Paul Nauert
Music Department
University of California
1156 High Street
Santa Cruz, California 95060 USA
dogsbark@cats.ucsc.edu

Division- and Addition-Based Models of Rhythm in a Computer-Assisted Composition System

Reading descriptions of different musical traditions or of the practice of different individual composers, it is not uncommon to see a rhythmic idiom described as "additive," meaning in simple terms that it consists of durations formed by combining, or adding together, the units provided by a rapid underlying pulse (Koetting 1970; Pratt 1987; Clayton 2000). Such a characterization distinguishes the rhythmic idiom from others involving the division of the units of some referential pulse into variable numbers of smaller parts, an important feature of tonal Western classical music (Lerdahl and Jackendoff 1983). My present concern is not the ability of addition- or division-based models of rhythm to accurately describe existing musical traditions, but the value of these models as a basis for generating rhythms in an algorithmic-composition system, in which potentially novel rhythmic idioms may be sought. In particular, I will describe a series of algorithms I have used in my own work as a composer, and I will investigate some of the advantages and disadvantages each of these algorithms inherits from the particular model of rhythm on which it is based. The development of these algorithms was shaped by at least two compositional goals—to create music for human performance, and to express it using common-practice Western music notation (Byrd 1994). My investigations will be made in the context of the same goals.

For the sake of making the distinction between the two types of models clearer, Figure 1 presents simple illustrations of rhythms generated by purely addition-based and purely division-based means. In Figure 1a, the upper row of the diagram represents a stream of timespans constituting the units of some isochronous pulse. The lower row of this diagram shows these timespans grouped into durations of 3, 5, 3, 2, 5, and 2 units to constitute the resulting rhythm. (These values are labeled in the diagram, but of course we could also count unit pulses to determine them.) Although the resulting rhythm lacks the isochrony of the original unit pulse, we can treat it like another pulse and combine adjacent timespans to form larger durations. In this way, the additive process is capable of forming hierarchies of timespans deeper than the two levels illustrated here.

Figure 1b provides a contrasting illustration of purely division-based rhythm generation. In this case, the process begins with a single large timespan, represented by the top row of the diagram. The second row shows a division of this timespan into three parts whose durations are related by the chain of proportions 3:5:3. Each of the resulting timespans is divided into smaller parts in the third row of the diagram, and portions of the structure extend to a fourth level according to the same principle.

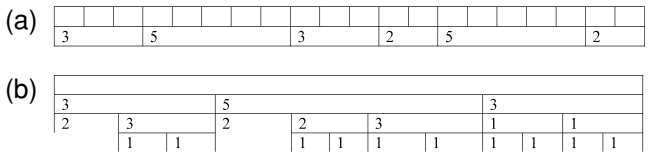
Simple though they may be, both of these models are capable of generating a variety of rhythmic surfaces. Indeed, the question arises whether the range of possible outputs from one process is identical to that of the other and, if so, how meaningful the difference between the models might be.

Figure 2 transcribes the addition- and division-based results into conventional music notation. In each case, the transcription reflects the generative process. In Figure 2a, the transcription of the addition-based rhythm associates a simple notational unit (eighth notes) with the unit pulse, inviting performers to count that pulse to execute the rhythm accurately.

Transcription of the division-based rhythm taxes our notational resources slightly more, requiring various tuplets to show how the timespans expressed as measures are divided into various numbers of equal parts. Some of the operations involved in producing Figure 2b are addition-based: The top-level timespan is not assigned any special representation, but it is formed by the sum of the measure lengths,

Computer Music Journal, 31:4, pp. 59–70, Winter 2007

Figure 1. Simple models of (a) purely addition-based and (b) purely division-based rhythm.



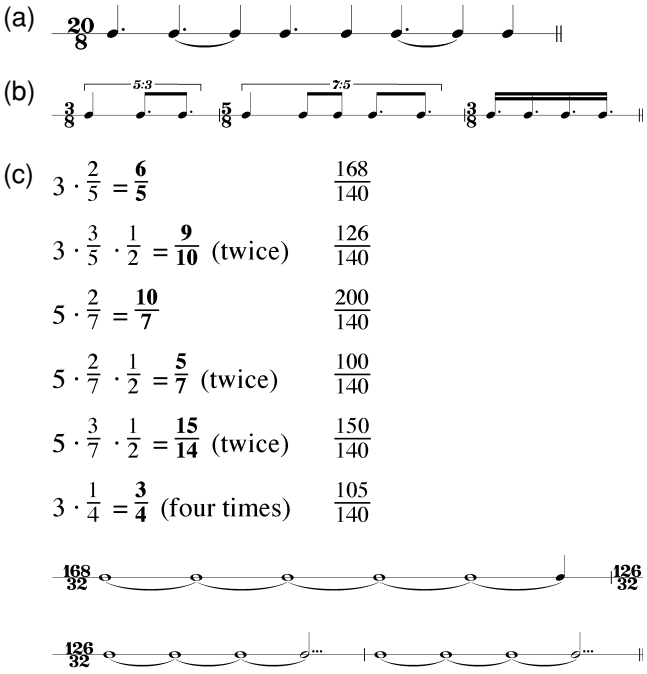
and the tuplet-derived pulses within each measure are grouped into larger timespans that can be divided according to the next stage of the division-based model. For instance, the 2:3 division at the beginning of the third-largest level implies both a division into five parts and an additive grouping of those parts into 2 and then 3. Although the resulting notation is relatively complex, it expresses the rhythm using a combination of divided timespans and grouped pulses that are reasonably accessible to an experienced performer, whose main challenge will be learning to change pace from the five pulses of the first measure to the seven of the second to the four of the third.

