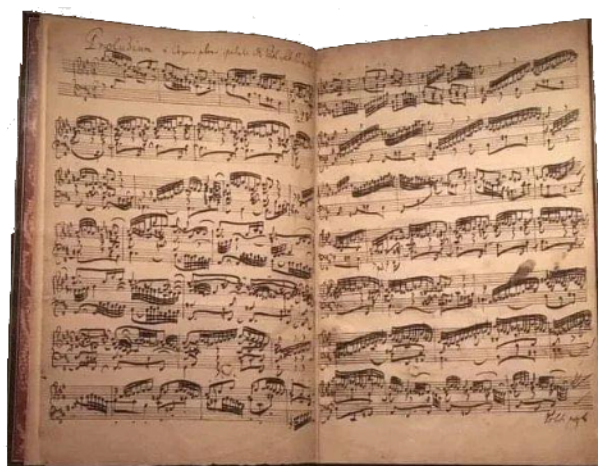


Capitolo 4

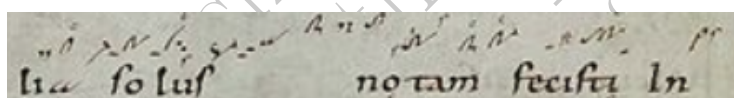
Rappresentazione e analisi della musica



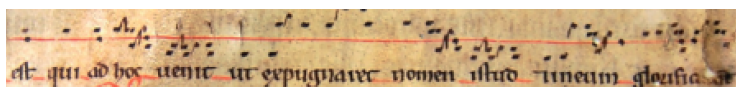
Per *rappresentazione* della musica intendiamo i modi in cui la musica può essere descritta. Ovviamente la musica, nel senso più stretto del termine, è una entità sonora che percepiamo attraverso l'udito. La rappresentazione più classica è quella della partitura (spartito musicale) che è una rappresentazione simbolica risultato di una lunga evoluzione storica. Ma non è l'unica. In particolare le moderne tecnologie permettono di memorizzare la musica in formati digitali che possono spaziare su vari piani, dalla rappresentazione MIDI a quella audio. Al formato MIDI abbiamo già dedicato un intero capitolo. In questo capitolo approfondiremo il formato MusicXML e la libreria Python music21. Parleremo inoltre di rappresentazioni specifiche.

4.1 Partitura

La rappresentazione simbolica della musica è fatta, come dice il nome, per mezzo di simboli. I *neumi* sono i primi esempi di rappresentazione simbolica; sono segni che venivano posti sopra i testi dei canti ecclesiastici, a partire dal IX secolo circa, per dare indicazioni su come il canto doveva essere eseguito. L'evoluzione storica dei neumi è passata attraverso varie fasi. Dai segni sopra le parole



si è passati all'utilizzo dei segni su un rigo,



poi su 4



e infine su 5 per arrivare all'attuale pentagramma, mostrato nella figura seguente; i neumi stessi hanno subito una notevole evoluzione passando dai segni alla notazione quadrata fino ad arrivare alla moderna rappresentazione tonda¹.



La rappresentazione tramite partitura è specifica per lo studio della musica da parte dei musicisti esecutori. Tuttavia per altri scopi non è la più comoda; è possibile immaginare svariate altre rappresentazioni costruite tramite elementi facilmente rappresentabili in formato digitale e efficientemente manipolabili da un computer. Per formato digitale non intendiamo una semplice rappresentazione digitale dell'immagine di una partitura ma di formati particolari che permettano di implementare agevolmente delle specifiche operazioni.

4.2 MusicXML

MusicXML è la specializzazione del formato XML (eXtended Markup Language) alle partiture musicali. Il suo uso principale è quello di creare un formato che permetta di scambiare facilmente spartiti musicali. Il formato MusicXML permette di descrivere tutti gli elementi di una partitura in modo tale da poter ricreare la partitura stessa. Rispetto alla partitura standard ha il vantaggio di poter essere facilmente manipolata tramite software.

