# Encoding Matters

Néstor Nápoles
McGill University, CIRMMT
Montréal, QC
nestor.napoleslopez@mail.mcgill.ca

Gabriel Vigliensoni
McGill University, CIRMMT
Montréal, QC
gabriel@music.mcgill.ca

Ichiro Fujinaga
McGill University, CIRMMT
Montréal, QC
ichiro.fujinaga@mcgill.ca

## ABSTRACT

In this paper, we discuss how different encodings in symbolic music files can have consequences for music analysis, where a truthful representation, not only of the musical score, but of the semantics of the music, can change the results of music analysis tools. We introduce a series of examples in which different encodings effectively modify the content of two—apparently equivalent—symbolic music files. These examples have been obtained from comparing three different encodings of a string quartet movement by Ludwig van Beethoven.

We present two scenarios in which encoding discrepancies may be introduced. In the first scenario, they have been introduced during the encoding of the symbolic music file by either the music notation software or the human encoder. The discrepancies introduced in this scenario are typically difficult to notice because they are *visually* identical to an accurate encoding. In the second scenario, the discrepancies have been introduced during the translation of the original file into other symbolic formats. In this scenario, the discrepancies may be related to propagating errors in the original encoding or to an erroneous translation of certain attributes of the musical content. Finally, we discuss the possibility of using the examples provided here for the mitigation of some of these discrepancies in the future.

## CCS CONCEPTS

• **Information systems** → **Music retrieval**; • **Software and its engineering** → *Extensible Markup Language (XML)*; *Software design tradeoffs*; Software libraries and repositories; • **Applied computing** → Sound and music computing;

## KEYWORDS

Symbolic music, music notation, music encoding, music transcription, music information retrieval, musicxml, mei, humdrum, verovio, humlib, vis, music21

**ACM Reference Format:**
Néstor Nápoles, Gabriel Vigliensoni, and Ichiro Fujinaga. 2018. Encoding Matters. In *5th International Conference on Digital Libraries for Musicology (DLfM '18), September 28, 2018, Paris, France.* Paris, France, 5 pages. https://doi.org/10.1145/3273024.3273027


*DLfM '18, September 28, 2018, Paris, France*

ACM ISBN 978-1-4503-6522-2/18/09…$15.00
https://doi.org/10.1145/3273024.3273027

## 1 INTRODUCTION

The content of music scores stored in digital music libraries has one of two forms, either scores stored as images (e.g., scans of a manuscript) or a content-based representation (e.g., staff, clef, voices, pitches, durations) in a symbolic music file [5]. Whenever we want to visualize a music score, images might be enough. For example, a scan of the music score document allows a digital music library to distribute a digital version that users can access and visualize in electronic devices. However, in activities that require searching or browsing through the content of the score (e.g., finding instances of a particular melody within the score or counting the number of occurrences of all notes), images are certainly not enough. In these cases, a content-based representation of the score is needed.

The additional benefit of having a content-based representation is that it is typically possible—and easy—to generate a new rendering of the music score from it. In this way, content-based representations are able to produce the visualization of a music score at the same time they provide access to its musical content. Unfortunately, encoding a content-based representation of a music score is a time-consuming process that involves, in most cases, the use of music notation software. For this reason, content-based representations are still scarce and digital music libraries mainly offer the musical content in their catalogues as scanned images [3, 5]. With the improvement of Optical Music Recognition (OMR) technologies, we expect digital music libraries to transition from images and scans to content-based representations more easily. Meanwhile, we rely in music notation software for encoding content-based representations of music scores.

When a music score is encoded using music notation software, the main feedback given to the user is the visualization of the score itself. Most music notation software follow the well-known *What You See Is What You Get* (WYSIWYG) paradigm, which is widely used for different editing applications. One particular problem that arises from this is that two symbolic scores might look almost identical visually, however, their underlying encodings may be nonequivalent. These differences may emerge due to discrepancies introduced by the users who encode the music score, but also due to assumptions from the software that imports or exports the symbolic music files.

In the following sections, we explore the discrepancies that have been found in three encodings produced by different human encoders using different music notation software.

