SYNCOPATION AS TRANSFORMATION

Georgios Sioros

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Digital Media — Audiovisual and Interactive Content Creation

Dr. Carlos Guedes, Advisor

August 2015

Prof. Carlos Guedes, Advisor

[illegible]

In memory of Ariadne Magrabe

In memory of Marika Skouliou

Abstract:

Syncopation is a rhythmic phenomenon present in various musical styles and cultures. I present in this dissertation a set of simple rhythmic transformations that can serve as a formalized model for syncopation for the purposes of generation and analysis of music. The transformations are based on fundamental features of the musical meter and syncopation, as seen from a cognitive and a musical perspective. Based on this model, rhythmic patterns can be organized in tree structures where patterns are interconnected through simple transformations. The model was applied in the design of three listening experiments that explore the relation between syncopation and groove, i.e. the sensation of wanting to move when listening to music. A software plugin is presented as an application of the model. It manipulates the syncopation of audio and MIDI “clips” by automatically de-syncopating and syncopating its content.

Resumo:

Síncopa é um fenómeno rítmico presente em várias culturas e estilos musicais. Nesta dissertação, eu apresento um conjunto de transformações rítmicas simples que constituem um modelo formalizado da síncopa para fins de geração e análise de música. As transformações são baseadas em características fundamentais do compasso musical e síncopa vistos de um ponto de vista cognitivo e musical. Com base neste modelo, padrões rítmicos podem ser organizados em estruturas de árvore (tree structures) onde os padrões são interligados através de transformações simples. O modelo foi aplicado no projeto de três experiências de escuta que exploram a relação entre síncopa e “groove", ou seja, a sensação de querer mover quando ouvir música. Um plugin é apresentado como uma aplicação do modelo que manipula a síncopa de MIDI e áudio "clips" automaticamente.

TABLE OF CONTENTS:

ACKNOWLEDGMENTS

It is a genuine pleasure to express my deepest appreciation to my advisor, Carlos Guedes, whose expert orientation in key points of the research—from helping shaping the initial doctoral proposal in the early stages and providing me with important pointers in the literature, to his insightful criticism of the work and the final manuscript—was invaluable. Our discussions have been an indispensable source of inspiration. His trust was the ultimate condition that enabled me to unrestrictedly develop and explore ideas. I cannot imagine how this work would have been possible without his guidance.

I want to acknowledge all the members of the Sound and Music Computing research team, here in Porto, with whom I had the fortune to meet and work during the past years. Their warm welcome and constant support made me feel the city of Porto as home since the first day I arrived.

I am deeply indebted to Guy Madison and Fabien Gouyon for introducing me and guiding me in the research on musical Groove.

I would like to express my gratitude to Diogo Cocharro for his contribution in the development of the software included in this dissertation and in the experiments on musical Groove. Diogo was a companion in many adventurous discussions about syncopation, rhythm, music, and I am thankful to him for that as well.

I want to thank the members of the thesis jury, Peter Vuust, José António Oliveira Martins, Nuno Manuel Robalo Correia, José Miguel Santos Araújo Carvalhais Fonseca, as well as the President of the jury Jorge Manuel Pinho de Sousa, for their expert comments and sincere criticism.

I want to thank all the institutions from whom I benefited:

The Portuguese “Fundação para a Ciência e a Tecnologia” (FCT) for financially supporting me through the PhD grant with reference SFRH / BD / 51743 / 2011 in the framework of the UT Austin | Portugal Colab.

The Engineering Faculty of the University of Porto (FEUP) and INESC-TEC for hosting and supporting the research.

The New York University of Abu Dhabi for organizing and supporting my participation in the two International Workshops on “Musical Rhythm: Cross-Disciplinary and Multi-Cultural Perspectives” and for supporting my visit and stay in the campus in November 2014.

The Univesity of Umeå for supporting part of the experiments on Groove.

The University of Texas in Austin.

Finally, a special thanks to my family and friends, whose encouragement made it possible for me to complete this work.

LIST OF FIGURES:

CONTENTS OF THE CD-ROM

ExpA, ExpB, ExpC	MIDI and audio files used in the experiments of chapter 5.
Max4Live	The Max4Live devices described in chapter 6.
MaxMSP	Max/MSP (version 7) implementation of the algorithms described in chapter 4. Load the file [syncopation as transformation.maxproj] in Max/MSP in order to run the individual Max/MSP patches.
Tutorial_Videos	Short video-tutorials of the included software applications. s

1 INTRODUCTION

I started working on syncopation in 2010 when I joined the Sound and Music Computing (SMC) group in the INESC-TEC institute of Porto as a research assistant of Carlos Guedes. At that time, looping short excerpts of music was a very popular technique in the production of various music styles. The rapid advancements in digital technology played an important role in that as it made equipment such as the hardware samplers and the drum machines of the 90's more affordable, and later various computer software like Ableton Live that provided us with precise and easy ways of creating and reproducing short music loops. While I enjoyed the minimalistic qualities of repetitive music, I found that loops sounded motionless, monotonous and unnatural. For my own music projects, I preferred to develop and use generative algorithms that could produce rhythmic patterns with a strong sense of regularity without being excessively repetitive.

When I joined the SMC group, I found out that Carlos had similar ideas and wanted to create small generative plugins that could replace the static loops and generate rhythmic variation. We started working together on a few of his ideas. Our discussions often revolved around the question of how we can generate variation in rhythms so that they can deviate from the square metronomic prototype of a certain meter and sound natural. Two main topics dominated our discussions and experimentation: rhythms without a strong sense of meter that still evoke the sensation of a steady pulse and syncopation. A strong sense of pulse without clear metrical cues could evoke an ambiguous and unstable beat sensation. We understood that syncopation must lie at the other end: a particular way of diverging from the monotonous deadpan rhythms, which maintains a strong and steady metrical feel. It was then that I first attempted to create an algorithm that can generate syncopation.

Working on these ideas, I began to have a deeper understanding of what syncopation is and I realized its importance. Syncopation has been an essential phenomenon for several and diverse music styles of Western music, as well as in non-Western music, from the classical composers[1] to almost all dance and folk music. It has been fundamental in African music and Afro-American music and through it has conquer the entire world in the past century. It creates music tension and contributes to rhythmic complexity (Alf Gabrielsson, 1973; Gómez, Thul, & Toussaint, 2007). It can produce strong emotional effects such as enjoyment, happiness, and arousal (Keller & Schubert, 2011) and makes us want to move (Witek, Clarke, Wallentin, Kringelbach, & Vuust, 2014).

Almost two years later, I joined an ongoing research of other members of the SMC group on the movement inducing properties of music, what is often described by the less formal term groove[2]. At the time, a clear connection between syncopation and groove had not yet been established. From personal experience and in private discussions with musicians it was clear that it should be systematically investigated. In addition, there were already indications in support of it (Madison, Gouyon, Ullén, & Hörnström, 2011).

Guy Madison, Fabien Gouyon, Matthew Davis and I decided to work closely together and with the help of the rest of the SMC group we started a series of listening experiments that would shed light into the details of the relation between groove and syncopation. Our approach was to try to systematically generate music examples of various degrees of syncopation and have listeners rating the groove. The first problem we faced was how we can generate syncopation convincingly in an automatic and controlled way without affecting other qualities of the rhythm. The algorithms we had developed so far[3]

[1] Syncopation can be found as early as the 14th century in European music (Apel, 1946).

[2] Groove is often described as the "pleasant sense of wanting to move along with the music" (Janata et al., 2012).

[3] The most effective was the kin.rhythmicator (Sioros & Guedes, 2011b)

could generate musically meaningful syncopated patterns but were based on stochastic processes and we needed a more structured and predictable output for our experiments.

I started working on an algorithm that would introduce syncopation in simple piano melodies. This was the beginning of the syncopation model I developed and present in this study.

1.1 Motivation

I developed this comprehensive model of syncopation with the intention to become the basis for generative algorithms, whether they were stochastic or not, as well as for the analysis of rhythms. It was clear that my personal music projects as well as the work we were doing in the SMC group would benefit from a formalized and unified way of looking at syncopation. I could use the model in a wide range of generative applications. I was picturing applications that would produce from completely style agnostic generic rhythms, to style driven variation of user input patterns in order to replace static loops, all the way to rigorously designed algorithms that could be used in listening experiments in order to produce natural sounding music examples in strict “laboratory” conditions. The controls and specific aspects of each generative process would vary depending on the intended use but they could all be built upon a single formalization of meter and syncopation.

My motivation extends beyond the generative projects that I was already working on. I saw that the model could serve as a systematic way to study and describe the syncopation of entire corpora of music, classify and categorize rhythms or organize music material. Several tools could be developed for musicians and musicologists to help them in their studies of rhythm.

1.2 Description of the work

I started seeing meter and syncopation as the two most indispensable components of any metrical rhythm: meter as a solid ground upon which rhythms are built, and syncopation as a musical way crafted through centuries of musical practice of deviating from the colorless march like patterns.

A clear formalization of meter could already be found in the works of Lerdahl and Jackendoff (1983) and Clarence Barlow (1987). **The research problem was simple: how can I model syncopation, in order to algorithmically generate and analyze it?** My idea was then to approach syncopation as a transformation of unsyncopated rhythms.

The idea that syncopation can be described as a transformation came naturally. The notion that a non-syncopated rhythmic pattern can be syncopated was familiar among musicians. Even entire music styles could be considered as coming out of some other style by syncopating, like Dixieland music was a syncopated version of traditional two-step dance music (Peretti, 1994, p. 79). The musicologist Richard Middleton already used the concept of transformation to describe syncopation (1990, p. 212). However, neither the intuitive approach of musicians nor the more methodical one of musicologists could directly serve as a basis for computer algorithms. A more formalized and systematic model was needed[4].

1.2.1 Rhythm generation vs analysis

Seeing syncopation as a transformation of non-syncopated rhythms brought together the generative and analytical aspirations of my work. Syncopating a rhythm is certainly a generative process. However,

[4] Formalized transformations in music theory are not new. Most noticeable is Lewin's Theory of Generalized Musical Intervals and Transformations (2007), which focuses on the transformation of pitch intervals. Although Lewin's transformations can in principle be extended to durations, I followed a different approach in developing the syncopation transformations presented in this dissertation.

the reverse process of de-syncopation implies an understanding of the initial syncopation; before de-syncopating one has first to identify and describe the particular occurrences of syncopation in the given rhythmic pattern. Similarly, before syncopating one has to decide which notes one wants to syncopate and how to do it[5]. I understood that the way a rhythm is described should be independent of the intention to generate or analyze it. Ultimately, any rhythm can be codified as a non-syncopated pattern and a set of syncopation transformations—assuming of course it is felt in a certain meter. While rhythm generation starts from the non-syncopated metrical pattern and applies the syncopation, an analysis would operate in the opposite direction —beginning with a syncopated pattern and reaching its square counterpart characterizing all syncopation on the way.

This approach opens a vast range of applications where the boundaries between analysis and generation are blurred. For instance, one can develop applications in which the user input is automatically transformed and variation is generated. One such example is the *Syncopalooza* application (Sioros, Miron, Cocharro, Guedes, & Gouyon, 2013) described in detail in chapter 6. The user inputs a rhythm and the application generates a number of variations by transforming the syncopation found in it. The user controls the syncopation in the output by means of a slider while the output can contain syncopation automatically generated, not found in the user input.

Separating rhythm into two components, an unsyncopated pattern and the syncopation transformations, allows for more meaningful and effective variation of other aspects of the rhythm besides the syncopation. For example, varying the density of notes per bar is easier and more consistent when performed on a unsyncopated pattern; removing a note at the eighth note metrical level has a

[5] For example, a way to syncopate a melody might be to shift some quarter notes from their original metrical positions to earlier positions. One should decide which notes to shift and whether to shift them to the previous sixteenth or eighth note positions.

completely different effect on the rhythm if this note was actually a syncopated quarter note. Other applications could be based in the analysis of a specific corpus of music. One can then apply the syncopation characteristic of the corpus on a user input rhythm. In the last chapter, I discuss in more detail some possible applications that are based on the principle of separating the rhythm into an unsyncopated part and the part of syncopation as a transformation.

1.2.2 A limited set of elementary transformations

I designed the model so that the syncopation in a pattern can be broken down in a series of simple elementary transformation steps. If a complete and limited set of elementary transformations were determined, then one could generate all the syncopated patterns that correspond to a certain unsyncopated pattern by applying all possible combinations of the elementary steps. A significant part of my work focuses on carefully formalizing these elementary transformations which take the form of simple shifts of notes from their original position to a neighbor position. After determining the details of the shifts, applying them in different combinations gave rise to structures that resemble hierarchical trees. They can be used in generative or analytical applications, for instance, in defining distance measures between patterns based on how many syncopation steps separate them or in clustering rhythms depending on their corresponding unsyncopated counterpart. I describe the formalization of the limited set of transformations and the syncopation trees that emerge in chapter 4.

Another important aspect of the work is that the idea of formalized elementary transformations can be expanded beyond the modeling of syncopation. Other transformations of similar form, for example other kinds of note shifts with different effect and restrictions, can help connect patterns that cannot be connected through the syncopation transformations alone.

Finally, I need to mention that the scope of my research is not to exhaustively explore the applications of the syncopation model. Instead, I wanted to provide with a basic set of tools for developing new software and applications. To this end, I coded a Max/MSP external in C++ that can generate the elementary transformation steps and can serve as the basis for more elaborate algorithms. I include some examples of Max/MSP patches as supplementary material.

1.2.3 A generative model with a cognitive twist

The syncopation model presented in this thesis was developed in order to codify rhythm in an efficient way. I put a lot of effort in basing it on solid ground in order for it to be musically meaningful and its applications effective and relevant. At the beginning of the development of the model I imposed some restrictions to the transformations driven by the applications that the model was intended for. The most important restriction is reversibility, that is, the requirement that every transformation step should have its unique reverse counterpart.

However, in order to fulfill such restrictions, I often referred to the cognitive aspects of syncopation. For instance, I base myself on the fact that syncopation is something felt in a certain silent moment and is attributed retrospectively to a preceding sound event. With this in mind, I limited the syncopation transformation to always be an anticipation of a note whereas its reverse, the de-syncopation, should always be a delay. As I discuss in chapter 4, even though other choices were available, the reversibility restriction required that these choices could not be used all at once and the entire model should only be based in one of them. Perhaps some of the other choices could be practically equally useful but they lack some deeper justification. This way of thinking is found behind most of the choices related to the formalization of the syncopation as a transformation.

The model is primarily motivated by its generative and analytical applications in computer algorithms and does not aspire to be a cognitive model of syncopation. It is not intended to describe or to be perfectly aligned with the listener's perception of the rhythms. Instead, attention was given in providing with a systematic tool to musicians and music scholars. Nevertheless, some of the most important features of it are related to the view of meter and syncopation as cognitive mechanisms. For this reason, I present first, in chapter 2, a review of the relevant literature on meter perception and a definition of syncopation as a feeling followed in chapter 3 by more operational definitions used in syncopation measure algorithms.

1.2.4 A model for the symbolic domain

The syncopation transformations presented here are meant to work on symbolic representations of rhythmic patterns such as music scores, MIDI files and binary representations that provide with information about the timing of the music events. The elementary transformations were first designed for binary patterns, that is, for simple representations of rhythms consisting of a series of pulses of equal duration where each pulse either contains an onset or not[6]. Other more complex representations such as music scores could be converted into binary patterns in order to apply the transformations. This is the way the applications described in chapter 6 function.

In order to account for dynamic accents and stresses that are found in music performances, I extended the elementary transformations to operate on onsets that contain amplitude information[7]. This way, the model accounts for syncopation that arises from the placement of accents on the notes and not only on the position of the notes themselves. The extended model is equivalent to the binary version when all

[6] They are commonly called binary because they can be written as a binary string. For example, a rhythm containing four quarter notes can be written in binary form as a series of eighth notes: 10101010

[7] Such a representation would be an extension of the binary patterns that each pulse can take values other than 0 and 1. For example, if the two of four quarter notes of the previous example were stressed: 1 0 .5 0 1 0 .5 0

notes in pattern have equal amplitudes. Finally, in chapter 6, I present an example of an application that works on the audio domain, the loopalooza plugin. It converts the audio input into a binary pattern by detecting the onsets and segmenting the duration into distinct music events.

1.2.5 Two example uses of the model

I present in this thesis two main projects in which I used the syncopation transformations: 1) in generating music examples for listening experiments investigating the relation between syncopation and groove and 2) in music software for the automatic generation of variations of a user input.

In the first project, the research on groove that is presented in chapter 5, I used the model to generate syncopation in simple piano melodies as well as in more complex polyphonic music examples. I present this work here for two reasons. First, it demonstrates the application of transformations on specific music material. In the piano melodies, I followed a very controlled method of creating syncopation where the structure of the melodies was respected. In the polyphonic examples, I used a stochastic approach since the material was more diverse. Second, the conclusions drawn from the research, besides depicting the relation between groove and syncopation, reveal also more general aspects of syncopation and its effect on rhythm.

In the second project, which is presented in chapter 6, I used the model in the development of a software plugin that takes a user input and creates variations with various degrees of syncopation. I first developed the software to operate on MIDI data (Sioros, Miron, et al., 2013) but we later extended it to handle audio using an onset detection algorithm (Cocharro, Sioros, Caetano, & Davies, 2014).

1.3 Organization of the text

This dissertation comprises a literature review part, which includes chapter 2 and 3, and the part which I present and discuss the syncopation model in chapters 4, 5 and 6.

Chapter 2 presents a review on the rhythm perception related to meter and syncopation.

Chapter 3 presents the most important operational definitions that quantify syncopation. In this chapter, besides the already existing measures in the literature, I present two novel algorithms I developed to account for syncopation due to dynamic accents. I also discuss the concepts related to and problems in measuring syncopation.

Chapter 4 presents the details of the syncopation model. Here, I formalize the syncopation transformations and discuss how they were developed. I present the concept of syncopation trees, a structure to organize and explore syncopation. I present two algorithms based on the transformations, an algorithm for the analysis of the syncopation in a given pattern and an algorithm for generating syncopation trees. Finally, I extend the model to transformations of dynamic accents.

Chapter 5 presents the research on the relation between syncopation and groove. I discuss in detail the way the examples were generated and the important conclusions of the research.

Chapter 6 presents the generative software plugins that were developed in the context of this study and the details of the algorithms I used therein.

Finally, in chapter 7, I discuss the conclusions of the study, the original contribution and its implications in music creation and systematic analysis of music and some directions towards extending this work.

2 Rhythm, Meter, Syncopation

Definitions of syncopation often relate it to the musical meter (Kennedy & Bourne Kennedy, 2012; Randel, 1986). To understand meter and its relation to a rhythmic pattern and syncopation we must include physical properties of the rhythm—for instance Inter-Onset Intervals (IOIs), physical accents etc.—and the processes that interpret them in the listener's mind—grouping, categories, internal clocks and expectations. In this chapter, I am reviewing the most important models and definitions of meter viewed as a cognitive mechanism and their implications in rhythm perception. Finally, I present and discuss the feeling of syncopation arising from the interaction between rhythm and meter.

Musical rhythms elicit a sense of periodicity and regularity in listeners (Parncutt, 1987, 1994). Simple or complex music, even when it is not repetitive, it often evokes the sensation of a regular pulse in listeners evident when they tap in synchrony with the music. And often those pulses feature accents so that some of them are perceived as stronger than others. These structured, accented pulses are the basis for a definition of meter as a cognitive mechanism in the listener's mind (Jones, 2008; McAuley, 2010, p. 168).

In music notation, a similar notion of meter exists. The time signature divides time in regular intervals delimited by bar lines, e.g. a 4/4 time signature informs the performer of an underlying periodicity every 4 quarter notes or beats of the metronome while a 3/4 time signature implies a different periodicity of 3 beats. Additionally, the notated meter implies a difference of importance of those beats, the first beat being more important than the rest in a group. The performer has different ways of expressing the differences between the beats, e.g. by stressing specific events. However, the perceived meter in the listener's mind is often different than the notated meter and usually the listener does not know the score nor has he bar lines in his mind to mark time. The notated meter might or might not be conveyed to the listener during a performance (Honing, 2012, p. 381; McAuley, 2010, p. 167).

Moreover, a rhythm can be metrically ambiguous and more than one kind of meter could “fit” a particular sequence of sound events (London, 2012). Such ambiguities are often intentional in the composed music. A performer can use expressive tools to establish a clearer metrical context or he can choose to preserve and emphasize the metrical ambiguities of the score. Most performances are different from the notated music in several expressive aspects including deviations from the exact metronomic positions of note events and their relative loudness. The perceived meter, as a cognitive mechanism, is more dynamic than the notated meter in a score (London, 2012, p. 70), although, after a sense of meter has been established, it possesses a certain inertia and it is hard to contradict it (Large, 2008, p. 194). The listener could more easily change his or her metrical reference in the face of metrical ambiguity (London, 2012, p. 105).

Meter cannot be directly measured in a sound signal through some universal, objective and impartial algorithm. Although meter depends on the actual regularities existing in the heard rhythm, it is inferred by the listener through a cognitive mechanism in the mind (Honing, 2012). The listener essentially interprets a rhythm by constructing a metrical context. Meter is, therefore, subjective and may differ between listeners (Clarke, 1987a, 1987b; Fraisse, 1982; Parncutt, 1994).

Lerdahl and Jackendoff, in their *Generative Theory of Tonal Music (GTTM)* (1983), approach rhythm and meter from the music theoretical view using preference rules to infer the meter. They provide a way of analyzing the music score and constructing an appropriate metrical structure by using rules that reflect to a certain extent the preferences of an experienced listener (Lerdahl & Jackendoff, 1983). Usually the rules are adapted to a specific music style. For example, in Lerdahl and Jackendoff’s theory, the listener is supposed to be exposed to and to know the particular style of western classical music. In contrast, cognitive studies aim at understanding more universal aspects of rhythm perception and production, although they often acknowledge that the exposure of the listener to a specific music culture may have a strong effect on his perception of music. An important weakness of several cognitive studies is their ecological validity. In most psychological experiments, a set of stimuli is used to study the sensation they evoke in the listener. It is often the case that the stimuli used are too simple and poor in terms of musical context, in a way that they can influence significantly the results of the experiment (Cook, 2001, p. 153), although this trend seems to be changing in the last years.

Another rather more complex dichotomy exists: that between meter and rhythm. Meter and rhythm have been seen traditionally as separate and independent[8]. The term rhythm has been used both for describing a pattern of durations and the perception of that pattern (McAuley, 2010, p. 166). In this thesis, by rhythm, I will refer to the serial pattern of sound events[9] or durations[10]. In particular, I am concerned with musical rhythms, that is, with those series of durations that contain regularities that can evoke the sensation of a pulse[11]. The perception of musical rhythm involves two distinct mechanisms: serial grouping and periodic grouping. Serial grouping, referred to also as figural coding, groups a series of durations or events based on their proximity in time. Periodic grouping, referred to also as meter, is a structure of embedded or overlapping time spans that groups events at different time scales. Simply

[8] With few exceptions such as Christopher Hasty's Meter as Rhythm (1997).

[9] An event consists of more than just its onset and offset timings. It might include physical properties such as loudness, frequency and pitch or timbre.

[10] The serial pattern of durations is usually found in the physical sound pattern or signal. However, since in this thesis I am mostly concerned with symbolic representations of rhythms, I will refer by the term rhythm to any series of durations, even for nominal durations such as the ones found in a music score. However, as I discuss in the following section, the relation between such nominal durations and actual inter-onset intervals depend on the perception of rhythm and especially on the perception of meter.

[11] Parncutt (1987) distinguishes between rhythms and non-rhythms based on whether the durational pattern elicits or not a sensation of pulse. Here, I make a similar distinction between musical rhythms and any other rhythm.

put, a rhythmic figure groups adjacent events forming schemes of nominal durations like short-long-long (S-L-L figure). Meter, on the other hand, groups non adjacent events, in a periodic way, e.g. all events that occur every 4 ticks of the metronome (Parncutt, 1994, p. 412).

Recent theories in music cognition tend to bridge this dichotomy. They rely on neural oscillations in the brain that resonate with the rhythms to form a metrical framework that can persist even in the absence of the external stimuli. An important notion in understanding those theories is that of entrainment: "Entrainment is a biological process that realizes adaptive synchrony of internal attending oscillations with an external event" (Jones, 2008, p. 83). Advancements in neuroscience and brain imaging provide some evidence of such internal oscillations (Large & Snyder, 2009; Snyder, Large, & Penhune, 2009).

The neural oscillations manifest themselves when the listeners focus their attention in particular moments in time when they expect events to occur (Jones, Moynihan, MacKenzie, & Puente, 2002). Syncopation is felt when the sounding rhythm violates this expectation under certain condition. Not any violation of the metrical expectations is felt as syncopation.

2.1 From Sound to Categories

When a listener listens to a sound signal, he infers a number of different information on the rhythmic aspects of it. Some of this information is more congruent between different listeners while other is more subjective and may vary from one person to another. Listeners segment the sound signal into sound events. Each event consists of an onset, i.e. the moment when it is considered to occur, and a time interval commonly called Inter-Onset Interval (IOI), i.e. the time span between two onsets (Figure 2-1). The onset timing is critical in rhythmic perception and the IOI duration plays a more significant role than the duration of the sounds or tones, i.e. the duration between an onset and the following offset when the sound stops (Handel, 1993).

The IOIs are classified by the listener in discrete categories forming in that way a rhythmic pattern that consists only of basic durational categories (Fraisse, 1982; Dirk - Jan Povel, 1981; Schulze, 1989). A rhythmic pattern is essentially coded as a series of symbolic durations of simple ratios such as long and short, or 2:1, 3:1 etc., or **nominal durations** multiple of some basic pulse. The categories are influenced by the sensation of a pulse or meter that might be evoked in the listener, so that a certain duration or IOI might be interpreted as a different category when listened in a different metrical framework (Clarke, 1987a; Desain & Honing, 1989, 2003; Honing, 2012, p. 382; Jongsma, Desain, & Honing, 2004). In that sense, the musical score is a better representation of rhythmic patterns as it represents exactly those perceived categories instead of the absolute and accurate timing of notes.

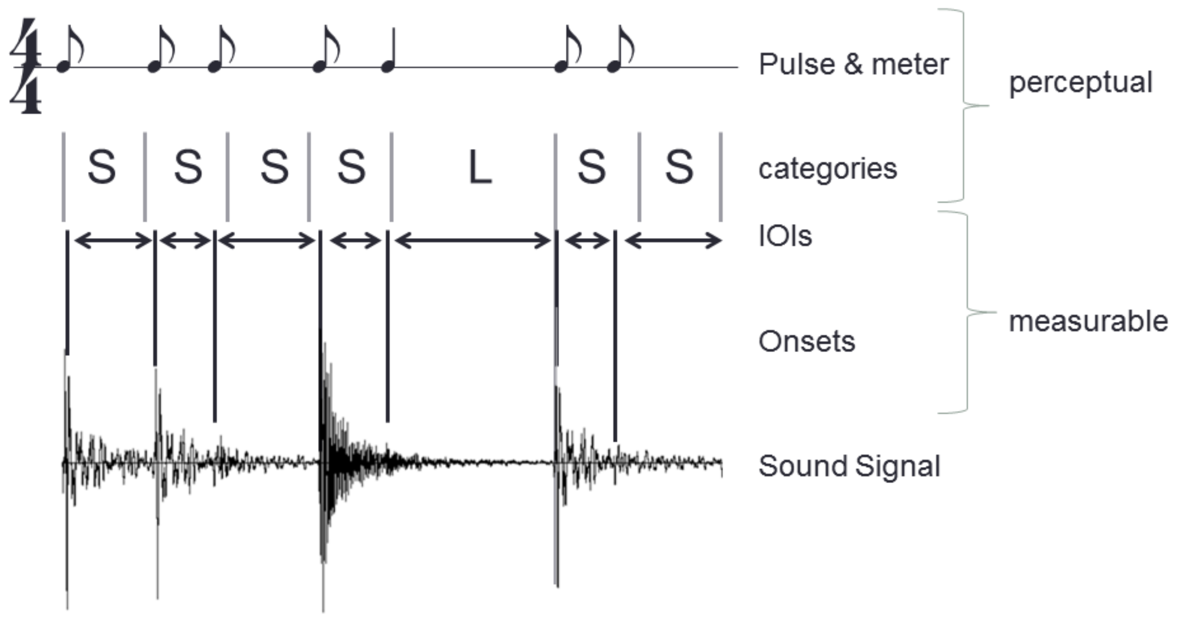


Figure 2-1: From sound signal to higher perceptual aspects of rhythm. The continuous sound is first segmented according to the perceived onsets. The IOIs are then translated to categories such as short (S) or long (L). These categories are strongly influenced and interpreted by the particular metrical framework in which they are heard.

While the IOIs are perceived as belonging to discrete categories, such as the note durations of a musical score, continuous time changes and deviations from the canonical version of those categories are usually understood at a different structural level as tempo changes and expressive timing (Clarke, 1987b). Rhythmic patterns with slight variations in the actual IOIs are considered as being the same pattern of durations performed, perhaps, with a different “feel” such as “laid-back” (Desain & Honing, 2003). The timing deviations are not ignored by the listener but are perceived in reference to the derived categories. The cognitive process that leads from the sound signal to a rhythmic pattern that can be recognized even when performed differently is called categorization (Desain & Honing, 1989; Honing, 2012).

Studies on rhythm categorization are commonly based on experiments where IOI intervals of rhythmic stimuli are varied systematically, interpolating between nominal ratios. The listeners have to identify the category that an IOI belongs to (identification task) or judge when two patterns are different (discrimination) or both. Clarke (1987a) varied the duration of two notes in a pattern between the ratios 1:1 and 1:2 and presented the resulted sequences to listeners in different metrical contexts, duple or triplet. He observed that when IOI were perceived as belonging to the same category (identification), the resulted sequences were also perceived as the same rhythmic pattern (discrimination).

On the other hand, the identified categories are not absolute but depend on a number of factors. Most noticeable is the dependence on the metrical context observed in Clarke’s study (1987a). Schulze (1989) observed later that the identified categories depended also on the training that the participants had prior to the experiments: after training, the participants were able to identify more categories.

The categories are not the result of a straight quantization process similar to the rounding that occurs in software sequencers. Each category has a different sensitivity to variations which depends on the actual rhythmic pattern that it belongs (Desain & Honing, 2003). Some IOIs in each specific pattern are allowed to vary substantially without the pattern being perceived as different. Other IOIs might be less tolerant to variations, with small deviations leading to an altogether different rhythm.

Moreover, rhythms sound more like what they are expected when they deviate from the metronomic IOIs. In Desain and Honing's experiment (2003, Table 3), the typical IOI of a category or rhythm, i.e. the set IOIs that was most frequently identified as belonging to a certain rhythm, does not coincide with the canonical or metronomic IOI. For example, a pattern of three equal intervals (1:1:1) was more often identified as that particular pattern when the durations were not exactly equal (0.33-0.31-0.36s) than when the ratios were actually equal (0.33-0.33-0.33s).

The cognitive process of categorization in rhythm is complex and at the basis of rhythm perception and the perception of timing in music. Time in music cannot be considered as a simple continuous quantity but it can neither be considered as a structure of simple, discrete and integer-related metronomic intervals. It is strongly influenced by our memory, our expectations and our culture (Jones, 2008).

2.2 METER AS EXPECTATION

Mari Jones et al. (2002) showed that the listeners judgments about pitch relations were more accurate in strong metrical positions. This result was interpreted as that the listeners form expectations about when it is most probable an event to occur and focus their attention on those moments. These expectations are reflected in the relative strength of the metrical positions. Meter focuses our attention on the "important" moments of the rhythm. In fact, Caroline Palmer and Carol Krumhansl (1990) found that frequently occurring events on strong metrical positions, relative to sparse events in weak positions, function as important cues for the perception of meter. More importantly, they showed that the most common metrical positions are also judged by listeners as better fitting; an event heard where it is expected fits better with what has already been heard. The relation of the metrical strength of a position to the probability of an event occurring in that position is also explored from a more theoretical point of view by Clarence Barlow (1987) with his indispensability formula. Barlow gives a theoretical score to each metrical position in a meter based on the time signature. This score describes how indispensable an event occurring in a certain position is and, in an indirect way, it describes how probable it is to find an event in that position.

The categorical perception of IOIs and the principle of simple ratios between adjacent IOIs have a strong impact on perceived temporal regularities. Richard Parncutt (1994) named *pulse sensation* any perceived regularity evoked in a listener by a rhythmic sequence. A *pulse sensation* may be evoked even when the heard periodicities deviate significantly from a mechanical or metronomic repetition.

A cognitive mechanism that explains how regularities are perceived was proposed by Dirk-Jan Povel and Peter Essens (1985). According to their model, an internal clock is generated by the perceiver and is adjusted to the distribution of events in the rhythmic pattern. Such a clock ticks in regular intervals and is assumed to be tuned to the primary and most salient *pulse sensation*. From all the possible clocks, the “best” clock is the one that has the least ticks on silence or on weak events, while at the same time it preserves isochrony similarly to a metronome. Events in the rhythm are distinguished between weak and strong based on grouping criteria. Strong events in this model are considered: i) isolated events, ii) the second of a group of two, iii) the first and last of a larger group (Dirk-Jan Povel & Okkerman, 1981). J. Devin McAuley and Peter Semple (McAuley & Semple, 1999) suggested that a clock can also be reinforced when ticks occur on strong events, thus introducing a positive feedback in the clock induction instead of relying only counter-evidence. Their tapping experiments showed that musician’s behavior is best modeled by a hybrid clock based on both confirmations and violations of the clock, while non-musicians’ behavior complies more with Povel and Essens’ clock. An internal clock is essentially an oscillation, adapted in phase and frequency to coincide with the heard events.

The biological process that “synchronizes” the internal clocks or oscillators to the external rhythm is called entrainment. IOIs existing in musical rhythm can elicit neural oscillations that resonate with the regularities in the rhythm (Large & Jones, 1999; Large & Kolen, 1994; Large & Palmer, 2002; Large & Snyder, 2009; Velasco & Large, 2011). These internal oscillations can be thought of as the embodiment of temporal expectations. The peak of an oscillation marks that point in time where an event is expected to happen. When these expectations are not verified the phase and frequency of the oscillation is adapted. At the same time that the internal oscillations are adaptive, they are also stable. When an oscillation is established, an event that contradicts it or the absence of an expected event does not disturb the oscillation. However, In the presence of contradictions that persist and that strongly suggest a different frequency or alignment of the oscillation to the events the oscillation will adapt again, forming new expectancies for the future. The above model of adaptive oscillators is a simple Dynamic Attending Theory (DAT) model for meter perception (Jones, 2008, p. 85).

Edward W. Large and Joel S. Snyder (2009) propose that pulse and meter could be explained by neural oscillations happening in the brain. Such internal oscillators can resonate with the external rhythmic stimuli and give rise to meter and pulse. Based on evidence in brain imaging studies they attempt to explain the perception of pulse and meter as the result of bursts of high-frequency neural activity. Although the evidence is not conclusive, their framework provides a starting hypothesis that remains to be verified by further experiments. More recently, a model of nonlinear oscillations was proposed for the detection of pulse and meter in heavily syncopated rhythmic patterns (Velasco & Large, 2011).

Several parallel oscillations can be evoked at the same time corresponding to different time spans in the rhythm. Edward W. Large and John F. Kolen (1994) proposed a model of several internal oscillators that track a number of periodicities in the auditory signal and combine them to track the meter. The model was later extended to account for more complex music examples with temporal irregularities (Large & Palmer, 2002).

Marie Jones (Jones, 2008, para. 25) suggests that several parallel oscillations can bind together in a metrical structure under certain conditions. Simultaneous oscillations that coincide in phase and have some simple frequency relation are gradually bonded together (Figure 2-2). The metric binding hypothesis aims at explaining the effect of culture or familiarity of the listener with certain music styles, where trained listeners might entrain faster and easier to a certain rhythmic style. This bond gets stronger over time forming a metrical cluster. When one of the oscillations is evoked by a rhythm, the rest of the oscillations are also present in the listener's mind even if no event in the rhythm directly suggests them. On the other hand, when an oscillation that directly corresponds to a particular time span in the rhythm does not resonate with the rest of the metrical cluster, it is suppressed and eventually will disappear.

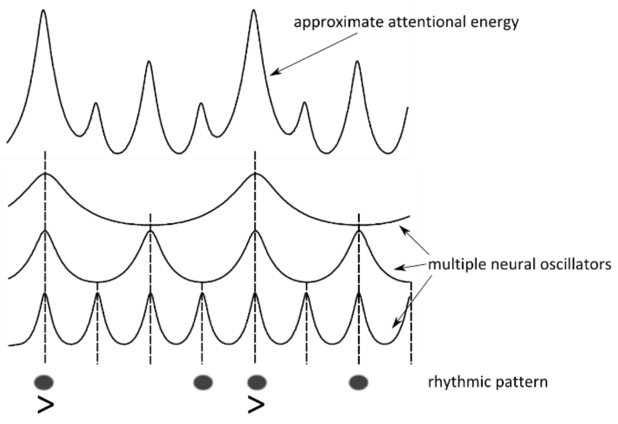


Figure 2-2: The metric binding hypothesis. Multiple neural oscillators are evoked simultaneously and bind together when they are aligned together. The combined activity of the oscillators forms an attentional framework.

In this way, meter, although a universal cognitive process, might be "biased" because of our exposure to the culture that constantly enforces specific temporal expectations. For instance, the findings of Palmer and Krumhansl (1990) mentioned above are indications of such biases. Theoretical predictions based on meter about the frequency of occurrence of note events were verified as statistical regularities in the music of four western composers of different music styles. Certain meters might appear more natural for listeners belonging to the western music styles just because those listeners have been more exposed to rhythms that contained the matching regularities. Other possible metrical structures might sound foreign and could be interpreted in a different metrical framework that is more common in the listener's musical environment. However, gradually he can build new corresponding expectations after being exposed sufficiently to the fresh and unfamiliar statistical regularities.

2.3 Metrical Hierarchy

Meter is not a mere temporal regularity of alternating strong and weak beats but is a hierarchical organization based on the perceived temporal regularities. According to Maury Yeston (1976), meter is the superposition of several simultaneous *rhythmic strata*, i.e. a periodic articulation of the music consisting of pulses in which all pulses of slower levels coincide with pulses in the faster levels (Figure 2-3). Similarly, the definition of meter given by Lerdahl and Jackendoff (1983) in *GTTM* consider meter as a hierarchical structure consisting of more than one level of beats, while "if a beat is felt to be strong at a particular level it is also a beat at the next larger level" (p. 19). The rhythmic strata of Yeston or the beats of Lerdahl and Jackendoff correspond roughly to the neural oscillation described in section 2.2. The theoretical models above can sometimes be over-simplified representations of a meter that results from complex cognitive processes. However, they can serve as the basis to create simple metrical templates that can be easily incorporated in computer algorithms for rhythm generation or analysis.

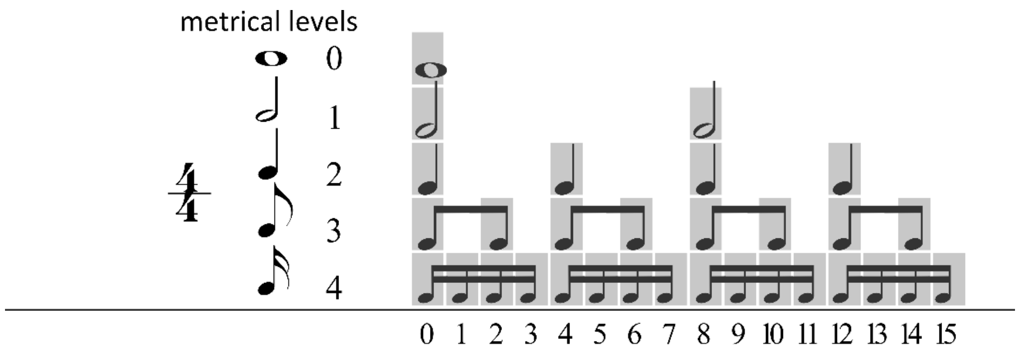


Figure 2-3: the metrical hierarchy of the 4/4 meter.

2.3.1 Constructing metrical templates

One can construct metrical templates based on the hierarchical structures described in *GTTM* (Lerdahl & Jackendoff, 1983), i.e. a template that describes the metrical structure of a certain time signature and that can be easily understood by and be incorporated in a computer algorithm. In this section a method is described that automatically constructs a metrical template based on the given time signature and fastest metrical subdivision. The method can be used to feed metrical information in various syncopation algorithms, such as the syncopation model of Longuet-Higgins and Lee (1984) and of course the syncopation transformations developed in this thesis.

The automatic construction of the metrical templates used in this thesis is based on the stratification of meter as described by Barlow (1987) . The time signature together with the fastest allowed metrical subdivision 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 2-4). 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.

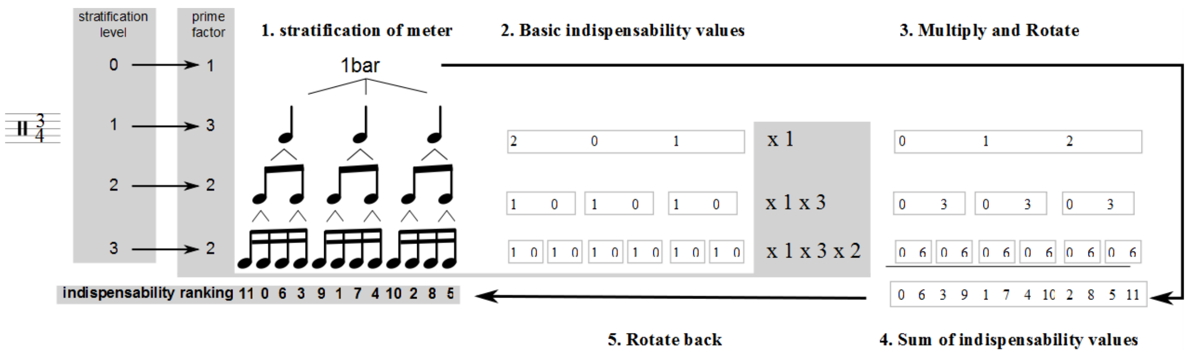

stratification level
prime factor
1. stratification of meter
2. Basic indispensability values
3. Multiply and Rotate
1bar
x 1
x 1 x 3
x 1 x 3 x 2
indispensability ranking 11 0 6 3 9 1 7 4 10 2 8 5
5. Rotate back
4. Sum of indispensability values

Figure 2-4: Stratification algorithm, the construction of a metrical template and the indispensability values

Barlow's stratification of meter distinguishes between simple and compound meters. 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. I devised an automatic way to generate the correct order of prime factors in simple and compound meters (Sioros & Guedes, 2011a). For meters with a ternary subdivision, the number of binary subdivisions that come before the ternary subdivision is determined by the time the numerator can be divided by 2. The corresponding number of "2s" is put first in the list of prime factors followed by the "3s" and the rest of the "2s". For instance, in the 6/8 meter, the numerator can be divided once by 2 (6/2 = 3) resulting in a 1x2x3x2 hierarchy. In any other case, i.e. in simple meters, the prime factors are always ordered in descending order, from the greatest to the smallest.

Barlow´s indispensability algorithm 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. 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. An example of how the algorithm is applied for a 3/4 meter is shown in Figure 2-4. 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.

2.4 Temporal boundaries of meter

Temporal boundaries are imposed to meter perception by our limits in sensing the beat and our ability in organizing the sounding events hierarchically. Several cognitive mechanisms are employed, each one at different duration scales. Meter integrates several cognitive mechanisms or *different kinds of time*, as London describes them, into one attentional framework. (see chapter 2 in London, 2012).

A lower boundary of 100 ms is found to be the shortest perceptual duration or fastest metrical subdivision in many studies of rhythm perception and performance (Friberg & Sundström, 2002; Repp, 2005a, 2005b, 2006). Parncutt (1994) quantified the salience of pulses and the metrical accents that arise from the interaction of the pulses, based on experiments he carried out himself and measurements he found in the literature. He found a lower boundary for the tactus or beat to be roughly double of the fastest metrical subdivision, around 250 ms. London (2012, p. 35) suggests that the 2:1 ratio could be due to a need of the listener to subdivide the beat in order for the sensation of that beat to be evoked by IOIs. This implies that the sense of a beat is only possible if we entrain to two or more periodicities at the same time and that beat and meter emerge together in the listener.

Parncutt restricts the tactus in the range between 250 ms to 2 sec with a peak salience between 500 ms to 700 ms. Although people can tap along a steady metronome tone at rates slower than 2 sec, they report it is a laborious task (Bååth & Madison, 2012). As the beat duration gets longer the variability in tapping duration is increased. More importantly, people tend more to react to the metronome tones instead of anticipating them (Repp & Doggett, 2007). As I discuss above, the beat sensation is linked to our expectations and therefore to how one anticipates when the next beat occurs. Bruno Repp (2005c, 2006) suggests a limit of 1800 ms as the point where a qualitative change happens from anticipating the next beat to simple responding to it after is heard, and therefore the sensation of beat being lost. Parncutt's thresholds correspond to tempo values between 240 to 30 bpm (beats per minute) with a peak value around 100 bpm. These tempo values are found in the Metronome and are used in music practice for centuries. Furthermore, they agree with research done on subjective rhythmization and spontaneous taping (Fraisse, 1982).

Synchronization is still possible at slower rates up to about one event every 3.3 sec (Woodrow, 1932) but slow periodicities do not induce a hierarchical clock. They can be produced by concatenating several faster beat periods (Clarke, 1999). The sensation of meter could therefore extend up to 5-6 sec, containing two or three beats, depending also on the musical context.

2.5 RHYTHM AND METER

Parncutt's term *pulse sensation* (1994) could bridge the simple theoretical models to the more realistic models of neural oscillations. Pulse sensation refers to the sensation of a salient periodic beat that emerges in the listener when listening to a musical rhythm. Overlapping pulse sensations give rise to meter. The hierarchical superposition of pulses corresponds to the metric binding of neural oscillations suggested by Jones (2008).

Parncutt (1994, p. 412) calls the metrical organization of rhythm the periodic grouping of events since meter groups together non adjacent events “such as all nth beats of a bar”. As mentioned above, aside of the periodic grouping, serial grouping exists, also referred to as sequential or figural grouping, which groups together temporally adjacent events. Both meter and serial grouping are based on the listener and reflect his perception of the organization of the rhythm[12], but they are distinct mechanisms in the mind (McAuley, 2010, p. 183).

A strong interaction exists between the grouping mechanisms and rhythm that is most noticeably reflected on the notion of accents. Some accents are physical like the loudness changes or the IOIs changes and they can be tracked back to the properties of the events in a rhythm. Oher accents are created in the listeners’ mind; they reflect the perceived temporal organization and they cannot be traced back to the rhythm. Nevertheless, as I discuss in the next paragraphs, we experience them as very similar or even indistinguishable to physical accents.

Frederik Jacobus Johannes Buytendijk and Adriaanus Meesters (1942) , and later, Dirk-Jan Povel and Hans Okkerman (1981), studied how variations in the time spans of sequences of otherwise identical tones are perceived as accents. It is concluded in their studies that when the IOIs alternate between short and long, the tones are grouped every two with the second tone of the group—the one that corresponds to the longer interval—perceived as accented[13]. Furthermore, the accent corresponds to an intensity difference of 4db. The inverse is also true; in an isochronous sequence of tones, if the intensity alternates between quiet and loud, the tones are again perceived in groups beginning with the loud tone. Interestingly, the interval of the last tone in the group—the one that corresponds to the quiet

[12] In contrast, as I mentioned earlier, in this thesis the term rhythm refers to the pattern of durations actually present in the music and it is based on the music signal.

[13] This is true only when the difference between the long and short interval is relatively large (>10%). Otherwise, the accent is felt on the event that corresponds on the short interval.

tone—is perceived as longer, although the actual IOIs are equal (Bolton, 1894; Woodrow, 1909). The above example is a manifestation of serial grouping in the form of a *falsely* perceived accent.

Meter also manifests as accents. The metrical accents are generated by the metrical expectations and the way we focus our attention at specific moments. This attentional focus is not directly bound to any one event and therefore cannot be tracked back to the properties of the sound signal. However, it closely resembles an actual physical accent. John Iversen, Bruno Repp and Aniruddh Patel found direct evidence of the similarity of physical and metrical accents in the listener's mind (2009). Using magnetoencephalography (MEG), they measured neural responses to tones of a rhythmic sequence, specifically in the upper beta range (20-30Hz). When listeners were instructed to impose a metrical interpretation to the rhythm by imagining the downbeat on a certain tone in the sequence, the beta response on that tone was stronger and resembled that of a physical accent due to an increased intensity.

A most interesting example of metrical accents is a phenomenon called subjective rhythmization (Bolton, 1894). According to this phenomenon, in a sequence of totally identical tones, such as the ticks of a metronome, some tones are felt as accented in a periodic way—for instance, every other tone perceived as more salient—similarly to the periodic grouping of meter. In fact, physiological evidence indicate that the grouping corresponds to the periodic focusing of attention due to metrical expectations (Brochard, Abecasis, Potter, Ragot, & Drake, 2003). This accent is perceived as an intensity difference, as a lengthening of the last interval of the group or both.

The behavioral and physiological studies I mentioned suggest that accents, whether physical or cognitive in nature, are indistinguishable in the way they manifest themselves. The above examples demonstrate the strong interaction between the three distinct elements, the rhythm and the serial and period grouping of events. Physical accents in the rhythm function as cues guiding the listener to group events while at the same time the internal grouping is used to project accents on the events that are not physically present. In this thesis I am focusing mostly on the periodic grouping of events, i.e. meter.

Traditionally, meter and rhythm are thought clearly as separate notions, making it easier to create robust constructions and models for them. The interaction between rhythm and meter is a cognitive process that projects the metrical expectations of the listener onto the rhythm. As I will discuss in the following section, syncopation arises from the placement of an accent when it is not expected. In this sense, syncopation is the result of the interaction between the musical signal and the listener[14] (Honing, 2012, p. 380). Recent neural oscillation models could bridge meter and rhythm by providing a possible explanation on how a strongly syncopated rhythm can evoke the sensation of a regular beat (Velasco & Large, 2011).

2.6 The syncopation feel

In this section, I present syncopation as a cognitive mechanism. Some important aspects of the syncopation feel are found in definitions given in case studies of particular music styles (Huron & Ommen, 2006; Temperley, 1999; Volk & Haas, 2013); nevertheless, I believe their principles are generalizable. Other, more formalized definitions of syncopations, approach it as a quantity that can be measured and that can be the subject of comparison between rhythmic patterns: patterns are considered to be more or less syncopated than others (Fitch & Rosenfeld, 2007; Gómez, Melvin,

[14] An important theoretical example where syncopation is treated as the most important interaction between a certain rhythmic pattern and a distinct and independent metrical context is the model of meter by Longuet-Higgins and Lee (1984). In this model, the preferred meter is determined by the syncopation that it creates and is the one that minimizes it. A description of the corresponding algorithm that measures the syncopation is given in chapter 3.

Rappaport, & Toussaint, 2005; Huron & Ommen, 2006; Keith, 1991; Palmer & Krumhansl, 1990). I discuss such formalized definitions and measures in the next chapter.

In the Harvard Dictionary of Music (Randel, 1986, p. 861), syncopation is defined as *"a momentary contradiction of the prevailing meter or pulse"*. The dictionary makes a distinction between *hemiola*, where the meter changes temporarily, for example from duple to triple, and syncopation that arises from the placement of certain notes or accents in such a way that they contradict the established meter. Two important points about syncopation are made in this definition. First, syncopation requires a sense of regularity that is related to the musical meter and that is expressed as an alternation of strong and weak beats. Second, syncopation does not undermine the meter; it contradicts it only *momentary* and it does not challenge the *prevailing* meter. David Huron, in his book "Sweet Anticipation" (2006, p. 303), expresses this clearly: "*Syncopation only challenges metric perceptions; it never annihilates meter. In order for syncopation to exist, it is essential to maintain normal (unsyncopated) metric expectations. […] Meter and pulse are preserved*." He describes syncopation as "*a certain class of violations of temporal expectation*" (p. 297). In other words, meter forms our expectations and syncopation surprises us! At the same time, syncopation patterns can be learned by the listeners, for example as part of a music style, so that they become predictable (Huron, 2006, p. 304).

David Huron and Ann Ommen, in their study of the American popular music, used the term "Lacuna" to describe syncopation (2006, p. 4). They defined the term as follows: "*a relatively strong metric position that does not coincide with a note onset and that is preceded by a note onset in a weaker metric position"*. This definition of syncopation arises from its cognitive aspects. An event in a weak metrical position raises the expectation of an event occurring in the next stronger position. When the expectation is failed we experience the feeling of syncopation (Huron, 2006, p. 295). In this sense, an event in a weak metrical position is usually bound to an event in the following strong metrical position;

when the expected strong note does not occur the weak one is left hanging (London, 2012, p. 107). The lack of a proper weak-strong binding of events is the basis of several of the syncopation measures described in the next chapter.

The above definition of syncopation is in agreement with David Temperley's metrical anchoring (2009, p. 8). According to Temperley, the probability of an onset in a certain metrical position depends on whether onsets exist in the surrounding positions of slower metrical levels. For example, a note at the eighth-note level is more likely to exist if it is preceded and followed by notes at the quarter note level. Furthermore, he provides statistical evidence that this probability depends predominantly on the existence of an onset in the following pulse and to a lesser extent on the previous pulse.

Approaching syncopation as a violation of expectations links it to the Dynamic Attending Theory, the neural oscillations of meter perception (Large & Jones, 1999) and the metrical binding. In Figure 2-5, an example of how the feeling of syncopation is evoked is given. The rhythmic pattern in the figure suggests a certain metrical hierarchy. According to the metrical binding hypothesis of Marie Jones (2008, para. 25), a number of neural oscillators tune in gradually to the regularities of the heard pattern and bind together. When the event marked as "onset" in the figure is heard, the fast oscillation is strengthened. In the following pulse, our expectation of an event is even stronger, since two more slower oscillators coincide with the fast one. If the expected event occurs, the oscillators would be reinforced by the corresponding IOIs in the rhythm and in their next respective cycle the expectation and attentional focus would be even higher. When an event does not follow, the slow oscillators are contradicted but not the fastest one. If they were independent, the slow oscillators would have to adjust their rate and phase to account for the early occurrence of the onset, for instance by increasing their rate. On the other hand, the fast oscillator has already coincided with the onset and the corresponding IOIs match its rate. However, all oscillators are bound together into a metrical cluster and cannot be independently adjusted. The oscillations persist as a cluster. The metrical feel continues to be strong at the same time that it is contradicted giving rise to the feeling of syncopation.

