Tools for Music Scholarship and their Interactions: A Case Study

David Lewis & **Ronald Woodley**
Birmingham Conservatoire
Birmingham City University
david.lewis@bcu.ac.uk

Tim Crawford, **Jamie Forth**,
Christophe Rhodes & **Geraint Wiggins**
Intelligent Sound and Music Systems
Goldsmiths, University of London

Abstract

In this paper, we introduce AMusE, a music reasoning framework built using Abstract Data Types, describing some of the advantages of such an approach. We illustrate this with a particular set of tools capable of tapping into the framework's capabilities: a score editor with the capability to edit some early music notations, and implementations of the SIA family of Music Information Retrieval tools. Putting these together, we discuss how the framework enables the general-purpose pattern-matching tool to operate on very specialised forms of notation. We conclude with a discussion of further work and the need for a more logic-based design and a more outward-looking deployment.

1 Introduction

Documents representing musical scores cannot, in general, be analysed or otherwise manipulated satisfactorily using standard, text-based content-analysis tools. A wide variety of music-encoding methods exist, often tailored to the specific requirements of a particular musical repertory or notational type. In general, however, there coexist many higher-level musical entities or concepts above this machine-readable content level which are understood by musicians as existing in common across repertories or notations (pitches, scales, rhythms, structural/formal features, etc.). It is therefore naturally desirable – at least for musicians – to be able to access, analyse and compare the low-level musical data using higher-level musical descriptions. Since the relationship between low-level musical encodings and higher-level musical concepts is, in general, implicit and contextual, rather than a simple and unambiguous mapping, this inevitably requires a degree of inferential musical reasoning.

The issue becomes all the more acute when dealing with unfamiliar or obsolete historical musical notations which have been intensely studied by specialists, but are little understood by musicians in general. Such is the case with lute tablature, the subject of the ECOLM project,[1] in which pitches are notated pragmatically by the physical position of the player's fingers upon the fingerboard of the instrument rather than by any abstract musical concept. Another example is the family of notations that evolved during the Renaissance commonly referred to as 'mensural' notation. Although superficially resembling modern staff notation, this assumes a number of conventions affecting both pitch and rhythm which give rise to significant differences from the modern system. Low-level encodings of the graphical features of such disparate notations are, for obvious reasons, likely to be completely incompatible; at the higher, music-conceptual level, however, searches and analyses could readily be performed across both repertories, so that a query expressed as an extract from a lute tablature, say, could be matched within a set of polyphonic scores encoded as mensural notation, or vice versa. A strong motivation for the work described here is exactly this need to make general musical concepts accessible from specialised information sources.

There exists, however, a similar problem at a higher conceptual level. Musicology and, more generally, expert interaction with music encompasses a wide variety of domains, ranging from psychological modelling and experimentation to editing, performing and composing. As these applications become more focussed, so the skills and vocabulary used become more specialised. This tends to fragment the range of computer tools, between those catering for a particular special task or trying to generalise at the risk of shallowness.

In the past, one of two solutions was commonly proposed for these difficulties – either everyone should use a common, extensible system or, alternatively, they should adopt a common interchange format between the various specialised software packages capable of encoding either the common ground between them or the union of all the features they support. The difficulties and limitations inherent in such approaches prompted us to construct instead a framework based on abstractions that allow the integrity of a tool or special representation to be preserved whilst those higher-level operations and data structures that *are* common can be shared. Separate applications can thus share not only data but also, more importantly, functionality. For academic rigour and the integrity of curated collections, as well as for copyright considerations, this is a powerful advantage.

In our paper, after a brief discussion of background work in music representation, we introduce AMusE, our music reasoning framework built using Abstract Data Types (ADTs), and outline some applications and analysis tools in which it is used. We then describe GSharp, score-editing software that is being extended for a digitisation project using historical music notations. We note how providing a

[1] `www.ecolm.org`

simple extension for GSharp to support the AMusE abstraction allows us to employ all those tools that use the framework to access and analyse the newly digitised data – thus, a music editing project still makes data available for other projects *without* having to lose sight of its specialised nature. Finally, we describe current developments making AMusE more powerful for automated reasoning and more available to others through Semantic Web technologies.

2 Music representation

The ability to analyse or search through musical corpora is highly dependent on the ability of the analyst (human or computer) to understand the way in which the musical information is presented. It is clear, then, that for an automated process to have any sophistication, it must have the music and associated data available in ways that make as much information as possible available to it.

