A Formal Approach for High-Level Automatic Rhythm Generation

George Sioros
University of Porto (Faculty of Engineering)
and INESC - Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto
Portugal
E-mail: gsioros@gmail.com

Carlos Guedes
University of Porto (Faculty of Engineering)
and INESC - Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto
Portugal
cguedes@fe.up.pt

Abstract

In this paper, we present a novel algorithm for automatically generating rhythms in real time in a given meter. The generated rhythms are "generic" in the sense that they are characteristic of each time signature without belonging to a specific musical style. A stochastic model in which various aspects and qualities of the generated rhythm can be controlled intuitively and in real time was developed. Such qualities are the density of the generated events per bar, the amount of variation in generation, the amount of syncopation, the metrical strength, and, of course, the meter. The kin.rhythmicator software application was developed to implement the algorithm.

Introduction

In this paper, we propose an approach for real-time rhythm generation based on a stochastic model. This approach contrasts with recent ones involving evolutionary methods such as genetic algorithms [1][2], cultural algorithms [3] or connectionist approaches [4], as well as other non-stochastic methods like in [5][6]. In our approach, the algorithm produces a rather static output with slight variations due to the stochastic nature of the algorithm. It is up to the user to modify the output of the algorithm by altering descriptive musical parameters that produce perceivable changes in the output, such as the amount of syncopation, the degree of metrical strength, and the density of events. In this sense, the algorithm behaves like a musical companion that responds musically to requests made by the user in musical terms.

The algorithm is based on a stochastic model that automatically generates rhythms in a certain meter and metrical subdivision level defined by the user. The stochastic model comes from the output of the metrical indispensability algorithm by Clarence Barlow [7]. The generated rhythms are "generic" in the sense that they are characteristic of the specified meter but do not belong to a specific musical style. Several musical parameters of the performance can be intuitively controlled in real time. Such parameters are the density of events per bar, the amount of syncopation, the amount of variation in generation, the metrical strength of the rhythm being generated and, of course, the meter itself. The kin.rhythmicator application, that implements the algorithm, is developed as a Max/MSP [8] abstraction and a Max4Live [9] device.

The Algorithm

The algorithm has two distinct phases. First, the musical meter entered by the user is subdivided into the number of pulses of a specified metrical subdivision level. Each pulse is assigned a weight value according to its importance in the meter so that a pattern characteristic of the meter emerges. These weights can be thought of as a measure of how much each pulse contributes to the character of the meter

as prescribed by Barlow's indispensability. In the second phase, the weight values are used to generate a stochastic performance. A direct mapping of the weights to probabilities of triggering events gives rise to simple rhythmic patterns expected for a given meter. The weights are processed in order to enforce or weaken the metrical feel, syncopate according to the specified meter and control the variations in the generated rhythm. During the performance the user controls these values, as well as the events' articulation (staccato or legato), indirectly through graphic controls. This gives a very intuitive control over these parameters and over the real-time rhythm generation. In the upcoming sections we describe in detail the steps taken to achieve these results.

Calculating the weights. The first phase in the algorithm is articulated in two steps: sorting the pulses by metric indispensability and then calculating the weights based on the stratification levels.

The pulses are assigned weights based on the order of their importance in the meter according to Clarence Barlow's metric indispensability formula and the stratification of the meter [7]. The user inputs meter information in the form of a time signature and a metrical subdivision level which defines the number of pulses the measure is divided into – e.g. a 3/4 meter at the 16th note metrical subdivision level has 12 pulses. Based on this information the meter is stratified by decomposing the number of pulses into prime factors (see Figure 1). Each prime factor describes how each stratification level is subdivided. The stratification level at index 0 is always a whole bar (prime factor 1). In the above example, a 3/4 meter at the 16th note metrical level is factorized as 1x3x2x2, which means that the first stratification level divides the bar into 3 quarter notes, each quarter note is divided in 2 eight notes and each eight note is divided in 2 sixteen notes, yielding the product of 12 pulses per bar.

