MusicDiff – A Diff Tool for MEI

Kristin Herold
Beethovens Werkstatt
herold@beethovens-werkstatt.de

Dr. Johannes Kepper
Beethovens Werkstatt
kepper@edirom.de

Ran Mo
Beethovens Werkstatt
mo@beethovens-werkstatt.de

Agnes Seipelt
Beethovens Werkstatt
seipelt@beethovens-werkstatt.de

Introduction

For musicologists, the collation of multiple sources of the same work is a frequent task. By comparing different witnesses, they seek to identify variation, describe dependencies, and ultimately understand the genesis and transmission of (musical) works. Obviously, the need for such comparison is independent from the medium in which a musical work is manifested.

In computing, comparing files for difference is a common task, and the well-known Unix utility *diff* is almost 46 years old [1]. However, *diff*, like many other such tools, operates on plain text. While many music encoding formats based on plain text exist, formats used in the field of Digital Humanities are typically based on XML. There are dedicated algorithms for comparing XML as well,[1] but they only focus on the syntax of XML, but not the semantic structures modelled into such standards as MEI. MEI seeks to describe musical structures, and the XML syntax is just a means to express those structures. A diff tool for music should focus on comparing musical structures, but not the specifics of their serialization into a file format.

In *Beethovens Werkstatt*, a 16-year project focussed on exploring the concepts and requirements of digital genetic editions of music, based on and arguing with examples from Ludwig van Beethoven, a case-bound diff tool for music was developed. The following paper discusses how that specific tool can be generalized, and which use cases such a tool may support.

VideApp$_{Arr}$

Beethovens Werkstatt seeks to explore compositional processes from different perspectives. In its recently completed second module, the project dealt with a number of Beethoven's works that the composer re-arranged for other performing forces. For these works, printed editions of both the original works and their respective arrangements were fully encoded in MEI, following a rather plain style, i.e. no typographical or genetical details about the sources were preserved. Instead, an additional file per comparison with merely more than pointers to both source encodings was provided. With this data model, it is possible to automatically align both files and present them from multiple perspectives with an application called VideApp$_{Arr}$ – the component dealing with arrangements within the (modular) VideApp.[2]

1 https://pypi.org/project/xmldiff/, http://diffxml.sourceforge.net/, https://www.oxygenxml.com/files_compare_img.html

2 https://videapp-arr.beethovens-werkstatt.de

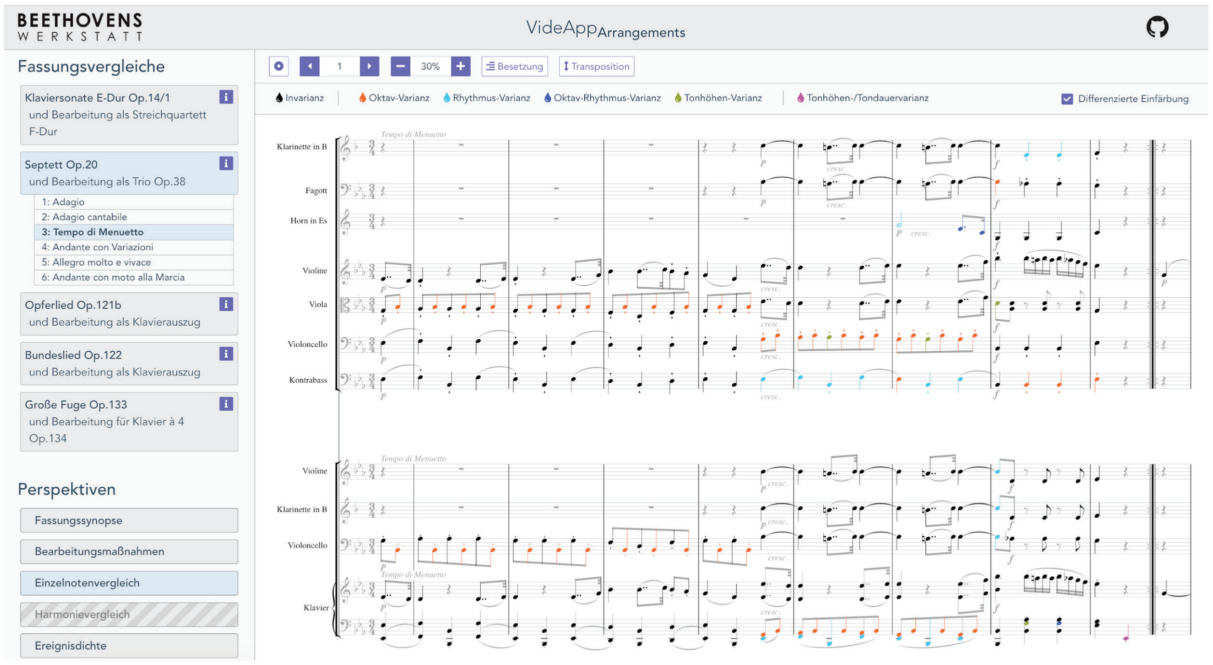
BEETHOVENS WERKSTATT
VideApp Arrangements
Fassungsvergleiche
Klaviersonate E-Dur Op.14/1 und Bearbeitung als Streichquartett F-Dur
Septett Op.20 und Bearbeitung als Trio Op.38
1: Adagio
2: Adagio cantabile
3: Tempo di Menuetto
4: Andante con Variazioni
5: Allegro molto e vivace
6: Andante con moto alla Marcia
Opferlied Op.121b und Bearbeitung als Klavierauszug
Bundeslied Op.122 und Bearbeitung als Klavierauszug
Große Fuge Op.133 und Bearbeitung für Klavier à 4 Op.134
Perspektiven
Fassungssynopse
Bearbeitungsmaßnahmen
Einzelnotenvergleich
Harmonievergleich
Ereignisdichte
Besetzung
Transposition
Invarianz
Oktav-Varianz
Rhythmus-Varianz
Oktav-Rhythmus-Varianz
Tonhöhen-Varianz
Tonhöhen-/Tondauervarianz
Differenzierte Einfärbung
Klarinette in B
Fagott
Horn in Es
Violine
Viola
Violoncello
Kontrabass
Klavier
Tempo di Menuetto