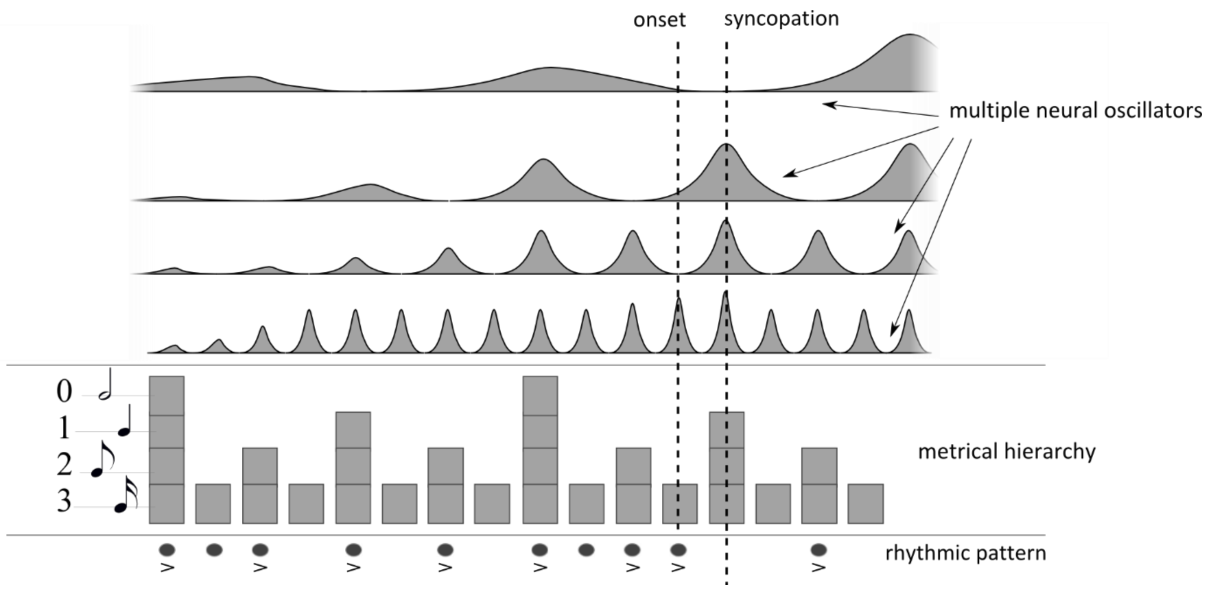


Figure 2-5: An onset in a weak position raises the expectation for an onset in the following strong position.

Therefore, syncopation could be the feeling that arises when the rhythmic pattern reinforces certain internal oscillations while at the same time contradicts others. The oscillations bind together in a hierarchical manner similar to the way the metrical templates are constructed. Hence, when an oscillation is contradicted then all slower ones are also contradicted. Conversely, when an oscillation is reinforced, all faster oscillations are also reinforced.

The rate of all slower oscillators must be some integer multiple of the fastest oscillator, but their phase can still be adjusted so that they coincide with most of the onsets. Excessive syncopation in the rhythm could provoke a simultaneous change of the phase of slow oscillators so that there is a better agreement between them and the onsets and minimize the syncopation. This would be felt as a shift of the downbeat but without a change of the character of the meter. Such shifts are often experienced in various music styles. The Longuet-Higgins and Lee algorithm (1984) for determining the meter in a given pattern is based on the same principle of minimum syncopation.

In contrast, the slightly different example of Figure 2-6 is not felt as syncopated. The event marked as "onset" is not aligned to the metrical grid and does not coincide with any of the already bonded oscillations. At the same time, it is not possible for a new oscillation to bind with a simple relation to the existing ones. Therefore, it is more likely that all oscillators will adapt simultaneously in order to incorporate the unexpected early timing of the event—which now feels more like expressive timing or an accelerando. In this example, the adjustment of the oscillations does not break their binding and their simple relation is maintained.

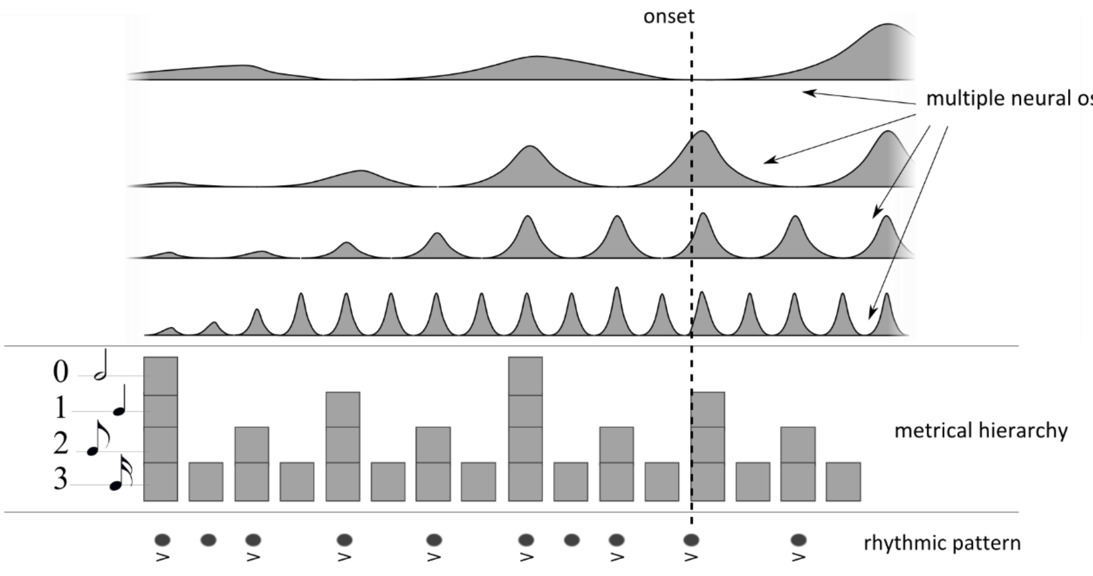


Figure 2-6: Not every contradiction of our expectation is felt as syncopation.

In the model he developed for exploring the uses of syncopation in rock, Temperley (1999) reaches a similar description of syncopation but from a generative perspective. In his model, syncopation is defined as the displacement of events from metrically strong positions to preceding weaker ones resulting in *hanging* events. He argues that the listener perceives those events as belonging to their original "un-shifted" strong metrical positions and that a de-syncopation process takes place in the listener's mind. Temperley's syncopation model is an extension of the preference rules that the *GTTM* uses to describe how a sense of meter is built in the listeners mind (Lerdahl & Jackendoff, 1983). Temperley's "extra" rules describe how one perceives some off-beat events as shifted forward and belonging to "stronger" metrical positions. The listener in that manner creates a de-syncopated version of the rhythm where the metrical feel is strong. He describes the actual rhythm as the *surface structure* and the corresponding de-syncopated version created in the listener's mind as the *deep structure.* In Figure 2-7, an example recreated from Temperley's study is shown.

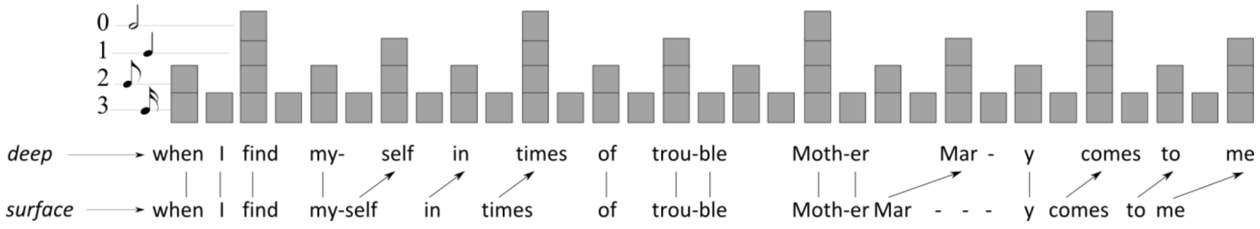

0
1
2
3
deep
when I find my- self in times of trou-ble Moth-er Mar - y comes to me
surface
when I find my-self in times of trou-ble Moth-er Mar - - - y comes to me

Figure 2-7: Example of Temperley's syncopation shifts in the Beatles song "Let it be". The syncopated syllables of the lyrics are shifted from their original position forward to stronger metrical positions where they do not syncopate revealing the deep structure.

2.7 Syncopation from dynamic accents

Dynamic accents are produced by the relative changes in the loudness of consecutive events. When an event is louder than its neighbor is felt as stressed. One can extend Huron's (2006, p. 295) definition of syncopation for binary patterns to account for dynamic accents: when an event in a weak metrical position is accented relatively to an event in the following strong position, syncopation is felt. The syncopation will be felt stronger when the onset in the weak position is stressed more and when the metrical levels of the two pulses are further apart. As I described in section 2.5, loudness changes are distinct from metrical accents. With this distinction in mind, syncopation can be described as "*a special kind of temporary mismatch of the phenomenal and metrical accents*".

In Figure 2-8, an example of syncopation arising from dynamic accents is shown. A loud event in pulse 0 is followed by an equally loud event in pulse 2. Although pulse 2 belongs to a weak metrical position, it is not against our metrical expectations to hear an onset there. However, the loud event in pulse 2 will raise our expectations for a loud event in pulse 4. When a much quieter event actually follows, the previous event sounds relatively stressed. This stress is against our metrical expectations and the feeling of syncopation is evoked. As the difference in loudness gets greater the feeling is more intense. In the extreme case that no onset is heard on the strong pulse, our definition of syncopation is equivalent to the definitions for binary patterns of section 2.6.

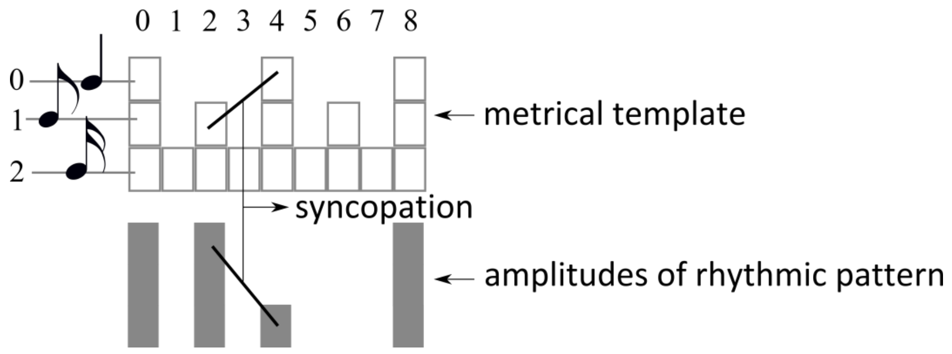


Figure 2-8: An example of a temporary mismatch of the dynamic and metrical accents. The grey bars represent the amplitude of each event in the pattern. The absence of a bar is equivalent to the absence of any event at the respective pulse.

2.8 Conclusions

This thesis is concerned with musical rhythms. I use the term rhythm to refer to a mere series of time intervals and the term musical to distinguish those series of durations that contain regularities which form metrical expectations in the listener’s mind (Parncutt, 1987). These expectations are strongly affected by various biases, such as cultural biases (Jones, 2008, para. 25) and of neurological basis (Grahn & McAuley, 2009). According to the Dynamic Attending Theory the listener focuses his attention on the moments he anticipates music events to occur. The attentional energy can be described by nonlinear periodic oscillators that are coupled to a regular external stimulus. This periodic focusing of attention creates the sensation of a steady pulse and manifests itself when a listener taps his foot or claps in sync with the music. The metrical binding hypothesis suggests that multiple oscillators are bound together when their phase and rate have simple relations. They are layered in a hierarchical manner giving in this way rise to the alternating strong and weak beats of the common musical meters.

A weakness of this cognitive model is that it does not account for non-isochronous meters, in which the beat (or tactus) includes pulses of different durations (for example 7/8). The same weakness is encountered in the stratification of meter in order to construct metrical templates; stratifying the meter consists in successively subdividing the bar into isochronous metrical levels.

Although the above cognitive and the music theory models of meter pose this strict restriction on metrical levels, they still provide a solid ground on which I build the operational definitions and algorithms presented in the following chapters of this thesis. At the same time, I believe they grasp the most essential cognitive and musical aspects of meter giving in this way a cognitive basis for the syncopation model I developed and the flexibility required for generative and analytic applications. For the purposes of this thesis, in chapter 4, I present a workaround on how one can automatically create metrical templates for non-isochronous meters. Justin London discusses comprehensively non-isochronous meters and the corresponding metrical structures in *Hearing in Time* (2012, Chapters 8, 9).

The syncopation feeling arises when the rhythm violates momentarily the listener’s expectations in such a way that a steady metrical feel is maintained. I presented here a hypothesis on how this is achieved: the onsets in the rhythm are placed in moments that they violate only the expectations described by the slower oscillations while they still enforce the faster ones. Because of the metrical binding of the faster and slower oscillations together, the period of the slow oscillations is not adjusted. On the other hand, the phase of the slow metrical levels could be shifted if the expectations are violated for a prolonged period. This could result, for example, into a shift of the perceived downbeat.

Syncopation is a manifestation of the interaction between the rhythm and the meter in the listener’s mind. Syncopation is felt in a moment of silence but it is retrospectively attributed to a sounding event (Huron, 2006, p. 260). The perception of the sounded event is changed not because of what is heard but because of the listener’s expectations about what should follow. The meter is formed pre-attentively in the mind (Ladinig, Honing, Háden, & Winkler, 2009) and expectations are formed effortlessly. In a sense, syncopation can unveil the listener’s expectations making him more aware of their existence by challenging them without annihilating them. As the metrical feel and the sensation of beat is often related to activity in the motor areas of the brain (Grahn & Brett, 2007, 2009; Grahn, 2009), it is possible that syncopation, by bringing forward the metrical expectations, prompts for a stronger tendency to move to the beat as a physical manifestation of the meter. In chapter 5, I present the listening experiments we conducted that relate syncopation to the sensation of groove which is defined as a pleasant feeling of wanting to move in synchrony with the music (Janata, Tomic, & Haberman, 2012).

Temperley’s idea of a de-syncopation process taking place in the listener’s mind could be related to the idea that syncopation is attributed retrospectively as a quality of the sounding event. The sounding event is initially characterized at the moment of its onset. However, later, when a strong silent pulse follows, the character of the event is changed in the listener’s mind. Now, he is not characterizing the event solely based on the timing of its onset but he directly connects it to the strong silent metrical position. One could say that he “transformed” the event. Although it is difficult to verify experimentally that he shifts the sounding event and the two structures described by Temperley actually exist in his mind, the idea that a direct transformation exists between a syncopated rhythm and a non-syncopated counterpart is musically relevant and seems natural. The syncopation model presented in the following chapters of this thesis is inspired by Temperey's shifts.

Finally, I presented a special case of syncopation arising from dynamic accents and linked it to the metrical template and the definitions of syncopation for binary patterns. This link will be useful in the following chapter where I propose two syncopation measures for patterns which include accented onsets.

3 QUANTIFYING SYNCOPATION

The formal definitions of syncopation presented in the previous chapter capture much of its essence. However, operational definitions or models are often required, for instance, in order to compare how syncopated two music excerpts are. Such definitions or models have a limited scope, inside the framework in which they are used and can only partially capture our intuition about the notion of "syncopation". They commonly treat syncopation as a scalar quantity that can be measured in short rhythmic patterns. Although they might fail to grasp at once all the nuances of the meaning of the term, they serve as excellent tools for studying syncopation and its effects in rhythm perception.

In what follows, I will describe six measures of syncopation and the corresponding operational definitions besides Huron and Ommen's (2006) Lacunas that I already discussed in the previous chapter as part of the definition of the syncopation feeling. Huron and Ommen counted the number of Lacunas in a number of music passages. They examined their relation to pre-Lacuna and post-Lacuna moments to study the evolution of syncopation in American popular music across five decades, from the late 19th to the early 20th century. The definition of syncopation as "Lacuna" served as a simple and effective statistical measure of how much the music passages of one period were syncopated relative to the passages of some other period. The measures presented in this chapter are based on more elaborate definitions and algorithms for identifying and quantizing syncopation.

From the six measures that follow, some can be considered rhythmic complexity measures, since syncopation contributes also to the perceptual complexity of rhythms. The first three and the last one—namely the Longuet-Higgins and Lee (1984), Toussaint's metrical complexity (2002) , Pressing's cognitive complexity (1997) and Keith's measure (1991)—make use of the entire metrical hierarchy. The other two—namely off-beatness (Gómez et al., 2005; Toussaint, 2004) and the weighted note-to-beat distance (Gómez et al., 2005)—use only the most prominent metrical level, i.e. the tactus or beat level. Pressing's cognitive complexity and Keith's measure define syncopation in terms of specific prototype patterns. The definitions found in these measures bare strong similarities with the simple *expectation principle* in Huron's cognitive definition that describes syncopation as a failure of the proper binding of weak-strong pairs of events.

Finally, I present two syncopation measures I developed in order to measure syncopation arising from dynamic accents in sequences of amplitudes. The first one is a direct extension of the Longuet-Higgins and Lee (1984) algorithm for binary patterns. The second one is a novel syncopation measure that is based on the idea that unexpectedly loud events in weak metrical positions are felt as syncopation.

3.1 Longuet-Higgins and Lee (LHL)

Christopher Longuet-Higgins and Christopher Lee (1984) proposed a syncopation model in order to determine the meter in monophonic rhythmic sequences. Their basic assumption is that a listener will always choose to interpret the sequence in a metrical context in which it is perceived as least syncopated. They identified syncopation in the pairs of events and the rests following them, so that a rest in a strong metrical position preceded by an event in a weak metrical position constitutes a syncopation.

The syncopation of Longuet-Higgins and Lee (abbreviated *lhl* hereafter) is based on the metrical hierarchy described in chapter 2. A metrical template needs to be constructed first in a similar manner as described in section 2.3.1. Based on the template each pulse is assigned a metrical weight. The weights are used to identify the individual instances of syncopation and assign them a score. Then the total syncopation of the rhythmic pattern is calculated as the sum of the individual scores. The weights are calculated as the negative of the metrical level index of each pulse[15]. For example, for a 4/4 meter, the weight of the downbeat would be 0, of the half bar -1, of the second and fourth quarter notes -2, of the eight notes -3, and so on.

According to the lhl, a pair of note onset with a following rest or tied note constitutes a syncopation when the weight of the rest is higher than the weight of the note. In this case, the syncopation score of the pair is the difference of the two weights. In the original lhl algorithm, the metrical template has the form of a tree where only the metrical positions that contain an onset or a rest are present (Longuet-Higgins & Lee, 1984, p. 429). In Figure 3-1, an example of such a tree is shown for a certain pattern in 4/4. According to the weights, the syncopations are identified and the corresponding scores calculated. The total syncopation for the pattern would be the sum of the individual scores: 2+1+3+1 = 7.

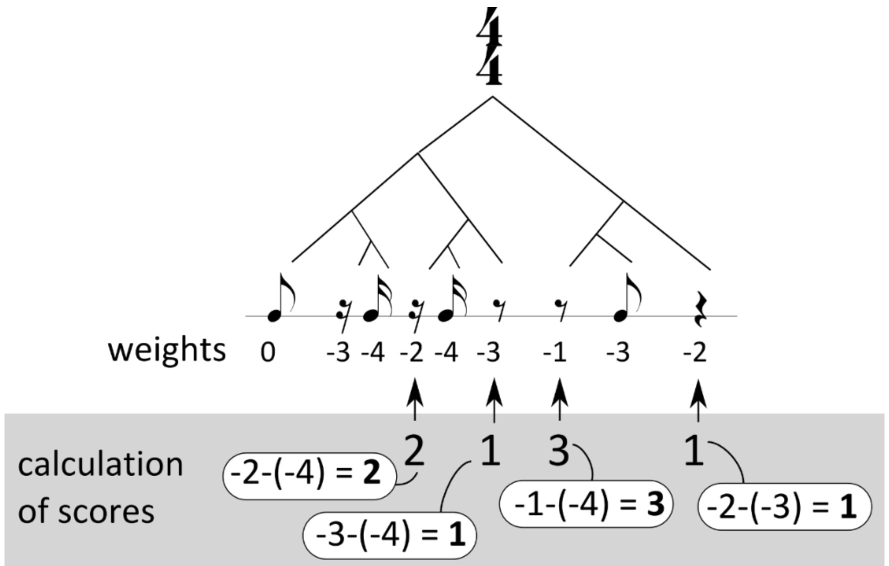


Figure 3-1: Example of the original lhl algorithm. A metrical template is first constructed in the form of a tree. Identified syncopations are marked with an arrow. At the bottom, the calculation of each syncopation score is shown.

[15] More accurately, the weight of a pulse is the negative of the index of the slowest metrical level it belongs to or, as it was defined by Longuet-Higgins and Lee (p. 430): "*The weight of a given note or rest is the level of the highest metrical unit that it initiates*" (They used negative metrical level indexes).

I describe here an alternative algorithm for calculating the lhl syncopation. One can convert a given rhythm into a binary pattern ignoring the duration of the notes and only considering the note onsets. First, a metrical template needs to be constructed as described in section 2.3.1 for the given meter resulting in a series of pulses of equal duration. Each of these pulses either carries an onset or is silent (Figure 3-2). The original lhl definition of syncopation is now rephrased: syncopation consists of a silent pulse at a slow metrical level preceded by an onset at a faster metrical level. The score of each individual syncopation is the difference of the metrical level indexes of the two pulses. A rule must now be applied in order for the alternative algorithm to be equivalent to the original lhl: the pair of silent pulse and onset constitutes a syncopation only if no other onset exists between them, neither a pulse of the same or slower metrical level as the silent pulse. The rule already exists in the original lhl calculations but it is hidden in the construction of the metrical tree. The metrical tree only has nodes in the metrical positions with an onset or rest while the metrical template includes all pulses at the fastest metrical level of the pattern.

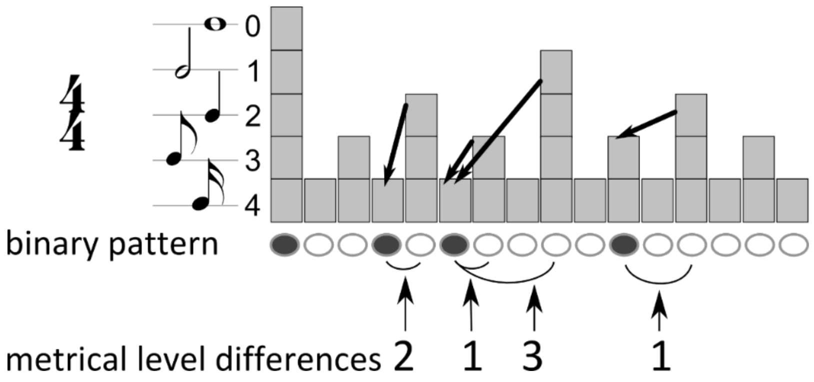


Figure 3-2: Example of an alternative algorithm for calculating the LHL syncopation scores. Pairs of pulses that constitute syncopations are marked with arcs. The bold arrows depict the corresponding metrical level differences on the metrical template.

The two algorithms above are completely equivalent. They only differ in the way the metrical templates and rhythmic patterns are represented. In the original lhl algorithm the metrical template is constructed particularly for each rhythmic pattern while the alternative presented here treats the templates and the patterns independently. For this reason, the alternative algorithm is easier to implement in computer software.

Apart from the original implementation by Longuet-Higgins and Lee, two more implementations of the measure exist (Fitch & Rosenfeld, 2007; Smith & Honing, 2006). Smith and Honing substituted the theoretical metrical weights in the Longuet-Higgins and Lee algorithm with weights derived from the human ratings taken from the study of Palmer and Krumhansl (1990), where listeners rated the fitness of events at different locations in the metrical grid.

3.2 TOUSSAINT'S METRICAL COMPLEXITY

Godfried Toussaint (2002) proposed a measure of metrical complexity that is closely related to the notion of syncopation. According to Toussaint, metrical complexity is the inverse of metrical simplicity which he calls *metricity*. Metricity is a measure of simplicity to the extent that meter is a highly organized and symmetric structure. Metricity is equivalent to the metrical strength of a pattern and is defined in the context of a metrical template as the one described in section 2.3.1. Each pulse of the template is assigned a metrical strength value that is the number of metrical levels it belongs to (Figure 3-3). Metricity is the sum of the metrical strength of each pulse carrying an onset. Metrical complexity is defined as the difference between the calculated metricity and the maximum metricity for the same number of onsets and the same template. In the example of Figure 3-3, the metricity of the pattern is the sum of the metrical strength of pulses 0, 3, 5 and 10. The maximum metricity for 4 onsets is the metricity of a pattern with onsets in pulses 0, 4, 8 and 12.

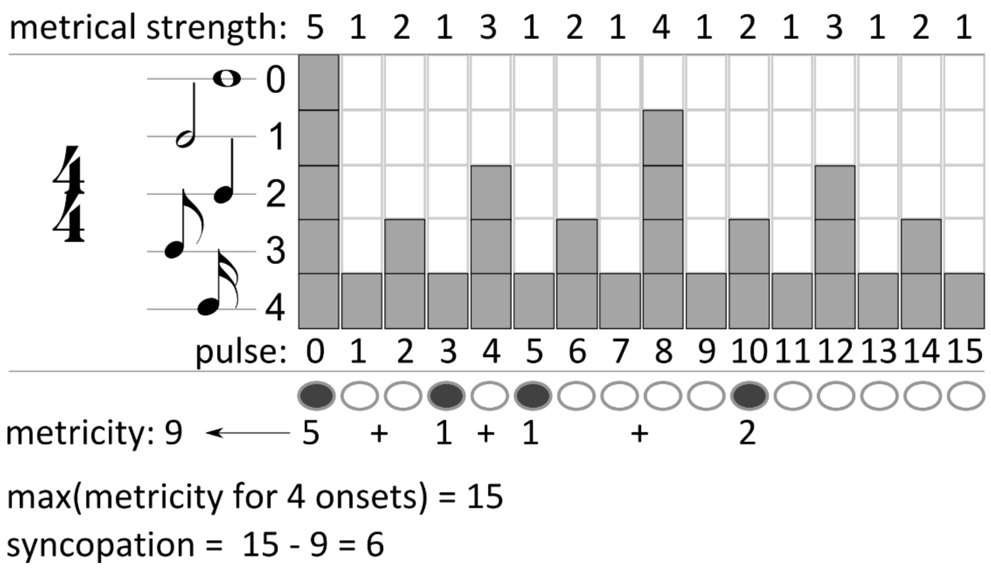

metrical strength: 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
0
1
2
3
4
pulse: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
metricity: 9 ← 5 + 1 + 1 + 2
max(metricity for 4 onsets) = 15
syncopation = 15 - 9 = 6

Figure 3-3: Toussaint's metrical complexity or metricity is the inverse of syncopation.

Toussaint's metricity is roughly proportional to how well a rhythmic pattern fits the meter. As discussed in chapter 2, the metrical strength of a pulse is related to the probability and to our expectations of an onset occurring in that pulse given that the rhythm is heard in the meter described by the template (Palmer & Krumhansl, 1990). Since syncopation is defined as a momentary contradiction of the meter or a violation of our metrical expectations (see section 2.6), we can assume that it is proportional to the inverse of metricity or, in other words, that is proportional to Toussaint's metrical complexity.

3.3 PRESSING COGNITIVE COMPLEXITY

Jeffrey Pressing (1997) proposed a measure of cognitive complexity of musical rhythms that is based on the syncopation at different metrical levels. He assigned a different cognitive cost to six binary patterns that serve as prototypes. A rhythm should be analyzed at each metrical level separately. Each unit of a metrical level, for example each quarter note duration at the quarter note level, is matched to a prototype and the corresponding cost is attributed. The total cost for a metrical level is the average of the costs of all units at that level. The overall cost is the sum of the costs of all metrical levels.

The six prototype patterns with their cognitive costs are shown in the table below:

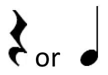
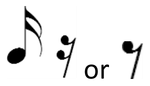
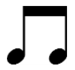
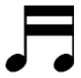
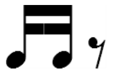
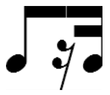
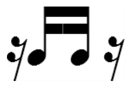
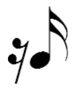
pattern	cost	description	Example at the quarter level	Example at the eighth level
null	0	Empty or only one onset at the beginning		
filled	1	An onset at each position		
run	2	An onset at the beginning followed by a run of onsets		
Upbeat	3	An onset at the beginning followed by consecutive pickup(s) to first position onsets		
Subbeat	4	Not defined for 4 pulse cycle		
Fully syncopated	5	Starting and ending at off beat positions		

Table 3-1: The six prototype patterns of the Cognitive Complexity measure.

The “subbeat” prototype is not defined in the original article which only states that it does not occur in cycles of 4 pulses (Pressing, 1997).

I will illustrate the process of pattern matching in the example of Figure 3-4 starting from the fastest metrical level. The sixteenth note metrical level is not included since it consists of single pulses and therefore by definition they can only be matched to the null prototype. At the eight note level, there are only three prototypes with two pulses length, as can be seen in Table 3-1. The first and sixth unit of the rhythm start with an onset and therefore their cost is 0. The second and third units match the fully syncopated prototype definition and their cost is 5. The rest of the units are empty. At the quarter note level, the first unit is an upbeat unit with cost 3. The second and third begin and end at off beat positions and therefore they are considered fully syncopated with a cost of 5. The last unit is empty. At the half note level, the first unit begins with an onset and contains 2 onsets in offbeat positions. Although this pattern does not match the upbeat definition above and since there is no definition for the subbeat pattern, I assigned to it a cost of 3. The second unit at the half note level is fully syncopated. At the whole bar level, the single unit starts with an onset that is followed by onsets in offbeat positions. As in the half note level, I assigned again the cost of 3. For each level the average of the costs is calculated and then summed across the levels to yield the total cognitive complexity of 11.25.

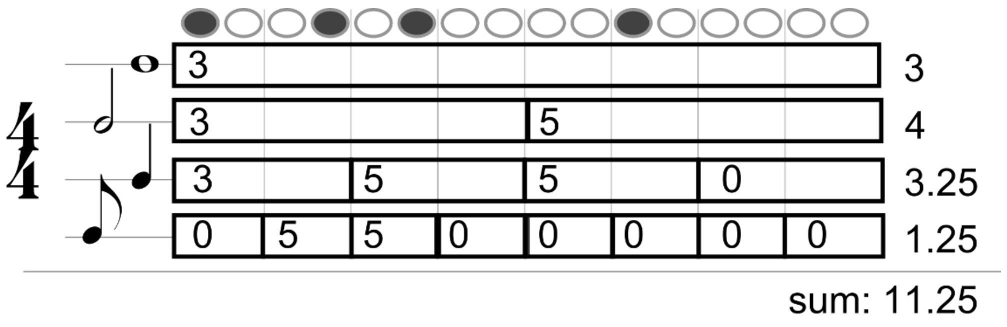


Figure 3-4: Calculation of Pressing's cognitive complexity for an example binary pattern. At each metrical level the individual costs are shown. At the right, the averages are calculated. At the bottom the sum represents the overall the cognitive complexity of the pattern.

The cognitive complexity is not a syncopation measure in itself. It uses the metrical hierarchy to identify and rank various syncopated patterns. For example, the prototypes *run* and *fully syncopated* fall in the definitions of syncopation provided in the previous chapter. On the other hand, the prototype *filled*, although it is not syncopated, it is assigned a cost different than 0. Similarly, the pattern *upbeat* is not syncopated (the sixteenth note in the example of Table 3-1 is followed by an onset on the beat since the examples are considered as loops). Nevertheless, its cost is greater than the syncopated prototype *run*. I included the cognitive complexity in this thesis as it directly connects the metrical hierarchy to the cognitive mechanisms of rhythm perception.

3.4 Off-beatness

The off-beatness (Gómez et al., 2005; Toussaint, 2004) measure counts the number of onsets that do not fall on any possible beat-cycle. It determines which pulses in the duration of the bar belong to a potential beat level and which ones do not. However, it does not consider the actual beat level of the bar but any *potential* beat level of any meter that fits the number of pulses in a bar.

In order to determine the possible beat-cycles, one needs first to find all the numbers that evenly divide the number of pulses in the bar. One can subdivide the bar length to shorter cycles according to those numbers similarly to the process of subdividing the meter in metrical levels based on prime factors (see section 2.3). The pulses that initiate these subdivisions are potential beats of a possible meter. The rest of the pulses are considered off beat positions. In that sense, the off-beatness is essentially polyrhythmic as it considers any possible metrical interpretation of a repeating pattern and not just the one that is determined by a time signature[16].

In Figure 3-5, I calculate the possible subdivisions or beat-cycles of a 16 pulses bar. The pattern shown on the figure has two onsets in off-beat positions, so that its off-beatness is 2. Although the off-beatness does not make use of the metrical hierarchy as the lhl measure and Toussaint's metrical complexity all subdivisions in the example are metrical levels of the 4/4 meter. This is due to the fact that a 16-pulses long cycle can only be interpreted as a binary meter. However, a 12 pulses long cycle can be interpreted as either binary or ternary since it can be divided by 6, 4, 3 and 2. Dividing by 6 or 2 results in metrical subdivisions of a 6/8 meter (corresponds to a metrical hierarchy of 2x3x2), while dividing by 6 and 3 results in metrical subdivisions of the 3/4 meter (corresponds to a metrical hierarchy of 3x2x2). Dividing

[16] For any possible metrical interpretation one needs to consider also all possible rotations of the pattern. However, in the off-beatness measure the pattern has a predefined beginning.

by 4, however, does not correspond to a time signature although it can be thought of as the quarter note subdivision of a 4/4 bar that is further subdivided into eighth note triplets.

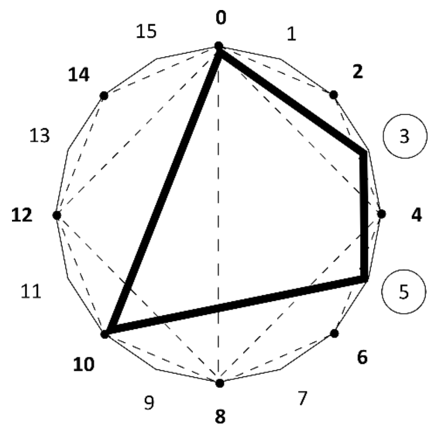


Figure 3-5: Calculation of the off-beatness of a pattern in a cycle of 16 pulses. The solid polygon represents the bar cycle. The dashed polygons represent the three possible subdivisions in 8, 4 and 2 parts. The bold numbers represent the pulses that belong in a beat-cycle. The bold line represents the rhythmic pattern with onsets on pulses 0, 3, 5 and 10. In circles, the position of the off-beat onsets are depicted.

3.5 Weighted Note-to-Beat Distance

The Weighted Note-to-Beat Distance (WNBD) syncopation measure (Gómez et al., 2005), in contrast to the measures presented above, does not make use of the metrical hierarchy. Instead, it defines syncopation as a contradiction to the most prominent metrical level, that is, the tactus or beat, and ignores all other metrical levels. Onsets that do not occur on the beat are considered as syncopated and the syncopation is felt stronger when the distance from the beat position (in terms of duration) is shorter. A double weight is given to the onsets that cross over a beat, for example a note that is tied to the following beat position.

The formal definition of the measure takes the following form. Let *d* be the distance from the closest beat position in units of beat duration (as a fraction of the beat duration). Then, the syncopation of an onset is:

$$Syncopation_{WNDB} = \begin{cases} 0, & \text{if } d = 0 \\ \frac{w}{d}, & \text{if } d \neq 0 \end{cases} \qquad 3.1$$

The weighting factor *w* is either 2, if the next onset is found in the duration of the following beat, i.e. if the onset is crossing over a beat, or 1 in any other case. The total syncopation of a pattern is the sum of the syncopation of each individual onset.

The calculation of the WNDB is illustrated in the example of Figure 3-6. The first onset has a syncopation of 0 as it falls on the beat. The second and third onset have a distance of ¼ from the nearest beat and they both have weight *w*=2 because the onsets that follow them fall in the duration of the next beat. Finally, the fourth onset is found exactly in the middle of two beats and thus the distance is ½. The next beat duration contains no onsets (the next onset would be on the downbeat if the pattern is repeated), so that the weight is *w*=1. The total syncopation for the pattern is 0 + 2/.25 + 2/.25 + 1/.5 = 18.

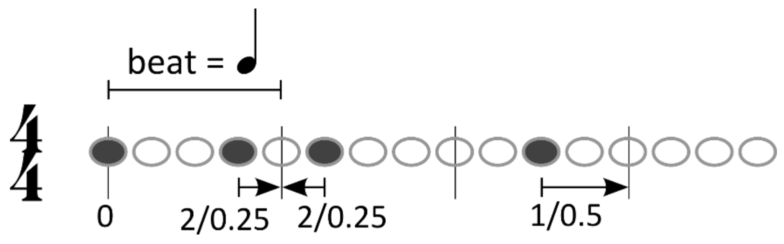


Figure 3-6: Example of calculation of the Weighted Note-to-Beat Distance syncopation measure.

3.6 KEITH'S SYNCOPATION MEASURE

Michael Keith (1991, p. 135) proposed a measure of syncopation that is based on matching in a rhythm one of three different patterns, which he calls *hesitation*, *anticipation* and *syncopation*. The patterns are defined in a similar manner to the WNBD of Gómez et al. (2005) according to when a note begins and ends in relation to the beat position. He considers all three as different forms of syncopation of different strengths with the *syncopation* pattern being the combination of the other two and felt the strongest.

The three patterns are assigned the arbitrary syncopation scores of 1, 2 and 3 respectively and are defined in the following table:

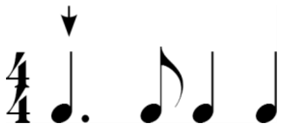
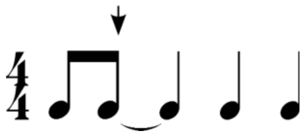
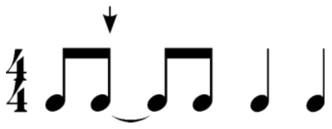
pattern	**description**	**example**
Hesitation (1)	IOI starts on the beat and ends at an off-beat position	
Anticipation (2)	IOI starts off the beat and ends on the beat	
Syncopation (3)	IOI starts off the beat and ends off the beat	

Table 3-2: Keith's definitions of the three types of syncopation in order of strength.

The total syncopation of a rhythm is the sum of all the syncopation scores of the individual IOIs.

An important part of Keith's measure is the definition of the beat positions. The beat in the original algorithm depends on the duration of the IOI under consideration and is, therefore, different for each onset. A limitation in Keith's measure comes about from the algorithm that determines the beat which is described only for meters that contain a number of pulses that is a power of 2. Thus, the algorithm generates only binary subdivisions of the meter.

I will present here an alternative definition of the beat level in the context of Keith's measure that can be applied on any meter as it uses the metrical hierarchy and subdivisions described in section 2.3: t*he beat level is chosen to be the metrical level that is equal in duration to the IOI under consideration or the level immediately faster in case a level with an equal duration does not exist*. For example, the first note of the hesitation example in Table 3-2 corresponds to an IOI of a doted quarter note and thus the beat level is the quarter note. For the second note in the same example, the IOI is an eighth note and thus the beat level is the eighth note level. Therefore, this IOI does not constitute any type of syncopation as it starts and ends on the beat. The alternative definition is completely equivalent to the original one when the number of pulses is a power of 2.

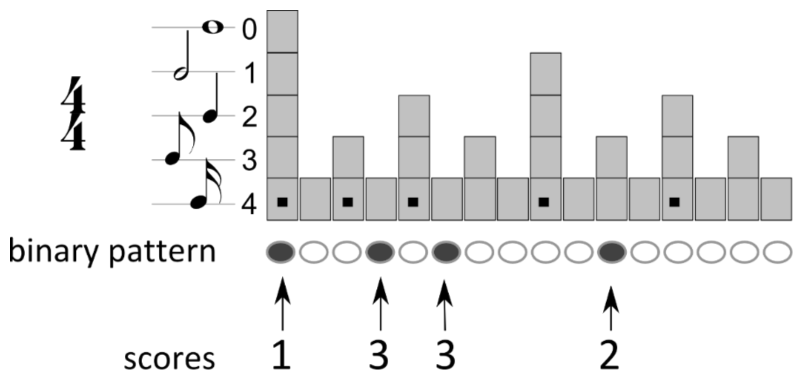


Figure 3-7: Calculation of Keith's measure of syncopation. The small black rectangles in the metrical structure represent the beat positions relevant to determining the syncopation types for the onsets.

In Figure 3-7, the calculation of Keith's measure is shown for an example pattern. The first onset falls on the first pulse which belongs to all possible beat levels. However, the corresponding IOI is a doted eighth note and thus the beat level appropriate to examine whether it ends on or off the beat is the eighth note. As the second onset is found on the sixteen note level the IOI begins on the beat and ends off the beat so that it is a *hesitation* pattern of a score equal to 1. The second onset corresponds to an IOI of an eighth note and therefore the appropriate beat level is again the eighth note level. The IOI begins and ends on sixteenth note pulses so that it is a *syncopation* pattern of a score equal to 3. The third onset corresponds to an IOI 5 pulses long. The immediately faster metrical level would be that of the quarter note (4 pulses long). Since the fourth onset falls on an eighth note pulse, the IOI begins and ends on off-beat positions and is a *syncopation* of score 3. The pattern is repeating, so that the last onset corresponds to an IOI of doted quarter note. Therefore, the beat level is the quarter note and the IOI begins off the beat but ends on the beat and matches the definition of the *anticipation* of score 2. The overall syncopation for the pattern is the sum of the individual ones 1+3+3+2 = 9.

3.7 Extending LHL to dynamic accents

Here, I extend the LHL measure presented in section 3.1 in order to make use of the amplitude of onsets when identifying and quantifying syncopation. This extended LHL algorithm is more suitable to measure syncopation in performed music, for example, in recorded MIDI files.

Similarly to the LHL measure, the extended LHL algorithm (hereafter E-LHL) uses a metrical template that associates each metrical position to the metrical level it initiates. Syncopation is then attributed to a note articulated in a weak metrical position (fast metrical level) and that is not followed by an onset of equal or higher amplitude at the next stronger metrical position, i.e. a position of a slower metrical level. Syncopation is quantified in the same fashion as in the lhl measure according to the difference of the metrical weights between the two positions. However, in the extended measure, syncopation is also scaled by the difference in amplitude of the two onsets, so that the louder is the first event in comparison to the second the more intense is the syncopation feel. The absence of an onset at a metrical position is equivalent to an onset of zero amplitude. The algorithm is designed in such a way so that the E-LHL approaches and coincides with the original LHL as the amplitudes pattern approaches a binary pattern (i.e. as all the amplitudes become equal). In the following, I describe the steps of the algorithm in detail demonstrating them in the example of Figure 3-8. Although the example is not particularly musical, it can be thought of as a small portion of some longer rhythm that is examined separately for the sake of simplicity.

In order to calculate the syncopation arising from the amplitudes in a given pattern, one needs to examine and calculate separately the syncopation for each pulse. To that end, amplitude differences are taken between each such pulse and the previous pulses that belong to faster metrical levels:

$$D_p^s = A_p - A_s, \qquad where\ p < s, \qquad A_p > A_s \tag{3.2}$$

where D_p^s the amplitude difference between pulse *p* and pulse *s*, and $A_{s,p}$ is the amplitude of pulse *s* or *p* respectively. In the above equation, pulse *s* is the silent pulse that the syncopation is felt and pulse *p* is the position of the preceding onset that belongs to a faster metrical level (see Figure 3-8 B or C for an example). Negative differences are ignored, i.e. syncopation is only calculated when the onset on the fast metrical level has a higher amplitude than the one in the following slower level, in accordance with the operational definition of syncopation given in section 2.7.

The amplitude difference is taken even if there are other onsets in between positions *p* and *s*. This is needed since a loud onset in an offbeat position will still syncopate even if a quite onset comes before the following beat. However, in the event that that the in-between onset has a high amplitude, it will "mask" the following quitter onset on the beat and weaken the syncopation feel. Accordingly, equation 3.2 takes a more general form in which the amplitude A_s is replaced with the highest amplitude value A_k of the onsets found between positions *i* and *j* (including position *j*):

$$D_p^s = A_p - A_k, \qquad where\ A_p > A_k, \qquad A_k = \max(A_{p+1}, \dots A_s) \tag{3.3}$$

If A_s is the highest amplitude then equation 3.3 falls into equation 3.2.

The calculated amplitude differences are scaled by the difference in the metrical weights of each pair of metrical position. Finally, they are summed to give the amount of syncopation in the pattern in much the same way that the metrical weights are summed in the LHL algorithm.

Let us examine the example of Figure 3-8 in order to demonstrate each step of the process. The pattern has 5 pulses which should be examined separately. Pulses 0 and 1 cannot syncopate since they are not preceded by pulses in faster metrical levels. The onset in pulse 2 is preceded by an onset in pulse 1 which belongs to a faster metrical level and is louder (B). Thus, a syncopation score is calculated as the amplitude difference $D_1^2 = A_1 - A_2$. The difference is then multiplied by the metrical weight difference of the two levels that, in this case, is 2-1 = 1. Pulse 3 cannot syncopate since it belongs to the fastest metrical level and thus it cannot be followed by a faster level pulse. Pulse 4 has three pulses in faster metrical levels that precede it (pulses 1, 2 and 3). As in the case of pulse 2 with pulse 1, the syncopation score between pulses 4 and 3 will be $D_3^4 = A_3 - A_4$ multiplied by the corresponding metrical weight difference: 2-0=2. The next pair to be examined is pulses 2 and 4. Between them comes pulse 3, which has a higher amplitude value than pulse 4 and masks it, so that the amplitude difference becomes $D_2^4 \rightarrow D_2^3 = A_2 - A_3 < 0$. As the difference is negative it is not considered syncopation. Similarly, in the pair 1 – 4, the amplitude difference becomes $D_1^4 \rightarrow D_1^3 = A_1 - A_3$, which now is positive. Again, it is multiplied by the corresponding weight difference between pulses 1 and 4: 2-0=2. As there are no other pairs to be examined the calculation finishes by summing the scaled amplitude differences:

$$Syncopation = D_1^2 \times 1 + D_3^4 \times 2 + D_1^4 \times 2 = (A_1 - A_2) + (A_3 - A_4) \times 2 + (A_1 - A_3) \times 2$$

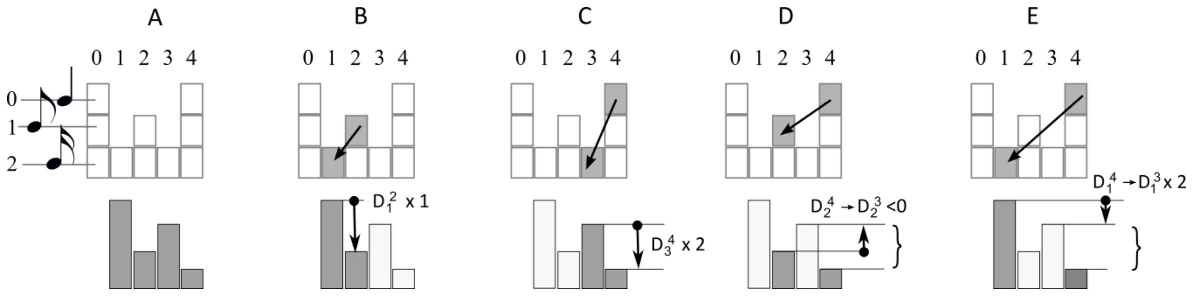


Figure 3-8: Example of the calculation of the extended LHL syncopation measure. A: The metrical template (top) and the pattern of amplitudes (bottom). In B, C, D and E the calculation of the amplitude differences for the syncopation of pulse 2 and 4 is given. The D_i^j arrows denote the amplitude differences that need to be calculated. The "}" marks the masking of the amplitude of pulse 3 on pulse 4. Each amplitude is multiplied by the difference of the metrical weights.

The reason for the amplitude "masking" becomes clear if one treats a binary pattern as a pattern of amplitudes with a single amplitude level. In that case, the two algorithms—the original binary LHL and the E-LHL—should coincide. Consider the example of Figure 3-9; if no amplitude masking takes place then all onsets preceding a silent pulse of a slow metrical level would syncopate (B). However, the original LHL measure for binary patterns only considers a single previous event as syncopating, in agreement with the common experience of syncopation. In the example, the binary LHL counts two syncopations in pulses 2 arising from the onset in pulse 1 and in pulse 4 arising from the onset in pulse 3. The E-LHL however takes an additional amplitude difference between pulses 1 - 4. If no masking takes place then the 1-4 difference is counted as an extra syncopation on pulse 4. If pulse 3 masks pulse 2, then the amplitude difference between pulses 1 and 4 will be replaced with the difference between 1 and 3 which is zero since all onsets have equal amplitudes in a binary pattern. If we let the amplitude of pulse 3 to go gradually to zero, the 1-4 difference gradually increases and the 3 – 4 gradually decreases. At the limit where $A_3 = 0$, the two syncopation measures agree again, since now, no onset exists on pulse 3 and the original LHL will count the two syncopations between pulses 2 – 1 and 4 – 1 (A).

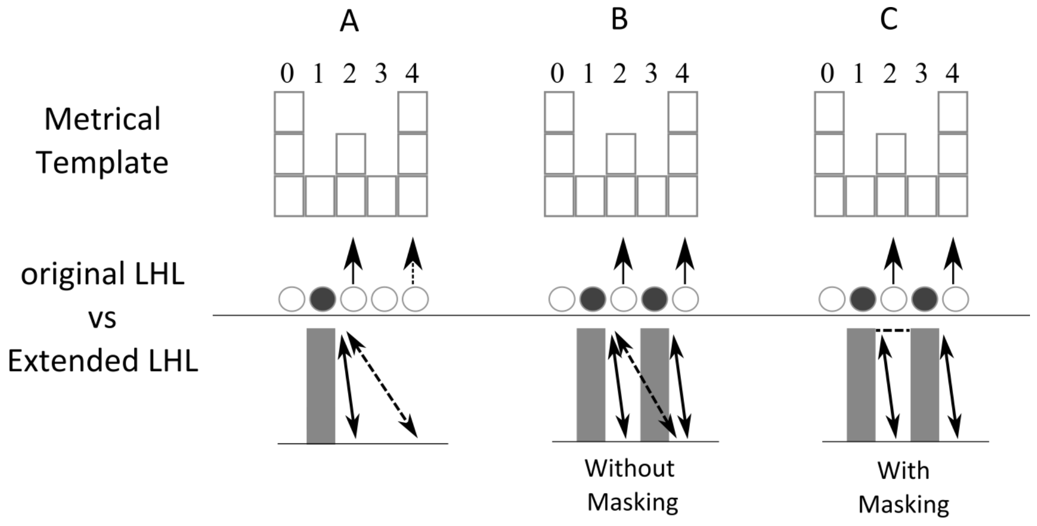


Figure 3-9: The E-LHL approaches the original LHL when the pattern of amplitudes is a binary pattern. The masking effect is necessary in order to avoid miscalculating non-existing syncopations (right).

3.8 The Sioros-Guedes (SG) measure

In the metrical anchoring principle of the generative model of David Temperley (2009, p. 8), notes at fast metrical levels are much more likely to appear when they are preceded or followed by notes in a slower metrical level (metrical anchoring). Metrical expectations are closely related to the probability of occurrence of events (Palmer & Krumhansl, 1990), so that the unlikely notes that are not surrounded by notes on slower metrical levels can be thought of as violating these expectations and therefore consist a case of syncopation. The SG algorithm presented in this section uses this principle and extends it to the amplitudes of the onsets in order to measure syncopation.

The SG algorithm (Sioros & Guedes, 2011c) attributes syncopation to the relatively loud notes articulated in weak metrical positions: an event articulated in a weak metrical position is expected to be preceded and followed by equally loud or louder onsets in stronger metrical positions; an event in a weak position that is surrounded by quieter events stands out as unexpected and is felt as syncopating. The louder is an onset and the weaker its position, the stronger the syncopation. When the weakly articulated event is not surrounded by other onsets the syncopation is maximized. On the other end, when it is surrounded by louder events, it becomes part of a regular pulse and does not produce syncopation.

Therefore, two parameters and their values need to be determined for each onset in order to quantify syncopation: (1) how louder than its neighbours the onset is and (2) how the metrical strength of its position is translated into a syncopation weight. The syncopation of the onset is then calculated as the product of these two factors. Summing the syncopation of all onsets in a pattern yields the total amount of syncopation. As in the LHL and the E-LHL measures the strength of the metrical positions is determined in a metrical template as described in chapter 2.

Notes are considered loud and therefore generating syncopation if they are louder than the notes in their vicinity. In order to quantify how much a note stands out as loud, amplitude differences are taken between that note and the notes in the preceding and following metrical positions. These differences are taken for all metrical levels the note belongs to.

As the hierarchical template dictates, a metrical position belongs to the level it initiates and all faster ones. For example, an 8th note position belongs also to the faster 16th note metrical level. Therefore, for an 8th note position, amplitude differences are taken from the preceding and following quarter note positions (the quarter note positions belong also to the faster 8th note level) and the preceding and following 16th note positions. Only the onsets of notes are considered and the offsets are ignored. The absence of an onset in a metrical position is equivalent to an onset of zero amplitude, so that amplitude differences are taken even when a position does not carry an actual onset. Amplitude differences are constrained to positive values. Negative differences, i.e. when the onset under examination is quitter than the onsets in its vicinity, are set equal to 0. Then, the average of each pair of amplitude differences—from the preceding and following positions—is calculated for each one of the metrical levels ℓ a note belongs to.

$$D_{i,\ell} = Average\left(D_i^{preceding\ in\ \ell}, D_i^{following\ in\ \ell}\right) where\ D_i^j = A_i - A_j \qquad 3.4$$

Subsequently, the lowest of the calculated averages is kept and the rest of the metrical levels are ignored for that pulse so that the syncopation score for pulse i is:

$$Score_i = min_\ell\left(D_{i,\ell}\right) \qquad 3.5$$

In that way, it is enough for a note to be preceded and followed by louder notes in one metrical level, to not be considered loud or syncopating. For example, an eighth note between two quarter notes of equal amplitude will not be considered as a loud note even if the surrounding 16th notes are quiet or absent.

In the SG algorithm, two additional weighting factors are used during the calculation of the amplitude differences. In what follows I describe why these factors are necessary and how they are formalized.

