, equivalent to a B♭5), C♯4+1/10t (a C sharp plus one tenth of a tone). Also see Figure 3 for an illustration.

The same representation is essentially used when a pitch is returned as text. As hinted at above, the same pitch can be represented through several representations: for example, B5-1/2t and Bb5 represent the same pitch, and the same goes for C#v3 and Cq^3. It is also possible to invent 'absurd' representations, such as C#b#b2 for C2, or Dvvvv4 for Db4. In principle, for each pitch there are infinite representations. Among those, each pitch has a 'normal form', that is, the representation with the shortest combination of same-direction accidental signs and the alteration with the smallest absolute value (or, if possible, no alteration at all: accidentals are preferred over alteration).

The musical notation editors of *bach* (namely, the bach.roll and bach.score objects) are now capable to accept pitches as their input. [10] Mathematical expression among pitches

[10] This is not completely new, as in previous versions of *bach* there was a way to assign a specific enharmonic spelling to a note, but it was a cumbersome one: besides entering the pitch in midicents, it was (and, for backwards compatibility sake, still is) possible to specify that the graphical representation of the note was composed by a given 'white key' pitch and a given alteration. There was even a sort of 'shortcut' for this, in that, by entering, say, 'Db4', the appropriate pitch and graphical information

can be evaluated via the usual *bach* arithmetic modules [11] : the *bach* evaluator can now perform operations on pitches, just like it does with regular numerical types, following the explanation provided in section 3.2. In order to handle results of indeterminate operations (such as divisions by C0), a special NaP ('not a pitch') value is returned. In addition to the set of functions explicitly supporting pitches, any mathematical operator and function can accept pitches, which are implicitly promoted to integer, rational or floating-point midicents and operated upon as such: for instance, calculating the square root of E8, corresponding to 10000 midicents, returns the floating point value 100., as the sqrt function only operates upon floats, and promotes to a float all the other number types.

5. PITCH SPELLING ALGORITHMS

Finding the best possible spelling for sequences of notes and chords is far from a trivial issue, requiring knowledge of the musical context as well as computation time—which is why essentially all computer-aided composition environments tolerate default awkward spellings such as the ones in Figure 1. A certain number of pitch spelling algorithms have been proposed in the last few decades [11], aiming at finding, to some respect, the 'best' spelling of notes, given their MIDI numbers, onsets and durations.

In the new *bach* release, both bach.roll and bach.score feature three pitch spelling algorithms, triggered via a 'respell' message:

- a trivial algorithm, providing automatic note-by-note respelling, without any context or memory. For each step in the semitonal (or microtonal) scale, a 'standard' enharmonic representation is used. Either such representation is provided by the user (via an enharmonic spelling table), or a hard-coded choice, depending on the current key signature, is used;
- the algorithm proposed by Chew and Chen [12] in [12], based on Chew's spiral representation of pitches [6];
- a new 'atonal' algorithm, described in section 5.1.

Each of these algorithms can operate either voice by voice (so as to provide consistent readability for single, specific voices) or globally (so as to provide diatonic consistence across different voices). They can also limit their scope to subsets of the line of fifths (see Figure 4), by defining a 'sharpest' and/or a 'flattest' representable pitch (Figure 6).

5.1 General outline of the atonal algorithm

Although it is true that pitch spelling is imperative in tonal music (as stated in the introduction, an F♯ might be substantially wrong inside a piece in G♭ major), it also plays a crucial role in the portion of non-tonal music where diatonicism has some importance. And yet, all the pitch spelling algorithms compared in [11] are essentially designed to work with tonal musical data, and they are hence only compared on historic tonal works. An important part of their workings deal with detecting harmonic modulations as precisely as possible.

The algorithm we propose is not tailored for this purpose—which is also why any comparison with the existing algorithms would be meaningless—but is rather meant to make general, non-tonal sequences of notes and chords 'as readable as possible' for musicians. In this context, detecting the precise position of a modulation is not a concern, whereas it is decisive to provide the players with a simple-to-read spelling for sequences of notes. We have developed our 'atonal' pitch spelling algorithm with these considerations in mind. As a side note, it should be remarked that the atonal algorithm can of course be applied to portions of modal and tonal music—which is why key signatures are also accounted for.