In questa sezione forniremo una breve descrizione. Una trattazione completa e dettagliata del formato MusicXML può essere trovata visitando il seguente sito web:

<https://www.w3.org/2021/06/musicxml40/tutorial/introduction/>

4.2.1 XML


Lo standard XML (eXtensible Markup Language) è un linguaggio detto di “marcatura” che serve a descrivere dati strutturati. La struttura dei dati è descritta tramite cosiddetti *tag* (etichette) che vanno aperti e (obbligatoriamente) chiusi con la seguente sintassi:

`<NOMETAG> </NOMETAG>`

Un tag può contenere altri tag al suo interno creando così una struttura gerarchica ad albero. Un tag aperto all'interno di un altro tag deve essere chiuso all'interno del tag in cui è stato aperto (altrimenti non si avrebbe la struttura ad albero).

Un tag può opzionalmente avere degli attributi che specificano informazioni aggiuntive. Gli attributi vengono specificati all'apertura del tag.

¹Per chi vuole approfondire il seguente link porta a un videocorso di storia della musica di Elia Bertolazzi, noto sul web come Mastro Elia. In particolare gli episodi 22 e 33 parlano della notazione musicale. https://www.youtube.com/playlist?list=PLpONvz1ts9_4bN-nmPTDq7Sz1sQMDidb.



```

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
  "-//Recordare//DTD MusicXML 4.0 Partwise//EN"
  "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="4.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>G</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>

```

Figura 4.1: “Ciao mondo” in Music XML

<NOMETAG attributo1="valore1" attributo2="valore2" ...>

Come si può notare anche da queste poche nozioni lo standard XML è molto generale. Un DTD (Document Type Definition) serve a definire la struttura di un documento XML tramite delle regole sintattiche: specifica quali sono gli elementi (tag), quale è la relazione gerarchica fra di essi, in che ordine devono apparire e quali elementi e quali attributi sono opzionali e quali obbligatori. Gli XSD (XML Schema Definition) sono una evoluzione dei DTD.

Un documento XML inizia sempre specificando, tramite un link web, lo schema (XSD) o il tipo (DTD) utilizzato. Le definizioni degli schemi e dei tipi sono forniti pubblicamente dal W3C tramite github, all’indirizzo

<https://github.com/w3c/musicxml/releases/tag/v4.0>

Il file compresso contiene sia i DTD che gli XSD. Si noti che dalla versione 4.0 i DTD sono obsoleti e quindi è necessario usare gli XSD.

4.2.2 “Ciao mondo” in Music XML

Quando si approccia un nuovo linguaggio di programmazione solitamente si parte da un esempio di programma che è il più semplice possibile e che serve solo a fornire in output un saluto (“Ciao mondo”). In modo analogo possiamo iniziare la descrizione del formato MusicXML dalla codifica più semplice possibile: una sola battuta con una sola nota. La Figura 4.1 mostra tale rappresentazione.

Come tutti i file XML inizia con una linea che specifica la versione di XML usata e il tipo di codifica per i caratteri. La seconda linea specifica il DTD:

La vera codifica inizia alla terza linea con `<score-partwise version="4.0">` che specifica l’elemento radice del file XML che in questo caso è uno `<score-partwise>` e anche la versione di MusicXML che viene utilizzata. Oltre a `<score-partwise>` esiste anche l’elemento `<score-timewise>`.

Il file XML inizia con una lista, `<part-list>`, che elenca le varie parti componenti la partitura. In questo esempio la lista è quella minima possibile: contiene un solo elemento `<score-part>` di cui viene specificato il nome con l'elemento `<part-name>`. Ogni parte deve avere un identificativo univoco che è una stringa che inizia con una lettera. Una scelta molto comune è quella di far iniziare l'identificativo con "P" e poi usare un intero progressivo. In questo esempio abbiamo una sola parte.

