The Song Remains The Same: Lossless Conversion and Streaming of MIDI to RDF and Back

Albert Meroño-Peñuela[1,2] and Rinke Hoekstra[1,3]

[1] Department of Computer Science, Vrije Universiteit Amsterdam, NL
{albert.merono,rinke.hoekstra}@vu.nl
[2] Data Archiving and Networked Services, KNAW, NL
[3] Faculty of Law, University of Amsterdam, NL

Abstract. In this demo, we explore the potential of RDF as a representation format for digital music. Digital music is broadly used today in many professional music production environments. For decades, MIDI (Musical Instrument Digital Interface) has been the standard for digital music exchange between musicians and devices, albeit not in a Web friendly way. We show the potential of expressing digital music as Linked Data, using our `midi2rdf` suite of tools to convert and stream digital music in MIDI format to RDF. The conversion allows for lossless round tripping: we can reconstruct a MIDI file identical to the original using its RDF representation. The streaming uses an existing, novel generative audio matching algorithm that we use to broadcast, with very low latency, RDF triples of MIDI events coming from arbitrary *analog* instruments.

Keywords: MIDI, RDF, Music streams, Linked Data

1 Introduction

The Semantic Web is all about data diversity. The use of Linked Data principles [4, a.o.] boosted the growth of the Web of Data in a wide variety of domains. At the heart of Linked Data lies RDF, the Resource Description Framework (RDF)[4], which is a data model for expressing metadata *about any* resource. The Linked Open Data Cloud [6] and LOD Laundromat [1] consist of thousands of millions of such *metadata* statements. With a focus on what these RDF statements are *about*, we can distinguish between two types of Linked Data datasets. First of all, datasets that directly describe a domain – the statements are about real world facts. Examples are Bio2RDF, GeoNames and DBPedia.[5] The second type capture *metadata* only – the statements are about data artifacts that themselves describe data. Examples are L3S DBLP and Semantic Web Dogfood[6] on bibliographic data, and – in our case – Linked Data about *music*.

[4] See `http://www.w3.org/RDF`

[5] See `http://bio2rdf.org`, `http://www.geonames.org` and `http://dbpedia.org`, respectively

[6] See `http://dblp.l3s.de/d2r/` and `http://data.semanticweb.org`

In the case of the latter type of Linked Data, the actual *digital objects* represented by RDF statements are not (yet) machine interpretable from a semantic perspective (texts, images, video, audio), or require the use of legacy tooling to do so. Thus, although the resources that *describe* digital objects are promoted to first-class citizens of the Web – the objects themselves remain, to a large extent, non-interoperable.

One of the most prominent types of digital objects are *music* pieces. Music metadata has received a lot of attention by the Linked Data community. For example, DBTune.org links several music-related data sources on the Semantic Web [7], among them MusicBrainz [8], MySpace, and BBC Music. Other works have looked into publishing music recording metadata, and results from audio analysis algorithms [2]. Although the publication of music metadata about artists, songs, albums, musical events, and experimental results is an obvious contribution to the variety of Linked Data, music itself is currently only composed, published, and exchanged offline or using monolithic systems.

The fact that Linked Music Data is limited to metadata proper means that musicians cannot exploit Web technologies to their full potential. The composition and interpretation of music occurs in a data-silo setup, with no use of Web-global, machine readable, or resource-linkable formats. Consequently, existing musical data is not shared nor reused when musicians create, mix, and publish music. Reuse only happens at an abstract, hardly reproducible level – listening, reading scores, transcribing etc.. A Web-scale analysis of musical compositions and artifacts, and the relationships between them, their properties, fundamental nature, and intended meaning, is, with current Web resources, impossible.

Many works have studied symbolic representations for music suitable for the Web. The Notation Interchange File Format (NIFF), the Music Encoding Initiative (MEI), and MusicXML [3] are standards aimed at exchanging digital sheet music in a machine-readable way. Beyond music scores, MIDI (Musical Instrument Digital Interface) allows machines to exchange, manipulate and interpret musical events, and is to date the only symbolic music interchange format with wide adoption. The W3C Audio Working Group is working on a Web MIDI API[7] for "enabling web applications to enumerate and select MIDI input and output devices on the client system, and send and receive MIDI messages". This relates to the actual music in the same way that semantic web services relate to Linked Data. Some works presented at the International Workshop on Semantic Music and Media [8] (SMAM) specifically address the task of broadcasting chord and other recognized information from analog musical events in RDF on the Web.