First, a weighting factor is introduced that is related to the metrical level difference between the pulses. An event in a weak metrical position can evoke a stronger syncopation feeling if it is found in the vicinity of a much stronger position. For example, a 16th note will be felt as more syncopated if it is heard just before a silent downbeat than just before a silent 8th note in the last beat of the bar. In order to account for this effect, the amplitude differences are scaled by a factor proportional to the difference of the metrical levels of the corresponding positions.

$$W_{level}(i,j) = 0.5 + \frac{\left|\ell_i - \ell_j\right|}{8}, \qquad W_{level} \in [0.5, 1] \qquad 3.6$$

This weighting factor has a minimum value of 0.5 and increases as the metrical level difference increases. However, it is constrained to a maximum value equal to 1 which is reached when the metrical level difference is 4. A more general form of the above equation is the following:

$$W_{level}(i,j) = \begin{cases} 1 - \mathrm{r} + \dfrac{|\ell_i - \ell_j| \times r}{4} & , |\ell_i - \ell_j| \leq 4 \;\&\; r \in [0,1] \\ 1 & , |\ell_i - \ell_j| > 4 \end{cases} \qquad 3.7$$

The factor r appearing in the equation is related to the range of the weighting in relation to the level difference (Figure 3-10). When $r = 0$, the level difference plays no role in the calculation of the syncopation and the weighting is constant and equal to 1 for any level difference. When $r = 1$, the weighting factor has the largest range. Equation 3.7 falls into equation 3.6 for the default value of $r = 0.5$.

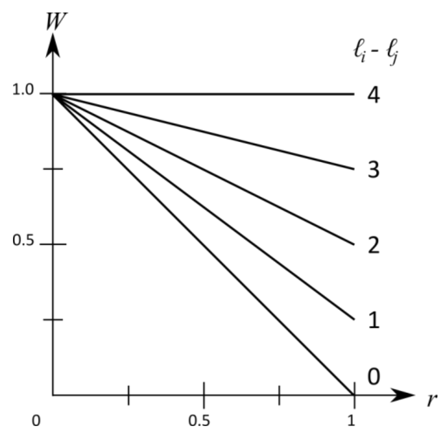


Figure 3-10: The weighting factor $\boldsymbol{W_{level}}$ as a function of the factor r and the metrical level difference $|\boldsymbol{\ell_i - \ell_j}|$.

Second, a lower weight is given to the difference from the preceding metrical position than from the following one:

$$Wprec \; \in [0,1] \qquad 3.8$$

An important aspect of rhythmic patterns is the direction in which they are always performed. Time in music, as in everything else, flows in only one direction. Pulses succeed one another in a specific order. The relation of a pulse to its previous pulse is not equivalent to that to the following pulse. Two equally loud events one after another create the impression of an accent on the second event rather than on the first (Buytendijk & Meesters, 1942; Dirk-Jan Povel & Okkerman, 1981).

Equation 3.4 now becomes:

$$D_{i,\ell} = Average\big(W_{level}(i,j) \times Wprec \ \times D_i^j,\ W_{level}(i,k) \times D_i^k\ \big),$$
$$where\ j = preceding\ in\ \ell\ and\ k = following\ in\ \ell \qquad 3.9$$

Since it is difficult to accurately specify the value for the two weighting factors, the two relevant parameters, i.e. $Wprec$ in equation 3.8 and r in equation 3.7, are left as user defined variables that can be adjusted according to the musical context that the SG syncopation is used in.

The second of the two quantities needed in order to calculate the syncopation is the syncopation potential for each metrical position. The syncopation potentials are essentially the inverse of the metrical strength, so that weak positions in fast metrical levels have higher syncopation potential. They are calculated according to following formula:

$$Sp_i = 1 - w_i = 1 - 0.5^{\ell_i} \qquad 3.10$$

where Sp_i is the syncopation potential at the metrical position i, and ℓ_i is the metrical level index starting with 0 for the slowest metrical level and incrementing by one for each faster subdivision. The range of S_i is always between 0 and 1, with 0 corresponding to the slowest metrical level and approaching 1 asymptotically as one goes to faster metrical levels. One can generalize the above formula using a parameter r':

$$Sp_i = 1 - w_i = 1 - r'^{\ell_i} \qquad 3.11$$

That way, one can tune the SG syncopation measure to the particularities of the music material that is analysed. For example, if the rhythmic patterns that are to be measured, originate from a MIDI stream recorded during a performance, the parameter r' can be used to match the velocity curve of the MIDI instruments.

Finally, the syncopation for each note is calculated as the product of the minimum average amplitude difference calculated above and the syncopation potential for the corresponding metrical position. The syncopation of the melody is the sum of the syncopation values of all notes.

To summarize, the syncopation in the SG measure is computed in 5 steps:

1. Amplitude differences at each metrical level a pulse belongs to:

   Multiply each amplitude difference by the weighting factor of equation 3.7.

   Multiply the difference from preceding by the corresponding factor in equation 3.8.
2. Calculate the average at each metrical level between the preceding and following pulses $D_{i,\ell}$ as in equation 3.9.
3. Syncopation score for each pulse = the minimum average difference value (equation 3.5)
4. Multiply the score of each pulse by the syncopation potential of equation 3.11.
5. Total syncopation score = sum of all scores

In Figure 3-11, an example illustrates the steps 1 to 3. The steps 4 and 5 for the same example are shown in Figure 3-12.

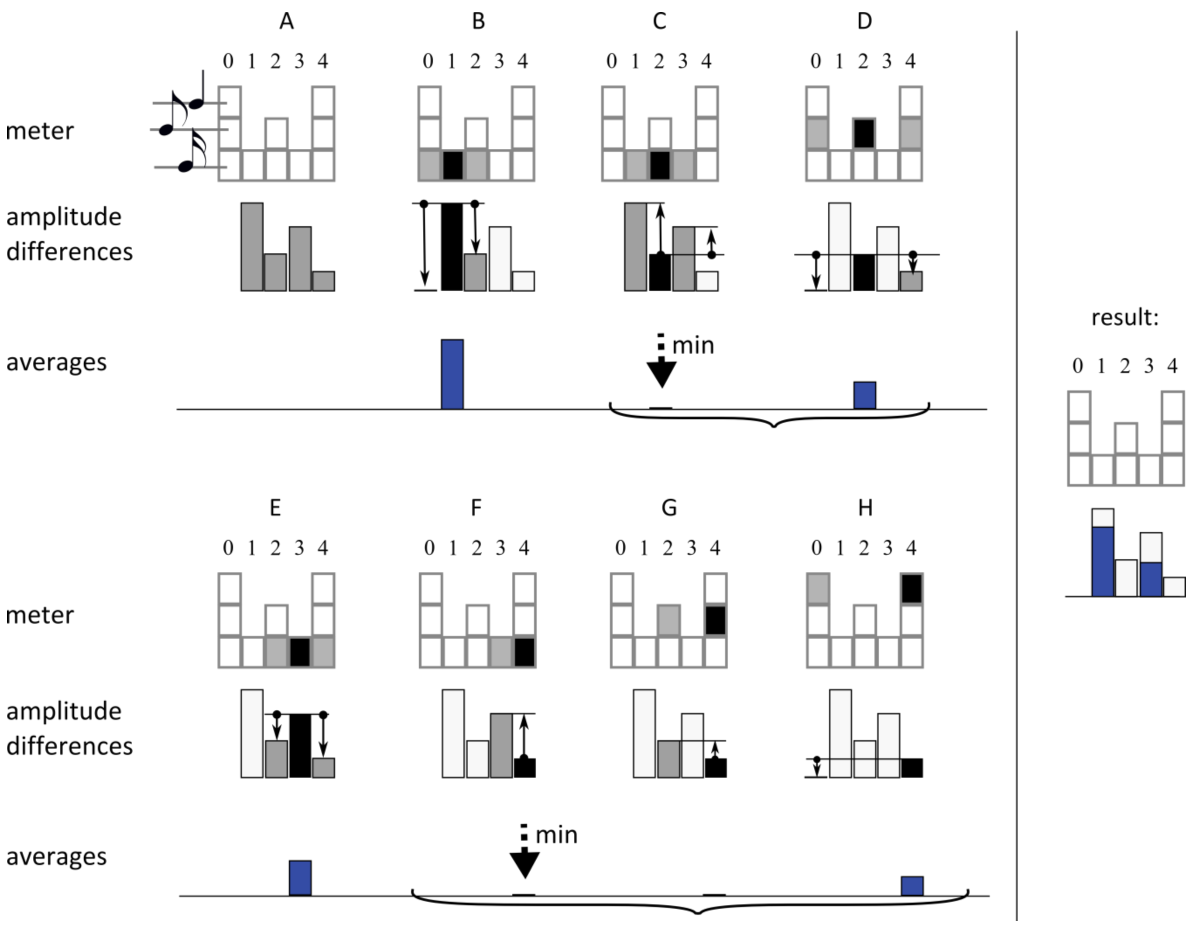


Figure 3-11: Calculation of the amplitude differences in the SG syncopation measure. Top rows: the metrical template. Middle rows: pattern of amplitudes. Bottom rows: syncopation scores as averages of amplitude differences. Amplitude differences are taken for each pulse at the metrical levels it belongs to: B) for pulse 1, C) and D) for pulse 2, E) for pulse 3 and F), G) and H) for pulse 4. The braces mark the amplitude differences that belong to the same pulse. For each set only the minimum value (marked but the dashed arrow) is kept. On the right, the result (in blue) together with the pattern and the metrical template (above) is shown.

First, each pulse is examined separately and amplitude differences are taken from the neighbour pulses. Pulse 0 of the example needs not to be examined since it contains no onset (amplitude = 0). In Figure 3-11 B, the amplitude differences for pulse 1 are shown. Pulse 1 belongs to the fastest metrical level so that only one pair of differences needs to be calculated and averaged. In Figure 3-11 C and D, the amplitude differences for pulse 2 are shown. Pulse 2 belongs to the 8th note level, so that two pairs of amplitude differences need to be considered: 1) from the previous and following 16th notes (C: pulses 1 and 3) and 2) from the previous and following quarter notes (D: pulses 0 and 4). In C, the amplitude differences are negative, since the amplitude of pulse 2 is lower than the one in pulses 1 and 3 and, therefore, they are set equal to 0. The average of each pair is calculated and from that average the minimum value is kept, which in this case is 0. In E, the average amplitude difference for pulse 3 is calculated. In F, G and H the pair of differences for pulse 4 is shown. Pulse 4 belongs to the slowest metrical level and therefore three amplitude differences need to be calculated, one for each of the faster metrical levels. Since, pulse 4 is the last pulse and the pattern of the example is not considered to be a loop, no difference from a following pulse can be taken.

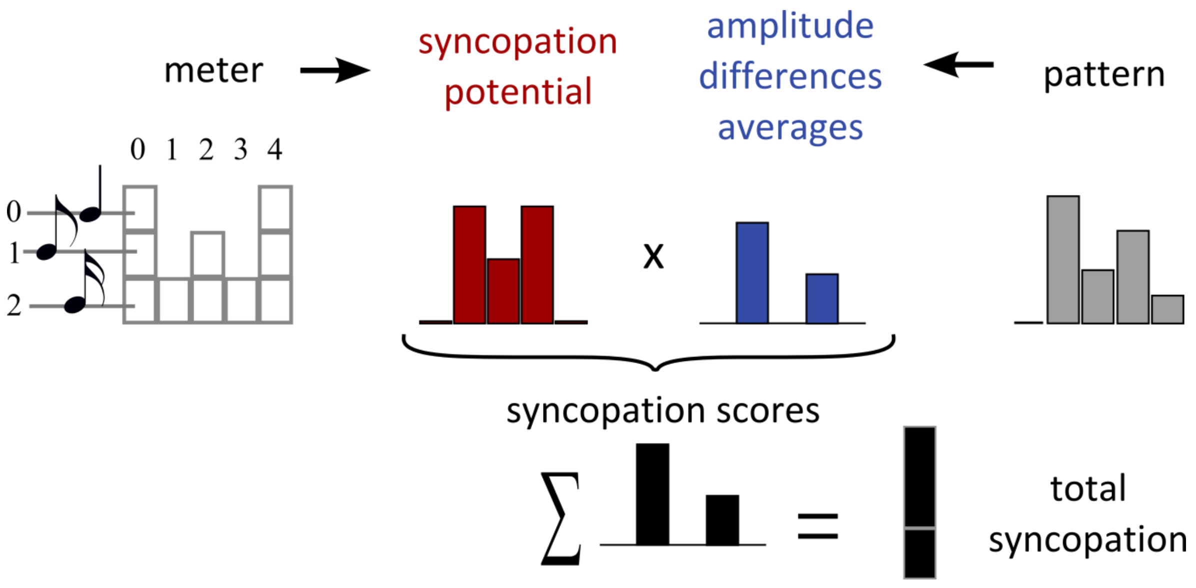


Figure 3-12: Overview of the SG syncopation algorithm. The metrical template is converted to syncopation potential for each pulse. Amplitude differences are calculated from the pattern of amplitudes. Multiplying the syncopation potential by the amplitude differences gives the syncopation score for each pulse. Summing the scores of all pulses yields the total syncopation for the pattern

The algorithm was originally implemented in the programming language C as the Max/MSP[17] external [kin.offBeatDetector] (Sioros & Guedes, 2011c) that can be downloaded as part of the "kinetic toolbox" [18]. I additionally developed a Max/MSP application around the external in order to measure syncopation

[17]http:// www.cycling74.com (accessed: August 1, 2015)

[18] http://smc.inescporto.pt/kinetic (accessed: August 1, 2015)

in MIDI files. The application opens and reads MIDI files of performances, extracts all the necessary information (time signature, note positions and MIDI velocities) and formats it according to the requirements of the external.

In order to evaluate the SG syncopation measure, first, I measured the syncopation of six clave and bell rhythmic patterns, namely the Shiko, Son, Soukous, Rumba, Gahu and Bossa-Nova (Sioros & Guedes, 2011c). These patterns are all five-note patterns with 4/4 time signature and are some of the most frequently used in the African, Cuban and Brazilian music.

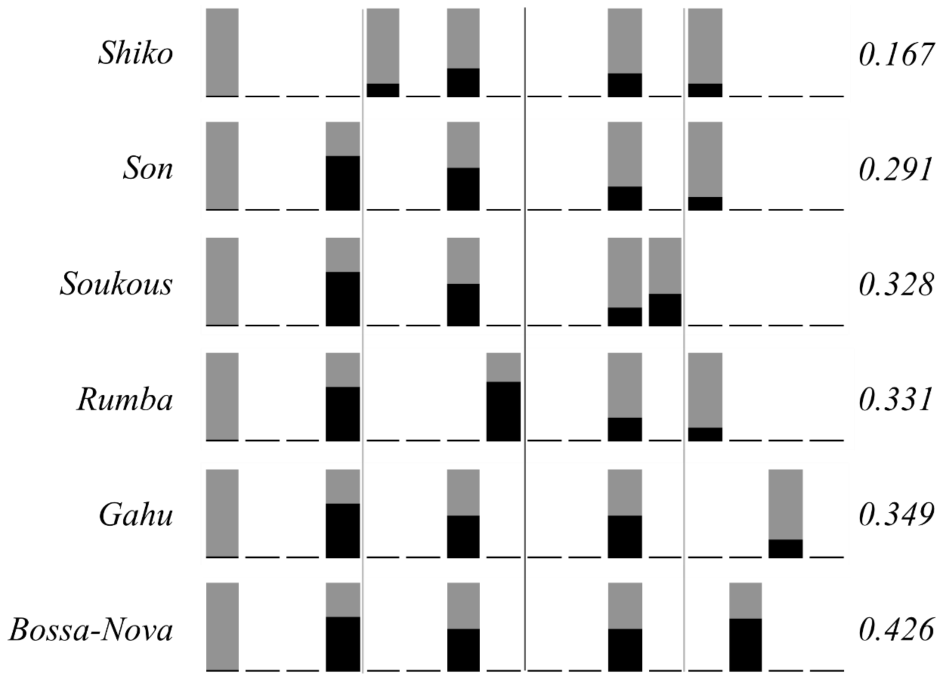


Figure 3-13: The syncopation of the six fundamental 4/4 clave and bell patterns is measured. Grey bars represent the onsets. Black bars notate the syncopation score corresponding to each onset. Next to each pattern, the total syncopation is given.

In Figure 3-13, the six patterns are presented together with their syncopation scores. Dynamic accents were not considered; all pulses have either maximum amplitude or are completely silent. The black bars represent the relative contribution of each pulse to the total syncopation score. The scores obtained by the calculations seem to agree with our experience that Shiko is the easiest pattern, Rumba is of medium complexity and Gahu and Bossa-Nova are amongst the most difficult to perform. The order from simple to complex is also in agreement with the cognitive complexity measure proposed by J. Pressing (see Pressing, 1997; T. Toussaint, 2002).

In our study on how to drive an interactive music system based on syncopation measurements (Sioros, Holzapfel, & Guedes, 2012), we evaluated the SG algorithm by comparing it to the more frequently used algorithm proposed by Longuet-Higgins and Lee (lhl) (1984) and that has already been tested and compared against human judgments (Fitch & Rosenfeld, 2007; Gómez et al., 2007). The comparison between the SG and the LHL measures showed that they are in good agreement.

Finally, an evaluation of several syncopation measures, including the SG measure, against musicians' judgments was recently presented in the doctoral thesis of Chunyang Song (2014, Chapter 5). In this study, Song asked musicians to rate the syncopation of 111 rhythms constructed by combining binary (2 pules long) and ternary (3 pulses long) patterns. They then measured the syncopation using the measures presented in this chapter with the exception of the extended LHL. In the calculations, the combination of binary patterns was considered as 4/4 meter rhythms while the combination of ternary patterns was considered as 6/8 meter rhythms. The combination of binary and ternary rhythms were considered polyrhythms. Only the Off-beatness, the WNDB and the Keith's measures can be applied to patterns combining two time signatures so that polyrhythms were used only in the evaluation of these measures.

The results show that the SG measure performed well. In the 4/4 rhythms, the Spearman-rank correlation coefficient for the SG measure was $r = 0.88$ ($p < 0.001$) and it was outperformed by Pressing's cognitive complexity ($r = 0.95$) and Toussaint's metrical complexity ($r = 0.92$). For the same rhythms the lhl measure performed very similarly to the SG with $r = 0.86$. The other three measures follow: Keith’s measure with r = 0.79, WNDB with r = 0.52 and Toussaint’s off-beatness with r = 0.36. In the 6/8 rhythms, the correlation coefficient for the SG measure was r = 0.73 ($p < 0.001$) and it was only outperformed by Pressing’s cognitive complexity (r = 0.76). The rest of the measures follow: lhl with r=0.68, Toussaint’s metrical complexity with r = 0.67, WNDB with r = 0.47 and off-beatness with r =0.17.

3.9 Discussion

In this chapter, I presented the most important operational definitions of syncopation and its measures. In most of them, a metrical hierarchy that consists of strong and weak beats is needed. In some definitions, like the off-beatness and the weighted note-to-beat distance, syncopation is defined against a single metrical level, the beat level, and slower metrical levels are essentially ignored.

The operational definitions of syncopation presented here make part of syncopation measures that attempt to define a scale that describes the intensity of the syncopation feeling. To this end, the syncopation measures define two different syncopation feelings: 1) the syncopation felt at a particular moment in a rhythm and that is attributed to a certain onset and 2) an overall syncopation feel that is attributed to the entire rhythmic pattern and that arises from aggregating the individual moments when syncopation is felt. The objective behind defining the later and the corresponding scale is to determine a way of comparing the syncopation of different rhythmic patterns and enable us to rank them from the least to the most syncopated.

The need for such a comparison comes about from the intended use of each syncopation measure. For example, the lhl algorithm was created to determine under which metrical context a rhythmic pattern is least syncopated. In this way, one could determine the meter of a rhythm. In other cases, like in the metrical complexity and cognitive complexity or the SG measure, as their names suggest, the goal is to rank a set of rhythmic patterns according to their complexity.

Complexity is easily understood as a quality of a rhythmic pattern and the subject of comparison. It is easy to imagine oneself comparing the complexity of two rhythms. Even if a formalized and objective measure of this complexity is not available, the feeling and cognitive effort one makes while listening to the rhythms can be the subject of the comparison.

In contrast, syncopation, which contributes to complexity, arises as a feeling in particular moments and is more difficult to be understood as an overall quality of a rhythm. The above syncopation definitions and formalized measures only give definitions for single instances of syncopation and avoid defining an overall syncopation feel. Instead, they sum the strength of individual syncopation, which, while useful in many cases, does not provide us with a natural scale of the syncopation feel. When one tries to compare the syncopation of two rhythms instead of their complexity, important questions about the nature of the comparison arise. The syncopation felt on a particular moment can be compared in terms of the intensity of the feeling to another syncopation moment, even when the two moments belong in different rhythms. The strength of an individual syncopation is related to how strongly it contradicts our metrical expectations at this particular moment. Nevertheless, an overall syncopation feeling is not clear how it arises. After all, the momentary syncopations by definition do not challenge the metrical perception and therefore one cannot talk about an overall contradiction of the meter by a certain syncopated rhythmic pattern and even less so about a comparison of such a contradiction between two rhythms. The nature of an overall syncopation feel is more obscure when one examines longer music passages instead of repeating patterns.

Although a definition of an overall syncopation feeling might be elusive, the fact that syncopation contributes to the complexity of a rhythm, at least when restricted to a specific musical context, is largely accepted. This is reflected in the way the terms syncopation and complexity are fused into each other in several of the above measures: in some measures, comparing the syncopation is equivalent to comparing the complexity. In addition, human judgment and performance studies focus on the complexity of patterns rather than their syncopation even when the main contribution to the complexity is syncopation (Essens, 1995; Dirk-Jan Povel & Essens, 1985; Shmulevich & Povel, 2000). When all other factors that contribute to complexity are kept constant, the comparison based on syncopation becomes meaningful. For example, the lhl syncopation measure is intended to determine which metrical hierarchy fits best a certain rhythm. To this end, the comparison of an overall syncopation feel of the various interpretations of the same rhythmic pattern under different metrical contexts is meaningful. However, care is needed as one moves away from this use and compares different rhythmic patterns in different meters. Similarly, various versions of the same melody can be more or less syncopated; a comparison to a different melody might not be always meaningful.

The studies that compare syncopation between rhythms are restricted to repeating patterns with the same number of onsets and pulses (Gómez et al., 2007; T. Toussaint, 2002; Toussaint, 2003)[19]. In such cases, summing the individual syncopation scores gives a meaningful measure of how many and how strong syncopations each pattern contains in comparison to another pattern. When patterns with different number of onsets or different number of pulses need to be compared, the subject of the comparison is less obvious. For instance, imagine the comparison between the two syncopated patterns of Figure 3-14, where the first has two onsets and the second has five. The two patterns have both two individual syncopations. Let us also assume that the metrical context is imposed externally in both cases by a metronome so that the metrical hierarchy is the same for both patterns. The comparison about which pattern is more syncopated is difficult to make.

[19] With the exception of Chunyang Song's Doctoral Thesis (2014).

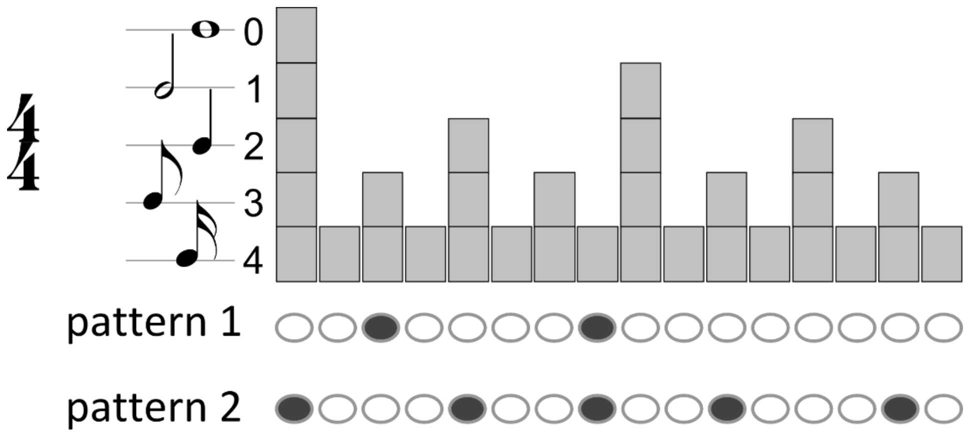


Figure 3-14: Comparison of the overall syncopation between two rhythms with different number of onsets.

Measure	**Pattern 1**	**Pattern 2**
LHL	1 + 3 = **4**	1 + 3 = **4**
Metrical complexity	9 – (2+1) = **6**	17 – (5+3+1+2+2) = **4**
Cognitive complexity	**11.25**	**10.875**
Off-beatness	**1**	**1**
WNBD	2/.5+1/.25 = **8**	2/.25+2/.5 =**12**
Keith	3+3 = **6**	3+3 = **6**
SG	0.55+0.72 =**1.27**	0.26+0.72+0.55+0.24 =**1.77**

Table 3-3: The syncopation scores of the two patterns of Figure 3-14.

In Table 3-3, the syncopation calculated by the different measures for the two patterns is shown. Looking at the results, one can see that there is no consensus about what the result of the comparison should be. The lhl, off-beatness and Keith's measure show the patterns as equally syncopated. Metrical complexity and cognitive complexity show that the first pattern with the fewer onsets is more syncopated. Finally, the WNDB and the SG measure show the second pattern as more syncopated. The disagreement in the results derives in part from the different normalizations to the number of onsets. The lhl, off-beatness and Keith's measures do not normalize their sums. Metrical complexity uses the maximum possible score for the specific number of onsets as the normalization. The WNDB measure is affected by the density of onsets indirectly through the weighting factor which tends to be 2 for IOIs of similar duration to the beat duration and 1 for longer or shorter IOIs. Finally, the SG does not normalize the result, while it detects more syncopating onsets due to the way it defines syncopation.

One can argue in favor of all three different results. First, according to the definition of the feeling of syncopation the two individual syncopations are equivalent between the two rhythms. Therefore, the two patterns can be considered as similarly and equally syncopated. Second, the second pattern has three additional onsets that do not contradict the meter. If these onsets were placed in off beat positions the pattern would be more syncopated. If the feeling of syncopation is relative, then the second pattern syncopates only partially and therefore is less syncopated than the fully syncopated first pattern. Another way of looking at this second argument is that the first pattern has only onsets that contradict the imposed meter while the second pattern has three onsets that reinforce it. The third argument is relevant to the positions of the onsets and the “density” of the syncopation. In the first pattern the two syncopating onsets are further apart resulting in a weaker overall syncopation feeling. In the second pattern, the two syncopations are close together—one after the other at a distance less than the duration of a beat—creating in this way more tension that is then resolved on the downbeat.

The SG syncopation measure is part of a sorting calculation that explicitly relates the density of events to the complexity of the given pattern. I developed the SG algorithm in order to sort a bunch of MIDI loops in order of their complexity as part of the kin.recombinator software application (Sioros & Guedes, 2011c). In the context of the software, we defined the complexity of a rhythmic pattern as a linear combination of the degree of syncopation and the density of events[20]. Here I presented the syncopation algorithm alone without the calculation of the complexity. The SG syncopation algorithm alone does not consider the density of events and does not normalize the amount of syncopation to the number of

[20] The density of events was replaced in the final algorithm with the sum of the MIDI velocities so that quiet notes are naturally ignored.

notes. However, one should keep in mind the intended use of the SG measure and should include some other suitable normalization of the result when used outside of its original scope.

The above examples are illustrative of the difficulty in determining the nature and subject of comparison of patterns based on the amount or degree of syncopation. Depending on the patterns and music context at hand, one could compare two music passages based on different qualities that relate to their syncopation but that might not be a simple measure of their degree of syncopation, such as complexity, rhythmic tension or strength of individual syncopations. In addition, such quantities might not have a linear relation with the number of syncopations in the pattern. For instance, a rhythmic pattern where all onsets are equidistant but in off beat positions is a rather simple pattern although it is considered by all measures as highly syncopated.

In the above discussion, the notion of density of syncopation was introduced. Such a quantity can be useful in the analysis of music as it can reveal longer duration patterns of rhythmic tension and resolution. It can be defined as the syncopation per unit time. The syncopation can be calculated as in the above measures by summing the individual syncopations' strength in a certain time window. The duration of the window can vary depending on the use. A sensible duration for the calculation would be between half a bar and 2 bars.

In (Sioros et al., 2012) we present a couple of examples of using the density of syncopation in order to obtain an overview of an entire song. The study was part of the development of an interactive music system that uses syncopation to mediate the interaction between a musician performing live and an automatic rhythm generator. Syncopation in this system is measured in single bar windows. While developing the system, we measured the syncopation in MIDI files of different music genres instead of in live streams. We used the same principle of measuring syncopation in single bars separately for each instrument. We obtained in this way an overview of how syncopation is distributed in the duration of each song and between the different instruments. In Figure 3-15, one of the published examples is shown.

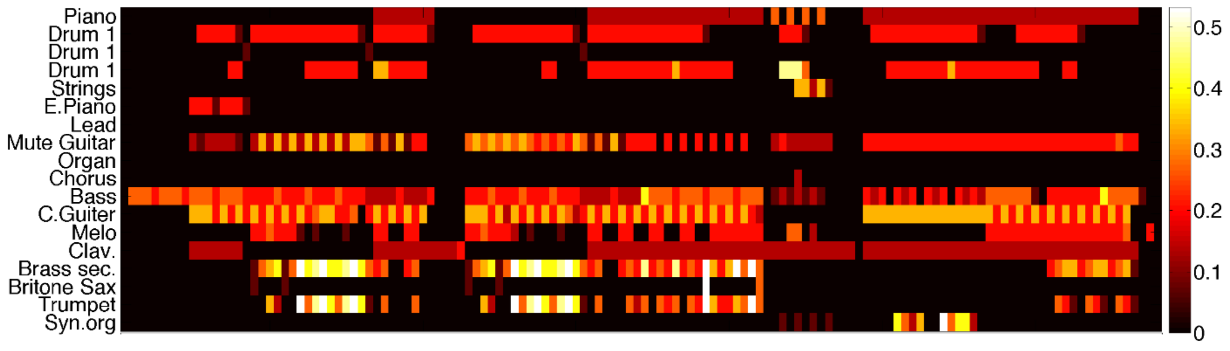


Figure 3-15: Syncopation scores for song No 22 of the RWC collection[21] (Genre: Funk): "Get on up and dance". The horizontal axis marks the bar number. The syncopation score for each bar is shown according to the color scale on the right. The LHL syncopation measure was used in the calculations.

[21] The RWC Music Genre dataset: http://staff.aist.go.jp/m.goto/RWC-MDB/ (accessed: December 5, 2012)

4 Syncopation Transformations

In this chapter I will present a set of formalized generic transformations that can analyze, generate and manipulate the syncopation in rhythmic patterns. I begin with syncopation transformations for *binary patterns*, i.e. in symbolic representations of rhythmic patterns containing only the quantized positions of the onsets[22]. The transformations are derived by the definition of syncopation as a cognitive mechanism related to metrical expectations (Huron, 2006). The metrical template described in chapter 2 is used to describe those expectations and in this chapter it is adjusted to meet the particular needs of the syncopation transformations.

The transformations are formalized with the aid of the *transformation vector*—a consistent way of notating the transformations. I introduce the concept of the *syncopation tree* as a way of organizing and interconnecting patterns. A syncopation tree is essentially a hierarchical lattice of patterns, where each pattern is connected to other patterns through a series of simple transformations. Patterns belonging in the same tree originate from the same *root*, i.e. they originate from the same non-syncopating pattern. The tree structure can be useful in developing models for clustering rhythms together, in defining measures of rhythmic similarity and distance, as well as in generative algorithms to produce rhythmic variation. Then, I generalize the binary syncopation transformations to a method for transforming rhythmic patterns that include dynamic accents, i.e. a sequence of quantized onsets each with its own amplitude value. Such accents may be the choice of the composer or they might result from a performance and often evoke the feeling of syncopation. The generalized transformations are based on the same principles of the binary ones in order to manipulate and generate syncopation that comes

[22] In a binary representation, each position of a metrical grid of equal intervals, e.g. the sixteenth-note quantization grid, can take one of two values: either it contains an onset (1) or not (0). A rhythmic pattern is then reduced to a sequence of 0s and 1s.

about from dynamic accents. Using the same concept of the transformation vector to describe syncopation both in binary patterns and from accents, the binary transformations become a special case of the generalized ones.

An important requirement, which, in many aspects, shaped the way the transformations are formalized, is that of *reversibility*: any elementary[23] syncopation transformation should always be reversible through a uniquely defined (reverse) transformation. Thus, not all possible transformations are allowed; they are limited to those that fulfill the requirement and can be reversed in a unique way. The reversibility constraint comes about directly from the definition of syncopation. As I will describe in section 4.2, the transformations are first defined as a de-syncopation method. The syncopation then is defined as the reverse transformation. Reversibility is therefore a core property of the transformations.

I describe how the metrical template of section 2.3 should be adapted for modeling syncopation in section 4.1. In section 4.2, I describe the syncopation transformations for binary patterns and, in 4.3, I introduce the concept of a syncopation tree. In section 4.4, I give two examples of computer algorithms that I developed based on the syncopation transformations. In section 4.5, I evaluate the transformations by measuring the syncopation in a collection of patterns before and after they were transformed. Finally, in section 4.6, I generalize the transformations to manipulate dynamic accents.

4.1 Metrical Template for Syncopation

The metrical template is constructed automatically for each meter and tempo. A description of the automatic construction of the metrical template was given in chapter 2. Here, I present the necessary adaptations of the metrical template in order to be used in the syncopation transformations. These

[23] By elementary transformation, I mean a transformation that cannot be analyzed further to a series of steps.

adaptations are an important and integral part of the syncopation transformations as they affect core properties of the transformations and the syncopation tree.

The metrical template described in chapter 2 consists of pulses with different metrical strength values that represent the alternating strong and weak beats commonly found in a musical meter (see Figure 2-3 for an example). Similarly to the metrical structure used by Longuet-Higgins and Lee in their syncopation definition (Longuet-Higgins & Lee, 1984), each pulse initiates a metrical level and its metrical strength is proportional to that level. The metrical hierarchy can be thought of as a superposition of layers of pulses with different periods, in which the period of one layer is an integer multiple of all faster ones (Parncutt, 1994; Yeston, 1976). The pulses constitute a metrical grid that quantizes the time positions of the onsets of the events.

The duration of each metrical subdivision depends on the tempo. As I discussed in chapter 2, the lower threshold for the duration of a metrical subdivision has been estimated in several studies to be roughly around 100ms (London, 2012, p. 29; Parncutt, 1994; Repp, 2006). The fastest metrical subdivision that I included in the metrical template is the fastest subdivision above that threshold.

An upper threshold for the duration of the metrical levels needs also to be determined. Before determining the upper threshold, we need a definition for the duration of syncopation and examine its relation to the metrical levels. The definition of syncopation, as formulated by Longuet-Higgins and Lee (1984) and Huron (2006, p. 295), and also adopted here, attributes syncopation in the pair of an onset on a weak pulse with the following silent strong pulse. One can, therefore, talk about and define the duration of the syncopation as the duration between those two pulses. The duration of the syncopation depends, on one hand, on the metrical levels of the corresponding pulses, and on the other hand, on the tempo. One question that arises is: how does the duration of the syncopation (and tempo) affect the feeling of syncopation?

The effect of the duration of syncopation on the feeling of syncopation becomes apparent when one tries to de-syncopate a rhythmic pattern. If one tries to de-syncopate the patterns of Figure 4-1, one will follow different approaches for pattern *A* and *B* (assuming that both are performed at the same tempo, for example at 100bpm). In pattern *A*, the tied eighth note should clearly be moved to the following quarter note. When it comes to pattern *B*, the note durations are twice as long, and the tied note is now a quarter note that falls on the beat. Therefore, intuitively, it does not need to be de-syncopated. However, the two cases are identical with respect to the definition of syncopation and the metrical template. The difference between the two is the duration of the metrical levels and whether they are faster than the beat level (or tactus) or not.

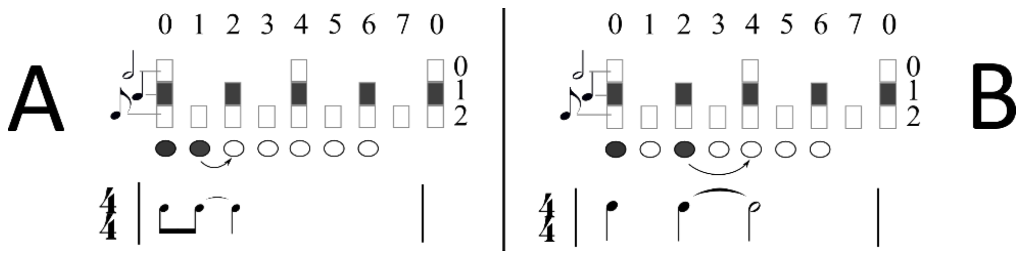


Figure 4-1. Syncopation at slow metrical levels. A: a pattern syncopating at the eighth-note metrical level. B: The same pattern at half speed. Above the two patterns the corresponding metrical template is shown. The beat level is noted with black rectangles.

But the listener does not have the score or the notated meter in his mind. He can only tell the difference based on the duration; *A* is performed twice as fast as *B*. For example, if *B* is performed at 100BPM the duration of the quarter note will be 600ms. Performing the same pattern twice as fast, i.e accelerating the quarter note duration to 300ms, is equivalent to converting the quarter note to eighth notes leading to pattern (*A*). In general, the perceived meter and beat will depend on the musical context (London, Himberg, & Cross, 2006; Moelants & Mckinney, 2004). Nevertheless, performing a pattern twice as fast will not result in the halving of the beat duration. The metrical salience of each level depends predominantly by its period, with a peak salience in the region between 500ms – 1s (Duke, 1989; London, 2012, Chapter 2; Parncutt, 1994). With this in mind, we can assume that, more often than not, the perceived beat—the most salient metrical level—will be the one with the duration of 600ms in both performances. The second note of pattern *A* falls on the faster metrical level of 300ms which is significantly less salient and therefore perceived as offbeat and as strongly syncopated. On the contrary, in pattern *B*, it falls on the beat so that no syncopation is felt.

This aspect of syncopation and its relation to pulse salience has not been thoroughly studied. However, one can assume that syncopation involving only slower metrical levels is not felt as strong (Cooper & Meyer., 1960, p. 100). The above example illustrates the more general assumption that syncopation with duration equal or longer than that of the most salient level is felt significantly less strong. The LHL syncopation measure (Longuet-Higgins & Lee, 1984) takes this effect into account indirectly, by giving to the syncopation that involves adjacent metrical levels a relatively small weight. Other syncopation measures, such as the weight-note-to-beat distance (WNBD) (Gómez et al., 2005), relate syncopation directly to the beat level, ignoring all slower metrical levels.

Chunyang Song (2014, Chapter 6) explored the effect of tempo on syncopation in her PhD Thesis. In her listening experiment, she asked musicians to rate the syncopation of different rhythmic patterns presented in a large range of tempi. The main finding of this study is that syncopation is felt stronger at moderate tempi around 120 bpm and is very weak for very fast or very slow tempi. Although this result does not directly show a relation between syncopation and pulse salience, it provides indirect evidence of it.

In order to take the above effect of tempo into consideration in the construction of the template, I employ a similar approach to the WNBD by essentially “chopping off” the slower metrical levels. I chose the level that falls in the range between 500ms and 1s as the slowest metrical level represented in the metrical template. For example, in the case of a 4/4 meter at 160bpm (quarter-note = 375ms), the metrical template of Figure 2-3 will become as in Figure 4-2. At this tempo, only 3 metrical levels survive with corresponding durations of 750ms (0), 375ms (1) and 187.5ms (2).

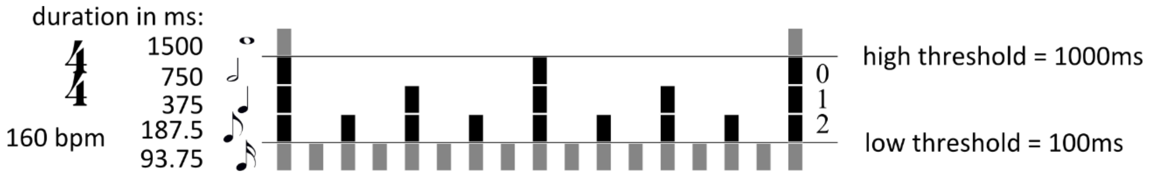


Figure 4-2. Example of a metrical template in which very fast metrical levels (below 100ms duration) and very slow ones (above 1s duration) are disregarded.

The above template can easily be adjusted to non-isochronous (NI) meters. NI meters are not subdivided into beats of equal duration (London, 2012) but can be represented as an additive process of pulses. For example, the 7/8 meter can be represented as 3+2+2, where the first beat has a duration of 3 pulses and the following two beats have a duration of 2 pulses. Although it is out of the scope of this chapter to offer an algorithm for the automatic generation of such templates, I describe here the main principles that can be used in such algorithms.

As described in section 2.3, the first step to the generation of a metrical template is the prime factorization of the number of pulses. In the case that the numerator of the time signature is a prime number such as the 7/8, the bar is immediately subdivided by that prime number, e.g. 7 pulses of equal duration. This means that, assuming a tempo in which the eighth note equals 250ms, the metrical levels are the very slow whole bar (1750ms) and immediately after the very fast eighth-note level (250ms). In other words, there is no metrical level with a period close to the region that the pulse salience has its peak (500ms – 1000ms). Since I require for the syncopation definition and transformation that the slowest metrical level in the template is the one with duration in that region of high pulse salience, such a level needs to be constructed for any meter.

The following solution generates a metrical level that is not isochronous (NI meter), i.e. the pulses at this level have different durations. The method consists in finding the smaller number of different durations (therefore construct an NI meter) that: 1) their sum equals the duration of the bar and 2) each one of the durations falls in the high salience region. In the above example of the 7/8, the 7 pulses can be decomposed into 3+2+2, where the long beat has duration of 750ms and the short beat of 500ms. A different solution that fulfills the requirements is 7 = 4 + 3 with durations of 1000ms and 750ms respectively[24]. The order in which the beat durations appear in the meter depends largely on the musical context. However, the default choice could be in descending order so that the long intervals come first (3+2+2 instead of 2+3+2 or 2+2+3). This choice can be justified by the fact that a short-long interval, in general, creates an accent on the second tone (Dirk-Jan Povel & Okkerman, 1981) which results in naturally stressing the downbeat.

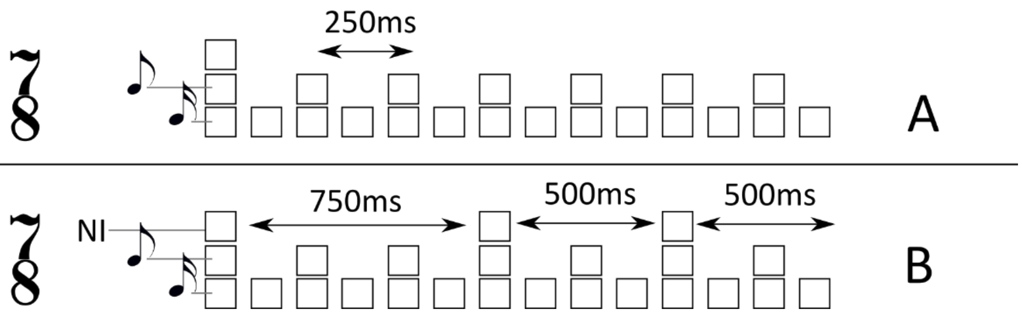


Figure 4-3: An example of a NI meter: the 7/8 meter at 240 bpm (1 beat = eighth-note). A: template automatically constructed according to the method of chapter 2. B: The 7 x 8th notes decomposed into 3+2+2.

[24] If one needs to uniquely define the pulse durations, the second requirement can be modified so that each pulse at the beat level must have the shortest duration possible between 500ms and 1000ms.

London (2012) considers an extensive list of possible NI meters and their beat durations. Although the above method for constructing a NI metrical template does not cover all possible NI meters, it provides with a solution for the cases where a tactus with duration between 500ms and 1s cannot be found by simple prime factorization of the length of the bar. In such cases, it breaks automatically the metrical level with a long duration, usually the entire bar duration, down to a set of shorter beats in an additive manner.

4.2 TRANSFORMING BINARY PATTERNS

According to the definition of syncopation of Longuet-Higgins and Lee (1984) and Huron (2006, p. 295), if a syncopating event in a weak metrical position is shifted to the following stronger position the syncopation would be eliminated. The event at the weak position can be thought of as belonging to the strong position but been anticipated at an earlier position (Temperley, 1999). Therefore, one can imagine the inverse shift as a way of generating syncopation at a strong metrical position, i.e. by shifting an event to an earlier, weaker pulse.

The transformations described in this section are such simple shifts of onsets in a binary pattern (Figure 4-4). The following formalization of the transformations consists of defining the conditions under which such shifts remove the syncopation or generate syncopation in a given binary rhythmic pattern and a consistent way of representing such transformations.

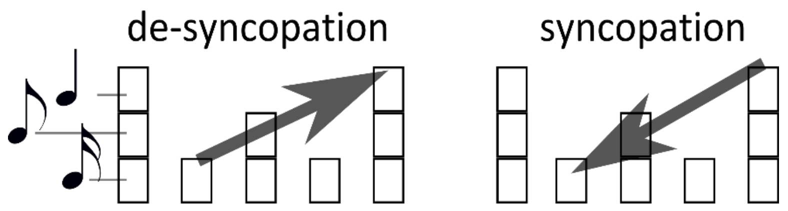


Figure 4-4. The de-syncopation transformation shift events forward to slower metrical levels. The syncopation transformation shifts events backwards to faster metrical levels.

In order to remove syncopation, onsets are shifted forward to stronger metrical positions (Figure 4-4, left). Pulses that belong to slower metrical levels are considered to be stronger, so that the de-syncopation process moves events from fast metrical levels to slower ones. In order to generate syncopation, onsets are anticipated from their original strong metrical positions to weaker positions (Figure 4-4, right). The syncopation process, opposite to the de-syncopation process, “pulls” the events to the faster metrical levels. A detailed description of the syncopation transformation and its inverse—the de-syncopation transformation—follows in sections 4.2.1 and 4.2.2. In section 4.5, I make a preliminary evaluation of the binary transformations.

4.2.1 Generating Syncopation

The syncopation transformation takes a binary pattern and generates syncopation by anticipating the events found in strong metrical positions. Onsets in pulses that belong to strong metrical positions (slow metrical levels, low metrical indexes) are shifted to preceding pulses belonging to weaker metrical positions (faster metrical levels, higher level indexes).

A strong pulse might be preceded by more than one weak pulses, e.g. a pulse that belongs to the quarter-note level is preceded by a pulse at the eighth-note level and another one at the sixteenth-note level, both belonging in faster metrical levels than the initial one (Figure 4-5). Thus, when syncopating, there might be more than one pulse to shift an onset to. In other words, it is possible to syncopate in more than one way at a certain metrical position.

I define as the *type* of syncopation the value of the difference of the metrical levels of the two pulses: the pulse to which the onset belongs originally and the one to which is shifted. Which pulse is chosen for a syncopation transformation is determined by the type chosen for the transformation.

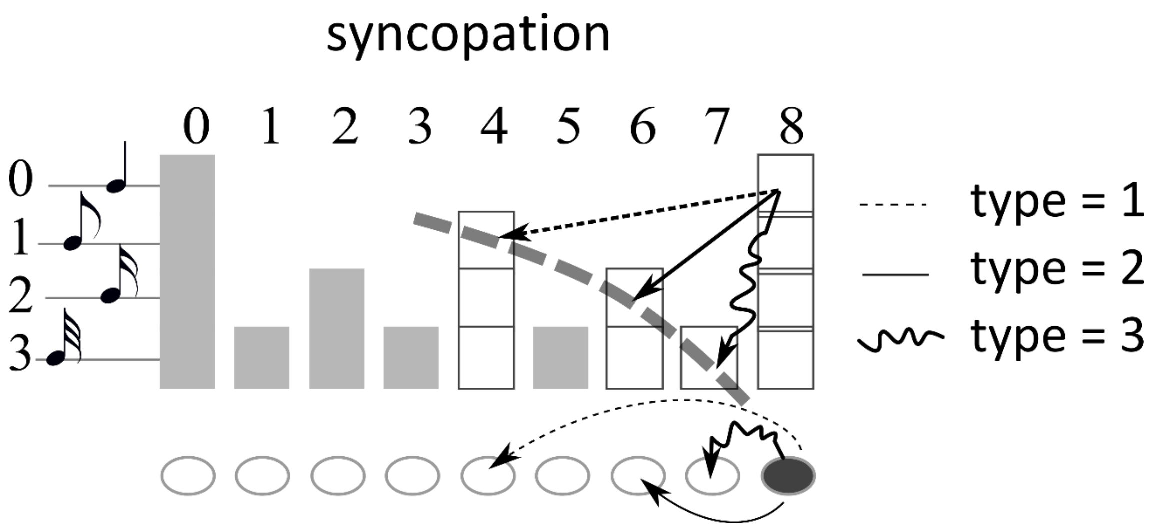


Figure 4-5. An example of the types of syncopation transformations available for an onset at a specific pulse (pulse 8). White squares represent the preceding pulses of faster metrical levels that are available for syncopating pulse 8. The remaining pulses have been greyed out. The corresponding de-syncopation transformation is obtained by reversing the direction of the arrows.

For example, if the type is set to 1 (the difference of the metrical indexes = 1), a quarter note is shifted to the preceding eight note since the eight-note level is one metrical subdivision faster than the quarter-note level; while, if the type is set to 2, the same quarter note is shifted at the preceding sixteenth note, and for type 3 to the thirty-second note (Figure 4-5). In general, the larger the type value (the difference of the metrical indexes), the shorter it is the duration of the generated syncopation. This is a direct consequence of the alternating character of strong and weak pulses of the metrical template. The number of syncopation types that is available for each pulse depends on how many faster metrical levels exist in the template.

Alternatively, I could have encoded the transformation as the pair of pulses, the initial pulse of the onset and the pulse it is shifted to. This way of encoding correctly describes the particular transformation. However, as it will become apparent in the following sections, using the level differences is a more general representation that reveals the relation of syncopation to the metrical hierarchy.

The following simplified pseudocode integrates the above rules:

```
Syncopation (InPattern, Template, OutPattern, Vector{pos, type})
  OutPattern == InPattern
  TargetLevel = Template[pos] + type
  precedingPos = Find_Preceding_Pulse(pos, TargetLevel)
  If (precedingPos >= 0)
    If ( (Type ==0 AND Template[pos] >0) OR Type >0)
      OutPattern[pos] = FALSE
      OutPattern[precedingPos] = TRUE
    End
  End
Return OutPattern
End

Find_Preceding_Pulse(startingPulse, TargetLevel)
P = startingPulse
LMIN = Template[p]
p--
CurrentML = Template [p]
  While (Template[p] > LMIN AND InPattern[p]==FALSE AND Template[p]!= TargetLevel)
    p--
  End
  If (InPattern[p]==FALSE AND Template[p]==TargetLevel)
    Return p
  Else Return -1
End
```

The above algorithm has been implemented in C++ code as part of the Max/MSP external [R.transformation] that is part of this thesis and it can be found in APENDIX A.

4.2.1.1 Transformation Vector

Each syncopation shift is described by two numbers: the pulse that originally carries the onset and the type of syncopation shift that the onset undergoes. The two numbers can be thought of as the "coordinates" of the syncopation in the metrical template, where one dimension corresponds to the pulse indexes and the other to the metrical levels. In that way, they form ***the transformation vector*** consisting of a "horizontal" coordinate, that is, the pulse index as the origin of the transformation, and a "vertical" shift, that is, the type value: {index, type}. In the example of Figure 4-5, the three vectors would be {8, 1} for the shift to pulse 4, {8, 2} for the shift to pulse 6 and {8, 3} for the shift to pulse 7.

4.2.2 Removing syncopation

The de-syncopation transformation is essentially the inverse of the syncopation transformation; it shifts an event from a weak metrical position to the following stronger one. In Figure 4-5, onsets that might be found in pulses 4, 6, or 7, can be de-syncopated by reversing the direction of the arrows. As an onset gets de-syncopated, it reveals the type of the syncopation transformation that it had previously undergone. The de-syncopation can be thought of as the analysis process yielding the type of syncopation that was previously generated by a syncopation transformation.

The above formulation of the definition of syncopation attributes the syncopation to the event being anticipated. However, the syncopation is only felt at the moment of the following silent pulse and not when the event is heard. As we tend to hear syncopated events rather than syncopating silent moments, the syncopation is attributed to the event retrospectively (Huron, 2006, p. 200) (Figure 4-6). The following alternative phrasing of the syncopation definition attributes syncopation to the silent pulse where the syncopation is actually felt: *syncopation is felt in the silent pulses that belong in strong metrical positions (slow levels, low indexes) and that are preceded by onsets in one of the immediately preceding weaker metrical positions*. Accordingly, it is not the events that should be de-syncopated but the silent pulses.

For example, in Figure 4-7, instead of talking about the de-syncopation of the onset in pulse 6, we could be talking about the de-syncopation of the silent pulse 8 where the syncopation is felt. I this way, shifting the onset from pulse 6 to pulse 8, yields the corresponding syncopation type and the transformation vector {8, 2}. In this way, the output vector of the de-syncopation is the same as the input vector that generates the syncopation when shifting the onset from pulse 8 to pulse 6.

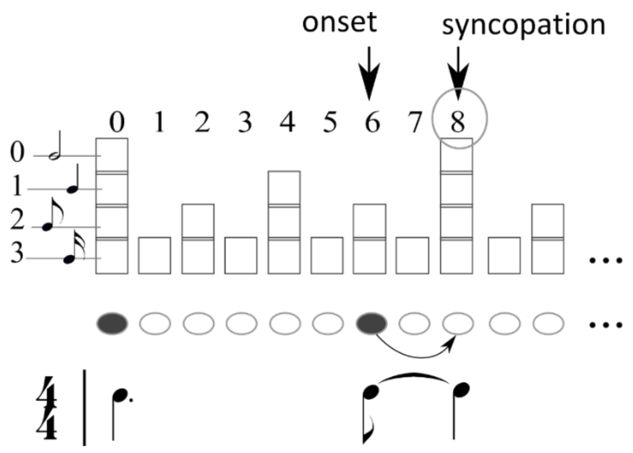


Figure 4-6: When the onset on pulse 6 is heard it is not felt yet as syncopated. The syncopation is only felt at the strong silent pulse that follows (8) and is attributed retrospectively to the onset.

