
David Temperley

Eastman School of Music
26 Gibbs St.
Rochester, NY 14604 USA
dtemperley@esm.rochester.edu

An Evaluation System for Metrical Models

In the young and rapidly expanding field of music artificial intelligence, one particularly active area of research has been metrical analysis (also known as meter-finding, beat tracking, beat induction, rhythm parsing, and rhythm transcription)—the problem of extracting metrical information from music. Indeed, it would probably be fair to say that no problem in the field has attracted as much attention and energy as this one. Table 1 shows a list of 25 studies that present computational models of metrical analysis. (The table includes all published studies—not master’s and Ph.D. theses—that I have been able to identify and obtain. It includes only models that have been implemented or at least completely specified; well-known models in music theory such as Lerdahl and Jackendoff’s (1983) are excluded for this reason. Also excluded are models that identify tempo only without identifying actual beat locations, such as Brown 1993. In cases where several studies present close variants of a single model, only one study is listed.)

The models in Table 1 reflect a variety of perspectives on the metrical analysis problem. They might be categorized along several different lines. One fundamental distinction concerns the nature of the input; until recently, almost all systems worked from symbolic (“note”) input of some kind, but in the last few years several models have been proposed which operate directly from audio data. Some models assume a quantized input (for example, with durations represented by small integer values), whereas some allow the fluctuations in timing characteristic of human performance; some models generate just a single level of beats, whereas others generate several. Of course, the models might also be categorized in terms of their approach (rule-based, connectionist, oscillator-based, probabilistic, etc.), but this is a more complex matter, so I do not consider it in Table 1.

Perhaps the most basic question to address about a model—though it is not always addressed—is its

goal. Some metrical analysis systems are clearly intended to model human cognition; others are simply designed to solve the practical problem of meter-finding in whatever way seems most effective. The importance of meter-finding as a cognitive process seems self-evident; meter is a basic part of musical experience and has been shown to influence other aspects of music cognition as well, such as melodic similarity (Gabrielsson 1973), expectation (Jones et al. 2002), harmony perception (Temperley 2001), performance expression (Sloboda 1983; Palmer 1997), and performance errors (Palmer and Pfordresher 2003). However, the practical problem is important as well. In particular, generating music notation for a piece requires identification of its meter. As argued in Temperley (2001), if we conceive of a metrical structure as a multi-leveled framework of beats (whole-note beats, half-note beats, and so on, down to the smallest rhythmic level in the piece), the metrical structure of a piece essentially provides the information required to rhythmically notate it. And, precisely because of the central role of metrical structure in music cognition (as argued above), it will inevitably come into play in other problems of musical engineering. For example, tasks such as matching queries to a musical database, categorizing pieces by style or mood, or generating an accompaniment for a melody will surely require the consideration of metrical information.

Whatever the goals and assumptions of a metrical model, an important and obvious question to ask is, “How good is it?” That is, what percentage of the time does it actually produce the correct result? (The “correct result” can be defined as the metrical structure inferred by competent listeners. There might, of course, be some subjective differences among listeners; one might also take the music notation for the piece to represent its meter. But in most cases, I would argue, there will be agreement among these sources.) If a model is intended as a practical tool for meter-finding, the importance of this question hardly needs defending. If the model is intended as a hypothesis about cognition,

Table 1. Models of Metrical Analysis

<i>Reference</i>	<i>Input: Audio or Symbolic?</i>	<i>Input: Quantized or Performed?</i>	<i>Output: Multiple Levels?</i>
Longuet-Higgins and Steedman (1971)	symbolic	quantized	yes
Longuet-Higgins (1976)	symbolic	performed	yes
Steedman (1977)	symbolic	quantized	yes
Chafe et al. (1982)	symbolic	performed	yes
Longuet-Higgins and Lee (1982)	symbolic	quantized	yes
Povel and Essens (1985)	symbolic	quantized	no
Desain and Honing (1989)	symbolic	performed	yes
Allen and Dannenberg (1990)	symbolic	performed	no
Lee (1991)	symbolic	quantized	yes
Miller et al. (1992)	symbolic	quantized	yes
Rosenthal (1992)	symbolic	performed	yes
Rowe (1993)	symbolic	performed	yes
Large and Kolen (1994)	symbolic	performed	yes
McAuley (1994)	symbolic	performed	no
Parncutt (1994)	symbolic	quantized	no
Scheirer (1998)	audio	performed	no
Temperley and Sleator (1999)	symbolic	performed	yes
Cemgil et al. (2000a, 2000b)	symbolic	performed	yes
Dixon (2001a)	symbolic or audio	performed	no
Eck (2001)	symbolic	quantized	no
Goto (2001)	audio	performed	yes
Raphael (2001)	symbolic	performed	yes
Sethares and Staley (2001)	audio	performed	yes
Spiro (2002)	symbolic	quantized	yes
van Zaanen et al. (2003)	symbolic	quantized	yes

the relevance of its level of performance is less clear. A model could perform perfectly, producing exactly the correct structure (i.e., the one inferred by most listeners) in 100 percent of cases, and yet bear no resemblance whatsoever to the cognitive process of meter-finding; a model that was only correct 20 percent of the time might still turn out to capture important aspects of the cognitive process. Yet it seems to be generally accepted in cognitive science that the level of performance of a cognitive model is at least one valid criterion to consider in evaluating it, though there are certainly others. (One might consider, for example, how well the model accords with other experimental evidence about the cognitive process, whether the model's architecture and computational demands are psychologically plausible, and so on.) Given two cognitive models A and B, otherwise equal in

cognitive plausibility, if model A's output is much better (closer to that of humans) than model B's, this surely gives model A greater credibility as a hypothesis about cognition.

In short, the level of performance of a metrical model is of central importance to the practical goal and of at least some importance to the cognitive goal. Given the large number of metrical models that have been proposed, then, it seems worthwhile to examine the quality of their performance. The aim of the current article is actually not to answer this issue directly, but rather to address a preliminary question: "How can we decide how good a metrical model is?" In what follows, I propose a system for evaluating metrical models with the goal of measuring their performance and comparing their strengths and weaknesses in this regard.

Four Requirements for an Evaluation System

The problem at hand exemplifies a commonly occurring situation in artificial intelligence and other fields. The ultimate goal is to develop systems that can retrieve some kind of information from data; many models are available for the task, and we wish to be able to evaluate their success. For such an evaluation system, I submit that four things are needed: an agreed-upon way of representing the kind of information to be retrieved; a suitably large and representative corpus of data; correct analyses of the corpus (representing the information to be retrieved); and an agreed-upon way of comparing a model's analyses of the corpus to the correct analyses and scoring the model on its success at matching the correct analyses.

