Automated Music Transcription based on Formal Language Models
E. Granados, Goyescas

typesetted with Lilypond
Why studying Music Notation Processing?

**Western Music Notation** = graphical format for music practice, in use since ~1000 years (Guido d’Arezzo)

(digital) music scores, a natural language for

- performers
  performance: real-time reading or memoization

- composers
  authoring, exchange

- teachers & students
  transmission

- editors
  access digital score libraries e.g. nkoda.com

- librarians
  cultural heritage preservation: e.g. Gallica

- scholars (historians, musicologists...)
  research, analysis

Philippe Manoury
Tensio for string quartet and electronics
Music Notation Processing

- Structured music score models
  hierarchical representation of music scores
- Music scores languages
  finite representations of \( \infty \) sets of scores (style)
- Search and retrieval
  indexing, exact or approximate search, faceted
- Similarity metrics
  string and tree edit-distances

Applications

- Databases of digital music scores
  Cultural heritage preservation \( \text{H2020 Polifonia} - \text{U. Bologna, Open University, King's College, Vrije U. Amsterdam} \)
- Computational Musicology
  \text{neuma.huma-num.fr} - \text{IReMus (Paris), AlgoMus (Lille)}
- Optical Music Recognition, Crowdsourced correction
  \text{ANR Collabscore} - \text{IRISA (Renne), French National Library, Royaumont}
- Automated Music Transcription
  \text{JSPS 採譜, grant Yamaha Music Foundation} - \text{JAIST, Nagoya U.}
Conversion of a recorded music performance into a music score ~ speech-to-text in NLP
a holy graal in Computer Music since 1970's
Automated Music Transcription today

Conversion of a recorded music performance into a music score

Audio recording

MIDI device (score edition)

Algorithmic composition DAW

source(s)

intermediate representation

- piano roll (MIDI file)
- unquantized onsets, durations
- quantized pitches

audio Music Information Retrieval
- fundamental freq. estimation
- onset detection
- beat tracking …

symbolic Music Information Retrieval
- rhythm quantization
- tempo tracking
- score engraving…

target

music score (XML file)
Grid based Approaches to Rhythm Quantization

Rhythm quantization with grids, e.g. MIDI files import
• in score editors (Finale, Sibelius, Dorico, Musescore…),
• or in DAWs (Ableton Live, Logic…)

Alignment of every input time point (onset) to the closest position in a grid = sequence of equidistant time position.

- Grids based on different rhythms:
  - Grid 16th note
  - Grid 32th note
  - Hierarchical grid

- Alignment examples:
  - Poor fit, good readability
  - Good fit, bad readability
  - Closer to intuition
Regular vs Irregular Grids

regular grids
- search of a best quantization is possible by a brute-force enumeration:
  8th note grid, 16th, 32th, 64th…
- result not always optimal
- problems with tuplets (so called “irrationals” 3, 5, 7…)

hierarchical grids
- more “natural” results
- brute force enumeration impossible
- how to specify the grids to try?
Polonaise in D minor from Notebook for Anna Magdalena Bach  

- **beamed**

- **unbeamed**

- **hierarchical note durations**
metric structure

<table>
<thead>
<tr>
<th>bar</th>
<th>beat</th>
<th>subbeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1</td>
<td>1.1.1</td>
</tr>
<tr>
<td>1.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2.1</td>
<td>2.1.1</td>
</tr>
<tr>
<td>2.2</td>
<td>2.2.1</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>2.3.1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.1</td>
<td>3.1.1</td>
</tr>
<tr>
<td>3.2</td>
<td>3.1.2</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>3.3.1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.1</td>
<td>4.1.1</td>
</tr>
<tr>
<td>4.2</td>
<td>4.1.2</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>4.2.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5.1</td>
<td>5.1.1</td>
</tr>
<tr>
<td></td>
<td>5.2</td>
<td>5.2.1</td>
</tr>
<tr>
<td></td>
<td>5.3</td>
<td>5.2.3</td>
</tr>
</tbody>
</table>

beamed

unbeamed

grouping notes with measure bars and beams
- eases readability (player reads in a real-time context)
- highlight the metric structure
hierarchy of strong / weak beats
Polonaise in D minor from Notebook for Anna Magdalena Bach

