Interactive Improvisational Music Companionship:

A User-Modeling Approach

BELINDA THOM
School of Computer Science, Carnegie Mellon University,Pittsburgh, USA
email: bthom@cs.cmu.edu; http://www.cs.cmu.edu/˜bthom

1. Introduction

In this paper we present a novel domain for user modeling and intelligent agents – agents as specific users' *improvisational music companions (IMCs)*. We also introduce *Band-OUT-of-a-Box (BoB)*, an agent designed to trade live, customized solos with a specific improvising musician/user.

A primary technical challenge in this domain is to computationally specify the agent's musical intelligence, a highly personalized and context-dependent concept. To address this issue, we develop an unsupervised machine learning algorithm that can learn to model the musician's behavior after listening to them improvise for awhile. This model learns to abstract bars of solos into user-specific playing modes, which allows the agent to perceive what the musician plays in a user-specific manner.

User-specific perception is only part of what the agent needs to respond to the musician. Another key challenge is to 'close the learning loop.' We address this issue by incorporating the learned model's parameters back into a stochastic process that, when sampled, can produce solos that realize specific user playing modes. By integrating these two components together, BoB can participate in user- and context-specific solo trading.

These technologies are evaluated in two simulated real-time contexts, the improvisations of saxophonist Charlie Parker and violinist Stephane Grappelli.

1.1. Motivation

Imagine an agent who is your own personal music companion: someone who plays music with you, trades licks and riffs with you, improvises with you. This companion gets to know you and your musical personality, listening and interacting with you in such a way as paraphrased by drummer Keith Copeland:

> ... he can answer me halfway through my phrase, bringing me to the point where I can actually sing what he's going to play next, and then, instead of playing that, he'll play something against it which compliments what I'm singing in my head.

This kind of interaction, which inspires our research, paraphrases Copeland's musings about improvising with jazz pianist Ahmad Jamal.[1] Toward this end, we have created BoB, an agent that uses machine learning techniques in order to trade solos with its user in a musician- and context-specific manner.

1.2. A New Agenda For Musical Intelligence

Improvised companionship motivates a different research agenda than is typically pursued in artificial intelligence (AI) based music systems (Thom, 2000a). By definition, an IMC is a specific musician's personal companion, and this directly affects how we think about the agent and specify what is important when simulating its musical intelligence. We envision IMCs as tools for enhancing musicians' solo practice efforts. IMCs must be able to transform what the user plays in interesting new ways, reflecting their performances back so that the comingling that results makes the user's spontaneity more tangible and explorable. What is needed is an extremely flexible and adaptive model of melodic improvisation that – and this is the key point – can leverage off of the spontaneous creativity of the user.

Providing an adaptive and robust basis for reasonably competent musical behavior in a wide variety of improvisational settings becomes more important than simulating expertise in a specific genre. For example, we envision users and IMCs collaborating in many different contexts, e.g.:

- the rhythmic, modal types of playing one finds in Ali Farque Toure's or Dr.Didg's music;
- well-delineated chord progressions, as one might find in blues, bluegrass, folk, or standard jazz tunes; and
- unaccompanied settings where, aside from adhering to a fixed tempo, improvised melodies are unconstrained.

To address this need, BoB's architecture was purposefully designed at its most basic level to function in the absence of specific jazz or blues theories, and this is one reason why neither harmonic nor metrical knowledge was hand-coded into the representation. This approach is in direct contrast to more knowledge-intensive systems (Cope, 1992; Pennycook and Stammen, 1993; Pachet, 2000) where specific rules are fine-tuned for a particular setting. Instead, we develop a computational model of improvisation that tightly couples with and fully functions from within the *sole context of* the user's improvised melodies. In terms of computer music, this unique agenda is one of BoB's more interesting features.

Because IMCs cannot rely on having an expert around to 'twiddle the settings' or directly supervise musical behavior, using machine learning (ML) to replace

[1] As compiled in (Berliner, 1994).

explicit authoring is a primary concern. ML is applied to configure a computational improvisation model for a particular arbitrary musician playing at a particular arbitrary time and in a particular arbitrary context. However, in order for ML to be useful, training data collection must be natural and non-obtrusive, accommodating amateur musicians with varying levels of skill.[2] An important aspect of BoB's learning scenario is that it only requires the musician to improvise. They do not have to label or analyze any data.

Learning must also be feasible when there is little training data because this allows the IMC to model the musician's playing style 'right now,' in the current solo-trading context. In contrast, researchers typically combat over-fitting by training their models on a large corpus of songs. For example, many compositions from J.S. Bach, M. Reger, and S. Scheidt were used to train neural networks that could identify their composing styles (Hörnel and Menzel, 1998). With IMCs, corpus-based training is less appropriate because musicians will play very differently at different times and in different settings. We propose accommodating smaller data sets by simplifying the underlying mathematical model that is to be learned, the assumption being that a less accurate model trained on more immediately relevant data will provide a more intimate and appropriate interaction with the user than a more complex model that was trained on a larger hodge-podge of the musician's behavior would.

1.3. BAND-OUT-OF-A-BOX

BoB is an IMC whose purpose is trading live, improvised solos with its user in real time. This system explores the idea that a solo-trading companion can be *operationalized* – i.e., algorithmically specified – using a probabilistic melody improvisation model that fits its parameters to the user's improvisations in an unsupervised fashion.

1.3.1. *Band-in-a-Box*

The name 'BoB' pays tribute to *Band-in-a-Box (BiB)* (PGMusic, 1999), a commercial software product that provides fixed accompaniment to musicians while they improvise. Although BiB has enhanced many improvisers' at-home practice efforts, it neither listens to nor responds to what the user plays.[3] BiB's success has much to do with the fact that it comes with a huge library of expertly crafted styles, offering a wide range of choice over what type of accompaniment can be performed (e.g., rock, latin, jazz, etc.).

[2] Providing non-obtrusive and/or user-friendly data collection is also an important consideration in the learn-based systems of Mukherjee et al. and Eliassi-Rad and Shelvik (this issue).

[3] While the new version can improvise, creating solos and/or harmonies, this 'creativity' is not interactive.

Should a user care to build their own accompaniment style, licks can be recorded and various parameters can be modified to control how these licks are combined and applied. However, from an artistic standpoint, this interface is quite restrictive because it means that the musician must configure the system's aesthetics. The difficulty is that the user must somehow manage to express – i.e., to *author* – their intuitive musical desires using the program's constructs. This authoring can require a great deal of experimentation, where the user continues to twiddle settings until some form of 'interesting' behavior emerges.

1.3.2. *Live Interaction*

General-purpose music systems that interact with human performers present a number of difficult engineering problems and are the subject of much research (Rowe, 1993; Winkler, 1998; Raphael, 1999; Cemgil and Kappen, 2001). In order to simplify BoB, three important issues are ignored: pitch-tracking, tempo-tracking, and rhythm quantization (RQ). Rather, we assume that BoB already receives the musician's input in a *transcribed* form, i.e., raw performance data has already been transformed into a score-based symbolic notation. Our reliance on a transcribed notation gives us the freedom to focus on a most crucial technology: to develop a model for perceiving and generating melodic sequences in a user-specific manner. An important aspect of this model is that it can handle sequences of arbitrary length. Currently, melodies are segmented on a per-bar basis.

Until real-time transcription is provided, BoB's live interaction with the user is *simulated*. Simulation is accomplished by parsing input from human-transcribed solo files in real-time. With respect to the model's timing and data flow, simulated interaction is sufficient for evaluating what BoB can learn and to what extent the learning loop can be closed. However, only in a live setting can the user reciprocate and respond to the system.

Live interaction is such an important part of BoB's evaluation that we have already begun to integrate its functionality. Although in general, automated audio transcription is difficult, the task is feasible in BoB for several reasons. First, tempo-tracking concerns are eliminated by requiring that the user 'keep up' with the accompaniment. Second, human-computer interaction is mediated by a *lead sheet*, an interface in which the musician authors those aspects of the interaction that are easy for them to specify up front.[4] Third, RQ is simplified. In particular, quantization is handled periodically, once per bar, and only a restricted set of transcribed rhythms will ever be produced. In this setting, real-time transcription should be feasible using a probabilistic Bayesian RQ approach (Cemgil et al., 2000) and off-the-shelf pitch-to-MIDI technology (Franklin, 2001).