L'elemento `<attributes>` permette di specificare vari parametri relativi alla parte in cui è incluso. In particolare permette di specificare la chiave, l'armatura, il tempo e la granularità che si vuole raggiungere nella rappresentazione. L'elemento `<divisions>` indica la granularità: specifica l'elemento minimo in termini di divisione della semiminima. In questo caso non ha molta importanza in quanto c'è solo una semibreve quindi non scendiamo mai al di sotto di una semiminima, pertanto il valore specificato è 1. L'elemento `<key>` specifica l'armatura, cioè le alterazioni in chiave indicando il numero di bemolle (`<0>`) o di diesis (`>0`) da utilizzare con l'attributo `<fifth>` mentre l'elemento `<clef>` permette di specificare la chiave, in questo caso una chiave di violino (`sign=G`) sul secondo rigo (`line=2`). L'elemento `<time>` specifica il tempo, in questo caso un 4/4.

Infine ci sono le note, in questo caso una sola, specificate con l'elemento `<note>`. Per le note si specifica l'altezza, indicando la nota, in questo caso C, e l'ottava, in questo caso 4, la durata, come multiplo dell'elemento unitario, in questo caso 4 in quanto come elemento unitario abbiamo usato la semiminima visto che `<divisions>` vale 1. La durata viene anche specificata in modo esplicito con l'attributo `<type>`, anche se può essere derivata dall'attributo `<duration>`.

4.2.3 partwise vs score-wise

I dati di un file XML sono organizzati in una struttura gerarchica. Una partitura musicale ha una struttura che può essere vista "verticalmente", se pensiamo ad una singola battuta per tutti i righi musicali, oppure "orizzontalmente" se pensiamo ad ogni rigo musicale composto di tante battute. Per questo motivo esistono 2 diversi schemi: `<score-partwise>` e `<score-timewise>`. Il primo dà priorità all'organizzazione gerarchica vista orizzontalmente, cioè le battute musicali sono contenute nella parte, mentre il secondo dà priorità all'organizzazione gerarchica vista verticalmente, cioè le parti, i singoli rigi musicali, sono contenuti nella "battuta". In funzione della specifica applicazione un approccio può essere preferibile all'altro. Le due possibilità sono equivalenti ed è ovviamente possibile convertire da un formato all'altro.

La struttura generale di un file XML è, nel caso di un'organizzazione basata sulle parti, la seguente

```
<!ELEMENT score-partwise (%score-header;, part+)>
<!ELEMENT part (measure+)>
<!ELEMENT measure (%music-data;)>
```

mentre, nel caso di un'organizzazione basata sul tempo, è:

```
<!ELEMENT score-timewise (%score-header;, measure+)>
<!ELEMENT measure (part+)>
<!ELEMENT part (%music-data;)>
```

Come si può notare l'unica differenza è che nel primo caso l'elemento "part" contiene le battute, mentre nel secondo caso l'elemento "measure" (battuta) contiene la parti. In entrambi i casi l'elemento di livello più interno contiene i dati musicali veri e propri, definiti come:

```
<!ENTITY % music-data
"(note | backup | forward | direction | attributes |
```

```
harmony | figured-bass | print | sound | barline |
grouping | link | bookmark)*">
```

Inoltre c'è una sezione di intestazione che serve a specificare informazioni generali:

```
<!ENTITY % score-header
"(work?, movement-number?, movement-title?,
identification?, defaults?, credit*, part-list)">
```

4.2.4 Un esempio più complesso

<https://www.w3.org/2021/06/musicxml40/tutorial/notation-basics/>

4.2.5 Documentazione Consorzio W3

<https://www.w3.org/2021/06/musicxml40/>

4.3 Libreria Python: music21

Music21 (web.mit.edu/music21) è una libreria Python che offre strumenti per la manipolazione della musica in formati simbolici. Oltre ad altre funzioni permette di gestire file in formato MusicXML. Music21 utilizza un approccio ad oggetti per gestire i dati musicali. Gli oggetti sono organizzati in moduli; i nomi dei moduli iniziano con una lettera minuscola mentre i nomi degli oggetti iniziano con una lettera maiuscola. Ad esempio il modulo `note` contiene l'oggetto `Note` che, come suggerisce il nome, rappresenta una nota musicale e quindi possiamo considerarlo come l'oggetto di base con il quale si costruiscono oggetti più complessi fino a rappresentare intere partiture. Il modulo `note` contiene vari altri oggetti basilari utili per rappresentare tutti gli elementi di una partitura, come, ad esempio `Lyric`, `Rest` o anche `Duration`, `Volume` che in realtà sono parte di sottomoduli (risp. `duration` e `volume`).