Generally, de-syncopating silent pulses instead of the preceding onsets makes the two transformations reversible. In a syncopation transformation, a vector that defines the pulse and type needs to be input. When reversing the transformation by de-syncopating the silent pulse, the same vector is output. An event might need to be shifted more than once in order to be completely de-syncopated. For example, an event found in pulse 5 in Figure 4-6 needs to be shifted first to pulse 6 and then to pulse 8 yielding the corresponding syncopation transformations {6, 1} and {8, 1} expressed as vectors. By de-syncopating a pattern, we retrieve the syncopation transformations that the pattern has previously undergone.

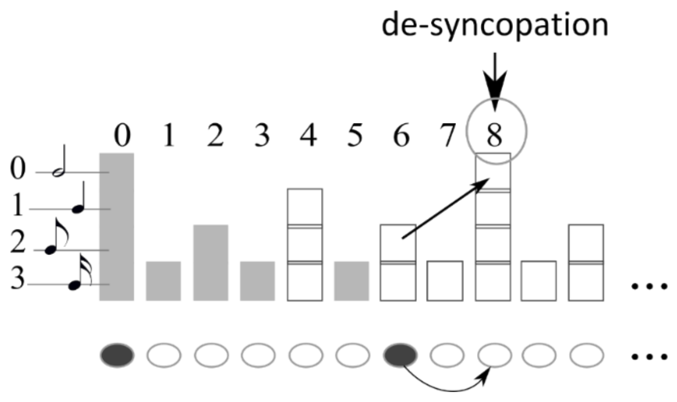


Figure 4-7: De-syncopation of the silent pulse 8 is performed by shifting the previous onset (pulse 6).

The following pseudocode integrates the above rules:

```
DeSyncopation (InPattern, Template, pos, OutPattern, OutVector)
  OutPattern = InPattern
  OutVector = {}
  if (InPattern[pos]==FALSE AND Template[p]<=max(Template))
    precedingOn = Find_Preceding_Onset(pos)
    If (precedingOn >= 0 )
            OutPattern[precedingOn] =FALSE
            OutPattern[pos] =TRUE
            OutVector = {pos, Template[pos]-Template[precedingOn] }
    End
  End
Return OutPattern, OutVector
End

Find_Preceding_Onset(p)
  LMIN = Template[p]
  p--
  CurrentML = Template[p]
  While (Template[p] > LMIN AND InPattern[pos]==FALSE)
      p--
      CurrentML = min(Template[p], CurrentML)
  End
  If (InPattern[p]==TRUE AND Template[p]== CurrentML AND Template[p]>LMIN)
    Return p
  Else Return -1
End
```

The above algorithm has been implemented in C++ code as part of the Max/MSP external [R.transformation] that is part of this thesis and can be found in APENDIX A.

4.2.3 Reversibility constraints and properties of the transformations

The following subsections present the implications of the two major constraints of the syncopation transformations:

1) Each step must be reversible through a single unique transformation;
2) A transformation should not alter the original order of onsets in the pattern. The onsets correspond to music events that ultimately have properties that make them distinct, such as dynamic accents or pitch; therefore their order must be preserved. This way the model does not

loses its generality and can be extended to include any type of accents, such as arising from timbre changes, or other properties of the events that can make them distinct.

4.2.3.1 Order of onsets

The de-syncopation transformation preserves the order of the onsets in a given pattern by nature but for the syncopation transformation the corresponding rule needs to be imposed. For example, in a pattern of three onsets, if one tries to syncopate by displacing the third onset, this onset cannot be shifted to a position before the second onset. Generally, an onset cannot be shifted if one or more onsets block its way. This rule ensures that the order of events in the pattern is preserved. In the example of Figure 4-5, we could not apply the syncopation transformation {8, 1} if an onset is found in any of the pulses 5, 6 or 7. The transformation would be forbidden.

The reversibility of the transformation is a deeper reason for preserving the order of onsets. If one could jump over an existing onset when syncopating, then the transformations could not have a 1-1 correspondence. Imagine an onset originally existing in pulse 4 of Figure 4-8 and apply the {4, 1} syncopation transformation (A: dashed arrow). If one would then de-syncopate pulse 4 (B), it would result in a different pattern (C) than the one we started with and the de-syncopation would return a different syncopation transformation—of type 2. Thus, reversing two onsets would result in non – reversible transformations.

In addition, onsets cannot be shifted to pulses that already carry an onset. If such a shift was allowed, the overlapping onsets would collapse to a single onset in a binary representation, so that reversing the transformation would always result in a pattern with fewer onsets. If such overlaps are desirable for certain events, then the onsets should be treated as belonging to different binary patterns. For example, different instruments in the same music piece are often considered as independent streams.

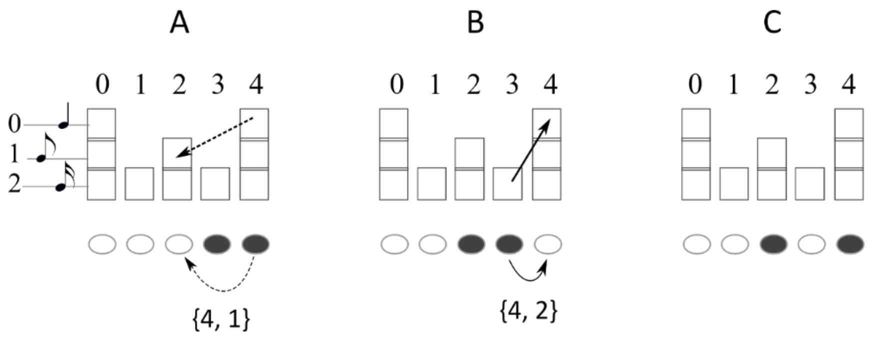


Figure 4-8. If the order of onsets was not preserved during the transformations, the 1-1 correspondence of the two transformations would be lost. A: Applying the forbidden transformation {4,1} to the original non-syncopated pattern jumping over the onset in pulse 3. B: De-syncopating the resulting pattern yields a different transformation {4,2}. C: The resulting de-syncopated pattern is different than the original.

4.2.3.2 Direction of shifts

The de-syncopation transformation guarantees that the resulting pattern will not syncopate. This is not true for other similar transformations. In some cases, one could de-syncopate offbeat events by shifting them to the preceding on-beat position instead of the following one. However, the two directions are not equivalent. A rhythmic pattern is always heard in the forward direction of time and, as a consequence, the operational definition of syncopation attributes the syncopation feel to the lack of an event in the following strong position. The forward shifting of events guarantees the resulting pattern will not syncopate, as the “empty” strong pulse that was causing previously the syncopation will now receive an onset. The reversibility constraint requires then that the syncopation transformation will have the opposite direction.

The above argument could be rephrased putting the emphasis on the syncopation. The syncopation transformations guarantee that syncopation will be felt on the pulse that is transformed. This is not true for other similar transformations that could in certain cases introduce syncopation but not in a systematic way. For example, one could generate syncopation by moving an onset from an on-beat position to a following weak pulse (instead of a preceding one). Such a displacement does not generally create syncopation but syncopation could be introduced under certain conditions: 1) if an onset existed in a preceding weak pulse and 2) if no onset existed in the following beat. In both cases the result depends on the positions of the neighbor onsets. Especially in the second case, even if syncopation was generated, it would be felt on the following beat and not on the pulse that the transformation was applied on. Moreover, two syncopations could be generated with one shift. Defining a transformation vector for such shifts would be impossible. The reverse transformation would also be impossible to define in a unique way.

The above arguments come as a direct consequence of the fact that syncopation is felt on a strong silent pulse and is retrospectively attributed to the preceding event (Huron, 2006, p. 260).

4.2.3.3 Type 0 transformations

Type 0 transformations are transformations that shift an onset to the preceding pulse that belongs to the same metrical level as the initial pulse. They are generally forbidden with the exception of ternary subdivisions of the beat, such as the subdivisions in the 6/8 meter as I will show in what follows. By nature, a syncopation transformation is supposed to shift an onset to a faster metrical level than the one it originally belonged to as a direct consequence of the definition of syncopation. The metrical template was adjusted by omitting the metrical levels slower than the beat based exactly on this definition, so that in effect no syncopation shifts can occur between the beat positions (Figure 4-9). For this reason, type 0 transformations are not allowed at the beat level. In faster metrical levels, the type 0 is restricted due to the reversibility constraint as can be seen in Figure 4-10.

A type 0 transformation is only allowed in ternary subdivisions of the beat as in Figure 4-11. In such metrical templates, a type 0 syncopation is needed. For example, in the template of Figure 4-11, a potential onset on pulse 12 cannot be shifted to pulse 8, although it belongs to one metrical level faster; the {12, 1} transformation would move the onset to pulse 10 instead. Such a transformation requires two steps: 1) shift the onset from pulse 12 to pulse 10 and 2) shift the onset from pulse 10 to pulse 8. The second step is a type 0 transformation and without it would be impossible for the onset to "reach" pulse 8. On the other hand, the onset on pulse 6 should not be allowed to be shifted to pulse 0, because such a shift would not in fact generate syncopation.

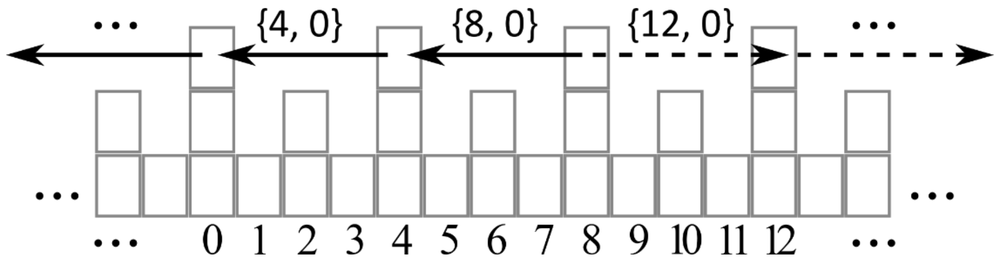


Figure 4-9: Type 0 transformations are not allowed at the slowest metrical level as a direct consequence of the definition of syncopation. Otherwise onsets would be moved freely backward without introducing syncopation (solid arrows) or forward without removing syncopation (dashed arrows).

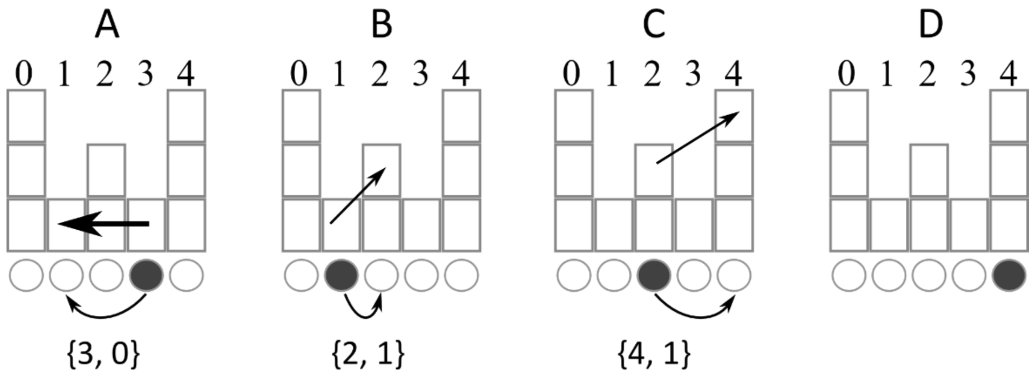


Figure 4-10: A: a forbidden type 0 transformation at a fast metrical level. B - D: de-syncopating the resulting pattern results in different patterns (C and D) than the original (A) and yields two different transformation vectors (B, C).

An example of type 0 transformation is discussed by Povel and Okkerman (1981) and is shown here in Figure 4-12. They argue that the rhythmic figure in the third bar—although it does not constitute a typical example of syncopation—it is "creating a special rhythmical tension" because of the ambivalence between the metrical accents and the accent caused by the repeating Short-Long IOIs (see section 2.5 for a discussion on accents). In contrast, in the previous bar, the reversed rhythmic figure of Long-Short

IOIs does not create any tension and is in agreement with the 6/8 meter. The syncopation transformations suggest—and I believe it is true—that this "special tension" is of the same nature as the more typical examples of syncopation with, perhaps, the difference that the type 0 syncopation is felt weaker.

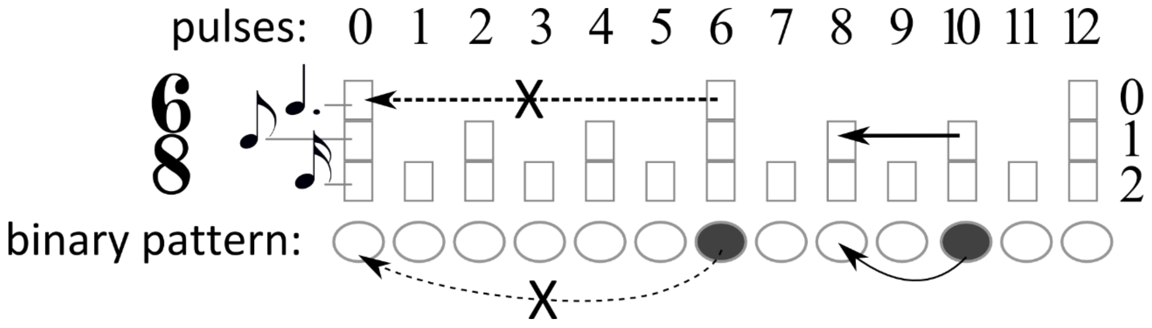


Figure 4-11: type 0 syncopation is only allowed in subdivisions of the beat. In this example, the {6, 0} transformation (dashed arrow) is forbidden while the {10, 0} (solid arrow) is allowed.

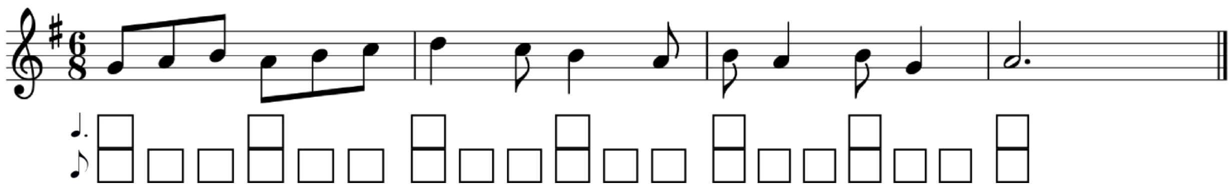


Figure 4-12: Example of a type 0 syncopation taken from (Dirk-Jan Povel & Okkerman, 1981)*. The metrical hierarchy is shown under the staff.*

4.2.3.4 The beat duration

An important property of the transformations comes about the structure of the metrical template of alternating strong and weak pulses and the prominent role given to the beat—the most salient of the metrical levels. As we saw, the transformations can shift events either *forward* to slower metrical levels or *backwards* to faster metrical levels. The beat level is the slowest metrical level included in the template, and at this level type 0 shifts are forbidden, so that each transformation has local character; it displaces events within the duration of a single beat. However, the duration of the beat is offset by a single pulse to what commonly is considered the duration of the beat (see Figure 4-13). The beat duration here ends ON the beat and includes all preceding pulses between the current and the previous beat. In contrast with the more common view that a strong pulse initiates the duration of the corresponding metrical level, under the lens of the syncopation transformations, a strong pulse is the end of a "motion" that begins at the preceding weaker pulses. Thus, any pattern longer than a beat can be considered a concatenation of shorter and independent—with regard to the syncopation transformations—single beat patterns. For that reason, several examples given in this thesis are given as patterns containing only a single beat.

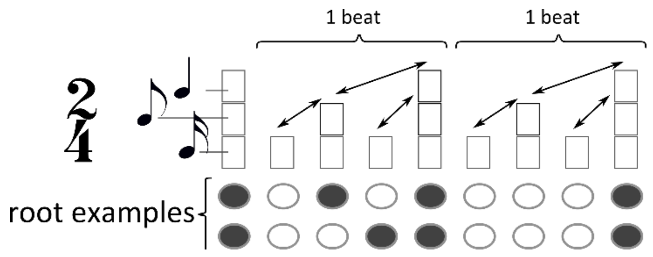


Figure 4-13. The syncopation shifts can only move onsets in the duration of a single beat.

4.2.3.5 Arrays of transformation vectors

Certain transformations comprise more than one step in order to be reversible. The different steps must be performed in a certain order. Each step is a simple transformation as the ones described in 4.2.1 and 4.2.2 and is represented by a vector, so that the compound transformation is represented by an *array of vectors*. In this section, I give two such examples.

The first example, example I, is shown in Figure 4-14. If we could de-syncopate pulse 4 in (A) by shifting the onset from pulse 1 directly to pulse 4, we would get as a result the {4,2} transformation vector. If, then, we reverse the transformation by applying the syncopation {4,2}, we would shift the onset to pulse 3 resulting in a different pattern (B). In order to maintain the reversibility, the pattern of Figure

4-14 needs to be de-syncopated in two steps (as in C). We need to de-syncopate pulse 2 first, yielding the vector {2, 1}, and then pulse 4, yielding {4,1}. Reversing the transformation consists in applying these vectors in the reverse order. The *array of transformation vectors* that results by the compound de-syncopation is [{2,1} , {4,1}].

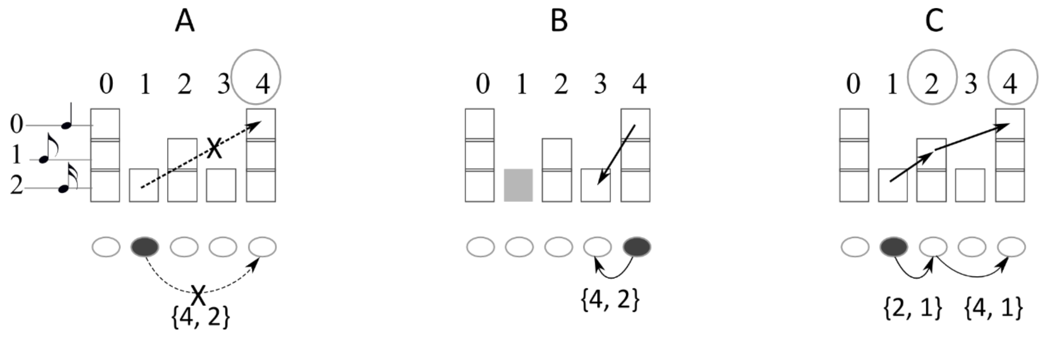


Figure 4-14: Example I of a compound transformation. The single-step de-syncopation from pulse 1 to pulse 4 in A is not allowed because the reverse transformation in B would result in a different pattern. In C, the de-syncopation is performed in two completely reversible steps.

The second example, example II, of a compound transformation is given in Figure 4-15. In this example, the {4,1} syncopation transformation in (A) is blocked by the onset found in pulse 2. However, if we syncopate pulse 2 first, as in (B), the {4,1} syncopation is unblocked (C). The resulting pattern can be de-syncopated in two steps as shown in (D) yielding the array of vectors [{4,1}, {2,1}].

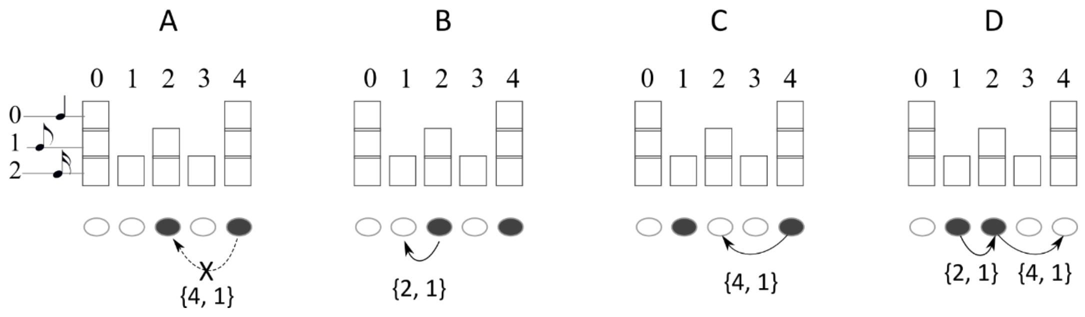


Figure 4-15: Example II of a compound transformation. The syncopation {4,1} in A is not allowed as it is blocked by the onset in pulse 2. Pulse 2 needs to be syncopated first as in B before applying the {4,1} transformation in C. In D, the compound transformation is reversible by de-syncopating first pulse 4 and then pulse 2.

The two examples involve the exact same transformation vectors: $V_1=\{2,1\}$ and $V_2=\{4,1\}$. However, the arrays of vectors are different in that include the two vectors in the reverse order. De-syncopation of the pattern of Figure 4-14 C (example I) yields the array $[V_1, V_2]$, while de-syncopation of the pattern in Figure 4-15 D (example II), yields $[V_2, V_1]$.

The syncopation transformations need to be applied on the de-syncopated patterns of the examples in the correct order, otherwise they would result in forbidden transformations and different patterns. In example I, syncopating consists in applying the array $[V_2, V_1]$ (the reverse of what the de-syncopation of that example yields). Applying first the $V_1=\{2,1\}$ would have no effect, since pulse 2 would not contain an onset. Accordingly, in example II, syncopating consists in applying the array $[V_1, V_2]$. If we would try to apply the $V_2=\{4,1\}$ first, we would find that it is blocked by the onset in pulse 2. The above examples show the importance of the order of the vectors in an array and how to be used in order to reverse a transformation.

The most important application of the arrays is the creation of syncopation trees, described in the following sections.

4.3 THE SYNCOPATION TREE

The two transformations described above can be used to generate paths of syncopation transformations, i.e. a series of specific transformation steps, each of which represents the displacement of a single event. Removing syncopation is a process of forward shifting of the onsets that syncopate to stronger metrical positions (Figure 4-4, left). Each shift of an event results in a new pattern with decreased syncopation. When all events have been shifted to their non-syncopating positions the resulting de-syncopated pattern has no syncopation. Syncopation is generated by anticipating the onsets found in strong metrical position by shifting them to weaker positions (Figure 4-4, right). This time, each shift of an event results in a new pattern with increased syncopation. When all "strong" onsets have been moved to weaker positions the resulting syncopated pattern cannot be further syncopated. On the left side of Figure 4-16, I present an example of how a series of transformations is applied in order to generate a number of patterns with various degrees of syncopation. A different set of transformations would generate a different set of patterns.

In that way, the formalized transformations give rise to hierarchical networks of interconnected patterns that resemble tree structures[25]. The root of each syncopation tree is a non-syncopating pattern. A "branch" of a tree represents the path of transformations that connect two binary patterns; for example, the branch that connects a syncopated pattern to its non-syncopating root or to some other syncopated pattern. The branches that connect all possible patterns originating from a single root form a syncopation tree. In this section I provide with an overview of the generation of syncopation trees and some of their properties. In section 4.4, I provide with recursive algorithms for automatically generate specific branches.

In the example of Figure 4-16, a branch is generated starting from the root pattern by applying a series of transformations until the end of the branch is reached where no pulse can be syncopated further. The root pattern of the example has 3 events and is considered to be a repeating loop (the first event is repeated at the end).

A branch consists of a series of transformations of specific types indicated by the transformation vectors next to each pattern. The specific branch is generated by applying the transformations in a particular order and it is completely described by the array of transformation vectors. The order of the

[25] Syncopation trees are strictly talking lattices and not trees. However, I find the term *tree* preferable as it implies a convenient nomenclature (roots and branches) to describe several aspects of the transformations.

transformations can change to generate a different branch. As long as the type of each syncopation is kept the same, the end pattern will always be the same. In other words, as long as we include the same transformations but apply them in a different order, we will reach the same end pattern. Some intermediate patterns will also be shared among the branches. In the right side of Figure 4-16, we show the branches of the corresponding syncopation tree that include all possible permutations of the transformations in the left (solid lines, the thick solid line is the branch shown in detail at the left). It must be noted that not all permutations are possible. Here, the transformation {8, 1} must always be applied before the {6, 1} otherwise there will be no onset in pulse 6 to be shifted.

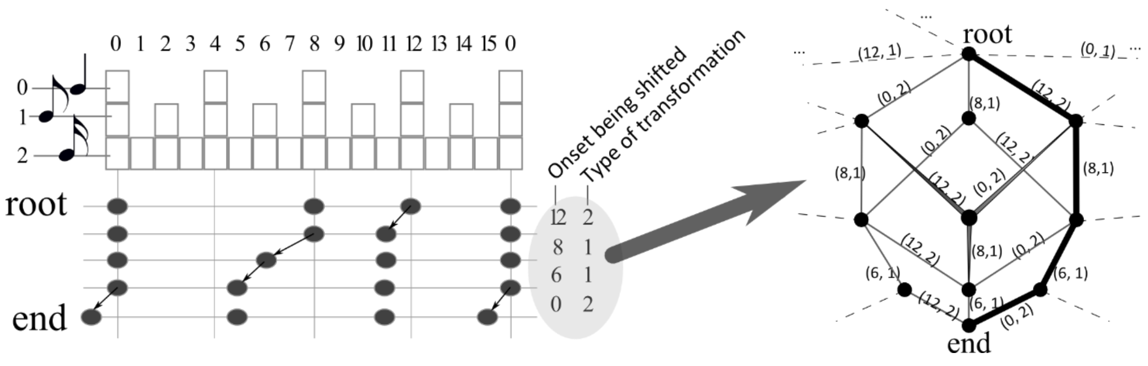


Figure 4-16. Left: An example of a branch starting from the root pattern and finishing to an end pattern that cannot be further syncopated. Right: Example of a syncopation tree (partially shown). The patterns are shown as dots and the lines connecting them correspond to the indicated transformations. The thick line corresponds to the branch shown on the left. The patterns shown are connected with the same set of syncopation transformations applied in a different order.

Each dot in the right part of the figure represents a specific pattern and is connected to the root pattern with a specific number of steps, independently of the exact branch that one might follow. This is a general property of all syncopation trees and not of this particular example. The root pattern as well as the number of de-syncopation steps for a given syncopated pattern is independent of the order in which the pulses are de-syncopated. Only the intermediate patterns differ.

The entire tree is formed by several branches like the ones shown in Figure 4-16. These branches form a network that begins at the root and has several ends. The number of ends depends on the number of possible transformations that can be performed within each beat duration. In our example, the 3 onsets belong to different beats and can undergo 2 different transformations each: either the {*i*, 1} or the {*i*, 2} followed by the {*i*-2, 1}, where *i* is the index of the pulse. Therefore, we have $2^3 = 8$ different ends in this tree. In Figure 4-16, we show only the branches that lead to end pattern shown on the left. The branches that lead to rest of the ends are connected to the patterns of the figure through the dashed lines.

The complexity of the tree depends on the root pattern and the metrical template used. Visualization of the tree can become challenging as the number of the connections of each pattern can be large.

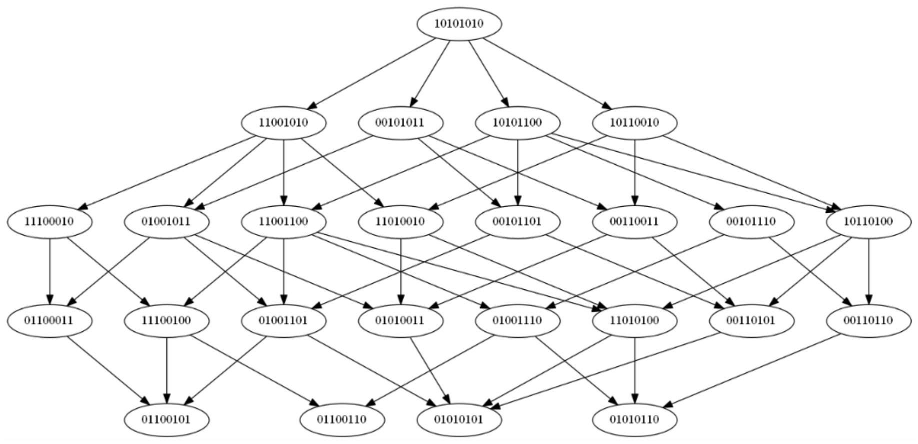


Figure 4-17: An example of syncopation tree. The root pattern in binary notation is 1000 0010 1000 1000 and the corresponding meter is 4/4 stratified to the eighth note level.

Any two patterns in the tree have at least one branch that connects them passing through several other patterns. Generating such branches is equivalent to analyzing the relations between the generated patterns.

Since events cannot be shifted from one beat to another, each beat must contain the same number of events in all patterns of the same tree. As a consequence, patterns with different distribution of events in the beats will have different roots and therefore belong in different trees. However, more than one root patterns exist that have the same number of events in each of their beats (see Figure 4-13 for an example). As consequence, the distribution of events in beats is not the only criterion for two patterns to belong in the same tree.

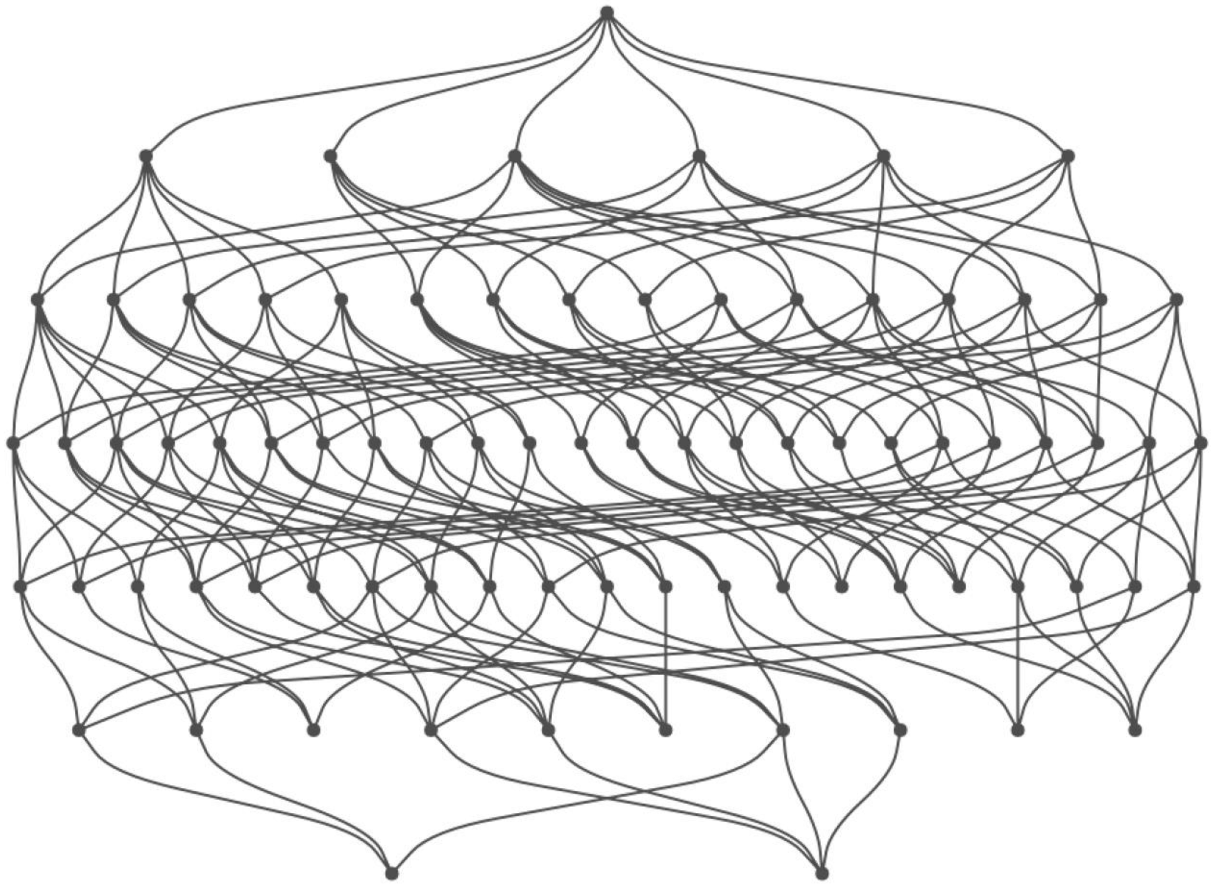


Figure 4-18: An example of a syncopation tree using a different visualization style. The root pattern in binary notation is 1000 0010 1000 and the corresponding meter 4/4 stratified to the sixteenth note level.

4.4 Recursive Algorithms

The generation of a branch of a tree consists in determining the corresponding series of syncopation transformations expressed as an array of transformation vectors. Such arrays can be generated automatically using some recursive process, e.g. by recursively applying a de-syncopation transformation for each pulse that is found to syncopate until there is no syncopating pulse. In this section I give two examples of such processes: in section 4.4.1, an algorithm that can generate an entire syncopation tree given a root pattern and a matching metrical template and in section 4.4.2, an algorithm that can de-syncopate any input pattern with respect to a metrical template and find its root. The two algorithms presented here can serve as examples of how the transformations can be used in recursive processes and they are useful in a wide range of applications in analysis and generation of rhythms. However, other more specific algorithms can be devised that are more effective depending on the intended application.

4.4.1 Generating syncopation trees

In this section I describe an automatic algorithm that can generate an entire tree, starting from the root pattern and gradually generating all possible syncopated patterns. It systematically finds and applies all possible syncopation transformations.

The algorithm takes as an input a binary pattern and a metrical template of equal length. It begins by scanning the input pattern and finding all the pulses that contain an onset. For each onset, all possible transformations are applied. This step consists in determining which syncopation types are allowed for the particular onset with respect to its position in the metrical template. The transformations are applied to the input pattern independently and not one after the other, so that all generated patterns have only one onset shifted compared to the input pattern. Each transformation generates a new pattern that is stored in a list. After applying all possible transformations to the input pattern, the algorithm begins again but this time it uses as an input pattern the next pattern in the list of generated patterns. As the process continues, newly generated patterns are added to the list, which serves as a queue of patterns waiting to be transformed. The process stops when the last pattern in the list is reached and that pattern cannot be syncopated further. With each transformation applied, the algorithm outputs three elements: the initial pattern, the generated pattern and the transformation vector. The three elements represent all edges of the syncopation tree and can be used in order to analyze, navigate or visualize it.

The following pseudo-code illustrates the above process:

```
GenerateTree (RootPattern ¸Template)
  PATTERN_LIST = {RootPattern} // list of patterns, begins with only the root pattern
  int index = 0;
  While (index < PATTERN_LIST.length)
      For each pos in {PATTERN_LIST[index][pos] == True} // for each position of an
                                                          onset in the pattern
    MaxType = max(Template) - Template[pos] //the maximum type possible for the onset
    For (type = 0 to MaxType)
        Vector ={pos,type}
        If (syncopation (PATTERN_LIST[index], Template, result, Vector))
            Append_Pattern (PATTERN_LIST, result) // append the result to the end of
                                                    the patterns list
        Output (PATTERN_LIST[index], result, Vector)
      End
    End
  End
  index ++; // proceed to the next pattern in the list
  End
End
```

The above algorithm has been implemented in the Max/MSP patch [syncopationTree.maxpat] that is part of this dissertation.

4.4.2 De-syncopation

Using the de-syncopation process described here, one can find the root of a given pattern with respect to a certain metrical template. The algorithm generates all intermediate patterns from the original input to the corresponding root as well as a transformation array containing all syncopation vectors.

Figure 4-19 illustrates the process through an example. Pulses are examined in a certain order and when a silent pulse is preceded by an onset in a faster metrical level, the onset is shifted as described in section 4.2.2. In the example the pattern is scanned from right to left. However, the order in which the pulses are examined can be, in principle, freely chosen. As we already saw, the root pattern is independent of the order in which the pulses are de-syncopated. Using a different order can produce a different branch; however, all branches will have the same number of nodes.

The first pulse examined is pulse 12, which carries an onset and therefore does not syncopate. Pulse 11 as well as all pulses of the fastest metrical level cannot syncopate. Pulse 10 is also ignored, as it carries an onset. Pulse 8, 6 and 4 do not syncopate since they are not preceded by an onset in a faster metrical level. The first silent pulse found to syncopate is pulse 2 that belongs to the eight-note level. Event ① precedes it in pulse 1 (sixteenth-note level). Therefore, it is shifted to pulse 2 yielding the undergone syncopation transformation {2, 1}.

The process restarts from pulse 12 and now the first syncopating pulse is 4. The event ① found now in pulse 2 needs to be shifted and the type of syncopation would be 0. Since the metrical level of pulse 4 and 2 is not the slowest one (≠0), the transformation is allowed yielding the transformation vector {4, 0}. In the final round, pulse 6 needs to be de-syncopated resulting in the {6, 1} transformation.

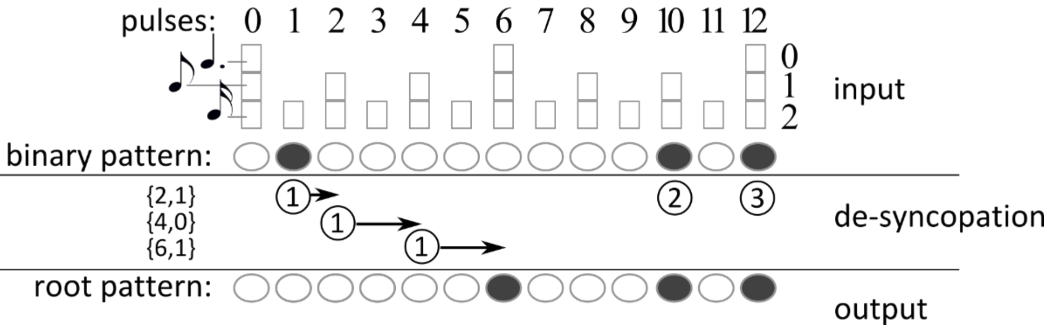


Figure 4-19. Illustration of a recursive de-syncopation process. The metrical template shown corresponds to a 6/8 meter at 180bpm (quarter-note = 1000ms). Each onset of the binary pattern is numbered in a circle. The de-syncopated process is shown as arrows.

The following pseudo code illustrates the basic steps in the de-syncopation algorithm:

```
Recursive_DeSyncopation (InPattern, Template, OrderOfPulses)
  MAXLEVEL = max(Tempalte) //Max metrical level in the template
  Pattern = InPattern
  PATERN_LIST = {empty}
  ARRAY_OF_VECTORS = {empty]}
  Repeat
    For each position pos in OrderOfPulses
        DeSyncopation (Pattern, Template, pos, Pattern, OutVector) // input and output
                                                                 patterns are the same
      Append (PATERN_LIST, Pattern)
      Append (ARRAY_OF_VECTORS, OutVector)
    End
  Until no onsets can be shifted
  Return PATERN_LIST, ARRAY_OF_VECTORS
End
```

The function `DeSyncopation` was defined in section 4.2.2.

The above code receives as input three arrays: 1) the pattern as a binary string (`InPattern`), 2) the metrical template as an array of the metrical indexes (`Template`) and 3) the `OrderOfPulses` array which represents the order in which the pulses should be scanned and the found syncopations should be removed. The algorithm outputs a list of the de-syncopated patterns (`PATERN_LIST`) and the corresponding array of the transformations that connect them (`ARRAY_OF_VECTORS`).

In the example of Figure 4-19 the array of transformations would be:

```
ARRAY_OF_VECTORS = [{2,1} , {4,0} , {6,1}]
```

The above algorithm has been implemented in the Max/MSP patch [de-syncopate.maxpat] that can be found in the accompanying CD.

A couple of examples of default values for the input order array would be the pulses in their natural order, from left to right (0, 1, 2, etc.), or in their order of importance, e.g. according to their metrical level or the indispensability values of Clarence Barlow (Barlow & Lohner, 1987; Barlow, 1987). Not all pulses need to be contained in the order array, since not all pulses carry onsets that need to be de-syncopated. However, it is important to ensure that all pulses that will syncopate in any of the intermediate steps of the de-syncopation process will be included in the input order array. As long as this is ensured, the order of transformations only affects the intermediate steps. The root pattern and the syncopations found are the same for any order. The number of elements in the output arrays, as well as the values of each vector in the arrays, is unaffected by the input order. Only their position in the arrays could in certain cases change, depending on the pattern. In the example of Figure 4-19, the result is always the same; all output arrays and patterns will be exactly the same, independently of the input order.

Another way of ensuring that the result of the de-syncopation process will reach the root pattern uses a feedback loop in the process as shown in Figure 4-20. The principle of the algorithm is to use the initial array of pulses only once but each time a pulse in the array is de-syncopated a feedback loop will automatically de-syncopate all other pulses as needed. For example, in Figure 4-19, in order de-syncopate pulse 6 we needed first to de-syncopate pulse 2 and pulse 4.

The initial array should contain the pulses in the order of the metrical level they belong to. For the template of Figure 4-19, the initial array of pulses would be [0 6 10 8 4 2]. The second pulse in the array is pulse 6. Pulse 4 and 2 need to be de-syncopated first so they are fed back to the input of the de-syncopation algorithm before pulse 6 is processed.

The max/MSP patch [de-syncopate_duration.maxpat] included in the accompanying CD performs this kind of feedback loop. The corresponding max help file demonstrates the entire algorithm.

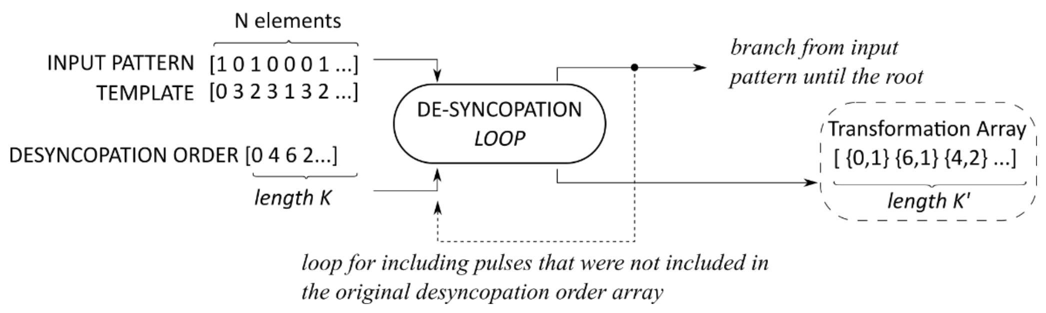

N elements
INPUT PATTERN [1 0 1 0 0 0 1 ...]
TEMPLATE [0 3 2 3 1 3 2 ...]
DESYNCOPATION ORDER [0 4 6 2...]
length K
DE-SYNCOPATION LOOP
branch from input pattern until the root
Transformation Array [{0,1} {6,1} {4,2} ...]
length K'
loop for including pulses that were not included in the original desyncopation order array

Figure 4-20: A recursive de-syncopation process that ensures the root pattern is reached.

It is important to note that the de-syncopation process functions as an analysis, even though it actually generates all the intermediates between the root and the input pattern as a “byproduct”. The root pattern together with the resulting transformation array gives a complete picture of the generated patterns and their relations in a compressed form. The mere collection of the actual intermediate patterns can be thought of as a byproduct of the process.

4.5 Preliminary Evaluation of Transformations

In this section I evaluate the syncopation transformations described in 4.2 in practice, i.e. with actual musical material and not just theoretical binary patterns. I applied the transformations on a collection of MIDI drum loops. Then, I used the LHL syncopation metric described in chapter 3 to measure the syncopation in the original loops and in three more versions that have undergone syncopation transformations: 1) the root pattern, 2) a syncopated version with 30% of the all the possible transformations applied, and 3) a syncopated version with 70% of all the possible transformations applied[26]. The two syncopated versions were automatically syncopated using a default transformation array which might not include the original pattern.

[26] The total of transformation vectors that belong in a branch that connects the root to an end node. For example, if the branch has 11 nodes (10 edges) then 30% of syncopation corresponds to the 4th pattern and 70% to the 8th pattern.

The MIDI collection was taken from (Miron, Davies, & Gouyon, 2013) and consists of 160 drum loops of different music styles. The MIDI files were segregated into three separate MIDI streams according to the MIDI note numbers that correspond to the kick drums, snare drums and hi-hats. All 480 loops (3 x 160) were in a 4/4 meter and quantized to the sixteenth-note grid and they were converted to binary patterns before applying any transformation. In all cases I used a metrical template that corresponds to a 4/4 meter at 100bpm.

First, they were de-syncopated according to the process described in 4.4.2. Second, the resulted de-syncopated patterns were re-syncopated with 30% and 70% of total of the possible syncopations applied. The type of the syncopation was always 2, forcing the syncopation shifts always to two metrical subdivisions faster. The order of the transformations was random. We used the LHL metric to measure the syncopation in all versions of each of the 480 loops. An overview of the results is shown in Figure 4-21.

The de-syncopation and re-syncopation algorithms performed as expected. The de-syncopation algorithm completely removed the syncopation in all 480 binary patterns (syncopation score = 0). This was expected as the de-syncopation process directly corresponds to the way onsets are matched to following silent pulses in the definition of syncopation in the LHL algorithm.

The re-syncopation process increased gradually the syncopation. The 30% transformation increased the syncopation in the majority of the patterns (in 469 out of 480 patterns). The 70% transformation increased further the syncopation for the majority of the patterns (in 445 out of 480 patterns). No pattern had less syncopation in the 70% than in the 30% transformations. In a few exceptions, the patterns had the same syncopation score for the de-syncopated and 30% transformation (11 patterns), or for the 30% and 70% transformations (35 patterns) (horizontal lines in Figure 4-21). In 6 patterns out of the total 480 the syncopation transformations did not create any syncopation.

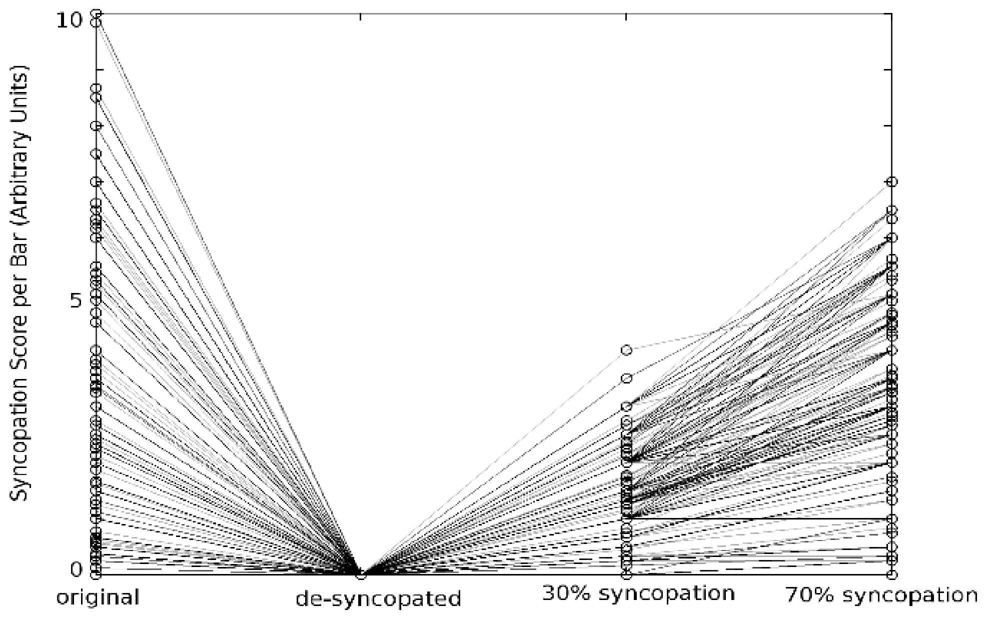


Figure 4-21. Syncopation measurements using the LHL algorithm on 4 different versions of the patterns found in the 480 binary patterns. Each circle represents a single measurement. The grey lines connect the measurements for the 4 different versions of each pattern. As the number of measurements is large, several lines naturally overlap. A darker line means more lines overlap.

The exact relation between the syncopation scores and the percentage of the applied transformations depends on two factors: 1) the number of onsets per bar in the pattern that can actually be shifted and are not blocked by other onsets and 2) their metrical positions. When only a small number of events can be shifted there are accordingly few steps between the de-syncopated version and the fully syncopated version (100% of the transformations applied). The number of events that can be shifted in the pattern strongly depends on the density of events per bar. For very low density (e.g. only a couple of events per bar) or very high density (when almost all pulses are occupied) the onsets available for syncopation shifts are very few. The evaluation shows that the syncopation transformations increase the syncopation with each step.

4.6 Transforming Dynamic Accents

According to the extended definition of syncopation given in section 2.7, accents can produce syncopation. In this section, I focus on dynamic accents that can be expressed as amplitude values of the onsets in a pattern. I will present a method of transforming the amplitudes that is based on the binary transformations presented above and has the same form. In the case where all amplitudes in a pattern are equal then the amplitudes transformation converges to the binary transformation.

If we were to convert the example of Figure 4-22 into a binary pattern ignoring the dynamic accents and the amplitude of the onsets, that pattern would not contain any syncopation. Thus, one cannot use directly the de-syncopation transformations described in the previous sections and cannot shift the syncopating onset from pulse 2 to pulse 4 since pulse 4 already contains an onset. In this section, I propose a method for the de-syncopation of patterns of events of different amplitudes based on the segregation of the rhythms into separate rhythmic layers, grouping onsets according to their amplitudes in a hierarchical manner. In what follows, I first present a simple example of the de-syncopation process that demonstrates the segregation of layers in practice. Then, I present in detail the general method and steps taken in order to transform any rhythmic pattern that contains dynamic accents.

One can think of the event in pulse 2 (Figure 4-22 A) as two overlapping identical events with different amplitudes. As they are heard simultaneously, they cannot be distinguished and they are perceived as a single loud event. These two events can be considered as belonging to separate rhythmic layers and, in that sense, they can be de-syncopated separately. The de-syncopation process is shown in Figure 4-22. First, the rhythm (A) is segregated into two rhythmic layers (B) so that each layer contains only onsets of equal amplitudes. The lower pattern contains no syncopation. The upper pattern is de-syncopated (C) yielding the transformation vector {4, 1}. Finally, the two streams are joined together by adding the amplitudes of each stream. The result (D) is no more syncopated.

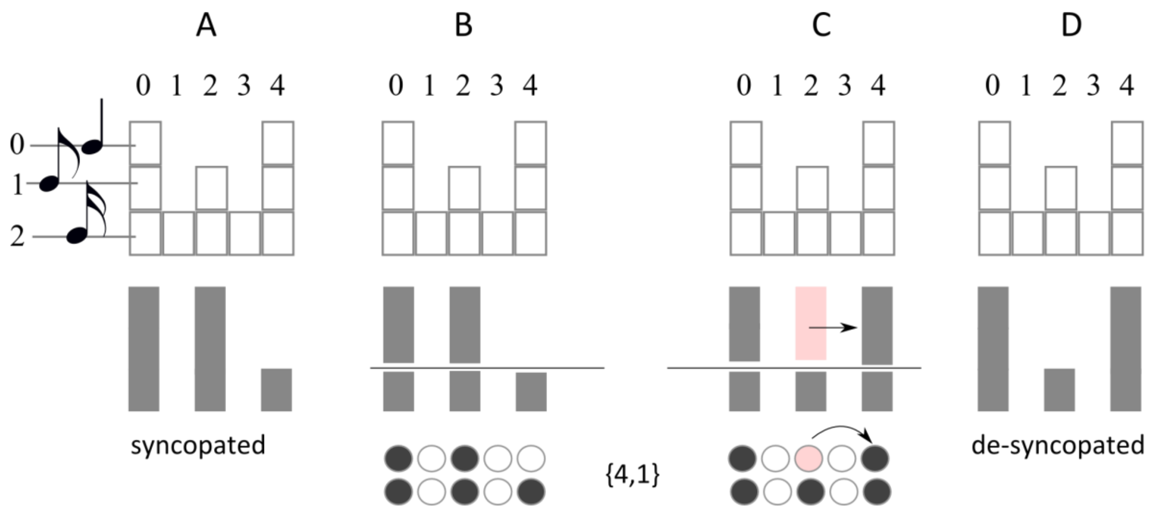


Figure 4-22: example of de-syncopation of dynamic accents (A) by segregating the pattern into hierarchical rhythmic layers (B) that are transformed according to the binary transformations (C). The grey bars represent the amplitude of each event. Above, the metrical template is shown. At the bottom, the de-syncopation of the two rhythmic layers treated as binary patterns is shown.

The above transformation was made possible by turning the accents in the rhythmic pattern into layers of binary patterns. These layers, however, are not independent from each other but they are “stacked” one on top of the other forming a hierarchical structure. By definition, if a pulse in a higher layer contains an onset then all lower layers in the stack also contain an onset at the same position. The hierarchical nature must be maintained throughout the entire process and in the final generated patterns, since combining them back together at the last step of the process is the exact reverse of the segregation.

The general process of applying any syncopation transformations to a pattern of accents comprises three main stages: 1) segregate the accents to rhythmic layers, 2) Apply the transformations, i.e. generate a branch and 3) convert the stack of rhythmic layers into amplitudes. I will demonstrate the details of the three stages using the more complicated example shown in Figure 4-23.

In the first stage, the pattern is converted into a stack of rhythmic layers each of which corresponds to an amplitude value found in the pattern. For each different amplitude value found, a binary pattern that contains onsets in the pulses with equal or higher amplitude is created. The patterns are essentially slices of ranges of amplitude. The binary patterns are stacked in the order of the corresponding slices, i.e. the lower the amplitude the lower the position of the pattern in the stack. In Figure 4-23 B, 4 different layers are created. The lowest amplitude corresponds to pulse 3. Since all other onsets have higher amplitudes (besides of course pulse 0 that contains no onset), the lowest pattern created will have onsets in all pulses (1, 2, 3 and 4). The immediately higher amplitude is found in pulse 4 and the corresponding pattern will only contain onsets in pulses 1, 2 and 4, lacking an onset in pulse 3 that has lower amplitude. In the same manner, two more layers corresponding to the amplitudes of pulse 1 and pulse 2 are created, containing two and one onset respectively.

