



HAL
open science

Automated Transcription of Electronic Drumkits

Martin Digard, Florent Jacquemard, Lydia Rodriguez-de la Nava

► **To cite this version:**

Martin Digard, Florent Jacquemard, Lydia Rodriguez-de la Nava. Automated Transcription of Electronic Drumkits. 4th International Workshop on Reading Music Systems (WoRMS), Nov 2022, online, Spain. hal-03815760v3

HAL Id: hal-03815760

<https://hal.science/hal-03815760v3>

Submitted on 25 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Automated Transcription of Electronic Drumkits

Martin Digard
Inria & Inalco, Univ. Sorbonne Paris-Cité
Paris, France
martindigard@proton.me

Florent Jacquemard
Inria & CNAM/Cedric
Paris, France
florent.jacquemard@inria.fr

Lydia Rodriguez-de la Nava
Inria & CNAM/Cedric
Paris, France
lydia.rodriguez-de-la-nava@inria.fr

Abstract—We present a new approach for the transcription of drum performances captured on a MIDI drum kit into scores in conventional Western notation. It works by parsing an input MIDI sequence into a tree-structured intermediate representation, using formal language techniques, post-processing using term rewriting, and finally exporting into an XML score file in the MEI encoding. An experimentation was conducted on the Groove MIDI Dataset.

Index Terms—Drum notation, Automated Music Transcription.

I. INTRODUCTION

Born in the twentieth century, drums have long gone without music scores; Drummers were initially expected to improvise rhythmic accompaniments based from their fellow musician’s scores (e.g. from music style, chord progressions and melodic themes). Later, with the emergence of drum schools such as Dante Agostini Drum School in Europe¹, a drum notation was settled as a vector for the preservation of performances for future references, and the transmission of different styles to the apprentice or professional drummers.

Automatic Music Transcription is the problem of converting a musical performance into a music score. The particular case of drums has given rise to many studies recently, with a focus mostly on transcription of audio signals to unquantized MIDI files [14]. However, to our knowledge, fewer works, if any, consider the problem of drum score production.

In this work, we study the problem of parsing drum performances captured in MIDI into music scores conforming to notation standards for drums, that are easy to read. Our transcription procedure ought to *align* the input MIDI events to discrete time values expressible with musical notation. Moreover, simultaneously, the musical events obtained are grouped into hierarchies of *rhythmic structures*. These two tasks are performed jointly thanks to the use of a prior formal language model, Section III-A. Since several input events might be aligned to the same time position, an additional difficulty is to determine whether such *vertical grouping* is possible or not, Section III-B. Then (Section IV), we perform *voice separation*, and some post-processing by *term rewriting* before score typesetting. We experimented this approach on MIDI recordings of the GMD, Section V.

This work has been partly supported by Inria Exploratory Project Codex and JSPS KAKENHI Grant Number JP20H04302. The 1st author was a student of Inalco and Inria at the first time of preparation of this work [4].

¹<https://www.danteagostini.com>

II. PRINCIPLES OF DRUM NOTATION

We present in this section a notation for drum sheet music in Conventional Western Notation (CWN). There is actually no official standard, and we follow here the notation in the drum lessons of the the Agostini Drum School (the first European drum school) [1] and the collection of drum pieces by Juskowiak [10]. Other variants, like the one used in the US, called universal, only differ by minor details described below (see Figure 2). Other notation systems not based on CWN are left out of the scope of this work.