## 2 METHODOLOGY

Before we introduce the distinct types of discrepancies found in the encodings, we will briefly describe the encodings we used, their source, and the software that was used to produce them.

## 2.1 Encodings

We have selected three different—publicly available—encodings of the first movement of the string quartet composed by Ludwig van Beethoven, *Op.18 No.1*.

The music notation software used to produce the encodings are: Finale,[1] MuseScore,[2] and Sibelius.[3]

*2.1.1 Finale encoding.* This encoding was obtained from the Gutenberg project.[4] According to the documentation, this encoding was produced by Geof Pawlicki. The score is freely available in Finale's proprietary format *.MUS*. We have used Finale v25.4.1.163 to export it as a MusicXML file.

*2.1.2 MuseScore encoding.* This file was obtained from the community of MuseScore users, where it can be freely downloaded in several formats, for example, MuseScore's proprietary format *.mscz*. This encoding was produced by Gavin Ailes. We used MuseScore v2.2.1 to export it as a MusicXML file.

*2.1.3 Sibelius encoding.* This encoding has been obtained from Tes,[5] a global network of education professionals. The music score has been submitted by the user *dunhallin* and it is available for purchase in Sibelius' proprietary format *.SIB*. We used Sibelius v2018.1 to export it as a MusicXML file.

In each case, we have made sure that the music notation software used to generate the MusicXML files is the latest stable version available. That is also true for the software used to translate the MusicXML files that we describe next.

## 2.2 Translations

When parsing symbolic music files, it is common that the software will only accept a certain number of symbolic formats. Whenever that happens, it may be necessary to translate the original file into other formats. Therefore, in addition to comparing the different encodings, we are also interested in investigating the effects of translating a file into other symbolic formats. For this reason, two additional symbolic music formats have been used: the Music Encoding Initiative (MEI) format, and Humdrum (**kern).

*2.2.1 Music Encoding Initiative (MEI).* The Verovio toolkit is an engraving library for MEI files[6]. As part of its features, it can be used to translate MusicXML files into MEI. We used the Verovio toolkit v2.0.0 to translate the MusicXML files into MEI.

*2.2.2 Humdrum (**kern).* We used the *musicxml2hum* program inside the *humlib*[6] library to translate the MusicXML files into Humdrum (**kern).

## 2.3 Randomization

Because our aim is to evaluate the *differences* between encodings and not finding out if one encoding is better than another, we have anonymously named the different encodings. We refer to them as *Encoding A, B, and C*. In the same manner, we refer to the translations as *Translation A1, A2, B1, B2, C1,* and *C2*. Figure 1 summarizes the randomized identifiers.[7]

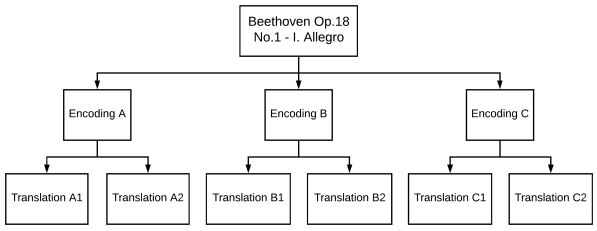


**Figure 1: Symbolic files used for the comparisons. The top row represents the score document. The three boxes in the second row represent a MusicXML exported from a specific music notation software. The boxes in the third row represent a translation of the exported MusicXML to two additional symbolic music files.**

# 3 FINDING ENCODING DISCREPANCIES

Having already introduced the source of the encodings, music notation software, translation software, and randomized identifiers, we now introduce our procedure for finding encoding discrepancies.

## 3.1 Procedure

In order to discover discrepancies between two different encodings, we list—for each instrument in both encodings—all the onsets in which new notes or rests appear. We refer to these as note/rest onsets. Two note/rest onsets are considered identical if they start at the same time (i.e., onset synchronicity) and have the same value (i.e., both are the *same note* or both are *rests*). Figure 2 shows a summary of the comparisons between all three encodings. Blank sections within the plot indicate that at least one instrument has a different value compared to the other encoding. If an encoding was compared to itself, the plot would have no blank sections. The data used for this plot has been obtained by using music21 [2] and VIS-framework [1]. To allow reproducibility, we make this code available.[8]