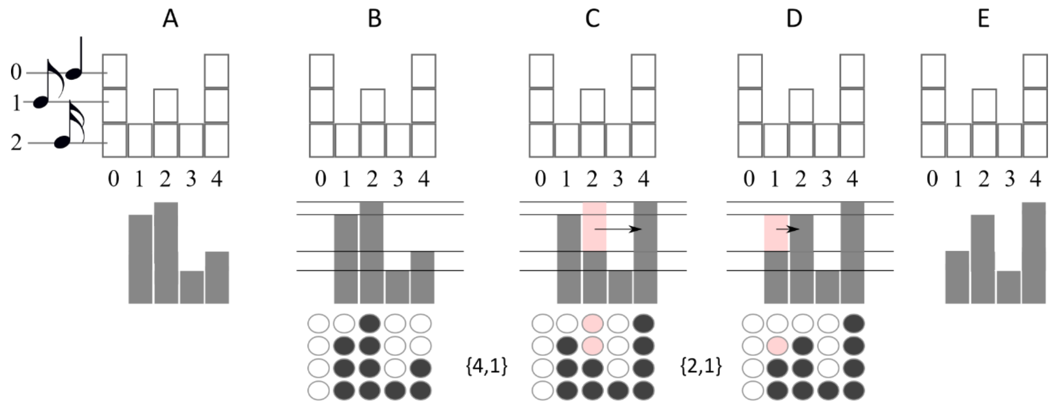


Figure 4-23: a step by step example of de-syncopation of dynamic accents. The process is completely reversible so that following the steps backwards (right to left), syncopation is generated.

In the second stage the transformations are applied. In order to maintain the hierarchy of the stacked layers, the binary transformations cannot be applied independently on each layer but in a hierarchical manner: when an onset in a higher layer is shifted, it must always be placed at a pulse where all lower layers contain an onset. In addition to this rule and in order to preserve the hierarchy in all stages, applying a transformation consists, first, on finding the lowest layer that the transformation is possible and, then, applying the transformation to that layer and all higher layers at the same time. The lower layers, in which the transformation is not possible, and the higher layers, which contain no onsets in the respective pulses, are left unchanged.

In Figure 4-23 C, pulse 4 is de-syncopated. The lowest two layers do not syncopate at this pulse. The third layer syncopates and therefore is de-syncopated by shifting the onset in pulse 2 to pulse 4 yielding the {4,1} transformation. The highest layer is also transformed at the same time. The second transformation applied is the de-syncopation of pulse 2 (Figure 4-23 D). Again the lowest two layers do not syncopate while the third one is de-syncopated yielding the {2,1} vector. The highest layer is not transformed now, since it contains no onset in pulse 1.

The third and final stage consists in converting the stack of rhythmic layers into amplitudes. The process is straightforward. The layers were created from slices of amplitude range, so that summing the amplitude values of the corresponding ranges at the pulses with onsets gives the final amplitudes. In the example of Figure 4-23, the slices shown in (D) give the amplitude pattern of (E).

An important property of the method described here is that the above three stages are perfectly reversible. In the example of Figure 4-23, one can follow the reverse order (from E to A) and apply the two syncopation vectors {2,1} and {4,1} resulting in the initial syncopated pattern. However, the hierarchy of layers must be preserved at all stages of the transformation in order for it to be reversible.

The example of Figure 4-24 shows how a transformation might break the layer hierarchy and how this leads to non-reversible transformations. Applying the syncopation {4,1} is possible on the highest rhythmic layer, though the hierarchy would be broken since the highest layer would contain an onset on pulse 2 but not all lower layers contain onsets in that pulse (B). If, nevertheless, one applies the transformation and tries to sum the amplitudes the result will be the pattern shown in (C). The new pattern would have different amplitudes levels than the original so that when one tries to de-syncopate that pattern as in (C) the resulting pattern would be different than the original in (A). Therefore, breaking the hierarchy results in non-reversible transformations. Such cases however can only occur in syncopation transformations and not in de-syncopation that guarantees, as in the binary patterns, that the resulting pattern can and will always be fully de-syncopated.

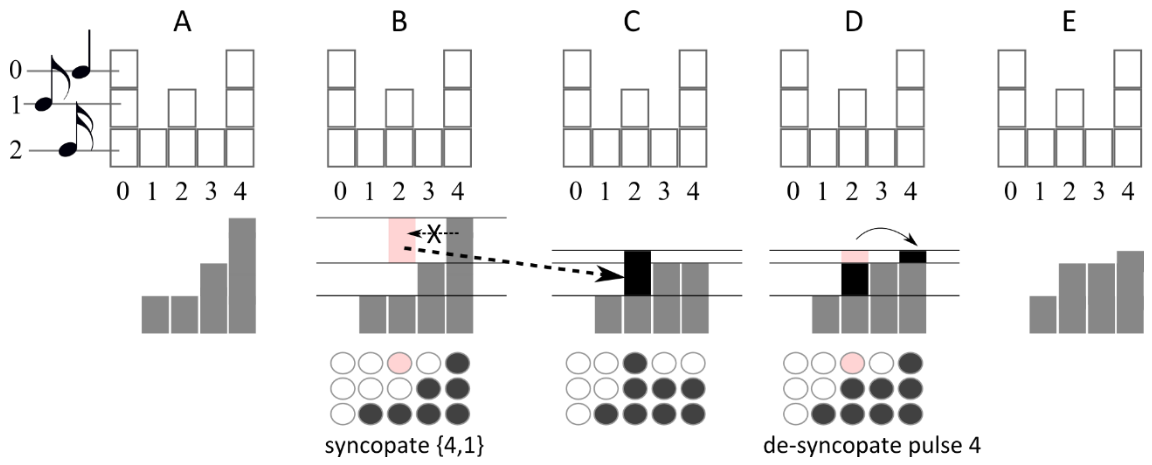


Figure 4-24: If a transformation breaks the hierarchy of layers the result is not reversible. A) The pattern of this example does not contain syncopation. B) The syncopation transformation {4,1} breaks the hierarchy of the layers in pulse 2. C)The result of the {4,1} transformation has different amplitude levels than the original pattern in A. D) de-syncopating pulse 4 returns the vector {4,1} as expected for the reverse transformation. E) However, the resulting pattern is not the initial pattern A.

As a result of respecting the hierarchy of layers at all stages, the transformations do not change the amplitude levels found in a pattern but only their metrical position. They can be thought of as a rearrangement of the amplitudes following the principles of the binary syncopation transformations. As a consequence, the sum of the amplitudes in a pattern is kept constant.

The transformations of amplitude patterns have the same form and properties as the ones for binary patterns and the transformation vectors of the two can be used indistinguishably. In fact, the binary patterns are a degenerate version of accented patterns that only have one amplitude value and therefore consist of a single rhythmic layer. In that way, the algorithms described in section 4.4 can be directly applied on patterns of amplitudes by substituting the single step transformations of section 4.2 with the ones described here. In that way, syncopation trees can be generated for patterns of onsets of different amplitudes. The amplitude levels and the sum of the amplitudes is the same for all the patterns in any given tree.

4.7 Discussion

In this chapter I presented a set of formalized generic transformations that can analyze, generate and manipulate syncopation in binary patterns. The transformations are based on the cognitive definition of syncopation as a violation of metrical expectancies. The transformations serve as a tool for analyzing and generating syncopation in rhythmic patterns. In fact, the two processes are combined under the concept of a syncopation transformation; the transformation describes the syncopation in a pattern at the same time that it generates the pattern itself. I extended the main transformations to include dynamic accents.

The formalization of the transformations is based on the concept of the transformation vector that describes any syncopation in a rhythmic pattern as a metrical position coupled to a metrical level difference[27]. This way of encoding syncopation reveals the nature of syncopation as a manifestation of the interaction of the rhythmic figure coupled to a metrical hierarchy. The rhythm and its figural grouping, i.e. the sequence of durations that constitute the rhythmic pattern and their grouping in rhythmic figures, and the metrical grouping, i.e. the grouping of events according to a metrical

[27] Using two numbers that determine: 1) the non-syncopating metrical position of the onset and 2) the metrical level that the onset is shifted in order to syncopate.

hierarchy, are commonly treated as independent. The transformation vector combines them in the generation of the two dimensional space of rhythm transformations with one dimension being time and the other the perceived meter. However, the transformation space cannot be considered as the combination of two dimensions created independently by different perceptual and cognitive mechanisms. Rather, the space it seems more probable that it is first created and formed as a single entity by the cognitive mechanisms related to rhythm perception in the listener's mind[28]. Then, the two dimensions manifest themselves in different phenomena either independently (e.g. tapping the beat, gestalt perception etc…) or in a coupled form (e.g. the syncopation feel). Even after the transformation space is formed, its two dimensions are not independent and continue to constantly interact.

The transformation vector and the way one can create arrays of transformation vectors leads to the concept of the syncopation tree. The rhythmic patterns found on the branches of the tree are interconnected through specific series of transformations. Any two patterns on a tree can be transformed from one to the other following a branch of single-step transformations. One can navigate from one pattern to another unveiling the relations between the patterns. In each tree, it exists only one pattern with no syncopation, called the root pattern. The particular branch that connects a pattern to its root provides with a detailed description of the syncopation in the pattern, such as the number of syncopating events, the position of the syncopations and the metrical levels involved.

The transformations can also be applied on non-binary patterns where the onsets have different amplitudes. The method includes the segregation of the pattern into rhythmic layers according to the amplitudes. The rhythmic layers, that are stacked in a hierarchical manner, are then transformed as if

[28] As one listens to a rhythmic sequence he creates durational categories, such as short-long or quarter-note, eighth-note, sixteenth-note etc., which are grouped in rhythmic figures the same time as he creates the metrical hierarchy containing the same categories!

they were binary patterns. In the end, the transformed layers are joined back together to form onsets of different amplitudes. In that way, the transformations of binary patterns are a special case of the more general transformation of amplitudes where all onsets have equal amplitudes.

The transformations and syncopation tree can serve as the basis for syncopation, rhythmic similarity or distance metrics. The following syncopation measure demonstrates the working principles of such metrics. The recursive de-syncopation algorithm described in section 4.4.2 identifies and outputs each instance of syncopation found in a given pattern in the form of an array of transformation vectors. In the proposed syncopation measure, the vectors are used to assign weights to the respective syncopations. The overall degree of syncopation is then obtained by summing the weights. If the weights are linearly mapped to the syncopation type found in each vector, then the syncopation measure would be a variant of the LHL algorithm (Fitch & Rosenfeld, 2007; Longuet-Higgins & Lee, 1984). Additional weights can be used corresponding to the first dimension of the vector, i.e. the pulse index, in combination with the metrical template to give, for example, more weight to syncopations on the beat level (tactus). Especially when the de-syncopation is applied on patterns of different amplitudes, each transformation vector corresponds to a certain change in the amplitudes of the onsets while the sum of the amplitudes is kept constant. This amplitude change can be an additional weight since small changes correspond to weaker syncopations.

The syncopation tree and the above syncopation measure algorithms can be viewed also as similarity or distance measures. The syncopation tree can be used to find the shortest path between two patterns that share the same root. The algorithm is a variant of the edit distance that counts the minimum number of editing steps needed to transform one pattern to another, with the editing steps directly corresponding to transformation vectors. In the case of patterns of amplitudes, the algorithm resembles the earth mover’s distance (Rubner, Tomasi, & Guibas, 2000) but with the important difference that the distance is weighted according to the metrical template and the transformation vectors.

The way syncopation is encoded and described by vectors helps in creatively exploring its uses. For example, one can think of modeling the syncopation of one particular music style as specific transformations on certain root patterns. These transformations could then be applied to root patterns that do not belong to the modeled styled. The modeling of a musical style can be done by automatically de-syncopating a number of patterns characteristic to the style, e.g. rhythmic patterns performed simultaneously by different instruments. The resulted transformations could then be combined and applied to the root of any other pattern effectively “copying” the syncopation style. The modeling of a certain style could also be done by statistically analyzing a number of characteristic-to-the-style examples (e.g. histograms of syncopation vectors or Markov chains of syncopation patterns). The application of such a model to a user pattern can be done in real time, thus introducing variation in the performed rhythm that is musically relevant to the modeled style.

In the following chapters, I describe two applications of the syncopation transformations. In chapter 5, I give an example of how the transformations can be used in listening experiments in order to control the syncopation in music examples without affecting the other qualities of the music that commonly are manipulated in a performance, such as expressive timing, e.g microtiming deviations or rubato, the density of events per bar or the loudness of the events. In chapter 6, I present an example of a software plugin which has the form of a loop pedal with a syncopation slider. It is based on an algorithm for the generation syncopation in a given pattern according to simple user restrictions on the syncopation types. The two examples demonstrate the usage of the transformations in practice and can serve as points of departure for the development of other applications. In the final chapter, I discuss other applications of the model in the analysis and generation of rhythms and I present ideas on expanding the model beyond syncopation.

5 UNVEILING THE RELATION BETWEEN SYNCOPATION AND GROOVE

USING THE SYNCOPATION TRANSFORMATIONS

In this chapter, I will describe three listening experiments which aim at better understanding the musical properties that increase the sensation of groove—that is, the sensation of wanting to move when listening to music. Two of the experiments and the corresponding results are published in (Sioros, Miron, Davies, Gouyon, & Madison, 2014) while the third is unpublished work. In these experiments, we examined listeners' experience in groove to synthesized musical stimuli covering a range of syncopation levels and densities of musical events. The stimuli were created automatically by transforming already existing music material through generative processes that affect only specific aspects of the rhythmic patterns such as syncopation and the density of notes.

Such targeted processes can be of great use in listening experiments. They allow us to focus on particular properties of the listening examples that usually are manipulated by musicians jointly with other expressive factors. They provide greater control of the experimental conditions and design at the same time as ecologically valid music examples (MEs) and thus lead to more sound conclusions. The experiments presented in this chapter are examples of how generative processes can be used in listening experiments.

The description of the transformations employed in the experiments and their use in practice as well as some of the results shall shed some light on how the transformations can affect the music material they are applied on. In the first two experiments, simple transformations were applied on piano melodies. In the third experiment, stochastic transformations were applied on more complex material of short rock and funk loops containing three different instruments such as bass, drums and guitar. The transformations employed in the experiments are a combination of syncopation transformations of the kind presented in chapter 4 with other transformations that do not produce syncopation. The non-syncopating transformations were used in order to emphasize the meter and as control conditions against which the effect of the syncopation was verified. In this dissertation, I mainly focus on the design of the transformations and secondarily on the results, while other details of the experiments can be found on the published paper (Sioros et al., 2014). In the detailed description of the particular transformations, I will take the opportunity on occasion to make more general observations concerning the syncopation transformations.

It must be noted that the experiments where designed and conducted before the syncopation model of chapter 4 was fully formalized. This is one of the reasons why the transformations described in this chapter do not faithfully follow the rules presented in the previous chapter. The experiments, besides their main motivation in studying groove, served partially as a way of experimenting with various transformations and discovering their limitations and would eventually lead to the details of the formalization of the syncopation model of this thesis.

In this chapter, the term transformation refers to entire combinations of simple transformation steps that are applied on each rhythmic pattern even when the transformations do not produce syncopation. Because the combined transformations do not correspond to syncopation vectors and arrays, a different naming scheme has been followed when labeling them. Besides these labels, for the single syncopation shifts, their transformation vectors are also given.

The results of the experiments indicate that moderate levels of syncopation in the piano melodies lead to significantly higher groove ratings than no syncopation or the maximum possible syncopation. A comparison between the various transformations and the way they were rated shows that there is no simple relation between the magnitude of syncopation and groove. The results of the third experiment, although they are not as clear as the other two experiments due to the stochastic nature of the applied transformations, are still informative. They are in agreement with the other two experiments in that it is not enough to measure the magnitude of syncopation as a simple scalar quantity in order to understand its effect in high level perceptual properties of rhythm such as groove.

I begin this chapter with a brief introduction to the relevant groove literature in section 5.1. In sections 5.2, 5.3, and 5.4, I will present the three experiments. Finally, in section 5.5, I will discuss the findings of the experiments.

5.1 Groove in the Literature

Certain types of music induce the desire in humans to tap their feet, move their body, and dance, a feeling that is commonly known as groove. Janata et al. (2012) crowed-sourced a definition of groove by asking participants in a synchronization experiment to describe their experience. Based on the frequency of the words in the responses, they formed a definition: “groove is the aspect of the music that induces a pleasant sense of wanting to move along with the music” (p. 56). In the following, we adopted Madison’s (2006) definition of groove: “the sensation of wanting to move some part of the body” (p. 203) in relation to some aspect of the sound pattern.

Madison, Gouyon, Ullén, & Hörnström (2011) have shown that the propensity to move differs substantially from one piece of music to another, but can be linked to certain rhythmical properties of the musical signal. They found that low level descriptors, such as beat salience, event density and fast metrical levels, highly correlate with groove. In contrast and unlike what musicological literature points towards (Keil, 1995), microtiming does not increase groove. The latter was confirmed by a systematic analysis conducted by Davies, Madison, Silva, & Gouyon (2013) on simple rhythms. Madison et al. (2011)

claim that tempo alone cannot explain groove. However, Janata, Tomic, & Haberman (2012) conducted an extensive study on sensorimotor coupling, where the computational part of the analysis suggests that music genre and faster tempi have a strong effect on the desire to move along to the music.

Groove has a strong affective component, as well as a strong correlation to music appreciation (Madison, 2006). According to Janata et al. (2012), we find pleasurable the music that makes us want to dance and such music elicits spontaneous rhythmic movements. Syncopation was also reported to elicit pleasure to listeners (Keller & Schubert, 2011).

The experiments I will present in this chapter were motivated by the results of a previous experiment on what musicians do to create groove when they are confined to a monophonic melody and an isochronous beat (Madison & Sioros, 2014). The study revealed that musicians tended to shift the onset and offset positions of the notes to faster metrical levels in order to increase groove in their performances. When they were asked to minimize the groove they employed two different methods: either they simplified the rhythmic structure towards slower metrical levels, i.e., from 16th and 8th notes to 8th, 4th or half notes, or by breaking the rhythm such that the sense of a regular beat disappeared.

While the results from Madison and Sioros (2014) appear to indicate that the main physical correlate of groove was syncopation, the design of the experiments was not fully controlled, since the musicians could—and did—change several aspects of their performances simultaneously. It was therefore difficult to determine if the changes in some performance parameters were redundant or interdependent, and also the unique effect of each one of them. We needed therefore to experimentally test the hypothesis that syncopation per se induces the sensation of groove independently of other expressive or structural features of the performances.

In another study related to groove and syncopation, Witek et al. (2014) demonstrated a listener preference for moderate syncopation in drum patterns. In the study, funk drum-breaks with varying degrees of syncopation were taken in their majority from real music tracks (36 out of 50) and their degree of syncopation was then measured using a variant of the lhl algorithm (1984). The study concluded that gradually increasing the syncopation also increases the desire to move, but only up to a certain point beyond which desire decreases.

In our study, we aimed at eliminating all other expressive factors of a musical performance and focus solely on syncopation as the musical property that can drive the perception of groove. To this end, I developed computer algorithms that generate syncopation in monophonic melodies as well as more complex music material including several instruments. The algorithms are based on changing only the onsets' metrical positions, without altering other expressive or structural characteristics, according to the syncopation transformations described in this thesis. While in the Witek study (2014) syncopation was measured in pre-existing drum loops, in our study we can generate and vary using an automatic algorithm over which we have complete control.

We designed two listener experiments using simple piano melodies and a third one using short funk loops that were transformed in an automatic and controlled way in order to generate syncopation without affecting other rhythmic or structural characteristics. In the first experiment we explore the more general hypothesis that increasing the syncopation to a moderate amount and introducing faster metrical levels results in an increased groove sensation. Then, in the second experiment, we aim to gain more insight on how the strength of each individual syncopation felt in a melody influences groove as well as distinguishing between different ways of introducing faster metrical levels. The third experiment aimed at looking at the relation of syncopation to groove in more complex music signals that contain more than one rhythmic stream. Parallel to the syncopation transformations, density transformations that increase the number of events in the MEs are used as controls in order to verify the effect of syncopation.

Our predictions for the outcome of the listening experiments were: (1) the deadpan versions of the melodies that contain no syncopation would be rated lowest in groove, (2) introducing a small number of strongly syncopating notes would be rated highest in groove, (3) an increased number of syncopated notes would not have a proportional increase in groove ratings or would be rated lower.

5.2 Experiment A

Our aim was to measure and understand the effect of syncopation on the sensation of groove, independently of other structural or expressive factors. We wanted, for example, to eliminate any expressive features of human performances, such as timing deviations from the metrical grid or dynamic accents. To this end, we used simple piano melodies with a clear metrical structure that we algorithmically transformed. All melodies were in 4/4 meter and consisted of short melodic phrases of two to four bars as in the example of Figure 5-1. The melodies contain simple rhythmic figures such as the ones found in melodies for children and were distinguished into two groups: 1) five simple melodies composed for the purposes of the experiment (hereafter SIMPLE), and 2) more complex traditional songs for children from different cultures (hereafter COMPLEX). For each melody, we generated four additional variations by displacing certain notes from their original positions.

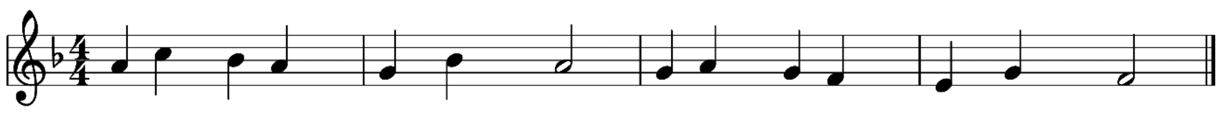


Figure 5-1: Example of a piano melody of experiment A.

All melodies were of moderate tempo (120 BPM). The original melodies included only metrical levels equal to or slower than the quarter note. They contained no syncopation (Figure 5-1) and were all quantized so that their timing did not deviate from the metronomic positions. While the original version contained only slower metrical levels, applying any of the transformations (Figure 5-2, staves B to E) introduced the eighth note metrical level (250ms) and/or the sixteenth note metrical level (125 ms).

Simple Max for Live devices were developed in order to automatically apply the corresponding transformations. A detailed description of the transformations follows. The MIDI files of all melodies and their respective variations are available as supplementary material in the CD that accompanies this desertation.

5.2.1 The transformations

Syncopation is introduced to the melodies by anticipating the last note of each melodic phrase. In Figure 5-2, an example melody is shown together with the syncopation transformations marked with red arrows and the corresponding vectors. The metrical template for the 4/4 meter is also shown below the staves[29]. One of our main concerns in this experiment was to be able to have as pronounced results as possible using an automated computer algorithm for introducing syncopation. To this end, the syncopation transformations were accompanied by the non-syncopating shifts depicted with green arrows in Figure 5-2. Such secondary shifts (green) would maximize the feeling of each syncopation (red) at the end of each phrase as I explain in the following paragraph, with the expectation that this would be reflected in the groove ratings. Although we considered including in the experiment a variation of the

[29] As the tempo is 120 BPM, the duration of the quarter note is 500ms and that of the half note is 1s. The slowest metrical level of the template is supposed to be the one that falls in the region between 500ms and 1s (see section 4.1) and, therefore, in this case, it could be either the quarter-note or the half-note, since they both fall at the limits. Here it was arbitrarily chosen the quarter-note level.

MEs with only the syncopation shifts, we decided against it as it would increase significantly the duration of the experiment.

The secondary shifts were employed in order to enhance the metrical feel before the syncopation occurs at the phrase boundaries. They consist in delaying the notes found on the weak beats of the 4/4 bar, i.e. the second and fourth beat of each bar. In this way, the notes articulated on the strong beats, the first and third beats, is now lengthened and therefore felt as accented (Dirk-Jan Povel & Okkerman, 1981). The above transformations result in a strong metrical feel in the beginning of each phrase, so that the anticipated note at the end of each phrase is felt as strongly syncopated (Figure 5-2, staves B to E). Some aspects of this non-syncopating shifts are discussed in sections 0 (as transformations that cannot systematically produce syncopation) and 7.3.3 (as future work in expanding the model beyond syncopation transformations).

Two different syncopation transformations are applied of *type* **1** (staff B and D, vectors {*i*, 1}) and **2** (staff C and E, vectors {*i*, 2}). The value of the *type* of the syncopation shifts depends on the beat level, that is, the slowest metrical level included in the metrical template, but not on the fastest subdivision of the beat. In contrast, the index number *i* of each pulse is defined for the particular fastest metrical subdivision chosen when stratifying the meter and is independent of the choice of beat level. Essentially, the pulse index is used only to translate the position of the transformation to the reference of the metrical template. In the case of the transformations applied here, the positions are determined by the boundaries of the phrases, which are then translated into pulse indexes only to be put in the corresponding vectors. In general, depending on the application, one can replace the pulse indexes in the transformation vectors with a corresponding descriptive rule that can unequivocally determine the syncopation positions. In this way, the dependence of the transformation vector to the metrical template is more relaxed. For example, the transformation vectors could be expressed as {Phrase Boundary, *type*} so that they can be applied to any melody irrelevant of its meter.

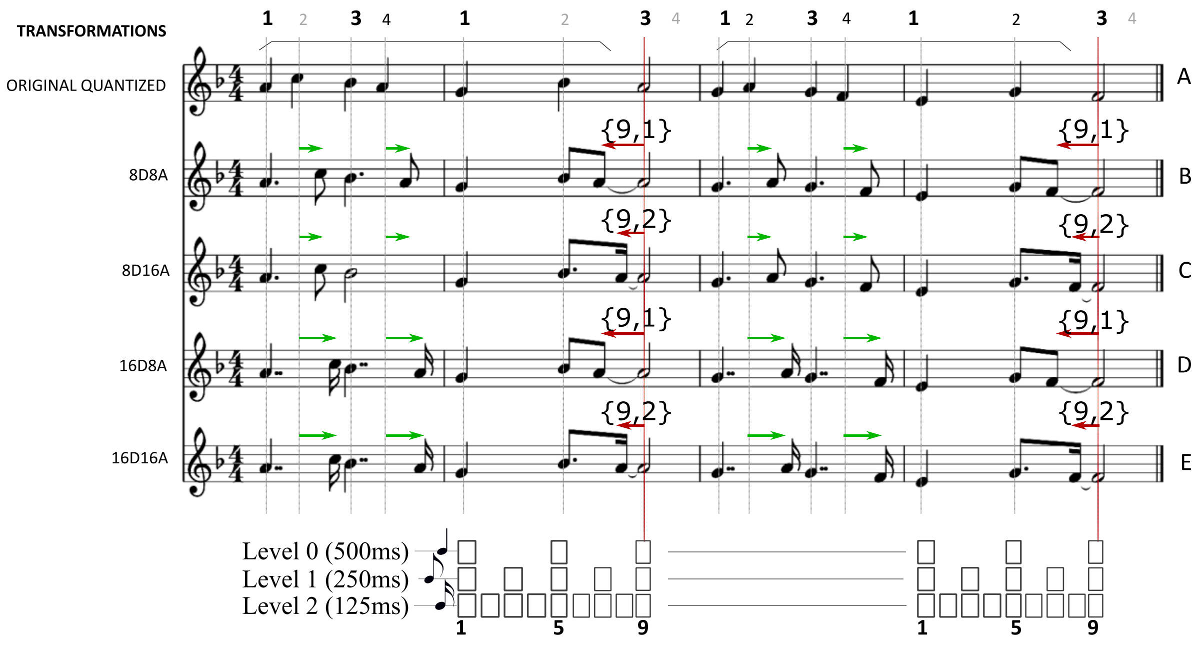


Figure 5-2: The piano melody of Figure 5-1 and all the variations of experiment A. The beats are annotated above the staves. Below the staves the metrical template meter is shown. Below the metrical template the pulse numbers are shown. The red arrows depict the syncopation transformations and the corresponding vectors are shown in curly brackets. The green arrows depict note shifts that do not produce syncopation. (A) The original quantized melody. An arch marks the duration of each phrase of the melody (in this example: the first two bars and the last two bars). (B) 8D8A. The last "strong" note of each phrase is anticipated by an eighth note. The "weak" quarter notes are shifted forward to eighth note positions. (C) 8D16A. The last "strong" note of each phrase is anticipated by a sixteenth note. The "weak" quarter notes are shifted forward to eighth note positions. (D) 16D8A. The last "strong" note of each phrase is anticipated by an eighth note. The "weak" quarter notes are shifted forward to sixteenth note positions. (E) 16D16A. The last "strong" note of each phrase is anticipated by a sixteenth note. The "weak" quarter notes are shifted forward to sixteenth note positions.

Four different variations were generated that differ on the metrical subdivisions that the displaced notes were found after the transformations. They are labeled by two numbers, *x* and *y*, in the following form *xDyA*, as in the left side of Figure 5-2. The *x* denotes the metrical level that the weak quarter notes were shifted to (green arrows) and *y* is the metrical level of the syncopation of the last note of each phrase. For example, in the *16D8A* transformation (Figure 5-2, staff D) the notes originally articulated on weak quarter notes were delayed to the sixteenth note position directly preceding the next strong beat, while the last note of each phrase in the melodies was anticipated to the preceding eighth note metrical position.

5.2.2 Analysis

The details of the statistical analysis can be found in Sioros et al. (2014). In Figure 5-3, the mean groove ratings are shown. Significant differences in the groove ratings can be seen between the different versions where the mean value of groove of one version does not overlap with the confidence interval of another. The simple melodies generally got lower ratings, but the effect of the transformations was larger than on the complex melodies. The transformed versions have only small differences between the simple and complex melodies. A small but still significant difference is observed for the complex melodies, where the 16D*A note variations are somewhat more effective than the 8D*A ones.

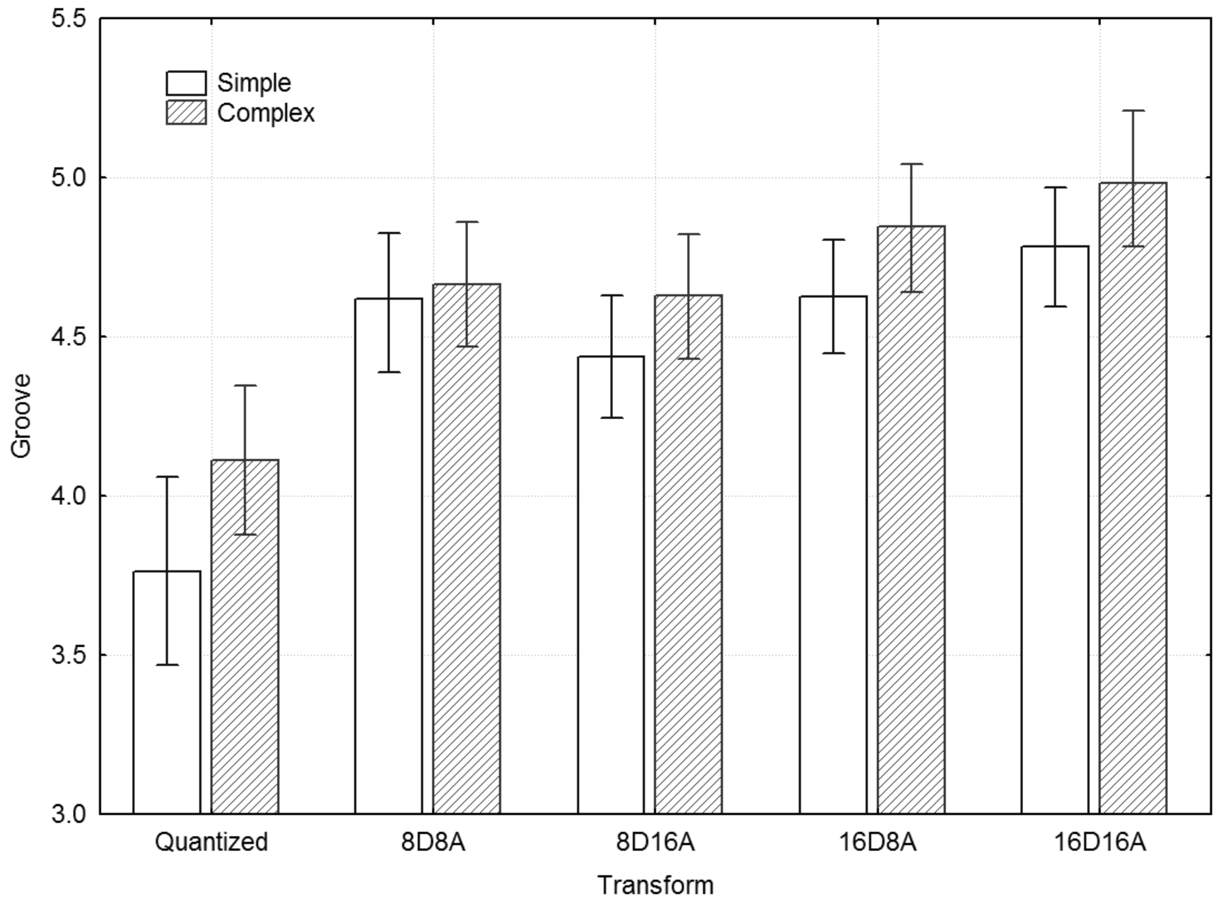


Figure 5-3: Mean groove ratings for variation and melody type across participants.

5.3 Experiment B

In Experiment A, we observed that the transformations increased the groove ratings of the simple melodies. However, the observed effect might arise from the faster metrical levels introduced by the transformations and not be particular to the generated syncopation. In other words, any transformation that can introduce faster metrical levels but preserves the structure of the melodies could result in higher groove ratings even if it does not syncopate. In Experiment B, we aimed to investigate if syncopation is indeed required by comparing the effect of syncopation to other non-syncopation transformations.

Additionally, we employed a set of three transformations that vary in the degree and strength of the syncopation that they generate. We wanted, in that way, to examine in more detail how syncopation affects the sensation of groove. A hi-hat sound was used as metronome to provide a strong metrical reference in order to be able to achieve a higher degree of syncopation without altering the perceived meter. Five of the traditional complex songs from Experiment A and all five simple melodies that were com-posed for the purposes of Experiment A were used.

We applied two kinds of transformations: (1) transformations that introduce syncopation (Figure 5-4, staves B, C and D) and (2) density transformations that double the number of notes per bar in the melodies without introducing syncopation. (Figure 5-4, staves E and F). In total, we generated six different versions of each melody: the original, three syncopated versions and two with increased density. The melodies were manually segmented into short phrases of two to four bars as in Experiment A. The transformations were then applied on each melodic phrase separately.

Similarly to experiment A, the original version of each melody contained only slower metrical levels. The transformations introduced the faster eighth note metrical level (250 ms) but no sixteenth notes. A description of the syncopation transformations follows. The details of the density transformations can be found in the published paper (Sioros et al., 2014). The MEs were generated in Ableton Live in the same way as in experiment A. The MIDI files of all melodies and their respective variations are available as supplementary material in the accompanying CD.

5.3.1.1 Syncopation Transformations

We applied three transformations that generate syncopation of different degrees. The first two, labeled 8D and 8A in Figure 5-4, are similar to the transformations employed in experiment A. The 8D transformation is identical to the 8D8A of Experiment A that introduced a syncopating event at the end of each phrase (Figure 5-2, staff B) by anticipating the last note of each phrase by an eighth note.

The second transformation, the 8A in staff C of Figure 5-4, has several additional syncopations and therefore the overall syncopation feel of the melody is also stronger. We generated additional syncopation by anticipating the notes found on the weak beats in each phrase, i.e. on the second and fourth quarter note of each bar. Those syncopations are felt less strongly since they involve faster and weaker metrical levels (see chapter 3 on measuring syncopation), i.e. the quarter note metrical subdivision instead of the half bar or the bar.

This example brings to the fore an ambiguity in how the metrical templates are used in measuring the strength of the syncopation feel and in generating syncopation. In the syncopation measures I presented in chapter 3, the metrical template is generally used for two reasons: 1) to identify where a syncopation occurs and 2) to determine the weight or strength of the syncopation. In such syncopation measures, the two steps are performed using the same metrical template. As I discussed in chapter 4, the de-syncopation transformations could be used in order to identify the syncopations in a pattern. In that way, this thesis suggests that the first step of identifying the syncopations should be performed using a metrical template where the slower metrical levels (slower than the beat level) are ignored. However, when it comes to defining weights in order to measure the strength of each syncopation the complete template is more appropriate, so that a syncopation at the downbeat is weighted more than on a weaker beat. One could even include hyper-metrical levels—of longer duration than the bar that are usually related with more structural characteristics of the music such as the length of the phrase—in order to give more weight to certain metrical positions like, in this case, the last syncopated note of each phrase.

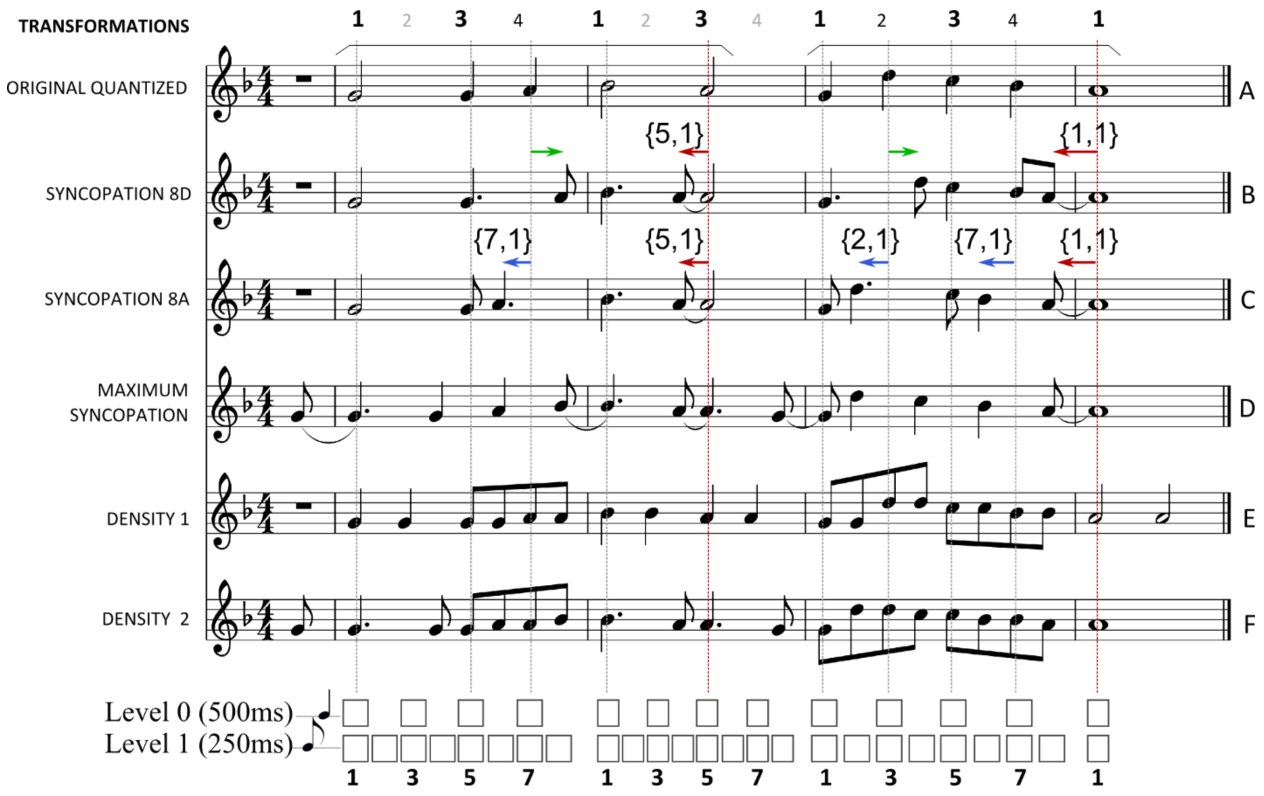


Figure 5-4: Example of a piano melody and all the applied transformations of experiment B. The beats are annotated above the staves. Under the staves, the corresponding metrical template is shown. (A) The original quantized melody. An arch marks the duration of each phrase of the melody (in this example: the first two bars and the last two bars). (B) 8D. The last "strong" note of each phrase is anticipated by an eighth note. The "weak" quarter notes are delayed by an eighth note. (C) 8A. The last "strong" note of each phrase is anticipated by an eighth note. The "weak" quarter notes are anticipated by an eight note. (D) Maximum syncopation. All notes have been anticipated by an eighth note. (E) Density1. All note durations have been divided in two thus duplicating each note. (F) Density2. Before each note of the original melody, a new note of the same pitch is introduced.

In the two transformations described above the downbeat and the half bar notes were kept in their original positions, thus the perceived meter was not affected. In contrast, in the third syncopation transformation, labeled “maximum syncopation” in Figure 5-4 (staff D), all notes of the original melodies were anticipated by an eighth note. Although such a transformation shifts the entire metrical perception, syncopation was produced through the phase offset between the melody and the metronome. The transformation could be expressed in a series of transformation vectors of the form {*i*, 1} where *i* takes the values of the indexes of all the pulses at the beat level, that is, 1, 3, 5, and 7 in each bar. The vectors are omitted in the figure for clarity.

5.3.2 Analysis

The details of the analysis can be found in Sioros et al. (2014). Figure 5-5 summarizes the main findings. The melody type—simple or complex—did not have a significant effect on the groove ratings.

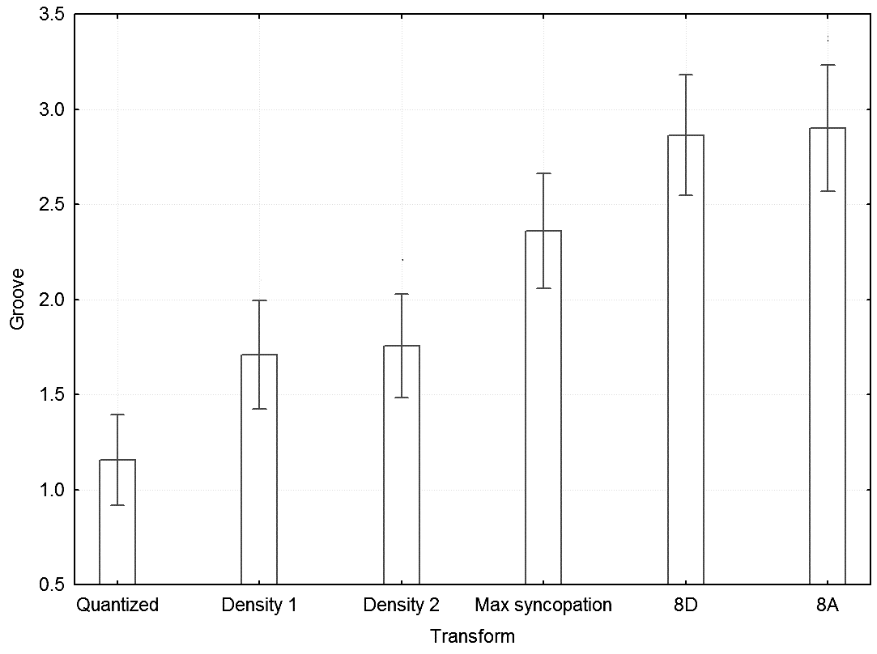


Figure 5-5: Mean groove ratings for each transform, across participants and melodies (including melody type: simple or complex)

Significant differences in the groove ratings are indicated by the confidence intervals in Figure 5-5. For example, the max syncopation transformation has a significant difference in the groove ratings from all other transformations, while the mean rating of the density 1 transformation is not significantly different from that of density 2. The result clearly shows that the two sparse syncopation transformations were the most effective.

5.4 EXPERIMENT C

In the first two experiments, a correlation between the syncopation transformations in the simple monophonic piano melodies and the sensation of groove was found. The aim of this third experiment was to explore how syncopation transformations could have a similar effect on more complex polyphonic music examples. At the same time, we wanted to explore the relation between the amount of syncopation present and the evoked groove sensation. To this end, I created an algorithm based on the transformations of chapter 4, which can be automatically applied on short loops containing multiple instruments.

5.4.1 Participants and Procedure

Twenty seven participants (10 female, 27 male) took part in the experiment (mean =32.7 years, SD = 6.9 years). Of the twenty seven participants, eleven of them had no music training and seventeen of them were professional musicians with more than eight years of music training. The participants were recruited via email, did not participate in experiment A or B and did not receive any remuneration for their participation. The procedure and scales were the same as in experiment B (Sioros, Madison, et al., 2013).

5.4.2 Stimuli

Ten short loops of two to four bars were created for the purpose of this experiment. We aimed at a high ecological validity of the MEs, while, at the same time, eliminating all other expressive factors of a musical performance and focusing on syncopation. To this end, we based the MEs in preexisting songs that we adapted accordingly in order to control for the various factors. Six MEs were based on music excerpts taken from commercial songs and four were based on songs from the RWC Music Genre[30] dataset, all in the funk or rock music genre. The original excerpts included several instruments; however, the MEs were created from MIDI versions that were adapted for only three instruments—bass, drums and guitar or keyboards—and were quantized so that they contained only sixteenth notes. A tempo of 120bpm was chosen for all of them regardless of the original tempo of the songs. A set of transformations was then applied to the original loops resulting in five different versions for each loop.

The MEs were created and generated in Ableton Live as short loops. I developed a Max for Live device that automatically applied the corresponding transformations. MIDI files were generated for all 6 versions of each melody and were then rendered into 16 bit wave files using high quality samples.

Unlike the MEs of the previous two experiments, in experiment C, the original MEs were already syncopated. Therefore, two separate algorithms were employed: 1) to remove the existing syncopation and, 2) to generate new original syncopation. The de-syncopation algorithm used is the one described in section 4.4.2. The syncopation algorithm was developed for the specific needs of the experiment and is described in detail in the following subsection. Besides the original quantized version, four different versions were generated: first, the deadpan de-syncopated version was generated and then the syncopation algorithm was applied on the deadpan version generating 25%, 50% and 70% of the total

[30] http://staff.aist.go.jp/m.goto/RWC-MDB/ (accessed: December 5, 2012)

maximum syncopations. The transformations were applied on each instrument separately and independently. The drums were split into two streams, hihats/cymbals and kick/snare drums that were treated as different instruments. In Figure 5-6, an example of a loop and its transformations is given in piano roll representations (the 25% syncopation version is not shown for clarity).

The MIDI files of all melodies and their respective variations are available as supplementary material in the accompanying CD.

5.4.2.1 Syncopation transformations

The syncopation transformations were developed to satisfy two important requirements of experiment C. First, they should be applied on MEs that were short, polyphonic, repetitive loops with no long structure like the melodic phrases of the piano melodies. Second, we wanted to explore the relation between the amount of syncopation and the groove sensation, and therefore, the algorithm should give us control over the amount of syncopation introduced. For example, the algorithm should allow us to apply a certain percentage of syncopated notes independently of the rhythmic structure of each loop. For these reasons, I developed a stochastic algorithm that introduces syncopation in random positions instead of structurally specific positions.

The algorithm begins by counting the number of possible syncopations as the number of notes that do not belong to the fastest metrical level, in this case all notes that belong in the eighth-note level or slower. Then a certain pre-defined percentage (25%, 50% or 70%) of note events is selected at random to be syncopated. Each of the selected note events is then anticipated in an earlier position in order to be syncopated. It must be noted that the three syncopation transformations of different amounts are all applied independently on the deadpan version and that the selection of notes to be shifted is random and different for each of the 25%, 50% and 70% transformations.

Each note to be syncopated undergoes a *type* 1 transformation (transformation vector {*i*, 1}), i.e. it is shifted to a metrical position one metrical level faster than its position in the deadpan version. When such a transformation is blocked then a *type* 2 transformation is tried out (two metrical levels faster) if the corresponding metrical level exist. If it is blocked as well, then a *type* 3 is examined. In the bass piano roll representation of Figure 5-6, three transformations exist that demonstrate the above rules: {5, 1}, {9, 2} and {21, 2}.   Such rules were preferred to the simpler shifting always to the fastest metrical level because they give greater diversity in the metrical levels and inter-onset intervals present in the resulting patterns. Such complexity of the rhythmic structures is commonly found in groove music like in funk or rock.

The transformations that are blocked by a note in the fastest metrical level, i.e. at the sixteenth-note level, were also allowed in the generation of the MEs of this experiment in order to more accurately control the number of syncopations generated[31]. In this case, the sixteenth note blocking the transformation is shifted together with the syncopated note in the slower metrical level. In the piano roll representation of the bass in Figure 5-6, the note in pulse 1 of the deadpan version is displaced to pulse 29 in the 70% transformation. Such a shift is blocked by the sixteenth note existing in pulse 31. However, the sixteenth note is shifted together with the note on the downbeat turning the transformation possible.

In certain cases, the sixteenth note cannot be shifted because other notes in slower metrical levels are on the way. In the kick/snare piano roll representation in Figure 5-6, the kick in pulse 1 is shifted to pulse 32 where a snare event existed. The snare cannot be shifted together with the kick because of another

[31] In the MEs of moderate number of notes, the number of shifts is naturally limited by the number of notes. If some shifts are blocked, it can happen that the 70% variation cannot be generated.

snare existing already in pulse 30. In this case, the sixteenth-note snare in pulse 31 is muted and the transformation takes place as if it never existed.

In the two examples above, the notes found in sixteenth-note positions in the deadpan version are ignored and are treated as grouped with the following note that belongs in a slower metrical level. They are transformed together when needed, either by shifting them or by muting them. Such displacements or the muting of sixteenth notes in order to give space for a transformation to occur result in non-reversible transformations; if the resulting pattern was to be de-syncopated the displaced or muted sixteenth note would not return to its initial position or state and the deadpan version would not be recovered. This violates the main constraints of the transformations described in chapter 4 and therefore the transformations applied here do not share all the properties of the transformations of chapter 4. However, for the purposes of this experiment, properties such as reversibility are not important. Instead, it was preferred to shift at random any note of the deadpan version in order to generate syncopations that are better distributed throughout the duration of the loop as well as to provide a more precise control of the number of syncopations generated.

The original MIDI note durations were in most cases kept. Some durations were lengthened in order to respect the legato of the original versions or shortened in order to avoid overlaps. The adjustment of the note durations was done manually after inspecting the automatically transformed MIDI clips.

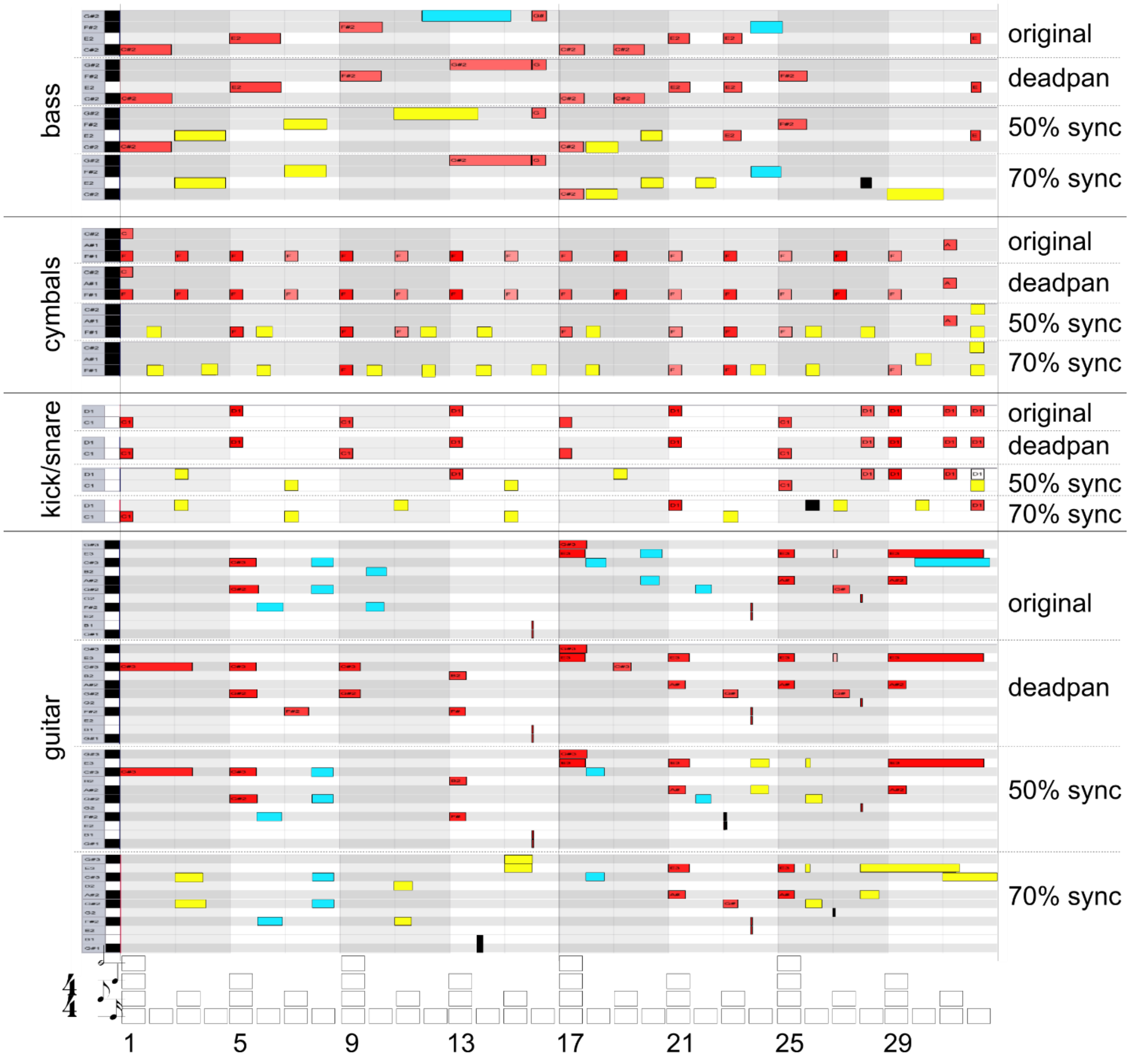
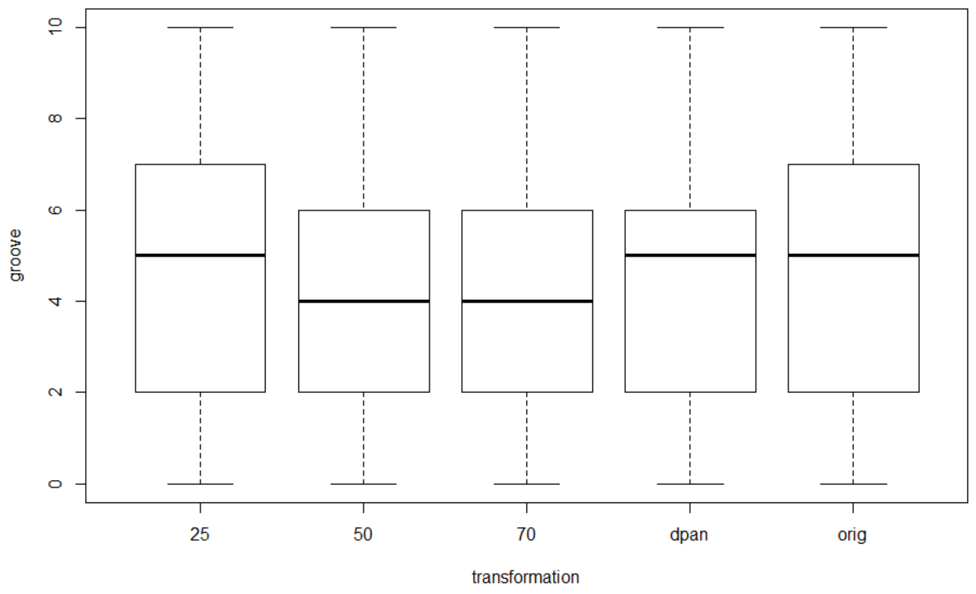


Figure 5-6: Piano roll representations of the "Ladies' night" example and the transformations applied. The 25% syncopation transformation was omitted. In red, the non-syncopating note events. In light blue, the syncopations that exist in the original versions. In yellow, the syncopations that only exist in the transformed versions. In black, shifts applied to notes that were blocking a syncopation transformation. Below the piano roll representations, the metrical template and the pulse indexes are shown.