[4] Tempo, rhythmic (and optional harmonic) background, and who will solo where and when are thus specified. In live improvised settings, musicians typically specify similar quantities before playing together, albeit in less precise, more fluid ways. Non-interactive accompaniment in BiB successfully relies on a similar lead-sheet strategy.

1.3.3. *Solo Trading*

BoB's interactivity is staged in terms of call and response. During a *call*, the callee-soloist improvises and the responder-soloist listens. During a *response*, the responder 'answers' the preceding call. When repeated over and over, the distinction between call and response blurs as each soloist will influence the other.

Trading fours is an interaction scheme where soloists repeatedly call and respond in four-bar chunks. Figure 1 provides an example, a different staff transcribed for each soloist. The musician's call is shown on the top staff. BoB is replying (bottom staff), having just performed its first two bars, now scheduling its third, and creating its fourth. The accompanist's part, which BoB plays back non-interactively, is listed underneath the bottom staff in standard chord notation.

Figure 1. BoB and the Musician Trade Fours.

1.3.4. *High-Level Architecture*

BoB is composed of the modules enclosed within the double-lined box in Figure 2; everything else is regarded as the *environment*. The lead sheet guides the high-level interaction of BoB and the user, only one of whom is playing at a given time. Within BoB, separate threads (dashed components) process real-time MIDI events. In this article, we focus on the *musically-intelligent components* that are outlined in bold.

In order to customize the agent to a particular user and improvisational context, BoB collects training data by recording what the musician plays during a preliminary *warmup* session. At this point, BoB is *off-line* and behaves in BiB's capacity, providing background accompaniment at a fixed tempo. On top of this accompaniment, the user improvises freely, constrained only in that they must 'keep time.' The purpose of this warmup is to configure the *Off-line Learned Knowledge (OLK)* Component, customizing BoB to the musician's current improvisational mood.

When BoB goes *on-line*, trading real-time solos with the user, the same background accompaniment will be used in order to increase the odds that OLK's knowledge remains relevant. A bar at a time, the *Perception* Component uses OLK to transform the musician's call into an abstraction that highlights salient aspects of their melodic behavior. The entire call is recorded by the *History* Component, producing a sequence of user-specific abstractions. A bar at a time, the *Generation* Component uses OLK to guide its inversion of this abstract sequence. In this way, BoB creates a novel user- and context-specific response.

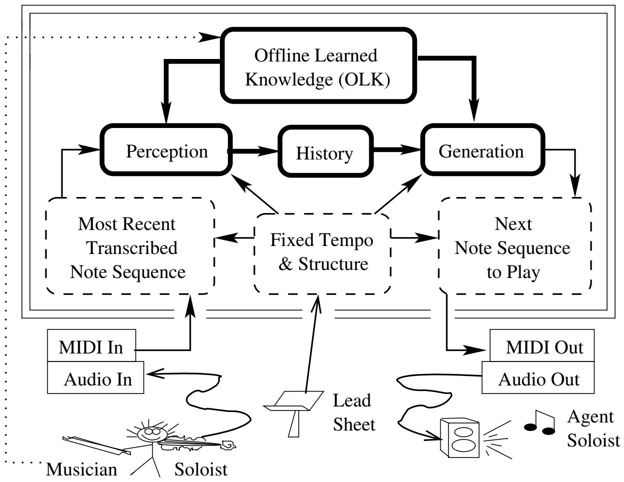


Figure 2. The Band-OUT-of-a-Box Architecture.

1.4. THE USER MODEL

BoB will be successful provided it can intimately couple with the musician – at all times they must feel listened to and responded to. A most basic hypothesis in our user model is that the necessary intimacy can be simulated provided the agent is powerful enough to detect and respond to the various *playing modes* that the musician uses during different parts of their improvisations in order to make them interesting.

1.4.1. *A Conglomerative View of Melody*

We assume that each user playing mode corresponds to a preference for certain pitch-class,[5] intervallic, and directional trends. Three different *melodic views* are constructed for a transcribed segment:

1. *pitch-class (PC)* histogram x^{P} records the number of times that certain pitch classes are used;
2. *intervallic (INT)* histogram x^{I} records the frequency with which various intervals between adjacent pitches occur; and
3. *directional (DIR)* histogram x^{D} records information about how the segment's melodic direction changes over time.

Thus defined, x^{P} relates to a segment's underlying tonality, x^{I} relates to how continuous or discontinuous it is, and x^{D} quantifies aspects of its melodic contour. For example, one playing mode might correspond to the user's penchant for a certain

[5] Pitch class is a psychoacoustic term and should not be confused with the playing-mode classes that BoB's learning algorithm infers.

musical scale whose tones are applied in such a way that the resulting segments tend to descend in a series of small steps.

A segment's overall view is referred to as *conglomerative (CGL)* histogram x^{C}, which consists of the three individual parts:

$$x^{\mathrm{C}} = \langle x^{\mathrm{P}}, x^{\mathrm{I}}, x^{\mathrm{D}} \rangle.$$

With this tuple/super-script notation, a variety of decompositions are conveniently at hand, e.g., if x^{C} is referred to, then the meaning of x^{P} is also clear. The term 'conglomeration' also highlights that each view is constructed and modeled in isolation. Although the content of the different histograms are related, no explicit dependences are considered, e.g., model Ω^{P} is only trained using data set X^{P}.

1.4.2. *Learning User-Specific Playing Modes*

BoB's user-model revolves around mapping $\mathcal{M}$, where histogram *feature vector* x is mapped onto one-of-K user-specific *playing modes* y:

$$\mathcal{M} : x \longrightarrow y.$$

Rather than trying to 'best explain' some target function (supervised learning), OLK seeks to 'best explain' the distribution of features within the training set, an unsupervised learning task. In BoB, this mapping is probabilistically modeled. We assume that an underlying *mixture distribution*, parameterized by Ω, generated the histograms in the training data. Learning amounts to estimating the parameters for some number of underlying components, K, thus fully specifying Ω. The different components of this mixture are inferred in such a way that probabilistically similar data points are 'better explained' by certain component distributions. Classifier $\mathcal{M}$ then works by mapping feature vector x onto that component that appears to have been its most likely generator. Thus defined, playing mode, cluster, component, and class will be used interchangeably.

1.4.3. *User- and Context-Specific Melody Generation*

In BoB's user-model, response is formulated in terms of inverting the abstractions that describe the user's call. From this vantage point, our strategy is similar to the one taken by Eliassi-Rad and Shelvik (this issue), for their system also learns a fairly abstract mapping and addresses solving the corresponding 'nearly inverse problem.'

In BoB, this inversion amounts to transforming a call's melody into a novel response that exhibits the same sequence of higher-level user-specific abstractions. What we are assuming is that such a capability will provide the user with a sense of being listened to and responded to. Since the response evolves a bar at a time, this inversion must also be able to constrain its generation procedure so as to ensure a smooth link into what was just played. Our generation scheme will be able to service an even more general set of constraints, providing a very powerful and open-ended mechanism for incorporating domain knowledge and/or user desires into the system when needed.

A hierarchical representation of a melody's rhythmic content will naturally lend itself to a stochastic method for transforming a call's rhythm into a novel but related response. What is more difficult is to assign a sequence of pitches to the durations that result. The problem is that a variety of perhaps conflicting PC, INT, and DIR trends and general purpose constraints must be simultaneously accommodated. These issues are tackled within the framework of a stochastic walk through a probabilistic graph whose vertices represent a set of allowed pitch values. By weighting the edges that connect these vertices in accordance with both the desired playing mode's learned parameters and any required external constraints, solutions can usually be found.

1.5. The Rest of this Paper

This article proceeds by taking an in-depth look at the nature of IMCs by examining their relationship with believable agents. We then describe how BoB's intelligent music components are integrated into a unified system. Next, BoB's core user-modeling technologies are detailed: melodic representation, playing-mode learning, and playing-mode-driven generation. The model's performance is then evaluated in two different training contexts, the improvisations of saxophonist Charlie Parker and violinist Stephane Grappelli. Finally, we describe some of the system's current limitations, our future research plans, and related work.