The summary of the comparisons presented in Figure 2 serves as a reference for finding zones of discrepancy between two encodings. In order to investigate the cause of the discrepancies, we manually inspected the rendering of the music score and the MusicXML file of a selected subset of discrepancies. We present a list of discrepancies that occur frequently. We have divided this list in two parts: discrepancies that are more related to music notation software and discrepancies that are more related to decisions made by human encoders.[9]

## 3.2 Related to music notation software

Music notation software may contribute to introducing discrepancies, for example, when it allows human encoders to export

[1] https://www.finalemusic.com/
[2] https://musescore.org/en
[3] https://www.avid.com/sibelius
[4] https://www.gutenberg.org/wiki/Gutenberg:The_Sheet_Music_Project
[5] https://www.tes.com/
[6] https://github.com/craigsapp/humlib

[7] The exact correspondence to the source, music notation software, and translation software that produced the symbolic music files can be requested from the authors.
[8] https://www.github.com/napulen/encoding_matters
[9] Although we recognize that this division can be ambiguous in some cases.

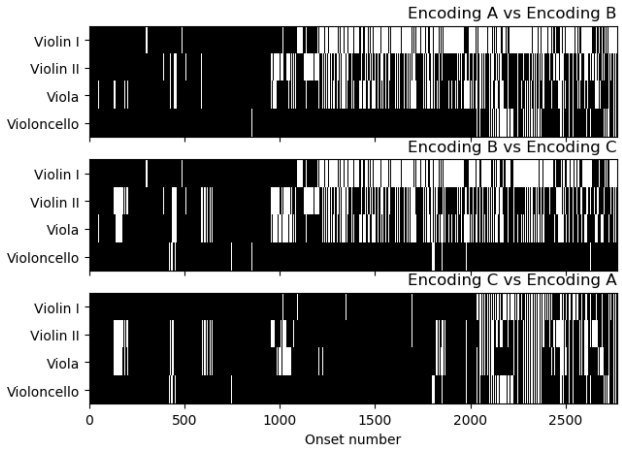


**Figure 2: Summary of the comparison between the three music encodings. The horizontal axis represents the combined sequence of note/rest onsets in chronological order. Blank zones indicate discrepancies between the encodings.**

inconsistent encodings (e.g., an *incomplete* or an *overrun* measure). More commonly, it may introduce discrepancies due to software bugs and unexpected behaviors.

*3.2.1 Unclosed tie.* One example of an unexpected behavior involves tied notes, more precisely, *unclosed* tied notes. In order to interpret the length of a tie, MusicXML parsers typically rely in music notation software to explicitly provide the starting and ending notes of a tie. In some cases, however, ties may start at a certain note and not end, that is, they remain unclosed. This leads to an ambiguous interpretation of the duration of a tie. Figure 3 shows an example of this discrepancy, the tied *E♭* in the upper voice of Violin II remains unclosed in the note of the next measure, where it is supposed to end. The interpretation of where this tie ends varies depending on the software that parses the MusicXML file.

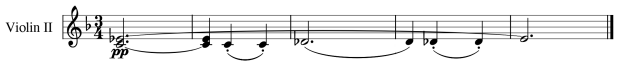


**Figure 3: Unclosed tie. *Encoding B,* measures 41–45.**

*3.2.2 Incomplete measure.* The next two examples involve the discrepancy of the duration of a measure and the duration of the notes *within* the measure. These are important because different software may also interpret them in different ways, moreover, these are important because once they happen, they *propagate* for the rest of the score. The first example is an incomplete measure.

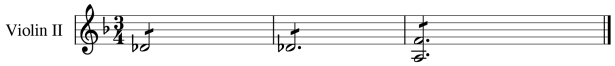


**Figure 4: Incomplete measure. *Encoding B,* measures 152–154. The half note at mm. 152 should be a dotted half note.**

Figure 4 shows an example of an incomplete measure in *Encoding B*. The half note in the first measure of the score snippet from Violin II is meant to be a dotted half note.

*3.2.3 Overrun measure.* Figure 5 shows a contrary example to the previous one. In this case, the human encoder that produced *Encoding B* has accidentally placed an augmentation dot in the third *high-C* note of Violin I. Figure 6 shows the interpretation of another music notation software when parsing the same MusicXML output.