The stratification algorithm that is used distinguishes between simple and compound meters which contain the same number of pulses. These meters are decomposed in to the same prime factors but these factors should be in a different order. In general, different permutations of the prime factors describe different metrical hierarchies. For example, the meters 3/4 and 6/8, although they contain the same number of subdivisions at the sixteenth-note level (12), the first is decomposed as 1x3x2x2, while the second as 1x2x3x2. The stratification algorithm takes care of the order of the prime factors by first subdividing the bar according to the time units specified by the time signature's denominator, e.g. 3 x (quarter notes) in the 3/4 meter and 6 x (eighth notes) in the 6/8 meter. If the nominator can be divided by both 2 and 3, it is first divided as many times as possible by 2, putting the corresponding number of "2s" first in the list of prime factors followed by the "3s" and the rest of the "2s". In any other case, i.e. in simple meters, the prime factors are always ordered in descending order, from the greatest to the smallest.

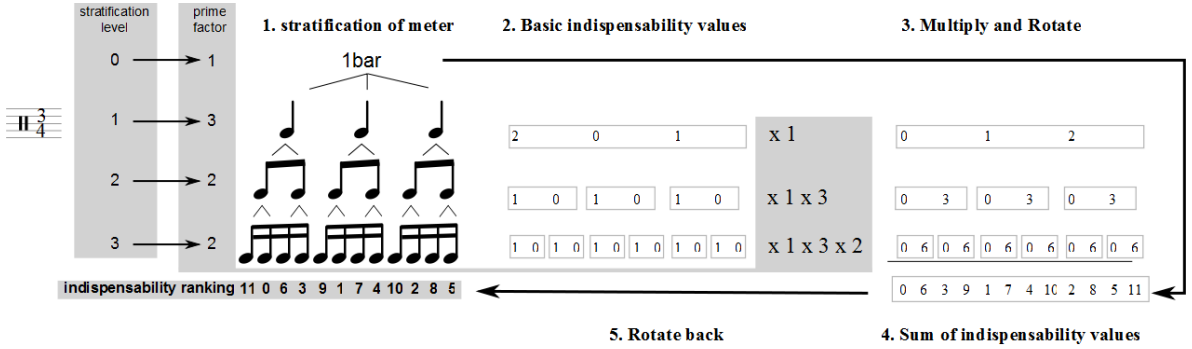

Figure 1: *Calculation of the indispensability values of a 3/4 meter*

Barlow´s indispensability algorithm [7] takes the prime factors of each stratification level and sorts the pulses in the meter according to how much each pulse contributes to the character of the meter, from the most indispensable to the least important (see Figure 1). The algorithm is based on what Barlow terms as the basic indispensability values that represent the order of importance of the subdivisions in each prime number stratification level. Each pulse is assigned with a separate indispensability value for each stratification level. These values are then multiplied by the prime factor of the corresponding level and all higher ones. The resulted arrays are rotated by one step to the left. The sum of all levels is calculated for each pulse. The resulting array is rotated back, i.e. one step to the right, yielding the indispensability values for the pulses.

Barlow's metrical indispensability algorithm [5] is summarized in the following equations.

$$\Psi(n)=\sum_{r=0}^{z-1}\left\{\prod_{i=0}^{z-r-1}p_i\psi_{p_{z-1}}\left(1+\left[1+\frac{(n-2)\bmod\prod_{j=1}^{z}p_j}{\prod_{k=0}^{r}p_{z+1-k}}\right]\bmod p_{z-r}\right)\right\} \tag{1}$$

where $\Psi(n)$ is the indispensability value of the n^{th} pulse of a meter decomposed into prime factors p_0 x p_1 x p_2 x... with $p_0 = p_{z+1} = 1$, z is the number of stratification levels, $\psi_p(x)$ is the basic indispensability of the x^{th} pulse of a first-order bar with the prime stratification p, mod denoted the modulation operator, and $[x]$ is the whole-number component of x. Additionally the basic indispensability of the n^{th} pulse of a first order meter with prime divisor p is given by:

$$\begin{aligned}
&p=2\rightarrow\psi_{p=2}(n)=p-n\\
&n=p-1\rightarrow\psi_p(n)=[p/4]\\
&else\rightarrow\psi_p(n)=\left[q+2\sqrt{\frac{q+1}{p}}\right],q=\Psi_{p-1}(n-[n/p])
\end{aligned} \tag{2}$$

where $\Psi_{p-1}(x)$ gives the indispensability values for a bar of pulses numbering p-1, factorized and stratified with prime factors in decreasing order.