The idea at the basis of the atonal respelling algorithm is that notes that are close in time should be transcribed with pitches as close as possible on the line of fifths. [13] Therefore, the general outline of the algorithm is the following:

1. The notes belonging to the voice to be respelled (or to the entire score, depending on the chosen operation mode) are rearranged in a tree data structure, so as to reveal the proximity of notes in time. More specifically, the tree is structured so as to allow being traversed as follows: the couple of notes that are closest in time in the original voice or score (let us call them A and B) is encountered and evaluated first, thus forming a "core couple"; then the note closest to the previous pair is encountered, thus allowing it to be evaluated alongside A and B; and so on. In this way, increasingly large temporal windows of the original voice or score are taken into account. If, at any point, two notes not having been considered yet are closer to each other than either of them is closer to the current window, then the current window is put aside and the two new notes are considered as a new core couple, and the process moves forward from there. Further on, the new window may grow large enough to enclose the previous one, and in any case at the end of the process a single window containing the whole voice or score will be formed.
2. The rearranged tree is traversed according to the pattern described above, and over each step of the traversal the line-of-fifths distance of the pitches contained in the currently evaluated window is minimized, by searching for the combination of enharmonic representations with the smallest line-of-fifths standard deviation while respecting some ancillary constraints. If such standard deviation is within a given range, then the new enharmonic spelling is accepted, otherwise the algorithm settles upon the previously found

was automatically set. On the other hand, this kind of representation did not allow to perform arithmetic operations on pitches, and the extra information made the structure of the score more complex and less readable.

[11] In the actual implementation, integer division is performed towards zero and the remainder has the sign of the dividend, mirroring the behavior of the corresponding C functions.

[12] The algorithm was chosen based on [11], also considering the fact that Meredith's pitch spelling algorithms are subjected to patents.

[13] Following [12], a version with a spiral array representation had also been tested, to replace the line of fifths, with no significant improvement.

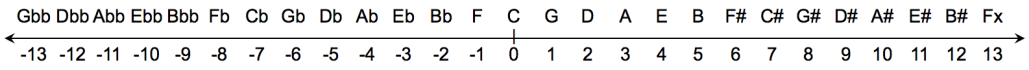

Figure 4. The line of fifths.

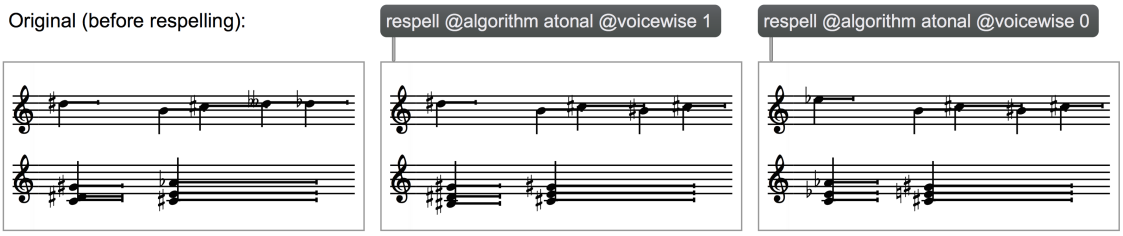

Figure 5. Voicewise versus non-voicewise respelling.

one and the traversal skips to the next core couple. The process goes on until there are core couples to be found.

A more detailed description of the algorithm, providing all the details needed for reimplementing it, and a practical example, are given below.

5.2 Detailed description of the atonal algorithm

A detailed description of the atonal respelling algorithm follows:

1. Respell the notes one by one via the aforementioned 'trivial' algorithm, providing a first rough spelling to be refined. This guarantees stability, since the rough spelling does not depend in any way from the original enharmonies, but only on the MIDI numbers. For instance, in a context with no key signature, spelling of portions of melodies in A♯ major, or in C♭♭ major, would be all equally respelled in B♭ major.

2. Build a list with all the notes of the voice, if the algorithm operates in a voice-wise fashion, or of the entire score otherwise. Chords are unpacked into notes with the same onset. At this stage, the list is flat; the next steps will reshape it into a tree, adding hierarchical levels (parens levels in bach *llll*s). Each node of the list contains some metadata, namely a 'starting time' s, an 'ending time' e (which at this stage both coincide with the onset of each specific note [14]) and a 'number of notes' n (at this stage $n = 1$).