2. Believable Agents

In this section, subtle yet crucial aspects of IMCs are highlighted in the course of examining their relationship with believable agents (BAs).

Just as BAs presented a new paradigm to the AI community (Mateas, 1999), IMCs present a new paradigm for intelligent music systems. Important commonalities and distinctions can be drawn between BAs and IMCs. Such comparisons provide deeper insight into the nature of BoB's technical requirements, for certain BA-centric perspectives naturally simplify the agent-building task, while other perspectives are inadequate and need to be extended or reinvented. The domains are related by their need to resolve the inherent conflict that arises when an intelligent system and an artist interact. The key is to find the right balance between artistic control (e.g., authoring) and autonomous intelligence. For an IMC, this amounts to automating musically-appropriate behavior in accordance with the musician's own improvisations.

Believable agents research explores building interactive virtual worlds whose characters and story lines are believable. While the nature of believable agents is debated, a central theme is creating interactive, autonomous agents that exhibit rich and compelling personalities (Elliot and Brzezinski, 1998). The most basic link between BAs and IMCs is their shared audience- and artist-centered focus, whereby

> [...] the building a new kind of cultural experience: engaging, compelling, and hopefully beautiful and profound [...] (Mateas, 1999)

is embraced. Of enormous benefit here is the belief that artistic goals are at least as valuable as purely technical ones, that good art transformed into a computational setting breeds good science, but that the converse is not necessarily true (Mateas, 2001).

2.1. Useful Methodologies

In creating computational interactive characters with the same rich personalities as those created by artists, BA researchers have attempted to understand the reflections of practicing character artists and to integrate these insights into their methodology. Not surprisingly, the reflections of practicing improvisers have proven similarly useful in our research, e.g., (Sudnow, 1978; Johnson-Laird, 1991; Bailey, 1992; Berliner, 1994).

How inherently artistic pursuits influence BA methodology is instructive. Because there is an emphasis on *building* interactive characters, an important goal is to provide character artists with the same level of artistic control that, for example, animators have. The analogous goal for BoB is to both foster intimacy, which requires a tight agent/musician coupling, and ensure autonomy, which allows the agent to generate novel material.

Some useful BA methodologies and their application to BoB's design are highlighted below.

Stress specificity. While traditional AI systems seek to capture the general theory of some domain, BoB must capture the notable aspects of its user's improvisations. Derek Bailey highlights this point:

> There is no general widely held theory of improvisation and I would have thought it self-evident that improvisation has no existence outside of its practice. (Bailey, 1992)

Traditionally, computational jazz systems have been rule-based, instantiating some combination of textbook theories about jazz (Levitt, 1981; Pennycook and Stammen, 1993; Pachet, 2000). BoB's methodology is a significant point of departure.

Measure success in terms of audience perception. Classical AI systems seek to optimize some form of objective measure. BoB must first be built and then evaluated in terms of how engaging its interactions with the user are.

Have strong affinities with behavioral AI. In particular, BoB's technology is focused on capturing broad, shallow musical capabilities. Fortunately, this insight makes the agent-building task more doable. Where a more traditional AI-based music approach would focus on realizing a particular musical expert, BoB merely needs to be able to provide a modicum of customized musical appropriateness in a variety of improvisational settings.

Assume the audience is willing to suspend their disbelief. This insight is most important because it directly affects how to think about the user (Sengers, 1998). With BoB, it is safe to assume that the musician *wants* to use it to share in an engaging musical exchange. As such, the user will be more likely to overlook BoB's shortcomings.[6] BiB's success leverages off of this fact. Many of its accompaniment styles are somewhat canned, yet the experience of playing with a 'cheesy' band, versus playing with no band at all, is a powerful one.

2.2. Points of Departure

The typical focus of BA research is character, personality, and story — expressive mediums that heavily rely on language-based, literal reasoning. It is because of this focus that some BA technologies do not provide viable solutions for controlling the spontaneous generation of musician- and context-specific melodic response.

With BAs, it makes sense to focus on *authorship*, to provide artists with a set of abstractions, tools, and programming language constructs that allow them to create the interactive, artistic experiences they envision. For example, it is technically feasible to consider improvised story telling precisely because one can reason about and author significant amounts of an agent's behavior up front. Thus, when building the *Oz* world (Bates, 1992), a character artist could specify that the shy *Woggle* always quivers when frightened, except when their fear is larger than some internally specified amount, in which case they become paralyzed. The important point is that operationally useful behavioral aspects can be specified: shake a certain number of cycles per minute, enlarge the eyes by a certain percentage when paralyzed, etc.

Unfortunately, with improvised musical expression, these more tangible aspects of character and personality break down. Musicians do speak of personality, but they usually do so in frustratingly vague terms. Imagine trying to operationalize jazzist Ronnie Scott's notion of such a concept:

> "I would like ideally to express my, I don't know, personality or whatever, musically, to the limits of my ability [.... to ...] play in such a way that it would be recognizable as me, and it would express something to people about the way I feel about things." (Bailey, 1992)

In BoB, it makes sense to author those things that can be easily specified in advance via the lead-sheet abstraction. The crucial challenge becomes providing a mechanism for spontaneously constructing context- and musician-specific melodies. In this domain, we cannot expect musicians to author operational details concerning what note sequence to play next. Derek Bailey, who interviewed an extensive range of a practicing improvisers, sums it up best:

> The improvisers I spoke to, in almost all cases, did not find any sort of technical description adequate. It wasn't that they weren't interested in technical matters.

[6] Such as performing an otherwise interesting solo mechanically.

> They just did not find them suitable for illuminating improvisation. They finally chose to describe it in so-called 'abstract' terms. And it became clear that, whatever its deficiencies, this is the best method available. (Bailey, 1992)

Fundamentally, BoB needs to be able to operationalize its behavior in a domain that is heavily structured by 'just-barely-conscious skills that humans cultivate,' and in this sense, our research is very similar in spirit to that of Cassell and Bickmore (this issue).

Whether or not a usable ontology for explicit melodic authoring *could* exist, we certainly do not have the ability to specify such behaviors in operational terms at the moment. Nonetheless, BoB needs to be able to understand the abstractions in which its user speaks, which requires having an operational musical vocabulary that can abstractly perceive the musician's call and respond accordingly. We propose to reduce the authoring burden by using ML technology to operationalize the perception of a user's distinct playing modes. Generation, in turn, is obtained 'for free' by inverting this learned knowledge. This approach contrasts with the philosophy adopted by some BA researchers who discount ML as undesirable because it might hinder or automate away aspects of character and plot that the author explicitly wishes to control.

Two other distinctions between IMCs and BAs are worth mentioning. First, a BA is typically authored by a team of experts (character artists, graphic artists, etc.). The audience that experiences their world is a completely separate entity. With IMCs, the improviser is both the audience and the author of the experience.[7] Second, one can make the argument that a traditional BA's performance is more straightforward to evaluate because one can focus on a user's engagement with a character post-facto. It is not clear that anything but an improviser's transitory, physical, and immediate musical experience is worthy of attention. On this point, organist Stephen Hicks notes that, with respect to the evaluation of his improvisations,

> "It's either good or bad but if you listen to an improvisation over and over again, it just gets worse. [...] It's something that should be heard, enjoyed or otherwise, and then completely forgotten." (Bailey, 1992)

One way in which we might address this issue in the future is to monitor a musician's feedback on-the-fly, using physiological or visual feedback.

3. Musically-Intelligent Components

In this section, we elaborate on how BoB's musically-intelligent components are integrated into a unified architecture, which amounts to specifying how the components operate upon and transform their inputs and interact with one another and the user.

[7] In this context, 'author' does not mean someone who uses authoring tools for an IMC's configuration. Rather the word reflects the fact that the musician's improvisations contain within them the essence of what constitutes musically-appropriate behavior.

3.1. THE OFF-LINE LEARNED KNOWLEDGE COMPONENT

Recall that OLK infers the parameters for its user-model by learning to map CGL feature vectors onto discrete playing modes. This mapping is constructed so that the improvisations collected for the user during their warmup session appear maximally likely.