A similar result to the indispensability ranking described above can be obtained using the *metrical structure* of Lerdahl and Jackendoff [10]. By following a process as the one described by G. Toussaint in [5], one can calculate the relative strength of each pulse according to the stratification level it belongs into. The result exhibits a strong similarity to the output of the indispensability algorithm. However, the indispensability algorithm has the advantage of producing a unique indispensability score for each pulse distinguishing, this way, between pulses even when they belong to the same hierarchical level. This leads to a richer performance when mapping those scores to pulse weights and then to probabilities and amplitudes.

We then assign to each pulse a weight based on the stratification level it belongs to and its indispensability ranking. Each level i has its own distinct range of weights W_i (see Figure 2):

$$W_i(\max,\min)=\left(R^{i-1},R^i\right) \tag{3}$$

where R is a parameter related to the density of events of the resulted performance and ranges between 0 and 1. Equation (3) implies that the calculation of the ranges begins with the highest stratification level for i = 1 and continues until it reaches the metrical level defined by the user.

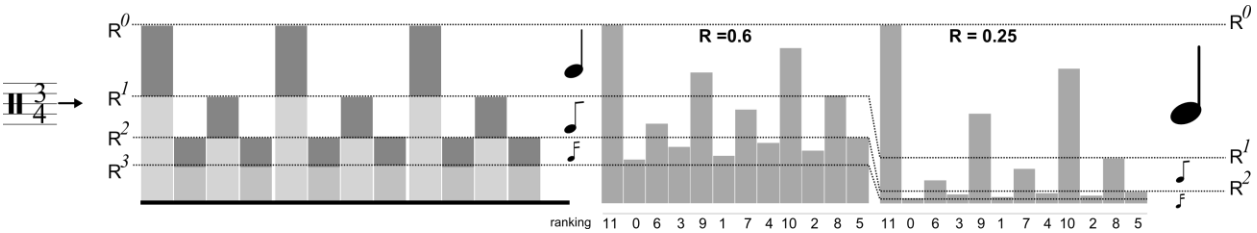

Figure 2: *Weight ranges (left side) and calculated weights for a ¾ meter stratified to the 16th note level.*

The pulse with the highest ranking value in each stratification level, i.e. the most indispensable, is assigned the maximum weight corresponding to the stratification level. The rest of the pulses in the stratification level are assigned smaller weights dividing the range into a corresponding number of equal parts. According to equation (3), for $R = 1$ all pulses have a weight equal to 1, while for $R = 0$ only the 1st stratification level survives.

Stochastic Performance. Once the weights for all the pulses are calculated, a performance is generated by cycling through each pulse comprising the metrical cycle and deciding if an event will be triggered in that position or not. The probability of triggering an event on a certain time position is derived by the corresponding weight according to a simple exponential relation:

$$p_\ell = n \cdot \left(W_\ell + \varepsilon\right)^M \tag{4}$$

where W_ℓ is the weight assigned previously to that pulse ℓ, n is a normalization factor, M is a user defined parameter related to the metrical strength and ranging between 0 and 1, and ε is a small offset which is non zero only for $W_\ell = 0$. The above equation functions as a "probability compressor", where for values of M close to 0, the differences in the probabilities are smoothed out, while for values close to 1, the original probabilities arise (see Figure 3). The small offset ε is needed only when the weight of a pulse is zero. In this case, in order for a finite probability to arise for $M < 1$, the base of the exponent needs to be different from zero.

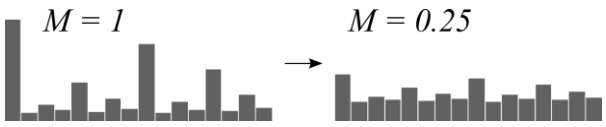

Figure 3: *Probabilities are exponentially scaled.*

The amplitudes of the triggered events are calculated independently from the probabilities. They are directly proportional to the pulse weights calculated for a fixed value of $R = 0.5$. The value was chosen such that the resulted rhythm is at the strongest metrical feel.

Generating Syncopation. Syncopation is introduced in the generated rhythm by "anticipating" pulses in stronger metrical positions. Events are triggered according to the probability assigned to the immediately following next pulse. At the same time, the amplitudes are also anticipated, so that the amplitude of a syncopated pulse sounds louder, thus creating a dynamic accent. The decision of syncopating pulses is made according to a user-controlled factor P_S, varying between 0 and 1, which represents the probability of anticipating a pulse and is related to the amount of syncopation in the resulted rhythm.