<table>
<thead>
<tr>
<th>bar</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>beat</td>
<td>1.1</td>
<td>2.1</td>
<td>3.1</td>
<td>4.1</td>
<td>5.1</td>
</tr>
<tr>
<td>subbeat</td>
<td>1.1.1</td>
<td>2.1.1</td>
<td>3.1.1</td>
<td>4.1.1</td>
<td>5.1.1</td>
</tr>
</tbody>
</table>

Durations:

- bar 1: 1 1 1 1 0 1 1 1 2
- bar 2: 1 1 1 2 1 1 1 1 1
- bar 3: 1 1 1 0 1 1 1 1 1
- bar 4: 1 1 1 1 1 1 1 1 1
- bar 5: 1 1 1 1 1 1 1 1 0

Keys:

- bar 1: C sharp minor
- bar 2: C sharp minor
- bar 3: D minor
- bar 4: D minor
- bar 5: C sharp minor
Tree representation of the proportional rhythmic notation with hierarchical encoding of durations: “the (duration) data is in the structure”

- the tree leaves contain the events
- the branching define durations, by partitioning of time intervals
Tree-structured Representation of Music Notation

J.S. Bach
Polonaise - Notenbüchlein für Anna Magdalena Bach

1.4 bar 1

1.5 bar 2

1.6 bars 1 & 2

measure 1

measure 2

single bar

double bar
(end marker)
defined by a Regular Tree Grammar:
- non-terminal symbols: $q, q_0, q_1, \ldots$
- terminal symbols (constants): • (1 note), •_2 (1 grace-note + 1 note), − (continuation)
- production rules:

\[
q \rightarrow m_2(q_0, q) \mid m_0
\]
\[
q_0 \rightarrow u_3(q_1, q_1, q_1) \mid •
\]
\[
q_1 \rightarrow b_2(q'_2, q_2) \mid • \mid •_2 \mid −
\]
\[
q'_2 \rightarrow b_2(q'_3, q_3) \mid • \mid •_2 \mid − \quad q_2 \rightarrow b_2(q_3, q_3) \mid • \mid −
\]
\[
q'_3 \rightarrow • \mid •_2 \mid − \quad q_3 \rightarrow • \mid −
\]

derivations (leftmost)

\[
q_1 \rightarrow b_2(q'_2, q_2) \rightarrow b_2(b_2(q'_3, q_3), q_2) \rightarrow b_2(b_2(•_2, q_3), q_2) \rightarrow b_2(b_2(•_2, •), q_2) \rightarrow b_2(b_2(•_2, •), •)
\]

\[
q \rightarrow m_2(q_0, q) \rightarrow m_2(u_3(q_1, q_1, q_1), q) \rightarrow m_2(u_3(b_2(q'_2, q_2), q_1, q_1), q) \rightarrow m_2(u_3(b_2(•, q_2), q_1, q_1), q) \rightarrow \ldots
\]
Transcription as Parsing

- **piano roll**
  - sequence of timestamped input events

- **parsing**
  - structuring a linear representation according to a language model

- leaves = output event sequence

- **Parse Tree**

- tree-structured representation of an output **music score**
  - conforming to a prior language (expected notation)

2 nested **extensions** of parsing are needed for the case music transcription:
- weighted extension
- symbolic weighted extension (quantitative parsing)
Conventional Parsing

terminal symbols: $e_0, \ldots$ in a finite alphabet

input sequence

yield (sequence of leaves)

parse-tree = representation of a leftmost derivation of $e_0 e_1 \ldots e_n$ by a prior CF-grammar $G$ with production rules: $q_0 \rightarrow q_1 q_2$ or $q_0 \rightarrow e$
(non-terminal symbols: $q_0, q_1, \ldots$)

Decision problem: (membership)
does there exists a parse tree (leftmost derivation) of $G$ that yields $e_0 e_1 \ldots e_n$?
Returning a parse tree of $G$ that yields $e_0 e_1 \ldots e_n$

With an ambiguous prior CF-grammar $G$ there might exist several parse trees (exponentially many).