It is useful to consider a successful solution to the evaluation problem in another domain—computational linguistics. Since the birth of the field, a central project in computational linguistics has been the development of systems for natural-language parsing—recovering syntactic information from written or spoken text. Until recently, progress in this area was hindered by the difficulty of evaluating models and comparing one model to another. In the early 1990s, this problem was largely solved by the development of the Penn Treebank (Marcus et al. 1993). The Penn Treebank is a corpus of several million words of naturally occurring text gathered from both written and spoken sources. The treebank is accompanied by syntactic analyses done by experts—a “constituent structure” for each sentence showing noun phrases, verb phrases, clauses, etc. Metrics have been proposed for comparing “treebank-style” analyses (Black et al. 1991), and programs are available that take two analyses of a set of sentences—the correct analysis set (known as a “goldfile”) and one produced by a model to be evaluated (a “testfile”)—and evaluate how well the testfile analyses match those of the goldfile.

For the problem of natural-language parsing, then, it can be seen that all four of the requirements listed above have been met. This achievement has led to a surge of progress in natural-language parsing. (For discussion, see Manning and Schütze 2000.) In

what follows, I consider possible solutions to these four requirements for the metrical-model evaluation problem.

Previous Work on the Evaluation of Metrical Models

For the most part, issues of evaluation have received little attention in the literature on metrical analysis. Many studies present no quantitative performance measures for the models presented. In the last few years, however, some important proposals have been offered in this area.

The appropriate way of evaluating a metrical model depends on the nature of the model. We might distinguish first between models accepting only quantized input and those accepting performed input. In quantized-input models, time-points are generally represented as multiples of a short rhythmic value or “base unit,” such as a sixteenth-note. Such models generally assume that the metrical structure is perfectly regular throughout. In such a case, each level of the metrical structure can be characterized by two numbers: one indicating its period (in base units), or the distance between beats; the other indicating the phase (again in base units)—how long after the beginning of the piece the first beat occurs. A level generated by the model could be said to be correct if its period and phase exactly match one of the levels in the correct metrical structure. This is essentially the approach used by Desain and Honing (1999). Desain and Honing evaluate three metrical models (the model of Longuet-Higgins and Lee 1982 and two later variants of it) using three test corpora: a set of randomly generated rhythms, a set of “metrical” rhythms (in which each note is exactly one beat long at some metrical level), and a corpus of national anthem melodies. Each of the three models tested generates only a single beat level. For each model, the authors present data regarding the proportion of input cases for which the model's beat matches the main beat level in the correct structure (in both period and phase); they also present data comparing the model's beat to other beat levels in the correct structure. (Van Zaanen et al.

2003 use the same anthem corpus and a similar testing strategy.) Desain and Honing's approach seems very sensible for evaluating quantized-input models. In what follows, we focus mainly on performed-input models; as can be seen from Table 1, most models in recent years have addressed performed input.

An approach to evaluating performed-input metrical models is presented by Cemgil et al. (2000b). This requires data in which the locations of beats are explicitly indicated. (Just one level of beats is assumed.) Assume that the correct beat level for a piece is $S = s_1, s_2, \dots, s_I$ (where each s_n is the time point of a beat), and the model's output beat level is $T = t_1, t_2, \dots, t_J$. The score W for the closeness of any two beats s_i and t_j is expressed using a Gaussian window function (so that two exactly simultaneous beats receive a score of 1). The similarity function between S and T is then

$$\frac{\sum_i \max_j W(s_i, t_j)}{(I + J) / 2} \cdot 100$$

This formula takes each correct beat, pairs it with the closest beat to it in the model's output, and adds together the W scores for these beat pairs. This is then divided by the average of I and J (which means that the model's output will be penalized if it has many more beats than the correct structure) and multiplied by 100. The result is a single number that roughly indicates the percentage of correct beats that were matched by the model's beats. Using this evaluation technique, Cemgil et al. (2000b) tested their own model on keyboard performances of the Beatles song "Yesterday," played multiple times by different performers and under different tempo instructions (fast, normal, or slow). (The model was essentially given the correct initial tempo and phase.) Dixon (2001b) later tested his model against that of Cemgil et al. using a similar evaluation method and the same data set, and he reported overall scores of 94 ± 9 for his model versus 91 ± 7 for the model of Cemgil et al.

A somewhat similar approach is proposed by Goto and Muraoka (1997) for testing an audio-input metrical model. The system requires that the audio input be supplemented with markers, added by

hand, indicating the correct beat locations. Goto and Muraoka present an audio-input model that outputs multiple levels of beats; the model is evaluated in the following way. For each correct beat C_n , the closest beat B_m in the model's output is found, and an error is calculated, which is the time difference between C_n and B_m as a proportion of the interval between correct beats. The longest correctly tracked portion of the piece is found (i.e., the longest span in which the error is less than a certain value). The test set consisted of 40 songs (from popular music recordings without drums), each at least 1 min long, by a variety of artists. The authors define a correct analysis as one in which (1) the longest correctly tracked portion begins no later than 45 sec after the beginning of the piece and extends to the end, and (2) the mean, variance, and maximum of the error are within certain values. (For example, the mean error must be less than 0.2.) By this criterion, the model analyzed 87.5 percent of the input songs correctly at the quarter-note level.

The approaches of Cemgil et al. and Goto and Muraoka offer valuable contributions toward the evaluation of performed-input metrical models. However, they also encounter certain difficulties. In particular, they require the location of every beat to be exactly specified, when in fact the exact location of beats is often somewhat indeterminate. Consider a MIDI file of a classical piece generated from a piano performance (or an audio file, for that matter). There will likely be many chords in which several notes are understood as simultaneous and coinciding with a certain beat. But most likely they will not be played exactly simultaneously; so where exactly is the beat? There may also be beats with no coinciding note, so again the exact location of the beat is uncertain. This problem can be solved by having human annotators provide the beat information according to their intuition. But this solution is far from ideal; such hand-annotation is time-consuming and also somewhat subjective. It seems that, in evaluating a metrical analysis, we should focus on the aspects of the metrical structure that are clear and uncontroversial. Certain beats correspond with certain notes, and other beats may be in between but not necessarily at

determinate locations. (Concrete examples of this will be given later.)

One method that addresses this problem is what could be called the “score-time” system. In this system, the correct metrical structure for a piece is used to generate a quantized rhythmic representation such that each event has a correct location or “score-time” in the piece. For example, if quarter-note beats are defined as integers, a note on the first quarter-note beat is at score-time 0.0, a note on the next beat is at 1.0, and a note one eighth-note later is at 1.5. Such a system for rhythmic representation has been used for various purposes—for example, in score-performance matching (Heijink et al. 2000). A metrical model could then operate by assigning a score time to each note; the model could be evaluated according to (for example) the proportion of notes that were assigned the correct score-time. Notice that such a model need not decide the exact location of indeterminate beats that contain no notes. However, this system in effect only represents one level of meter: generally the main beat level or “tactus.” Score-times do not indicate higher metrical levels, e.g., whether tactus beats are grouped in duples or triples (though such information could be indicated in other ways; see for example Cemgil et al. 2000a). This measure is also rather inflexible. If a model inserted one extra beat near the beginning of the piece, then all of the score-times for subsequent events would be judged as incorrect, which seems unduly harsh.