Extensibility has long been an important feature of the more powerful tools for handling musical information digitally; the most comprehensive example is David Huron's Humdrum Toolkit (Huron, 1997). A recurring problem, however, is that this extensibility is usually at the level of the representation or its serialisation, leading either to parallel, mutually independent languages inhabiting the same carrier format or to an enforced and artificial commonality that can cause distortion and misrepresentation.

Certainly a naive encoding of graphical musical symbols is dangerous when the context is unavailable – as Selfridge-Field (1997b) points out: 'the apparent continuity of graphic symbols over centuries does not guarantee the same degree of continuity in practice'. Similarly, means of transmission for musical information can contain very different levels of detail about given parameters – for example, many tablatures supply relative pitch information only indirectly, providing no absolute pitch at all, with note durations left largely to the performer.

The *practical* necessity for abstraction in handling music digitally was acknowledged in the 1960s (e.g. Kassler, 1966), but its use to permit multiple, extensible underlying representations was first described by Wiggins et al. (1989) and called CHARM (Common Hierarchical Abstract Representation for Music). The advantage here is that, through ADTs, diverse musical structures with a variety of representations may be handled without the need for them to be prescribed *a priori*. This is illustrated by Smaill et al. (1993), using the same analytical method on two quite different representations of Debussy's *Syrinx* (yielding equivalent results) and also on a quarter-tone-based composition by Charles Ives. There was little further progress towards more extensive implementation of this idea until the creation of AMusE at the Intelligent Sound and Music Systems lab and of Music21 at MIT (Cuthbert and Ariza, 2010). Both systems use the object and class capabilities in their respective implementation languages to allow pluggable representations and tools, though neither system yet fully implements CHARM in the form originally described.

3 Components

3.1 The abstraction layer: AMusE

Within the core of AMusE, we define a variety of abstract classes for common musical concepts. These classes include `pitch` and `moment` for points in pitch and time respectively, and `pitch-interval` and `period` for pitch and time regions. Basic arithmetic and logical functions for interrogating or relating these permit addition, subtraction, comparison and equality, along with those required by the temporal interval relations of Allen (1984) (`meets`, `interval=`, `before`, `overlaps`, `during`, `starts` and `ends`). More 'typical' concrete classes, representing time on a simple number line, or pitch on the 12-note chromatic system (as in MIDI) or the diatonic system (as in Western European 'common music notation') are provided, but in general, the developer of an individual `implementation` is responsible for supplying appropriate concrete classes and behaviours.

3.2 AMusE implementations

An AMusE `implementation` provides a set of concrete (sub-)classes and methods that allow the use of a musical collection of some kind. Quite commonly, this is simply an import mechanism for a music representation such as MIDI or MusicXML, but, crucially, it may also be tailored for a specific repository: for example, the lab holds a collection of over 14,000 professionally-produced MIDI arrangements of pop music, which have special conventions of encoding, realised within AMusE as a separate implementation which primarily inherits behaviours from the MIDI implementation, but with added features. Currently, AMusE supports MIDI, MusicXML (Good, 2001), TabCode (an encoding for editing lute tablature; Crawford, 1991; Rhodes and Lewis, 2006) and some MEI (Roland, 2002) and `**kern` (Huron, 1997). As will be seen later, some of these implementations are realised through using the music representation abilities of GSharp.

Lack of information in a representation may come in a variety of forms, and the behaviour of tools in each case should be differentiated. An example of some of these differences should illustrate the point. A tool that requires tempo information will certainly be accommodated by a MIDI implementation, since that standard includes the parameter and specifies a default value should it be unspecified in a given file. In European notated music, however, precise tempo specifications are only available in as 'metronome marks' – annotations of the number of times a duration may fit in a minute – and consequently only available after the invention and successful marketing of the metronome in the nineteenth century. Less prescriptive verbal indications occur from the Baroque period onwards, but the association between these indications and any numerical value has generally been loose.[2]

[2]Fallows (2011) gives an example from the Polonaise of Bach's B minor orchestral suite, where the indication *lentement* is given to the violins, but *moderato* to the flute, though most modern readers would probably have assumed the former to refer to a slower tempo than the latter.