A warmup session might last for five or ten minutes. Beforehand, the user creates a lead sheet that defines the accompaniment over which they desire on-line interaction. The user then solos on top of this accompaniment, and N bars of transcription $\{s_1, s_2, \cdots .s_n, \cdots, s_N\}$ are collected. As will be described in Section 4, each transcribed segment is then converted into a CGL feature vector. In this way, training set:

$$X^{\mathrm{C}} = \{x_1^{\mathrm{C}}, x_2^{\mathrm{C}}, \cdots, x_n^{\mathrm{C}}, \cdots, x_N^{\mathrm{C}}\}$$

is constructed.

Using the methodology that will be described in Section 5, the parameters of the user model are estimated from X^{C}. This model is learned before on-line interaction takes place. In other words, we are inherently assuming that the probability distribution over the user's melodic behavior during warmup and solo trading does not change.[8] Given that the same accompaniment is used in both settings and that warmup takes place immediately prior to solo trading, this assumption is reasonable.

3.2. ON-LINE INTERACTION

The user-model operationalizes a customized, playing-mode specific view of what the user plays. This knowledge is explicitly used by both the Perception and the Generation Components.

3.2.1. *The Perception Component*

For real-time Bar r, perception transforms a call into several new levels of abstraction. Perception operates on two inputs, transcribed segment s_r and user-model Ω^{C}.[9] Two stages of abstraction are output. In the first stage, the segment's basic representation is encoded in tree v_r and corresponding histogram x_r^{C}. In stage two, the user model customizes x_r^{C}, mapping it into playing mode y_r^{C}. Since this mapping is probabilistic, likelihood l_r^{C} can also be quantified. Given the user model and

[8] Fortunately, it will be easy to relax this assumption because the parameter estimation technique is MAP-based, i.e., Ω is estimated so as to locally maximize the posterior distribution (Mitchell, 1997). To extend this model on-line, Ω will just need to incorporate new evidence every time a bar of data arrives. To reestimate the number of components in a user model on-line might be more problematic.

[9] Subscripts n and r to distinguish between training and on-line data respectively.

estimate y_r^{C}, we interpret l^{C} in relation to how *musically surprising* x_r^{C} appears to be.

In the left plot of Figure 3, an example of how BoB might perceive a musician's fours are presented. The discrete symbols (square, circle, diamond) emphasize each mode's discrete and nominal character. In the right plot, perception's generalization mechanism is demonstrated by the many different transcriptions that map into the same cluster.

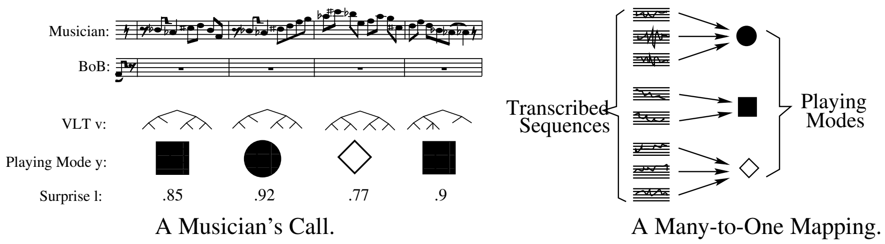


Figure 3. Perception.

3.2.2. *The History Component*

History stores the current musical environment for the evolving musician's call. Since each Bar r follows the *most recently played note (MRP)* of the previous bar, it is convenient to introduce mrp_r as the placeholder for this information. In total, history records the following data for each call bar r: s_r, v_r, mrp_r, x_r^{C}, y_r^{C}, and l_r^{C}.

3.2.3. *The Generation Component*

The generator creates a novel response to the musician's most recent call. Each bar is produced in real-time, one bar at a time. An example with BoB and the musician trading fours is presented in Figure 4. The interaction scheme that underlies this figure employs an extremely simple *abstract cut-and-paste* methodology. The simplicity arises because the user's abstractions in Bar r of the call are used to drive the generation of the agent's response in Bar $r + 4$.[10] The beauty of this scheme is that it provides a viable compromise between the user, who needs some degree of artistic control, and BoB, who needs an operational method for responding autonomously. The foundation of this compromise relies on synthesizing two types of information, that which the musician directly controls (via their call) and that which they have indirect influence over (via OLK).

[10] Since the underlying harmony during call and response might not necessarily be compatible, BoB's cut-and-paste operation is actually a bit more complicated. Only intervallic and directional abstractions are copied in this way. An appropriate tonal abstraction is obtained by looking at what the user played over some compatible bar of harmony.

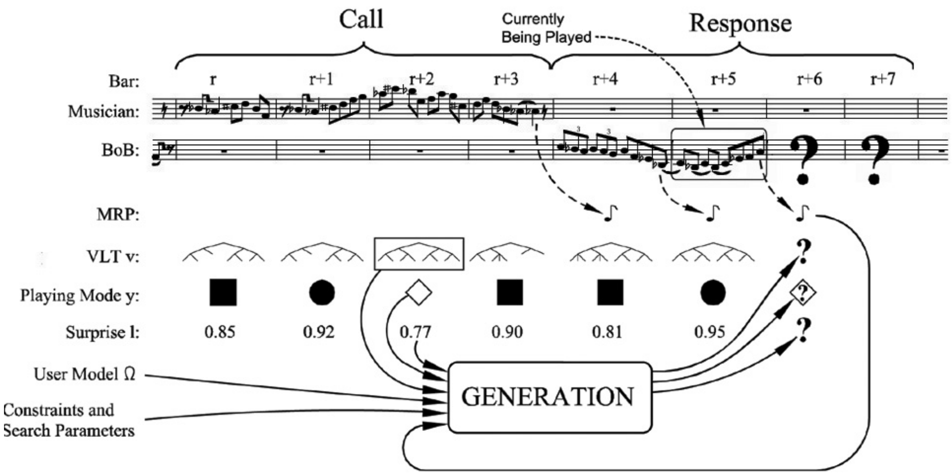


Figure 4. Generation: Call and Response.

As demonstrated in Figure 4, the generation of Bar r is driven by the following inputs: most recent context mrp_r; the relevant melodic abstraction contained in v_{r-4}, y^{C}_{r-4}, and l^{C}_{r-4}; the parameters of the user model; high-level search parameters (e.g., how long to search); and a fairly general set of optional constraints. As output, the generator creates a new solo tree, v_r, which BoB performs in real-time during Bar r. A subtle but important assumption made in this figure is that the generator is capable of inverting a particular user abstraction. This inversion is goal driven in that the solo that results will tend to exhibit a particular playing mode's trends in such a way that a particular degree of surprise is also implied. Thus, for example, the CGL trends in Bars $r + 1$ and $r + 5$ both map into the 'circle' playing mode, and the likelihood 0.92 is quite similar to 0.95. This technology, which 'closes the learning loop,' will be detailed in Section 6.

The way abstract cut-and-paste is defined, each call/response pair must occupy an identical number of bars. Each call must also occupy at least two bars. This latter requirement arises because, during the musician's first call bar, BoB waits to receive the whole segment before perceiving it. In subsequent call bars, multi-threaded techniques are used to simultaneously buffer the currently performed call bar, perceive the previously played call bar, and use this perception to compute an appropriate response for that bar.

4. Basic Solo Representation

In this section, we introduce BoB's most *basic* — i.e., non-customized – *representation*. A transcribed segment is transformed into two different data structures, a *variable-length tree (VLT)*, which provides a powerful rhythmic encoding of the segment, and a CGL histogram, which provides a more abstract view of the sequence's melodic content. Figure 5 presents an example.

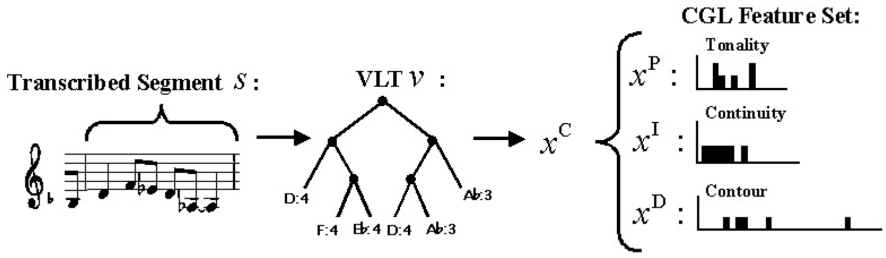


Figure 5. Converting a Bar of Solo into its Basic Representation.