Figure 1: VideApp$_{\text{Arr}}$ showing the “Single Note Comparison” of Beethoven’s op. 20 (top) and op. 38 (bottom).

Most of these perspectives are based on the comparison of individual notes in three different “dimensions”: metrical position, pitch, and rhythm. Metrical position means that only notes sounding simultaneously will be compared. For pitch, octave and pitch class are evaluated independently, while rhythm is taken into account directly. Variation of these three parameters is organized into different combinations, such as notes in a different octave, notes with different duration or other types of variation, but also notes which have an exact match. No attention is paid to beams and similar features, as they are mostly visual artifacts, which typically do not affect the musical structure. By intention, accidental aspects of the score such as dynamic markings are not taken into account for comparison either, as their high incidence may easily conceal the more significant substantial differences. Voice leading is also ignored by this comparison, as it would be misleading in the context of a comparison of rearranged works. Especially in a piano reduction, “voices” from multiple instruments are condensed in a way that frequently fails to show the same “melodic lines” for middle voices and others, so that the aspect of preceding and / or succeeding notes can hardly be made a default criterion for comparing two arrangements.

Generalising VideApp$_{\text{Arr}}$ to MusicDiff

The data model underlying VideApp$_{\text{Arr}}$ is a rather strict version of MEI, disallowing variation, editorial intervention, and other more complex concepts of MEI. While it took significant effort to ensure correctness of the encodings used in *Beethovens Werkstatt*, the generation of these encodings was straightforward in principle, as they were just transcribed from the original prints using scorewriting applications, and then transformed into MEI via MusicXML conversion. This workflow is all but unique, and we anticipate that numerous other projects create MEI files with about the same information value, though perhaps expressed in slightly different models of MEI.

In the process of proofreading the files relevant for the second module of the project, it became apparent that the VideApp$_{\text{Arr}}$ is actually very supportive in this task, as it consequently highlights differences of all kinds, even when some of which are not visible in a rendered score. This is particularly true for the correct encoding of *gestural* information in MEI, which in this context means sounding pitch affected by the general key signature at the beginning of the piece, but not local accidentals.

This observation led to the idea of broadening the scope of this tool beyond the original context of *Beethovens Werkstatt*, and to modify it so that users can actually upload and *diff* their own MEI files. While several of the perspectives offered by the VideApp$_{Arr}$ may be useful for this purpose, we intentionally focussed on the most simple diff view to begin with. This view has been condensed into a separate web application called MusicDiff.[3] The following examples illustrate the use of this app for musicological purposes beyond the original scope of *Beethovens Werkstatt*.