We can re-notate the division-based rhythm by finding a pulse that divides evenly into all of its constituent durations. Figure 2c shows the necessary calculations and a portion of the results notated relative to a thirty-second-note unit pulse. Of course, these calculations suggest how the same rhythm could be generated as well by a purely additive process. However, if the tempo of this last result is fast enough to reproduce the absolute durations we probably imagined when reading Figure 2b, then the unit pulse will be much too fast for a performer to count as a basis for executing the rhythm.

These observations suggest that addition- and division-based processes are more than different means to equivalent ends, at least in cases where human performance is involved. Certain results are idiomatic to each model, and in cases that respect these idioms, the models are not just responsible for producing the rhythm but may also inform how the rhythm is expressed in notation and how a performer responds to that result. These are the issues I shall be probing in what follows, as I consider how three rhythm-generating processes can engage the addition- and division-based aspects of conventional music notation and the associated strategies that are accessible to experienced performers.

I should state up front that my assumptions about these performance strategies are informed more by my own experience as a composer and performer than by the results of any formal experiments. The techniques that expert performers develop to overcome the challenges presented by rhythmically complex music are diverse and potentially idiosyncratic (Weisberg 1993; Schick 1994; Cox 2002). Of course, many of these techniques mirror the same underlying cognitive constraints, and my argument is informed at points by results from music psychology studies, many of which have been summarized in recent work by Justin London (2004).

Figure 2. (a) Transcription of the addition-based rhythm from Figure 1a; (b) transcription of the division-based rhythm from Figure 1b; (c) conversion of (b) to an addition-based representation. In (c), the first three durations are expressed in "additive" form (with a unit pulse of a thirty-second note).



An Addition-Based Model in OMTimePack: *m-rhythm*

Figure 3 presents a simplified version of an algorithm I have employed to generate rhythmic streams for use in composition regularly over the past fifteen years or so. There have been various implementations of this algorithm, the latest of which is the `m-rhythm` function in the current version of OMTimePack (OMTP) (Nauert 2006). In `m-rhythm`,

Figure 3. Simplified `m-rhythm` *algorithm.*

Figure 4. Generating an idiomatic additive rhythm with `m-rhythm`*: sample output.*

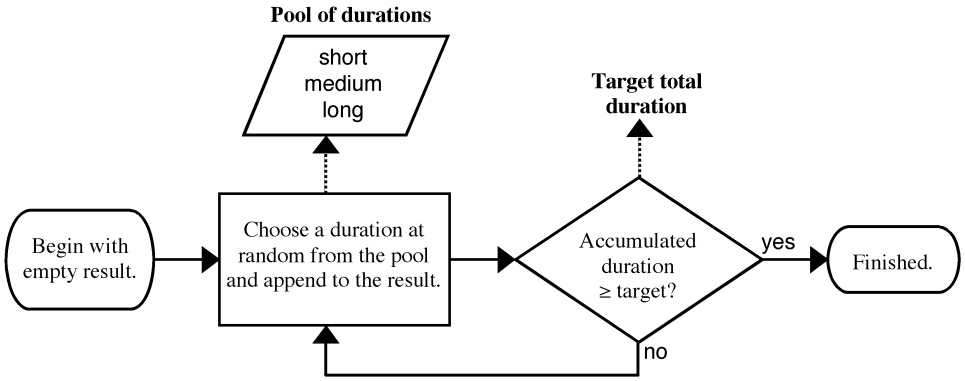


Figure 3

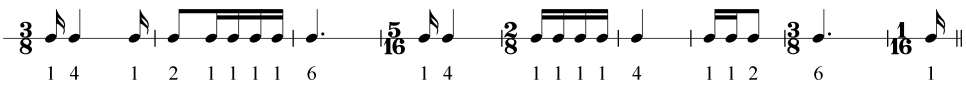


Figure 4

Table 1. Durations and Associated Probabilities of Figure 4

Duration	1	2	4	6
Probability (%)	65	18	10	7

the random-selection process is a Markov chain of order zero, one, or two. The user specifies probabilities in a list or transition matrix manually, in addition to specifying a pool of possible durations and a target total duration.

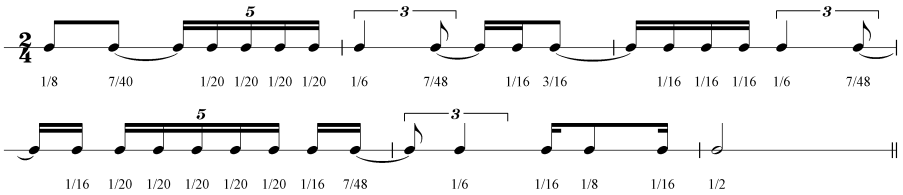
In either its simplest form or in more elaborate ones, the rhythm-generating process illustrated in Figure 3 is implicitly additive. This feature becomes clearer if each duration in the pool is expressed as an integer value; and when all the integer values are relatively small, we have the "idiomatic" case in which durations are multiples of a unit pulse that performers can count. Figure 4 and Table 1 shows the operation of `m-rhythm` under these conditions. Here, the pool consists of the four durations (1, 2, 4, 6), each associated with a static probability, and the output is transcribed using a sixteenth note as the unit pulse. But how does this algorithm perform when it is configured to produce results outside the range of idiomatic addition-based rhythms?

Consider a simple rhythm, depicted in Figure 5a in a form that clearly engages a performer's ability to execute various regular pulses by dividing a larger referential pulse. Table 2 is an inventory of inter-onset intervals representing the rhythm's "duration vocabulary." These are expressed as fractions of a whole note, which makes power-of-2 divisions such as eighth notes immediately recognizable as multiples of the slowest unit pulse (namely, 1/120 of a whole note) that can accommodate the entire vocabulary. This confirms that the rhythm lies beyond the idiomatic reach of a purely additive model, because it is not practical to base performance timings on, for example, the fifteen-unit pulses constituting the first eighth note, assuming a moderate tempo for this passage.