4.1. Variable-Length Trees

The VLT is BoB's lowest-level melodic encoding. v is obtained by parsing a transcribed melodic sequence, producing a tree whose subdivisions are arranged according to the transcription's rhythmic hierarchy. A VLT stores information in two different data structures. *Leaf nodes* encode pitch information and *internal nodes* encode duration information.

Each leaf node contains either a pitch or a rest value. Pitch is decomposed into two parts. The *register* identifies the octave in which it resides,[11] and the *pitch class* identifies its location within the octave. In Figure 5, register and pitch-class values are separated by a colon.

A powerful aspect of the VLT is that its rhythm is entirely determined by the structure of its internal nodes because each node's duration is obtained by subdividing the duration of its parent. When deciding how to limit the structure of the VLT encoding, we drew on our experience with a wide range of published transcriptions.[12] A VLT's structure is restricted as follows:

1. each internal node must have 2 or 3 children;
2. a leaf node can occupy no less than $\frac{1}{6}$ or $\frac{1}{8}$ of a beat; and
3. a VLT is composed using the smallest number of nodes.

[11] A value from the ordered set $\{0, 1, \cdots, 7\}$. The value 4 corresponds to that octave whose pitches span from Middle C to the B above it.

[12] Blues, folk, jazz, bluegrass, swing, and cajun transcriptions (as published in *Fiddler* magazine) and artist-specific compendiums (e.g., Charlie Parker, Jeff Beck, Stephan Grappelli, Vassar Clements, Led Zeppelin, Carlos Santana).

The first two restrictions limit the number of durations that may be used to construct a VLT. To understand the last restriction, it is necessary to consider what happens when two adjacent leaves contain the same pitch value. In this case, an additional *tie* value is stored in the first leaf. This value indicates whether or not the two leaves should be interpreted as a pair of *distinct* events or as a *single* longer event. With this mechanism, leaves can be combined to form longer durations. The third restriction ensures that, for each parent, at least some of its children are distinct.

The tie construct has two important benefits. First, it enables a VLT to encode syncopated rhythms, e.g., the dotted-quarter note in Figure 5 could not have been encoded without two $\mathrm{A}\flat$:3 leaves. Second, from a real-time point of view, the tie makes semantic sense. The decision to silence the currently sounding pitch is made in the future, when it is known for certain that a distinct new pitch should be played.

4.2. THE CONGLOMERATIVE FEATURE SET

The CGL feature set transforms a variable-sized leaf sequence into three fixed-size histogram vectors. CGL is motivated by our belief that deeper aspects of a melody's structure will emerge when one views its contents from a more abstract level than its immediate surface. In particular, these histograms transform the string of pitches encoded in v's leaves into a set of local averages that capture more general tonal, intervallic, and directional trends. As different segments may contain different number of events, it is convenient to define a histogram's *sample size* – the total number of counts distributed amongst all of its bins – as variable sz.

CGL is constructed from the sequence of pitches implied by the leaves of v. When constructing this feature vector, rests are ignored. Thus, the contents of x^{C} only encodes a tree's pitch content (rests and durations only affect this context indirectly). Figure 6 details the determination of CGL for the segment displayed in Figure 5. Additional details in this figure are fleshed out in the next three subsections.

4.2.1. *PC Histogram*

x^{P} contains 12 dimensions, each bin recording how many leaves contain a particular pitch-class field. x^{P} only encodes the pitch classes' distribution, capturing essential aspects of a bar's tonality. Because this histogram completely ignores the pitches' ordering, it is a *0-order* Markovian abstraction. The histogram's sample size, sz^{P}, equals the total number of pitch-valued leaves in the tree.

4.2.2. *INT Histogram*

x^{I} contains 13 dimensions, each bin recording how many adjacent pitch-valued leaves are separated by a particular absolute interval. The jagged, dotted lines that link the leaves in Figure 6 identify the intervals within the segment. The unison

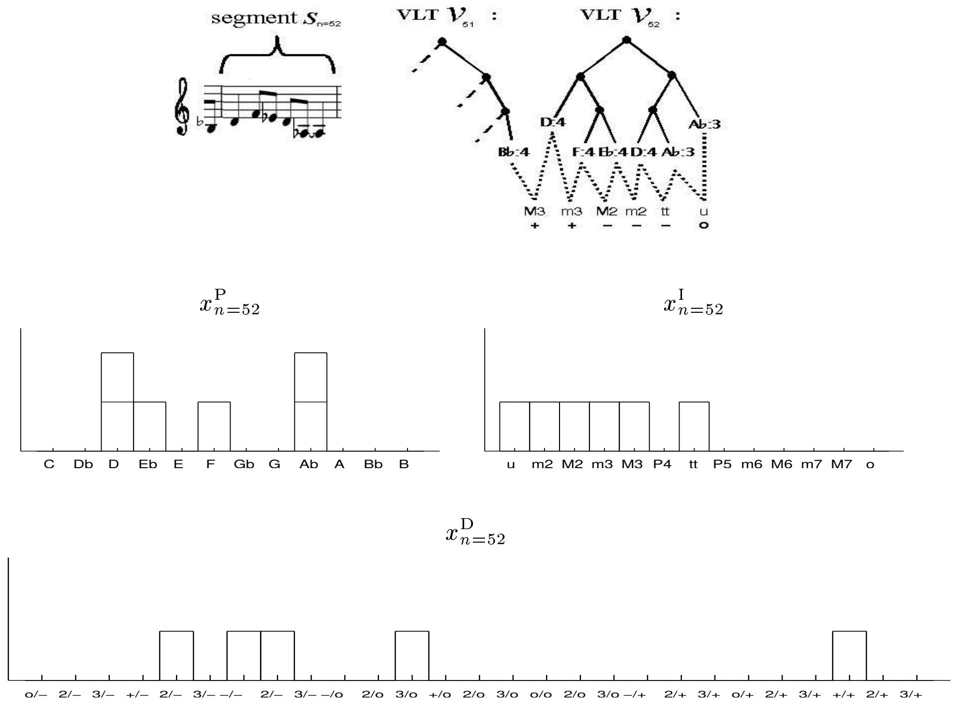


Figure 6. A Conglomerative Feature Set Example. The subscripts in this figure refer to specific bars in Charlie Parker's *Mohawk* transcription.

interval (bin 'u') corresponds to a zero semitone difference between adjacent pitches. The Minor-2nd interval (bin 'm2') corresponds to a difference of one semitone. Bin 'M2' corresponds to the Major-2nd interval, a two-semitone difference. The Major- and Minor-3rd, 6th, and 7th intervals are similarly labeled. The Perfect-4th and -5th, bins 'P4' and 'P5' respectively, surround the Tritone ('tt'), whose six-semitone difference divides the octave into two logarithmically equal parts. Bin 'o' is reserved for all intervals greater than or equal to an octave. Thus defined, intervals increase from left to right, each a semitone larger than their predecessors.

Intervallic differences are recorded in absolute terms (an interval's sign is only considered in the DIR histogram). x^{I} captures essential aspects of a sequence's melodic continuity. Because this histogram is based on pairs of pitches, it provides a *1st-order* Markovian abstraction. By linking x^{I}'s calculation into the previous bar via mrp, sz^{I} and sz^{P} are equal.

4.2.3. *DIR Histogram*

x^{D} is a 27-dimensional DIR histogram that records how the signs of the pairwise intervals progress over time. With respect to *directional motion,* three distinctions are made. For an *upwards motion*, $+$, the following pitch in an interval is higher.

In *downwards motion*, $-$, the following pitch is lower. With *zero motion*, o, both pitches are identical.

At each point in the leaf sequence, a *relevant directional history*, h, is recorded. h is defined by how many consecutive previous intervals, len, exhibit the same motion as some current interval, $from$. Each bin in x^{D} records how many times some history h is followed by some directional motion. x^{D} can be interpreted in terms of an underlying Markov chain that contains nine states:

$$h \in \{+, ++, +++, -, --, ---, \mathrm{o}, \mathrm{oo}, \mathrm{ooo}\}$$

and twenty-seven transitions, one for each state and directional motion. x^{D} records those transitions that are taken when the sequence of directions defined by a pitch sequence is parsed by this chain (Thom, 2001).