A. Elements of Drum Kit and Modes

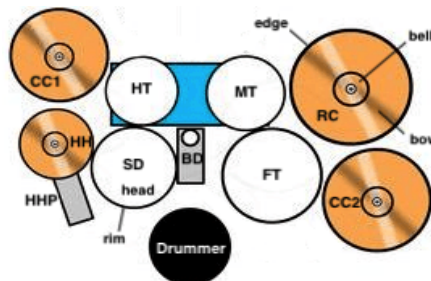


Fig. 1. The elements of a drum kits

In a typical drum kit (Figure 1), the central part is the Bass Drum (BD, also called *kick*), that produces the lowest pitch of the kit, and is struck when stepping on a pedal with the right foot. On its left (from a right-handed drummer’s point of view) is the Snare Drum (SD), and 3 toms: from left to right, High (HT), Medium (MT) and Floor (FT). The BD, HT and FT might be doubled (it is not the case in Figure 1). The SD and toms are played with drum sticks, in different modes: the stick can either hit the *head* (i.e. the skin, the most common case) or the *rim* of the SD or tom. These two modes can be combined in particular SD techniques: in a *rimshot* the rim and the head of the SD are hit simultaneously in order to obtain a brighter and sharper sound; in a *cross-stick* (also denoted *X-stick*), the tip of one stick is maintained against the head, in order to attenuate resonance, while the other end of the stick hits the rim. The detection of rimshots and cross-sticks by a MIDI drum kit can be source of difficulties in the context of transcription, see Section III-B.

The Hi-Hat (HH), on the left of the SD, is made of two cymbals that can be joined (*closed*) or disjoined (*open*) using a

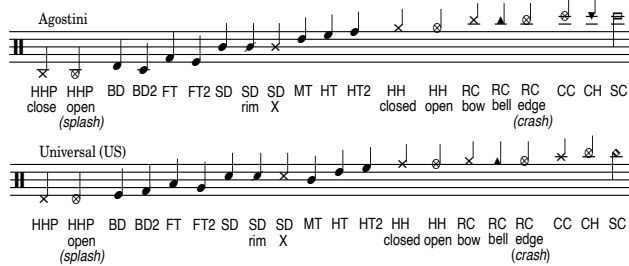


Fig. 2. Pitches and note heads denoting the drum kit elements and modes.

pedal activated with the left foot. Each configuration produces a different sound when the HH is hit with drum sticks. Closing the HH with the pedal also produces a distinguished sound by itself, with a specific notation (see below). There are different kinds of cymbals: *ride* (RC), for steady beats and patterns played either on the *bow* (the body) or the *bell* (the top part) of the cymbal, and *crash* (CC) or *splash* (SC), generally struck along the *edge* to produce an explosive sound (the drum kit in Figure 1 has two CC). The RC can also be hit on the edge to produce a crash sound. The SC are generally smaller than the CC, and produce a higher tone.

In drum notation, every part of a drum kit is associated with a specific pitch (Figure 2). The height of the pitch corresponds roughly to the position in space of the instrument: the pedals (BD and HH) have the lowest pitches, and cymbals the highest, and the SD and toms are in the middle. Moreover, the shape of the note head indicates the mode (rimshot, X-stick, bow, edge or bell for cymbals...).

B. Dynamics

Notes with louder dynamics are marked with standard *accent* symbols (see Figure 3). According to the related pitch, it may indicate that a particular technique ought to be used. On the opposite, the so called *ghost notes*, denoted within parentheses, are played with low dynamics, although firmly.

C. Ornaments

A *flam* is a figure made of one grace note, played with a significantly lower intensity and slightly ahead of one normal or accented note (main note). It is denoted like an *acciaccatura* (Figure 3). It is executed with two hands, using a particular drum technique producing a sound described by the onomatopoeic term of *flam*. A flam can either be played on one single element of the drum kit, or two distinct elements, e.g. SD and a tom.

Most of the time, in drum notation, the *rolls* are quantified. More common for the orchestral snare drum, the tremolo notation should be denoted literally for drums.

D. Timings

Drum events are *transients*: they have no duration. Therefore, the usual symbols of CWN (flags, beams, ties, dots and rest symbols) represent, in drum notation, the temporal distance between one event's start date (onset) and next event's

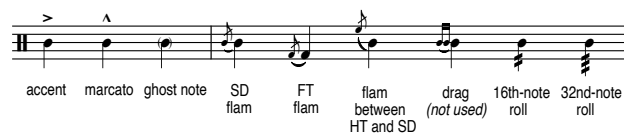


Fig. 3. Accents, ghost notes, and flams.