Nel seguito descriveremo brevemente le nozioni di base di Music21. Si faccia riferimento alla documentazione ufficiale, web.mit.edu/music21/doc/, per informazioni più dettagliate.

4.3.1 Oggetti Note, Pitch, Duration

Per creare un oggetto `Note` usiamo il costruttore, `note.Note("A4")`, e ovviamente possiamo salvare l'oggetto in una variabile: `a = note.Note("A4")`. Quindi possiamo accedere agli attributi dell'oggetto, ad esempio, `a.name` restituirà `A`, mentre `a.octave` restituirà `4`. In realtà entrambe queste informazioni sono memorizzate nell'oggetto `pitch.Pitch`, cioè l'oggetto `Pitch` del modulo `pitch`, al quale possiamo accedere con `a.pitch`. Nell'oggetto `Pitch` ci sono anche altre informazioni, come ad esempio la frequenza: `a.pitch.frequency` restituirà `440`. Ovviamente i singoli attributi possono anche essere cambiati dopo la creazione della nota. Ad esempio `a.note.octave=3` cambierà l'ottava della nota; di conseguenza la frequenza verrà cambiata da `440` a `220`. Analogamente, `a.pitch.frequency=329.6275569` cambierà la frequenza della nota e di conseguenza il "nome" simbolico, in `E4`. L'attributo `pitchClass` vale `0` per il `Do`, `1` per il `Do#` e così via fino a `11` per il `Si`. L'attributo `midi` invece fornisce il codice MIDI della nota. Per `a = note.Note("A4")` avremo `a.pitch.midi` pari a `69`. L'oggetto `duration.Duration` memorizza la durata della nota. Le durate vengono specificate in multipli o sottomultipli (in entrambi i casi anche non interi) di semiminime (cioè note di $1/4$). Per le note dalla croma in su, si possono utilizzare i nomi ("maxima", "longa", "whole", "half", "quarter", "eight") mentre dalla semicroma in giù si usano nomi con in numeri ("16th", "32nd", "64th", ... fino addirittura a "2048th"). Oltre ai nomi si può usare

il fattore moltiplicativo. Ad esempio `duration.Duration(1.5)` crea una durata corrispondente a un semiminima con il punto. Chiaramente usando l'opportuno moltiplicatore si possono rappresentare durate arbitrarie. L'attributo `quarterLength` fornisce la durata in termini di numero di semiminime (multipli non interi o frazioni). L'attributo `dots` specifica i "punti" musicali che allungano la durata della metà. Un oggetto `Note` di fatto è costituito da un oggetto `Pitch` che specifica l'altezza della nota e da un oggetto `Duration` che specifica la durata.

4.3.2 Liste e flussi di note

Le note possono essere raggruppate in liste e flussi per creare degli spartiti. Una lista di note è sostanzialmente un array con gli oggetti `<music21.note.Note>`. La gestione della lista ovviamente può essere fatta con gli strumenti che Python mette a disposizione. Ad esempio, possiamo iterare su tutte le note di una lista con `for n in noteList: print(n)`. I flussi (stream) di note sono contenitori di oggetti musicali, come note, accordi, chiave, indicazioni di tempo, etc. Sono molto simili alle liste. Gli oggetti contenuti in un flusso sono normalmente organizzati in funzione del tempo: per ogni oggetto viene specificato l'offset rispetto all'inizio del flusso, in termini di quarti. Ad esempio in una battuta di 4/4 con due minime, la prima avrà offset 0.0, mentre la seconda 2.0. Diversamente da una lista, un flusso può contenere altri flussi creando così un'organizzazione gerarchica. La classe `Stream` ha varie sottoclassi come ad esempio `Measure` per le battute, `Part` per le parti. Ad esempio se abbiamo due battute in un flusso `Part`, la prima avrà offset 0.0 e la seconda 4.0, sempre considerando un tempo di 4/4. Un'altra sottoclasse è `Score`. Quindi uno spartito potrebbe essere rappresentato da un flusso `Score`, organizzato in sottoflussi di tipo `Part`, ognuno dei quali corrispondente, ad esempio, ad uno strumento, e ogni parte fatta di un flusso di `Measure`.