5.4.3 Analysis

In Figure 5-7, the groove ratings for each transformation are shown. Because of the great variability in the ratings it is difficult to draw conclusions directly from the mean values. To better explore the data and reduce between-rater variability the groove ratings were Z-transformed32 and the mean values of the result are shown in Figure 5-8 (for each ME) and in Figure 5-9 (across the MEs). As one can see in the figures, the original version is rated significantly higher in groove from all transformed versions. However, no particular transformation was rated significantly higher than the deadpan version. In contrast, the 50% and 70% transformations were rated lower than the 25% or deadpan versions. The difference between the ratings for the 25% and deadpan version is not statistically significant.



.

Figure 5-7: Box plot of the groove ratings vs the transformations for experiment C.

[32] Transformation $Z = \frac{X-\bar{x}}{s}$, where X is the original data points and s is their standard deviation. Z represents the relation of the data points (groove ratings in this case) to the mean value in units of the standard deviation and it is typically used when one needs to compare data from different data sets.

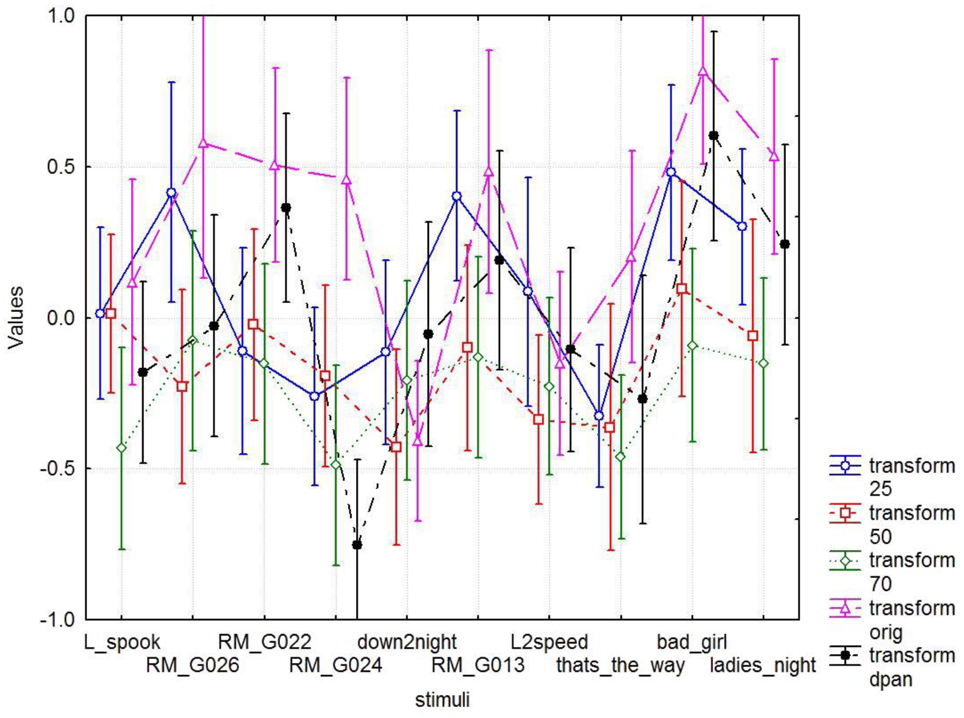


Figure 5-8: Mean values and confidence intervals (95%) of the Z-transformed groove ratings for transformation and melody, across participants.

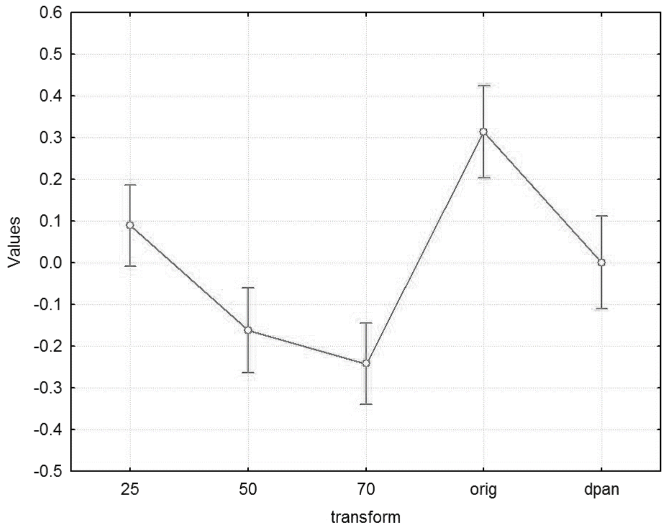


Figure 5-9: Mean values and confidence intervals (95%) of the Z-transformed groove ratings for each transformation, across participants and melodies.

5.5 Results and Discussion

The purpose of the study was to examine the effect of syncopation on the perception of groove independently of other expressive features of music performances. To this end, I designed algorithms for the generation of syncopation in a systematic and automatic way. The experiments focused on two different types of stimuli: simple monophonic piano melodies in experiments A and B, and more complex polyphonic rhythmic loops in experiment C.

We hypothesized that the syncopation transformations would result in higher groove ratings than the deadpan versions of the MEs with no syncopation, at least for moderate amount of syncopation. We predicted that a higher number of introduced syncopations would not have a proportional effect in the perception of groove and that the relation between syncopation and groove was complex— in the sense that it is not simply the amount of syncopation, but how it is expressed which causes the increase in the sensation of groove. In the simplest form this could follow an inverted U shape although an understanding of the relation between syncopation and groove requires a closer examination of the characteristics of individual instances of syncopation.

In Experiment A, all transformations generated syncopation by anticipating the last note of each phrase in the melodies at the same time that they introduced faster metrical levels. The transformations displaced notes from the slower metrical levels, quarter note or half-note levels in the 4/4 meter, to the faster eighth or sixteenth note levels generating a variety of combinations of note durations. However, this variety was not reflected in the groove ratings which were increased at similar levels for all the transformations.

In Experiment B, we verified that the increase in the groove ratings was in fact particular to the syncopation, and that other types of transformations tended to have weaker effects on groove. In addition, Experiment B examined whether the strength of the syncopation had an effect on groove. Our hypothesis was that a moderate number of syncopating notes that contradicted the meter only momentarily would increase the feeling of groove, while, in contrast, a high number of syncopating notes that constantly challenged the meter would not contribute to the groove perception. The hypothesis was tested by using two different strategies in generating syncopation. The *8A* and *8D* transformations syncopated certain notes while keeping most of the notes in place on the strong beats. In contrast, the *maximum syncopation* transformation displaced the entire melody and created a constant phase offset between the melody and the metronome sound. Both strategies resulted in higher groove ratings compared to the original versions, but less so for *maximum syncopation*.

Conversely, the *8A* and *8D* transformations did not differ in their respective groove ratings although they differ in the number of syncopating notes they generated. While *8D* generated only a single syncopating note at the strong beat at the end of each phrase, *8A* syncopated a substantial number of additional notes found on weak beats and that were simply delayed in *8D*. One possible explanation for this result is that the effect is due to the introduction of faster metrical levels and not specific to the syncopation. However, the two density transformations—which introduce the same faster metrical levels as the syncopation transformations without generating any syncopation—were rated significantly lower than the syncopation transformations.

Another explanation of the above results considers the nature of the syncopation in each transformation. In *8A* and *8D*, the 4/4 meter is first clearly established and a strong beat sensation is created at the half bar metrical level. Then, at the end of each phrase, a single syncopating event violates our strong expectation that the last note should occur on the pre-established beat. The syncopation is felt very strongly because of the well-established beat and the position of syncopation in relation to the structure of the melody.

The “extra” syncopation in the *8A* version that results from the anticipation of the notes articulated on the weak quarter note level, i.e. on the second and fourth quarter note positions, is felt less strongly. The weak quarter note level comprises a subdivision of the more salient half bar pulse, which means that our expectation about events on these positions is accordingly lower. This is in accordance with several metrics of syncopation (see chapter 3) as well as with the recent study of Song, Simpson, Harte, Pearce and Sandler (2013) that points out the importance for the felt syncopation of the location of the individual instances of syncopation in relation to the meter. At the same time their position in each phrase results in grouping them together with the previous strong beat creating a Short – Long rhythmic figure with the short duration on the beat. The same kind of rhythmic figure is found in the *8D* variation but with the long note on the beat, resulting in no syncopation. This rhythmic figure repeats at least three times in most phrases before the final strong syncopation arrives. Together with the metronome and the harmonic structure of the melodies, these repetitions may enforce the meter, rather than contradict it, in both versions. Groove may therefore be affected only by the syncopating notes that clearly contradict our metrical expectation at metrically strong positions at the beat level and not by the weaker syncopation at the subdivision of the beat.

On the other hand, in the *maximum syncopation*, syncopation comes about through the interaction between the metronome and the melody. The melody, if it was to be heard alone, contains no syncopation. In this case, the syncopation cannot be attributed to individual syncopating events but to the interaction of rhythmic layers during the entire duration of the music, resulting in an overall feeling of syncopation with no distinct moments.

However, if the listeners ignored the metronome sound they could have perceived the melodies as non-syncopating and identical to the original versions. Although this would explain the lower ratings compared to the other syncopating versions, it would raise new questions about the higher ratings compared to the original versions or to those of the two density transformations. At the same time, the metronome sound with an emphasized downbeat was heard for one bar alone before each melody. On this basis, it seems unlikely that the listeners would ignore it the moment the syncopating melodies began. Then, even if, in the course of hearing the melody, they shifted their perception to align the meter to it, the contradiction with the stable metronome would persist, thus evoking a constant feeling of syncopation together with a strong sense of a stable meter and beat.

Hence, the ratings show that a moderate amount of syncopation arising from notable salient instances of syncopation that underline the melodic boundaries contribute more to the perception of groove than a higher degree of syncopation that is uniformly distributed along the entire duration and does not relate to the melodic structure. Most noticeably, the weakening of the groove sensation when the amount of syncopation is increased is not due to a breakdown of the perceived meter, but it is particular to the relation between syncopation and groove.

The syncopation transformations have the secondary effect of increasing the number of different note durations in the melodies. This is a direct consequence of their design. Excluding the final notes in each melody, the original melodies contained only two durations, which can be thought of as long (half notes) and short (quarter notes). The syncopation transformations in both experiments, with the exception of the *maximum syncopation* transformation, enriched the rhythms by introducing a variety of shorter and in-between short and long durations. The density and the *maximum syncopation* transformations do not have this effect. Although they all shorten the durations of the notes in the original melodies, they produce simple Short – Long relations. The result of Experiment B, that the *maximum syncopation* is rated lower than the other two syncopated versions, suggests that a variety in note durations could have a contribution to the sensation of groove. The transformations of experiment B introduce the various note durations gradually with the 8D16A and 16D8A resulting in higher variety than the 8D8A and 16D16A, due to the fact that they mix the sixteenth and eighth note metrical subdivisions. However, this is not reflected in the groove ratings that would be expected to be higher for the 8D16A and 16D8A if a direct relation between the variety in note durations and groove perception existed. Furthermore, despite the density 1 transformation generating greater number of durations and faster metrical levels compared to maximal syncopation—although much less compared to the 8D or 8A—, maximal syncopation was rated higher than density 1. The relation between the diversity in note durations or the articulation of faster metrical levels in rhythms and groove perception therefore needs to be further explored as the results of the current study are inconclusive on this question.

In contrast to experiment A and B, experiment C did not show any statistically significant increase of the groove sensation when introducing syncopation to a deadpan non-syncopating loop. Moreover, introducing a higher amount of syncopation (50% or 70%) had a negative effect on groove ratings. Only the original syncopation of each loop had a positive and significant effect on groove. This raises the question of how the original syncopation is different from the automatic transformations. Having in mind that the de-syncopation algorithm is also an automatic transformation that is completely reversible, the question can be rephrased: are there certain syncopation transformations that create groove, such as the original syncopation of experiment C or the ones in experiments A and B, while others don't? Can one determine their characteristics in a systematic way? A more general question would be how could one define systematic rules for the application of syncopation transformations?

The experiments presented here cannot provide definite answers to these questions but they can serve as a starting point for further research and experimentation by forming a hypothesis for the above results. The most noticeable difference between the algorithms of experiments A or B and the one of experiment C is the stochastic nature of the later. Syncopation was introduced indiscriminately in various positions without considering the rhythmic, melodic or harmonic structure of each loop.

Furthermore, the polyphonic nature of the loops of experiment C implied the same randomness in the distribution of syncopation among the instruments. The simultaneous randomness in both the horizontal dimension (in time) and the vertical dimension (among instruments) of the loops could have weakened the metrical feel and brought about ambiguity in the perception of the beat. This ambiguity should increase with the number of random syncopation shifts introduced in agreement with the results that show that the 25% transformation did not affect negatively the sensation of groove but rather had a small positive effect.

In the original versions of the examples of experiment C, the syncopation was not uniformly distributed between the instruments: certain instruments might be heavily syncopated while others were not syncopated at all. In the example of Figure 5-6, the drum section has no syncopation and the bass line has only one syncopation per bar. In contrast, the guitar riff carries much of the syncopation of the loop (an equivalent to a 60% transformation). In a previous study (Sioros et al., 2012), we had observed a similar distribution of syncopation among the instruments in a number of songs taken from the RWC Music genre[33] dataset. I presented an example song of this analysis in, Figure 3-15, on page 72 of this thesis. These observations are also related to a more general assumption that a stable rhythmic layer is needed in order to establish a metrical framework against which the syncopation is felt.

The results of the three experiments are compatible with the recent study by Witek et al. (2014) in which a moderate degree of syncopation was preferred to a high degree of syncopation. However, the synthesized stimuli of the experiments presented here differ significantly from the ones in Witek et al. Witek used a number of real drum-breaks containing several timbres and each different example had a

[33] http://staff.aist.go.jp/m.goto/RWC-MDB/ (accessed: December 5, 2012)

different degree of syncopation. In our experiments syncopation was systematically generated by a computer algorithm resulting into different syncopated versions of the same music examples.

In the case of experiments A and B, the examples were monophonic piano melodies that originally contained no syncopation. The simple rhythmic structure of the original melodies allowed for applying deterministic algorithms to manipulate their rhythmic characteristics. The automatic transformations of the experiments affected the particular rhythmic properties they were designed to change exclusively, that is, syncopation and density. More importantly, this method allowed us to draw specific conclusions about how syncopation can be applied effectively in order to create groove.

In experiment C, we used already complex commercial loops that we automatically de-syncopated. In contrast to experiments A and B, we employed a stochastic algorithm to introduce syncopation, which had a strong effect on the perceived meter. In some cases the effect was more destructive, creating an ambiguous metrical sensation, while in other cases it was a simple shift of the downbeat. As mentioned above, we find pleasurable the music that makes us want to dance (Janata et al., 2012). Even though, syncopation is considered generally pleasurable (Keller & Schubert, 2011), the effect of the particular stochastic algorithm on the perception of meter might have also a strong effect on how pleasurable and natural the examples sounded, which in turn reflected in the groove ratings.

The three experiments above provide a systematic method for designing algorithms that can produce versions of melodies or short loops that contain faster and slower metrical levels as well as various degrees and diversity of note durations and different distributions of notes among those durations. It provides a method for directly comparing the different versions and examining them against listener ratings.

In conclusion, syncopation alone was sufficient to increase groove in simple melodies while removing the syncopation of the more complex polyphonic examples was enough to decrease the sensation of groove. Nevertheless, the systematic random generation of syncopation in the polyphonic examples did not increase groove. We confirmed that a moderate amount of syncopation is more effective in creating the sensation of groove than a very high amount. However, the amount of syncopation, as a scalar quantity, that characterized each melody was not sufficient to explain how syncopation increased the sensation of groove. To further explore this relation, an analysis of the individual instances of syncopation in the melodies was needed. This analysis showed that individual instances of syncopation that momentarily created rhythmic tension were more important in groove than an overall higher degree of syncopation. Our maximum syncopation condition in the simple melodies, even though it increased the syncopation did not cause the breakdown of the perceived meter despite constantly challenging it. In contrast, in the polyphonic examples, excessive syncopation caused a negative effect on the sensation of groove, most probably due to the weakening of the metrical feel. This result further supports the idea that syncopation should not be treated as a scalar quantity that overall characterizes a music excerpt, but an analysis of the particular instances of syncopation is needed in order to understand its effect to higher level rhythmic properties such as groove. Additionally, a richer rhythmic structure in note durations and metrical levels was considered a possible explanation for the relation between syncopation and groove as syncopation is inherently related to such structures. However, no definite conclusion was possible based on the results of the experiments.

6 GENERATIVE APPLICATIONS

The syncopation transformations presented in chapter 4 are by nature generative and therefore have important applications in computer assisted composition and automatic music generation. In this chapter, I will present three Max4Live devices that make use of the syncopation transformations in order to generate variations of a user input music signal that vary in their degree of syncopation. The three devices have similar functionality adapted for different input types. They can be thought of as three different flavors of a single system that 1) removes all syncopation in a rhythmic stream, and 2) introduces new syncopation that differs from the original by applying the recursive algorithm I described in section 4.4.2. The re-syncopation algorithm is also recursive and is described in detail in this chapter. The system essentially takes an input pattern and generates an entire branch of a syncopation tree that begins at the root pattern, passes by the input pattern and continues to an end pattern of maximum syncopation.

The most important feature of the devices is the syncopation slider that controls the amount of syncopation in the output. The slider corresponds to the generated branch, with the lowest position of the slider corresponding to the root pattern and gradually passing by all the patterns in the branch reaching at the top most position the maximum syncopation.

Two of the Max4Live devices are MIDI devises and the other one is audio. The first MIDI device, “g.syncopalooza.MCLIP”, takes the currently playing MIDI clip in a channel and automatically transforms it and plays back the transformed version. The other two devices, “g.syncopalooza” (MIDI) and “loopalooza” (audio), work as rhythmically enhanced looper pedals. They record a duration of a bar of the real time feed in their input, either the MIDI notes or live audio, and play it back as a loop while automatically transforming the syncopation. The musician can control in real time the transformations through the syncopation slider generating a MIDI or audio output that ranges from no syncopation (the root of the corresponding syncopation tree) to maximum syncopation (the end of the branch).

Before applying the syncopation transformations, the input streams are converted to binary patterns. For MIDI input, whether it is a MIDI clip or a live stream of MIDI notes, the binary pattern is based on a quantized version of the MIDI note On positions. For the audio input, an onset detection algorithm segments the audio into sound events and converts it into a binary pattern. The Max4Live global transport provides all necessary information, that is, time signature, tempo and current playback position, needed in the metrical template for the quantization of the patterns, for recording, transforming and playing them back.

The devices make use of an additional algorithm that generates and transforms dynamic accents in order to enhance the metrical or syncopation feel. It assigns dynamic accents to the onsets on the beat (thus enhancing the metrical feel) and then shifts them to offbeat positions (thus enhancing the syncopation feel). Although the algorithms presented in section 4.6 could in principle be used in the Max4Live devices in order to manipulate the original accents in the input signal, I preferred to develop a simpler and easier to implement algorithm that does not require any accents in the input but instead it generates them. It gives greater flexibility on the user input; for example, it allows for a binary pattern to be directly input on the user interface as if it was a step sequencer (see Figure 6-7 on page 160) without accents.

In section 6.1 I give a detailed description of the common syncopation algorithm applied on the binary patterns in all three devices. In section 6.2, I describe the two MIDI devices, their particular functionalities and user interface, and in section 6.3 the audio device.

6.1 TRANSFORMATION ALGORITHM

The syncopation transformations are applied in two steps (Figure 6-1). First the input pattern is de-syncopated according to the recursive algorithm described in 4.4.2, which results in a root pattern and an array of transformation vectors of the syncopations found while de-syncopating. The second step consists in generating a complete branch of the tree that begins at this root. The branch can include the input pattern or it can be a completely new branch. In order to include the input pattern, the array resulting from the de-syncopation loop is reversed and used as input for the syncopation loop. The array needs to be extended with default transformations that will generate all the patterns from the input until the end pattern of maximum syncopation. For a completely new branch that does not pass from the input pattern, the result of the de-syncopation loop is ignored and a new original transformation vector with some default transformations is used. Essentially, the syncopation loop fills the transformation array with vectors by recursively applying on the root pattern the appropriate default transformations and/or the transformations found during the de-syncopation. It results in a series of binary patterns, each pattern differing from the previous by a single syncopation shift.

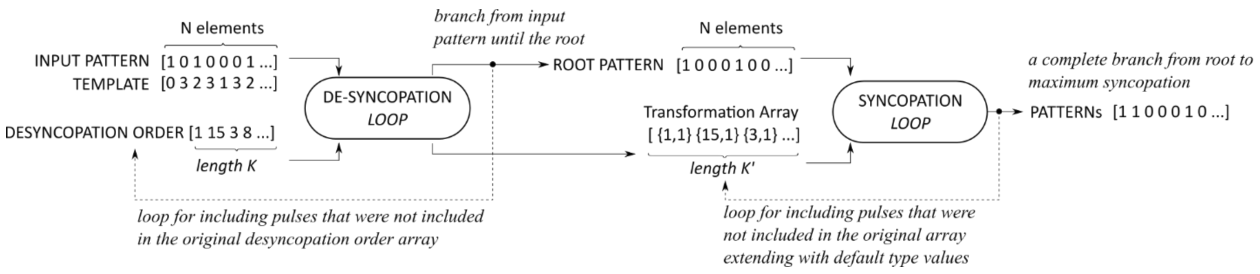

N elements
INPUT PATTERN [1 0 1 0 0 0 1 ...]
TEMPLATE [0 3 2 3 1 3 2 ...]
DESYNCOPATION ORDER [1 15 3 8 ...]
length K
DE-SYNCOPATION LOOP
branch from input pattern until the root
ROOT PATTERN [1 0 0 0 1 0 0 ...]
N elements
Transformation Array [{1,1} {15,1} {3,1} ...]
length K'
SYNCOPATION LOOP
a complete branch from root to maximum syncopation
PATTERNs [1 1 0 0 0 1 0 ...]
loop for including pulses that were not included in the original desyncopation order array
loop for including pulses that were not included in the original array extending with default type values

Figure 6-1: Overview of the generation of a branch of syncopation transformations that passes through a given input rhythmic pattern.

Figure 6-1 shows an overview of the entire process which is an extended version of the de-syncopation process presented in section 4.4.2. The de-syncopation loop together with its inputs and outputs is already described in 4.4.2. In the following section, I will describe in detail the second step in the process, the syncopation loop.

6.1.1 Generating a syncopation branch

In order to automatically syncopate a pattern, one can apply repeatedly a series of syncopation transformations until no onsets can be further shifted. If the pattern is a root pattern then a complete branch of a syncopation tree is generated. Which branch is produced depends on the transformations applied and on the order they are applied. I will first demonstrate the main features of the process through the examples of Figure 6-2.

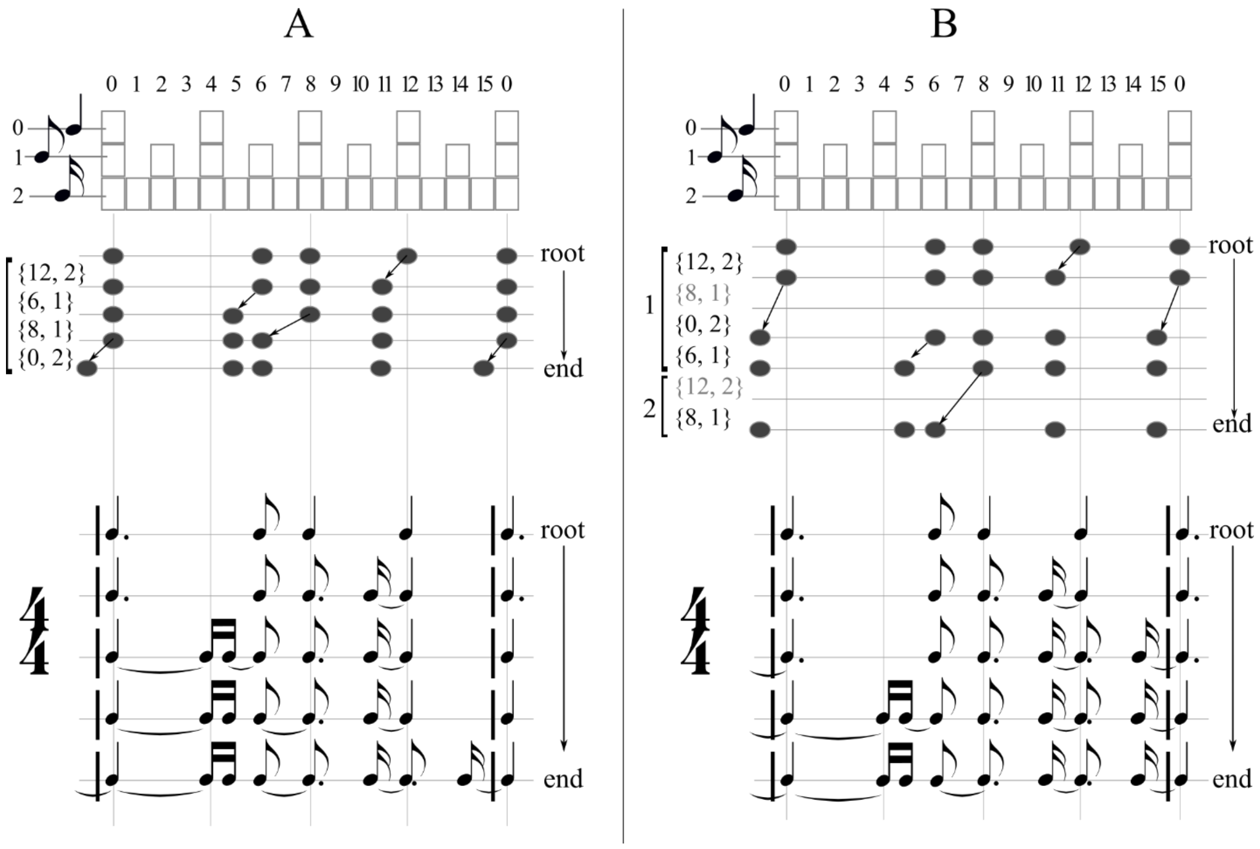


Figure 6-2: Two examples of the generation of a complete branches that begin at the same root pattern and reach the same end pattern. The difference in the two branches is in the order in which the transformations are applied. In B, the grayed out vectors represent not valid transformations. The black vectors constitute the array of the resulting branch. The same branches are shown in music notation under their binary representations.

The two examples create two different branches. However, the two branches have several characteristics in common: the same root and end patterns as well as the same number of intermediate patterns (5 in total). They differ only in two of the intermediate patterns. In example A, a series of transformations represented by the array of vectors at the left is applied. The array is applied only once, since with the last transformation the end pattern is reached. In example B, the same transformations are combined in a different order resulting in a different array: the {6, 1} transformation is put at the end of the array. The {8, 1} transformation comes before the {6, 1} and is initially blocked. Therefore when the end of the array is reached after the {6, 1} vector the branch has not yet reached its end since pulse 8 can now be syncopated. In this case, the array needs to be applied a second time. The process starts over and the same array is applied again. This time, the {12, 2} transformation is not valid since there is no onset in pulse 12, but the {8, 1} transformation is now “un-blocked”. With this last transformation the end of the branch is reached and the process stops.

The following pseudo code illustrates the basic steps in the syncopation algorithm:

```
Recursive_Syncopation (InPattern, Template, TransformationArray)
  MAXLEVEL = max(Template) //Max metrical level in the template
  Pattern = InPattern
  PATERN_LIST = {empty}
  ARRAY_OF_VECTORS = {empty}
  Repeat
    For each Vector{pos, type} in TransformationArray
      If (Pattern[pos]==TRUE AND Template[pos]<MAXLEVEL)
        If (Syncopation (Pattern, Template, Pattern, Vector)) //input and out patterns
                                                                are the same
          Append (PATERN_LIST, Pattern)
          Append (ARRAY_OF_VECTORS, OutVector)
        End
      End
    End
  Until no onsets can be shifted
  Return PATERN_LIST, ARRAY_OF_VECTORS
End
```

At the end of the algorithm shown above, besides the generated patterns an array of transformations is output (`ARRAY_OF_VECTORS`). This array can differ from the input array (`TransformationArray`). As the pattern is scanned according to the input vectors, some transformations might be blocked or skipped when an onset is not found, thus, resulting in a different final array. The output array contains the actual transformations that were performed.

The algorithm is implemented in the Max/MSP patch [syncopate.maxpat] that can be found in the accompanying CD.

In the examples of Figure 6-2, the arrays were chosen so that when applied the end branch is reached. This was possible because the vectors in the array contained all the pulses where onsets are found. For this particular branch, it was enough to include all onsets found in the root pattern. In general, the transformation array needs to include all onsets in any pattern produced by the transformations, not only in the root pattern, otherwise the end of the branch cannot be reached.

In the above examples the arrays of transformation vectors were constructed manually so that they generate the exact branches. However, arrays that generate complete branches can be constructed automatically. In the simplest case, they can be generic and filled out by vectors with default values in a default order. For example, a default order of transformations can be generated by starting from the pulses in faster metrical levels and continue to slower ones until all pulses are included (the fastest metrical level can be ignored).

For each pulse, a type of transformation should also be assigned. Several options for default values are available. The simplest is a constant value for all pulses. The type values of the transformations can also be chosen according to a certain rule. For example, all notes could be shifted to the preceding sixteenth note. That would be a type 2 for the quarter-note positions and a type 1 for the eight-note positions. After applying the possible transformations to all the pulses in the array, the process repeats, applying the same array again and again on the output pattern until no onset can be shifted.

The above algorithm results in complete branches when applied on a root pattern. In principle, it can be applied on any pattern in order to create a branch that begins at the particular pattern and reaches the end. With this in mind, one can combine the de-syncopation process of section 4.4.2 with the recursive syncopation algorithm above in order to create a branch that passes through a certain user input pattern. In Figure 6-3, an example of how such an array of transformations can be constructed is shown. The part of the array between the user input pattern and the root is constructed by reversing the array that result of de-syncopating the input pattern. The rest of the array, from the user input until the end of the branch, is constructed by recursively applying a default array of transformations, omitting any transformation that is blocked or otherwise impossible. The complete branch consists of the concatenation of the two parts.

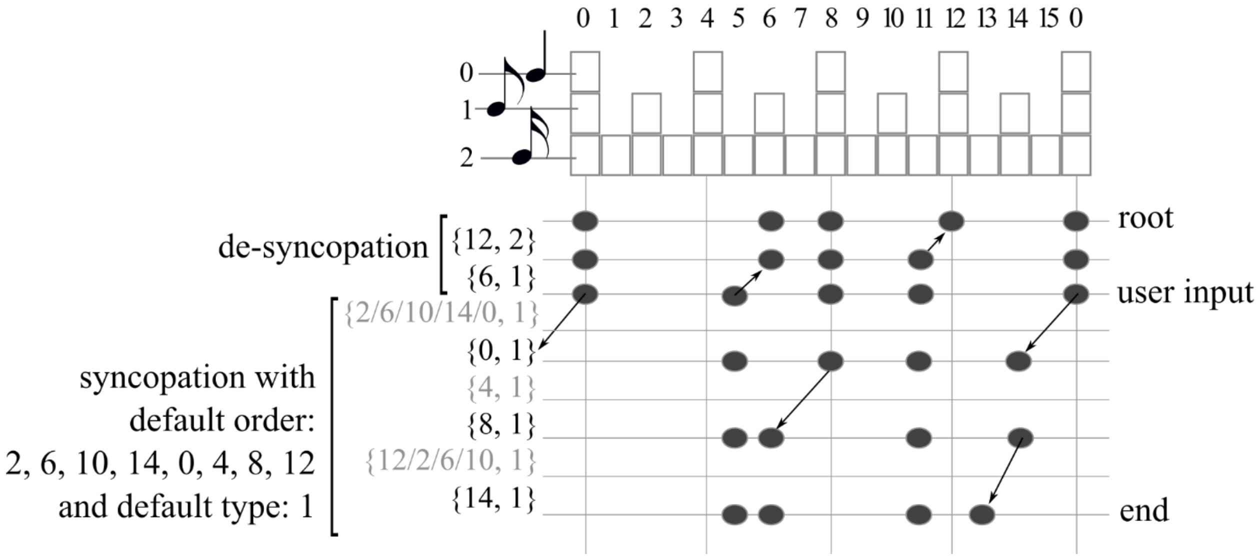


Figure 6-3: Constructing an array of transformations that goes from root to end and passes through a certain user defined pattern. Omitting the grayed out vectors at the left and including only the black ones gives the array of vectors of the complete branch.

An implementation of the combination of the de-syncopation process and the generation of a complete branch can be found in the [de-Re-syncopation.maxpat] in the accompanying CD.

The above process of extending the transformation array is shown in Figure 6-1 with a dashed line. The syncopation loop takes as an input the array created during the de-syncopation process. Applying this array on the root generates all the intermediate patterns until the user input pattern. Then, the array is extended with default vectors according to which pulses carry onsets in the output of each step of the syncopation process.

6.1.2 Transforming Dynamic Accents

In section 4.6, I presented an extension of the binary transformations that introduces syncopation by shifting accents when the onsets themselves cannot be shifted. The algorithm I present in this section differs from the one in section 4.6; while the algorithm of section 4.6 is applied to the amplitude values already existing in a pattern, the algorithm presented here automatically generates accents in binary patterns prior to syncopate by shifting them in off beat positions. Although the transformations of section 4.6 could be applied on the generated accents, the transformations described in this section are specifically designed for the automatically generated accents and are much simpler to implement as a computer algorithm.

In the following subsections, I describe how the accents are assigned to the onsets and the algorithm that shifts the accents in weaker metrical position generating syncopation.

6.1.2.1 Generating Accents

The metrical feel is enhanced when phenomenal accents coincide with the metrical accents, i.e. when changes in the loudness, pitch etc. coincide with the alternation of strong and weak pulses characteristic of the meter. Based on this principle, accents are assigned to the onsets of the root pattern according to the strength of their metrical positions.

The following mathematical formula is used to calculate the amplitudes of onsets:

$$A(i) = C^{L(p_i)} \qquad 6.1$$

where A(*i*) (from 0 to 1) is the amplitude of event *i*, C is a parameter that controls the "contrast" between strong and weak metrical positions, p_i is the pulse that event *i* belongs to (in the root pattern) and $L(p_i)$ is the metrical level of that position. The parameter *C* ranges between 0, where only the slowest metrical level survives, and 1, where all metrical levels receive the highest amplitude. In Figure 6-4, the amplitudes are shown calculated for a 4/4 template and two intermediate values of the parameter *C*.

I first used equation 6.1 in the generative algorithm of kin.rhythmicator (Sioros & Guedes, 2011a, 2011b). Kin.rhythmicator is a stochastic rhythm generator that triggers notes in certain metrical positions in real time according to probabilities and amplitudes that follow the form of equation 6.1 (see Figure 6-4). Since notes are more probable on stronger metrical positions, the sense of meter in the generated rhythm emerges.

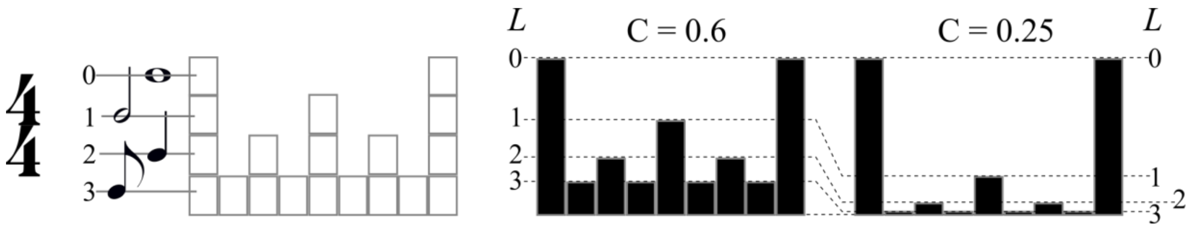


Figure 6-4: Amplitudes are calculated according to equation 6.1 for two different values of the parameter C.

The amplitudes are assigned to the onsets themselves and not to the pulses, so that when the onsets are shifted to different positions in the various syncopated versions, they still carry their amplitudes (Figure 6-5). For example, an onset found at the beat or quarter note level in the root pattern will still be a loud onset with a high amplitude value even when it is shifted to the faster sixteenth-note metrical level. In that way, the feeling of syncopation is intensified. The syncopating event is distinct from other events in fast metrical levels that belong in fast metrical levels in the root pattern and that receive accordingly lower amplitudes.

In contrast, if the onsets were assigned different amplitudes in each syncopating version according to their current metrical position, the syncopation would weaken without necessary enhancing the metrical feel. The onsets will still be syncopating but their amplitudes would receive a low amplitude according to the metrical strength of their syncopating position, thus weakening the syncopation feel.

6.1.2.2 Transforming accents

The algorithm is motivated by the way syncopation is produced in the *kin.rhythmicator* software (Sioros & Guedes, 2011a, 2011b). In the kin.rhythmicator, the probabilities follow the form of Figure 6-4, so that the stronger the metrical position the more probable it is to trigger a note. Syncopation is introduced in the generated rhythm by “anticipating” the entire metrical template by one pulse, so that notes are triggered according to the probability assigned to the immediately following pulse. Such temporary shifts of the metrical template, produce a rhythm that belongs in a shifted meter compared to the already established meter, thus evoking the feeling of syncopation.

The same principle is followed here in order to transform the generated accents. Imagine that an onset A in a strong pulse cannot be shifted because another onset B in the previous weak pulse is blocking it. Then, instead of moving the onset, we can shift the template of metrical accents by one pulse so that the onsets are assigned new amplitudes that syncopate (see Figure 6-5, pulses 3 & 4 for an example). Onset B takes the higher amplitude of onset A while onset A takes a lower amplitude that corresponds to the following weak pulse (irrelevant of whether an onset exists there or not). As a result, the onset in the weak position is now accented relatively to the following attenuated onset in the strong position giving rise to the syncopation feel. This is a direct consequence of how the metrical template is constructed, with the strong and weak pulses are always alternating. Strong pulses, which belong to slow metrical levels, are always followed by weak pulses, which belong to faster metrical levels. Shifting the metrical template to the left by one pulse always results in syncopating pairs of amplitudes. The shift of the template is “local” for the two pulses in question while the rest of the onsets are assigned the normal metrical accents.

The template shift is employed as a secondary mechanism that is needed only when the onsets themselves cannot be shifted. Otherwise, the onsets carry with them the dynamic accents, so that onsets and accents are anticipated together (Figure 6-5, pulse 2).

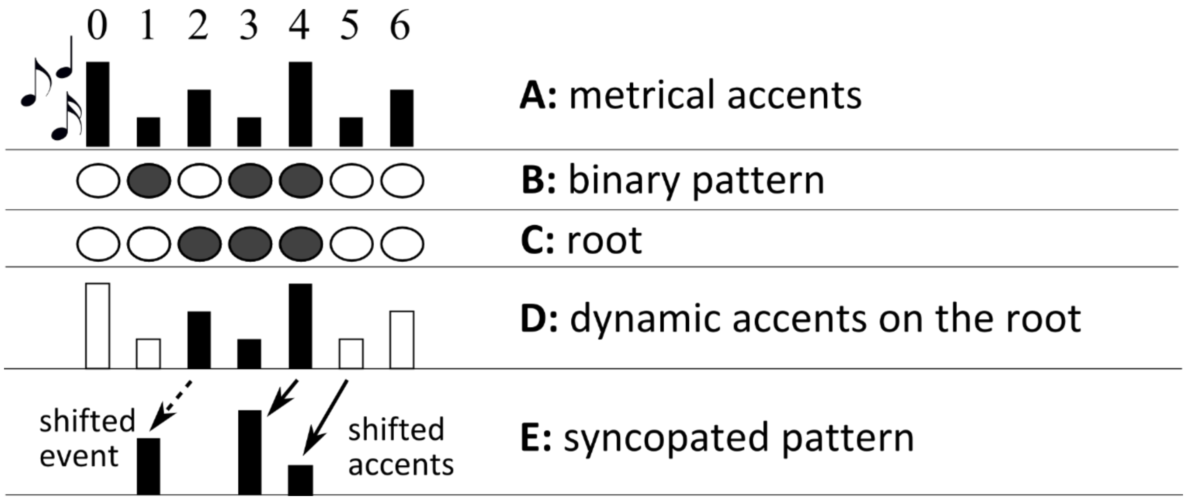


Figure 6-5: Example illustrating the process of assigning and manipulating dynamic accents.

The example in Figure 6-5 illustrates how dynamic accents are used. Metrical accents are calculated according to the metrical template (A). The binary pattern (B) is first de-syncopated (C) and then amplitudes are assigned to its onsets (D). Finally, syncopation is generated in two ways (E): 1) by shifting the onset in pulse 2 that also carries the accent, and 2) by shifting the metrical accents of pulses 3 and 4 instead of the onsets themselves.

6.2 MAX4LIVE MIDI DEVICES: G.SYNCOPALOOZA

I implemented the algorithms described above in two Max4Live MIDI devices that manipulate effectively and in real time the syncopation in binary patterns. The [g.syncopalooza.MCLIP] device manipulates the syncopation in MIDI clips. The [g.syncopalooza] transforms single bar MIDI loops that it records in real time through its input. They systematically remove the existing syncopation and create new syncopation. The devices construct automatically a metrical template according to section 2.3.1 that corresponds to the time signature and tempo settings of the Ableton Live Set and that is aligned to the Live transport. The template defines the slowest and fastest metrical subdivisions, a quantization grid, and the number of pulses or steps in the pattern. All transformations are then applied with regard to this template.

The most important element on both MIDI device is the syncopation slider (Figure 6-6).The syncopation slider controls which pattern in the generated branch is selected for playback. At the lower end of the slider, the root pattern is found. As one raises the slider, syncopation is added. The original midi clip is marked as a red square on the slider. Raising the slider further leads to further syncopation until the end of the branch is reached at the upper end of the slider. Each step of the slider corresponds to a single shift of an onset. The type of the default transformations needed to syncopate the midi notes beyond the original syncopation, as in Figure 6-3, can be set on the devices through a number box named “style”. The default order is chosen according to the indispensability values of Barlow’s algorithm (Barlow & Lohner, 1987; Barlow, 1987). A different branch can be generated by de-activating the red button named “orig” right above the slider. In this case, the original syncopation is ignored and a branch is generated for the same root pattern based on the default order and the user input type value. The default order can be randomized in the advanced controls of the devices. In this way, a random branch of the same tree can be generated.

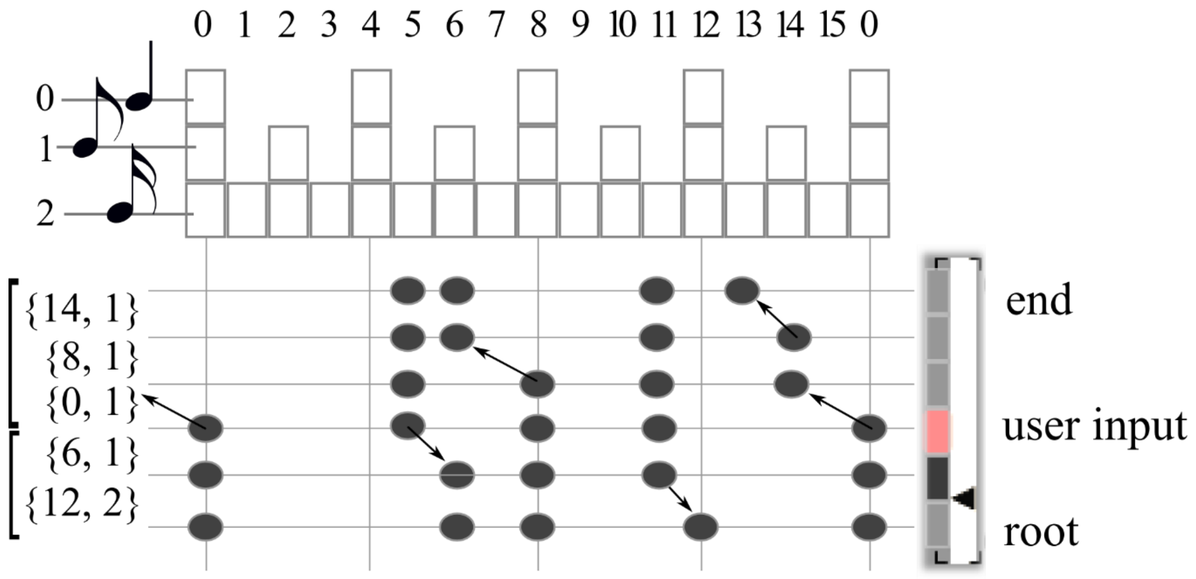
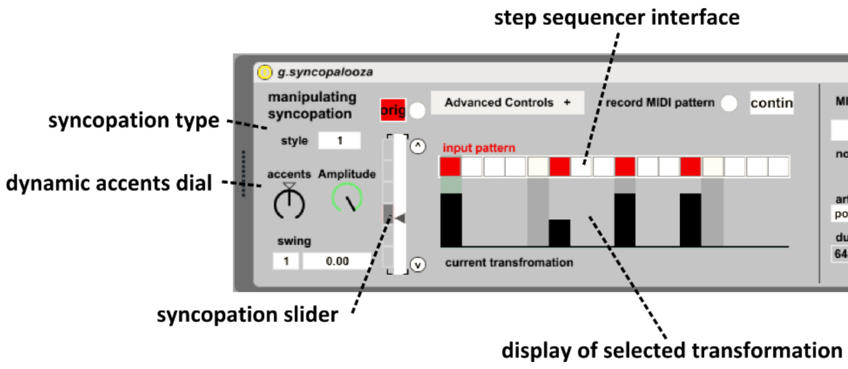


Figure 6-6: The syncopation slider on the Max4Live devices controls which variation is being currently played back.

For the onsets that cannot be shifted, their dynamic accents are shifted as described in section 6.1.2.2. A dial at the left side of the device controls the dynamic accents applied. When the dial is turned to the left, no accents are applied and all onsets have maximum velocity. When the dial is turned fully right, only the slowest metrical level survives. In the center position, the velocity of the events is relative to the metrical strength of their positions.

The user interfaces of the two devices are very similar as shown in Figure 6-8 and Figure 6-7. Besides the syncopation slider, the central area of the devices is used to display the input pattern with red squares and the current transformation directly below as black bars. The height of each bar is relative to the MIDI velocity assigned to the MIDI note events. Grey bars represent the “beat”, i.e. the pulses that belong to the slowest metrical level of the template.

The g.syncopalooza device (Figure 6-7) transforms a loop of a one bar length. The loop can be created either by recording the pattern in its MIDI input[34] or using the step sequencer interface of the device. When recording the input pattern all MIDI note information is lost and only a binary pattern is stored. After recording the pattern the user can edit it normally in the step sequencer interface. The user chooses a single MIDI note number for all onsets in the pattern on the right side of the device. The transformations are applied automatically as the pattern is changed and playback continues always uninterrupted.



style 1
swing 1 0.00
MIDI thru
note E3
articulation porta...
duration 64n

Figure 6-7: User Interface of the g.Syncopalooza Max4Live device for transforming a live MIDI input.

In the case of the g.syncopalooza.MCLIP device, when a midi clip is playing in the same track as the device, its contents are automatically "loaded" in the Max4Live device. The midi notes are quantized and grouped together according to which pulse they belong. Since the onset positions are quantized to the metrical grid, the output is always quantized even for the original syncopation. Chords are treated as single events of a binary pattern so that all notes of a chord are displaced together. In the center of the device, the display area presents a rough representation of the midi contents as a binary pattern.

[34] The device records and processes the input stream as separate single bar loops. This introduces a one bar delay between the input and the output. Such a delay is commonly found in audio loopers.

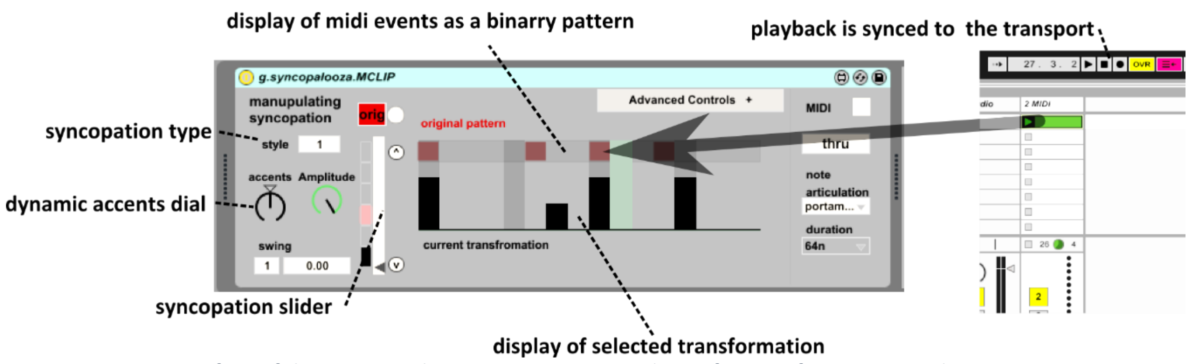


Figure 6-8: User interface of the g.Syncopalooza.MCLIP Max4Live device for transforming MIDI clips.

6.3 Audio device: Loopalooza

The loopalooza Max4Live audio device (Cocharro et al., 2014) is similar to the g.syncopalooza MIDI device described above with the difference that it works with live audio instead of a live stream of MIDI notes. The audio input is first segmented according to an onset detection algorithm (see Figure 6-9 for an overview of the system). The detected onsets are treated in the same way as the MIDI onsets in the g.syncopalooza device described above. A binary pattern is constructed which is transformed according to the transformations of section 6.1. Finally, the audio is reconstructed with the onsets shifted from their original positions and it is output to a Live audio channel.

Figure 6-10 illustrates the three stages of the method. The top panel shows the analysis, the middle the transformation, and the bottom the reconstruction. In the original audio, the inter-onset intervals (IOIs) of the audio segments have the same duration as the segments. The displacement of the onsets after quantizing and applying the syncopation transformations changes the IOIs resulting in potential silent gaps or overlapping segments. Figure 6-10 shows an example with a silent gap R between segments B and C and with two overlapping segments C and D. The reconstruction of the events in the audio output aims at generating a smooth transition between the audio segments. The application provides two audio reconstruction methods, a time scaling algorithm and a resampling technique. The details of the analysis and the reconstruction methods can be found in the published paper (Cocharro et al., 2014).

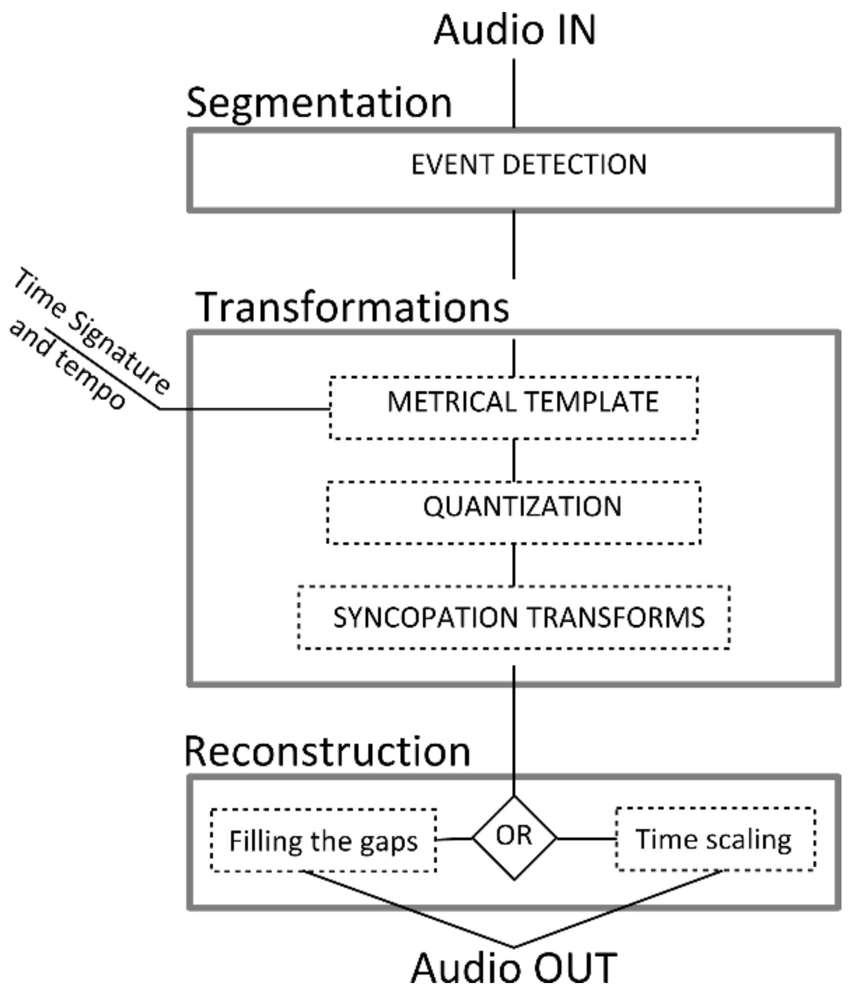


Figure 6-9: Overview of the stages that comprise the Loopalooza audio Max4Live device.

The continuous audio stream is constantly stored in a cyclic buffer in the device. However, the system only processes the input as single bar loops. Each bar is processed, i.e. analyzed, transformed and reconstructed, separately as a loop after it is recorded in the buffer. This introduces a one bar delay between the input and the transformed output. Since the transformations are computed in real-time and the reconstruction takes place on the fly, there is no additional delay. Changes to parameters via the interface take effect immediately.