In this paper, we are interested in investigating whether RDF is a suitable format to represent the *content* of digital music originally in MIDI format, and to explore the potential of such a representation. The novelty of the demonstration lies in three aspects. First, we study the essential concepts of MIDI to create a conversion workflow of MIDI music to RDF. Second, we invert this transformation, finding that it is possible to recompose the original MIDI from

[7] `https://www.w3.org/TR/webmidi/`

[8] See `http://semanticmedia.org.uk/smam2013/`

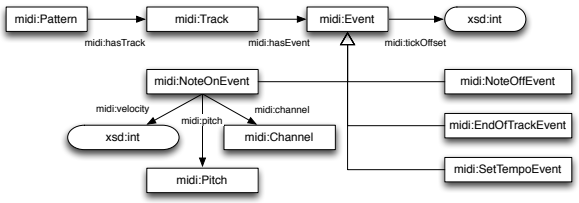

Fig. 1. Data model of MIDI concepts and their relations: patterns, tracks, events, and their attributes.

its RDF representation, thus providing a lossless round-trip. Third, we reuse a novel generative audio matching algorithm [5] to create RDF streams of music from any analog instrument, using MIDI as a discretization proxy.

The rest of the paper is organized as follows. In Section 2 we describe the basic concepts of MIDI, and we propose two methods: one to transform MIDI files to RDF and back; and another to stream live music from analog instruments encoded in RDF, using a novel generative audio matching algorithm. We also detail the key technology used in their implementations. In Section 3 we illustrate the contents of the demonstration, focusing on the lessons that can be learned from it, and we briefly discuss future work.

2 `midi2rdf`

MIDI is a standard that allows communication between a wide variety of electronic musical instruments, computers, and other devices. As a universal synthesizer interface, it abstracts musical events from the hardware, facilitating music exchange in a hardware independent manner.

Figure 1 depicts the fundamental MIDI concepts of *patterns*, *tracks* and *events*. A pattern acts as a high level container of a musical work (e.g. a song). It contains global MIDI metadata, such as resolution and format, and a list of tracks. Tracks are the logical divisions of a pattern, and typically represent musical instruments that play simultaneously. Finally, tracks contain events, which are sequential occurrences of instrumental actions, such as "a note starts playing" (`mid:NoteOnEvent`) or "a note stops playing" (`mid:NoteOffEvent`). MIDI defines 28 different types of events that control various aspects of the musical play. All MIDI events have associated a *tick* offset, which indicates the relative distance between events in discrete units. Each event type has its own attributes. For example, a `mid:NoteOnEvent` has a pitch (the note to be played), a velocity (its duration), and a channel (the tone or "instrument").

We use the data model of Figure 1 to implement a conversion workflow transforming the internal components of a MIDI to an RDF model. `midi2rdf`[9] is an open-source suite of programs for encoding and decoding digital music between

[9]Source code available at `https://github.com/albertmeronyo/midi2rdf`.

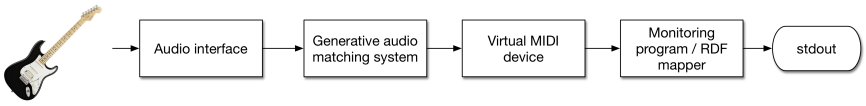

Fig. 2. Workflow for converting instrument analog signals to MIDI-like RDF streams.

the MIDI and RDF formats. `midi2rdf` consists of several tools addressing two tasks: *conversion*, and *streaming*.

Conversion. The conversion programs are `midi2rdf` and `rdf2midi`. Both are Python scripts written on top of the `python-midi`[10] (a comprehensive abstraction over MIDI that facilitates reading and writing contents from and to MIDI files) and `rdflib`[11] libraries. We use `midi2rdf` to read and process the patterns, tracks and events of any MIDI file, and transform them to an RDF graph as shown in Figure 1. The inverse process is carried out by `rdf2midi`, which transforms any RDF graph compliant with the model of Figure 1 back to MIDI. Auxiliary tools for displaying contents of MIDI files, and for playing RDF files with MIDI contents in just one command, are also supplied.