3. Reshape the list constructed at point 2 into a tree, in the following way:

   3a. If the root level has a single node, then jump to step 4; otherwise find the closest nodes in the root level of the note list, i.e., find the two nodes such that the ending time of the first is closest to the starting time of the second. If there is a tie, take the first couple in temporal order.

   3b. Wrap the two nodes found in 3a in a new level (i.e., add a hierarchical node). If (s_L, e_L, n_L) and (s_R, e_R, n_R) are the metadata, respectively, of the earliest (left) and latest (right) node, then set the metadata of the new node to $(s_L, e_R, n_L + n_R)$.

   3c. Go to step 3a.

4. Perform the actual respelling. Obtain the list of nodes of the constructed tree via reversed breadth-first search and traverse it (deepest nodes are processed first). Process each node in the following way:

   4a. Let n be the number of notes of the node and $\mathcal{M} = (m_1, \ldots, m_n)$ be the list of MIDI numbers of the notes of the node. Also let $\mathcal{K} = (k_1, \ldots, k_n)$ be the key signatures of the voices to which the notes belong, and let $\mu_{\mathcal{K}}$ be the average of such signatures. Each $k_i \in \mathbf{Z}$ represents the number of sharps (if positive) or flats (if negative) of the key. If a node has a single note ($n = 1$), do nothing and jump to processing the next node. Otherwise continue to 4b.

   4b. Obtain the list of enharmonic possibilities for each $m_i \in \mathcal{M}$, in the form of an integer number (the position on the line of fifths, Figure 4) accounting for the 'sharpest' and 'flattest' parameters. Suppose that m_i has p_i enharmonic possibilities: let $\mathcal{C}_i = \{c_{i,1}, \ldots, c_{i,p_i}\}$ be the set of such numbers, $c_{i,j} \in \mathbf{Z}$. Let

   $$\mathcal{C} = \bigcup_i \mathcal{C}_i = \{c_{1,1}, \ldots, c_{1,p_1}, c_{2,1}, \ldots, c_{2,p_2}, \ldots, c_{n,1}, \ldots, c_{n,p_n}\}$$

   be the collection of the enharmonic possibilities for each note.

   4b1. Consider each one of $c_{i,j} \in \mathcal{C}$ as a candidate 'center of effect' on the line of fifth, and respell each element of $\mathcal{M}$ so that its position on the line of fifths is as close as possible to $c_{i,j}$. Let

   $$\mathcal{S}_{c_{i,j}} = (s_{c_{i,j},1}, \ldots, s_{c_{i,j},n})$$

   be the array of respelled positions, $s_{c_{i,j},k} \in \mathbf{Z}$.

[14] Notice that we call 'ending time' the largest note onset inside the hierarchical level, hence not accounting for note durations.

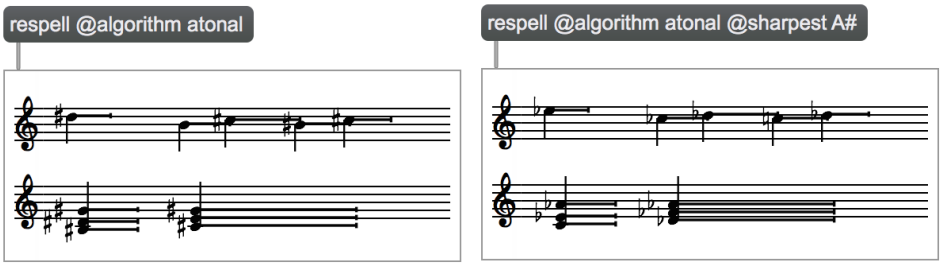

Figure 6. Defining 'sharpest' or 'flattest' notes has a global influence on the spelling algorithm.

4b2. Get the average $\mu_{c_{i,j}}$ and the standard deviation $\sigma_{c_{i,j}}$ of the $s_{c_{i,j},k}$'s. Normalize $\mu_{c_{i,j}}$ by subtracting the average of the key signatures $\mu_{\mathcal{K}}$ and add an additional bias, by default set to -2, accounting for the fact that first flat note appears at -2 on the line of fifths, while the first sharp note appears at 6 (flat and sharp notes are hence equally distant from the origin).

