# Chunking: A new Approach to Algorithmic Composition of Rhythm and Metre for Csound

Georg Boenn

University of Lethbridge, Faculty of Fine Arts, Music Department
`georg.boenn@uleth.ca`

**Abstract.** A new concept for generating non-isochronous musical metres is introduced, which produces complete rhythmic sequences on the basis of integer partitions and combinatorics. It was realized as a command-line tool called *chunking*, written in C++ and published under the GPL licence. *Chunking*[1] produces scores for Csound[2] and standard notation output using Lilypond[3]. A new shorthand notation for rhythm is presented as intermediate data that can be sent to different backends. The algorithm uses a musical hierarchy of sentences, phrases, patterns and rhythmic chunks. The design of the algorithms was influenced by recent studies in music phenomenology, and makes references to psychology and cognition as well.

**Keywords:** Rhythm, NI-Metre, Musical Sentence, Algorithmic Composition, Symmetry, Csound Score Generators.

## 1 Introduction

There are a large number of powerful tools that enable algorithmic composition with *Csound*: *CsoundAC*[8], *blue*[11], or *Common Music*[9], for example. For a good overview about the subject, the reader is also referred to the book by Nierhaus[7]. This paper focuses on a specific algorithm written in C++ to produce musical sentences and to generate input files for *Csound* and *lilypond*.

Non-isochronous metres (NI metres) are metres that have different beat lengths, which alternate in very specific patterns. They have been analyzed by London[5] who defines them with a series of well-formedness rules. With the software presented in this paper (from now on called *chunking*) it is possible to generate NI metres. It is even possible to construct sequences of changing NI metres, which are held together by an over-arching musical concept, namely tension and release. Thakar[10] coined the terms 'impact and resolution' for he is concerned with the phenomenological dynamics of musical energy. In his theory, several factors can contribute to the build-up and decline of musical energy; contrast between rhythmic values is one of them. *Chunking* was designed specifically to implement the directed impact and release of musical energy in rhythm,

[1] `https://github.com/gboenn/chunking`
[2] `www.csounds.com`
[3] `www.lilypond.org`

thereby enabling the composition of musically interesting rhythmic sentences. Further inspiration was drawn from the analysis of Eastern-European rhythmics, notably the Aksak[4] rhythms that were studied by Arom[1] and Brăiloiu[2], for example. The theory of Aksak offers prime examples of NI metres. An Aksak metre can be described as a multi-set of beats that are either 2 or 3 pulses long, for example $\{2, 2, 2, 3\}$.

Miller[6] made an important discovery in the field of cognitive psychology by finding a 'magic number' that reappeared again and again in various experiments to test the capacity of human cognition and memory. The number $7 \pm 2$ seems to accurately describe how many items can be held in short-term memory. This number can be increased by a technique called 'chunking'. If a long list of items is cognitively separated into groups (chunks) it will be easier for the mind to grasp, memorize and to reproduce the entire list correctly. Gobet[3] and others have expanded Miller's theory. Their model includes a template mechanism, which allows for slots with variable content in addition to a core of chunks. A chess position, for example, is memorized by building chunks of pieces in the mind, which group together via proximity. There is a certain number of fixed chunks (groups of chess pieces), but also slots which may contain a different variation of a chunk. In this case only a pointer is needed that will lead to the right chunk in long term memory. Chunks and templates facilitate information processing in our minds. The algorithm we describe in this paper makes use of both concepts for the creative process of composing rhythmic phrases.

## 2 The Sentence Algorithm

*Chunking* uses a top-down approach for generating rhythmically interesting musical sentences. The software is working on the assumption that musical rhythms are often based on a small underlying pulsation. There are two main input-arguments for chunking: First, the number of pulses $n$ ranging between 1 and 120. Secondly, the number of distinct parts $k$ can range between 1 and 5. The output of chunking consists of a series of recursive partitions of $n$ into $k$ parts. A partition of an integer $n$ is simply any sum of integers $< n$ resulting in $n$, for example $16 = 7 + 5 + 4$, which is a partition into three distinct parts ($k = 3$). For the reminder of the paper, partitions will be written as (multi-)sets of integers with subscript $n$.[5] Chunking creates a single musical sentence of length $n$ by partitioning $n$ into $k$ phrases. Phrases are further partitioned into at most $k$ patterns, which finally partition into chunks of 2s and 3s. The proposed hierarchy of all musical components is: $Sentence, Phrases, Patterns, Chunks$. The algorithm is designed to make sure that the lengths of the components on each level are always co-prime. For example, $Sentence\{19\}, Phrases\{10, 9\}_{19}, Patterns\{\{7, 3\}_{10}, \{4, 5\}_9\}_{19}$, $Chunks\{\{\{3, 2, 2\}_7, \{3\}_3\}_{10}, \{\{2, 2\}_4, \{3, 2\}_5\}_9\}_{19}$. After experimentation it was found that the co-primality between individual parts of a rhythmic sentence is a condition, which yields the best musical results amongst partitions because of