In what follows, I propose an alternative approach to metrical evaluation. This builds on the “score-time” idea, but allows multiple metrical levels and also permits a more flexible comparison of metrical analyses. It should be stated at the outset that the system is intended primarily for symbolic-input models. Whether it will also be useful for audio-input models is an open question; I return to this at the end of the article.

A Representation System

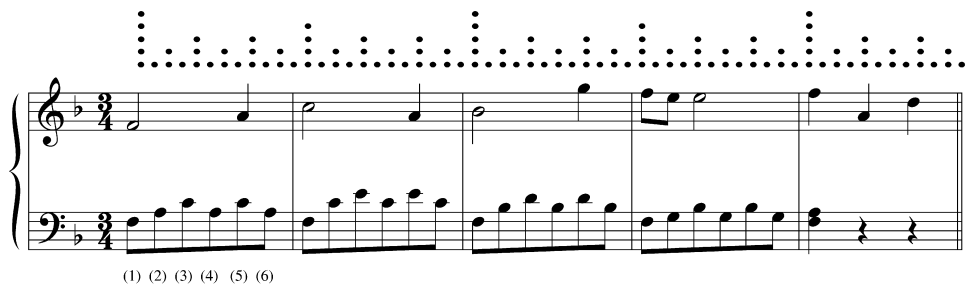
We must first consider how to represent the input data—the data to be analyzed. Following many other models, I will assume a representation con-

sisting of a list of notes encoded with “ontime” and “offtime” (both in msec) and pitch (in integer notation, with middle C = 60). Such a representation, similar to a MIDI file, is sometimes known as a “notelist.” Figure 1 shows the opening of a Mozart piano sonata; the leftmost column below the score shows the notelist for the first two measures. (The numbers in parentheses have been added for reference in the following discussion.) Notice that this system does not assume that the events are quantized (except at the very low level of msec); the notelist in Figure 1 was generated from a performance on a MIDI keyboard, and it can be seen that the timing is somewhat irregular.

We now turn to the representation of the metrical structure itself. Most metrical models produce some kind of representation of beats aligned with the music that was given as input. Beats are simply points in time, subjectively understood as accented, though not necessarily coinciding with any event. Some systems (as discussed above) generate several levels of beats, or—to put it another way—beats of varying strength, where “strong” beats are present at higher levels and “weak” beats only at lower levels. A metrical structure can be represented graphically as a framework of dots (Lerdahl and Jackendoff 1983), as shown in Figure 1 above the staff. In Temperley and Sleator (1999), we proposed encoding such a structure as a list of beat statements, each one with a timepoint and a level number representing the highest level at which that beat is present; the “beatlist” for the first two measures of the metrical grid in Figure 1 is shown in the middle column below the score. We assume a structure of five metrical levels, numbered 0–4, with higher numbers representing higher levels; level 2 is the “tactus” or main beat, the quarter-note level in this case. (The assumption of five levels seems optimal for common-practice music in general, though some pieces may call for more or fewer levels; see Lerdahl and Jackendoff 1983 and Temperley 2001 for discussion.)

Given this representation scheme, one possible idea for an evaluation system is as follows. Let us assume a corpus of pieces or excerpts encoded in notelist format. Each excerpt could be annotated

Figure 1. Mozart, Sonata KV 332, I, mm. 1–5, showing metrical grid (above the staff), notelist, beatlist, and note-address list.



Notelist	Beatlist	Note-Address List
Note 2882 3935 65	Beat 2882 4	ANote 2882 3935 65 10000
Note 2903 3159 53 (1)	Beat 3060 0	ANote 2903 3159 53 10000 (1)
Note 3132 3402 57 (2)	Beat 3132 1	ANote 3132 3402 57 10010 (2)
Note 3392 3652 60 (3)	Beat 3262 0	ANote 3392 3652 60 10100 (3)
Note 3645 3877 57 (4)	Beat 3392 2	ANote 3645 3877 57 10110 (4)
Note 3888 4452 69	Beat 3500 0	ANote 3888 4452 69 10200
Note 3900 4125 60 (5)	Beat 3635 1	ANote 3900 4125 60 10200 (5)
Note 4133 4385 57 (6)	Beat 3760 0	ANote 4133 4385 57 10210 (6)
Note 4412 5477 72	Beat 3888 2	ANote 4412 5477 72 11000
Note 4413 4694 53	Beat 4010 0	ANote 4413 4694 53 11000
Note 4689 4948 60	Beat 4133 1	ANote 4689 4948 60 11010
Note 4936 5164 63	Beat 4270 0	ANote 4936 5164 63 11100
Note 5194 5448 60	Beat 4412 3	ANote 5194 5448 60 11110
Note 5447 6011 69	Beat 4550 0	ANote 5447 6011 69 11200
Note 5456 5677 63	Beat 4689 1	ANote 5456 5677 63 11200
Note 5688 5954 60	Beat 4810 0	ANote 5688 5954 60 11210
.	Beat 4936 2	.
.	Beat 5060 0	.
.	Beat 5194 1	.
	Beat 5320 0	
	Beat 5447 2	
	Beat 5570 0	
	Beat 5688 1	
	Beat 5810 0	
	.	
	.	
	.	

with the correct beatlist; this could then be compared with a beatlist generated by the model being tested (similar to the approaches of Goto and Muraoka 1997 and Cemgil et al. 2000b). A problem arises here, as has already been mentioned: the location of beats is often indeterminate. In Figure 1, the first notes in the right hand and the left hand are understood as coinciding with the first beat, but they are not exactly simultaneous. Is the beat located at 2882, 2903, or somewhere in between?

(The beatlist in Figure 1 arbitrarily aligns the beat with the first of these two onsets.) In addition, many beats have no coinciding note. In Figures 2a and 2b (excerpts from later in the same piece), where exactly is the second quarter-note beat of m. 12, or the third quarter-note beat in m. 40? The same could be said of all the level-0 beats in Figure 1, none of which coincide with notes. Indeed, one might question whether level-0 beats are even present in this passage. The location of these indeter-

Figure 2. Three excerpts from Mozart, Sonata KV 332, I, showing note-address lists at right.

(Figures 2a and 2b show five-value note-addresses; Figure 2c shows six-value note addresses.)