In order to choose one (or some) parse trees, rank them according to their weight values, computed by Weighted Tree Grammar.
Weighted Tree Language (tree series)

Weighted Regular Tree Grammar $G$:
- non-terminal symbols: $q, q_0, q_1, \ldots$
- terminal symbols (constants): $\bullet$ (1 note), $\bullet_2$ (1 grace-note + 1 note), $-$ (continuation)
- every production rule is assigned a weight value (e.g. cost to read):

$$
\begin{align*}
q & \rightarrow m_2(q_0, q) \\
q_0 & \rightarrow u_3(q_1, q_1, q_1) \\
q_1 & \rightarrow b_2(q'_2, q_2) \\
q_2 & \rightarrow b_2(q_3, q_3) \\
q_3 & \rightarrow \cdots \\
q_0 & \rightarrow \bullet \\
q_1 & \rightarrow \bullet \\
q_2 & \rightarrow \bullet \\
q_3 & \rightarrow \bullet \\
q_0 & \rightarrow \cdot \\
q_1 & \rightarrow \cdot \\
q_2 & \rightarrow \cdot \\
q_3 & \rightarrow \cdot \\
\end{align*}
$$

measure

beat = $\downarrow$

sub-beat = 8th-note = $\downarrow$

sub-sub-beat = 16th note = $\downarrow$

derivation (leftmost): $d : q_1 \rightarrow b_2(q'_2, q_2) \rightarrow b_2(b_2(q'_3, q_3), q_2) \rightarrow b_2(b_2(\bullet_2, q_3), q_2) \rightarrow b_2(b_2(\bullet_2, \bullet), q_2) \rightarrow b_2(b_2(\bullet_2, \bullet), \bullet)$

cost of derivation: $\text{weight}(d) = 0.1 + 0.1 + 3.25 + 1 + 1$

learning weight values from corpus statistics

Francesco Foscarin
In general, the weight values are taken in a **commutative Semiring** \((\mathbb{S}, \oplus, \otimes, \otimes, \mathbb{I})\):

- \(\oplus\) and \(\otimes\) are associative and commutative, with neutral elements \(\mathbb{O}\) and \(\mathbb{I}\)
- \(\otimes\) distributes over \(\oplus\): \(x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)\)
- \(\mathbb{O}\) is absorbing for \(\otimes\): \(\mathbb{O} \otimes x = \mathbb{O}\)

Moreover, \(\oplus\) is assumed to extend to **infinite sums**: there is an operation \(\bigoplus_{i \in I} x_i\) for all \(I \subseteq \mathbb{N}\) such that:

\[
\text{infinite sums extend finite sums: } \forall j, k \in \mathbb{N}, j \neq k, \bigoplus_{i \in \emptyset} x_i = \mathbb{O}, \bigoplus_{i \in \{j\}} x_i = x_j, \bigoplus_{i \in \{j,k\}} x_i = x_j \oplus x_k
\]

**associativity and commutativity:**

for all partition \((I_j)_{j \in J}\) of \(I\), \(\bigoplus_{j \in J} \bigoplus_{i \in I_j} x_i = \bigoplus_{i \in I} x_i\)

**distributivity** of products over infinite sums: for all \(I \subseteq \mathbb{N}\), \(\forall x, y \in \mathbb{S}\)

\[
\bigoplus_{i \in I} (x \otimes y_i) = x \otimes \bigoplus_{i \in I} y_i \quad \text{and} \quad \bigoplus_{i \in I} (x_i \otimes y) = \bigoplus_{i \in I} x_i \otimes y
\]
Weight of derivations and trees

<table>
<thead>
<tr>
<th>Domain</th>
<th>$\oplus$</th>
<th>$\otimes$</th>
<th>$\ominus$</th>
<th>$\top$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>$\lor$</td>
<td>$\land$</td>
<td>$\bot$</td>
<td>$T$</td>
</tr>
<tr>
<td>Viterbi</td>
<td>$\max$</td>
<td>$\times$</td>
<td>$0$</td>
<td>$1$</td>
</tr>
<tr>
<td>Tropical min-plus</td>
<td>$\min$</td>
<td>$+$</td>
<td>$+\infty$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