4b3. Determine whether the respelling $\mathcal{S}_{c_{i,j}}$ is acceptable. Only three conditions would make a respelling not acceptable:

- if any note is sharpest than the 'sharpest' acceptable or flattest than the 'flattest' acceptable;
- if altered repetitions (such as the sequence E♭-E♮) appear in $\mathcal{S}_{c_{i,j}}$ — but only in case a specific parameter to discard altered repetitions of the same pitch is set.
- if the standard deviation $\sigma_{c_{i,j}}$ is above a certain threshold $\widetilde{\sigma}$ (the threshold is a user-definable formula, defaulting to $\widetilde{\sigma} = \frac{21}{n+1}$). In other words, by default the threshold decreases as the number of notes of the set $\mathcal{M}$ increases.

If the $\mathcal{S}_{c_{i,j}}$ is not acceptable move to 4b5.

4b4. Determine if $\mathcal{S}_{c_{i,j}}$ is the 'best spelling' so far, i.e., the one having the smallest $\sigma_{c_{i,j}}$. In a tie, the spelling with the smallest $|\mu_{c_{i,j}}|$ is retained. If $\mathcal{S}_{c_{i,j}}$ is the best spelling, keep it as candidate.

4b5. Test the next possible candidate 'center of effect', i.e., go back to point 4b1 and move to testing $c_{i,j+1}$, or, if $j+1 > p_i$ then move to the element $c_{i+1,1}$; if $i+1 > n$, i.e., if all c_i's have been tried, move to 4c.

4c Once all $c_{i,j}$'s have been tested, there may or may not be a candidate for the respelling.

If there is no candidate, the node cannot be respelled, and all nodes containing it in the list of point 4 are dropped from the search.

If there is a candidate $\mathcal{S}_{\widetilde{c_{i,j}}}$, perform the respell of all notes according to it.

4d Jump back to point 4a and continue with the next node, until all nodes are completed.

This algorithm roughly provides a natural-to-read respelling of group of notes.

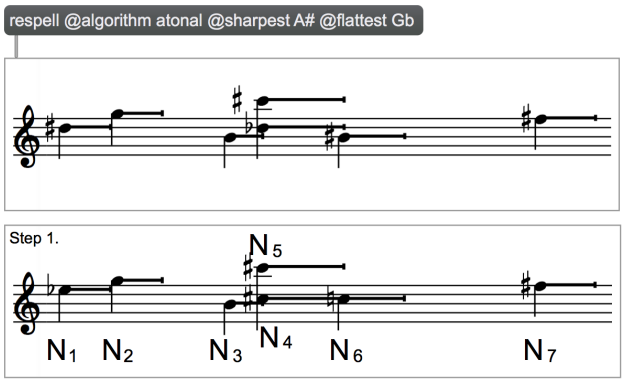

Figure 7. A simple example as a test for the algorithm.

5.3 An example case

To follow the behavior of the algorithm in a simple, concrete case, consider the score in Figure 7 and let $N_1, \ldots, N_7$ be the notes to be respelled. As per step 1, we respell each note with standard enharmonic tables. Then, as per step 2, we obtain the list of individual notes N_i, and via step 3. we arrange it in tree form according to their distances. Since the two notes forming a chord are the nearest ones (according to their onsets), they will be the first to be wrapped in a level, yielding $N_1N_2N_3(N_4N_5)N_6N_7$. Then, the two nearest nodes are the note N_3 and the node (N_4N_5), hence we wrap them yielding $N_1N_2(N_3(N_4N_5))N_6$. We repeat the process, until we have a single node at the root level of the list, yielding the list displayed in Figure 8.

Once the tree is constructed, we apply step 4 and build the list of nodes to be visited, in reversed breadth-first search. This list is (due to 4a, we can drop the final nodes having a single note): (N_4N_5), $(N_3(N_4N_5))$, $((N_3(N_4N_5))N_6)$, (N_1N_2), $((N_1N_2)((N_3(N_4N_5))N_6))$, $(((N_1N_2)((N_3(N_4N_5))N_6))N_7)$.