Two restrictions are imposed in order for the generated result to be more musical. The first restriction is a mechanism which forces syncopation to stop when too many consecutive pulses are syncopated; otherwise for values of P_S close to 1 the resulted rhythm would be just an offset version of the non-syncopated one. This competing mechanism forces pulses to not syncopate ignoring the

syncopation decisions. The probability of a pulse to be forced to not syncopate depends on the number of the consecutive pulses already been syncopated and to the weight of the current pulse. This way syncopation tends to resolve to the stressed pulses creating an effect of "off-beat" syncopation.

The second restriction relates to the fact that the feeling of syncopation is weakened when two consecutive events get triggered, one on a weak pulse and the second on the stressed pulse that follows. To avoid this behavior, a pulse is forced to always be mute and not trigger an event if the previous pulse was anticipating and an event was triggered. This rule is applied only when the consecutive anticipated pulses are less than three so that the "off-beat" type of syncopation is not canceled out.

Controlling density. The density of events *D* refers to the average number of triggered events per cycle. It is equal to the sum of the probabilities in all pulses:

$$D = \sum_{\ell = \text{all pulses}} p_\ell \tag{5}$$

The density of events and the metrical feel are by nature interrelated. This can be easily seen in extreme cases such as when the density is zero. Zero density means that no events are triggered which is, by definition, a non-metrical state. This degenerate rhythm is not at all characteristic of the meter. In fact, the outcome of the algorithm could belong to any meter and tempo. Similarly, the metrical feel is weakened when an event is triggered on every pulse, in other words when the density is maximum, and thus the meter can only be inferred from the amplitudes of the triggered events.

The density of events can be controlled by the parameter *R* in equation (3). Although the value of R cannot be used as a measure of the actual density of events it serves as an effective way of controlling it without affecting the metrical feel. How the actual density varies with *R* depends on the meter and the corresponding stratification. The probabilities are distributed to the pulses taking into account their order of importance and additionally the stratification level they belong to, preserving the hierarchy and structure of the meter for any value of *R*. The behavior of the probabilities when decreasing the value of *R* in equation (3) tends to keep the metrical feel strong. As the value goes to zero the pulses of the highest level of stratification take up more and more "space" squeezing the rest to lower probability values until all the lower levels, one by one, disappear leaving present only the 1st level (see Figure 2). Events are triggered on the "most important" pulses even for $R = 0$ keeping always the metrical feel strong. To further lower the density of events a linear scaling of the residues probabilities is used. On the other hand, the amplitudes of the triggered events are not affected by the changes in the parameter *R*. This way, when the density reaches its maximum, i.e. when $R = 1$ and events are triggered on every pulse, the character of the meter is made evident by the amplitudes of the triggered events.

Controlling metrical strength. The strength of the metrical feel depends, on the one hand, on the probabilities assigned to the pulses and, on the other hand, on the amplitudes of the generated events. A sense of meter is established when the events are triggered in important pulses (the most indispensable ones). The way the weights are calculated ensures that the more important a pulse is, the more often an event will be triggered in that position and this event will accordingly have higher amplitude. The more the indispensability relation is preserved among the pulses, the stronger the metrical feel. When all pulses have similar probabilities of triggering events and the amplitudes of the triggered events are random, not organized and do not establish a pattern, the resulted rhythm sounds random, not belonging to a specific meter. Therefore, in order to effectively control the strength of the metrical feel probabilities and amplitudes of the triggered events need to be adjusted simultaneously.

The probabilities can be directly manipulated through the exponent *M* in equation (4). The normalization factor *n* ensures that the density of events is not affected by the changes in the exponent *M*.

Values of *M* close to 1 result in probabilities proportional to the weights and, thus, in a stronger metrical feel. While *M* approaches 0 the probabilities are evened out and the metrical feel gets weaker. The amplitudes of the triggered events are proportional to the pulse weights at the strongest metrical feel, i.e. for a fixed value of *R*. In order to weaken the metrical feel as the value of M decreases, the amplitudes also get randomized but in a way that the distribution of amplitudes over time is kept constant.

Figure 4 summarizes the main aspects of the performance and their relation to the parameters of the algorithm.