Example use cases

In opera, the music was usually adjusted to local requirements, settings, and expectations. Pieces from different works were frequently integrated (in)to performances, which led to the need to create smooth transitions between those pieces. The research project "Pasticcio. Ways of arranging attractive Operas"[4] explores such pasticcii. This includes the recitative "Ah Per te solo" from the pasticcio "Catone" by G. F. Handel, which was first performed in London in 1732. In Handel's manuscript, two versions of this recitative are transmitted, one ending on G# major, leading to the aria "Care faci del ben mio" (E major), and one leading to the replacement aria "Sento in riva all'altre sponde" (A major). Obviously, the different key of the substituted succeeding aria required some adjustments to the music.

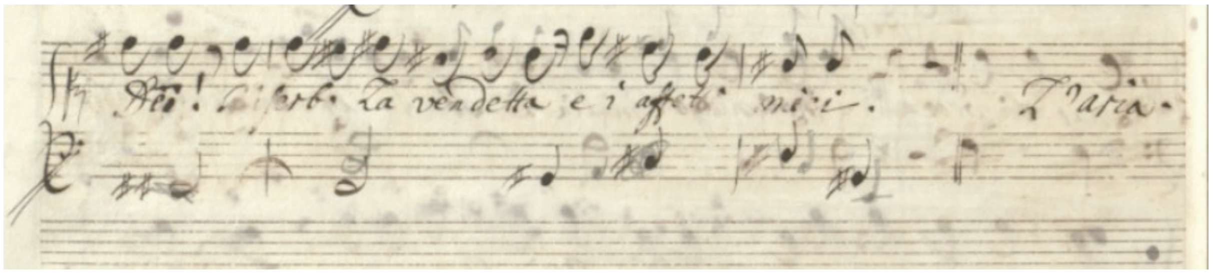

Figure 2: Recitative "Ah Per te solo" from the Pasticcio "Catone" by G. F. Handel. Staats- und Universitätsbibliothek Hamburg Carl von Ossietzky, D-Hs M A/1012, p. 187.[5]

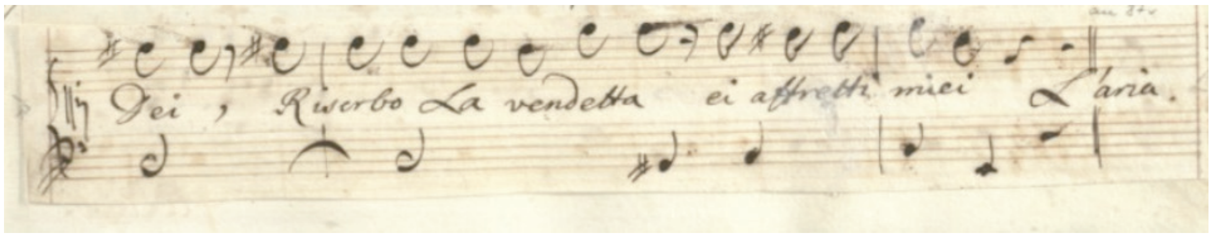

Figure 3: Substituted recitative "Ah Per te solo" from the Pasticcio "Catone" by G. F. Handel. Staats- und Universitätsbibliothek Hamburg Carl von Ossietzky, D-Hs M A/1012, p. 184.[6]

Obviously, it is possible to manually compare the score images, and there are also tools supporting such an approach at least with musical prints,[7] this approach doesn't scale well and may take significant time when comparing works larger than these four measures. However, when looking at the rendition provided by MusicDiff, the difference between both versions becomes immediately imminent:

3 Available from https://music-diff.edirom.de

4 https://www.pasticcio-project.eu/

5 https://digitalisate.sub.uni-hamburg.de/de/nc/detail.html?id=1901&tx_dlf%5Bid%5D=22734&tx_dlf%5Bpage%5D=187

6 https://digitalisate.sub.uni-hamburg.de/de/nc/detail.html?id=1901&tx_dlf%5Bid%5D=22734&tx_dlf%5Bpage%5D=184