An important aspect of h is that it is context dependent. Whether it is based on three, four, or five adjacent leaves depends on the trend of the contour in its immediate past. By carefully defining context in this way, the abstraction covers a variable range of Markovian orders (from two to four) using a relatively small number of dimensions (27 versus 81). x^{D} captures essential aspects of the melody's contour by observing how pitch changes over time, keeping track of repeated directions of up to length three. x^{D} is also calculated using mrp, but this abstraction is constructed from pairs of intervals, and so $sz^{\mathrm{D}} = sz^{\mathrm{I}} - 1$.

The DIR bins in Figure 6 are abbreviated to make them readable. The '/' in each label separates relevant history h (left of slash) from its subsequent melodic direction (right of slash). For example, bin '+/-' indicates that a downwards motion follows a single upwards history (+). In contrast, the next two bins to the right, '2/-' and '3/-' respectively, correspond to downwards motions that follow an upwards history of length two (++) and three or more (+++).

4.3. Benefits of this Basic Representation

The VLT's leaf sequence encodes a melody as a string of pitches. While such views have been successfully used in other melodic settings (Mongeau and Sankoff, 1990; Pennycook and Stammen, 1993; Rolland and Ganascia, 1996; Hörnel and Menzel, 1998; Franklin, 2001), CGL is motivated by the desire to capture melodic structure at an even deeper level. The key benefit of this abstraction is that, while the histograms ignore a segment's exact pitch-by-pitch temporal behavior, their conglomeration still captures a significant portion of it.

Clearly, different notions of similarity will emerge when a melody is encoded as a string of pitches versus a CGL histogram. String-of-pitch-based comparisons naturally lead to an *edit-based* distance metric.[13] Although one could consider pairs of

[13] Edit-based measures are calculated using dynamic programming. Typically, the distance between two sequences reflects the number of insertions and deletions that are needed to transform one sequence into another.

CGL histograms within an edit-based framework, histograms naturally lend themselves to being modeled by a probability distribution. This distribution provides the operational basis for customizing our user model.

The value of encoding a melody using multiple views has long been appreciated, for each can be readily perceived by humans (Krumhansl, 1990; Bartlett, 1993). Cambouropoulos et al. have examined the pros and cons of various pitch-, intervallic-, and contour-based encodings. Sophisticated encodings that combine all three views have also been used in real systems (Hörnel and Menzel, 1998; Franklin, 2001), although such sophistication tends to make reasoning about a system's overall performance difficult. One of our goals in developing CGL was to provide a simpler, more integrated approach to combining these views together.

Another benefit of the VLT/CGL combination is that it provides a more flexible way of dealing with rhythm. Typically, a melodic model requires that each musical pitch must occupy a fixed duration (Hörnel et al., 1999; Franklin, 2001) which means that long durations can only be encoded by tying together long sequences of identical pitches. With the VLT/CGL combination, a rhythm's hierarchical nature is explicitly handled. This hierarchy also promotes a natural and intuitive method for modifying rhythms, i.e., embellish or simplify rhythmic content (Marsden, 1999) by collapsing or expanding various parts of a tree's internal structure. BoB transforms a call's rhythmic content in exactly this way.

5. Learning Musician-Specific Perception

We now describe how BoB learns to model the musician's playing modes using unlabeled CGL data. Our probabilistic learning method assumes that the data was generated by a *mixture model of variable-sized Multinomials (*vMn*)*. Learning amounts to estimating a set of parameters, Ω^{C}, such that X^{C} look maximally likely to have been generated by independently drawing N samples from $\mathrm{vMn}(\Omega^{\mathrm{C}})$. In the rest of this section, we briefly introduce the vMn model and explore the pros and cons of applying it to melodic improvisation. For a more detailed mathematical treatment, see (Thom, 2000b; Thom, 2001). First, however, we clarify exactly how the conglomerative assumption affects the model's implementation.

Recall that CGL assumes that PC, INT, and DIR counts are independent of one another. Under this assumption, BoB is free to model each view separately, through which Ω^{C} is fully determined. Similarly, mapping $\mathcal{M} : x^{\mathrm{C}} \longrightarrow y^{\mathrm{C}}$ estimates a segment's playing mode by estimating each view's maximal posterior probability in isolation, e.g.,

$$y^{\mathrm{P}} = \mathrm{Argmax}_{1 \leq k \leq K^{\mathrm{P}}} \left(\Pr\Big(y^{\mathrm{P}} = k \mid x^{\mathrm{P}}; \mathrm{vMn}(\Omega^{\mathrm{P}}) \Big) \right).$$

Together, these classifications fully define the mapping:

$$y^{\mathrm{C}} = \langle y^{\mathrm{P}}, y^{\mathrm{I}}, y^{\mathrm{D}} \rangle.$$

Independence also results in a surprise value that is multiplicative:

$$l^{\mathrm{C}} = l^{\mathrm{P}} \cdot l^{\mathrm{P}} \cdot l^{\mathrm{D}}.$$

Each term is again calculated separately, e.g.,[14]

$$l^{\mathrm{P}} = \Pr\Big(x^{\mathrm{P}} \mid y^{\mathrm{P}}; \mathrm{vMn}(\Omega^{\mathrm{P}})\Big).$$

With the tuple construction, K^{C} is also multiplicative, although in practice, many of the possible y^{C}-tuples that could arise are never seen when classifying data.

Figure 7 presents a concrete example of $\mathcal{M}$ for $K^{\mathrm{C}} = 3$. This figure highlights that both the playing mode and the counts in the CGL histogram are nominal and discrete. An aspect of this mapping not explicit in figure 7 is that x^{C} need not contain a constant number of counts – sz^{C} can vary. This flexibility is important when modeling melody because, even in our simple per-bar segmentation scheme, different bars will contain different numbers of counts.[15]

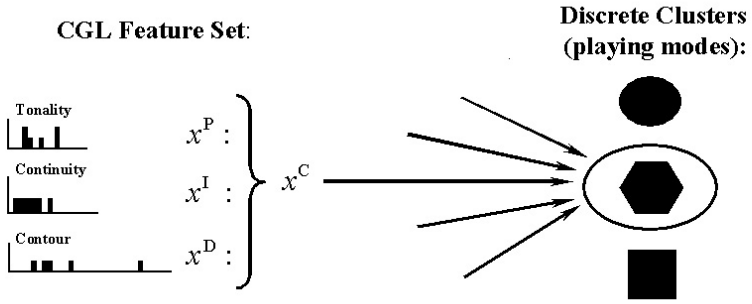


Figure 7. User-Specific Playing Mode Classification.

5.1. Probabilistic Clustering via a Mixture Model

There are two reasons why a probabilistic model for clustering was chosen over a more ad-hoc approach. First, a probabilistic model forces one to explicitly identify what assumptions are being made, and this makes it easier to identify the underlying strengths and weaknesses of the method. Second, a probabilistic model is generative, which means that a learned model can be used to simulate new histograms. This ability is especially valuable when training data is limited because it means that a particular model's stability can be explored through simulation.

[14] Rather than defining surprise in terms of unlabeled likelihood, this definition is based on the assumption that the classification estimate is correct. This quantity relates to how probabilistically far away a histogram is from a component's mean.

[15] How to automatically segment a melody into its ideal 'salient chunks' is an open research question (Cambouropoulos, 2001; Bod, 2001). Our model's flexibility means that the future incorporation of such techniques will be straightforward.

Mixture models assume that an arbitrary data point is independently sampled from 1-of-K components. The probability that Component k is chosen is solely defined by mixture weight π_k. The crucial assumption is that one and only one playing mode generates a given histogram. The use of a mixture model carries enormous import, for only the most mundane musician would perform notes according to a fixed distribution throughout his or her entire solo. With vMn, each segment can be derived from one-of-K different distributions.

To infer a mixture model is difficult because important information is hidden:

– the number of components;
– the mixture weights; and
– each component's individual parameters.

This information is only difficult to infer because we do not know which components generated which training histograms. Rather, component membership must be inferred along with the parameters themselves, and each depends on the other.