Figure 5b provides a transition table based on a first-order Markov analysis of the original rhythm. (The analysis treats the concluding half note as if it wrapped around to the beginning of the rhythm, so that this note-value has a successor.) Figure 5c shows a representative output from `m-rhythm` using the established duration vocabulary and analysis-based transition table, and Figure 5d transcribes the beginning of this new result. Although the original rhythm and this newly generated one employ the

Figure 5. (a) Simple rhythm; (b) transition table produced by first-order Markov analysis of original rhythm; (c) sample output from `m-rhythm` *using the established duration vocabulary and transition table; (d) transcription of a portion of this output.*

(a)



(b)

Probability of next event (percent values)

Current event	1/20	1/16	1/8	7/48	1/6	7/40	3/16	1/2
1/20	78	11	0	0	11	0	0	0
1/16	12.5	25	12.5	12.5	12.5	0	12.5	12.5
1/8	0	50	0	0	0	50	0	0
7/48	0	67	0	0	33	0	0	0
1/6	0	33	0	67	0	0	0	0
7/40	100	0	0	0	0	0	0	0
3/16	0	100	0	0	0	0	0	0
1/2	0	0	100	0	0	0	0	0

(c)

```
(1/16 1/2 1/8 1/16 1/8 1/16 1/6 7/48 1/16 1/20 1/20 1/20 1/20 1/20 1/20 1/20 1/20
1/20 1/6 7/48 1/16 1/16 1/6 1/16 1/16 1/16 1/2 1/8 7/40 1/20 1/20 1/20 1/20 1/20
1/16 1/6 7/48 1/6 7/48 1/16 1/16 1/16 1/2 1/8 7/40 1/20 1/20 1/20 1/20 1/20 1/20
1/20 1/20 1/20 1/20 1/20 1/20 1/20 1/20 1/6 7/48 1/16 3/16 1/16 1/16 7/48 1/6 7/48
1/16 1/16 1/16 1/20 1/20 1/20 1/20 1/16 3/16 1/16 1/6 1/16 1/6 7/48 1/6 7/48 1/6
1/16 7/48 1/16 1/16 1/8 1/16 1/6 7/48 1/16 1/20 1/20 1/6)
```

(d)

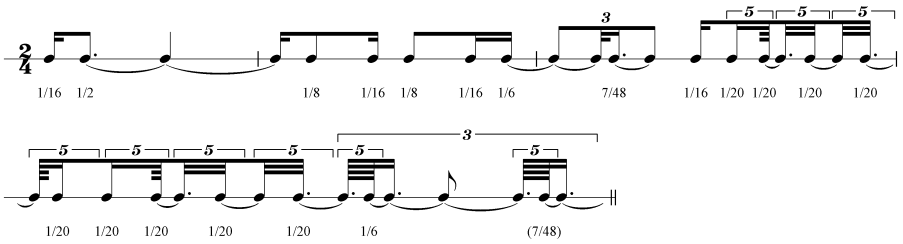
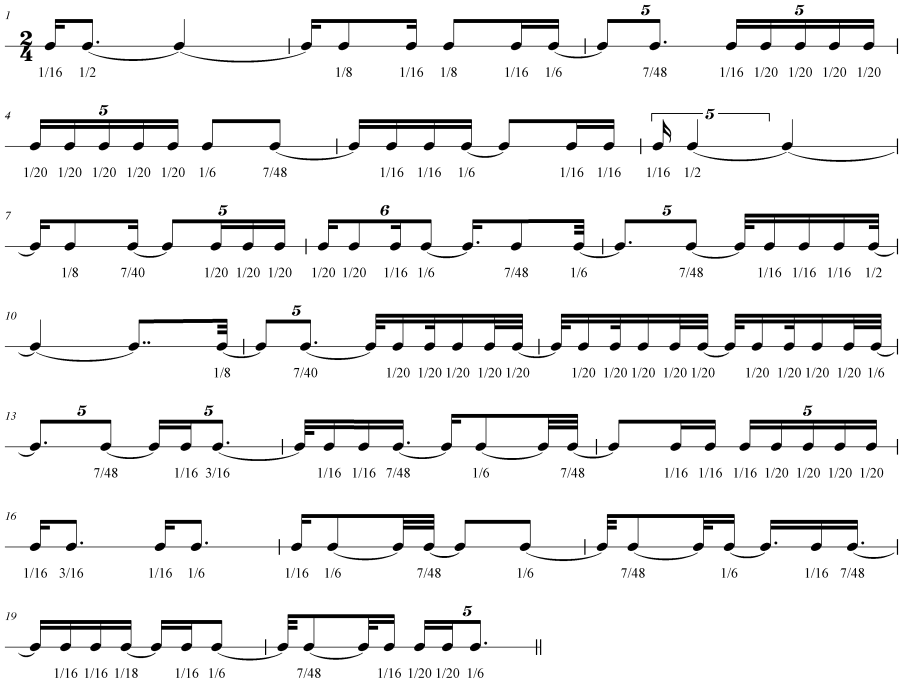


Table 2. Duration Vocabulary of Figure 5

Duration (fraction of a whole note)	1/20	1/16	1/8	7/48	1/6	7/40	3/16	1/2
Duration (as integer multiples of 1/240 of a whole note)	12	15	30	35	40	42	45	120

Figure 6. The rhythm from Figure 5c, quantized using the `omquantify` *routine in OpenMusic, with original (unquantized) values shown below the staff.*



same duration vocabulary and are statistically similar, clearly the arrangement of durations is less idiomatic in the latter case, leading to results that are far more notationally complex and difficult to perform (Perkins 1965).