(a) m. 12



```
ANote 20094 20411 65 61000
ANote 20106 20430 57 61000
ANote 20115 20430 41 61000
ANote 21277 21683 84 61200
ANote 21281 21658 69 61200
ANote 21289 21662 72 61200
ANote 21676 21843 67 61211
ANote 21683 21783 70 61211
ANote 21684 21829 82 61211
```


(b) m. 40



```
ANote 62872 63097 79 201000
ANote 62901 63136 55 201000
ANote 63450 63748 43 201110
ANote 63453 63747 67 201110
```

(c) m. 8





```
ANote 13736 14309 43 410000
ANote 13737 15368 67 410000
ANote 14086 14119 64 410111
ANote 14112 14265 65 410112
ANote 14255 15082 48 411000
ANote 14270 14529 64 411000
ANote 14531 14824 62 411100
ANote 14789 15327 64 412000
ANote 15071 15311 45 412100
```

minate beats could perhaps be determined in perceptual terms—we do have some intuitions about where the second beat of m. 12 occurs (probably roughly halfway between the first and third quarter-note beats)—but for a human annotator to produce the correct answer would be quite difficult.

As suggested earlier, for the purposes of metrical evaluation, we should focus on what is uncontroversial. And some things do seem uncontroversial: in Figure 2a, the notes of the first chord fall on the beat, even though they may not be exactly simultaneous. This is followed by an empty beat (though its exact timepoint is indeterminate), followed by another chord on the third beat of the measure. (Notice also that the exact location of beat 2 of m. 12 is not needed for the purpose of converting the beatlist into notation. As long as it is known that the first chord coincides with the first level-2 beat of the measure, and the second chord with the third level-2 beat, with one other level-2 beat somewhere in between, this is all that is required.)

One representation system that meets this requirement is what I will call the “note-address” system—shown in the right-hand column of Figure 1 for the opening of the Mozart sonata excerpt. It can be seen that the list of events here includes all the information of the original notelist, but the Note statements have become ANote statements, consisting of an ontime, offtime, pitch, and “note address”—a number representing the note’s position in the metrical grid. This number should really be thought of as a five-valued vector, but it can be represented as a single integer, as none of the values ever exceed 9 except for the leftmost one. (We assume five levels for now, though in principle more or fewer could be used.) Each value in the vector corresponds to a level of the metrical grid (the rightmost value corresponds to level 0). The value represents the number of beats at that level that have elapsed at that point in the piece since the previous beat at the next level up.

As an example, consider the first six eighth notes of the left hand, indicated by the numbers in parentheses. For the quarter-note level (corresponding to the third digit from the right in the note addresses), the first and second of these notes have the value 0

(no quarter-note beats have occurred since the last dotted-half-note beat), the third and fourth have value 1 (one beat has occurred), and the fifth and sixth have value 2 (two beats have occurred). The highest-level (leftmost) value of the address, which is always set to 1 at the start of the piece, represents the number of highest-level beats that have elapsed in the piece as a whole (since there is no higher level from which to count). Eventually this could become a 2- or 3-digit number, producing addresses like 20-1-0-0-0 (as in Figure 2b). One could actually assign addresses to the beats themselves; in a perfectly regular duple grid, this would be equivalent to counting binary numbers (10000, 10001, 10010, 10011, 10100, . . .), with the exception that the leftmost digit is not binary. However, if we assign addresses only to notes, we avoid the earlier-discussed problem of having to decide on the location of indeterminate beats.

The note-address system also solves the problem created by nominally simultaneous notes. As observed earlier, the notes of a chord are generally understood as occurring on the same beat (and would be represented that way in notation), yet they are rarely performed exactly simultaneously. Thus assigning the “beat” to a single timepoint is difficult—indeed impossible, if we want all the notes of the chord to be represented as coinciding with the beat. The solution provided by the note-address system can be seen in Figure 1. In m. 1, the first right-hand and first left-hand notes can be given the same address, even though they do not exactly coincide, representing that they are understood as being simultaneous in metrical terms and should be notated that way.

One further refinement of the note-address system is needed. It is generally accepted in metrical theory that not all notes necessarily coincide with beats; some, like trills, turns, and grace notes, are “extrametrical” (Temperley 2001). Figure 2c shows an example from m. 8 of the Mozart excerpt. Two notes in the right-hand (notated in small note-heads, as is customary) do not really occur on any beat. To accommodate extrametrical notes, we introduce one further (rightmost) digit of the note-address. This digit is 0 for any note that coincides with a beat; for a note that is between two beats,

the address of the note is the address of the previous beat, except with 1 as the final digit. If there are multiple, non-coinciding notes between the two beats, the final digit of their addresses reflects their ordering in time: the first extrametrical note in Figure 2c is labeled 410111, the second 410112. (Problems would arise if there were more than nine non-coinciding notes between two beats, but this seems unlikely to occur.) A note address therefore has six values, which we will refer to as levels 4, 3, 2, 1, 0 (for the five metrical levels), and -1 (the extrametrical level).

The note-address system is not perfect. For one thing, not only does it fail to represent the location of beats with no coinciding note, it sometimes even fails to represent their existence. In Figure 2a, for example, there is no explicit indication of the second beat of the measure (which would be at address 61100). In this case, the existence of the missing beat is implicit in the fact that the following note has the address 61200; this implies that there must have been a 61100 as well. A more worrisome case is shown in Figure 2b. Here, the third beat of m. 40 is not represented even implicitly. If we just looked at the note addresses for the passage (even including the address of the following downbeat, 210000), we could not even be certain that m. 40 were a triple-meter measure. We should bear in mind, then, that certain errors metrical models might make would not be recognized under the note-address system: for example, if a metrical model omitted the third beat in m. 40, or added a fourth beat (or an extra ten beats) at the end of the measure. The seriousness of this problem is unclear; it depends on how often metrical models actually make these kinds of errors. (The problem could be addressed in various ways, for example, by supplementing each highest-level beat with information about the duple or triple divisions of the grid at each level.) Some other limitations of the note-address system will be discussed in later sections.

The Corpus

In selecting a corpus for testing, two considerations are important. First, the criteria used in assembling

the corpus should be as systematic as possible. Rather than selecting 20 pieces that one knows and happens to have handy, one should, if possible, devise objective criteria for inclusion in the corpus. (The danger is, of course, that those pieces we know and have handy are likely to be the ones we used, or at least thought about, in developing our model.)

Another requirement is that the corpus should be a representative sample of the larger corpus that the models to be tested were designed to analyze. This is problematic in the current case. Many models have specifically addressed themselves to “common-practice” music (roughly speaking, Western art music from 1700 to 1900). (Even here, there may not be total agreement on what common-practice music includes.) Other models have sought to accommodate other kinds of music, such as jazz, rock, and non-Western music (Scheirer 1998; Cemgil et al. 2000b; Goto 2001; Sethares and Staley 2001). Thus, I certainly do not claim that the corpus proposed below would be a suitable or fair one for all of the models listed in Table 1.