These issues around tempo are important: it is common for psychologically-motivated analysis to require timings in milliseconds, and in those cases, a decision must be made. What is expected of a system when tempo is requested might be something like this:

- If precise information is available, it is used
- If a verbal indication is given, indicate that no numerical tempo is available, offer either to provide the text used or guess at a tempo
- If no indication is given, indicate the fact. If enough is known about the music (if it is a dance movement, for example, there might be physical and generic limitations) to be able to make an educated guess, offer to do so
- If the notation is incapable of indications of tempi, indicate the fact.

Our feeling here is that the person best placed to 'guess' missing information is likely to be the one who collected the corpus, or at least created the specialised implementation. However, the client application may be best placed to decide how important a numerical tempo is: if the client is a music engraver, it would be better for it to know that there is no tempo present; if it is for an experiment, it may be necessary to omit instances without precise tempi, or an approximation may suffice.

This means that we must accept that, within AMusE, any parameter may be meaningless or underspecified in a given implementation or piece. In such cases an appropriate 'condition' is signalled.[3] A parameter might, alternatively, be meaningful but underspecified by the notation or by a particular source. In this case, it is expected that an implementation will still generate a condition, but may provide optional recovery strategies from which the user, or the function requiring the parameter, may choose. No pause in operation is required if recovery strategies are pre-selected. In this way, the implementation can take responsibility for the integrity of the information provided, whilst the user takes responsibility for the way in which it is used.

3.3 AMusE tools

Tools and applications developed with the AMusE data types include melodic segmentation based on multiple viewpoint models (Pearce and Wiggins, 2006), chord label generators (Rhodes et al., 2007), grouping analysis (Frankland and Cohen, 2004), pitch-spelling induction (Meredith, 2006b) and pattern search and discovery algorithms (Meredith et al., 2002; Meredith, 2006a; Forth and Wiggins, 2009). These work transparently on music encoded in any supported representation—any operation that is not directly available in a given implementation, as in the tempo examples given earlier, is handled in the condition system as described in the previous section.

[3]A feature of the Common Lisp condition system is that conditions may be signalled and recovery strategies suggested without rewinding the stack and losing system state – once a choice is made, the programme can carry on from exactly where the condition was raised.

A primary aim is to allow the interoperation of these tools and the combination and collation of their results so that a tool is never itself seen as the end point of the process, merely as an extra enabling step.[4]

3.3.1 Pattern discovery: SIA and SIA(TEC)

An illustrative example of a tool implemented with the framework is the SIA family of algorithms introduced by Meredith et al. (2002). These algorithms find repeating structures in multidimensional data, and are well suited to musical analysis, since their geometrical approach overcomes some of the limitations of string-based methods, especially for finding non-contiguous patterns. All SIA-derived algorithms are based around the detection of sets of points in a dataset that can be mapped, through translation, onto other sets of points in the dataset. In musical terms, its main use has been to find motivic repetition, whether transposed or at original pitch, and, since it is based on partial matches, it can still detect patterns in cases of extensive musical embellishment.

Although its development was motivated by a musical application, SIA is a general-purpose algorithm, with no domain-specific knowledge required. This makes it ideal for use with AMusE, since each dimension can be defined in terms of any ADT that has a sufficiently well-defined arithmetic. The use of this abstraction allows pattern-matching across implementations. For SIA(TEC), which groups patterns into equivalence classes, undefined values, such as duration (as opposed to inter-onset-interval) for tablature representations are easily accommodated, whilst heuristics that apply musical criteria for salience to filter the results have been developed (Forth and Wiggins, 2009).

4 GSharp: an extensible score editor

GSharp (Rhodes and Strandh, 2008) was developed in the 1990s by Robert Strandh as an interactive score editor, intended to be 'the Emacs of score editors'[5]. Its functionality was greatly expanded subsequently by Christophe Rhodes, who first developed experimental features for supporting less common music notations. It was clear from this work that the data model used by GSharp made it ideal to develop it as an editor for specialised and even mixed music notations.