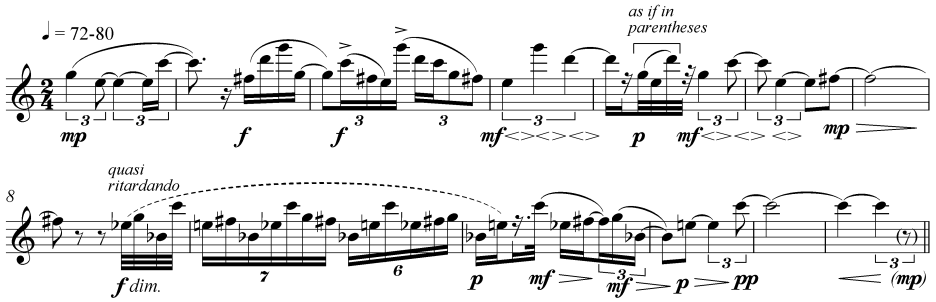
Quantization

Faced with the potential complexity of output from `m-rhythm`, one natural strategy is to simplify the results afterward, with the goal of reducing performance difficulty without significantly distorting the stream of durations. This is essentially a quantization problem, for which a variety of solutions are known (Agon et al. 1994; Nauert 1994). In simple cases, quantization may operate by resolving durations to a fixed isochronous grid, in which case the solutions are relatively coarse addition-based approximations of the rhythm. More sophisticated quantization strategies may select among a number of alternative grids to resolve different portions of the rhythm with minimal distortion. Because these latter strategies engage a performer's ability to divide a tactus into a variety of smaller pulses, they can be said to involve a division-based model of rhythm.

One such quantization process is the `omquantify` routine built into IRCAM's OpenMusic platform (Assayag, Baboni, and Haddad 2001). Figure 6 shows the result of processing the rhythm from Figure 5c with `omquantify`. The `max/` parameter in `omquantify` determines the largest number used for dividing beats into equal parts to produce candidate grids, and the `precision` parameter controls a trade-off between simplicity (favoring sparser grids) and accuracy (favoring smaller round-off errors). Leaving `max/` at its default value of 8, while boosting `precision` slightly, produces the result shown here. To the extent that this quantized rhythm is easier to perform than the original, yet sounds comparable to it, the quantization process has succeeded.

There is, of course, some loss of precision even in successful cases of quantization. Note for instance the lack of differentiation between regular and quintuplet sixteenth notes—that is, between 1/16 and 1/20—in measure 3 of Figure 6. Whether effects like these constitute a problem depends on a composer's

Figure 7. Nauert,
Arabesque, *mm. 1–13.*



particular commitments. For me, the appeal of `m-rhythm` is its ability to create output streams with distinctive "rhythmic behaviors" based on a relatively compact pool of 5–10 durations and appropriate first-order Markovian probabilities—slowly drifting motion with sporadic bursts of rapid activity, fast runs broken from time to time by an intervening long note, and incremental decelerations with occasional injections of fresh kinetic energy. The last of these behaviors is evident in the opening of my 1995 composition *Arabesque* for solo flute, shown in Figure 7. Table 3 shows the duration pool and transition table used to generate these rhythms. An expressive performance of this music will capture the gestural quality of its "deceleration" figures, an approach that various performance indications (and, if memory serves, some hand-tailoring of the computer-quantized durations in measures 8 and 9) are meant to encourage. Given my interest in the gestural/behavioral qualities of rhythm, small quantization errors are not a critical issue.

If the generate-and-quantize paradigm is limiting, it is not really a matter of imprecision in the quantization stage. The problem instead is the disconnect between the generative method (in which division-based principles play no part) and the notational apparatus that makes results accessible to performers (in which division-based principles play an important role), particularly when results featuring regular pulsation are sought. For instance, if I wish to produce a musical surface in which quintuplet-sixteenth-note runs occur frequently, the best I can do using `m-rhythm` followed by `omquantify` is to include a quintuplet-sixteenth note in the pool, set probabilities in the transition table so that this element is its own most likely successor, and hope

Table 3. Transition Table Used to Generate the Rhythm of Figure 7 Given for a Duration Pool of 0.15, 0.3, 0.6, 1.2, 4 (Expressed in Relation to a Quarter-Note Unit)

Current Event	*Probability of Next Event (%)*				
	0.15	*0.3*	*0.6*	*1.2*	*4*
0.15	70	30	0	0	0
0.3	0	70	30	0	0
0.6	20	0	40	40	0
1.2	25	0	0	50	25
4	80	20	0	0	0

that runs of this value in the raw output align with an available quantization grid in such a way that they appear as quintuplet sixteenths in the notated result. (In the rhythm of Figure 6, I was fortunate in measures 3 and 4, less so in measures 11 and 12.)

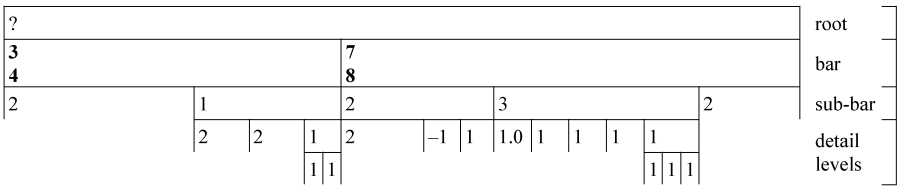
An alternative way of generating rhythms with a Markov process is to use a pool of cells or motives rather than single durations (Jones 1981). Various division-based features, such as contrasting tuplets, can be embedded in these cells, and as long as the length of each cell is an integral number of beats, then assemblages of them can be constructed with no need for subsequent quantization. This capability is built into `m-rhythm`, but I have rarely made use of it. Assembly-with-cells saturates the output rhythm with copies of a fixed number of motives, and I prefer to work with material that is more fluid and heterogeneous. These priorities, coupled with an awareness of the limitations of generate-and-quantize, described previously, have led me to explore various alternatives to `m-rhythm`; the

Figure 8. Rhythm tree (`rtree`) in (a) list and (b) graphical form, together with (c) conventional notation of the same rhythm.