We start with (N_4N_5). The set of possible positions on the line of fifths for each note is $\mathcal{C} = \{7, -5\}$, representing a C♯ and a D♭. No other options are possible, given our choice of 'sharpest' and 'flattest' pitches. Since N_4 and N_5 are the same note, $\sigma_7 = \sigma_{-5} = 0$, while $\mu_7 = 7 - 2 = 5$ and $\mu_{-5} = -5 - 2 = -7$, given a bias of 2. We accept $c_1 = 7$ as center of effect, and spell both notes as C♯.

We move to $(N_3(N_4N_5))$. The set of possible positions on the line of fifths is $\mathcal{C} = \{7, -5, 5\}$, corresponding to C♯, D♭ and B (no other enharmonic option is possible for

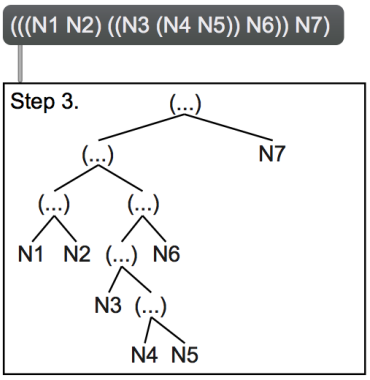

Figure 8. Tree of notes obtained after step 3.

B, given our choice of 'sharpest' and 'flattest' pitches). Again: $\sigma_7 = 0.94$, $\sigma_{-5} = 4.71$, $\sigma_5 = 0.94$, hence we choose $c_1 = 7$ as center of effect (since $\sigma_7 = 0.94 < 21/(3+1) = 5.25$ the solution is acceptable), and spell N_4 as B and both N_4, N_5 as C♯.

We move to $((N_3(N_4N_5))N_6)$, with $\mathcal{C} = \{7, -5, 5, 0\}$, corresponding to C♯, D♭, B and C. Using $c_i = 7$ would respell the consecutive notes N_5 as C♯ and N_6 as C♮, which is unacceptable if (as is by default) we choose to discard altered repetitions. The best acceptable scenario is hence $c_i = -5$, $\sigma_{-5} = 4.15 < 21/(4+1) = 4.2$. We hence respell N_3 as B, both N_4 and N_5 as D♭ and N_6 as C.

We move to (N_1N_2), with $\mathcal{C} = \{9, -3, 1\}$, corresponding to D♯, E♭ and G. The best solution is $c_2 = -3$, with $\sigma_2 = 2 < 21/(2+1)$ yielding N_1 as E♭ and N_2 as G.

We move to $((N_1N_2)((N_3(N_4N_5))N_6))$, with $\mathcal{C} = \{9, -3, 1, 7, -5, 5, 0\}$, which on the other hand has no acceptable solutions, either because of the altered repetitions, or because the standard deviations being greater than $21/(6+1) = 3$. We do not respell this node, and we also delete from the list all nodes containing this one, i.e., the node $(((N_1N_2)((N_3(N_4N_5))N_6))N_7)$. This concludes our process (the final result is displayed in Figure 9).

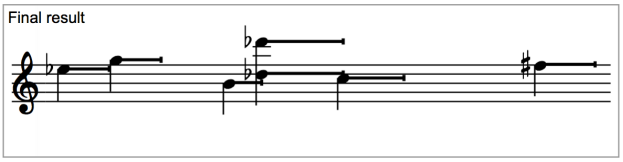

Figure 9. Final result.

5.4 Final considerations

The algorithm works for both bach.roll and bach.score and depends on the standard deviation threshold $\widetilde{\sigma}$. Such threshold can be set by the user, as shown in Figure 10. Higher values (or equations yielding higher values) for $\widetilde{\sigma}$ will allow respelling of larger temporal windows, at the expense of the quality of the transcription on smaller temporal windows (and at the expense of computation time).

Parameters for the $\widetilde{\sigma}$ function are 'numnotes' (the number of notes in the node to be respelled) and 'extension' (the temporal extension of the node in milliseconds). Among other things, one can fix spelling of chords only (as the ones in Figure 1) by providing a sufficiently high value for $\widetilde{\sigma}$ when the extension is 0, and a 0 value otherwise, e.g. $\widetilde{\sigma} = 1000000 * (\text{extension} == 0)$.