The Expectation-Maximization (EM) algorithm (Dempster et al., 1977) is applied to iteratively search for an optimal parameterization. When this search completes, EM returns that Ω that maximizes the *marginal likelihood* of the training data:

$$\Omega = \mathrm{Argmax}_{\Omega'}\left(\Pi_{n=1}^{N}\Sigma_{k=1}^{K}\mathrm{Pr}\Big(x_n \mid \mathrm{vMn}(\Omega')\ ;\ \mathrm{y_n = k}\Big)\right). \tag{1}$$

The product in Equation 1 is a result of the assumption that each histogram is independently generated. The summation, which produces a nonlinear system of equations, results from marginalizing over all possible component generators. In optimizing Equation 1, EM essentially performs gradient ascent, and so any particular solution is only guaranteed to be locally maximal. Thus, BoB reruns EM multiple times, each time starting at a new, randomly chosen set of parameter values.

Generally speaking, for a fixed value of K, Equation 1 will drive different components to be inferred so as to 'better explain' certain 'similar' data points, where these 'vague' terms are precisely specified by the mathematical behavior of the mixture model and the component Multinomial distributions. Unfortunately, with unlabeled data, we do not know how many clusters 'best' capture the musician's playing modes. Estimating an ideal number of components for an arbitrary data set remains largely unsolved (McLachlan and Basford, 1988). We use the following heuristic: pick the largest value for which a well-separated clustering of the training set can be repeatedly found. Repeatability ensures that the structure inferred from the data is genuine, and component separation ensures that the agent can recognize significantly different PC, INT, and DIR trends. This latter requirement has a subtle but important motivation. Although increasing K does improve a model's marginalized data fit, data posteriors will typically increase (components will typically overlap more), a result of the fact we are using Multinomial (versus Gaussian) Components – i.e., these components' variances are fully constrained by their means. For the data sets presented in Section 7, this heuristic has produced between 3 and 5 components per PC, INT, or DIR mixture.

5.2. Multinomials and Variable Sample Sizes

Each component is modeled as a *Multinomial (Mnom)* distribution, which treats each bin as nominal and discrete. As the Mnom's likelihood, sample statistics, etc., are readily available elsewhere, e.g., (Gelman et al., 1996), they are not described here. Suffice it to say that for a given component, those histograms that appear most likely will have their counts distributed on more probable bins, and this will drive histograms to be partitioned according to which of their bins are most and least heavily populated.

Our model extends Mnom so that the number of counts is treated as a random variable. In particular, before sampling a histogram from some generative component, the number of counts to be obtained is first sampled from an independent distribution.[16] This extension assumes that the number of pitch-valued leaves in a VLT is not correlated with the underlying playing mode. Histograms that have more counts will have more representation in the training set, however, and thus will have more impact on a distribution's parametric inference. For melodic improvisation, this behavior makes sense – e.g., a quick but longer run of notes probably does provide more information about what underlying scale a musician is drawing upon for inspiration.

5.3. Sparsity Concerns

With BoB, a major impediment to learning is that training data will be relatively *sparse*. One aspect of sparsity arises because we want to train on the musician 'right now,' where a typical warmup might consist of ten runs through a 12-bar blues, producing a mere 120 histograms for training.

Another aspect of sparsity arises because, given the number of counts that a typical bar might produce, the dimensions of PC, INT, and DIR are relatively large. Such a fine segmentation (per bar) will facilitate the correctness of the assumption that one and only one playing mode is operative within a given histogram, but it also means that there will be less data per histogram available for inference. In a perfect world, one would learn a clustering from sequences whose estimated segmentations perfectly matched their underlying generative boundaries. However, given the vMn model requires pre-segmented data and an ideal segmentation is unknown, the best strategy seems to be to prefer learning from solos that are too finely segmented than from solos whose segmentations are too large to allow the learner to adequately tease out individual playing-mode distinctions. Ideally, one would like to integrate the inference of segmentations and clusters into a single algorithm; we hope to address this issue in future work.

[16] Since sz is a fully observable quantity, this distribution's inference is straightforward.

To investigate the impact of sparsity on learning, experiments were run on simulated training data in order to verify that the overall procedure could learn useful structure (Thom, 2000b). We then learned vMn models for Parker's and Grappelli's CGL histogram data. Simulations were again used to verify that authentic structural information was inferred (Thom, 2001).

5.4. Minimizing Complexity

Mnom treats all histogram bins as discrete and nominal. As a result, counts in different bins are modeled as uncorrelated. A major benefit of this scheme is that it does not impose a predefined distance metric upon pitch-class, intervallic, or directional trends. In more concrete terms, the data itself will determine when, for example, pitch classes C and G should be treated as more similar to one another than to some other pitch classes. Specifically, only when a component has the same probability inferred for both bins will counts in each be treated identically.

Nonetheless, the underlying Mnom assumption is clearly inaccurate. For example, whether or not the musician plays a B♭ now will likely depend on whether or not he or she has just played certain other tones. The conglomerative and independently-sampled training data assumptions are suspect for similar reasons. The disadvantage of modeling correlated counts in an independent fashion is that the data one trains on will appear to contain more information than it really does. The advantage, however, is that there is less chance of over-fitting the model to the data because it is less complex. Ultimately, we settled on this simpler modeling approach because we felt that a less accurate model would provide a more intimate interaction with the user than a more complex model that was trained on a larger hodgepodge of the musician's behavior would.

6. Generating Musician-Specific Response

In this section, we present an algorithm for closing the learning/generation loop. At a high-level, this amounts to specifying the internals of the Generation Component in Figure 4. With the VLT hierarchy, reasonable options are available for transforming a call's rhythm into something new. We focus here on what to do with the new rhythm's durations. A method for assigning a pitch sequence that realizes a specific CGL trend must be developed.

To have such a capability requires an inversion of the arrows in Figure 7, except that now, the end result is not a CGL histogram but rather a pitch sequence that realizes its counts. Pitch-sequence generation becomes even more complicated when one introduces additional constraints, e.g., begin or end on a specific pitch, change melodic direction at a fixed point within the VLT, etc. Fortunately, the technology we develop for integrating CGL trends is flexible enough to accommodate a fairly general set of constraints, the result being a very powerful pitch sequence generator. For a more detailed treatment of all aspects of the solo-generation process, see (Thom, 2001).

6.1. PER-BAR RESPONSE GENERATOR

For each bar of response, the generator performs the following steps:

1. Transform v into v' by stochastically collapsing and/or growing its leaves.
2. Calculate sz', the number of pitch-valued leaves in v' (rests are excluded).
3. Determine ${\theta'}^{\mathrm{C}} = \left\langle {\theta'}^{\mathrm{P}}, {\theta'}^{\mathrm{I}}, {\theta'}^{\mathrm{D}} \right\rangle$ from y^{C} and Ω^{C}.
4. Initialize p' to some terrible solution.
5. Incrementing by 1, loop from $i = 1$ to num and do the following:
   (a) Generate pitch sequence tmp' by taking a random walk of sz' steps through the conglomerative Markov chain that was constructed for ${\theta'}^{\mathrm{C}}$.
   (b) Update p' as the best tmp' seen so far.
6. Assign p' in order to the pitch-valued leaves in v'.
7. Return v'.

In the first step, call rhythm v is transformed into response rhythm v'. The remaining steps attempt to fill in the pitch-valued leaves of v' so that the *generative goal* – to maintain playing mode y^{C} with a level of surprise that is similar to l^{C} – is met. Knowledge about y^{C} is contained in the user model, and Step 3 merely retrieves the parameters for these components. In Step 5a, ${\theta'}^{\mathrm{C}}$ is used to guide the generation of pitch sequence tmp'. The loop is necessary because the Markov chain is randomly sampled; we can never be sure that a given walk will produce the desired result. Steps 4 and 5b are included so that the algorithm can keep track of the best solution. In the event that more than one or more walks accomplishes y^{C}, the solution that produces the smallest deviation from l^{C} is chosen. Hopefully, after num tries, at least one of the walks will exhibit the pitch-sequence trends required for mapping onto y^{C}, in which case, the algorithm is considered *successful*. Success is then further quantified by its $rate$, which normalizes the number of walks that map into y^{C} by the number of attempts, num.