Building on this, Birmingham Conservatoire and, subsequently, the Arts and Humanities Research Council have funded the development of GSharp to support the preparation of a digital edition of the complete theoretical works of Johannes Tinctoris (c.1435–1511), a Brabantine musician, composer and lawyer. The twelve surviving treatises, written in the 1470s, are acknowledged as one of the most significant and comprehensive sources for late medieval music notation and compositional process. The texts form a central focus for important research on musical aesthetics and

[4]A more extended illustration of our some of our intentions in this regard is given in (Lewis et al., 2006)

[5]This quote comes from the original home page of GSharp, `http://www.labri.fr/perso/strandh/Gsharp/index.html` (accessed July 2011). Please note, this is not the current home of the project.

reception at a crucial point in Western European culture, and demonstrate not only an exceptional technical command of music notation and theory, but also an intimate acquaintance with contemporary polyphonic practice, derived from a wide knowledge of the composers of his day and their music. Despite his importance, many of Tinctoris' treatises have never been translated into English.[6]

Unsurprisingly, given their subject matter, the treatises are copiously illustrated with musical examples and, partly as a result of the technical nature of their texts, these example use an exceptionally wide range of notational practices, some common, others reflective of more specialised points under discussion. The main aim is to develop an intuitive and easy-to-use package to replace a graphics package in producing publication-quality music output. Figure 1 shows a simple music example being transcribed using the software.[7] This example of mensural notation illustrates some of the complexities of interpreting rhythm in this music: that the first breve (the square note on the top line) lasts for two semibreves, whilst the later breve (the last of the extract) lasts for three can only be decided based on a combination of the patterns of neighbouring notes and the prevailing mensuration sign (the circle at the start of the extract).

Clearly there is a danger of the project producing specialised software that does not connect well with other tools or have impact beyond the editing of these treatises. Exporting and importing files in standard formats—MEI (Roland, 2002), CMME (Dumitrescu, 2001) and MusicXML (Good, 2001)[8]—helps, as does an open-source licence and publicly available source code, but, for reasons discussed above, this is not a complete solution.

5 Using the AMusE framework

The minimal set of functions needed to allow AMusE to interact with GSharp was implemented before the start of the Tinctoris project, giving several advantages: it gives AMusE access to several score encoding formats that would otherwise require separate implementations; it acts as a visualisation tool for the result of AMusE analyses and searches (including an experimental web application synchronising musical score page turns with a recording); and, in the other direction, it provides GSharp with optional AMusE-extended functionality, such as the ability to provide enharmonic pitch-spelling guidance in the manner of a word processor's spelling checker.

We have expanded our AMusE implementation to allow access to the newly-added mensural and chant notation in GSharp. This not only enables searching, segmentation and other analyses to be carried out on the Tinctoris material, but opens up new research avenues not applicable to other notations or encoding styles.

In this notation, rhythm and, to some extent, pitch is an implicit feature, deduced by reference to a combination of the note forms, their rhythmic context and a set of rules derived from contemporary practice and education. As we discussed in section 2, the ability to analyse music is dependent on the information available and, in cases where important information is largely implicit, analyses that rely on this information must either use an algorithm to deduce it prior to analysis, or the information must be explicitly provided by an editor.[9]

Without such sources of extra information, two other paths are available. Firstly, one can substitute less ambiguously notated versions of the same concept – since AMusE has abstractions for pitch, it is perfectly reasonable to implement a class that only knows named pitch, without the chromatic inflections that are so problematic in early music, while for our rhythmic representation, we can substitute note forms, knowing only the pattern of longs, breves and semibreves. Another path is to focus on other features. Since these transcriptions come from treatises largely devoted to music notation, the details of scribal practice are potentially of enormous interest and, since tools like SIA(TEC) are not bound to any specific musical features, we can study notational and scribal aspecs of the documents, such as the use of ligatures, the grouping of notes with dots of division or the use of indicators for solmisation (the precursors of accidentals and key signatures). As yet, there is insufficient edited musical material to make large-scale analyses meaningful, however the tools have been tested for this corpus, and early signs are promising.

6 Future work: more unexpected benefits

Currently, AMusE itself is limited by a number of factors: firstly, a concrete class is understood largely in terms of its specific, explicitly-defined behaviour (through method definition), limiting the extent to which automated reasoning can be deployed; secondly, since it is an offline, Lisp-based framework, as yet without any public release of code, it is inaccessible to many potential users and, even within our research group, is limited to those who can either use Lisp directly or in some way harness the Lisp code as a client module to their main language.

The first limitation may be addressed by abstracting not only the types of musical information being encoded, but also generalising, where possible, the information structures in which they are encoded, in the manner of Lewin's generalised intervals (Lewin, 1987). This would make it much easier for automated processes to model the logic of musical data structures and, we hope, infer novel patterns from them.