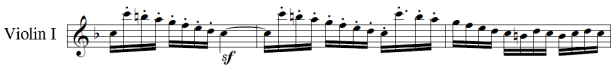


**Figure 5: Extraneous augmentation dot that overrruns the duration of the measure (*Encoding B,* measures 168–170). The augmentation dot is located in the third *high-C*.**

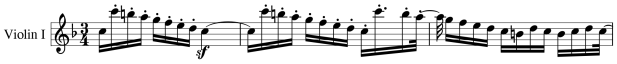


**Figure 6: Interpretation of the music in Figure 5 in a different music notation software.**

## 3.3 Related to human encoders

During this research, we have observed that the discrepancies related to music notation software involve corner-cases and ambiguities in how musical content is encoded. On the other hand, the discrepancies linked to human encoders are more related to the difficulty of visually perceiving differences between two music scores. In the *overrun measure* example presented above, a human encoder could be careful of correcting such errors, however, the articulation dots may easily obstruct the possibility of noticing the error in the first place.

*3.3.1 Slurs for ties.* A similar discrepancy—and more difficult to perceive—occurs when the human encoder places, by mistake, a *slur* symbol instead of a *tie*. This type of discrepancy is important for two reasons: 1) the visual representation of slurs and ties is close to identical, 2) a tied note does not generate a new onset of the note (i.e., the first note will be played and held for the duration of both tied notes), contrarily, a slur does. Due to the almost indistinguishable visual representation, it is hard to find these discrepancies without the aid of automated tools.

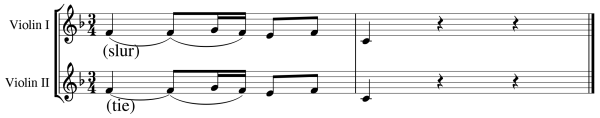


**Figure 7: The first two notes of Violin I have been *slurred* instead of *tied. Encoding C,* measures 1–2.**

Figure 7 shows an example of this discrepancy found when inspecting *Encoding C*. Other examples of this type of discrepancy have been found in *Encoding B*.

*3.3.2 Repeated notes, trills, and grace notes.* Many discrepancies in note/rest onsets shown in Figure 2 come from whether human encoders decide to explicitly repeat a sequence of notes with the same note value or use *abbreviation* symbols instead (e.g., slashes). Musically speaking, they are analogous, however, the symbolic files may be interpreted differently by music notation, and particularly, music analysis software. Figure 8 shows an example of this discrepancy between *Encodings A* and *B*. Similar examples have been found for *trills*, *grace notes*, and other ornaments.

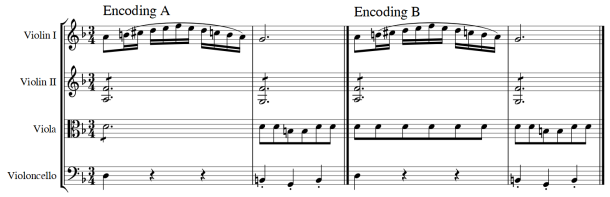


**Figure 8: Discrepancy in the notation of repeated notes between two encodings. *Encoding A* (left) and *B* (right), measures 30–31. The discrepancy is located in the Viola part of measure 30.**

## 4 FINDING TRANSLATION DISCREPANCIES

We have also decided to investigate the discrepancies introduced by translations of symbolic music formats. We present the procedure and results of six comparisons, two translations for each original encoding. We hypothesize that each note/rest onset in the original encoding should be found in the translation. That is, a note/rest onset should exist in the translation, such that, it is synchronized (i.e., starts at the same time), has the same note/rest value, and the same attributes as a note/rest onset in the original encoding.

### 4.1 Procedure

For each instrument in the encoding, we iterate over every note/rest onset attempting to find a synchronized note/rest onset in the translation, if one is found, we verify that the note/rest value and the additional properties are identical. The additional properties we consider are *duration*, *articulations*, and *ornaments*.[10]

Table 1 shows the percentage of onsets that have been found in the translation. We separate these percentages in three categories: synchronized note/rest onsets that are identical, synchronized note/rest onsets that have different properties, and note/rest onsets that were not found in the translation. The *Non-sync* category indicates the amount of note/rest onsets from the original encoding that were not found in the same location in the translation.