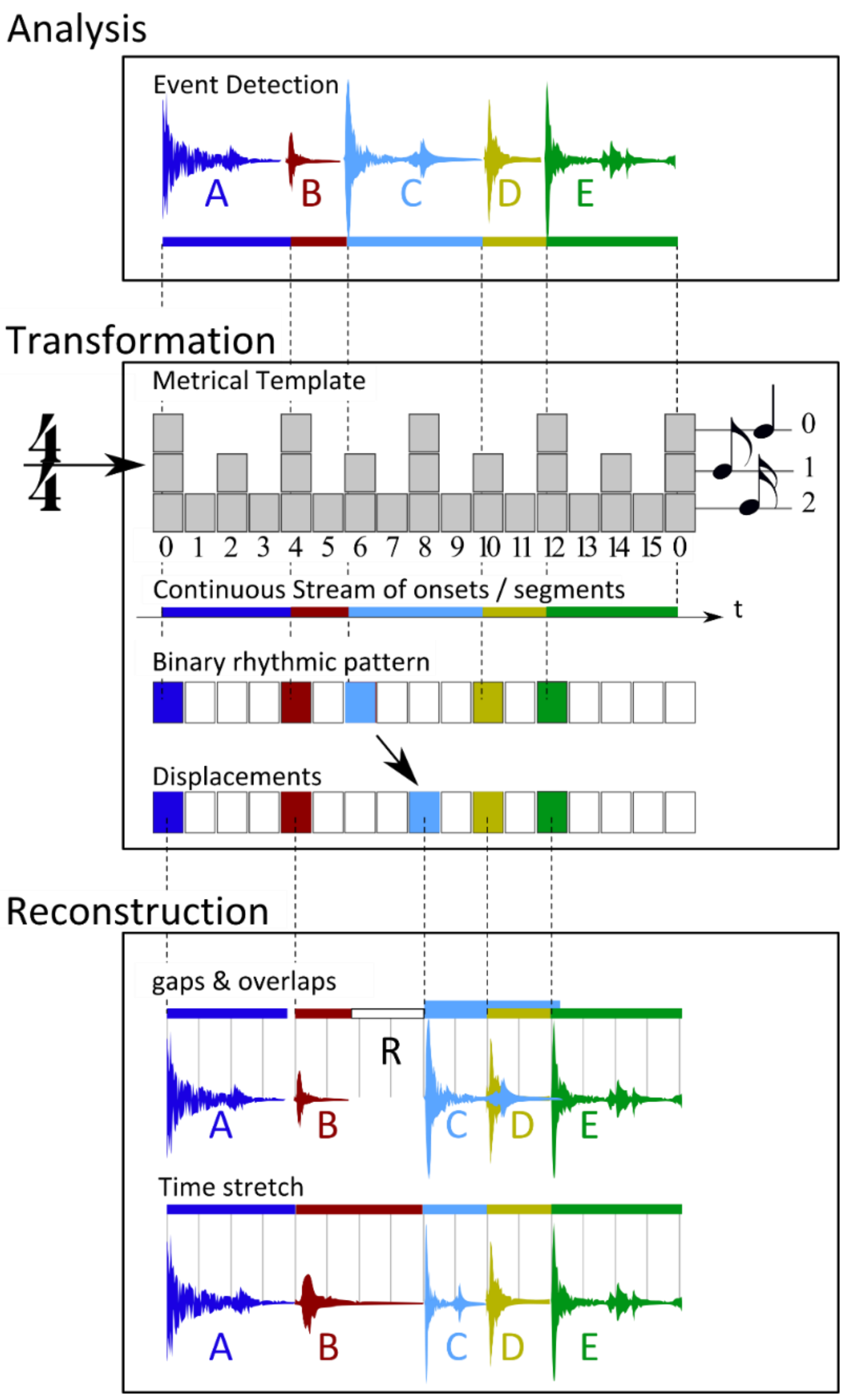


Figure 6-10: Example of rhythm transformation removing syncopation from the audio loop. Analysis: the audio input is analyzed for onsets and sliced into short segments (shown in different colors: A, B, C, D and E) forming a rhythmic pattern. Transformation: The onsets positions are quantized into a binary pattern according to the metrical template. The onset of event C is de-syncopated by displacing it from pulse 6 to pulse 8. Reconstruction: The new inter-onset interval durations can result in silent gaps (R between B—C) or overlapping segments (C—D). In this example, the selected transformation is reconstructed using time scaling.

The user interface of the audio Max4Live device is shown in Figure 6-11. Loopalooza shares most of the controls of the g.syncopalooza MIDI devices. Their main difference is that the loopalooza device is loaded in an audio channel and process a real time audio stream instead of MIDI information. Pulses containing onsets are marked as red bars. At the bottom, black bars indicate the displaced onsets. In addition two playback indicators (dark green bars) show the current playback position and which event of the original audio loop is being played back. The user can stop the recording of the input stream by de-activating the “recording” button on the right side of the device; in which case, the output continues using the last recorded bar as a loop.

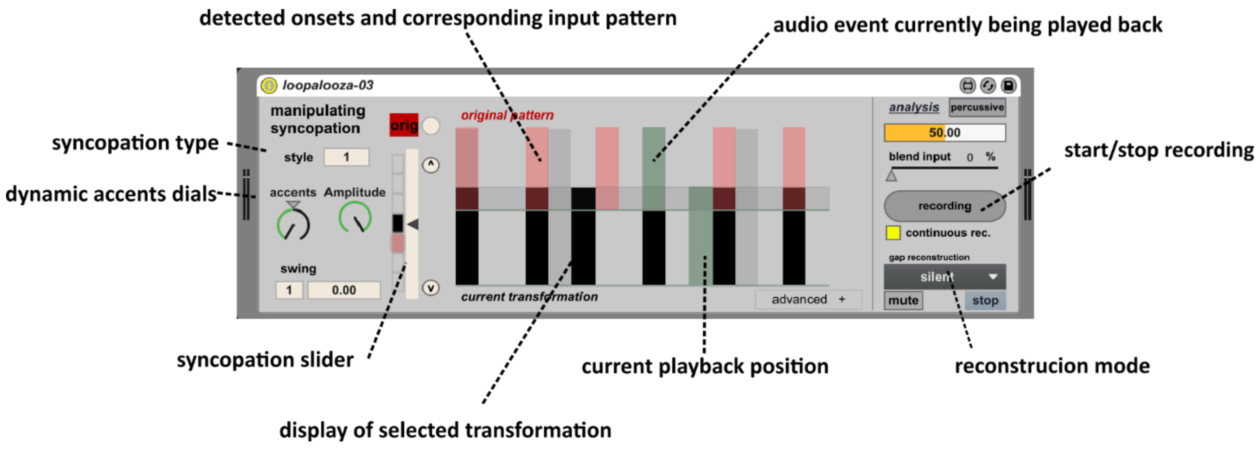


Figure 6-11: The Loopalooza Max4Live audio device user interface.

The sensitivity of the onset detection can be adjusted by controlling the “onset threshold” value found on the upper right corner of the device. This parameter controls the sensitivity of the onset detection. It affects the number of detected events in the input pattern. Hence, different rhythmic patterns and transformations can be generated with the same audio input. Loopalooza works with percussive sounds, like in drum loops, as well as melodic instruments or complex musical mixtures.

6.4 Discussion

In this chapter I presented three Max4Live devices that make use of the syncopation model in order to manipulate the rhythm of a MIDI or audio input. They all convert the input into a binary pattern which they then transform and playback. In the MIDI devices the conversion is straightforward since the timing of the MIDI events is known. In the audio device however, the conversion is preceded by an onset detection algorithm. The users have control over the onset detection sensitivity and consequently over the number of detected onsets.

Often interesting results are obtained using a lower sensitivity for the onset detection. In this way, the onsets are under-detected and the quieter events are overlooked detecting only the most prominent events. As discussed in section 4.5, syncopation is found more often in patterns with a moderate density of events per bar. Too many events result in a large number of “blocked” shifts and hence a weaker syncopation feel that comes about mainly from dynamic accents. On the other hand, if the density of events is very low there might be not enough events to shift for a sufficient number of variations. In a sense, when manipulating the syncopation of rhythmic patterns, only the most salient events are relevant. Setting a high sensitivity and detecting a larger number of onsets can be counter-productive with respect to the syncopation transformations.

The close relation of syncopation and the density of events is not a mere theoretical consequence of the nature of the transformations. A similar relation was also observed in the more intuitive variations created by professional musicians when they were asked to increase the groove in simple piano melodies in the listener experiment mentioned in section 5.1(Madison & Sioros, 2014). They introduced syncopation at the same time that they increased the density of events creating in this way richer rhythmic structures and effectively conveying groove to the listeners. However, in the experiments in sections 5.2 and 5.3 (Sioros, Madison, et al., 2013), the density of events alone was not as successful in increasing the sensation of groove as was syncopation. Although I do not know the intensions of the musicians, it is possible that they increased the density of notes, not to directly increase groove, but rather in order to provide with the necessary flexibility to introduce syncopation more effectively, for example syncopating in particular metrical positions where no notes existed in the original deadpan versions and thus enriching the rhythmic structure.

In MIDI, no onset detection algorithm is needed, since the information of the onsets of the MIDI notes is directly accessible in the input stream. However, the transformations are more effective when applied on binary patterns of a moderate density of events irrelevant of whether the original signal is MIDI or audio. An effective mechanism for manipulating the density of events would be beneficial in any system that generates automatic variations based on syncopation. A similar approach to the onset detection in audio can be employed in symbolic representations of music signals as well. For example, MIDI notes with low velocity values could be grouped together with the preceding loud notes instead of being treated as separate onsets, thus decreasing the density in the binary pattern without discarding notes from the input stream.

The transformation algorithm that is at the core of the devices is based on three steps: 1) the input binary pattern is de-syncopated and the root is obtained, 2) dynamic accents that enhance the meter are applied and 3) the original or new syncopation is introduced to the root. The approach can be generalized, so that in second step, one can introduce any other transformation that does not produce syncopation. The non-syncopating transformation would generate another root which will give rise to another syncopation tree. An example of such transformation is density transformations. In the following paragraphs, I will present some example algorithms for manipulating the density of root patterns that I believe are worth exploring.

The first example method for changing the number of events in a root pattern makes use of Barlow's (1987) indispensability algorithm. Onsets are added or removed in positions according to their indispensability index. Onsets are added first in the metrical positions that are most characteristic for the specific meter thus strengthening the metrical feel. Similarly, onsets are removed first from the positions least characteristic of the meter. The indispensability index orders the metrical positions according to the metrical level they belong to, so that onsets are added first in the slow levels and removed first from the fast levels. In this way, no syncopation can be introduced in a root pattern. The Barlow algorithm is generic and ignores the rhythmic structure of the input pattern.

In contrast, other methods can manipulate the density of onsets with respect to the characteristics of the original pattern. For example, one can begin adding onsets between existing events by dividing the existing IOIs. Starting from the longest IOI and moving to shorter ones will result in gradually including faster and faster metrical levels.

Another method moves the focus on the syncopation itself and the number of the possible transformations of the original pattern. In this case, one can add events according to two main criteria: 1) they should avoid blocking the transformations of existing onsets, and 2) each added onsets should be placed so that it can be shifted in as many different positons as possible, ensuring in this way, a greater number of syncopation transformations.

Finally, the focus could be placed on the musical style. Onsets could be added according to the analysis of "prototype" patterns of a certain style. As I discussed in section 4.7, de-syncopating a number of patterns in a certain style can yield arrays that are characteristic of that style. One can therefore add onsets in metrical positions according to the order of the positions included in such style arrays.

Combinations of the above methods are probably needed in most music applications. For instance, one could combine the style arrays with dividing the IOIs of the input pattern: onsets are first added in the positions found in some transformation arrays and then new onsets are added by dividing the existing IOIs.

The above methods can in principle be reversed in order to remove existing onsets and decrease the density in a root. Generally, onsets should be removed only when they are not directly preceded by other onsets in faster metrical levels.

Filling up the pulses with onsets blocks possible transformations. When we work with symbolic data such as MIDI, we can resort to dynamic accent transformations to generate syncopation. A similar approach could be implemented in the audio domain. First, the perceived loudness of each detected event or segment should be estimated. Then, the accents can be transformed by applying the proper attenuation. Metrical accents can be generated as in section 6.1.2.1.

7 CONCLUSION

In this study, I developed a coherent and systematic model of syncopation for the purposes of rhythm generation and analysis. Through a revision of literature in the music theory and cognition, I established the notion that syncopation is a feeling that arises from the interaction between a rhythmic pattern and the perceived meter in the listener's mind. Moreover, the literature review led me to the assumption that syncopation can be modeled as a transformation of a non-syncopating pattern. Based on my assumptions, I formalized a limited set of elementary transformations that generate or eliminate the syncopation in a rhythmic pattern given the metrical context in which it is heard. In the process of formalizing syncopation, I created important theoretical concepts, such as the transformation vectors and arrays, the duration of the beat and the syncopation tree. Together with the transformations themselves, these constructs can be used to create algorithms and software for the analysis and generation of rhythms. The algorithms and software are made available in the form of Max externals and patches. The corresponding C++ code is included as well.

During my research, I used the syncopation transformations in two projects: first, in generating music examples for a listening experiment on the relation between syncopation and groove, and second, in the development of software plugins. The first project drew important conclusions about syncopation and groove while at the same time it illustrated the details of the transformations on specific music material. The second project developed software plugins for the generation of rhythmic variations on a user input in MIDI and audio. The algorithms and methods used can serve as examples to other generative applications based on the model.

In the next section, I present a summary of the steps that led to the formalization of syncopation as transformation and the introduction of the most important theoretical concepts. I continue with a section on the contribution of this study and its potential in music making and systematic musicology. Finally, I provide some ideas and guidelines for the future expansion of the transformations beyond syncopation.

7.1 Summary

The syncopation model I presented in this thesis was based on three main observations:

1. Syncopation can be modeled as a transformation. A non-syncopating pattern can be syncopated and vice versa, a pattern can be de-syncopated into a non-syncopating counterpart. Such transformations can take the form of displacement of events;
2. Syncopation is felt at a particular silent moment and is attributed retrospectively to the preceding event;
3. Syncopation is the result of the interaction between the rhythm and a specific metrical context. The two can be treated as distinct and independent.

The model was intended to serve as the solid groundwork for further music generation and analysis algorithms. With this I mind, I required it to be as general and consistent as possible and I imposed a number of constraints:

1. The model should take the form of a limited set of elementary transformation steps. More complex transformation should be the result of the combination of several steps;
2. Each step should always have a unique reverse counterpart. This way syncopation and de-syncopation would correspond to the generation and analysis of syncopation;

3. The transformations should maintain other features of the music material as unaffected as possible. The transformations should not change the order of music events because such events are distinct through other properties than their timing, such as pitch, timbre etc.;
4. The transformations should be based on simple representations of rhythm (binary patterns and sequences of amplitudes) and meter (metrical templates). More complex representations could be reduced to such simple ones.

Based on the above I created a set of two kinds of transformation:

1. A syncopation transformation that takes the form of anticipation of events from strong metrical position to weaker ones;
2. A de-syncopation transformation that is the reverse of a previously applied syncopation and has the form of a forward shift of an event from a relatively weak metrical position to a following stronger one.

In encoded the transformations in two dimensional vectors of the form {pulse , type}, where *pulse* is the metrical position for the non-syncopating position of an event, and *type* is the metrical level difference of the corresponding syncopation. The syncopation vector represents the interaction between the rhythm and the meter with the first dimension depending on the timing of the onsets and the second dimension depending on the metrical template.

In order to introduce syncopation in a pattern, one has to determine a syncopation vector: 1) the position of the syncopation and 2) the type of syncopation. When de-syncopating, the syncopation vector is produced as the output of the process. In this sense, the syncopation transformations correspond to the generation while the de-syncopation to the analysis of syncopation.

Since the de-syncopation of a pattern might involve more than one step, the process produces a series of vectors: an array of syncopations. That way, any rhythmic pattern is codified as a non-syncopating pattern and an array of syncopation transformations; if one applies the array of vectors on the non-syncopating pattern, the original pattern is generated.

The syncopation vectors and arrays give rise to a hierarchical lattice, hierarchical structures that I call syncopation trees. A tree consists of a non-syncopating pattern, the root of the tree, and all possible syncopated patterns that originate from it. It is produced by applying all possible combinations of syncopation vectors on the root pattern.

Finally, I extended the transformations to include dynamic accents and amplitudes. This extension consists in segregating a pattern of amplitude into stacked binary patterns that correspond to the different amplitude levels. Then, a transformation of accents corresponds to the same binary transformation applied to all the stacked binary patterns respecting their hierarchical layering. The advantage of this approach is that both binary and accent transformations make use of the same syncopation vectors to describe same syncopations independently of the type of pattern at hand.

The syncopation transformations partly came out of the need for an effective and automatic algorithm for the manipulation of syncopation in the music examples of listening experiments on groove perception. I included in this dissertation these listening experiments as they are excellent examples of the syncopation transformations in practice but also because of the results and conclusions about syncopation and its relation to groove.

Through the development of the model, its applications in the listening experiment and my personal experience with the software that I developed, I came to confirm some of the important points I made in the discussion of the literature (chapters 2 and 3). In particular, I questioned the nature of a syncopation feeling that corresponds to a measurable quantity—an overall degree or amount of syncopation that characterizes the feeling of an entire pattern. Undoubtedly, one can compare the intensity of a specific instance of syncopation—how strong it is felt—and even compare more or less syncopated patterns. I argued, however, that such comparisons are sensible only when the patterns share some common characteristics. Some of the characteristics are more obvious, like belonging to the same meter, and other less obvious, like having the same number of onsets.

One cannot avoid noticing that patterns that belong in the same syncopation tree share such characteristics. The patterns of a single tree are alternate versions of each other that vary in their syncopation. Naturally, they have similar features, like similar distributions of notes or accents in the bar. The mere fact that we can transform one pattern of a tree to another following simple steps makes the comparison of their degree of syncopation almost instinctive and effortless. Although, the comparison of syncopation cannot be generally restricted between patterns of the same tree, the clustering of patterns into a single tree guarantees that the comparison will be valuable and sensible.

In the listening experiments of chapter 5, human judgments on groove were correlated to the syncopation in the music examples. The correlation was tested across examples that were a transformation of one another and therefore belonged by nature in the same syncopation tree. In those examples, a degree of syncopation was a sensible quantity, however, the results of the experiments showed that the effect of syncopation on the perception of groove cannot be explained by its magnitude alone. A more complex relation exists, in which the metrical and structural position of each syncopation plays a significant role. The contribution of syncopation on the perceptual qualities of rhythm such as groove might not always be a simple matter of its magnitude.

7.2 Contribution

The contribution of this study is threefold:

1) The main contribution, the syncopation model with its potential applications in music making, systematic musicology and music cognition research.
2) The application of the model in listening experiments which correlated syncopation and groove and shed light on the nature of their relation.
3) The generative software plugins I created based on the model that demonstrate the power of the transformations in music making.

Through the limited set of reversible transformations of the model, one can create generative algorithms that effectively and precisely control the syncopation of the result. The same transformations can be used in the analysis of music, from a single pattern to entire music collections. The model codifies any rhythmic pattern in a given meter in two parts: 1) a non-syncopating counterpart, the root, and 2) a series of syncopation transformations. In this way, generation and analysis can be combined seamlessly. An important consequence of the model is the organization of rhythms in hierarchical structures that I call syncopation trees. The trees can be the backbone of a wide range of novel algorithms and applications some of which I will outline in the following sections.

The way patterns are codified in their non-syncopating form accompanied by a series of transformations separates the rhythm from its syncopation. The de-syncopation process shifts each event to its non-syncopating position as if it was its original position in the meter[35]. The event can be characterized by its non-syncopating position in the meter irrelevant of where it might be found in some other syncopated

[35] In the non-syncopating position the physical and metrical accents coincide.

variation of the pattern. This approach is useful in both generation and analysis, as a number of properties of an event can be linked to its metrical position.

Finally, I would like to highlight the contribution of the transformations to the particular listening experiments we conducted and, in general, to the design of listening experiments. The transformations allowed us to automatically introduce syncopation in two different types of music material: monophonic piano melodies and polyphonic rhythm loops. The algorithms used in each case were different, adapted to the material at hand, but in their core, the same principles of syncopation transformations existed.

Two factors are crucial in the conduct of listening experiments involving such high level music qualities as groove or syncopation: 1) efficient control over the qualities and parameters that are varied and 2) a high ecological validity of the examples. Efficient control means that the automated algorithm should effectively modify the intended quality without at the same time affecting other qualities. Ecological validity essentially means that the music examples should sound as close as possible to “real” music performances; ultimately, the listener should not be able to tell whether the example was composed and performed by musicians or computers.

The experiments presented in chapter 5, show a way of developing algorithms for the manipulation of syncopation under the strict limitations above. The transformations can serve as a good basis for the accurate control of syncopation and they can be applied on real music examples. However, as the last, polyphonic experiment showed, more elaborate algorithms should and can be built on the basis of the transformations that will adapt better to the music material and the nature of the experiment. In the polyphonic experiment, for instance, a better coordination between the instruments could have resulted in a more stable metrical feel while not losing control over the syncopation introduced. It could have resulted in more natural examples and hence, more informative ratings from the participants.

7.3 Future Work

In this section, I outline potential uses of the syncopation model that reveal the implications of the model in both music making and musicology through some examples of generative processes and analysis methods that I find especially interesting. I believe that they demonstrate the power of the syncopation transformations and trees and that they will serve as points of departure for a great number of applications. Finally, I give guidelines on how to generalize the model to other transformations beyond syncopation.

7.3.1 Generative algorithms

A wide range of generative algorithms can especially benefit from the particular codification of rhythms. Purely stochastic algorithms, like the kin.rhythmicator (Sioros & Guedes, 2011b), can separate the various processes of generating variation in order to better control the outcome. For example, one algorithm could generate stochastic variation in the density of notes, another in their position but without either of them introducing any syncopation and subsequently introduce a series of syncopation transformations in the fashion of the S*yncopalooza* plugin (chapter 6). In this way, the different qualities of the resulting rhythm are controlled separately.

Moreover, as the last listening experiment on groove showed (section 5.4), a coordinated syncopation[36] between voices and instruments is essential to establish a coherent and stable metrical context. An algorithm with explicit control over the syncopation through the transformations can link the type of syncopation to the specific metrical positions in the bar simply by sharing the same array of transformation vectors between all voices. For example, the notes in all instruments that syncopate at

[36] By coordinated syncopation, I mean syncopation that is not independently introduced in each instrument but it is governed by a global strategy at the song level.

the half bar are shifted to the previous eighth note position. Of course not all instruments have to syncopate. Each instrument can still have its own independent control of the syncopation level as well its individual variation in other qualities of the rhythm.

The coordination of the syncopation between instruments could be based on the analysis of the syncopation of a particular music. For instance, de-syncopating patterns coming from a particular song or music style, would yield certain syncopation vectors characteristic of the style. The vectors create direct associations between metrical positions and syncopation types that could be used in a generative algorithm as the one described in the previous paragraph. Additionally, information about the order in which the vectors should be applied can be collected by de-syncopating patterns of the same style but with different degrees of syncopation. In this way, one could create a prototypical array of transformations characteristic of the style being analyzed. Such an analysis can in principle be automated on the basis of the de-syncopation algorithm of section 4.4.2. Ultimately, one could prototype syncopation styles and apply them during a performance or production in a similar manner that other rhythmic aspects, such as expressive timing, are prototyped and applied in modern digital audio workstations[37].

Besides the purely rhythmical features, properties like the pitch or timbre of a generated event can be determined based on its non-syncopating position and then carried with the event in any other syncopated position. For instance, the pitch of a note can be determined based on a harmonic progression linked to the strong beats of the meter.

[37] I am referring to Ableton Live’s “Grooves” or similar functions in other workstations, which can be used to modify the timing of audio or MIDI clips to match that of another MIDI clip or style.

7.3.2 Systematic analysis

One major implication of the model is in the analysis of music; rhythmic patterns can be clustered, categorized and analyzed based on their non-syncopating root counterparts. For instance, all the patterns found in an entire song could be reduced to a small number of roots that can be easier analyzed and compared. Similarly, patterns in an entire corpus of music could be reduced to a smaller number of root patterns and their corresponding syncopation vectors.

I would like to demonstrate the power of reducing rhythms into roots and clustering into trees with an example in the systematic analysis of a corpus of a music in search for typical rhythmic features of a certain style. The first step consists in de-syncopating all rhythmic patterns in the corpus and look for roots that occur regularly. Two root patterns might be the same at slower metrical levels and not at faster ones, so that the matching of patterns should be done including only the metrical levels that are common in both. For example, one pattern might contain sixteenth notes while another pattern has nothing faster than the eighth note level. We can compare these patterns disregarding the sixteenth notes. Such a comparison would fail if we were to match syncopated patterns; removing the syncopated notes found in fast metrical levels could result in removing important characteristic features of the patterns. This process would yield a set of root patterns that are the most common for the style. The most frequent ones would contain only slower metrical levels and therefore little detail. As faster levels are introduced, more rhythmic details of the style would be revealed.

The second step of the analysis involves finding the typical syncopation of the style. The analysis should look for frequent syncopation vectors as well as combination of vectors that regularly occur together. The process would yield a set of the most common syncopation arrays. Finally, the information of the two previous analyses would be combined. First, we would simply combine the most common roots with the most common syncopation arrays. However, as I mentioned above, the most common root patterns would naturally contain only slower metrical levels while the metrical positions in the arrays might correspond to faster levels. Since those syncopations were encountered often during the analysis they should be included in the output of the analysis by adding notes in the corresponding metrical positions.

The above analysis results in a bunch of root patterns and syncopation arrays that are typical for a certain style. Besides the useful information they provide about the style itself, they could serve as the style prototypes in generative processes. One could emulate a certain style and generate variation by combining and alternating between several root patterns and syncopation arrays.

The method I just described can be automated in computer algorithms in order to analyze efficiently large collections of music. The above method is an example of a number of novel analytical methods that can be based on the de-syncopation of rhythmic patterns and the analysis of roots and syncopation vectors separately. Volk and Haas (2013) recently presented their research on the rag-time syncopation. They aimed at characterizing the rag-time style by automatically analyzing its syncopation as the means for automatic genre classification. Such kind of research could make use of the syncopation transformations presented in this thesis.

The method outlined above was restricted in the analysis of purely rhythmic features but other aspects of the music could be examined as well. For instance, assigning the various structural features to the “correct” metrical positions is easier when no syncopation is involved. A harmonic progression might make more sense if the harmonic changes take place on the strong beats of the meter. Moreover, in polyphonic music, de-syncopating all voices results in a better matching of the notes between the voices that can be better interpreted by an automated algorithm.

7.3.3 Expanding the syncopation model

In this section, I will present some guidelines for the expansion of the syncopation model to other transformations beyond syncopation. The following general structure of a complete transformation model was conceived based on two main kinds of operations besides the syncopation transformations: 1) displacements of existing onsets that I call metrical shifts and 2) density transformations that remove or introduce new onsets. Other kind of operations could prove to be more effective and should not be excluded. Nevertheless, τhe general aspects of an expanded model described below apply to any set of transformations.

The syncopation transformations gave rise to the syncopation trees as a natural organization of rhythmic patterns resulting from systematically transforming one pattern into another. However, a syncopation tree connects together only a limited number of patterns that correspond to the same root. The syncopation model does not provide any information on the relation between different syncopation trees, between patterns that belong to different trees or between different roots. A more complete picture about the relation between patterns in a certain meter could be obtained if there were other transformations that could create a connection between different root patterns.

In general, a complete model of the form of the syncopation transformations consists in a limited set of operations that jointly interconnect all possible patterns of a certain length given a metrical template. The formalization of a new transformation must be built on top of any existing transformations and not independently and it should follow three constraints: 1) each operation must have a unique reverse counterpart, 2) all operations must be encoded into simple vectors and 3) each operation should *only* connect patterns that no other pre-existing operation in the set connects. The third constraint implies that the model is built bottom to top, beginning with the existing syncopation transformations and gradually formalizing new, higher ones that do not affect the ones existing under them. The complete set of the transformations should be able to generate all possible patterns starting from an empty pattern and a given metrical template.

In Figure 7-1, a representation of the organization of three different operations and the generated patterns is shown. The patterns are found in three different layers and in each layer only one kind of operation is allowed. In this organization, the syncopation trees would be found at the bottom layer. In this layer, root patterns are not connected. Part of the root patterns of the syncopation transformations would be interconnected through the metrical shifts one layer above. The rest of the root patterns, for example patterns that have a different number of onsets, would be interconnected through density transformations at the top layer. In the next paragraphs I will outline the form that the metrical shifts and the density transformations can take.

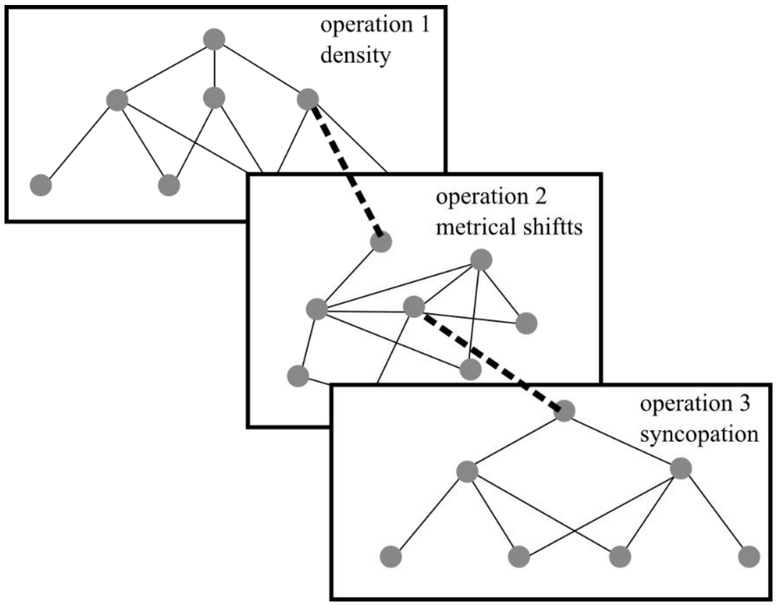


Figure 7-1: An abstract schematic representation of the hierarchical organization of patterns using a set of three different operations. The patters resulting from one operation are the "parents" in another one. The dashed lines do not correspond to a transformation. They simply denote a pattern that is generated by one operation and is the parent pattern for another. For clarity only one lattice or tree from one parent is shown for each operation, although a multitude of lattices and trees exist in each layer.

In chapter 4, I discussed how certain shifts although they might in certain cases produce syncopation they do not do this in a systematic way. I want to begin the expansion of the model by referring to those transformations and specifically to the two *complementary* shifts to the syncopation and de-syncopation pair shown in Figure 7-2. I call those complementary transformations forward and backward metrical shifts. A forward metrical shift is a delay from a strong metrical position to a weaker one. A backward metrical is the reverse shift, an anticipation from a weak position to stronger one.

I used such shifts in the experiments of chapter 5 (experiments A and B), in order to emphasize the meter in the bars preceding a strong syncopation. In these examples, the complimentary transformations were applied on non-syncopating patterns and introduced no syncopation. In other words, they transformed one root pattern to another. Therefore, the metrical shifts can be used to create meaningful connections between different syncopation trees.

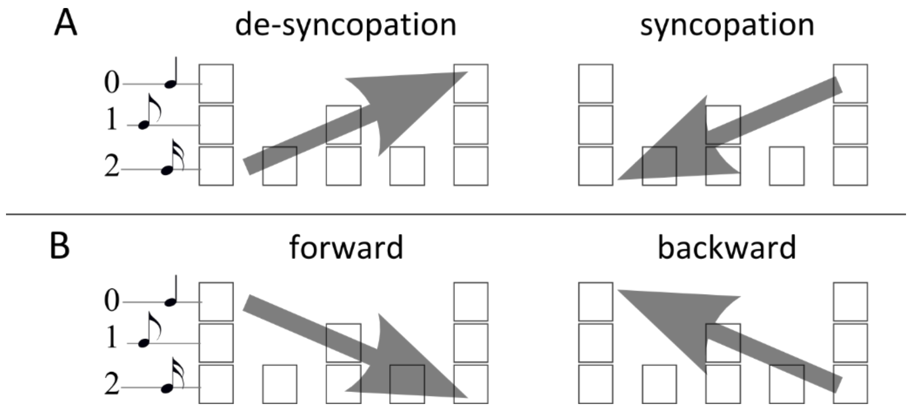


Figure 7-2: A: The syncopation and de-syncopation pair of transformations. B: Their complementary transformations, the forward and backward metrical shifts.

However, as I already discussed in section 4.2.3, in certain cases they can produce syncopation in a non-systematic and non-controlled way depending on the position of their surrounding onsets. A proper formalization of the metrical shifts should follow the general restrictions mentioned above and restrict them to transformations between root patterns preventing them from generating syncopation. This will strongly influence the form of the corresponding transformation vectors.

The second kind of operation I would like to discuss is the density transformations. An elementary density transform would add an onset to a pattern, while its reverse would remove the same onset. Their formalization appears to be a simple task at first glance. After all, the density transformations conform naturally to the first restriction and their encoding into vectors could be as simple as a single number denoting the pulse where the event should be added or removed. I discussed some algorithms for density transformations in section 6.4. However, those transformations were not considered a part of a coherent model but were aimed at more practical generative applications. In the following, I want to point out the difficulties and the theoretical restrictions involved in formalizing the density transformations in order to create a complete rhythm transformation model.

The third constraint drastically influences the form of the density transformations. It can be articulated as follows: the density transformations should *only* be connecting patterns that are not connected through syncopation transformations or metrical shifts. This effectively means that, starting with an empty pattern and a given metrical template, the density transformations should create all those root patterns that are not already interconnected through metrical shifts. The rest of the root patterns should be generated by those roots through metrical shifts and not through density transformations. This constraint would be easier to satisfy if all the root patterns with a specific number of onsets are connected through metrical shifts. Then, the density transformations would generate only patterns with different number of onsets.

The need for a third restriction becomes apparent if we imagine an unrestricted version of the density transformations. In this case, one could add or remove an onset at any position in a pattern creating any pattern possible without the need for any other transformation or the metrical template. This renders all transformations, metrical or syncopation shifts or any other kind including density operations, meaningless. The syncopation transformations presented here were created based on the principle that they have the particular meaning of generating or removing syncopation in a pattern. When expanding the model, one should follow the same principle and formulate transformations that have particular musical meanings.

An organization of rhythms such as the one described in Figure 7-1 allows us for the development of classification or clustering algorithms and meaningful rhythm similarity measures. Such algorithms could be based on the concept of the minimum edit distance (Wagner & Fischer, 1974): the minimum number of operations one needs to go from one pattern to another. More elaborate algorithms could explore the particular meaning each operation has. For instance, if a path contains a certain transformation and its reverse then they should cancel out and not be counted as two different steps. Imagine pattern A and B that belong in two different syncopation trees. The path between pattern A and B contains 1) the de-syncopation from pattern A to its root pattern, 2) an operation that leads to the root pattern of B, and 3) the syncopation transformations to pattern B. If the de-syncopation array in step 1 matches the syncopation array in step 3 that means that their syncopation is identical and the only difference between the patterns is the operation that connected their roots in step 2. In general, knowledge of the kind of operations that are involved in the transformation from one pattern to another can shed light to how the two patterns are different and not merely how far they are found.

I described all the above details of expanding the syncopation model in order to demonstrate the difficulties of the task. However, I believe that such an expansion not only is feasible but of great value and importance.

8 BIBLIOGRAPHY

Apel, W. (1946). The French Secular Music of the Late Fourteenth Century. *Acta Musicologica*, *18*, 17. http://doi.org/10.2307/932106

Bååth, R., & Madison, G. (2012). The Subjective Difficulty of Tapping to a Slow Beat. In E. Cambouropoulos, C. Tsougras, P. Mavromatis, & K. Pastiadis (Eds.), *Proceedings of the 12th International Conference on Music Perception and Cognition and the 8th Triennial Conference of the Europeab Society for the Cognitive Sciences of Music* (pp. 82–85). Thessaloniki - Greece.

Barlow, C. (1987). Corrections for Clarence Barlow ' s Article: Two Essays on Theory. *Computer Music Journal*, *11*(4), 10.

Barlow, C., & Lohner, H. (1987). Two essays on theory. *Computer Music Journal*, *11*(1), 44–60.

Bolton, T. L. (1894). Rhythm. *The American Journal of Psychology*, *6*(2), 145–238.

Brochard, R., Abecasis, D., Potter, D., Ragot, R., & Drake, C. (2003). The "Ticktock" of Our Internal Clock: Direct Brain Evidence of Subjective Accents in Isochronous Sequences. *Psychological Science*, *14*(4), 362–366. http://doi.org/10.1111/1467-9280.24441

Buytendijk, F. J. J., & Meesters, A. (1942). Duration and course of the auditory sensation. *Commentario. Pontificia Academia Scientiarum*, *6*, 557–576.

Clarke, E. F. (1987a). Categorical Rhythm Perception: an Ecological Perspective. In A. Gabrielsson (Ed.), *Action and Perception in Rhythm and Music* (pp. 19–33). Stockholm: Royal Swedish Academy of

Music.

Clarke, E. F. (1987b). Levels of structure in the organization of musical time. *Contemporary Music Review*, *2*(1), 211–238. http://doi.org/10.1080/07494468708567059

Clarke, E. F. (1999). Rhytym and Timing in Music. In D. Deutsch (Ed.), *The Psychology of Music* (2nd ed., pp. 473–500). New York: Academic Press.

Cocharro, D., Sioros, G., Caetano, M., & Davies, M. M. E. P. (2014). Real-time Manipulation of Syncopation in Audio Loops. In *Joint ICMC 2014 / SMC 2014* (pp. 536–541).

Cook, P. R. (2001). *Music, Cognition, and Computerized Sound*. The MIT Press.

Cooper, G., & Meyer., L. B. (1960). *The Rhythmic Structure of Music*. Chicago: University of Chicago Press.

Davies, M., Madison, G., Silva, P., & Gouyon, F. (2013). The Effect of Microtiming Deviations on the Perception of Groove in Short Rhythms. *Music Perception*, *30*(5), 497–510.

Desain, P., & Honing, H. (1989). The Quantization of Musical Time: A Connectionist Approach. *Computer Music Journal*, *13*(3), 56. http://doi.org/10.2307/3680012

Desain, P., & Honing, H. (2003). The formation of rhythmic categories and metric priming. *Perception*, *32*(3), 341–365. http://doi.org/10.1068/p3370

Duke, R. A. (1989). Musicians' Perception of Beat in Monotonic Stimuli. *Journal of Research in Music Education*, *37*(1), 61. http://doi.org/10.2307/3344953

Essens, P. (1995). Structuring temporal sequences: Comparison of models and factors of complexity. *Perception & Psychophysics*, *57*(4), 519–532. http://doi.org/10.3758/BF03213077

Fitch, W. T., & Rosenfeld, A. J. (2007). Perception and Production of Syncopated Rhythms. *Music Perception: An Interdisciplinary Journal*, *25*(1), 43–58. http://doi.org/10.1525/mp.2007.25.1.43

Fraisse, P. (1982). Rhythm and Tempo. In D. Deutsch (Ed.), *The Psychology of Music* (1st ed., pp. 149–180). New York: Academic Press.

Friberg, A., & Sundström, A. (2002). Swing Ratios and Ensemble Timing in Jazz Performance: Evidence for a Common Rhythmic Pattern. *Music Perception*, *19*(3), 333–349. http://doi.org/10.1525/mp.2002.19.3.333

Gabrielsson, A. (1973). Similarity ratings and dimension analyses of auditory rhythm patterns. II. *Scandinavian Journal of Psychology*, *14*(1), 161–176. http://doi.org/10.1111/j.1467-9450.1973.tb00106.x

Gómez, F., Melvin, A., Rappaport, D., & Toussaint, G. T. (2005). Mathematical measures of syncopation. In *Proc. BRIDGES: Mathematical Connections in Art, Music and Science* (pp. 73–84). Citeseer.

Gómez, F., Thul, E., & Toussaint, G. T. (2007). An experimental comparison of formal measures of rhythmic syncopation. In *Proceedings of the International Computer Music Conference* (pp. 101–104). Copenhagen, Denmark.

Grahn, J. A. (2009). The Role of the Basal Ganglia in Beat Perception. *Annals of the New York Academy of Sciences*, *1169*(1), 35–45. http://doi.org/10.1111/j.1749-6632.2009.04553.x

Grahn, J. A., & Brett, M. (2007). Rhythm and Beat Perception in Motor Areas of the Brain. *Journal of Cognitive Neuroscience*, *19*(5), 893–906. http://doi.org/10.1162/jocn.2007.19.5.893

Grahn, J. A., & Brett, M. (2009). Impairment of beat-based rhythm discrimination in Parkinson's disease. *Cortex*, *45*(1), 54–61. http://doi.org/10.1016/j.cortex.2008.01.005

Grahn, J. A., & McAuley, J. D. (2009). Neural bases of individual differences in beat perception. *NeuroImage*, *47*(4), 1894–1903. http://doi.org/10.1016/j.neuroimage.2009.04.039

Handel, S. (1993). The effect of tempo and tone duration on rhythm discrimination. *Perception & Psychophysics*, *54*(3), 370–382. http://doi.org/10.3758/BF03205273

Hasty, C. F. (1997). *Meter As Rhythm*. New York: Oxford University Press.

Honing, H. (2012). Structure and Interpretation of Rhythm in Music. In D. Deutsch (Ed.), *The Psychology of Music* (3rd ed., pp. 367–404). Academic Press, Elsevier.

Huron, D. (2006). *Sweet anticipation: music and the psychology of expectation*. Cambridge, Massachusetts / London, England: The MIT Press.

Huron, D., & Ommen, A. (2006). An Empirical Study of Syncopation in American Popular Music, 1890?1939. *Music Theory Spectrum*, *28*(2), 211–231. http://doi.org/10.1525/mts.2006.28.2.211

Iversen, J. R., Repp, B. H., & Patel, A. D. (2009). Top-Down Control of Rhythm Perception Modulates Early Auditory Responses. *Annals of the New York Academy of Sciences*, *1169*(1), 58–73. http://doi.org/10.1111/j.1749-6632.2009.04579.x

Janata, P., Tomic, S. T., & Haberman, J. M. (2012). Sensorimotor coupling in music and the psychology of the groove. *Journal of Experimental Psychology: General*, *141*(1), 54–75. http://doi.org/10.1037/a0024208

Jones, M. R. (2008). Musical time. In S. Hallam, I. Cross, & M. Thaut (Eds.), *The oxford handbook of music psychology* (pp. 81–92). New York: Oxford University Press. http://doi.org/10.1093/oxfordhb/9780199298457.013.0008

Jones, M. R., Moynihan, H., MacKenzie, N., & Puente, J. (2002). Temporal aspects of stimulus-driven attending in dynamic arrays. *Psychological Science*, *13*(4), 313–319. http://doi.org/10.1111/j.0956-7976.2002.00458.x

Jongsma, M. L. A., Desain, P., & Honing, H. (2004). Rhythmic context influences the auditory evoked potentials of musicians and nonmusicians. *Biological Psychology*, *66*(2), 129–152. http://doi.org/10.1016/j.biopsycho.2003.10.002

Keil, C. (1995). The Theory of Participatory Discrepancies: A Progress Report. *Ethnomusicology*, *39*, 1–19. http://doi.org/10.2307/852198

Keith, M. (1991). *From Polychords to Polya : Adventures in Musical Combinatorics*. Princeton: Vinculum Press.

Keller, P. E., & Schubert, E. (2011). Cognitive and affective judgements of syncopated musical themes. *Advances in Cognitive Psychology*, *7*, 142–156. http://doi.org/10.2478/v10053-008-0094-0

Kennedy, M., & Bourne Kennedy, J. (2012). *The Oxford Dictionary of Music*. (T. Rutherford-Johnson, Ed.).

Oxford University Press. http://doi.org/10.1093/acref/9780199578108.001.0001

Ladinig, O., Honing, H., Háden, G., & Winkler, I. (2009). Probing Attentive and Preattentive Emergent Meter in Adult Listeners without Extensive Music Training. *Music Perception: An Interdisciplinary Journal*, *26*(4), 377–386. http://doi.org/10.1525/mp.2009.26.4.377

Large, E. W. (2008). Resonating to Musical Rhythm : Theory and Experiment. In S. Grondin (Ed.), *The Psychology of Time* (pp. 189–231). Emerald Group Publishing Limited.

Large, E. W., & Jones, M. R. (1999). The dynamics of attending: How people track time-varying events. *Psychological Review*, *106*(1), 119–159. http://doi.org/10.1037//0033-295X.106.1.119

Large, E. W., & Kolen, J. F. (1994). Resonance and the Perception of Musical Meter. *Connection Science*, *6*(2-3), 177–208. http://doi.org/10.1080/09540099408915723

Large, E. W., & Palmer, C. (2002). Perceiving temporal regularity in music. *Cognitive Science*, *26*(1), 1–37. http://doi.org/10.1207/s15516709cog2601_1

Large, E. W., & Snyder, J. S. (2009). Pulse and Meter as Neural Resonance. *Annals of the New York Academy of Sciences*, *1169*(1), 46–57. http://doi.org/10.1111/j.1749-6632.2009.04550.x

Lerdahl, F., & Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. Cambridge, MA: The MIT Press.

Lewin, D. (2007). *Generalized Musical Intervals and Transformations*. New York: Oxford University Press. http://doi.org/10.1093/acprof:oso/9780195317138.001.0001

London, J. (2012). *Hearing in Time* (2nd ed.). New York: Oxford University Press.

London, J., Himberg, T., & Cross, I. (2006). The Effect of Tempo on the Perception of Anacruses. In *Proceedings of the 9th ICMPC.* (pp. 1641–1647). Bologna, Italy.

Longuet-Higgins, H. C., & Lee, C. S. (1984). The rhythmic interpretation of monophonic music. *Music Perception: An Interdisciplinary Journal*, *1*(4), 424–441. http://doi.org/10.2307/40285271

Madison, G. (2006). Experiencing Groove Induced by Music: Consistency and Phenomenology. *Music Perception*, *24*(2), 201–208. http://doi.org/10.1525/mp.2006.24.2.201

Madison, G., Gouyon, F., Ullén, F., & Hörnström, K. (2011). Modeling the tendency for music to induce movement in humans: First correlations with low-level audio descriptors across music genres. *Journal of Experimental Psychology: Human Perception and Performance*, *37*(5), 1578–1594. http://doi.org/10.1037/a0024323

Madison, G., & Sioros, G. (2014). What musicians do to induce the sensation of groove in simple and complex melodies, and how listeners perceive it. *Frontiers in Psychology*, *5*, 894. http://doi.org/10.3389/fpsyg.2014.00894

McAuley, J. D. (2010). Music Perception. In M. Riess Jones, R. R. Fay, & A. N. Popper (Eds.), *Music Perception* (pp. 165–199). New York, NY: Springer New York. http://doi.org/10.1007/978-1-4419-6114-3

McAuley, J. D., & Semple, P. (1999). The Effect of Tempo and Musical Experience on Perceived Beat. *Australian Journal of Psychology*, *51*(3), 176–187. http://doi.org/10.1080/00049539908255355

Middleton, R. (1990). *Studying Popular Music* (1st ed.). Open University Press.

Miron, M., Davies, M. E. P., & Gouyon, F. (2013). An open-source drum transcription system for Pure Data and Max MSP. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 221–225). Vancouver, Canada: IEEE. http://doi.org/10.1109/ICASSP.2013.6637641

Moelants, D., & Mckinney, M. F. (2004). Tempo perception and musical content: What makes a piece fast, slow, or temporally ambiguous? In *Proceedings of the 8th International Conference on Music Perception & Cognition* (pp. 558–562). Evanston, IL, USA.

Palmer, C., & Krumhansl, C. L. (1990). Mental representations for musical meter. *Journal of Experimental Psychology. Human Perception and Performance*, *16*(4), 728–41.

Parncutt, R. (1987). The perception of pulse in musical rhythm. In A. Gabrielsson (Ed.), *Action and Perception in Rhythm and Music* (pp. 127–138). Stockholm: Royal Swedish Academy of Music.

Parncutt, R. (1994). A perceptual model of pulse salience and metrical accent in musical rhythms. *Music Perception*, *11*(4), 409–464.

Peretti, B. W. (1994). *The Creation of Jazz: Music, Race, and Culture in Urban America*. University of Illinois Press.

Povel, D.-J. (1981). Internal representation of simple temporal patterns. *Journal of Experimental Psychology. Human Perception and Performance*, *7*(1), 3–18. http://doi.org/10.1037/0096-1523.7.1.3

Povel, D.-J., & Essens, P. (1985). Perception of Temporal Patterns. *Music Perception: An Interdisciplinary Journal*, *2*(4), 411–440. http://doi.org/10.2307/40285311

Povel, D.-J., & Okkerman, H. (1981). Accents in equitone sequences. *Perception & Psychophysics*, *30*(6), 565–572. http://doi.org/10.3758/BF03202011

Pressing, J. (1997). Cognitive complexity and the structure of musical patterns. In C. H. R. Heath, B. Hayes, A. Heathcote (Ed.), *Proceedings of the 4th Conference of the Australasian Cognitive Science Society*. Newcastle, Australia.

Randel, D. M. (1986). *The Harvard Dictionary of Music*. (D. M. Randel, Ed.) (4rth ed.). Cambridge, MA: Belknap Press of Harvard University Press.

Repp, B. H. (2005a). Rate Limits of On-Beat and Off-Beat Tapping With Simple Auditory Rhythms: 1. Qualitative Observations. *Music Perception: An Interdisciplinary Journal*, *22*(3), 479–496. http://doi.org/10.1525/mp.2005.22.3.479

Repp, B. H. (2005b). Rate Limits of On-Beat and Off-Beat Tapping With Simple Auditory Rhythms: 2. The Roles of Different Kinds of Accent. *Music Perception: An Interdisciplinary Journal*, *23*(2), 165–188. http://doi.org/10.1525/mp.2005.23.2.165

Repp, B. H. (2005c). Sensorimotor synchronization: A review of the tapping literature. *Psychonomic Bulletin & Review*, *12*(6), 969–992. http://doi.org/10.3758/BF03206433

Repp, B. H. (2006). Rate Limits of Sensorimotor Synchronization. *Advances in Cognitive Psychology*, *2*(2), 163–181. http://doi.org/10.2478/v10053-008-0053-9

Repp, B. H., & Doggett, R. (2007). Tapping to a Very Slow Beat: A Comparison of Musicians and Nonmusicians. *Music Perception: An Interdisciplinary Journal*, *24*(4), 367–376.

http://doi.org/10.1525/mp.2007.24.4.367

Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision*, *40*(2), 99–121. http://doi.org/10.1023/A:1026543900054

Schulze, H.-H. (1989). Categorical perception of rhythmic patterns. *Psychological Research*, *51*(1), 10–15. http://doi.org/10.1007/BF00309270

Shmulevich, I., & Povel, D.-J. (2000). Measures of Temporal Pattern Complexity. *Journal of New Music Research*, *29*(1), 61–69. http://doi.org/10.1076/0929-8215(200003)29:01;1-P;FT061

Sioros, G., & Guedes, C. (2011a). A formal approach for high-level automatic rhythm generation. In *Proceedings of the BRIDGES 2011 – Mathematics, Music, Art, Architecture, Culture Conference.* Coimbra, Portugal.

Sioros, G., & Guedes, C. (2011b). Automatic rhythmic performance in Max/MSP: the kin. rhythmicator. In A. R. Jensenius, A. Tveit, R. I. Godøy, & D. Overholt (Eds.), *Proceedings of the International Conference on New Interfaces for Musical Expression* (pp. 88–91). Oslo, Norway.

Sioros, G., & Guedes, C. (2011c). Complexity Driven Recombination of MIDI Loops. In *Proceedings of the 12th International Society for Music Information Retrieval Conference* (pp. 381–386). Miami,Florida (USA).

Sioros, G., Holzapfel, A., & Guedes, C. (2012). On Measuring Syncopation to Drive an Interactive Music System. In F. Gouyon, P. Herrera, L. G. Martins, & M. Mülle (Eds.), *Proceedings of the 13th International Society for Music Information Retrieval Conference* (pp. 283–288). Porto, Portugal:

FEUP Edições.

Sioros, G., Madison, G., Davies, M., Miron, M., Cocharro, D., & Gouyon, F. (2013). Adding syncopation to simple melodies increases the perception of groove. In *The biennial meeting of the Society for Music Perception and Cognition (SMPC)*. Toronto: Ryerson University.

Sioros, G., Miron, M., Cocharro, D., Guedes, C., & Gouyon, F. (2013). Syncopalooza : Manipulating the Syncopation in Rhythmic Performances. In *10th International Symposium on Computer Music Multidisciplinary Research* (Vol. 10th, pp. 454–469). Marseille, France: Laboratoire de Mécanique et d'Acoustique.

Sioros, G., Miron, M., Davies, M., Gouyon, F., & Madison, G. (2014). Syncopation creates the sensation of groove in synthesized music examples. *Frontiers in Psychology*, *5*, 1036. http://doi.org/10.3389/fpsyg.2014.01036

Smith, L. M., & Honing, H. (2006). Evaluating and extending computational models of rhythmic syncopation in music. In *Proceedings of the International Computer Music Conference* (pp. 688–691). New Orleans.

Snyder, J. S., Large, E. W., & Penhune, V. (2009). Part I introduction: rhythms in the brain: basic science and clinical perspectives. *Annals of the New York Academy of Sciences*, *1169*, 13–4. http://doi.org/10.1111/j.1749-6632.2009.04860.x

Song, C. (2014, June). *Syncopation: Unifying Music Theory and Perception*. Queen Mary, University of London.

Song, C., Simpson, A. J. R., Harte, C. a, Pearce, M. T., & Sandler, M. B. (2013). Syncopation and the Score. *PLoS ONE*, *8*(9), e74692. http://doi.org/10.1371/journal.pone.0074692

T. Toussaint, G. (2002). A Mathematical Analysis of African, Brazilian and Cuban Clave Rhythms. In R. Sarhangi (Ed.), *Bridges: Mathematical Connections in Art, Music, and Science* (pp. 157–168). Towson, Maryland: Bridges Conference.

Temperley, D. (1999). Syncopation in rock: a perceptual perspective. *Popular Music*, *18*(01), 19–40. http://doi.org/10.1017/S0261143000008710

Temperley, D. (2009). A Unified Probabilistic Model for Polyphonic Music Analysis. *Journal of New Music Research*, *38*(1), 3–18. http://doi.org/10.1080/09298210902928495

Toussaint, G. T. (2003). Classification and phylogenetic analysis of African ternary rhythm timelines. In J. Barrallo, N. Friedman, J. A. Maldonado, J. Martínez-Aroza, R. Sarhangi, & C. Séquin (Eds.), *Meeting Alhambra, ISAMA-BRIDGES Conference Proceedings* (pp. 2–36). Granada, Spain: University of Granada.

