Musical File Formats and Conversion

Quentin Stievenart

March 11, 2013

Outline

Existing (Open) Musical Notation File Formats

- MusicXML (`.xml`, `.mxml`)
- MuseScore (`.mscx`, `.mscz`)
- CapXML (`.capx`)
- MIDI (`.mid`)
- NIFF (`.nif`)

Existing Optical Music Recognition Software

Software	Fee	MusicXML	MIDI	NIFF	CapXML
Audiveris	Free	yes	no	no	no
OpenOMR	Free	no	yes	no	no
capella-scan	$250	yes	yes	no	yes
Forte Scan	$29	yes	yes	no	no
PhotoScore	$250	yes	yes	yes	no
SharpEye	$180	yes	yes	yes	no
SmartScore	$399	yes	yes	no	no
MusicReader	$59	no	no	no	no

Format Choice

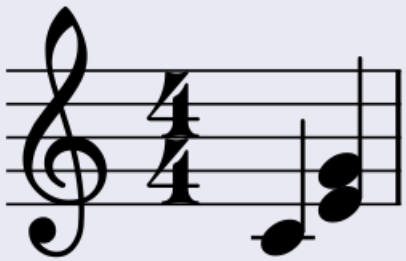
- MusicXML and MIDI are the most supported formats among musical notation software
- MIDI is **not** designed for notation, but for playback. There is a lot of information loss when representing a score with MIDI
- NIFF was designed to be a standard in music notation file formats, but it is now unmaintained and considered obsolete, their website closed in 2006
- Free notation software like MuseScore can do the translation between CapXML, MusicXML and MIDI
- The final choice is to only support MusicXML. If other formats are needed, they can be translated by softwares like MuseScore

MusicXML

- *MusicXML was designed from the ground up for sharing sheet music files between applications, and for archiving sheet music files for use in the future.*
  (`http://www.makemusic.com/musicxml`)
- History:
  - 2001: first beta
  - 2004: version 1.0
  - 2008: version 2.0
  - 2011: version 3.0
- Supported by most musical software (OMR, scorewriting programs, sequencers)
- Defined in well commented DTD files

Conversion

Classic notation



MusicXML

```
<score-partwise version="3.0">
  ...
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        ...
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>1</duration>
        ...
      </note>
      <note>
        <pitch>
          <step>E</step>
          <octave>4</octave>
        </pitch>
        <duration>1</duration>
        ...
      </note>
      ...
  </part>
</score-partwise>
```

Overtone notation

```
(defprog P1
  (bar
    (play-seq
      (play :C4 1)
      (play-chord
        (play :E4 1)
        (play :G4 1)))))

(defsong foo
  [P1 sampled-piano])
```

Conversion – Elements

- Elements we need to extract
  - Notes and their duration
  - Chords
  - Rests
  - Time signature
  - Tempo
  - Number of divisions
- Elements we don't need:
  - Clefs: needed in classical notation due to its graphic nature (*which line correspond to which note?*), useless if we know which note to play
  - Key signature: same reasons as the clefs

Conversion – Intermediate datastructure

- Instead of directly doing the translation, simple Clojure *records* are used as an intermediate datastructure

```
(defrecord song [time-signature tempo progs])
(defrecord prog [id bars])
(defrecord bar [number notes])
(defrecord chord [notes])
(defrecord note-seq [notes])
(defrecord note [descr duration])
```

- Records are easier to manipulate than s-expressions
- Allows to change the notation without changing the parser

Conversion – Notes

- Straightforward conversion
- The `type` element is only needed for graphical representation
- The meaning of the `duration` depends on the `divisions` attribute of the measure, which is explained later

MusicXML

```
<note>
  <pitch>
    <step>C</step>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
  <type>quarter</type>
</note>
```

Overtone notation

```
(play :C4 1)
```

Conversion – Rests

- Rests are simply notes without a `pitch` element, and a `rest` instead
- In the Overtone notation, a rest is represented by the keyword `:rest`

MusicXML

```
<note>
  <rest/>
  <duration>1</duration>
  <type>quarter</type>
</note>
```

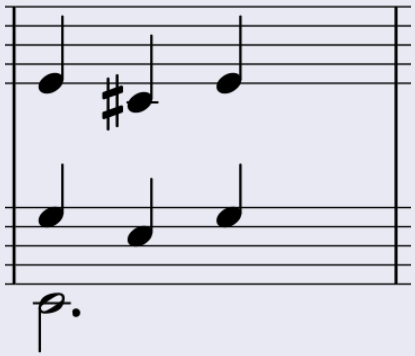
Overtone notation

```
(play :rest 1)
```

Conversion – Voices

- A sequence of note can be grouped into a *voice*
- Parse voices independently and play them as a chord of sequences

Classic notation



Overtone notation

```
(bar
  (play-chord
    (play-seq
      (play-note :E5 1)
      (play-note :C#5 1)
      (play-note :E5 1))
    (play-seq
      (play-note :G4 1)
      (play-note :E4 1)
      (play-note :G4 1))
    (play-note :E3 3)))
```

Conversion – Time signature

- In MusicXML, time signature is set per measure
- In most case, the time signature is set in the first measure, and remains the same (without having to write it) for the following measures
- The overtone notation use only one time signature for the entire song

MusicXML

```
<time>
  <beats>4</beats>
  <beat-type>4</beat-type>
</time>
```

Overtone notation

```
(defsong foo
  {:time-signature [4 4]}
  ...)
```

Conversion – Tempo

- Similar as for the time signature

MusicXML

```
<sound tempo="60"/>
```

Overtone notation

```
(defsong foo
  {:tempo 60}
  ...)
```

Conversion – Divisions

- *The divisions element indicates how many divisions per quarter note are used to indicate a note's duration. For example, if duration = 1 and divisions = 2, this is an eighth note duration.* (MusicXML's `attributes.mod`)
- In the Overtone notation, there is no such things as divisions:
  - A duration of 1 is a quarter note (which lasts 1 beat)
  - A duration of 1/2 is an eighth note (which lasts 1/2 beat)
  - ...
- MusicXML's divisions comes from MIDI, where the duration had to be stored as an integer
- We can simply store the number of divisions while building the internal structure, and build the notes with a duration of `duration/divisions`

Conversion – Chords

- In MusicXML, if a note have a `chord` element, it means that it is in the same chord as the **previous** note
- While converting, we have to keep the last chord, to potentially add notes to it

MusicXML

```
<note>
  <pitch>
    <step>E</step>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
</note>
<note>
  <chord/>
  <pitch>
    <step>G</step>
    <octave>4</octave>
  </pitch>
  <duration>1</duration>
</note>
```

Overtone notation

```
;; Intermediate structure
(->chord
  [(->note :E4 1)
   (->note :G4 1)])
;; Generated code
(play-chord
  (play :E4 1)
  (play :G4 1))
```

References and links

- M. Good, *MusicXML: An Internet-Friendly Format for Sheet Music*, 2001
- MusicXML DTD: `http://www.makemusic.com/musicxml/specification/dtd`
- Audiveris website: `http://audiveris.kenai.com/`