[4] The term Aksak is Turkish and means 'limping'

[5] For example, $\{7, 5, 4\}_{16}$ is equivalent to writing $16 = 7 + 5 + 4$.

the high amount of rhythmic impulse and resolution (tension and release) that can be generated. With the same reasoning, the algorithm searches for the partition whose distribution of distinct parts is as close as possible. This is achieved by making sure that the standard deviation, $\sigma$, is as small as possible. If all distinct parts are in an arithmetic progression with $\Delta = 1$, for example: $\{7, 6, 5\}_{18}$, then $\sigma \approx 0.816497$ reaches its smallest value amongst all partitions of 18 with 3 distinct parts (of course, for other $n$, $\Delta$ and $\sigma$ can be very different). In this little example, all conditions for searching a partition have been met: Co-primality of distinct parts, and the lengths of the parts are closest together. Partitions like these are the basis for the unique phrase and pattern lengths within the sentences.

Because the parts in all partitions are ordered from high to low integers, *chunking* reorders them into a musical triangle. A musical triangle is a special order of values where the largest one is in a central position, and the smaller values are leading up towards the central peak, and descending thereafter. For example, $\{1, 2, 4, 3\}$, or $\{2, 3, 4, 6, 5, 1\}$. There are fourteen musical triangles of this kind for the reordering of all partitions with three to seven distinct parts. Partitions with only two parts are being swapped randomly.

The next iteration uses these re-ordered partitions and constructs the lower structural level: patterns *within* each of the phrases. A new partition with distinct parts is generated, one that divides up each phrase length, thus generating a pattern. Subsequently, for each of the pattern, the re-ordering process applies again one of the musical triangles.

The final step towards the complete sentence is to take each of the patterns and to partition them into chunks of two or three pulses in length. 2 and 3 are the only integers allowed on this level of partitioning. If, for example, a pattern has the length 7, then the only partition into 2s and 3s is $\{3, 3, 2\}_7$.[6] Larger $n$ can produce many different partitions of this kind. In order to distinguish these partitions from the ones we have discussed so far, we call them *templates*, where 2s and 3s are the only *chunks* accepted. In the next step, one can *rotate* the templates in a cyclic manner, i.e. $\{3, 3, 3, 2\}_{11}, \{3, 3, 2, 3\}_{11}, \{3, 2, 3, 3\}_{11}$, and $\{2, 3, 3, 3\}_{11}$ are all rotations of the template $\{3, 3, 3, 2\}_{11}$. A template produces a unique set of combinations. Each one of the combinations has its consequences on the perception of the rhythm in a musical context, and it falls therefore into a special musical category.

We have reached the lowest level of the hierarchy, from a sentence to phrases, to patterns, to chunks. Patterns and phrases are organized according to the sets of musical triangles, like smaller waves leading to a larger one. With regard to the *templates*, i.e. partitions into 2s and 3s, *chunking* uses several categories of combinations that are useful for music composition. It is an interesting fact that the *templates* are equivalent to a special mathematical structure called *bracelets*[4].[7] A bracelet is a cyclic pattern with a special order, and although it

[6] For a partition, the order of the terms is irrelevant.

[7] To form a bracelet, the template itself has to be in its lexicographically lowest rotation.

can be rotated and also reversed, it would never be equal to any other bracelet (or template) that has the same sum of its elements, including under rotations and reverse directions. In general, every partition of $n$ into 2s and 3s, and after rotation into its lowest lexicographic position, generates a bracelet with a fixed content.[8] In order to compose music using the large amount of possible rhythmic patterns that emerge from templates, they had to be sorted into seven different musical categories. The names of these categories reflect the musical forces of impulse and resolution that govern them and that are a direct consequence of the specific order of 2s and 3s.