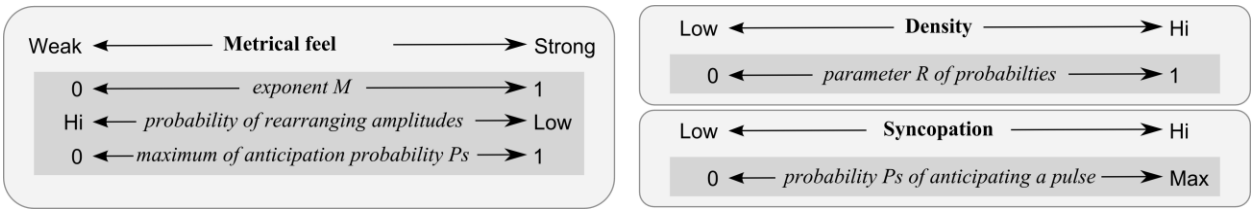

Figure 4: *A summary of the basic user controls and the corresponding parameters in the algorithm.*

Generating Variation. The generated rhythm varies and is non-repetitive due to its stochastic nature. The amount and type of variation can be controlled by restricting the mechanisms described above, namely the triggering of events and their syncopation. At each pulse, two different decisions are made. First, it is decided whether the pulse will anticipate the next one according to the amount of syncopation set by the user. Second, the triggering of an event is decided according to the probability of the corresponding pulse or the following one when anticipating. The variation in the resulted rhythm is controlled by restricting the number of such decisions that are allowed to change from one cycle to the next.

Two modes of variation have been implemented: the stable and the unstable. In the stable mode, the variation revolves around an initial pattern which is randomly generated. In the unstable mode, the rhythm departs from an initial pattern and follows a random walk. It evolves constantly into new patterns. In both modes, an initial pattern is generated at the beginning of the performance but the user can re-generate a new random pattern at any time, creating an abrupt change in the performance.

Events' articulation. The duration of the triggered events can be either fixed, in staccato mode, or can extend until the triggering of a new event, in legato mode. Syncopation is enhanced in legato mode by favoring the release of held events on stressed pulses even when no new event is triggered. A held event that was triggered on an anticipating pulse can be released on a following anticipating pulse, even if no new event is trigged. The probability of releasing an event on an anticipating pulse is proportional to its original weight.

Controlling the performance: the complexity space. The metrical feel, the amount of variation and the amount of syncopation form what we call a "space of complexity". A rhythm is considered to be simple, when the metrical feel is strong, the variations are kept to a minimum and there is no syncopation. On the other hand, when the metrical feel is weak or when syncopation is introduced into the rhythm or when the rhythm is constantly changing, then the rhythm is perceived to be more complex. Rhythmic complexity in this sense is attributed to combinations of different aspects of the rhythm: metrical strength, syncopation and variation.

We grouped the parameters of the algorithm related to complexity, namely the amount of syncopation, the exponent *M* and the amount of variation in the event triggering and syncopation

decisions, into a two dimensional map (see Figure 5). As one moves away from the center the resulted rhythm becomes more complex. The contribution of the above parameters to the complexity depends on the direction of the position vector on this map. Each of these parameters is a function of the position vector on this complexity plane, i.e. of the distance from the center and of the direction or angle between the vector and the horizontal axis (see Figure 5).

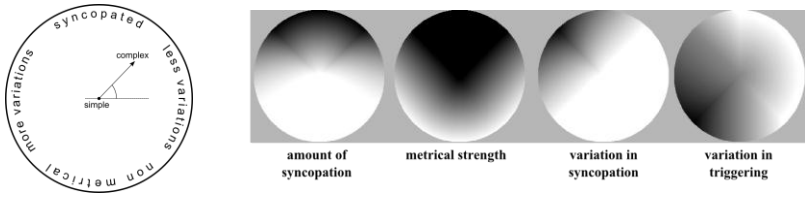

Figure 5: *Complexity space. On the left side a general description of the map is show. The contour plots of the functions to map the position coordinates follow on the right side.*

The functions of the various parameters are empirically set taking into consideration some basic restrictions derived from the nature of these parameters and our experience with various settings of the algorithm. Some of these restrictions are: i) when the metrical feel is low, syncopation is meaningless, ii) variation in the syncopation decisions apply only when the amount of syncopation is above a certain value, iii) when the amount of syncopation is significant the syncopation feeling is weakened by too much variation in the triggering decisions. Figure 5 shows the contours of the specific functions used for each control.