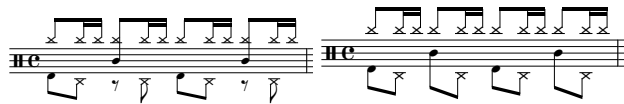


Fig. 4. Examples of voicing in drum scores

start date. Also, notehead cannot be used for scoring these distances because of their variety of use in drum notation. For examples, ♪ ♪ is preferred over ♪.

To improve the readability of the score, as few symbols as possible are used. For instance, the simple ♪ will be preferred over ♪. Moreover, in order for the score to reflect the drummer's way of thinking, the notation is segmented by pulse, for example, ♪ ♪ is preferred over ♪.

Although hi-hat openings are not considered as notes with duration, the fact that they sound until they are closed is taken into account in the scoring. Consequently, ties will sometimes be used for HH notation (but not for the other elements).

E. Voices

As for most polyphonic instruments [8], the notes in a drum score can be grouped into voices, denoted by the stem directions. In drum notation, voices are useful either to separate note played with hands, on the staff top, from those played with feet on staff bottom (Figure 4, left), or to distinguish between repeated rhythmic patterns (played e.g. on HH or RC), and other elements played independently, in a more sporadic way (Figure 4, right).

The voicing is generally defined a priori: every element is assigned a fixed voice number and does not change voice throughout the score.

III. PARSING MIDI DRUM PERFORMANCES

Our transcription process takes place in several steps. In the first, and most important step, an unstructured MIDI input is structured into a tree using parsing techniques. The input is a sequence of timestamped note-on events in a MIDI file. Note-off MIDI events are ignored in the case of drums.

A. Prior Weighted Rhythm Tree Language

The parsing is based on a prior language model whose aim is to define the time positions in a score where the input MIDI events can be aligned. In many quantization algorithms in commercial software, such as Digital Audio Workstations (DAW) or score editors, these positions are equidistant, defined by a regular *grid*. Here, we define those time positions with a generative Regular Tree Grammar (RTG) that reflects a metrical hierarchy [15]. Intuitively, in such a model, the higher

the metrical weight of a time position, the more likely it is going to *attract* an input event.

The RTG generates labeled trees by mean of non-terminal (NT) replacement, following *production rules* of one of the forms: $q_0 \rightarrow a$, where q_0 is a NT and a is a symbol representing one or several output symbols in score (e.g. one note or one flam), or $q_0 \rightarrow b(q_1, \dots, q_k)$, where q_0, \dots, q_k are NTs and b is a symbol representing an operation on time intervals. We consider in particular two such operations:

- *time division*, partitioning a given closed time interval $I = [\tau, \tau']$, into $k \geq 2$ sub-intervals I_1, \dots, I_k of same duration $\frac{\tau' - \tau}{k}$ (symbol b_k - *beamed* and u_k , *unbeamed*),
- *new bar*, partitioning a given open time interval $I = [\tau, +\infty[$ into two sub-intervals $I_1 = [\tau, \tau + 1[$, of duration 1 bar and $I_2 = [\tau + 1, +\infty[$ (symbol m_2).

Every tree t generated by the RTG defines nested time intervals and the bounds of these intervals are the time positions where input events can be aligned. Moreover, t defines hierarchical event grouping, represented by beams in the output score.

Some weight values, in a specific cost domain (*min-plus algebra* \mathbb{S} [12]), are associated to the RTG's production rules, in order to evaluate, for every generated tree t , a *cost* (in \mathbb{S}) of *readability* for the corresponding notation. The RTGs used for our experimental can be found in a repository², and Figure 6.

B. Vertical Alignments

Several input events might be aligned to the same time position defined by a tree. However, not every combination of notes and flams can be played simultaneously by a drummer with only 2 hands and 2 feet. Therefore, we define a language of appropriate combinations of input events, using a Finite State Machine (FSM) that cannot be described here because of space constraints. Moreover, we have noticed some errors in the captation of MIDI events in some modes. For instance, it happens sometimes that a *SD rimshot* is captured, by the MIDI sensors, as a *X-stick* closely followed by a standard (head) *SD*. Such an unlikely combo is detected by our FSM and corrected as a rimshot.

