Some thoughts about note−duration representation in the ABC language

1. Introduction

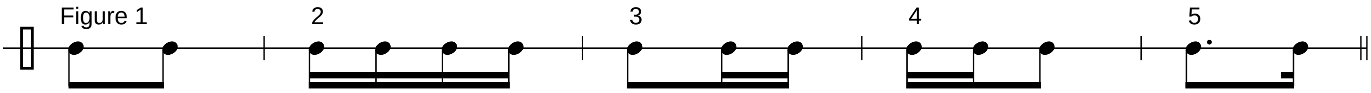
Music notation languages as ABC (abcnotation.com), Lilypond (lilypond.org) and Guido (www.salieri.org/GUIDO/) represent notes in a name−pitch−duration format. Comparison leads to strong favoring of the ABC−language, although this is personal. However, the ABC note−duration representations can be improved. More specifically, it could be closer to the principles of standard musical notation and to Steve Krug´s law of communication exchange, which states ´don´t make me think´. This is important in fast note entry and code reading. In accordance with this I have constructed my own version of the ABC−language. Purely for the sake of clarity, I refer to it as the ´MCM−language´. I didn´t, however, conceptualize the language, Chris Walshaw did, for which he deserves all credit. The ´MCM−language´ should be seen as a slight, but in my opinion, essential, modification of the ABC−language (www.mcmusiceditor.com).

I will share my ideas with you in the hope that it will inspire the ABC−community and ABC−software developers (thanks to Guido Gonzato − abcplus.sourceforge.net − for his suggestion).

2. Patterns in musical education

One goal of musical education is to develop the memory of pupils, who must cultivate the ability to ´think´ sounds (rhythm or pitch or both) without reproducing them aloud. When proficient, the pupil will have learnt to link up musical notation with its sound. And vice versa: the pupil should be able to associate sounds to an image of note values.

For an efficient and effective rhythm study we present pupils patterns that frequently recur in music. In figure 1 I show some of these basic patterns that pupils have to learn in a subsequent order.



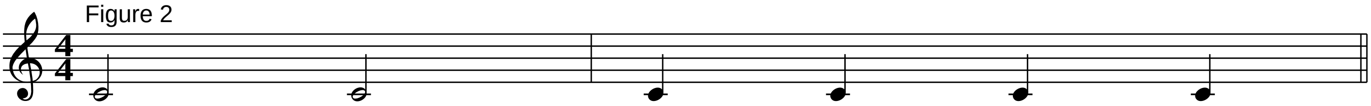
To perform a rhythm accurately, you have to recognize the rhythmic patterns. That recognition can be the result of the following process: matching the notated patterns against a mental inventory of previous learned patterns. The more patterns you have experience with, the less **computationally intensive** and error−prone your sight−reading and thus your performance will be.

Crucial to good sight reading is to grasp these groups, these patterns, as meaningful entities. The musical meaning of a pattern is one of the chief factors determining its difficulty.

In contrast to Lilypond and Guido, grouping of notes, i.e. making rhythm patterns visible in code, is in the ABC−language possible (e.g. CCCC or CC2C2). However, there is one problem: the ABC rhythm definitions of groups is not according our standard musical reading practice.

3. ABC musical notation

The ABC language has in case of note−duration representation a proportional system. The variable tag L: has to be defined with a basic note value, to which all other values are related. If we choose L:1/4 the whole note C is represented as C4 (because the duration of a whole note is 4 times the duration of a quarter note), the eighth note C/2 (because the duration of an eighth note is 1/2 times the duration of a quarter note) etc. A music piece can thus be coded in several ways depending on the choosen L−value. Study the following and try to write it in ABC−code.



The notes of figure 2 can in the ABC language be written in different ways. I gave three code solutions below, but note that still more are possible.

```
X:1
M:4/4
L:1/4
K:C clef=treble
C2 C2 | C C C C ||

X:2
M:4/4
L:1/2
K:C clef=treble
C C | C/2 C/2 C/2 C/2 ||

X:3
M:4/4
L:1/16
K:C clef=treble
C8 C8 | C4 C4 C4 C4 ||
```

This flexible way of coding has only in a limited number of cases advantages. In fact it leads to the main disadvantage of the ABC language:

1. it implies a disconnection to the practice of music notation (chapter 4 and 5)
2. it implies a disconnection to efficient and effective human information management, in our case music reading, especially recognition and memorizing of rhythmic groups (chapter 6)

4. From ABC−code to visual output

Suppose the following ABC−code

```
C3 C  C//2 C//2 C//2 C//2 ||
```

What does this mean? What is the connection with our musical notation? To answer these questions, more information is necessary. If the L−value is given, the notevalues can be computed .