In Temperley (2001), I presented the “Kostka-Payne Corpus” (hereafter the KP Corpus)—a set of 46 excerpts from the common-practice repertoire taken from the workbook accompanying Stefan Kostka’s and Dorothy Payne’s textbook *Tonal Harmony* (1995). The corpus includes all excerpts from the workbook of at least eight measures in length; it contains a total of 541 notated measures and 9,057 notes. Although the corpus is not especially big (a larger one would certainly be desirable), it has two important advantages: the excerpts were selected by an objective criterion; and they span a range of periods, genres, and composers within the common-practice idiom, so that the corpus can be considered a reasonably good sample of common-practice music. Notefiles were generated for all the excerpts in the corpus; these notefiles are quantized (i.e., generated from notation) and thus have perfectly regular timing (using tempi that seemed reasonable to me). However, for the 19 excerpts from the corpus for solo piano, I had a skilled (doctoral-level) pianist perform the excerpts and generated notefiles from those as well. (We will call

this the KP Performed Corpus.) In the quantized notefiles, no extrametrical notes were included; in the performed notefiles, the performer was allowed to include whatever extrametrical notes were desired. The performances contained a few extraneous notes (performance errors) that were subsequently deleted from the performed files (owing to the difficulty of deciding on the “correct” metrical location of these); notes erroneously omitted from the performed files were not restored.

Note-address files were generated for all excerpts in the corpus (both the quantized files and the performed ones) using the format shown in earlier examples (specifically, the six-value address format shown in Figure 2c). This was done by running the input files through the Melisma meter-finding program (discussed later), sometimes adjusting the parameters in *ad hoc* ways to get the output as close to correct as possible and then hand-correcting any remaining errors. With very few exceptions, this was unproblematic: it was clear, both for the quantized files and the performed files, what the correct address of each note should be. It was sometimes not obvious which metrical level should be defined as the tactus (level 2); I used my own musical judgment for these decisions. (In some slow movements, for example, the length of the tactus beats if the notation were taken literally—e.g., quarter-note beats in a 2/4 meter—would be over two seconds, beyond most estimates for the upper range of the perceived tactus.) Note-addresses were encoded only up to the notated measure indicated in the score; “hypermusical” levels (above the measure) were not included owing to their subjective nature.

Comparing Goldfiles and Testfiles

We now have a way of representing metrical structures (the note-address system), a corpus of data, and correct analyses of the data in the required format. The final requirement is a way of comparing a note-address file produced by a metrical model (a “testfile”) with the correct analysis (the “goldfile”). We assume for now that the goldfile and testfile contain exactly the same events (identified by their

pitches and timepoints), though we will modify this assumption slightly below. Therefore, there should be no difficulty in matching events in the goldfile with their corresponding events in the testfile. The problem is to compare the note addresses for corresponding events between the two files.

One very simple approach would be to use “exact matching,” that is, to score the testfile on the proportion of its events that are assigned exactly the same address in the goldfile. This would be a fairly harsh and unforgiving metric. For example, suppose a model erroneously inserted an extra measure at the beginning of the piece but otherwise analyzed the piece exactly correctly. The leftmost value of each note address after the inserted measure would be incorrect (one greater in the testfile than in the goldfile), so almost every note address in the piece would be incorrect—though the model’s analysis was in fact almost completely correct. (A similar problem arises with the “score-time” method of meter representation, as discussed earlier.)

A fairer approach would be to assign the testfile a score for every level, indicating the proportion of events that were correctly labeled at that level. This is the approach taken here, implemented in a program called *compare-na*. This program takes as input two note-address files, a goldfile and a testfile. For every level *L* that is present in the note-addresses in the goldfile (except for the top level—this is explained later), a score is assigned that indicates the proportion of events whose note-address value at level *L* is the same in the testfile as in the goldfile. The program also produces an overall score, which is simply the average of the individual level scores. (It is not clear how meaningful this is. One could also consider a weighted score, with some levels—e.g., the tactus level—weighted more than others. Alternatively, as I prefer, one could simply take the level scores individually as indicators of the model’s success at the tactus level, higher levels, and lower levels.)

An example is shown in Figure 3 to give a flavor of this scoring system. A rhythmic pattern is shown at left—a simple eighth-note pattern in a 12/8 meter. Four analyses are shown in the form of

Figure 3. A rhythmic pattern showing four analyses as metrical grids and note-address lists. Analysis A is

the correct analysis. For analyses B, C, and D, the scores for each analysis compared to analysis A,

according to the note-address evaluation system, are shown below.

	Analysis A (correct)	Analysis B	Analysis C	Analysis D
8				
12				
	100000 100100 100200 101000 101100 101200 110000 110100 110200 111000 111100 111200 200000	100000 100100 101000 101100 102000 102100 110000 110100 111000 111100 112000 112100 200000	100100 200000 200100 200200 201000 201100 201200 210000 210100 210200 211000 211100 211200	100000 100010 100020 100100 100110 100120 101000 101010 101020 101100 101110 101120 110000
	Level -1: 1.000 Level 0: 1.000 Level 1: 0.385 Level 2: 0.538 Level 3: 1.000 Total score = 0.785 (offset = 0)	Level -1: 1.000 Level 0: 1.000 Level 1: 0.000 Level 2: 0.692 Level 3: 0.846 Total score = 0.708 (offset = 0)	Level -1: 1.000 Level 0: 1.000 Level 1: 1.000 Level 2: 1.000 Level 3: 1.000 Total score = 1.000 (offset = 1)	

metrical grids, the correct one (analysis A) and three incorrect ones; the note-address list implied by each grid (for each note of the pattern) is shown at the right. The scores yielded by compare-na for analyses B, C, and D, when compared with analysis A, are shown below each analysis. In all four analyses, no notes are analyzed as extrametrical, thus the rightmost digit is 0 for all notes. Because all analyses agree totally with analysis A for level -1, they all receive a score of 1.000 at this level. In analyses B and C, as in analysis A, every note is analyzed as coinciding with a level 1 beat, so the second-from-the-right digit is 0 for all notes as well. (We will discuss analysis D in a moment.)

In analysis B, only level 2 of the metrical grid is incorrect: in essence, the passage is analyzed as 6/4 instead of 12/8. In this case, the level-2 values of the addresses are mostly incorrect, as are the level-

1 values, leading to low scores for level 2 (0.538) and level 1 (0.385). In analysis C, levels 2, 3, and 4 are all "out-of-phase" by one eighth-note beat (as if the piece was notated in 12/8 with the barlines one note too late). Many of the address values at levels 2 and 3 are now incorrect, as are all of the level 1 values. Values for levels 0 and -1, however, are unaffected.