7 https://ehinman.edirom.de/

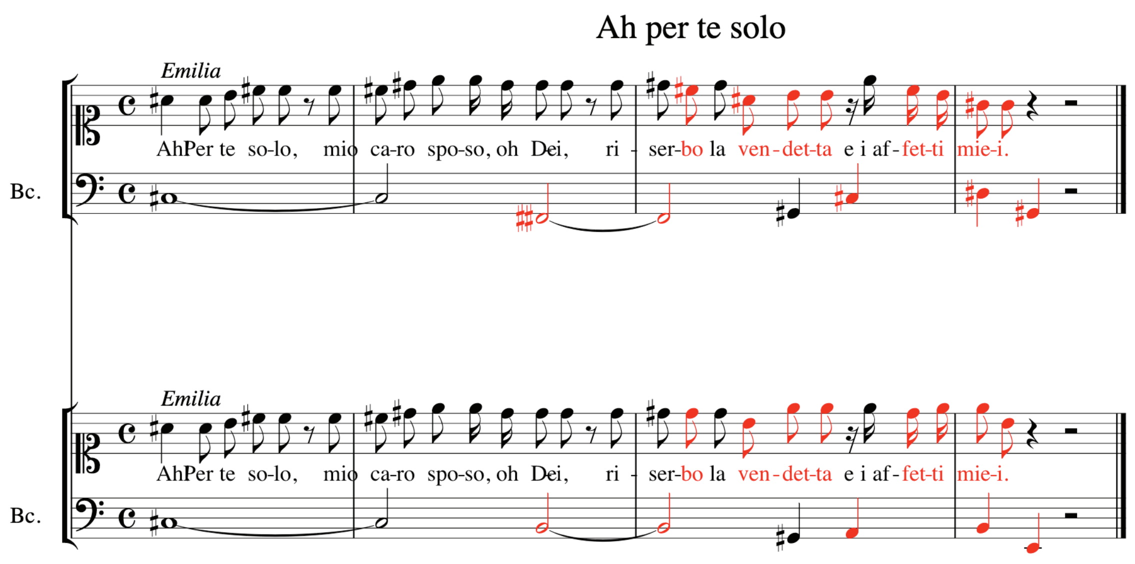
Ah per te solo
Emilia
AhPer te so-lo, mio ca-ro spo-so, oh Dei, ri - ser-bo la ven-det-ta e i af-fet-ti mie-i.
Bc.
Emilia
AhPer te so-lo, mio ca-ro spo-so, oh Dei, ri - ser-bo la ven-det-ta e i af-fet-ti mie-i.
Bc.

Figure 4: Comparing encodings from both versions of "Ah Per te solo" with MusicDiff.

A second example helps to illustrate the flexibility of MusicDiff, and why a regular diff tool would necessarily fail to recognize musical differences at the level of abstraction considered here. This example deals with two independent encodings of Grieg's "Erotikon" op. 43, Nr. 5. One of these encodings has been made available as Humdrum file by KernScores,[8] while the other comes as MusicXML file, derived from an original Capella transcription.[9] Even though the origins of both encodings do not suggest this interpretation, one may wonder if these files share a history, i.e. if one has been converted from the other, or both have been derived from an (unknown) original encoding. If that would be the case, their content would probably be almost identical, with differences being caused by transformation loss (and thus highly systematic differences). In order to answer these questions, both encodings have been transformed to MEI, and processed by MusicDiff.

8 https://kern.humdrum.org/cgi-bin/ksdata?location=users/craig/classical/grieg/op43&file=erotic-poem.krn&format=info

9 http://www.hausmusik.ch/notenregal/g/grieg/klavierstuecke/lyrische_stuecke/erotik_edvard_grieg_/