### 4.2 Translations A1, B1, and C1

These translations correspond to the same symbolic music format. It can be observed from Table 1 that, for all three encodings, the translation process was able to replicate all of the onsets in the same position as they were in the original encoding. After inspecting the conflicting note/rest onsets, we conclude that discrepancies in the note/rest values and attributes are related mainly to *grace notes* encoded with a different duration attribute in the translated file.

[10]These properties are validated using the default comparison of *note*, *rest*, and *chord* objects in music21. We rely on the accuracy of this process to guarantee the quality of our results.

**Table 1: Comparison of note/rest onsets between translations. Synchronized and non-sync indicate whether the onsets in the encoding exist in the same location of the translation or not. For synchronized onsets, identical or different indicate if the attributes have been preserved.**

| Encoding vs. Translation | Synchronized Identical / Different | Non-sync |
|---|---|---|
| A vs. A1 | 95.2% / 4.8% | 0.0% |
| A vs. A2 | 6.1% / 48.5% | 45.4% |
| B vs. B1 | 95.1% / 4.9% | 0.0% |
| B vs. B2 | 18.6% / 5.3% | 76.1% |
| C vs. C1 | 94.4% / 5.6% | 0.0% |
| C vs. C2 | 26.1% / 2.3% | 71.6% |

### 4.3 Translations A2, B2, and C2

Similarly, these translations correspond to the same symbolic music format. The number of de-synchronized note/rest onsets surpasses the number of synchronized ones. At some point during the translation process, propagation errors (e.g., overrun measure) displace the offset of new note/rest onsets in the translated symbolic format, causing discrepancies with the original encoding.

## 5 CONCLUSIONS

In this paper, we have presented examples of discrepancies in symbolic music files that were encoded by different human encoders and music notation software. We also investigated the discrepancies introduced when these files were translated into other symbolic music formats. Several of the discrepancies repeat systematically in the form of patterns (e.g., users place a *slur* between two notes of the same pitch when they meant to place a *tie* or incomplete measures are interpreted differently in distinct music notation software).

In many cases, developing better methodologies for symbolic music corpora creation [4] should set the basis for reliable data and reproducible research, however, whenever this is not sufficient, detecting the patterns leading to discrepancies—as the ones described here—seems to be a promising and worthy effort in the comparison, evaluation, and, possibly, the auto-correction of symbolic music files, which is in the interest of music researchers and digital music libraries.

For now, we know that searching or browsing through the content of symbolic music files leaves the user with an additional responsibility, the responsibility of knowing that encoding matters.

## 6 ACKNOWLEDGEMENTS

This research has been supported by the Social Sciences and Humanities Research Council of Canada (SSHRC) and the Fonds de recherche du Québec—Société et culture (FRQSC).

## REFERENCES


[1] Christopher Antila and Julie Cumming. 2014. The VIS Framework: Analyzing counterpoint in large datasets. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*. Taipei, Taiwan, 71–76.

[2] Michael Scott Cuthbert and Christopher Ariza. 2010. music21: A toolkit for computer-aided musicology and symbolic music data. In *Proceedings of the 11th International Society for Music Information Retrieval Conference*. Utrecht, Netherlands, 637–642.

[3] Audrey Laplante and Ichiro Fujinaga. 2016. Digitizing musical scores: Challenges and opportunities for libraries. In *Proceedings of the 3rd International Workshop on Digital Libraries for Musicology*. New York, NY, 45–48.

[4] Cory McKay and Julie Cumming. 2018. Methodologies for creating symbolic early music corpora for musicological research. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*. Paris, France.

[5] Laurent Pugin. 2015. The challenge of data in digital musicology. *Frontiers in Digital Humanities* 2 (2015), 4.

[6] Laurent Pugin, Rodolfo Zitellini, and Perry Roland. 2014. Verovio: A library for engraving MEI music notation into SVG. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*. Taipei, Taiwan, 107–112.