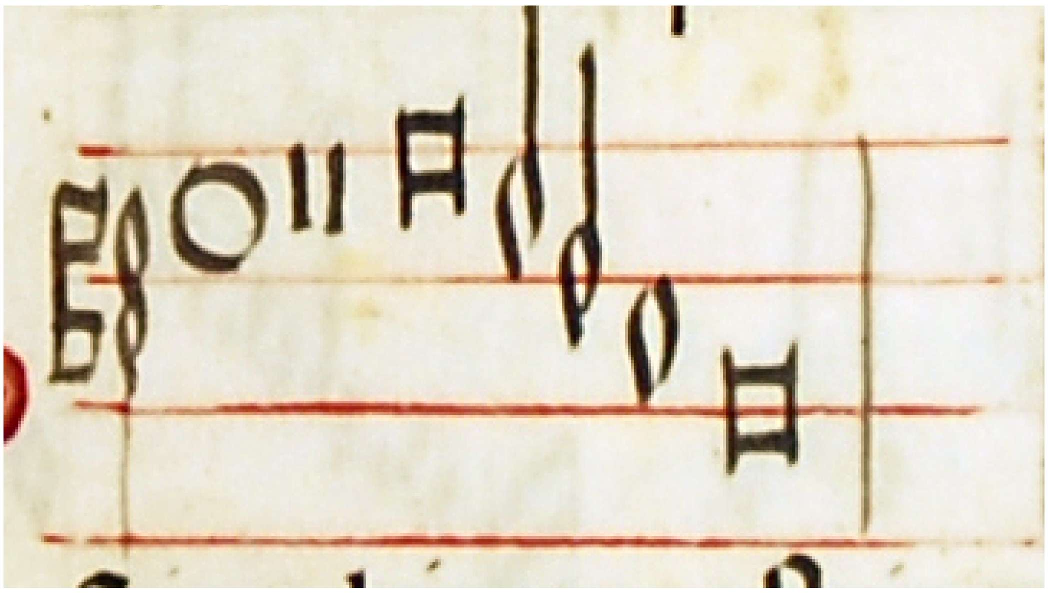
The second limitation can be only partly addressed by a simple release of code under a permissive licence. We believe that we also need to facilitate on-line access, and in particular, harness advances in the Semantic Web. The Music Ontology (Raimond et al., 2007) is a widely used vocabulary for describing information about music, ranging from high-level curatorial concepts such as musical works and

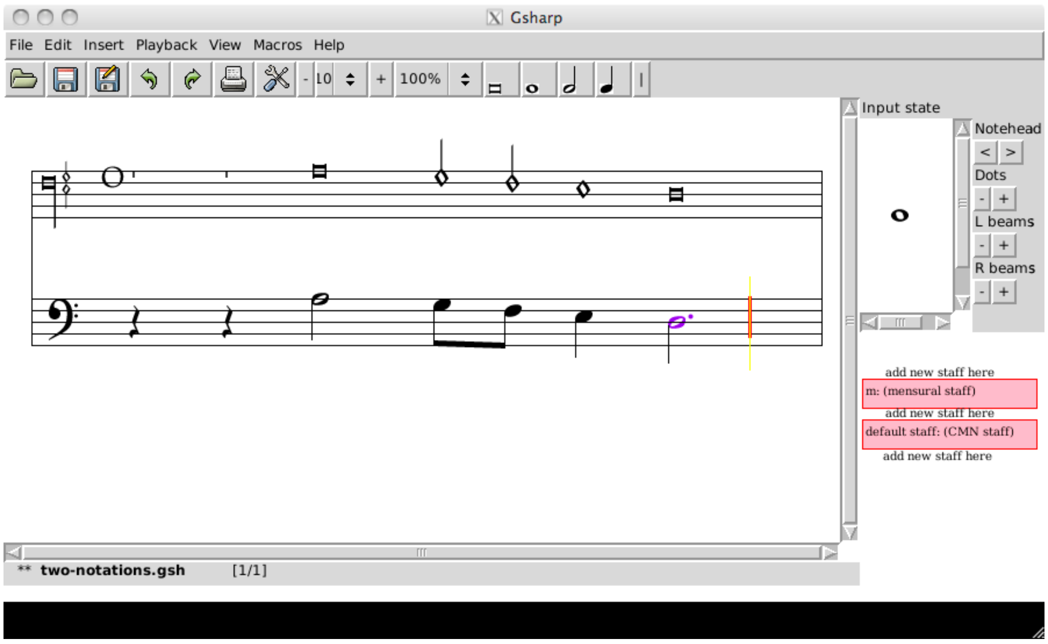
[6]For more information, see Tinctoris (2011).