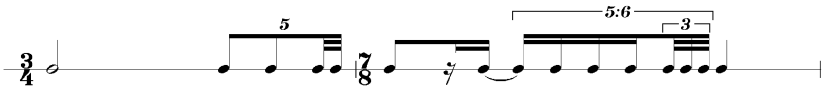
(a)

```
(? (((3 4) (2 (1 (2 2 (1 (1 1))))))
                ((7 8) ((2 (2 -1 1)) (3 (1.0 1 1 1 (1 (1 1 1)))) 2))))
```

(b)



(c)



remainder of this article focuses on two of these explorations.

What both of these alternative algorithms have in common with `m-rhythm` is that they build streams of durations according to a low-order Markov process. Whereas `m-rhythm` is essentially an addition-based process that required an auxiliary stage of division-based quantization, the first of these alternative methods (embodied in a largely abandoned project I called `m-rtree`) is intrinsically division-based, and the second (embodied in a family of functions that coexist with `m-rhythm` in the latest version of OMTP) is addition-based but involves conditions that ensure the possibility of an idiomatic division-based translation without quantization.

A Division-Based Approach

As I noted earlier, the `omquantify` routine in OpenMusic resolves rhythms to a grid that is flexible rather than fixed. To represent these division-based results, OpenMusic employs a data structure known as a "rhythm tree," or `rtree` for short. The `rtree` structure originated in Mikael Laurson's PatchWork (Laurson 1996; Mazzola 2003, pp. 976–978). An `rtree` represents a rhythm as a nested hierarchy of timespans; Figure 8a provides an example in the nested-list form used in OpenMusic. The same rhythm appears in an easier-to-read graphical form in Figure 8b and in conventional music notation in Figure 8c; these parts of the figure are aligned with one another to facilitate comparison.

Specific levels of an `rtree` have fixed musical meanings. The largest level, called the root level, is a single timespan extending from the beginning to the end of the rhythm. The children of the root level form the bar level, which consists of successive bars or measures. The children of these bars form the sub-bar level, which partitions the bars into beats or other, less symmetrical parts. OpenMusic expects to see root, bar, and sub-bar levels in every `rtree`; in addition, one or more smaller levels, called detail levels, may be present. In the nested-list form used by OpenMusic, an `rtree` is simply a pair $(D\ L)$, where D expresses the duration of the `rtree`, and L lists its contents in chronological order. Each element of L is either a smaller `rtree` $(D'\ L')$ or an empty timespan with duration D'. The form in which durations are expressed depends on the level at which the duration occurs. There are three cases.

The duration of a bar-level timespan is expressed as a pair of positive integers $(q\ r)$ representing the ratio q/r in relation to a whole-note unit. If r is a small power of two, then $(q\ r)$ symbolizes a conventional time signature q/r. The 3/4 and 7/8 time signatures in Figure 8 exemplify this feature. The duration of a sub-bar- or detail-level timespan is expressed as a positive integer k indicating how much of the parent timespan it occupies in proportion to its siblings. For example, the 3/4 bar in Figure 8 has two

children whose durations are expressed as 2 and 1, corresponding to sub-bar timespans of a half note and a quarter note, respectively. Finally, the duration of the root-level timespan can be computed by adding the durations of its bar-level children. The question-mark symbol ("?") is a customary placeholder for this sum.

Because functions that process an `rtree` always expect to see durations expressed as positive integers at the sub-bar and smaller levels, they know to interpret other types of numbers as positive integers that carry special coding. Specifically, negative integers are used to encode rests, and floating-point expressions are used to encode ties. Both of these features can be seen in the detail levels of Figure 8. These special codings exploit the handling of numerical data in Lisp, the programming language on which OpenMusic is based.

As I suggested earlier, OpenMusic's quantization function, `omquantify`, operates by matching lists of durations to a repertoire of `rtree` structures, each of which represents a different simplifying grid. During a period of research in 2003, I investigated the possibility of working with a stochastic process that loosely resembles my `m-rhythm` function but operates by constructing `rtree` expressions directly. Specifically, I devised a function `m-rtree` that operates on a list of empty measures or a fully structured `rtree` by embedding randomly selected patterns into its leaves. For instance, given the root and bar levels of the `rtree` in Figure 8, and given a pool that includes the patterns (2 1) and (2 3 2), `m-rtree` could create the sub-bar level of the same `rtree`. Additional applications of `m-rtree`, using suitable pools of patterns and statistical controls, could produce the detail levels of this `rtree`, generating the rhythmic result depicted in Figure 8c. Although the association of the output tree-structure with a specific notational representation is novel in `m-rtree` (to the best of my knowledge), there have been previous efforts to generate rhythms by recursively subdividing timespans, with the patterns of subdivision variously based on timings derived from speech (Reynolds 1965, analyzed in Harker 1994), derived from a twelve-tone series (Wuorinen 1979), or randomized (Ames 1982).

I quickly discovered it was necessary to supplement the Markov process that drives pattern selection in `m-rtree` with additional rules to limit the complexity of the resulting `rtree`s. Typical rules offer user-configurable control over the depth to which tuplets are nested, the ratios relating tuplet divisions in successive timespans, and the absolute density of onsets within a given timespan. Recall that `m-rtree` operates by traversing the leaves of an `rtree` and embedding a randomly selected pattern (or none) in each leaf.