### 2.1 Seven categories of rhythmic patterns

The seven categories of rhythmic patterns are *resistor, release, arch, catenary, growth, decline,* and *alternating.* The *resistor* pattern consists of one or more 2s followed by one or more 3s. For example, here is a *resistor* based upon a partition of 12: $\{2, 2, 2, 3, 3\}_{12}$. The name reflects the phenomenon that the transition from a pulsation of 2s into a pulsation of 3s forms a resistance against the flow of the binary pulsation.[9] We experience a fundamental change from binary to ternary pulsation. This phenomenon creates musical impulse, a force that is balanced at a later stage by a corresponding resolution.

The *release* pattern is the opposite form of the above: One or more 3s are followed by one or more 2s, for example $\{3, 2, 2, 2\}_9$, which is derived from a partition of 9. If one combines the *resistor* patterns with a *release* pattern, you obtain the *arch* pattern. This is a natural consequence of the fact that a musical impulse is counter-balanced by a force of release, similar to the situation where the force of pulling a spring out of its equilibrium position is opposed by the spring-force, which will return the system back to its original state as soon as the spring has been released. We call this configuration *arch*, because one or more 2s are surrounding one or more 3s. Example: $\{2, 2, 3, 3, 3, 2, 2, 2\}_{19}$ As it is the case with all the other patterns, the length of an *arch* can vary.

The shape opposite to the *arch* is the *catenary*[10], which is a combination of the *release* pattern followed by the *resistor* pattern. It has one or more 3s surrounding one or more 2s, for example: $\{3, 3, 2, 2, 2, 3\}_{15}$.

A *growth* pattern has, for example, this structure: $\{2, 3, 2, 2, 3, 2, 2, 2, 3\}_{21}$. Here, an initially small resistor pattern, $\{2, 3\}_5$, grows by inserting more and more 2s in front of the 3 at each repetition. There are many possible variations of this pattern, for example, one could let only the 3s grow: $\{2, 3, 2, 3, 3, 2, 3, 3, 3\}_{24}$. Or, one could develop both at the same time: $\{2, 3, 2, 2, 3, 3\}_{15}$. The maximum length for a growth pattern in the top-down approach is twenty-five pulses with

[8] An efficient algorithm for generating bracelets has been published by Karim et al.[4].

[9] Inspiration for the name came from the field of Electronics with regard to the resistance against the flow of electric charges.

[10] "In physics and geometry, a catenary is the curve that an idealized hanging chain or cable assumes under its own weight when supported only at its ends." (see fx-solver.com)

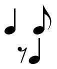
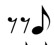
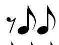
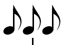
a maximum of nine elements. This is a consequence of applying Miller's 'magic number'. It is easy to see that one could chain resistor patterns together in order to form a phrase or an entire sentence as one large *growth* structure.

The *decline* pattern is like the *growth* pattern, only the direction is reversed. We start with a relatively long pattern, and recursively subtract elements from it. Example: $\{2,3,3,3,2,3,3,2,3\}_{24}$. Finally, the *alternating* pattern consists of repeating a smaller pattern of 2s and 3s, for example $\{3,2,2,3,2,2\}_{14}$.

## 2.2 Shorthand notation for rhythm as an intermediate output format

On the basis of the resulting 2-3 patterns discussed above, a technique is now presented for breaking down the binary and ternary chunks into specific rhythmic chunks. *Chunking* uses all possibilities of beats within a grid of either 2 and 3 pulses. The complete list of binary and ternary rhythm chunks is shown in table 1 along with binary and shorthand representations. For each 2 and 3 in a pattern, a weighted random choice is being made from the list of binary or ternary chunks. The probability weighting of the chunks is a matter of personal choice. After experimentation it was found that the probability of chunks that start with a rest had to be reduced, otherwise there would have been too much dissociation between the rhythmic patterns.