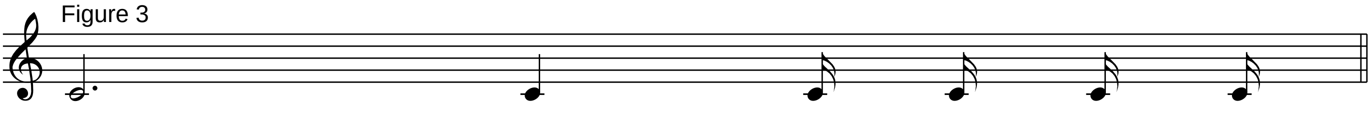
Suppose L:1/4
Then we can make the following computations

```
C3 = 3 * 1/4 = length of three quarter notes =
               length of 2 + 1 quarter notes =
               length of a halve note and the length of a quarter note =
               dotted halve note

C//2 = 1/2 * 1/2 * 1/4 = length of one quarter note divided by 4 =
                         sixteenth note
```

The result is figure 3.

Figure 3

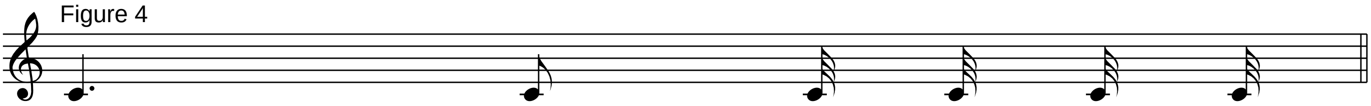


Suppose L:1/8
Then we can make the following computations

C3 = 3 * 1/8 = length of three eight notes =
length of 2 + 1 eight notes =
length of a quarter note and the length of a eighth note =
dotted quarter note

C//2 = 1/2 * 1/2 * 1/8 = length of one eight note divided by 4 =
thirty−second note

The result is figure 4.



I have two problems with this flexible way of code interpretation:

1. The same ABC code leads to different **visual** output (chapter 5).
2. One always needs the L: value to understand the code; so the code is in a sense ambigious and mentally intensive.

5. From visual output to comprehensive and unambigious code

Let’s reverse code and visual output. If I ask a musician to give the names of the note values of figure 3, I will always become an unambigious answer. E.g. the American, Dutch and German musicians −sharing the same terms− will NOT say something like:

C three times a quarter note, C quarter note and four C quarter notes divided by four

No, they will say

C dotted halve note, C quarter note and four C sixteenth notes.

This saying can be easily coded as the following − in my view− perfect **symbolic** translation shows:

C2. C4 C16 C16 C16 C16

Now, there is a nice match between note value image (figure 3) and code: C16 corresponds to a sixteenth note, C4 to a quarter note etc.. This is the key of my ABC−modification, for American, Dutch and German musicians straightforward. However, French or British musicians don’t use a fraction of the semibreve (whole) note as note names (’noire’ = ’crotchet’, ’croche’ = ’quaver’, ’double croche’ = ’semiquaver’ etc.).

In this case, one could start from Guido’s coding for figure 3:

[c1*1/2. c1*1/4 c1*1/16 c1*1/16 c1*1/16 c1*1/16]

or in the short form

[c1/2. c1/4 c1/16 c1/16 c1/16 c1/16]

This is theoretically correct, however it leads to complex code. Lilypond’s code version of figure 3 is more or less the same as the code in the MC Musiceditor.

In addition, from rhythmical point of view are the ABC representation C3, C and C//2 completely meaningless in relation to our standard musical notation. It makes only sense in relation to an abstract value L, after doing some math. The following ABC−alternative to C3 C C//2 C//2 C//2 C//2

C2 > C2 C//2 C//2 C//2 C//2 ||

is not an improvement when speaking about readabilty of code.

6. Don’t make me think

Chapter 2 said that a less error−prone performance of a rhythm will be possible in case of a better recognition. It also said that the less mental translations there are, the better the recognition will be! So a language as the ABC−language where different code such as

"C2 C2 | C C C C" = "C C | C/2 C/2 C/2 C/2" = "C8 C8 | C4 C4 C4 C4"

has the same output is from this point of view a sort of inhibition of pattern learning and pattern reading process. Only in special cases, i.e easy melodies, the ABC−code corresponds more or less to our standard musical notation. Two examples will suffice.