Max/MSP Applications

The algorithm was implemented as two Max/MSP externals. A Max/MSP bpatcher abstraction and a Max4Live device were developed around these externals. The kin.rhythmicator Max/MSP bpatcher abstraction is intended to be used in a variety of Max/MSP based applications and installations which implement some kind of rhythmic interaction. Such installations can take the form of virtual musical instruments, compositional tools or interactive installations. It is designed in a way that is easily integrated into any Max/MSP patch and can be controlled by several devices, from simple MIDI controllers to complex game controllers, and it is ready to directly trigger sound on any MIDI enabled synthesizer. The Max4Live MIDI device can be used as a compositional and/or performance tool, which dynamically generates rhythms. All parameters can be controlled through MIDI, automated with envelopes and saved together with the Live Set. They both implement all features of the algorithm described above.

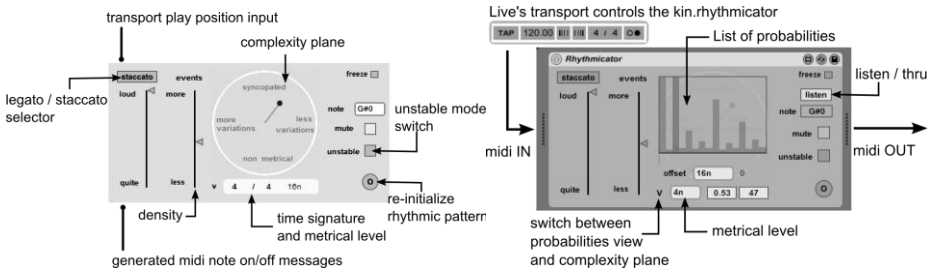

Figure 6: *The Max/MSP applications' interface.*

Conclusion and Future Work

An algorithm for generating rhythmic performances in a given meter was presented. The algorithm implements intuitive controls over various musical parameters that specify a stochastically generated performance. It can be thought of as a constrained improvisation that takes the place of a detailed music score. Among the musical parameters that can be effectively controlled are the metrical strength and the density of events. This has been made possible by using the stratification levels of the meter in the mapping process of the indispensability output of Barlow´s [7] formula to the weights and the corresponding probabilities. A syncopation algorithm based on the anticipation of pulses and which tends to keep a strong metrical feel is introduced. Variation in the performed rhythms is an innate quality of the algorithm arising from the use of probabilities in the performance. Future development of the algorithm and devices include the development of intelligent agents, which collaborate in generating a coherent output. The kin.rhythmicator Max/MSP abstraction and Max4Live device is available for download at our group website "http://smc.inescporto.pt/kinetic/".

Acknowledgments

This research was done as part of the project "Kinetic controller driven adaptive music composition systems", (ref. UTAustin/CD/0052/2008), supported by the Portuguese Foundation for Science and Technology (FCT) for the UT Austin| Portugal partnership in Digital Media.

References

[1] Bernardes, G., Guedes, C., Pennycook, B. "Style emulation of drum patterns by means of evolutionary methods and statistical analysis." *Proceedings of the Sound and Music Conference,* Barcelona, Spain, 2010.

[2] Eigenfeldt, A. "The Evolution of Evolutionary Software Intelligent Rhythm Generation in Kinetic Engine." *Proceedings of EvoMusArt 09, the European Conference on Evolutionary Computing*, Tübingen, Germany, 2009

[3] Martins, A. and Miranda, E. "Breeding rhythms with artificial life." *Proceedings of the Sound and Music Conference*, Berlin, Germany, 2008.

[4] Martins, A. and Miranda, E. "A connectionist architecture for the evolution of rhythms." *Proceedings of EvoWorkshops 2006 Lecture Notes in Computer Science*, Berlin: Springer-Verlag, Budapest, Hungary, 2006

[5] G. T. Toussaint, "A mathematical analysis of African, Brazilian, and Cuban clave rhythms," *Proceedings of Bridges: Mathematical Connections in Art, Music, and Science*, Towson University, Baltimore, Maryland, July 27-29, 2002

[6] G. T. Toussaint, "Generating "good" musical rhythms algorithmically," *Proceedings of the 8th International Conference on Arts and Humanities*, Honolulu, Hawaii, January 13-16, 2010

[7] Barlow, C. "Two essays on theory". *Computer Music Journal*, 11, 44-60, 1987

[8] http://cycling74.com/

[9] http://www.ableton.com/maxforlive

[10] Lerdahl, F. & Jackendoff, R. (1996). A Generative Theory of Tonal Music. Cambridge: The MIT Press.