To reconcile this stochastic process with hard constraints on complexity, `m-rtree` behaves as follows. At each step of the Markov process, each pattern is evaluated in relation to the complexity rules, and patterns that fail to satisfy one or more constraints (for instance, patterns that would cause excessively deep nesting of tuplets) have their probabilities reset to zero. The probabilities resulting from these adjustments control a random choice, and then the Markov process advances. Note that if every pattern is ruled out at a particular step, the current leaf of the `rtree` is simply left alone (no pattern is embedded in it); thus the process never encounters "dead ends" that would necessitate a technique like backtracking.

The deterministic rules in this system tend to dominate the stochastic ones, making the behavior of `m-rtree` less like that of `m-rhythm` than I had hoped. There are probably less clumsy ways of reconciling the Markovian and constraint-solving aspects of such a system; in particular, constraint-solving tools bundled with OpenMusic (Bonnet and Rueda 1999; Sandred 1999) might accommodate stochastic extensions. But at its core, `m-rtree` is a pattern-based system, and even if it were made to resemble `m-rhythm` more closely, it would be the assembly-with-cells form of `m-rhythm`, rather than the configuration based on a pool of single durations that I prefer. As a result of its current and prospective limitations, I have essentially abandoned the `m-rtree` approach to rhythm generation. It is perhaps worth noting that a related compositional system, based not on constraint-solving but on permutations and other, more idiosyncratic reconfigurations of `rtree` structure, has been explored by the composer Brian Ferneyhough in several works since the mid 1990s (Malt 1999). In Mr. Ferneyhough's

work there is less automation of the process and a greater tolerance for complexity in the results compared to the goals of my project.

Extensions to OMTP: An Addition- and Division-Based Hybrid

A side effect of these experiments with `rtrees` was a greater appreciation (on my part) of the difference between two types of timespan-partitioning patterns. On one hand, patterns that consist of strings of ones produce results that are locally grid-like. In Figure 8, for instance, the quintuplet in the middle of the second bar arises from a string of five ones (the first encoded to act as a tie), and the triplet embedded into the final unit of this quintuplet is another grid-like feature on a smaller level. Rhythms based on equal-value strings of this type invite performers to make maximum use of their ability to divide larger timespans into various regular pulses.

On the other hand, patterns containing unequal values yield rhythms that imply an addition-based organization of some more rapid underlying grid. In Figure 8, for instance, a performer is likely to execute measure 1, beat 3, by fitting the (2 2 1) pattern to a grid of quintuplet-sixteenth-note pulses, which arise, in turn, from division of a quarter-note timespan into equal parts. And on a larger level, the (2 1) pattern that spans all of measure 1 implies a particular additive grouping of quarter-note pulses.

Justin London's work on the psychology of musical meter is relevant here in at least two ways. First, London (2004) has argued that even in the case of recurrent metrical designs, patterns involving beats of unequal length depend for their comprehension on coordination between the level of the non-isochronous beats and a smaller level of isochronous pulses, although he resists calling this relationship "additive" to stress that both levels of the metrical structure belong to a single psychological gestalt. And second, London contends that the relationship between varied durational surface and regular underlying grid loses significance outside a certain range of time scales. In particular, he estimates the lower limit on the grid period to be approximately 100 msec in duration. (At a heart-rate tempo of 72 quarter-note beats per minute, this "metric floor" corresponds approximately to a thirty-second-note pulse.)

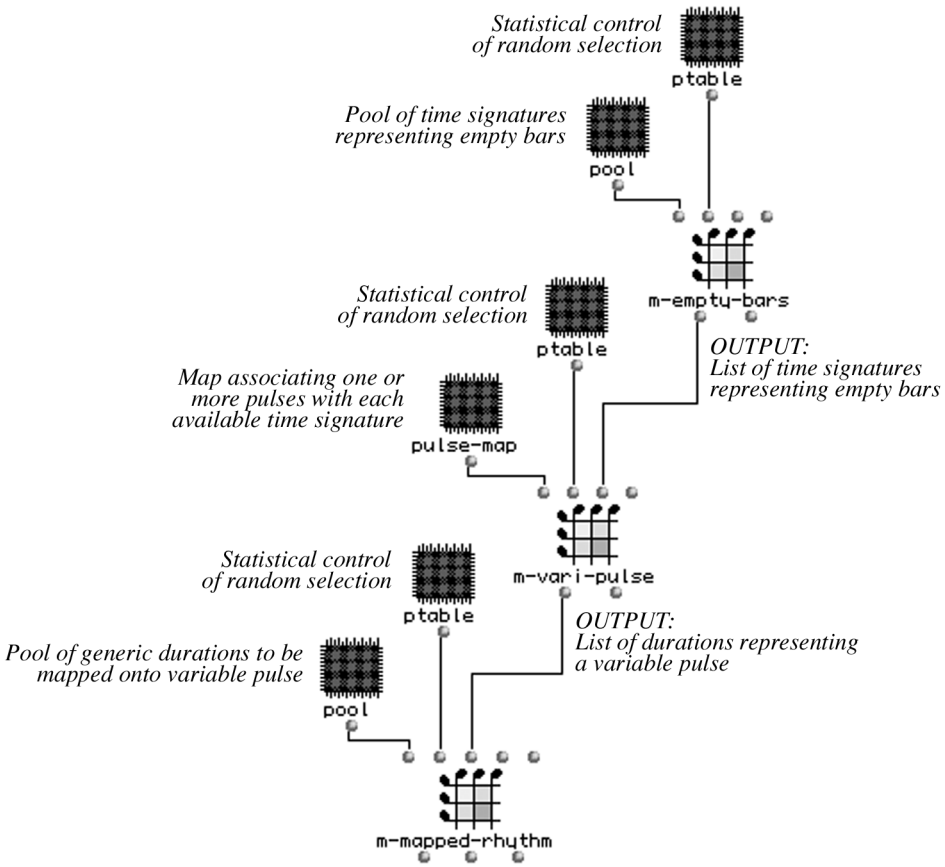
Of course, performers can execute figuration that is considerably more rapid than this—up to the limits of their dexterity. But I believe it is useful to regard London's limit as the threshold beyond which a pulse cannot support what I have been calling "idiomatic" addition-based rhythms. These considerations led me to the last of the rhythm-generating algorithms I will discuss, a hybrid model with both addition- and division-based features.