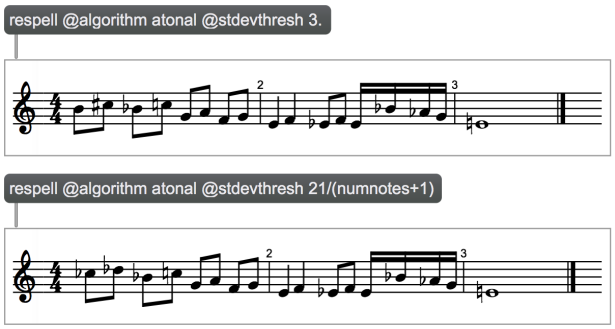

Figure 10. Different thresholds for $\widetilde{\sigma}$ affect the outcome.

Although the described algorithm provides a roughly natural respelling of general diatonic material, it also has two important shortcomings. For one thing, it is computationally expensive; notice, for instance, how respelling is performed multiple times on the notes N_4 and N_5 in the example above. The algorithm has been tailored for small portions of raw material and for short scores; as a consequences, for medium or large scores, the algorithm is essentially unusable in real time. To mitigate this issue, one can, however, adapt the equation for $\widetilde{\sigma}$ to only account for time extensions up to a certain threshold. In addition, given that the algorithm is based on a representation of diatonicism related to the line of fifths, it extends poorly on microtonal scenarios. The extension to microtonal music is of little concern for algorithms tailored on tonal music, such as the one by Chew and Chen; however, in our case, the possibility to improve the readability of microtones may constitute an important topic for future research.

6. CONCLUSION

We have presented a new framework for pitch representation in the *bach* library for Max, whose defining features are the ability to represent pitches with full enharmonic information, and the identification of pitches and intervals, meant to simplify and generalise the expression of arithmetic operations upon them. We also have described a novel algorithm for pitch respelling in the context of non-tonal music. We are aware of the fact that some aspects of this new system (in particular, the representation of intervals) might appear somehow confusing at first sight, but we hope that the simplification and generalisation they afford will outweight the initial difficulty, and that, overall, they will prove useful for implementing meaningful musical processes in a more straightforward and correct way than what the previous versions of *bach*, as well as other software tools, allow.

7. REFERENCES

[1] R. T. Kelley, "Reconciling tonal conflicts: Mod-7 transformations in chromatic music." [Online]. Available: http://www.robertkelleyphd.com/mod7.htm

[2] P. Tagg, *Everyday Tonality II*. The Mass Media Scholars Press, Inc., 2014-2016.

[3] A. Agostini and D. Ghisi, "Real-time computer-aided composition with *bach*," *Contemporary Music Review*, vol. 32, no. 1, pp. 41–48, 2013.

[4] R. Cohn, "Introduction to Neo-Riemannian Theory: A Survey and a Historical Perspective," *Journal of Music Theory*, vol. 42, no. 2, pp. 167–180, 1998.

[5] R. N. Shepard, "Circularity in judgements of relative pitch," *The Journal of the Acoustical Society of America*, vol. 36, no. 12, pp. 2346–2353, 1964.

[6] E. Chew, "Towards a mathematical model of tonality," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.

[7] E. Agmon, "A Mathematical Model of the Diatonic System," *Journal of Music Theory*, vol. 33, no. 1, pp. 1–25, 1989.

[8] R. T. Kelley, "A mathematical model of tonal function," in *Annual Meeting of Music Theory Southeast*, Chapel Hill, NC, USA, 2006.

[9] A. R. Brinkman, *PASCAL Programming for Music Research*. University of Chicago Press, 1990.

[10] P. J. Clements, "A System for the Complete Enharmonic Encoding of Musical Pitches and Intervals," in *Proceedings of the International Computer Music Conference (ICMC)*, Den Haag, Netherlands, 1986.

[11] D. Meredith and G. A. Wiggins, "Comparing Pitch Spelling Algorithms," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, London, UK, 2005.

[12] E. Chew and Y.-C. Chen, "Determining context-defining windows: Pitch spelling using the spiral array," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, Baltimore, USA, 2003.