Toussaint, G. T. (2004). A mathematical measure of preference in African rhythm. In *Abstracts of Papers Presented to the American Mathematical Society* (Vol. 25, p. 248). Phoenix, Arizona: American Mathematical Society.

Velasco, M. J., & Large, E. W. (2011). Pulse Detection in Syncopated Rhythms Using Neural Oscillators. In A. Klapuri & C. Leider (Eds.), *Proceedings of the 12th International Society for Music Information Retrieval Conference*. Miami, Florida, USA: University of Miami.

Volk, A., & Haas, W. B. de. (2013). A Corpus-Based Study on Ragtime Syncopation. In A. de S. B. Jr., F. Gouyon, & S. Dixon (Eds.), *Proceedings of the 14th International Society for Music Information Retrieval Conference* (pp. 163–168). Curitiba, Brazil.

Wagner, R. A., & Fischer, M. J. (1974). The String-to-String Correction Problem. *Journal of the ACM*, *21*(1), 168–173. http://doi.org/10.1145/321796.321811

Witek, M. a G., Clarke, E. F., Wallentin, M., Kringelbach, M. L., & Vuust, P. (2014). Syncopation, Body-Movement and Pleasure in Groove Music. *PLoS ONE*, *9*(4), e94446. http://doi.org/10.1371/journal.pone.0094446

Woodrow, H. A. M. (1909). *A quantitative study of rhythm. The effect of variations in intensity, rate and duration*. The Science press, New York, USA.

Woodrow, H. A. M. (1932). The effect of rate of sequence upon the accuracy of synchronization. *Journal of Experimental Psychology*, *15*(4), 357–379. http://doi.org/10.1037/h0071256

Yeston, M. (1976). *The Stratification of Musical Rhythm*. New Haven, CT: Yale University Press.

APENDIX A

```
/**

	R.transformation - Cycling 74 Max external
	Copyright 2014, George Sioros

*/

#include "ext.h"	// standard Max include, always required
#include "ext_obex.h" // required for new style Max object

//warp a negative index to the begining of the pattern's length
////////////////////////// object struct
typedef struct _R_transformation
{
	t_object		ob;// the object itself (must be first)
	long			m_in;// space for the inlet number used by all the proxies
	void			*p_proxyR;// proxy for the additional inlet (total 2 inlets)
	void			*p_outletL, *p_outletR;// pointers to the outlets of the
objects

	long			l_templateLength, l_numberOfLevels, l_patternLength;
	long			*pl_levels, *pl_pattern;

	bool			b_loop;
	bool			b_allow0type; //allow the 0 type transformations at the beat
level
	bool			b_fastestType;
	bool			b_supressErrors;
} t_R_transformation;

///////////////////////// function prototypes
//// standard set
void *R_transformation_new(t_symbol *s, long argc, t_atom *argv);
void R_transformation_free(t_R_transformation *x);
void R_transformation_assist(t_R_transformation *x, void *b, long m, long a, char *s);

//// responses to messages
void R_transformation_bang(t_R_transformation *x);
void R_transformation_list(t_R_transformation *x, t_symbol *s, short argc, t_atom *argv);
// response to list message received in one of the three inlets
void R_transformation_deSyncopate(t_R_transformation *x, long n);
void R_transformation_syncopate(t_R_transformation *x, t_symbol *s, short argc, t_atom
*argv);
void R_transformation_Mplus(t_R_transformation *x, t_symbol *s, short argc, t_atom
*argv);
void R_transformation_Mminus(t_R_transformation *x, long n);
void R_transformation_Plus(t_R_transformation *x, long pulse, long minType, long maxType,
long direction); //use for syncopate or M+ transformation
void R_transformation_Minus(t_R_transformation *x, long pulse, long direction);

//// free & allocate memory, process and store the input
void R_transformation_clearMemory(t_R_transformation *x);
```

```
bool R_transformation_setLevels(t_R_transformation *x, long *arr, long size);
bool R_transformation_setPattern(t_R_transformation *x, long *arr, long size);
void R_transformation_processList(t_R_transformation *x, short inletNum, short argc,
t_atom *argv);

//// transformation calculation and output
long R_transformation_findOnsetInFasterLevel(t_R_transformation *x, long initialPulse,
long direction, bool fasterLevel = true);
long R_transformation_findPulseInLevel(t_R_transformation *x, long initialPulse, long
level, long direction);
void R_transformation_outputResult(t_R_transformation *x);
void R_transformation_initializeTransformations(t_R_transformation *x);

//////////////////////// global class pointer variable
void *R_transformation_class;

int C74_EXPORT main(void)
{

       t_class *c;

       c = class_new("R.transformation", (method)R_transformation_new,
(method)R_transformation_free, (long)sizeof(t_R_transformation),
                              0L /* leave NULL!! */, A_GIMME, 0);

       class_addmethod(c, (method)R_transformation_assist,    "assist", A_CANT, 0);
       class_addmethod(c, (method)R_transformation_bang, "bang", A_NOTHING, 0);
       class_addmethod(c, (method)R_transformation_list,"list",   A_GIMME, 0);
       class_addmethod(c, (method)R_transformation_deSyncopate, "desync",  A_LONG, 0);
       class_addmethod(c, (method)R_transformation_deSyncopate, "D",   A_LONG, 0);
       class_addmethod(c, (method)R_transformation_syncopate,  "sync",   A_GIMME, 0);
       class_addmethod(c, (method)R_transformation_syncopate,  "S",   A_GIMME, 0);
       class_addmethod(c, (method)R_transformation_Mminus,   "M-",   A_LONG, 0);
       class_addmethod(c, (method)R_transformation_Mplus,   "M+",   A_GIMME, 0);

       CLASS_ATTR_CHAR (c, "0type", 0, t_R_transformation, b_allow0type);
       CLASS_ATTR_CHAR (c, "supressErrors", 0, t_R_transformation, b_supressErrors);
       CLASS_ATTR_CHAR (c, "loop", 0, t_R_transformation, b_loop);
       CLASS_ATTR_CHAR (c, "fastestLevel", 0, t_R_transformation, b_fastestType);

       CLASS_ATTR_DEFAULT_SAVE (c, "0type", 0 , "0");
       CLASS_ATTR_STYLE (c, "0type", 0, "onoff");
       CLASS_ATTR_DEFAULT_SAVE (c, "supressErrors", 0 , "1");
       CLASS_ATTR_STYLE (c, "supressErrors", 0, "onoff");
       CLASS_ATTR_DEFAULT_SAVE (c, "loop", 0 , "1");
       CLASS_ATTR_STYLE (c, "loop", 0, "onoff");
       CLASS_ATTR_DEFAULT_SAVE (c, "fastestLevel", 0 , "0");
       CLASS_ATTR_STYLE (c, "fastestLevel", 0, "onoff");
       class_register(CLASS_BOX, c); /* CLASS_NOBOX */
       R_transformation_class = c;

       return 0;
}

void R_transformation_assist(t_R_transformation *x, void *b, long m, long a, char *s)
{
       if (m == ASSIST_INLET) { // inlet
              switch (a)
```

```
            {
            case 0:
                   sprintf(s, "(list) pattern or sync / desycn message", a);
                   break;
            case 1:
                   sprintf(s, "(list) metrical levels", a);
                   break;
            }
       }
       else { // outlet
            switch (a)
            {
            case 0:
                   sprintf(s, "(list) pattern", a);
                   break;
            case 1:
                   sprintf(s, "transfromations performed", a);
                   break;
            }
       }
}

void R_transformation_free(t_R_transformation *x)
{
       R_transformation_clearMemory(x);
}

void R_transformation_clearMemory(t_R_transformation *x)
{

}

void *R_transformation_new(t_symbol *s, long argc, t_atom *argv)
{
       t_R_transformation *x = NULL;

       // object instantiation, NEW STYLE
       if (x = (t_R_transformation *)object_alloc((t_class*)R_transformation_class))
       {

              x->p_outletR = outlet_new(x, NULL);
              x->p_outletL = outlet_new(x, NULL);
              x->p_proxyR = proxy_new((t_object *)x, 1, &x->m_in);//set right proxy
              R_transformation_clearMemory(x);//set all pointers to NULL

              x->b_allow0type = 0;
              x->b_fastestType = 0;
              x->b_loop = 1;
              x->b_supressErrors = 1;

       }
       return (x);
}


void R_transformation_bang(t_R_transformation *x)
{
       R_transformation_outputResult(x);
```

```
}



//      response to the list message
//      it calls the R_transformation_processList including the inlet number
void R_transformation_list(t_R_transformation *x, t_symbol *s, short argc, t_atom *argv)
{
        long inletNum = proxy_getinlet((t_object *)x);
        R_transformation_processList(x, inletNum, argc, argv);
}

//      response to the desync message
void R_transformation_deSyncopate(t_R_transformation *x, long n)
{
        R_transformation_Minus (x, n, 1);
}

//response to the syncop message
void R_transformation_syncopate(t_R_transformation *x, t_symbol *s, short argc, t_atom
*argv)
{
        long _pulse, _typeMIN, _typeMAX;
        if (argc < 1)
                return;
        if (atom_gettype(argv) == A_LONG)
                _pulse = atom_getlong(argv);
        else
                return;
        if (argc == 2 && atom_gettype(argv+1) == A_LONG)
        {
                _typeMIN = atom_getlong(argv+1);
                _typeMAX = _typeMIN;
        } else          if (argc > 2 && atom_gettype(argv+1) == A_LONG &&
atom_gettype(argv+2) == A_LONG)
        {
                _typeMIN = atom_getlong(argv+1);
                _typeMAX = atom_getlong(argv+2);
        }else
        {
                _typeMIN = 0;
                _typeMAX = x->l_numberOfLevels;
        }

        R_transformation_Plus (x, _pulse, _typeMIN, _typeMAX, -1);
}


//response to the M- message (similar to desyncopation but in the forward direction)
void R_transformation_Mminus(t_R_transformation *x, long n)
{
        R_transformation_Minus (x, n, -1);
}

void R_transformation_Mplus(t_R_transformation *x, t_symbol *s, short argc, t_atom *argv)
{
        long _pulse, _typeMIN, _typeMAX;
        if (argc < 1)
```

```
		return;
	if (atom_gettype(argv) == A_LONG)
		_pulse = atom_getlong(argv);
	else
		return;
	if (argc == 2 && atom_gettype(argv+1) == A_LONG)
	{
		_typeMIN = atom_getlong(argv+1);
		_typeMAX = _typeMIN;
	} else		if (argc > 2 && atom_gettype(argv+1) == A_LONG &&
atom_gettype(argv+2) == A_LONG)
	{
		_typeMIN = atom_getlong(argv+1);
		_typeMAX = atom_getlong(argv+2);
	}else
	{
		_typeMIN = 0;
		_typeMAX = x->l_numberOfLevels;
	}
	R_transformation_Plus (x, _pulse, _typeMIN, _typeMAX, 1);
}


//	type checking the incoming list (all lists must contain only int, floats are
converted to ints)
void R_transformation_processList(t_R_transformation *x, short inletNum, short argc,
t_atom *argv)
{
	long i = 0;
	long *_l_Array = NULL;
	bool _NaN = false;

	_l_Array = new long[argc];
	while (i<argc && !_NaN)
	{
		if (atom_gettype(argv + i) == A_LONG)
			_l_Array[i] = atom_getlong(argv + i);
		else if (atom_gettype(argv + i) == A_FLOAT)
			_l_Array[i] = (long)atom_getfloat(argv + i);
		else
			_NaN = true;
		i++;
	}
	if (_NaN)
	{
		if (!x->b_supressErrors)
			object_error((t_object*)x, "the list must contain only numbers");
		delete[] _l_Array;
		return;//this means that not all ellements in the list were numbers
	}
	switch (inletNum)
	{
	case 0: //LEFT (pattern)
		if (!R_transformation_setPattern(x, _l_Array, argc)) //set the pattern
		{
			if (!x->b_supressErrors)
				object_error((t_object*)x, "pattern not set");
			delete[] _l_Array; //if pattern is not stored free allocated memory
```

```
            }
            break;
    case 1: //RIGHT (metrical levels)
            if (!R_transformation_setLevels(x, _l_Array, argc)) //set the metrical
levels
            {
                    if (!x->b_supressErrors)
                            object_error((t_object*)x, "metrical levels not set");
                    delete[] _l_Array; //if levels are not stored free allocated memory
            }
            break;
    }
}


// sets the current metrical levels' array to the arr array
// returns false only when size < 2
bool R_transformation_setLevels(t_R_transformation *x, long *arr, long size)
{
    int i, _slowestL, _fastestL;
    if (size != x->l_templateLength)//if the previous arrays have different sizes then
clear them
            R_transformation_clearMemory(x);
    if (size < 2)
            return false;
    //put the levels in order
    _slowestL = _fastestL = arr[0];
    for (i = 1; i<size; i++)
    {
            if (arr[i] < _slowestL)                 _slowestL = arr[i];
            if (arr[i] > _fastestL)                 _fastestL = arr[i];
    }
    //the slowest level should be 0, fatser levels >0
    if (_slowestL != 0)
            for (i = 0; i<size; i++)
                    arr[i] -= _slowestL;
    _fastestL -= _slowestL;
    x->pl_levels = arr;//store the new array
    x->l_templateLength = size;//set the new size
    x->l_numberOfLevels = _fastestL + 1;

    return true;

}

bool R_transformation_setPattern(t_R_transformation *x, long *arr, long size)
{
    if (x->pl_pattern) // delete previously allocated memory
            delete[] x->pl_pattern;
    x->pl_pattern = arr;//store newly allocated pointer
    x->l_patternLength = size;
    return true;
}

// Finds the position of the syncopating onset if it exists
//      returns the initial pulse if no syncopation was not found
//      otherwise returns the index of a pulse that contains the onset to be de-syncopated
//      OR
```

```
//	returns  the index  of a pulse that does not contain an onset but must be 
desyncopated first
long R_transformation_findOnsetInFasterLevel(t_R_transformation *x, long initialPulse, 
long direction, bool fasterLevel )
{
	long _case = 0, _lmin, _currentSlowestML, _currentSlowestMLPulse, _result = 
initialPulse;
	long _p = initialPulse;
	if (_p < 0 || _p >= x->l_patternLength || (direction != 1 && direction!=-1))
		return -1;
	//  if not in loop mode and we are at the begining of the pattern then there is no 
previous onset
	// OR if the initial pulse contains an onset already
	// OR if the pulse belongs to the fastest metrical level
	if ((!x->b_loop && _p == 0)
		|| x->pl_pattern[initialPulse] != 0
		//|| x->pl_levels[_p%x->l_templateLength] >= x->l_numberOfLevels-1
		)
		return _result;
	_lmin = x->pl_levels[_p%x->l_templateLength]; //this is the metrical level of 
initial pulse ans therefore the slowest metrical level allowed in the search
	if (fasterLevel)
		_lmin++; //if the result has to be a faster metrical level than the 
original then the slowest metrical level allowed is the next faster to the initial
	_currentSlowestML = x->l_numberOfLevels; //this is one more than the faster 
metrical level!!
	bool _continue = true;
	while (_continue)
	{
		_p += direction;
		if (x->b_loop)
		{
			if (_p < 0)
				_p += x->l_patternLength;
			else if (_p >= x->l_patternLength)
				_p -= x->l_patternLength;
			if (_p == initialPulse)
				break;//if we made an entire loop
		}
		else
		{
			if (_p < 0 || _p >= x->l_patternLength)
				break;//if we are out of boundaries
		}
		if (x->pl_pattern[_p] ) //if an onset is found
			break;//_continue = false;
		if (x->pl_levels[_p % x->l_templateLength] < _lmin )
			break; //if we are crossing a pulse at a  slower level of or equal 
to the initial pulse level
		if (_currentSlowestML > x->pl_levels[_p%x->l_templateLength]) //if no 
condition is met then check the current slowest metrical level value
		{
			_currentSlowestML = x->pl_levels[_p%x->l_templateLength];
			_currentSlowestMLPulse = _p;
			//set the current slowest metrical level (as we search backwards or 
forward, once a level is "crossed" no faster ones are allowed in the search
			//although we continue searching, if an onset is found in a metrical 
level faster than this one then the _currentSlowestMLPulse should be desyncopated first
```

```
			}
		}
//		object_post ((t_object*)x, "p=%ld  ons=%ld  l=%ld   csML=%ld, csP=%ld, lmin=%ld", 
_p, x->pl_pattern[_p], x->pl_levels[_p%x->l_templateLength], _currentSlowestML, 
_currentSlowestMLPulse, _lmin);
		if ( x->pl_pattern[_p] == 0 || _lmin > x->pl_levels[_p%x->l_templateLength])
			return initialPulse;
		else if ( x->pl_levels[_p%x->l_templateLength] < _currentSlowestML)
			return _p;
		else // if ( x->pl_levels[_p%x->l_templateLength] >= _currentSlowestML)
			return _currentSlowestMLPulse;
}


//searches for a previous unoccupied position that belongs to the faster metrical level 
(compared to the initial position level) "level"
//it returns the index of a pulse if the position is found
//OR the initial pulse if the position is not found
//OR the index of a pulse with an onset if it is blocking the transformation
long R_transformation_findPulseInLevel(t_R_transformation *x, long initialPulse, long 
level, long direction)
{
		long _lmin, _currentSlowestML, _currentSlowestMLPulse, _result = initialPulse;
		long _p = initialPulse;
		long _case = 0;
		//condition that cannot be transformed
		if (_p < 0 || _p >= x->l_patternLength || (direction != -1 && direction != 1)) 
//initial pulse is out of boundaries
			return -1;
		//  if not in loop mode and we are at the begining of the pattern then there is no 
previous pulse
		// OR if the initial pulse does not contain an onset already
		// OR if the pulse belongs to the fastest metrical level
		if ((!x->b_loop && _p == 0)
			|| x->pl_pattern[initialPulse] == 0
			//|| x->pl_levels[_p%x->l_templateLength] >= x->l_numberOfLevels - 1
			|| level<0
			|| level>=x->l_numberOfLevels)
			return _result;
		_lmin = x->pl_levels[_p%x->l_templateLength]; //slowest metrical level = the 
initial Metrical level
		_currentSlowestML = x->l_numberOfLevels;
		while (!_case)
		{
			_p += direction;
			if (_p < 0 && x->b_loop)
				_p += x->l_patternLength;
			else if (_p >= x->l_patternLength && x->b_loop)
				_p -= x->l_patternLength;
			if ((_p < 0 && !x->b_loop)
				|| ( _p >= x->l_patternLength && !x->b_loop)
				|| _p == initialPulse) // if no more pulses are left
				_case |= 1 << 1; //set the 2nd bit and don't check any other 
condition
			else
			{
				if (x->pl_levels[_p%x->l_templateLength] == level)
					_case |= 1; //set the 1st bit
```

```
                    if (x->pl_pattern[_p] != 0)// if we are crossing an onset
                           _case |= 1 << 2; //3rd bit
                    if (x->pl_levels[_p%x->l_templateLength] <= _lmin) // crossing the 
initial metrical level
                           _case |= 1 << 3; //set the 4rth bit
                    if (_currentSlowestML < level)
                           // or the  slowest metrical level so far is already slower 
than the one we look for
                           _case |= 1 << 4; //5th bit
                    if (!_case && _currentSlowestML >= x->pl_levels[_p%x-
>l_templateLength])
                    {
                           _currentSlowestML = x->pl_levels[_p%x->l_templateLength];
                           _currentSlowestMLPulse = _p;
                           //set the current slowest metrical level (as we search 
backwards, once a level is "crossed" no faster ones are allowed in the search
                           //although we continue searching backwards, if an onset is 
found in a metrical level faster than this one then the _currentSlowestMLPulse should be 
desyncopated first
                    }
             }
       }

       if (_case == 1 || _case & (1 << 2))
             _result = _p;//that means that a pulse is found with the correct level and 
no other case was set so no onset!
       // OR that an onset is blocking the transformation. In either cases the 
corresponding pulse is returned
       if ((_case & (1 << 3)) && (_case & 1)) //this is in the case that the "type = 0" -
> requested level = initial level.
             _result = _p; //then the transformation is valid as long as the level is 
not the slowest level!!!! (e.g. the case of the 6/8)
       //In all other cases the initial pulse is returnred as the corresponding metrical 
level was not found

       return _result;
}
void R_transformation_outputResult(t_R_transformation *x)
{
       int i;
       t_atom* _outputAtoms;
       if ( x->l_patternLength <= 0 || x->pl_pattern == NULL)
             return;
       _outputAtoms = new t_atom[x->l_patternLength];
       for (i = 0; i<x->l_patternLength; i++)
             atom_setlong(_outputAtoms + i, x->pl_pattern[i]);
       outlet_anything(x->p_outletL, gensym("list"), x->l_patternLength, _outputAtoms);
       delete[] _outputAtoms;
}


//
void R_transformation_Plus(t_R_transformation *x, long pulse, long minType, long maxType, 
long direction)
{
       long _pulse, _type, _level, _r, _case, _lastNoTransfrom;
       _case = 0;
       _lastNoTransfrom = 0;
```

```
	t_atom _outputAtoms[3]; //three numbers to be output out the right outlet after 
finished
	if (x->l_patternLength <= 0 || x->l_templateLength <= 0)
		return; // in case that the pattern or template anre not properly set then 
return
	if (x->l_patternLength % x->l_templateLength != 0) //if the pattern length is not 
an integer multiple of the template then return
	{
		if (!x->b_supressErrors)
			object_error((t_object*)x, "the length of the pattern is not an 
integer multpiple of the length of the metrical template");
		return;
	}
	if (_pulse < 0 || _pulse >= x->l_patternLength) //if the input pulse to 
desyncopate is not a valid index in the array then return
	{
		if (!x->b_supressErrors)
			object_error((t_object*)x, "not a valid pulse index; out of 
boundaries");
		return;
	}
	if (minType < 0 || maxType < minType)
	{
		if (!x->b_supressErrors)
			object_error((t_object*)x, "not a valid type; type must be a 
positive integer");
		return;
	}
	_type = minType;
	_level = x->pl_levels[pulse%x->l_templateLength] + _type;
	if (x->b_fastestType && _level >= x->l_numberOfLevels) //if the type value is too 
large always try to perform the transform to the fastest metrical level
		_type = x->l_numberOfLevels - x->pl_levels[pulse%x->l_templateLength] - 1;
	while (!_case) //0 initial condition, 1 no more levels to search, 2 transformation 
perfomred
	{
		_level = x->pl_levels[pulse%x->l_templateLength] + _type;
		if (_level >= x->l_numberOfLevels || _type > maxType) //stop search wihtout 
perfroming any trasformations
			_case = 1;
		else if (_level == 0 && !x->b_allow0type)
			_lastNoTransfrom == 4;
		else
		{
			_r = R_transformation_findPulseInLevel(x, pulse, _level, direction);
			if (_r < 0 || _r >= x->l_patternLength)
				_lastNoTransfrom = 0;
			else if (_r == _pulse) //if a valid position was not found
			{
				if (x->pl_pattern[_r])
					_lastNoTransfrom = 1; //no pulse
				else
					_lastNoTransfrom = 2; //no onset
			}
			else if (x->pl_pattern[_r] == 0)
			// a valid position was returned perfrom the transform
			{
				x->pl_pattern[_r] = x->pl_pattern[_pulse];
```

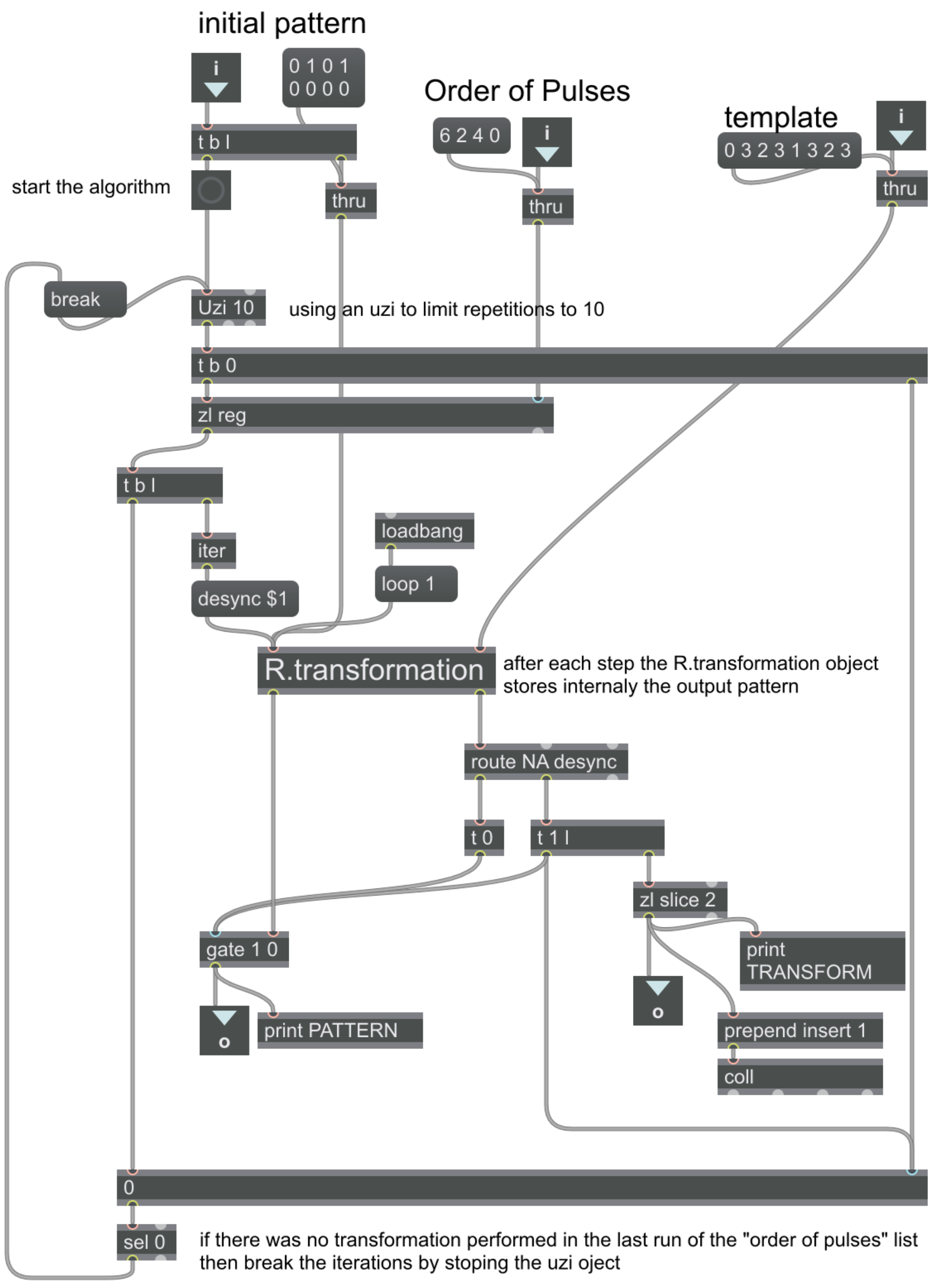
```
                                        x->pl_pattern[_pulse] = 0;
                                        //output the transformation out the right outlet
                                        atom_setlong(_outputAtoms, _pulse);
                                        //first int is the pulse (onset) that is being syncopated
                                        atom_setlong(_outputAtoms + 1, _type);
                                        // the type of syncopation (metrical level differences)
                                        atom_setlong(_outputAtoms + 2, _r);
                                        // the pulse that the onset was shifted to
                                        if (direction == -1)
                                               outlet_anything(x->p_outletR, gensym("sync"), 3,
_outputAtoms);
                                        else if (direction == 1)
                                               outlet_anything(x->p_outletR, gensym("M+"), 3,
_outputAtoms);
                                        _case = 2;
                                 }
                                 else if (x->pl_pattern[_r] != 0)
                                 //an onset is blocking the transformation
                                        _lastNoTransfrom = 3;
                          }
                          _type++;
              }
              if (_case == 1)
              {
                     if (_lastNoTransfrom == 0 || _lastNoTransfrom == 4)
                            outlet_anything(x->p_outletR, gensym("NA"), 0, NULL);
                     else if (_lastNoTransfrom == 1)
                     {
                            atom_setsym(_outputAtoms, gensym("noPulse"));
                            outlet_anything(x->p_outletR, gensym("NA"), 1, _outputAtoms);
                     } else if (_lastNoTransfrom == 2)
                     {
                            atom_setsym(_outputAtoms, gensym("noOnset"));
                            outlet_anything(x->p_outletR, gensym("NA"), 1, _outputAtoms);
                     } else if (_lastNoTransfrom ==3)
                     {
                            atom_setsym(_outputAtoms, gensym("onset"));
                            atom_setlong(_outputAtoms + 1, _r);
                            outlet_anything(x->p_outletR, gensym("NA"), 2, _outputAtoms);
                     }
              }
              R_transformation_outputResult(x);
}

void R_transformation_Minus(t_R_transformation *x, long pulse, long direction)
{
              long _r; // temporary variable to store the position of the previous onset in  a
faster metrical level
              t_atom _outputAtoms[3]; //three numbers to be output out the right outlet after
finished
              if (x->l_patternLength <= 0 || x->l_templateLength <= 0)
                     return; // in case that the pattern or template anre not properly set then
return
              if (x->l_patternLength % x->l_templateLength != 0) //if the pattern length is not
an integer multiple of the template then return
              {
                     if (!x->b_supressErrors)
```

```
                    object_error((t_object*)x, "the length of the pattern is not an
integer multpiple of the length of the metrical template");
                return;
        }
        if (pulse < 0 || pulse >= x->l_patternLength) //if the input pulse to desyncopate
is not a valid index in the array then return
        {
                if (!x->b_supressErrors)
                        object_error((t_object*)x, "not a valid pulse index");
                return;
        }
        if (x->pl_levels[pulse % x->l_templateLength]!=0 || x->b_allow0type) //if not at
the beat level or if 0 type is allowed
                _r = R_transformation_findOnsetInFasterLevel(x, pulse, -direction, false);
//then the returned pulse can be at the same level as the initial
        else
                _r = R_transformation_findOnsetInFasterLevel(x, pulse, -direction,
true);//otherwise the type 0 syncopation is forbiden and a pulse at a faster metrical
level must be returned
        if (_r >= 0 && _r < x->l_patternLength && _r != pulse )//if the pulse is
syncopating then an index >=0 && different from the original is returned that contains an
onset
        {
                if (x->pl_pattern[_r] != 0)
                {// perfrom the transformation here
                        x->pl_pattern[pulse] = x->pl_pattern[_r];
                        x->pl_pattern[_r] = 0;
                        //output the transformation out the right outlet
                        atom_setlong(_outputAtoms, pulse); //first int is the pulse that is
being desyncopated
                        atom_setlong(_outputAtoms + 1, x->pl_levels[_r%x->l_templateLength]
- x->pl_levels[pulse%x->l_templateLength]); // the type of syncopation (metrical level
differences)
                        atom_setlong(_outputAtoms + 2, _r); // the pulse that the onset was
shifted from
                        if (direction == 1)
                                outlet_anything(x->p_outletR, gensym("desync"), 3,
_outputAtoms);
                        else if (direction == -1)
                                outlet_anything(x->p_outletR, gensym("M-"), 3, _outputAtoms);
                }
                else //the pulse does not contain an onset and therefore needs to be de-
syncopated first
                {
                        atom_setlong(_outputAtoms + 1, _r);
                        if (direction == 1)
                                atom_setsym(_outputAtoms, gensym("desync"));
                        else if (direction == -1)
                                atom_setsym(_outputAtoms, gensym("M-"));
                        outlet_anything(x->p_outletR, gensym("NA"), 2, _outputAtoms);
                }
        }
        else if (_r == pulse)
        {//NOT SYNCOPATING PULSE
                if (direction == 1)
                        atom_setsym(_outputAtoms, gensym("noSyncopate"));
                else
                        atom_setsym(_outputAtoms, gensym("NA"));
```

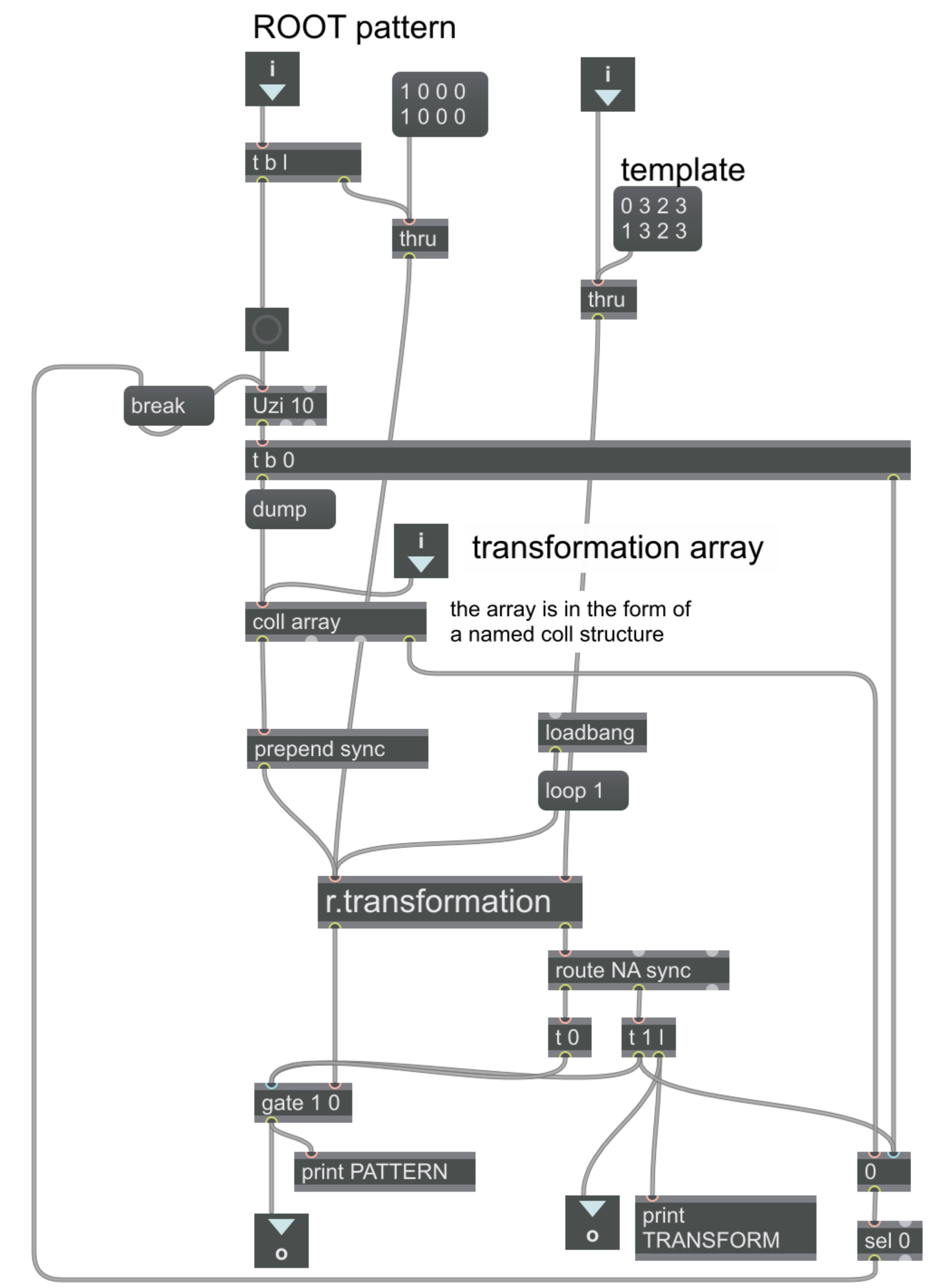
```
			outlet_anything(x->p_outletR, gensym("NA"), 1, _outputAtoms);
		}else
		{
			outlet_anything(x->p_outletR, gensym("error"), 0, NULL);
		}
		R_transformation_outputResult(x);
}
```

APENDIX B

1. [de-syncopate.maxpat]

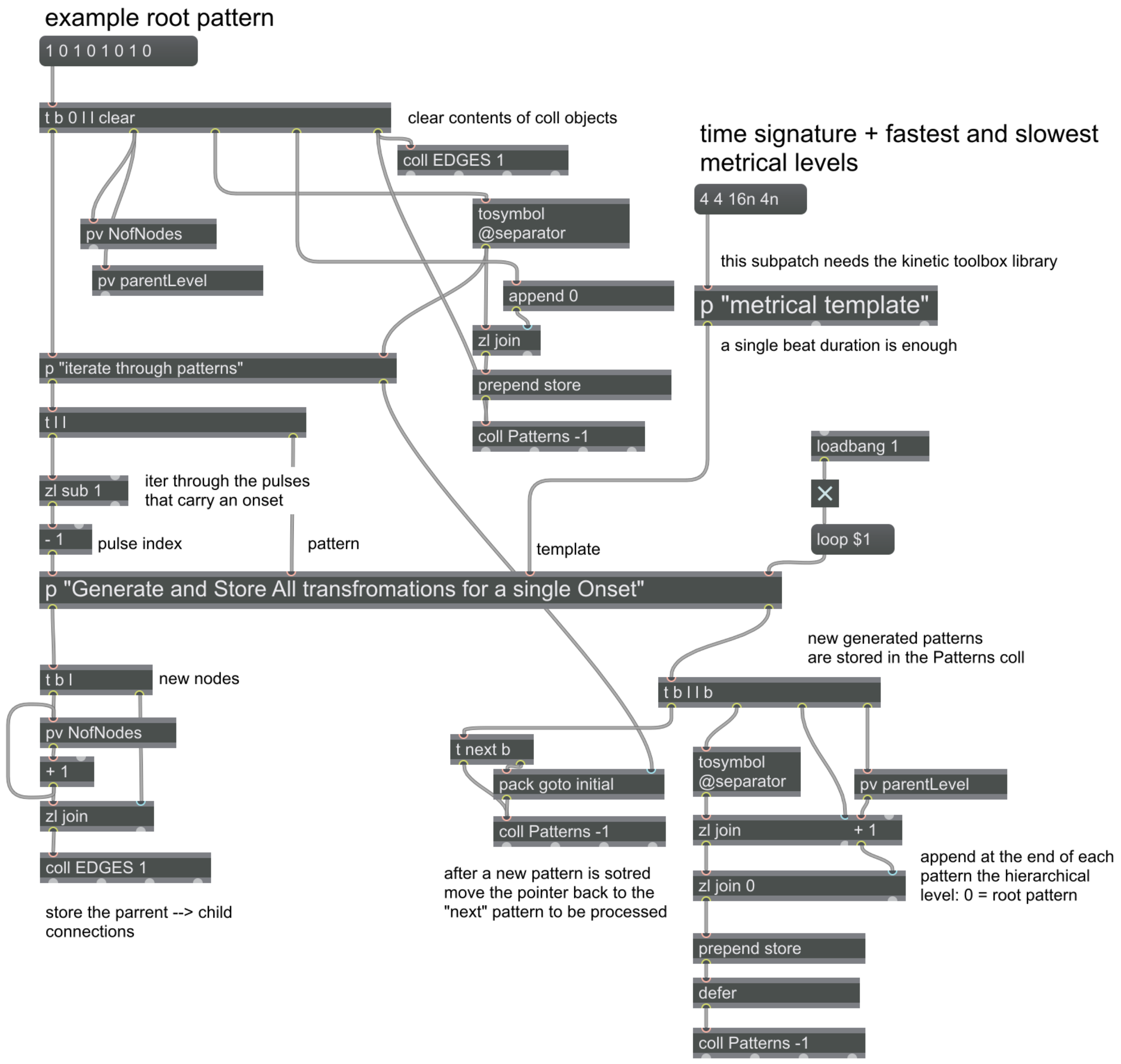


2. [syncopate.maxpat]

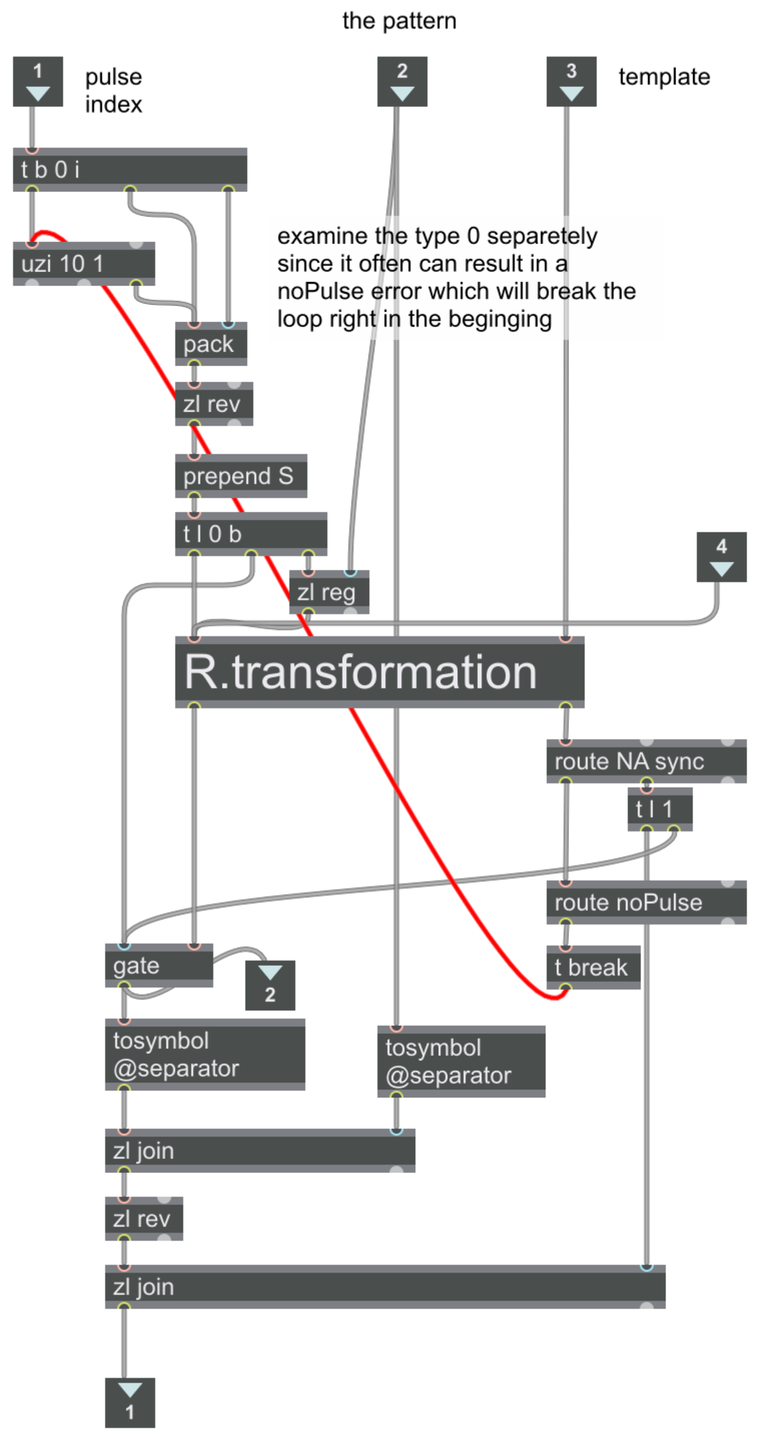


break the iterations if duing the last run of the transformation array there was no shift performed

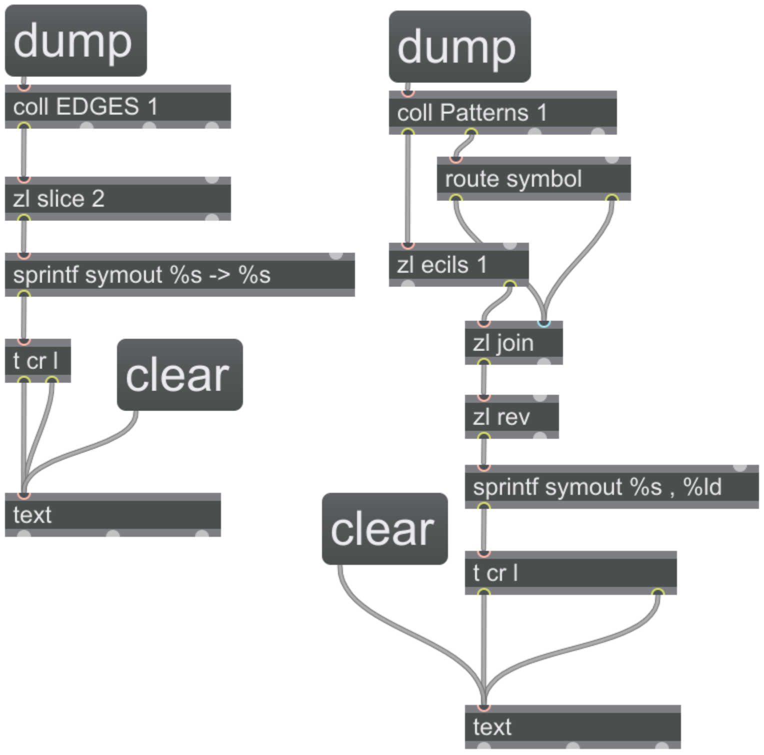
3. [syncopationTree.maxpat] > [generate tree]


example root pattern
1 0 1 0 1 0 1 0
t b 0 l l clear
clear contents of coll objects
coll EDGES 1
time signature + fastest and slowest metrical levels
4 4 16n 4n
tosymbol @separator
pv NofNodes
pv parentLevel
this subpatch needs the kinetic toolbox library
append 0
p "metrical template"
zl join
a single beat duration is enough
p "iterate through patterns"
prepend store
t l l
coll Patterns -1
loadbang 1
zl sub 1
iter through the pulses that carry an onset
- 1
pulse index
pattern
template
loop $1
p "Generate and Store All transfromations for a single Onset"
new generated patterns are stored in the Patterns coll
t b l
new nodes
t b l l b
pv NofNodes
t next b
tosymbol @separator
pv parentLevel
+ 1
pack goto initial
zl join
coll Patterns -1
zl join
+ 1
append at the end of each pattern the hierarchical level: 0 = root pattern
coll EDGES 1
after a new pattern is sotred move the pointer back to the "next" pattern to be processed
zl join 0
store the parrent --> child connections
prepend store
defer
coll Patterns -1

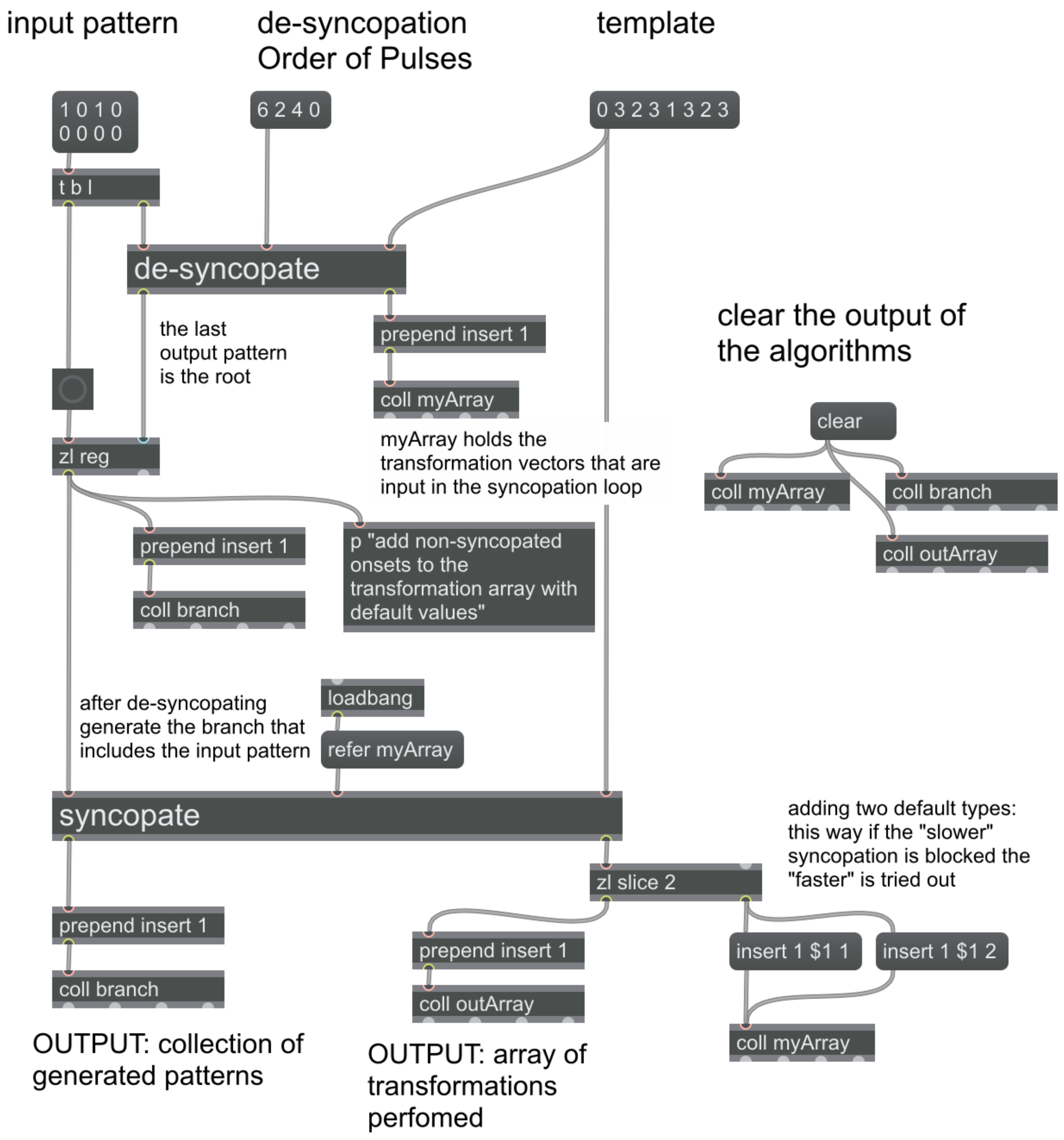
[Generate and Store All transfromations for a single Onset]



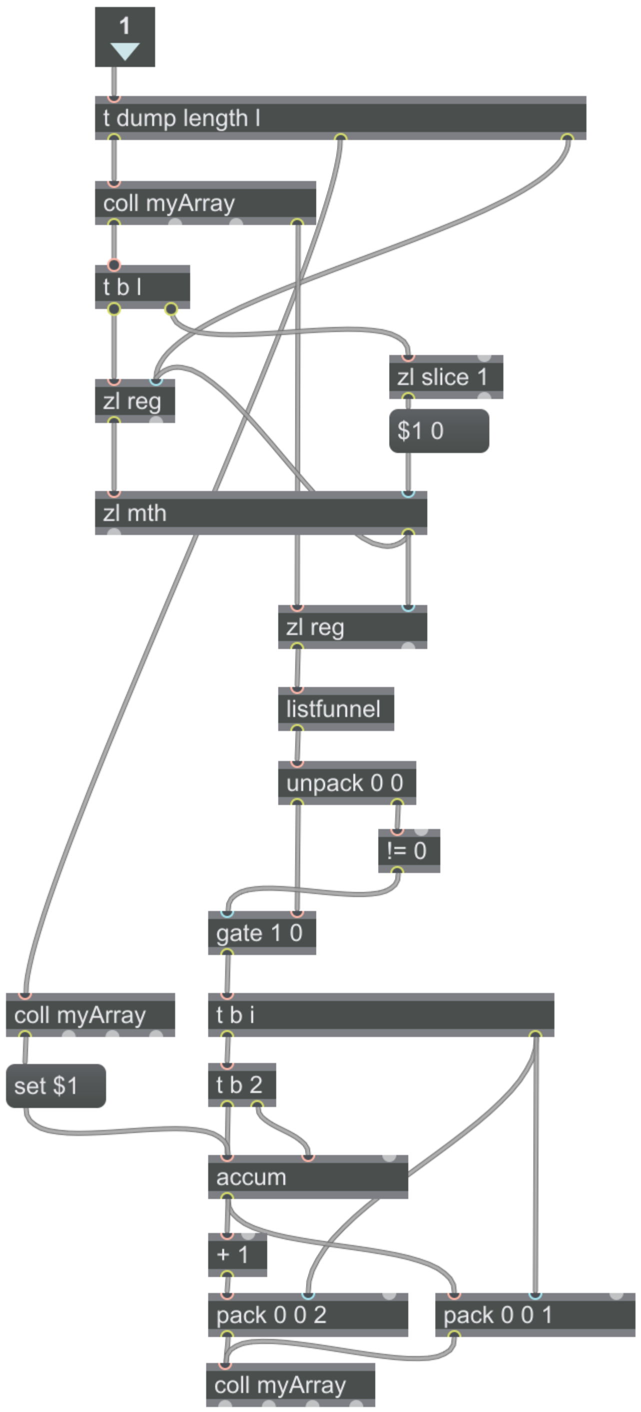
examine the type 0 separetely since it often can result in a noPulse error which will break the loop right in the beginging export to text files



4. [syncopation_Branch.maxpat] > [Generate Branch]


input pattern
de-syncopation Order of Pulses
template
1 0 1 0 0 0 0 0
6 2 4 0
0 3 2 3 1 3 2 3
t b l
de-syncopate
the last output pattern is the root
prepend insert 1
coll myArray
myArray holds the transformation vectors that are input in the syncopation loop
clear the output of the algorithms
clear
coll myArray
coll branch
coll outArray
zl reg
prepend insert 1
coll branch
p "add non-syncopated onsets to the transformation array with default values"
after de-syncopating generate the branch that includes the input pattern
loadbang
refer myArray
syncopate
adding two default types: this way if the "slower" syncopation is blocked the "faster" is tried out
zl slice 2
prepend insert 1
coll branch
prepend insert 1
coll outArray
insert 1 $1 1
insert 1 $1 2
coll myArray
OUTPUT: collection of generated patterns
OUTPUT: array of transformations perfomed

[add non-syncopated onsets to the transformation array with default values]


1
t dump length l
coll myArray
t b l
zl slice 1
zl reg
$1 0
zl mth
zl reg
listfunnel
unpack 0 0
!= 0
gate 1 0
coll myArray
t b i
set $1
t b 2
accum
+ 1
pack 0 0 2
pack 0 0 1
coll myArray