These examples point out an important, and rather counterintuitive, aspect of the note-address system. What would normally be thought of as an error at a particular metrical level will generally affect not only the address values at that level, but the values at the next lower level as well. For this reason, also, it seems superfluous to compare the highest metrical levels of the addresses, because any difference in this level will cause differences at the next level down. Thus the evaluation system

compares note-addresses only up to and including the second-highest level present in the goldfile.

(Notice the address assigned to the upbeat in analysis C. For the first beat in the piece, we assign a value of 1 to the highest level present in the beatlist; all other values are set to 0 except for the highest level at which that beat is present—level 1 in this case—which is set to 1.)

Important questions arise here regarding how the similarity between two metrical structures should be judged. By one criterion, analyses A and C in Figure 3 are very similar: both reflect a 12/8 metrical structure, and the structures are also the same in terms of the periods (intervals between beats) at each level. Yet the metrical strength of each event is incorrect; indeed, from this point of view, analysis C is maximally incorrect. In analysis B, the “time signature” (the duple/triple relationships among levels) is incorrect, but at least some of the events have the correct metrical strength. Which of these two analyses is more similar to analysis A? The note-address system, along with the comparison method proposed here, seems to give fairly intuitive results regarding the similarity between structures. For example, analysis B scores higher overall than analysis C here (0.785 versus 0.708), as I think it should. However, there may be no single correct answer in this regard; it may depend on the specific goals for which the model is intended.

Suppose, for the rhythmic pattern shown in Figure 3, a model produced the output shown in analysis D. This analysis essentially matches analysis A, except that the levels of the two analyses are not aligned: the tactus (dotted quarter note) is level 2 in the goldfile but level 1 in the testfile. It is unclear how this situation should be handled. If the two address lists were compared exactly as they are, the testfile would receive the very low overall score of 0.462. This solution is surely too harsh; analysis D is not as incorrect as a score of 0.462 would suggest.

Another approach would be to compare each level in the testfile with whichever level in the goldfile addresses it matches most closely. This is the approach taken by *compare-na*: different “offsets” of the testfile relative to the goldfile are tried, and the one is chosen that yields the best match.

The program also outputs the offset value that was chosen; an offset of 1 means that level L in the goldfile was matched to level $L-1$ in the testfile. At an offset of 1, analysis D yields a perfect overall score of 1.000. (Level -1 in the goldfile has no corresponding level in the testfile; all address values not explicitly indicated in the testfile are assumed to be 0.) Some might consider this approach too forgiving. Generally, the perception of a certain beat level as the tactus or main beat is assumed to be an important aspect of metrical cognition; the correct identification of this level might well be regarded as part of the meter-finding problem. (One solution would be to consider different alignments of the goldfile and testfile addresses, but factoring in a penalty for level misalignment.)

The *compare-na* program also contains a tolerance parameter for timing. Some models, such as the model of Temperley and Sleator (1999), slightly alter the timepoints of events. One reason for this is that it allows the notes of a chord—rarely played exactly simultaneously—to be made simultaneous and thus aligned with the same beat. Once the timepoints of an event are altered in the testfile, this could cause problems for the matching of events between the testfile and the goldfile. The tolerance parameter addresses this problem; if the tolerance is 50 msec, a goldfile event with an onset time of T can be matched to any testfile event of the same pitch with an onset time of $T \pm 50$ msec. (If for some reason no matching testfile event within the specified tolerance is found for a goldfile event, that goldfile is judged as unmatched at all levels.)

Given a series of outputs (from different pieces or excerpts) from *compare-na*, the program *tally-na* averages the figures for each level over the entire corpus (weighting each excerpt in the corpus equally). A sample output of *tally-na* is shown below:

```
level-1: average proportion correct = 0.998 (46)
level 0: average proportion correct = 0.966 (46)
level 1: average proportion correct = 0.928 (46)
level 2: average proportion correct = 0.899 (44)
level 3: average proportion correct = 0.814 (29)
Overall corpus score = 0.931; number with zero
offset = 36 out of 46
```

The numbers in parentheses indicate the number of excerpts for which that level was eligible for

evaluation in the goldfile data. (Bear in mind that the goldfile addresses only reflect levels that are explicitly indicated in the notation, and also that the highest level of the addresses is never evaluated. This means that, for example, level 3 is only eligible for evaluation if level 4 is present in the goldfile addresses.) An overall corpus score is also given, which simply averages all the overall scores for each excerpt. The program also gives information about how many testfile analyses reflected the correct ("zero") level offset.

Experiments with Melisma

As an illustration of the evaluation system proposed above, I present an evaluation of the meter-finding program proposed by Temperley and Sleator (1999), part of the Melisma system for music analysis (Temperley 2001). A brief description is needed of the Melisma metrical model. The model is a preference rule system that considers a large number of possible analyses and chooses the one that is optimal on balance with regard to several criteria. The main criteria are as follows: (1) prefer for beats at each level to be aligned with event-onsets, the more onsets the better; (2) prefer for beats to be aligned with longer events; (3) prefer for beats to be regularly spaced at each level. (At the tactus level, "regularity" simply implies that each beat interval—between two adjacent beats—should be close in length to the previous beat interval; at higher and lower levels, regularity refers to the relationship between levels, i.e., a consistently duple or triple relationship between levels is preferred.) The model contains about a dozen user-settable parameters (regarding the relative weight of the preference rules and other things); these were set on a trial-and-error basis, before any testing was done using the KP Corpus.

Table 2 shows results of some tests performed on both the KP Corpus and the KP Performed Corpus. For each corpus, the first row shows, for each level, the number of excerpts for which that level was eligible for consideration. Next is shown the performance statistics for the Melisma model using the

default parameters of the model (as defined in the most recently released 2001 version of the system). Recall that the score for each level indicates a simple average of the scores for that level over all the excerpts. Not surprisingly, the overall performance on the KP files (0.931) was somewhat higher than on the KP-performed files (0.908)—recall that the former is generated from notation, the latter from MIDI keyboard performances—though the difference is not very large. Regarding the values for different levels, it should be remembered that errors at one level are mainly reflected in the scores for the next level down, so the relatively low scores for levels 2 and 3 in the KP Corpus mainly indicate errors at levels 3 and 4. (The high score for level 3 on the KP Performed Corpus is surprising, but only nine of the performed excerpts contained eligible data for level 3, so the sample was fairly small.)

As a further test, the Melisma program was run on piano renditions of the Beatles songs "Michelle" and "Yesterday" using the performances used for testing by Cemgil et al. (2000b) and Dixon (2001b). (MIDI files of these are publicly available online at www.nici.kun.nl/mmm/archives.) Cemgil et al. and Dixon used many performances of the same two songs; the current test just used one performance of each song (the first performance by the first subject in the "classical" category, played at a "normal" tempo). Correct note-address files were generated, and these were compared with the output of the Melisma program. The results are shown in Table 3. On "Yesterday," the tactus level was largely correct (as indicated by the high values for level 1), as were lower levels; level 3 was not as good, as it switched from duple meter to triple meter for one section of the piece. On "Michelle," the program's analysis was almost completely correct; the only errors were due to a few triplet-quarter notes, and a few "smudged" chords (in which notes intended as simultaneous were not interpreted as such by the program).