Like the RTG of Section III-A, the above FSM is weighted, and its computation returns a value, in \mathbb{S} , representing the *cost of alignment*, of input events to a unique time point.

Finally, we select a tree t that minimizes, in \mathbb{S} , the combined costs of readability, as defined by the RTG of Section III-A and of cost of alignment, defined by the FSM of this section. The selection is done by k -best parsing [9], in polynomial time, using Dynamic Programming techniques.

IV. SCORE ENGRAVING

The tree t obtained from a MIDI input in Section III is converted straightforwardly into a tree-structured abstract Intermediate Score Representation called *Score Model* (SM), used for further post-processing, before exporting to XML.

A score model for drums is a sequence of *measures*, where every measure is a set of *voices*, and every voice is a sequence

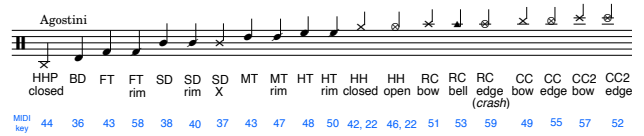


Fig. 5. Mapping of MIDI key numbers for the Roland TD-11 electronic drum kit (source: Groove MIDI Dataset).

of events, described by a *rhythm tree* (RT). The internal nodes of a RT are labeled by a time division symbols as above, and each of its leaves is labeled either by \emptyset , representing a tie or a dot in the score, or by one note, one flam, or by a set of notes, representing a chord in the score.

A. Score model construction and Voice separation

The estimation of note names and note heads is easy, since there is a 1-1 mapping (Figure 5) between MIDI key values, in 0..127, and pairs of pitches and note-heads of Figure 2.

The extraction of dynamics and accents, is based on the velocity value of MIDI events, in 0..127.

The voice of every note is estimated, following the previous principle, according to a prior partition of the elements of the drum kit into voices. Hence, the subtask of voice separation, is straightforward for drums, as opposed to the case of other polyphonic instruments like piano [13]. From the tree t , we extract one RT per voice, by projection, see Figure 7.

B. Term Rewriting

The last step consists in post processing the resulted rhythm trees. Each tree goes through transformations which are defined by term rewriting rules [3], which do not change pitches and durations. A term rewriting rule is an oriented equation that defines how one pattern will be rewritten to another. For example, with the rule: $b_2(x, \emptyset) \rightarrow x$, the last tree $b_2(BD, b_2(HHP, \emptyset))$ in Figure 7 (second voice) is rewritten into $b_2(BD, HHP)$, and $\text{♩} \text{♩} \text{♩}$ is rewritten, in two steps, into $\text{♩} \text{♩} \text{♩}$. Note that we are using term rewriting and not string rewriting: rules are applied to tree structures. For instance, $\text{♩} \text{♩} \text{♩}$ will not rewrite into $\text{♩} \text{♩} \text{♩}$ because the second 8-th note and the rest do not belong to the same subtree.

The purpose of this post processing step is to fix details of rhythm notation [11]. Drum notation particularly needs attention since although notes do not really have a duration, it is easier to read a score without too many rests. Fixing such details during the parsing would significantly increase the execution time, since rewriting is only applied on one score model, instead of on each candidate parse tree.

V. EVALUATION

A. Implementation

Our parsing approach to drum transcription has been implemented in a C++ library³. We have developed a k -best parsing algorithm with dynamic programming and tabulation techniques, for finding efficiency the parse tree that best corresponds to the input performance.