Streaming. To stress the importance of a symbolic and Web-friendly representation of music during a *performance*, we propose the workflow depicted in Figure 2. We implement this workflow in the program `stream-midi-rdf`, also included in `midi2rdf`. The basic idea is to create a stream of RDF data according with the model of Figure 1, using a discretization of input analog instruments through MIDI[12]. A key step here is the generative audio matching system: an algorithm that can translate analog, continuous input audio to digital, discrete MIDI events. For this, we use the algorithm provided in Guitar MIDI 2[13] [5]. We route the output of this algorithm to an IAC (Inter-Application Communication Driver) virtual MIDI device that can be read by user libraries. In this case, we use `pygame`[14], a set of Python modules designed for writing video games with good support of MIDI interfaces. Finally, we use again `rdflib` to create streams of RDF data symbolizing the read MIDI events.

3 Demonstration

The demonstration of the `midi2rdf` suite will consist of two parts: a validation on the lossless round-trip conversion between MIDI and RDF; and a live performance, with real instruments, showing the RDF streaming workflow of Figure 2. In the first part, visitors of the demonstration will be able to judge by

[10] See `https://github.com/vishnubob/python-midi`
[11] `https://github.com/RDFLib/rdflib`
[12] Instrument and audio interfaces can be replaced by any analog input (microphone).
[13] See also `http://www.jamorigin.com/products/midi-guitar/`
[14] See `http://pygame.org/`

themselves whether the round-trip conversion is lossless or not in a variety of MIDI files. They will learn the basic features of symbolic music representation, as well as the atomic concepts of MIDI, and how this explains that both representations are equivalent. We will show why the translation of these concepts to RDF is challenging, especially regarding issues like preserving the order of MIDI events. Furthermore, visitors will be challenged to think over a Web of Linked Music Data beyond musical metadata. We are interested in the advantages that Linked Data provides over current approaches like MusicXML, in e.g. uniquely (and globally) representing notes using URIs – thus enabling a more standard way of music comparison and combination –, the content of their dereferencing, and the ability to query across (streamed) music – something that is not possible with current standards.

In the second part, visitors will see the state of the art in generative audio matching in action, witnessing how a live performance with a real instrument is turned into an RDF stream of triples describing MIDI events of the music they hear. The audience will learn that, despite the precision and performance of these methods, detecting correct notes at real-time latency is a problem with an impact on the usefulness of such RDF streams. We hope to engage in discussions about the pros and cons of discretizing music this way.

We plan to improve this work in several ways. First, future releases of `stream-midi-rdf` will issue PROV triples indicating which `prov:Agent` performed which `prov:Activity` on what `prov:Entity`, in order to better document performances. Second, we will extend `midi2rdf` and the libraries it builds on to better support less common types of MIDI events. Finally, we plan to use the proposed tools to deploy a Linked Dataset of MIDI resources and link it to the LOD cloud.

References

1. Beek, W., Rietveld, L., Bazoobandi, H.R., Wielemaker, J., Schlobach, S.: LOD Laundromat: A Uniform Way of Publishing Other People's Dirty Data. In: The Semantic Web – ISWC 2014 (2014)
2. Cannam, C., Sandler, M., Jewell, M., Rhodes, C., d'Inverno, M.: Linked data and you: Bringing music research software into the semantic web. Journal of New Music Research 39(4), 313–325
3. Good, M.: MusicXML: An Internet-Friendly Format for Sheet Music (2001), `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.118.5431`
4. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan and Claypool, 1st edn. (2011)
5. Kristensen, O.: Generative audio matching game system (Mar 3 2011), `https://www.google.com/patents/WO2010142297A3?cl=en`, wO Patent App. PCT/DK2010/050,132
6. Max Schmachtenberg, Christian Bizer, A.J., Cyganiak, R.: Linking Open Data cloud diagram 2014. `http://lod-cloud.net/` (2014)
7. Raimond, Y., Sandler, M.: A Web of Musical Information. In: Ninth International Conference on Music Information Retrieval (ISMIR2008) (2008)
8. Swartz, A.: Musicbrainz: a semantic web service. IEEE Intelligent Systems 17, 76–77 (2002)