[7]As will be clear from the screenshot, this goal of 'publication-quality output' has not yet been fully achieved.

[8]Much of the development of GSharp's MusicXML functionality results from a Google Summer of Code project by Brian Gruber.

[9]That such information will be provided cannot necessarily be assumed in this case, since the primary task is a visual edition of the notation, and such information is only partially affected by this.


(a) Bologna, Biblioteca Universitaria (I-Bu) Ms 2573, f. 75r



add new staff here
m: (mensural staff)
add new staff here
default staff: (CMN staff)
add new staff here

** two-notations.gsh [1/1]

(b) Editing in GSharp

Figure 1: The first music example from Tinctoris' *Tractatus alterationum* (1470s) (Tinctoris, 2011), as it appears in a contemporary source and as entered into GSharp, with a CMN transcription below. Rhythmic values have been quartered – a common practice to facilitate reading.

artists, down to the description of individual events; and the recent Segment Ontology (Fields et al., 2011) provides further formality for representing segmentation, a central concept in music analysis. The encoding of conceptual spaces (Raubal and Adams, 2010) on the Semantic Web is also potentially useful for music psychology-oriented research. The ability to process *semantic* data within AMusE will allow personal data sources to be enriched by tapping into growing on-line Linked Data resources, as well as give researchers the means to publish new data in a standardised form directly usable by other researchers or automated processes. Moreover, emerging standards for data access and processing within the Semantic Web, such as the SPARQL Query Language[10] and SPARQL Update[11] are attractive technologies for Web-based research, an idea that has received much recent attention within the Music Information Retrieval community (De Roure et al., 2011). There is a wide range of current research and existing infrastructure relevant to the goal of deploying AMusE services in an online environment. In turn, the design of AMusE, informed from both musicological and computational perspectives, is pertinent to ongoing efforts within the Semantic Web community towards appropriate knowledge-engineering in the musical domain. Finally for musicology, enabling openness and interoperability of both data and processing tools, as realised by AMusE, is vital in supporting musicological research in the digital domain.

Acknowledgements

The authors and the research reported here have been supported by the following grants: AHRC (AH/I003827/1), 'The Complete Theoretical Works of Johannes Tinctoris: A New Digital Edition', EPSRC (EP/H01294X/1), 'Information and neural dynamics in the perception of musical structure' and Andrew W. Mellon Foundation, 'MeTAMuSE: Methodologies and Technologies for Advanced Musical Score Encoding'.

References

Allen, J. F. (1984). Towards a general theory of action and time. *Artificial Intelligence 23*(2), 123–154.

Crawford, T. (1991). Applications Involving Tablatures: *TabCode* for Lute Repertories. *Computing in Musicology 7*, 57–59.

Cuthbert, M. S. and C. Ariza (2010). Music21: A toolkit for computer-aided musicology and symbolic music data. In *ISMIR*, Utrecht, Netherlands, pp. 637–642.

De Roure, D., K. R. Page, B. Fields, T. Crawford, J. S. Downie, and I. Fujinaga (2011). An e-research approach to web-scale music analysis. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 369*(1949), 3300–3317.

Dumitrescu, T. (2001). *Corpus Mensurabilis Musice "Electronicum"*: Toward a flexible electronic representation of music in mensural notation. In W. B. Hewlett and E. Selfridge-Field (Eds.), *Computing in Musicology*, Volume 12 of *Computing in Musicology*, Chapter 1, pp. 3–18. Cambridge, MA and Stanford, CA: MIT Press and Center for Computer Assisted Research in the Humanities.

Fallows, D. (accessed October 2011). Lento. In *Grove Music Online*, pp. `http://www.oxfordmusiconline.com/subscriber/article/grove/music/16410`. Oxford Music Online.

Fields, B., K. R. Page, D. De Roure, and T. Crawford (2011, July). The segment ontology: Bridging music-generic and domain-specific. In *Proceedings of the 3rd International Workshop on Advances in Music Information Research (AdMIRe 2011)*, Barcelona, Spain.