4.3.3 Show

Il metodo `show` permette di "mostrare" un oggetto musicale. Il metodo sfrutta programmi esterni per visualizzare la musica o anche suonare il midi. Ad esempio `a=note.Note("C4"); a.show()` lancerà il programma predefinito per gestire file musicxml al quale passerà un file musicxml che descrive la nota C4. Analogamente `a.show('midi')` lancerà il programma predefinito per la gestione di file midi al quale verrà passato un file midi con la nota C4. Ovviamente il metodo `show` potrà essere usato su oggetti più complessi che descrivono intere partiture musicali.

4.4 Rappresentazioni specifiche

In alcuni casi può essere comodo usare delle rappresentazioni specifiche. Ad esempio una melodia potrebbe essere rappresentata tramite una sequenza di simboli alfanumerici. In questo modo la melodia intera sarebbe rappresentata da una stringa di caratteri e questo permetterebbe la sua manipolazione attraverso algoritmi che operano su stringhe. Due stringhe potrebbero essere "confrontate" per vedere quanto sono "diverse". Questo potrebbe essere utile ad esempio per individuare plagi musicali.

Un esempio di rappresentazione specifica è la `tinyNotation` fornita da Music21. L'istruzione `s = converter.parse('tinyNotation: 4/4 C4 D4 E4 F4 G4 A4 B4 c4')` rappresenta la partitura di una scala ascendente di Do maggiore, fatta di 8 semiminime. In questo caso la stringa "4/4 C4 D4 E4 F4 G4 A4 B4 c4" è la rappresentazione simbolica. In questa notazione la lettera "C" (maiuscola) indica il Do centrale (C4); la lettera "c" (minuscola) invece indica il Do dell'ottava superiore (C5). Il 4 dopo la C invece non indica l'ottava, bensì la durata: 4 corrisponde ad una semiminima (8 a una croma, 16 a una semicroma, etc.). Le altre ottave sono CC e CCC verso il



Figura 4.2: Tema di “O sole mio”, in Sol



Figura 4.3: Tema di “O sole mio”, in La

basso e c', c'', c''' verso l'alto. La stringa "4/4 CCC8 CC8 C8 c8 c'8 c''8 c'''8" rappresenta 8 crome con gli 8 Do del pianoforte.

Chiaramente la rappresentazione simbolica può essere personalizzata in funzione della specifica applicazione.

4.4.1 Rappresentazione NOTE

In questa rappresentazione una melodia è rappresentata da una stringa di metacaratteri. Ogni metacarattere è a sua volta una stringa che specifica una nota o una pausa. La stringa che rappresenta una nota composta da una o più caratteri che ci dicono il nome della nota e eventuali alterazioni (♯ e ♭) e un numero che indica l'ottava in cui si trova. Un carattere specifico, la lettera “p”, rappresenta la pausa.

Come esempio, consideriamo la melodia di *O sole mio* riportata nella Figura 4.2. La sua rappresentazione NOTE è: p D5 C5 B4 A4 G4 G4 A4 B5 G4 F♯4 E4 p F♯4 G4 A4 F♯4 E4 E4 p F♯4 G4 A4 E4 D4 D4 p D5 C5 B4 A4 G4 G4 A4 B5 G4 F♯4 E4 p D5 B4 A4 D5 B4 A4 G4 A4 B4 A4 B4 A4 G4.