As well as allowing comparison between models, the evaluation system proposed here might greatly facilitate the development and improvement of models. It is now very easy, for example, to make a change in the Melisma meter system, run the system on the KP Corpus, generate note-address files

Table 2. Tests of the Melisma Meter Model on the Kostka-Payne Corpus Using the Note-Address Evaluation System

<i>Corpus and Model Used</i>	<i>Level 3</i>	<i>Level 2</i>	<i>Level 1</i>	<i>Level 0</i>	<i>Level -1</i>	<i>Correct Offset</i>	<i>Overall Score</i>
KP Corpus (46 excerpts)							
# of excerpts with data	29	44	46	46	46	—	—
Melisma score							
With default parameters	0.814	0.899	0.928	0.966	0.998	36	0.931
Without length factor	0.664	0.774	0.895	0.955	0.997	36	0.877
With harmonic factor	0.707	0.926	0.921	0.965	0.997	37	0.923
KP Performed Corpus (19 excerpts)							
# of excerpts with data	9	18	19	19	19	—	—
Melisma score							
With default parameters	0.984	0.778	0.922	0.945	0.960	15	0.908
Without length factor	0.620	0.524	0.740	0.850	0.937	13	0.743
With harmonic factor	0.892	0.855	0.880	0.947	0.963	15	0.909

Table 3. Tests of the Melisma Model on Beatles Songs

<i>Input piece</i>	<i>Level 3</i>	<i>Level 2</i>	<i>Level 1</i>	<i>Level 0</i>	<i>Level -1</i>	<i>Correct Offset?</i>	<i>Overall Score</i>
"Yesterday"	0.836	0.755	0.958	0.986	1.000	Yes	0.907
"Michelle"	1.000	0.971	0.971	0.980	1.000	Yes	0.984

automatically from the resulting beatlists, and compare these to the correct note-address files to see if the change resulted in better, worse, or unchanged performance.

As an example, one might wonder how crucial the consideration of length (the preference for aligning beats with longer notes) is for the meter-finding process. This was examined by altering a parameter of the Melisma system so that all notes are assumed to be just 0.1 sec in length (shorter than the vast majority of actual notes)—thus, in effect, removing length distinctions between notes. The results are shown in Table 2. For the KP Corpus, ignoring note-length distinctions results in a fairly modest decrease in performance from 0.931 to 0.877; for the KP Performed Corpus, the loss of length information had a greater effect, reducing performance from 0.908 to 0.743.

Efforts are also underway to improve the Melisma model through the incorporation of other factors. One obvious factor to include is harmony. It is

generally agreed that harmonic structure can greatly influence metrical structure (particularly at higher metrical levels), in that strong beats tend to be perceived at points of harmonic change (Lerdahl and Jackendoff 1983; Temperley 2001). To incorporate the influence of harmony, the KP Corpus was run through the Melisma harmony program, creating a harmonic analysis of segments labeled with roots; the notelists plus harmonic information were then "piped" back into the meter program, modified to include a preference for strong beats on changes of harmony. After some parameter tweaking, the best performance that could be obtained is shown in Table 2 for both the KP Corpus and the KP Performed Corpus. For both corpora, it can be seen that the score for level 2 is indeed somewhat improved, but scores for levels 3 and 1 are worsened; overall, the "harmonic-factor" model and the default model are roughly equal in performance (the difference between their overall scores is less than 1 percent for both the quantized and per-

formed corpora). In situations such as this, a larger corpus would be desirable; differences in performance of 1 percent are probably not significant on the KP Corpus. (It would also be desirable to have one large portion of the corpus for model development and parameter-tuning, and another "held-out" portion for testing.)

Conclusions

It is hoped that the system presented here will facilitate the testing and comparison of metrical models. Although the system does have certain limitations, as discussed above, it offers a rational way of comparing metrical structures whose results accord reasonably well with intuition. The KP Corpus presented here also has limitations, especially the fact that it represents only common-practice music, which may not be fair to the aims of some metrical models. Of course, the note-address system could also be used with a different corpus (as illustrated by the tests on Beatles songs presented above), providing that the corpus is annotated with note-address information. Notice that the system does not in any way require note addresses with five levels; it could function perfectly well with, say, three levels.

Perhaps the greatest limitation of the note-address system is that it is limited to symbolic input. This is unfortunate, given the considerable number of models proposed in recent years that take audio data as input. The note-address system could, in principle, be used with audio models. The problem is that the indeterminacy of beat locations encountered with symbolic input is vastly greater with audio input. Even with a single note, understood as coinciding with a beat, it is often unclear where exactly the note begins (and thus where the beat is located). To solve this problem, the input files would have to be annotated with markers indicating the beat positions; at this point the approach becomes very similar to that of Goto and Muraoka (1997). However, the idea of note addresses might still be useful; even with audio input, one might argue that some beat locations are more certain than others (e.g., the location of a beat

that coincides with a note is more certain than a beat that does not), and a metrical model should perhaps only be evaluated on its identification of the more certain ones. Another problem is that, in the symbolic-input case, the model is given the address locations—i.e., the note onsets—whereas in the audio-input case, it is part of the model's task to find these locations. (In the audio situation, a model might assert an address where there was not one or fail to assert one where there was one.) Thus, the usefulness of the note-address system for audio-input metrical models remains to be seen.

Acknowledgments

Thanks are due to Henkjan Honing for valuable feedback on an earlier version of this article.

References

- Allen, P., and R. Dannenberg. 1990. "Tracking Musical Beats in Time." *Proceedings of the 1990 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 140–143.
- Black, E., et al. 1991. "A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars." *Proceedings of the Fourth DARPA Speech and Natural Language Workshop*. San Francisco: Morgan Kaufman, pp. 306–311.
- Brown, J. C. 1993. "Determination of the Meter of Musical Scores by Autocorrelation." *Journal of the Acoustical Society of America* 94:1953–1957.
- Cemgil, A. T., P. Desain, and B. Kappen. 2000a. "Rhythm Quantization for Transcription." *Computer Music Journal* 24(2):60–76.
- Cemgil, A. T., et al. 2000b. "On Tempo Tracking: Tempogram Representation and Kalman Filtering." *Journal of New Music Research* 29:259–273.
- Chafe, C., B. Mont-Reynaud, and L. Rush. 1982. "Toward an Intelligent Editor of Digital Audio: Recognition of Musical Constructs." *Computer Music Journal* 6(1):30–41.
- Desain, P., and H. Honing. 1989. "The Quantization of Musical Time: A Connectionist Approach." *Computer Music Journal* 13(3):56–66.
- Desain, P., and H. Honing. 1999. "Computational Models