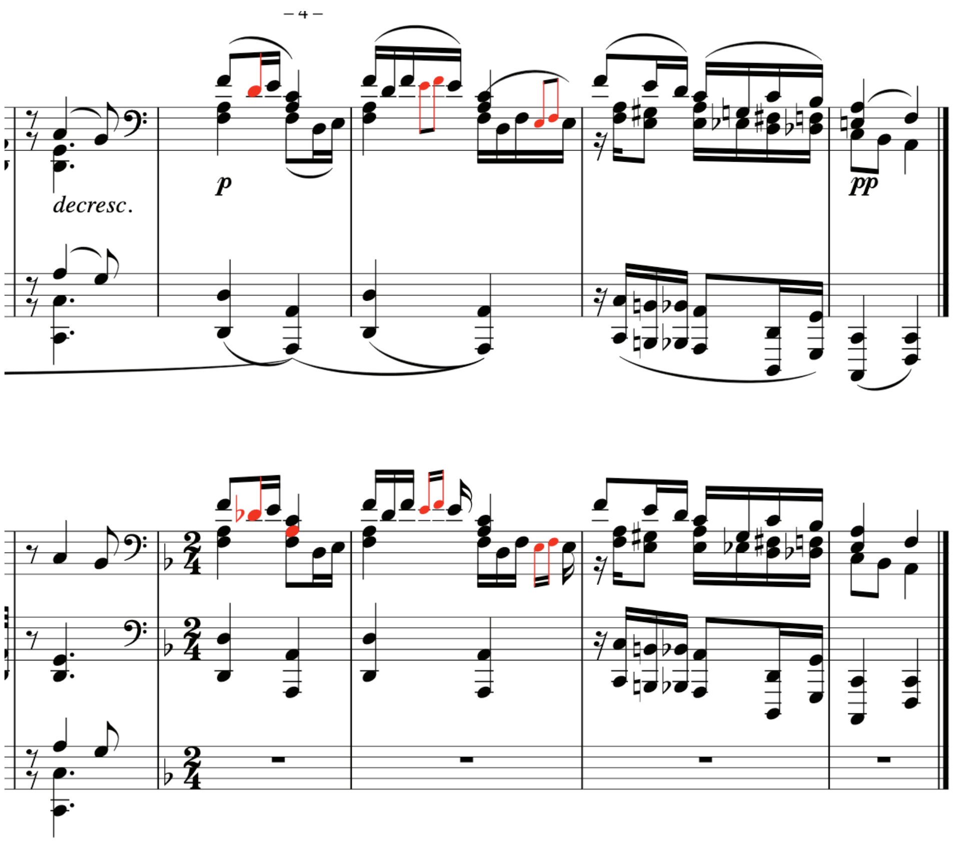

Figure 5: The last 5 measures of the lyrical piano piece "Erotikon" op. 43, Nr. 5 of Grieg (top: Humdrum file converted to MEI, bottom: MusicXML file converted to MEI.

Apparently, there is a small level of variation between both encodings, with only a small number of regular and grace notes being highlighted by MusicDiff. This seems to indicate that the original encodings have been generated independent of each other. However, this example perfectly illustrates how MusicDiff is able to overlook structural differences: The fact that the second version of Grieg's piece is laid out on three staves does not affect the comparison. In the same way, MusicDiff is able to go over music written in chords vs. music written in voices, or, more generic, layers. Admittedly, other interpretations of what qualifies as variation are possible and equally valid.[10]

Keeping an overview

While the collation provided by MusicDiff offers a very striking emphasis of the variation in the current viewport, this perspective does not provide a wider overview of the work in total – the user has to flip through all pages to get an impression of the distribution of differences between the compared encodings. In order to facilitate getting such an impression, *Beethovens Werkstatt* has integrated the concept of Sunburst diagrams[11] into VideApp$_{Arr}$, and this feature has been carried over to MusicDiff as well. Sunburst diagrams visualize hierarchical data by concentric circles. On the outer ring, all measures of a piece are given, while the second ring denotes musical sections, and the inner ring reflects movements. The user may click on any measure, and the page holding this measure will be displayed. However, as seen in Figure 6, the measures may be used to provide additional information as well.

10 It would be certainly possible to include those different interpretations into the stylesheets underlying MusicDiff and let the user pick the "strictness" of the comparison according to her specific needs, but this would require significant work clearly out of scope for Beethovens Werkstatt.

11 https://en.wikipedia.org/wiki/Sunburst_chart

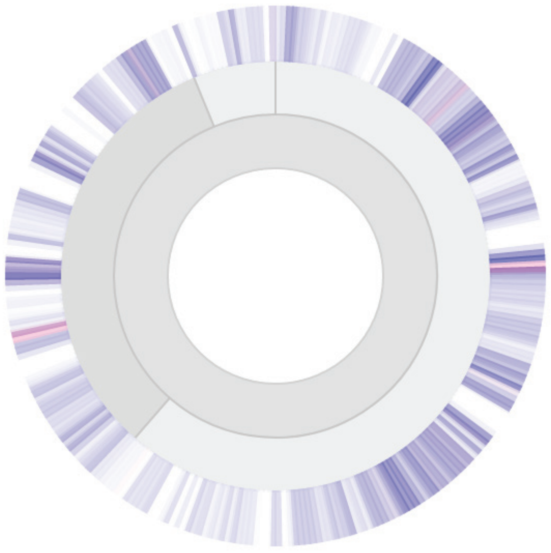

Figure 6: A Sunburst diagram for the comparison of Beethoven's op. 20 and the rearrangement into op. 38 based on it. White color indicates identity between both versions, blue indicates variance, and red indicates difference.