La Figura 4.3 mostra la stessa melodia trasposta nella tonalità di La. La sua rappresentazione NOTE è: p E5 D5 C♯5 B4 A4 A4 B4 C♯5 A4 G♯4 F♯4 p G♯4 A4 B4 G♯4 F♯4 F♯4 p G♯4 A4 B4 F♯4 E4 E4 p E5 D5 C♯5 B4 A4 A4 B4 C♯5 A4 G♯4 F♯4 p E5 C♯5 B4 E5 C♯5 B4 A4 B4 C♯5 B4 C♯5 B4 A4.

4.4.2 Rappresentazione INTERVAL

In questa rappresentazione concentriamo l'attenzione sugli intervalli misurati in semitoni. Il metacarattere inizia con il simbolo “+” o con il simbolo “-” per indicare la direzione dell'intervallo. Dopo il segno c'è il numero di semitoni. Poiché in questa rappresentazione si indica un intervallo non possiamo specificare la nota di partenza. Inoltre nel caso di una pausa l'intervallo non ha senso. Quindi la prima nota verrà ignorata, sia all'inizio del brano, sia dopo una pausa. La presenza della pausa però verrà comunque indicata dalla lettera p.

Con la rappresentazione INTERVAL, la melodia di *O sole mio* riportata nella Figura 4.2 corrisponde alla stringa: -2 -1 -2 -2 +0 +2 +2 -4 -1 -2 p +1 +2 -3 -2 +0 p +1 +2 -5 -2 +0 p -2 -1 -2 -2 +0 +2 +2 -4 -1 -2 p -1 -2 +5 -3 -2 -2 +2 +2 -2 -2 -2.

Con questa rappresentazione anche la melodia in La, Figura 4.3, sarà descritta dalla stessa stringa che rappresenta la melodia in Sol.

4.4.3 Rappresentazione DURATION

In questa rappresentazione concentriamo l'attenzione sulla durata delle note. Utilizzeremo la frazione che corrisponde alla durata della nota. Quindi una semibreve sarà rappresentata $1/1$, una minima sarà rappresentata da $1/2$, una semiminima sarà rappresentata da $1/4$, una croma sarà rappresentata da $1/8$, e così via. Si noti come questa rappresentazione permetta di rappresentare anche le durate delle note con un punto. Ad esempio una semimina con un punto sarà rappresentata da $3/8$. Inoltre in questa rappresentazione non c'è differenza fra pause e note. D'altra parte le pause sono parte integrante della melodia.

Con la rappresentazione DURATION, la melodia di *O sole mio* riportata nella Figura 4.2 corrisponde alla stringa: $1/8$ $1/8$ $1/8$ $1/8$ $1/4$ $1/4$ $1/8$ $1/8$ $1/8$ $1/8$ $1/4$ $1/4$ $1/8$ $1/8$ $1/8$ $1/8$ $1/8$ $1/8$ $1/4$ $1/8$ $1/8$ $1/8$ $1/8$ $1/4$ $1/4$ $1/8$ $1/8$ $1/8$ $1/8$ $1/4$ $1/4$ $1/8$ $1/8$ $1/8$ $1/8$ $1/4$ $1/4$ $1/16$ $1/16$ $1/16$ $3/8$.

Anche in questo caso il cambio di tonalità non comporta differenze nella rappresentazione.

Esercizi

1. Che cosa è il formato MusicXML?
2. In un file MusicXML quale è la differenza fra lo schema **partwise** e quello **timewise**?
3. Descrivere i principali tag usati in MusicXML per specificare gli elementi di una partitura.
4. Che cosa è **music21**?
5. Descrivere brevemente la rappresentazione **tinyNotation** di **music21** e fornire un esempio musicale in tale rappresentazione.
6. Che cosa sono i flussi di note in **music21**? In cosa si differenziano dalle liste?
7. Scrivere un programma Python che usa **music21** per generare un flusso di note che rappresenta una scala musicale.

Composizione Musicale Algoritmica
Dipartimento di Informatica
UniSa - A.A. 2024-2025
Prof. De Prisco