$\otimes$ is for composition of rule’s weights in derivations and $\oplus$ is for optimal choice:

For a Weighted Regular Tree Grammar $\mathcal{G}$

$$\text{weight}_{\mathcal{G}}(d : q \rightarrow \ldots \rightarrow t) = \bigotimes_{i=1}^{n} w_i \quad \text{and} \quad \text{weight}_{\mathcal{G}}(q, t) = \bigoplus_{d : q \rightarrow t} \text{weight}_{\mathcal{G}}(d)$$

or recursively:

$$\text{weight}_{\mathcal{G}}(q, a(t_1, \ldots, t_n)) = \bigoplus (w \otimes \bigotimes_{q \rightarrow a(q_1, \ldots, q_n) \in \mathcal{G}} \text{weight}_{\mathcal{G}}(q_i, t_i))$$
$n$-best Parsing

<table>
<thead>
<tr>
<th>domain</th>
<th>$\oplus$</th>
<th>$\otimes$</th>
<th>$\ominus$</th>
<th>$\sqcup$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>$\bot$, $T$</td>
<td>$\lor$</td>
<td>$\land$</td>
<td>$T$</td>
</tr>
<tr>
<td>Viterbi</td>
<td>$[0,1] \subseteq \mathbb{R}$</td>
<td>$\max$</td>
<td>$\times$</td>
<td>$0$, $1$</td>
</tr>
<tr>
<td>Tropical min-plus</td>
<td>$\mathbb{R}_+ \cup {+\infty}$</td>
<td>$\min$</td>
<td>$+$</td>
<td>$+\infty, 0$</td>
</tr>
</tbody>
</table>

$\mathbb{S}$ is assumed:
- **idempotent** $x \oplus x = x$
  that induces a partial ordering: $x \leq_{\oplus} y$ iff $x \oplus y = x$
- **total** : $\forall x, y \in \mathbb{S}$, either $x \oplus y = x$ or $x \oplus y = y$ i.e. $\leq_{\oplus}$ is total
- **bounded** : $\bot \oplus x = \bot$, or equivalently: $\forall x, y \in \mathbb{S}, x \leq_{\oplus} x \otimes y$
  i.e. combining elements with $\otimes$ always increases their weight,
see the **non-negative weights** condition for Dijkstra’s shortest path algorithm

\[ k \]-best parsing : enumeration of the $k$ best weighted trees wrt $\leq_{\oplus}$ for $\mathcal{G}$ and a non-terminal $q$, in PTIME, user the above assumptions.

Similar to best path search in hyper-graphs (Dynamic Programming)
- Viterbi algorithm in acyclic case
- Knuth generalization of Dijkstra’s algorithm in the general case
Quantitative Parsing (extension 2): IO measure

There is no 1-1 correspondence between input sequence and output leaf sequence.

We extend weighted parsing by ranking solutions with:

- A measure of input / output fitness
  - Cost of IO alignment

Measure of cost-to-read

Weight value

Computed by the Weighted Tree Grammar
measure of input/output fitness

\[
\begin{pmatrix}
E_{on} & E_{off} & D_{on} & D_{off} & C_{on} & C_{off} \\
0.11 & 0.19 & 0.22 & 0.48 & 0.53 & 1.08
\end{pmatrix}
\]

input sequence

linearisation of the output tree

Example alignment

\[
\begin{align*}
&b_2 \quad b_2 \quad b_2 \quad \ldots \quad b_2 \\
&0 \quad \frac{1}{4} \quad \frac{1}{2} \quad 1
\end{align*}
\]

cost of IO alignment computed by a
Weighted word-to-word Transducer
(stateful definition of an edit-distance)

\[
q_0 \xrightarrow{\langle E_{on}, b_2 \rangle} q_1 \xrightarrow{\langle E_{off}, \varepsilon \rangle} q_2 \xrightarrow{\langle D_{on}, \varepsilon \rangle} q_3 \\
\xrightarrow{\langle D_{off}, \cdot \rangle} q_4 \xrightarrow{\langle C_{on}, \varepsilon \rangle} q_5 \xrightarrow{\langle \varepsilon, - \rangle} q_5
\]
measure of input/output fitness

\[ \frac{2}{\text{uni2219}} - \frac{2}{\text{uni2219}} \]

linearisation of the output tree

input sequence

cost of IO alignment
computed by a Weighted word-to-word Transducer (stateful definition of an edit-distance)