**Table 1.** Set of symbols for rhythmic chunks.

| Symbol | Transcription | Binary Form | Category |
|---|---|---|---|
| . | | 1 | unary |
| I | | 10 | binary |
| : | | 11 | |
| v | | 01 | |
| X | | 110 | ternary |
| > |  | 101 | |
| < | | 010 | |
| w |  | 001 | |
| + |  | 011 | |
| i |  | 111 | |
| − | | 100 | |
| ˜ | e.g. I ˜ I = | 1000 | tie |
| ( ) | e.g. (X) I = | 00010 | silence |

The shorthand notation is a useful intermediate output of *chunking.* The notation is human readable and can be sent as a C++ string to an appropriate translator object that decodes the notation and generates automatically code for csound score events listed under the score section of a csd file. The csound #include directive is very useful here. The translator is also capable of generating a lilypond file, for which we can give a practical example in figure 1. It has the following structure: $Sentence\{37\}, Phrases\{19, 18\}, Patterns\{\{10, 9\}, \{7, 11\}\}$, $Chunks\{\{\{3, 3, 2, 2\}, \{2, 2, 2, 3\}\}, \{\{2, 2, 3\}, \{2, 2, 3, 2, 2\}\}\}$

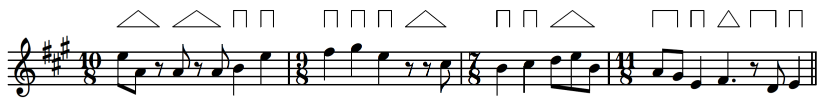


**Fig. 1.** A sentence with 37 pulses generated with *chunking* and rendered with *lilypond*. The conductor signs show the metric grouping in beats that are 2 or 3 pulses long.

## 3 The Csound score output

In order to add a musical performance quality to the synthesis, *chunking* adds phrase envelopes and an overarching sentence envelope. Phrase envelopes modify the overall amplitude of a phrase, and are calculated to reach a climax at the beginning of the second half of the phrase. The *expseg* envelope is used in the orchestra code.[11] The sentence envelope modifies the overall amplitude of the sentence and uses the *cosseg* opcode. Its climax is reached at the point when the first half of the phrases has just ended (rounded up for odd numbers of phrases). For example, if the sentence has five phrases, the envelope reaches its climax at the beginning of the fourth phrase. The time parameters for the envelopes are: total length, ascending, and descending phrase lengths in seconds. In order to control independent phrase and sentence envelopes, their effect is programmed into separate instruments, away from the sound generating source. This modular approach uses *Csound*'s global audio variables.

## 4 Conclusion

*Chunking*'s algorithm creates rhythmic sentences that subdivide into phrases, patterns of binary and ternary chunks, and it generates the specific composition of rhythms based on these chunks. The algorithm uses a top-down approach based on partitions, bracelets and seven musical categories of patterns. It adheres to Miller's magic number on multiple levels and it uses the concept of chunks and templates for musical variation. It further generates musical phrasing envelopes for Csound synthesis. The output in a general notation format helps to navigate choices during the composition process. That process does not stop with the output of *chunking*, but uses its material to compose entire pieces of music.

[11] Example *csd* files are provided with the source code.

## References

1. Arom, S.: L'aksak: Principes et typologie. Cahiers de Musiques Traditionnelles 17 (Formes musicales): 11–48 (2004)
2. Brăiloiu, C.: Le rythme Aksak. Revue de Musicologie 33(99 and 100) (December), 71–108 (1951)
3. Gobet, F. and Simon, H. A.: Templates in Chess Memory: A Mechanism for Recalling Several Boards. Cognitive Psychology 31(1), 1–40 (1996)
4. Karim, S. et al.: Generating bracelets with fixed content. Theoretical Computer Science, 475, 103-112 (2013)
5. London, J.: Hearing in Time. Psychological Aspects of Musical Meter, 2nd edn. Oxford University Press, Oxford (2012)
6. Miller, G. H.: The Magical Number Seven, Plus Or Minus Two: Some Limits on Our Capacity for Processing Information. Psychological Review, 63, 81–97 (1956)
7. Nierhaus, G.: Algorithmic Composition: Paradigms of Automated Music Generation, Springer, Wien (2010)
8. Phillips, D.: Introducing CsoundAC: Algorithmic Composition With Csound And Python. Linux Journal [online] (2010), `http://www.linuxjournal.com/content/introducing-csoundac-algorithmic-composition-csound-and-python`
9. Taube, H. K.: Notes from the Metalevel, Taylor & Francis, London and New York (2004)
10. Thakar, M.: Looking for the "Harp" Quartet: An Investigation into Musical Beauty, University of Rochester Press (2011)
11. Yi, S.: blue: a music composition environment for Csound `http://blue.kunstmusik.com`