²<https://gitlab.inria.fr/transcription/gmdscores>

³<https://gitlab.inria.fr/qparse/qparselib/>

$q_0 \xrightarrow{0} m_2(q_1, q_0)$	$q_0 \xrightarrow{0} m_0$	$q_4 \xrightarrow{1} c$	$q_8 \xrightarrow{1} c$	$q_6 \xrightarrow{1} c$
$q_1 \xrightarrow{1} c$	$q_2 \xrightarrow{1} c$	$q_4 \xrightarrow{1} a$	$q_8 \xrightarrow{1} a$	$q_6 \xrightarrow{1} a$
$q_1 \xrightarrow{1} a$	$q_2 \xrightarrow{1} a$	$q_4 \xrightarrow{2} f$	$q_8 \xrightarrow{2} f$	$q_6 \xrightarrow{3.5} f$
$q_1 \xrightarrow{2} f$	$q_2 \xrightarrow{2} f$	$q_4 \xrightarrow{0.1} b_2(q_2, q_8)$	$q_8 \xrightarrow{0.1} b_2(q_6, q_6)$	$q_6 \xrightarrow{0.1} b_2(q_7, q_7)$
$q_1 \xrightarrow{0.1} u_2(q_2, q_2)$	$q_2 \xrightarrow{0.1} u_2(q_4, q_4)$	$q_4 \xrightarrow{3.0} b_3(q_8, q_8, q_8)$		$q_7 \xrightarrow{1} c$
$q_1 \xrightarrow{3.0} u_2(q_2, q_2, q_2)$	$q_2 \xrightarrow{3.0} u_3(q_4, q_4, q_4)$	$q_4 \xrightarrow{0.15} b_4(q_6, q_6, q_6, q_6)$		$q_7 \xrightarrow{1} a$

Fig. 6. Prior language model as a wRTG. q_1 represents the level of a 4/4 measure, that can contain either one whole rest (c), one whole note (a), or one flam (f), or can be divided into 2 halves or 3 thirds (triplet of half-notes); q_2 represents the level of a half measure, q_4 of a quarter note (beat level), and so on. Finally, q_0 represents the level over the bar, with a first rule to create one measure, in NT q_1 , and a second one to finish the score (m_0 is a double bar).

B. Dataset for Experiments

For our experiment, we used the Groove MIDI Dataset [7] (GMD). Originally created with machine learning tasks in mind, the dataset proposes 1,150 MIDI files, captured by professional or semi-professional drummers, on an electronic drum kit ROLAND T-11, performing with a metronome. The performances are either short rhythmic fills, or full-length rhythm sequences, according to a specific style (rock, funk, jazz...). Overall, the GMD gathers over 22,000 measures of drumming. A unique mapping assigns a MIDI pitch to each of the parts of the drum kit, for all the MIDI file.

We used our code to transcribe a selection of around 30 files from the GMD, ranging through as many styles as possible, and the shortest being only 4 bars, and the longest 261 bars.

The tempo, the style, and the time signature are given in the file names of the GMD, and used for parsing. The audio files of the dataset, synthesized from the MIDI files are not used in our experiments.

C. Prior languages for evaluation

We have used essentially the same RTG (and one variant) for all the transcriptions performed during our experiments. These RTGs specify possible drum notations, by restricting metrical divisions, for measures in 4/4, as drum score are often written in this time signature. Figure 6 presents a simplified version of these RTGs. Each NT replacement rule of this RTG defines either a time division, or the creation of a terminal symbol, representing output events.

The RTGs used for conducting our experiments were not trained specifically for the GMD. In particular, the weight values are chosen arbitrarily, following roughly the principles that: every symbol like c or a induces a weight value of 1 (see the 2 first rules of every column of Figure 6), and that triplets are penalized compared to binary divisions (see rules 5 and 6 in the first and second column of Figure 6). Moreover, flams are more penalized for short notes than for longer ones (rule 4 in the first and second column of Figure 6). By lack of the digital drum scores corresponding to the MIDI performances in the GMD, we were not able to train a RTG model with grammatical inference techniques similar to, e.g. [2], [6].

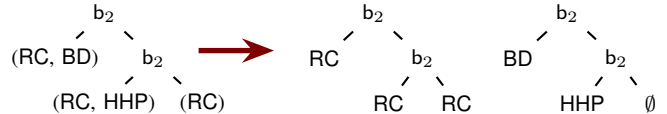


Fig. 7. On the left, the tree corresponding to the first beat in Figure 4. On the right, the trees for each voice (\emptyset represents a rest).