The different stages of this hybrid algorithm are distributed among three main functions in the current version of my OMTP software, and the typical interaction of these functions is illustrated in Figure 9. All three functions involve the same basic mechanism as my original `m-rhythm` procedure: They build results by repeatedly making controlled random selections from a designated pool.

Near the top of Figure 9, the function `m-empty-bars` assembles a list of time signatures representing empty bars. In the middle of the diagram, `m-vari-pulse` fills these bars with a variable pulse, according to a method I will describe shortly. And at the bottom of the diagram, `m-mapped-rhythm` completes the process by mapping generic durations onto the variable pulse. This mapping is the addition-based part of the process and operates just like the simple model presented in Figure 1a. In the case of `m-mapped-rhythm`, I call the elements of the pool "generic" durations, because they acquire different specific sizes depending on the details of the variable pulse wherever they occur. The generic/specific distinction, which is used for similar purposes by London (2004), originates in earlier work on the theory of scales, where the generic interval of a third, say, can have a specific size of either three or four semitones depending on its position within the diatonic scale (Clough and Myerson 1985).

The patch icons in Figure 9 are used to represent data. Those labeled "pool" are event spaces for random selection (empty bars, generic durations), and those labeled "ptable" contain probabilities in a list or transition table and are used to drive the corre-

Figure 9. OpenMusic patch (visual program) showing typical interaction of `m-empty-bars`, `m-vari-pulse`, *and* `m-mapped-rhythm`.



sponding Markov process. The "pulse map" describes different ways that a measure can be filled with isochronous pulses; its structure and purpose are described subsequently.

The construction of a variable pulse in `m-vari-pulse` is illustrated in Figure 10. The "pulse map" depicted in Figure 10a takes the role of the pool from which random selections are made. Each bar of this map represents the case of a different time signature that might be seen in the output from `m-empty-bars`, and each staff represents one of the available pulses. Once a pulse is selected, `m-vari-pulse` continues to track it until one of two conditions arises: (1) the current pulse is interrupted by a rest; or (2) a note in the current pulse coincides with a note in one or more of the other pulses. Each time either of these conditions is met, the pulse map is scanned to determine which pulses contain an onset at the current time, and one of these pulses is selected at random to be tracked next. If each of the source pulses is designed so that tuplets avoid cutting across the positions of tactus-level beats, then the resulting variable pulse will inherit this feature. This helps to regulate the performance complexity of rhythms mapped onto the variable pulse (Weisberg 1993, pp. 21–39).

Figure 10b provides an example of the tracking process that produces a variable pulse. Given Pulse-1 at the beginning of the process, the first tracking decision occurs at measure 1, beat 2, where Pulse-1 and Pulse-2 coincide; Pulse-1 continues to be tracked at this point. The second tracking decision occurs at measure 2, beat 1, where Pulse-1 and Pulse-2 again coincide; this time, Pulse-2 is selected. And the process continues in this manner, generating the variable pulse shown on the bottom staff of Figure 10b.

Figure 10. (a) Pulse map; (b) construction of variable pulse; (c) mapping generic durations onto a variable pulse.