Forth, J. and G. A. Wiggins (2009). An approache for identifying salient repetition in multidimensional representations of polyphonic music. In J. Chan, J. W. Daykin, and M. S. Rahman (Eds.), *London Algorithmics 2008: Theory and Practice*, Texts in Algorithmics, pp. 44–58. London, UK: College Publications.

Frankland, B. W. and A. J. Cohen (2004). Parsing of melody: Quantification and testing of the local grouping rules of Lerdahl and Jackendoff's *A Generative Theory of Tonal Music*. *Music Perception 21*(4), 499–543.

Good, M. (2001). MusicXML: An Internet-Friendly Format for Sheet Music. In *XML Conference & Exposition*.

Huron, D. (1997). *Humdrum* and *Kern*: Selective feature encoding. See Selfridge-Field (1997a), pp. 375–401.

Kassler, M. (1966). Toward Music Information Retrieval. *Perspectives of New Music 4*, 59–67.

Lewin, D. (1987). *Generalized musical intervals and transformations*. New Haven and London: Yale University Press.

Lewis, D., T. Crawford, G. Wiggins, and M. Gale (2006). Abstracting musical queries: Towards a musicologist's workbench. In R. Kronland-Martinet, T. Voinier, and S. Ystad (Eds.), *Computer Music Modeling and Retrieval: Third international symposium, CMMR 2005*, Number 3902 in LNCS, Berlin, Germany, pp. 249–258. Springer.

Meredith, D. (2006a). Point-set algorithms for pattern discovery and pattern matching in music. In T. Crawford and R. Veltkamp (Eds.), *Proceedings of the Dagstuhl Seminar on Content-Based Retrieval*, Number 06171 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. IBFI, Schloss Dagstuhl.

Meredith, D. (2006b). The ps13 pitch spelling algorithm. *Journal of New Music Research 35*(2), 121–159.

[10]`http://www.w3.org/TR/sparql11-query/`
[11]`http://www.w3.org/TR/sparql11-update/`

Meredith, D., K. Lemström, and G. A. Wiggins (2002). Algorithms for discovering repeated patterns in multidmensional representations of polyphonic music. *Journal of New Music Research 31*(4), 321–345.

Pearce, M. T. and G. A. Wiggins (2006). The information dynamics of melodic boundary detection. In M. Baroni, A. R. Addessi, R. Caterina, and M. Costa (Eds.), *Proceedings of the 9th International Conference of Music Perception and Cognition*, Bologna, Italy, pp. 860–867. SMPC and ESCOM.

Raimond, Y., S. Abdallah, M. Sandler, and F. Glasson (2007). The music ontology. In *ISMIR*, Vienna, Austria, pp. 417–22.

Raubal, M. and B. Adams (2010). The semantic web needs more cognition. *Semantic Web 1*(1-2), 69–74.

Rhodes, C. and D. Lewis (2006). An editor for lute tablature. In R. Kronland-Martinet, T. Voinier, and S. Ystad (Eds.), *Computer Music Modeling and Retrieval: Third international symposium, CMMR 2005*, Number 3902 in LNCS, Berlin, Germany, pp. 259–263. Springer.

Rhodes, C., D. Lewis, and D. Müllensiefen (2007). Bayesian Model Selection for Harmonic Labelling. In *Mathematics and Computation in Music*, Berlin, Germany.

Rhodes, C. and R. Strandh (2008). Gsharp, un éditeur de partitions de musique interactif et personnalisable. *Document Numérique 11*(3–4), 9–28.

Roland, P. (2002). The Music Encoding Initiative (MEI). In *Music Applications using XML*, New York, NY, pp. 55–59. IEEE.

Selfridge-Field, E. (1997a). *Beyond MIDI*. Cambridge, MA: MIT Press.

Selfridge-Field, E. (1997b). Describing Musical Information. See Selfridge-Field (1997a), Chapter 1, pp. 3–37.

Smaill, A., G. Wiggins, and M. Harris (1993). Hierarchical Music Representation for Composition and Analysis. *Journal of Computing and the Humanities 27*, 7–17.

Tinctoris, J. (1 July, 2011). The Theoretical Works of Johannes Tinctoris. `http://earlymusictheory.org/tinctoris/tinctoris.html`. Ed. R. Woodley.

Wiggins, G., M. Harris, and A. Smaill (1989). Representing Music for Analysis and Composition. `http://www.doc.gold.ac.uk/~mas02gw/papers/EWAIM89.pdf`.