Example 1
ABC−code: C2 D2 | E2 F2 | G2 A2 | G2 F2 | E2 D2 | C2 z2 with L:1/4

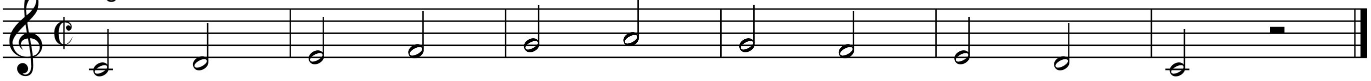


Figure 5

Example 2
ABC−code: C4 D4 | E4 F4 | G4 A4 | G4 F4 | E4 D4 | C4 z4 with L:1/16

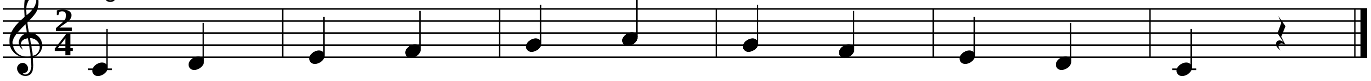


Figure 6

The short hand in the next example is a nice possibility of the ABC−language, however the rhythmic value information has disappeared. It is an option that makes readability of code again dependent on the variable L:

Example 3
ABC−code: C D | E F | G A | G F | E D | C z with L:1/4

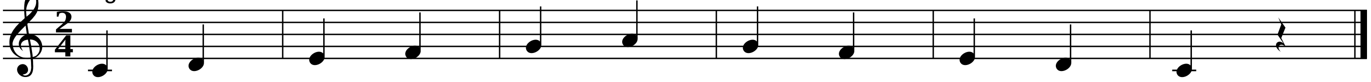


Figure 7

If other notevalues are needed, then the ABC code will be less readable. In contrast, the first code in chapter 5

```
C2. C4 C16 C16 C16 C16
```

has always the same output and is connected to the same mental image of the notes. As already said, the interpretation of this code is meaningful in relation to the practice of musical notation in contrast to

```
C3 C  C//2 C//2 C//2 C//2
```

Don’t make me think! In other words, it’s quite easy to learn a music notation language where series of characters correspond with patterns of notes.

7. ’MCM language’: an easy solution

As already indicated, I like the note−duration representations of Lilypond. However, I dislike Lilypond’s code complexity. I like the ABC language. However, I dislike its note representations. So I have implemented the idea of Lilypond’s note−duration representation in my MC Musiceditor.

That means that

− number 1 refers always to a Whole note
− number 2 refers always to a Half note
− number 4 refers always to a Quarter note
− number 8 refers always to an Eighth note
− number 16 refers always to a Sixteenth note
− number 32 refers always to a Thirty−second note

So:
− a group of four sixteenth notes C is always C16C16C16C16
− a group of two eighth notes C is always C8C8
− a triplet of thirty second notes C is always (3C8C8C8
etc.

Dotted notes are represented by dots. So C1. C2. C4. and C1.. C2.. C4.. etc. The ABC dotted notes solution as C > C and C >> C makes no sense.

These are the only modifications I made to the ABC−language that I implemented into my MC Musiceditor.

So MCM code doesn’t differ much from ABC, only the duration of notes have been defined in another way (the files of the MC Musiceditor have the extension .mcm)!

The MCM−advantages in short:

1. MCM note representations corresponds to definitions in the standard musical notation.
2. MCM note entry is always the same, in correspondence to music reading practice.
3. MCM note entry is for users with basic musical skills easy, i.e. without any computational efforts.
4. MCM−code can be easily visualized to images of ’real’ notes
5. Reading MCM−code is much easier than ABC−code

---

8. More ABC−modifications?

For me, there is only one problematic issue in the ABC−language: dynamics, phrasing marks, abbreviation etc. make the reading of the ABC and MCM code difficult. I don’t have a solution yet, although I have implemented some new marks in the MC Musiceditor. Example:

```
!begin_cresc!C8 D8 !end_cresc!E8 | G8 G8 G8 |
```

which at least for me can be read easily, instead of

```
!crescendo(!C8 D8 !crescendo)!E8 | G8 G8 G8 |
```

or

```
!<(!C8 D8 !>)!E8 | G8 G8 G8 |
```

I think this is more than a matter of taste.

9. Experience with MCM−language

Last year, my music students of the ArtEZ Conservatorium Netherlands −many of them are skilled Finale, Sibelius or Musescore users) created their homework with the MC Musiceditor. Results: note entry according the MCM−format is experienced as very ’intuitive’. The remarks I made frequently were about better readability: using tabs and space in the code, shortening the code lines and documenting the code.

10. Future MCM

I feel encouraged by my students to think further about a user−friendly and comprehensible MCM music notation language. When the MCM language is more or less completed, I will consider to develop an open source, cross−platform MC Musiceditor.

About the author:

Reinier Maliepaard is psychologist, software engineer, organist and teacher at the ArtEZ Conservatorium Netherlands (music theory and music history).

His freeware music notation programm MC Musiceditor (Windows) can be downloaded at www.mcmusiceditor.com

These notes have been typeset with MC Musiceditor 6.0.4