In this example from *Beethovens Werkstatt*, measures are colored depending on the comparison results. First, the saturation of a measure indicates the level of identity between original version and rearrangement – a measure displayed in white is unchanged, while a colorful measure has a high degree of variation. In *Beethovens Werkstatt*, however, a distinction is made between variant notes (which still share the pitch class or duration with their counterpart) and different notes (which have no counterpart at their respective metrical position at all). While variant notes typically indicate local adjustments of some sort, differences in this sense indicate major compositional processes. While the first are indicated by blue color, the latter make use of red tones. Both colors may blend according to the ratio of their respective notes within each measure. With this mechanism, it becomes possible to get a very quick overview over the distribution of variation across all 288 measures of this example, and to navigate within the piece for closer inspection very easily.

Technical setup, limitations, and potentials

As mentioned earlier, the MusicDiff app is a stripped-down version of VideApp$_{Arr}$. It allows the user to upload her own MEI encodings. At this point, no validation happens while processing the data – it is upon the user to ensure that the input conforms to the schema[12] expected by the tool.

MusicDiff itself does not come with a backend. Instead, it utilizes a varied toolbox[13] for converting between different music encoding formats, manipulating MEI instances, and other related workflows. It is based on TEI's OxGarage and actually uses the same backend, gently adjusted to music needs. The user's uploaded files are wrapped in a new file, and sent to MEIGarage, which runs a fairly complex series of XSLT transformations[14] on the files, enriching them with various information needed to perform the actual comparison. The output is ultimately sent back to the user of MusicDiff and displayed there. This setup allows MusicDiff to be a rather lightweight application, which could be integrated into other tools quite easily.

MusicDiff relies on the MEI profile developed for VideApp$_{Arr}$. This profile strictly requires a very simple use of MEI. While it wasn't available at the time when work on VideApp$_{Arr}$ was begun, the recent MEI Basic profile seeks to serve the same purpose: the definition of a strongly simplified and strictly controlled version of MEI which may serve as a common ground for interchange both within MEI (i.e., between projects relying on different richer flavors of MEI) and outside of MEI (i.e., to simplify conversion with other, less expressive formats). As we expect significant uptake of MEI Basic, it seems sensible to modify MusicDiff to operate on this profile

12 https://github.com/BeethovensWerkstatt/module2/blob/dev/data/odd/bw_module2_works.odd

13 https://meigarage.edirom.de

14 https://github.com/Edirom/data-configuration/blob/dev/scripts/compare.files.xsl

instead of the current one. As both have an almost identical coverage of MEI features, and merely differ in how they are expressed, this seems like a reasonable goal which will significantly help to improve the applicability of MusicDiff.

Some more interesting features are available in VideApp$_{Arr}$, which haven't been ported to MusicDiff yet. This includes the possibility to transpose the encodings to be compared to a common key, should they be written in different keys – the actual comparison is already capable of comparing encodings independent of the key they use, but it sometimes helps the user to bring everything to C Major / A minor for better legibility. Another feature already available in the underlying transformations is the possibility to omit one or more staves from the versions to be compared. That way, it becomes possible to answer questions like how the clarinet of version A relates with the clarinet of version B. Both of these features are fully functional in the underlying code, but don't have a user interface in MusicDiff yet. We hope to add support for both these features in the near future.

A significantly more challenging issue is a general limitation of the comparison scripts, which, at this point, require that both versions use the same number (and distribution) of measures, i.e. measures are compared according to their position in the piece. For the examples covered in the second module of *Beethovens Werkstatt*, this was a safe assumption to make, but obviously, this isn't generally true. However, it is fairly complex to recognize whether two measures differ because of some variation between them, or because an additional measure has been inserted in one of the compared versions. While this is clearly an interesting and challenging issue, we don't expect to support this use case anytime soon, but may instead ask the user to submit a concordance of measures.

Conclusion

MusicDiff is a compelling tool for various use cases, musicological and beyond. It allows comparison of two files with encoded music scores, and will clearly highlight the differences between these encodings. In larger scores, it directs the user to variant spots using a Sunburst diagram. That way, comparing two music encodings becomes significantly easier. This is especially true, because MusicDiff correctly handles differences between written and sounding pitches – it resolves transposition and correctly considers key signatures. It also puts aside visual structures and artifacts to some degree, and thus helps to focus on "real" differences. This requires MusicDiff to be a tool guided by certain concepts – it implements *Beethovens Werkstatt's* model of identity and variation, which may or may not apply equally well to other contexts. Being released under an open license, however, it can be adjusted to other concepts. Even as it stands, MusicDiff is the authoritative tool for a semantic comparison of encoded music scores.

Work Cited

[1] Hunt, James W., and M. Douglas McIlroy. "An Algorithm for Differential File Comparison". Computing Science Technical Report, Bell Laboratories (June 1976).