D. Evaluation

A quantitative evaluation of our transcription results would require, for each MIDI file, one corresponding reference score. Such ground truth does not exist for the GMD. Ideally, it would be prepared by professional drummers, but transcribing by hand each of the 1,150 MIDI files of the GMD seems unrealistic, considering in particular the size of files. Moreover, in our case, evaluation cannot be conducted by comparing MIDI files, like e.g. in [14], but would require a tool to diff XML score files, like [5].

To propose a qualitative evaluation, we used other common softwares for music score edition, with the same MIDI files, and visually compared the results. Overall, we found that our algorithm produces scores that are easier to read and closer to expectation. We have gathered all these results on GitLab repository², with all the MIDI files and the corresponding transcription in MEI, and a table of all the parsing times.

VI. CONCLUSION

We presented our first experiments for the automatic transcription of drum performances, by parsing a MIDI input into a structured score model, which can be exported to XML/MEI. Our first results are already promising, although the output scores have some errors, sometimes because of sensor bugs from the drum kit, and sometimes because of the parsing.

Our future objective is to create a complete dataset of drum scores from the GMD, initially transcribed by our parsing technique, then manually corrected by a professional drummer. This dataset could be a companion for the Groove MIDI dataset to use as a base to evaluate drum transcriptions, as well as for OMR for written drum contents.

REFERENCES

- [1] D. Agostini. *Studies for the Drums*, volume 1-4. Editions Dante Agostini, 7 bis rue Thénard, F-89100 Sens, 1977.
- [2] J. F. Bernabeu, J. Calera-Rubio, J. M. Iñesta, and D. Rizo. Melodic identification using probabilistic tree automata. *Journal of New Music Research*, 40(2):93–103, june 2011.
- [3] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume volume B: Formal Models and Semantics, chapter 6, pages 243–320. North-Holland, Amsterdam, 1990.
- [4] M. Digard. Modélisation d’un système de formes rythmiques pour la transcription automatique de la batterie. Master’s thesis, Institut National des Langues et Civilisations Orientales, 2021.
- [5] F. Foscarin, R. Fournier-S’Niehotta, and F. Jacquemard. A diff procedure for music score files. In *6th International Conference on Digital Libraries for Musicology (DLfM)*, page 7, The Hague, Netherlands, Nov. 2019. ACM.
- [6] F. Foscarin, F. Jacquemard, and P. Rigaux. Modeling and learning rhythm structure. In *Sound and Music Computing Conference (SMC)*, 2019.
- [7] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman. Learning to groove with inverse sequence transformations. In *International Conference on Machine Learning (ICML)*, 2019.
- [8] E. Gould. *Behind bars: the definitive guide to music notation*. Faber Music Ltd, 2016.
- [9] L. Huang and D. Chiang. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing ’05, pages 53–64, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.
- [10] O. L. J.-F. Juskowiak. *Agostini Systèmes Drums*, volume 2. MusicCom publications, Editions Joseph BÉHAR, 61, rue du Bois des Joncs Marins - 94120 Fontenay-sous-Bois, 2000.
- [11] F. Jacquemard, P. Donat-Bouillud, and J. Bresson. A Structural Theory of Rhythm Notation based on Tree Representations and Term Rewriting. In *5th International Conference on Mathematics and Computation in Music (MCM)*, volume 9110 of *LNAI*. Springer, 2015.
- [12] J.-E. Pin. Tropical Semirings. In J. Gunawardena, editor, *Idempotency (Bristol, 1994)*, Publ. Newton Inst. 11, pages 50–69. Cambridge Univ. Press, Cambridge, 1998.
- [13] K. Shibata, E. Nakamura, and K. Yoshii. Non-local musical statistics as guides for audio-to-score piano transcription. *arXiv preprint arXiv:2008.12710*, 2020.
- [14] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Muller, and A. Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 26(9):1457–1483, 2018.
- [15] J. Yust. *Organized Time*. Oxford University Press, 2018.