Example alignment

\[ q_0 \xrightarrow{\langle E_{on}, \ast \rangle} q_1 \xrightarrow{\langle E_{off}, \varepsilon \rangle} q_2 \xrightarrow{\langle D_{on}, \varepsilon \rangle} q_3 \]
\[ q_4 \xrightarrow{\langle D_{off}, \ast \rangle} q_5 \xrightarrow{\langle C_{on}, \varepsilon \rangle} q_5 \]
Quantitative Parsing (extension 2'): infinite alphabet

in the context of music transcription, the symbols are timestamped → infinite alphabet $\Sigma_{\text{inf}}$
the weighted formalisms below must be able to read such symbols → symbolic extension

input sequence

yield = sequence of leaves decorated with dates $\theta$
(computed with the durations encoded in the tree structure)

measure of input / output fitness
= cost of IO alignment computed by a
Weighted word-to-word Transducer

measure of cost-to-read
computed by the Weighted Tree Grammar
Symbolic Weighted Language Models

Sw-A: $\sum_{\text{inf}}^* \rightarrow S$

$\phi: \sum_{\text{inf}} \rightarrow S$

$q \rightarrow q'$

Weighted-A: $\sum_{\text{fin}}^* \rightarrow S$

$q^a, w \rightarrow q'$

$a \in \sum_{\text{fin}}, w \in S$

Symbolic-A: $\sum_{\text{inf}}^* \rightarrow \text{Bool}$

$q^\phi \rightarrow q'$

NFA: $\sum_{\text{fin}}^* \rightarrow \text{Bool}$

$q^a \rightarrow q'$

$a \in \sum_{\text{fin}}$

Droste, M., Kuich, W., Vogler
Handbook of WA, 2009

Veanes et al.
CAV 2017, CACM 2021
Automated Music Transcription: qparse framework

- **Symbolic-Weighted word-to-word Transducer**
  - Cost of IO alignment

- **Weighted Regular Tree Grammar**
  - Cost of readability
  - Converted into SW Visibly Pushdown Automata (intermediate model)

- **Input MIDI file (unstructured)**

- **SW word-to-tree Transducer**

- **SW Regular Tree Grammar**

- **Parse Tree**
  - 
  -  $k$-best parsing

- **Intermediate Score Representation (structured)**

- **Term rewriting**

- **New $k$-best parsing procedure for SW-VPA (hence SW-RTG)**

- **XML/MEI score file**

- **Music 21 structure**

- **Ontologies...**
for the transformation of the intermediate score representation

questions: rewrite strategies (e.g. IO or OI), conflicts...
Related work

- Piano transcription system (Kyoto U.)
  Non-local musical statistics as guides for audio-to-score piano transcription
  Kentaro Shibata, Eita Nakamura, Kazuyoshi Yoshii

- deep-neural-network-based multipitch detection
  audio to unquantized MIDI

- statistical-model-based (HMM) rhythm quantization
  unquantized MIDI to quantized MIDI

- delegate to Muse Score + Voice separation algorithm for
  quantized MIDI to score

- study of use of non-local statistics (pitch and rhythm)
  for the inference of global characteristics (metre, bar line positions…)

- Score Transformer (Yamaha) - piano transcription
  Score Transformer: Generating Musical Score from Note-level Representation
  Masahiro Suzuki

Transformer model trained with popular songs (piano arrangements), KernScores (piano Sonata)
MIDI to score (tokenization)
Implementation (FJ) of

- the above transcription by parsing framework
- the intermediate score model (w. Philippe Rigaux)
- other subtasks: pitch-spelling, key estimation, beat tracking…