- of Beat Induction: The Rule-Based Approach." *Journal of New Music Research* 28:29–42.
- Dixon, S. 2001a. "Automatic Extraction of Tempo and Beat from Expressive Performances." *Journal of New Music Research* 30:39–58.
- Dixon, S. 2001b. "An Empirical Comparison of Tempo Trackers." *Eighth Brazilian Symposium on Computer Music*, n.p., 832–840.
- Eck, D. 2001. "A Positive-Evidence Model for Rhythmical Beat Induction." *Journal of New Music Research* 30:187–200.
- Gabrielsson, A. 1973. "Studies in Rhythm." *Acta Universitatis Upsaliensis* 7:3–19.
- Goto, M. 2001. "An Audio-Based Real-Time Beat Tracking System for Music with or without Drum-Sounds." *Journal of New Music Research* 30:159–171.
- Goto, M., and Y. Muraoka. 1997. "Issues in Evaluating Beat Tracking Systems." *IJCAI-97 Workshop in Issues in AI and Music—Evaluation and Assessment*, n.p., 9–16.
- Heijink, H., et al. 2000. "Make Me a Match: An Evaluation of Different Approaches to Score-Performance Matching." *Computer Music Journal* 24(1):43–56.
- Jones, M. R., et al. 2002. "Temporal Aspects of Stimulus-Driven Attending in Dynamic Arrays." *Psychological Science* 13:313–319.
- Kostka, S., and D. Payne. 1995. *Workbook for Tonal Harmony*. New York: McGraw-Hill.
- Large, E. W., and J. F. Kolen. 1994. "Resonance and the Perception of Musical Meter." *Connection Science* 6:177–208.
- Lee, C. 1991. "The Perception of Metrical Structure: Experimental Evidence and a Model." In P. Howell, R. West, and I. Cross, eds. *Representing Musical Structure*. London: Academic Press, pp. 59–127.
- Lerdahl, F., and R. Jackendoff. 1983. *A Generative Theory of Tonal Music*. Cambridge, Massachusetts: MIT Press.
- Longuet-Higgins, H. C. 1976. "Perception of Melodies." *Nature* 263:646–653.
- Longuet-Higgins, H. C., and C. S. Lee. 1982. "The Perception of Musical Rhythms." *Perception* 11:115–128.
- Longuet-Higgins, H. C., and M. J. Steedman. 1971. "On Interpreting Bach." *Machine Intelligence* 6:221–41.
- Manning, C. D., and H. Schütze. 2000. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: MIT Press.
- Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz. 1993. "Building a Large Annotated Corpus of English: The Penn Treebank." *Computational Linguistics* 19:313–330.
- McAuley, J. 1994. "Time as Phase: A Dynamic Model of Time Perception." *Proceedings of the Sixteenth Annual Meeting of the Cognitive Science Society*. Hillsdale, NJ: Erlbaum, pp. 607–612.
- Miller, B. O., D. L. Scarborough, and J. A. Jones. 1992. "On the Perception of Meter." In M. Balaban, K. Ebcioglu, and O. Laske, eds. *Understanding Music with AI: Perspectives on Music Cognition*. Cambridge, Massachusetts: AAAI Press, pp. 428–447.
- Palmer, C. 1997. "Music Performance." *Annual Review of Psychology* 48:115–138.
- Palmer, C., and P. Q. Pfordresher. 2003. "Incremental Planning in Sequence Production." *Psychological Review* 110:683–712.
- Parncutt, R. 1994. "A Perceptual Model of Pulse Salience and Metrical Accent in Musical Rhythms." *Music Perception* 11:409–464.
- Povel, D.-J., and P. Essens. 1985. "Perception of Temporal Patterns." *Music Perception* 2:411–440.
- Raphael, C. 2001. "Automated Rhythm Transcription." *Proceedings of the 2nd Annual International Symposium on Music Information Retrieval*. Bloomington, Indiana: Indiana University, pp. 99–107.
- Rosenthal, D. 1992. "Emulation of Human Rhythm Perception." *Computer Music Journal* 16(1):64–76.
- Rowe, R. 1993. *Interactive Music Systems*. Cambridge, Massachusetts: MIT Press.
- Scheirer, E. D. 1998. "Tempo and Beat Analysis of Acoustic Musical Signals." *Journal of the Acoustical Society of America* 103:588–601.
- Sethares, W. A., and T. W. Staley. 2001. "Meter and Periodicity in Musical Performance." *Journal of New Music Research* 30:149–158.
- Sloboda, J. A. 1983. "The Communication of Musical Metre in Piano Performance." *Quarterly Journal of Experimental Psychology* 35:377–396.
- Spiro, N. 2002. "Combining Grammar-Based and Memory-Based Models of Perception of Time Signature and Phase." In C. Anagnostopoulou, M. Ferrand, and A. Smaill, eds. *Music and Artificial Intelligence*. Berlin: Springer, pp. 183–194.
- Steedman, M. 1977. "The Perception of Musical Rhythm and Metre." *Perception* 6:555–569.
- Temperley, D. 2001. *The Cognition of Basic Musical Structures*. Cambridge, Massachusetts: MIT Press.
- Temperley, D., and D. Sleator. 1999. "Modeling Meter and Harmony: A Preference-Rule Approach." *Computer Music Journal* 23(1):10–27.
- van Zaanen, M., R. Bod, and H. Honing. 2003. "A Memory-Based Approach to Meter Induction." *Proceedings of ESCOM 2003*, n.p., pp. 250–253.

Appendix A

All of the materials (data files and programs) described here are available at the Melisma Web site, www.link.cs.cmu.edu/melisma. (Programs are written in C and are available in source form.) These include the following:

1. Notefiles for the complete KP Corpus (notefiles/kp) and KP Performed Corpus (notefiles/kp-perf). (These files are also available as MIDI files: midifiles/kp, midifiles/kp-perf.)
2. The correct note-address files for the KP Corpus and KP Performed Corpus (nafiles/kp-correct, nafiles/kp-perf-correct).
3. compare-na: a program that compares two note-address files and evaluates their similarity.
4. tally-na: a program that takes a series of outputs of compare-na and averages them.
5. The source code for the Melisma meter-finding program.
6. gen-add: a program for generating note-address files from the "note-beat files" (note-list plus beatlist) produced by the Melisma meter program. (This could be useful for those who wish to experiment with modifications of the Melisma program. It may also be useful, perhaps with some alterations, to those whose models already produce a note-beat file or something similar to it and who wish to convert this to note-address format.)

Copyright of Computer Music Journal is the property of MIT Press and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.