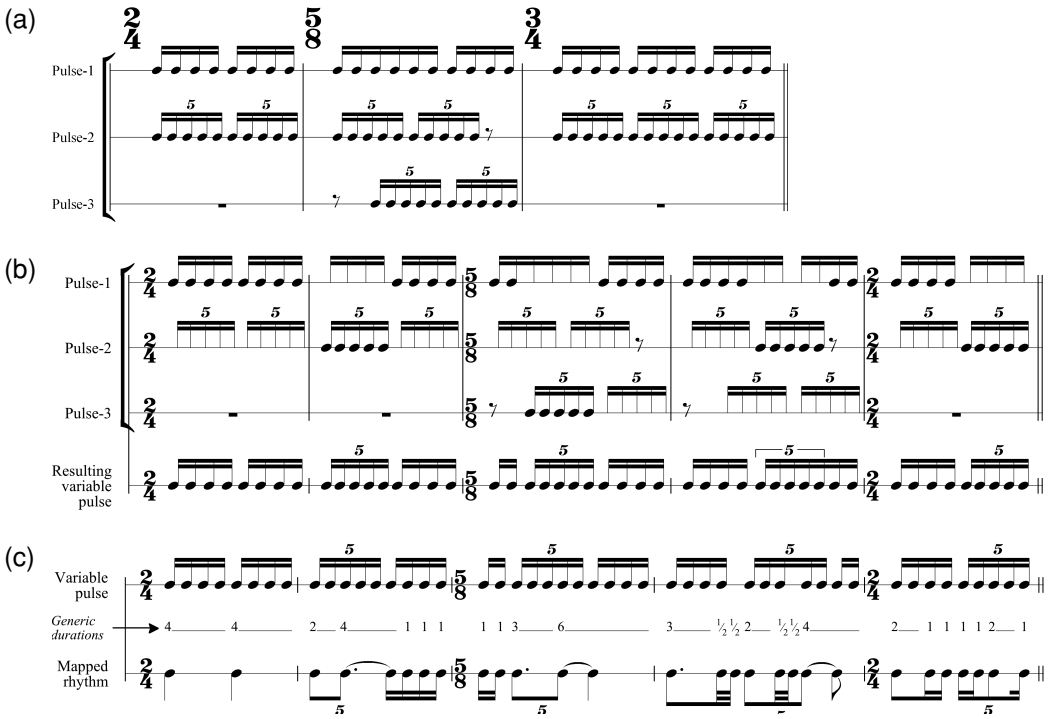


Figure 10c illustrates the mapping process that I have already described. One new detail can be seen in this example, however; there are fractional values among the generic durations. Of course, inserting an isolated fractional value into the mapped rhythm would cause the resulting rhythm to shift out of alignment with the grid that is supposed to regulate it. But by bundling fractions into strings of equal values with unit sums and treating each of these bundles as a single event in the pool of generic durations, a user can create localized rapid details in rhythms that remain aligned with the intended grid.

Consider how the various issues of complexity that I have discussed are managed within the `m-mapped-rhythm` model. Here, the various grids that are subject to addition-based structuring are explicitly represented in the pulse-map parameter, which places them in relation to various notated meters. The stochastic controls on pulse tracking can be configured to limit both the number and degree of variations in pulse rate. And as I have just noted, strings of fractions in the generic-duration pool allow access to levels of pulse smaller than London's "metric floor" without introducing inappropriately rapid grids on the additive side of the process.

Acknowledgments

This article began as a contribution to the special session organized by Diane Urista in honor of the composer-theorist Jonathan Kramer (1942–2004) at the 2006 meeting of the Music Theory Society of New York State. I am grateful to the editors of *Computer Music Journal* and to two anonymous reviewers for their guidance in the process of revising it for publication.

References

Agon, C., et al. 1994. "Kant: A Critique of Pure Quantification." *Proceedings of the 1994 International Computer Music Conference.* San Francisco, California: International Computer Music Association, pp. 52–59.

Ames, C. 1982. "*Crystals:* Recursive Structures in Automated Composition." *Computer Music Journal* 6(3):46–64.
Assayag, G., J. Baboni, and K. Haddad. 2001. *OpenMusic 4.0 User's Manual and Reference.* Paris: IRCAM.
Bonnet, A., and C. Rueda. 1999. *Situation 3: An OpenMusic Library for Constraints Programming.* Paris: IRCAM.
Byrd, D. 1994. "Music Notation Software and Intelligence." *Computer Music Journal* 18(1):17–20.
Clayton, M. 2000. *Time in Indian Music: Rhythm, Metre and Form in North Indian Rag Performance.* London: Oxford University Press.
Clough, J., and G. Myerson. 1985. "Variety and Multiplicity in Diatonic Systems." *Journal of Music Theory* 29(2):249–270.
Cox, F. 2002. "Notes Toward a Performance Practice for Complex Music." In C.-S. Mahnkopf, F. Cox, and W. Schurig, eds. *Polyphony and Complexity.* Hofheim, Germany: Wolke, pp. 70–132.
Harker, B. 1994. "Roger Reynolds's *Quick Are the Mouths of Earth* (1965): A Study of Methods and Materials." Master's Thesis, Brigham Young University.
Jones, K. 1981. "Compositional Applications of Stochastic Processes." *Computer Music Journal* 5(2):45–61.
Koetting, J. 1970. "Analysis and Notation of West African Drum Ensemble Music." UCLA *Selected Reports* 1(3).
Laurson, Mikael. 1996. *Patchwork.* Helsinki: Sibelius Academy.
Lerdahl, Fred, and Ray Jackendoff. 1983. *A Generative Theory of Tonal Music.* Cambridge, Massachusetts: MIT Press.
London, J. 2004. *Hearing in Time: Psychological Aspects of Musical Meter.* London: Oxford University Press.
Malt, M. 1999. "Brian Ferneyhough et l'aide informatique à l'écriture" ["Brian Ferneyhough and Computer-Assisted Composition"]. In P. Szendy, ed. *Compositeurs d'aujourd'hui* [Composers of Today]. Paris: IRCAM and L'Harmattan, pp. 61–106.
Mazzola, G. 2003. *The Topos of Music: Geometric Logic of Concepts, Theory, and Performance.* Basel: Birkhäuser.
Nauert, P. 1994. "A Theory of Complexity to Constrain the Approximation of Arbitrary Sequences of Timepoints." *Perspectives of New Music* 32(2):226–263.
Nauert, P. 2006. *OMTimePack 2.5: An OpenMusic Library for Creating and Transforming Rhythms.* Paris: IRCAM.
Perkins, J. M. 1965. "Note Values." *Perspectives of New Music* 3(2):47–57. Reprinted in B. Boretz and E. T. Cone, eds. *Perspectives on Notation and Performance.* New York: Norton, 1976.
Pratt, K. 1987. *Korean Music: Its History and Its Performance.* Seoul: Jungeumsa.
Reynolds, R. 1965. *Quick Are the Mouths of Earth.* New York: C. F. Peters.
Sandred, Ö. 1999. *OMRC 1.1: A Library for Controlling Rhythm by Constraints.* Paris: IRCAM.
Schick, S. 1994. "Developing an Interpretive Context: Learning Brian Ferneyhough's *Bone Alphabet.*" *Perspectives of New Music* 32(1):132–153.
Weisberg, A. 1993. *Performing Twentieth-Century Music: A Handbook for Conductors and Instrumentalists.* New Haven, Connecticut: Yale University Press.
Wuorinen, C. 1979. *Simple Composition.* New York: Longman.