**qparse**: 75 Kloc C++

- command lines tools:
  - monoparse, drumparse, grammar-learning, engraving (from quantified input)

- Python binding - Lydia Rodriguez-de la Nava
  - scripts for automatic evaluation

- online port, real-time - Leyla Villaroel

[https://gitlab.inria.fr/qparse/qparselib](https://gitlab.inria.fr/qparse/qparselib)
[https://qparse.gitlabpages.inria.fr](https://qparse.gitlabpages.inria.fr)
CommonLisp/CLOS, 350 functions, 4900 lines of code

UI: Open Music object, input from chord-seq (notes, onset, dur) + segmentation marks, output to voice (OM rhythm trees)
Score diff

by Francesco Foscarin

- identify the diff. between 2 XML music scores
- string/tree edit distance applied to a intermediate score representation
Lamarque-Goudard dataset (w. Francesco Foscarin, Teysir Baoueb)
- 283 monophonic extracts of classical repertoire inspired by a rhythm learning method
- ~ 20 measures per extract
- progressive difficulty cover a very large spectrum of rhythmic features
- score files (XML) and MIDI performances for evaluation and calibration of transcription tools

Generation of artificial performances
Madoka Goto, Masahiko Sakai (Nagoya U.), Satoshi Tojo (JAIST)
- construction of a GTTM tree
- segmentation accordingly
- performance generation by Director Musices (Anders Friberg)
monophonic: one note at a time
Good results for complex cases (ornaments, mixed tuplets, mixed note durations, silences...)
~ 100ms for the transcription of 1 score
Polonaise in D minor from Notebook for Anna Magdalena Bach   BWV Anh II 128

original score

transcription of MIDI recording by Finale
Beethoven, Trio for violin, cello and piano, op.70 n.2 (2d mov)

Monophonic transcription

original score

transcription of MIDI recording by qparse
Beethoven, Trio for violin, cello and piano, op.70 n.2 (2d mov)

Monophonic transcription

original score

---

transcription of MIDI recording by Finale

options:
- mixed rhythms,
- tuplets
- smallest note = 32nd
The time signature and the tempo are given.
**FiloBass** by John-Xavier Riley (QMUL, C4DM)
project “*Dig That Lick*”
- jazz bass lines, acc. of saxophone
- 48 tracks,
  24 recorded hours of melodies and improvisations
- qparse as backend of an audio-to-MIDI transcription procedure
- prior beat (measure) tracking
**Groove MIDI Dataset**
- by Google Magenta
- 13.6 hours, 1150 MIDI files, ~ 22000 measures recorded by professional drummers on an electronic drum kit
- audio (wav) files synthesized from (and aligned to) MIDI files for evaluation of audio-to-MIDI drum transcription
- no score files!

**Scoring the GMD with qparse**

*Martin Digard (INALCO)*
- all score files (XML) produced from the MIDI files with the same generic tree grammar (4/4 measure)
- polyphonic case-study, simpler than piano
- specific drumming constraints (hands ≤ 2, feet ≤ 2)
- processing errors from MIDI sensors
• Dataset **ASAP** - Francesco Foscarin, Andrew Mc Leod
  MIDI and audio recording from Yamaha piano competition
  + XML scores
  + alignments
  + beat tracking annotations

• **Voice separation** - Lydia Rodrigez-de la Nava, evaluation Augustin Bouquillard
  and for piano guitar transcription.
  integration in transcription:
  • before parsing, or
  • after parsing (on intermediate model), or
  • joint with parsing.
MIDI-to-Score Automated Music Transcription approach

- quantitative parsing technique
  based on Symbolic Weighted formal language formalisms
  Tree Automata and word-to-word Transducers
- with prior quantitative language of notation style
  and prior IO measure
- (abstract) hierarchical score model
  as intermediate representation for score generation
- can handle complex notation cases:
  ornaments, mixed tuplets, mixed note durations, silences…
- efficient

- case studies: Monophonic, Drums
- ongoing work on Polyphonic case studies: guitar, piano

MERCI!
THANK YOU!