6.2. CONGLOMERATIVE MARKOV CHAIN PITCH-SEQUENCE GENERATION

How does one generate a pitch sequence that exhibits a specific set of CGL trends? To sample from $\mathrm{vMn}(\Omega^{\mathrm{C}})$ is not sufficient, for a new CGL histogram tells us nothing about *how* to construct a sequence that displays the corresponding counts. Furthermore, not all sampled values from this distribution necessarily produce viable solutions. Consider, for example, Figure 8. There is no way to choose and order two pitch values so that one of them uses pitch-class D♭, the other uses E♭, and the interval between them is a Tritone or Perfect 4th.[17] Unfortunately, when x^{C} is not

[17] The left-over interval would connect to the response's MRP.

derived from a pitch sequence in the first place, its content is unhelpful; another method is needed.

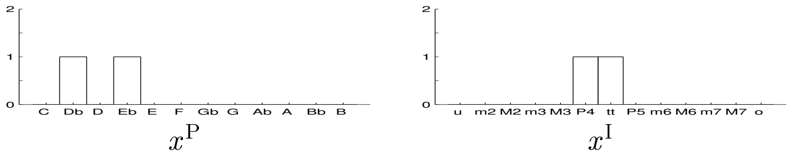


Figure 8. An Example that is Not a Viable Pitch Sequence.

The difficulty is that PC, INT, and DIR trends are all related in complex ways. In addition, trends may interfere with one another. For instance, one could ask for a goal that preferred three evenly-spaced pitch classes within the octave, yet discouraged less-continuous intervals. Granted, such a request may seem rather contrived, but it is easy to come up with musically-plausible scenarios where, in the limit (as $sz \rightarrow \infty$), the average behaviors of PC, INT, and DIR cannot be simultaneously maintained. Fortunately, because we are generating relatively short, sparse sequences, in practice, solutions can usually be found.

In order to generate the desired pitch sequence, we construct a *conglomerative Markov chain (CMC)* whose states are comprised of two parts, absolute pitch value a and relevant history h. States are notated as $z \in \langle a, h \rangle$, and these terms will be used interchangeably. The CMC is defined over a range of semitone-based pitches, a_{min} through a_{max}, producing a total of $n_a = a_{max} - a_{min} + 1$ different values. With the Markov assumption, the selection of the next pitch/history pair only depends on the current pair:

$$\forall t, \ \Pr(z_t \mid z_{t-1}) = \Pr(z_t \mid z_1, z_2, \cdots, z_{t-1}).$$

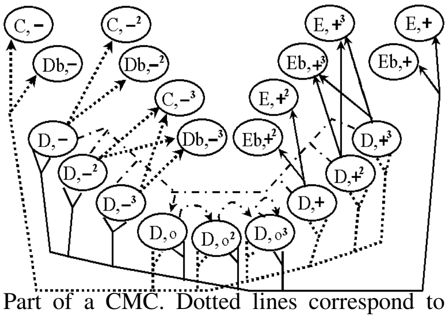
To fully specify the chain, this set of transition probabilities, $\Pr(z_t \mid z_{t-1})$, must be estimated. With nine relevant histories to consider, there are a total of $9 \cdot n_a$ states. The CMC is constructed so as to impose the same directional structure that was discussed in Section 4.2.3, except that there are now n_a transitions per-state (as opposed to just three).

The following heuristic is used to integrate the learned parameters, θ^{C}, into the estimation of the chain's transition probabilities:

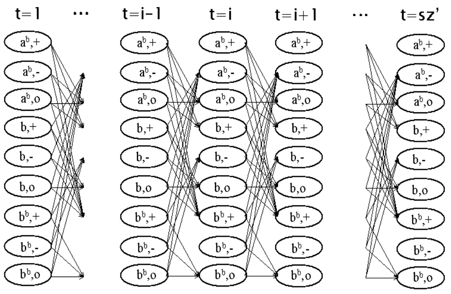
$$\Pr(z'|z) \propto \Pr(\texttt{pc}(a')) \cdot \Pr(\texttt{abs}(a' - a)) \cdot \Pr(h \rightarrow \texttt{sign}(a' - a)). \tag{2}$$

From θ^{P}, function `pc` retrieves the probability of the pitch class associated with pitch a'. Similarly, `abs` indexes θ^{I}, returning the probability of the absolute interval between the pitches a' and a. `sign` returns this interval's corresponding direction, which is used in conjunction with h to index into θ^{D}. The '$\propto$' indicates that each state's output transitions need to be normalized. Although the CGL assumption is used to justify the three multiplications in Equation 2, Figure 8 clearly demonstrated why this assumption is inaccurate. Fortunately, this heuristic seems to provide the search with adequate guidance in practice. Often, as little as 2 or 3 loop executions (Steps 5 in Section 6.1) produce the desired mapping. As the desired goal becomes more difficult to realize, a value of $num = 25$ usually suffices.

Figure 9 provides some concrete CMC examples. In the left plot, an example is constructed over the range $a \in \{\mathrm{C}, \mathrm{D}\flat, \mathrm{D}, \mathrm{E}\flat, \mathrm{E}\}$. In total, there are 45 states $(n_a = 5)$. Many of these states are not shown because they would clutter up the plot. Instead, only those states and arcs that are needed to specify the transitions out of states $z \in \langle \mathrm{D}, h\rangle$ are displayed. This plot only represents part of a single step, demonstrating the structure between h and a. In the right plot, an entire walk is displayed. In this case, the structure of a single step replicates sz' times, and the desired pitch sequence is obtained by returning the sequence of a values associated with a random walk through this graph.



Part of a CMC. Dotted lines correspond to downwards motions, solid lines to upwards motions, and dot-dashed lines to unison intervals. The notation of h is simplified, e.g., '$+^2$' corresponds to history $++$.



A sz' Step Walk. For simplicity, only pitches A$\flat$, B, and B$\flat$, as well as histories +, -, and o are shown.

Figure 9. CMC Pitch-Sequence Markov Chain Generation Examples.

6.3. Clamping and Other Extensions

Powerful extensions are possible for imposing additional structure on a random walk. For example, we could *clamp* certain states, specifying which of a particular subset of states may or may not be occupied at some time t. All that is required is for these future constraints to be propagated backwards through the graph (Thom, 2001).[18] Clamping is a powerful option, useful for incorporating domain knowledge, authoring user desires, and integrating smoothly into the evolving musical conversation. In BoB, there are two reasons why such extensions are straightforward. First, results from the statistics community can be used in order to extend BoB's model. Second, the sensible PC, INT, and DIR features have natural musical interpretations.

[18] The most expensive scenario would arise when the last column is clamped. In this case, propagation requires $9 \cdot n_a^2 \cdot sz'$ calculations.

Some details concerning the integration of constraints into a CMC were glossed over in the previous section. We briefly mention some of them in passing. First, the interval at time $t = 1$ always links back into a response's MRP. In addition, unison intervals are disallowed in certain walk locations to ensure that, when p' is assigned to v', the tree maintains its VLT properties. Finally, different PC parameterizations can also be used when sampling from states that are on versus off the metrical beat. These latter extensions are handled by calculating several different transition matrices and switching from one to another as a function of t.

7. Musician Training Data

BoB has been separately trained on both Parker's and Grappelli's improvisations. Various aspects of these two data sets are detailed in Table I. The interpretation of this table is best explained by example. The two $X(\mathcal{P})$-based rows pertain to Parker's training set, which was obtained by parsing the *Mohawk* transcriptions of Goldsen (1978). This tune's harmony is based on a 12-bar B♭ blues progression. Two transcriptions, *Mohawk I* and *Mohawk II*, each containing five runs through the 12-bar blues chorus, were concatenated to yield 120 bars of improvisation. The histograms for Bar 8 of Chorus 5 did not contain any counts because Parker did not play any notes in that bar, so only the remaining 119 histograms were used. The sz statistics for these histograms provide an indication of how sparse their bins are.

Table I. Improvised Training Sets.

Train Set	Artist	Transcription	Source	Harmony	Key
$X(\mathcal{P})$	Parker	*Mohawk I, Mohawk II*	(Goldsen, 1978)	12-bar blues	B♭
$X(\mathcal{G})$	Grappelli	*I've Found a New Baby I*	(Glaser and Grappelli, 1981)	32-bar AABA form	F

Train Set	Bars per Chorus	Num Choruses	N	Histogram Sample Size Min	Max	Ave
$X(\mathcal{P})$	12	10	119	1	17	7.3
$X(\mathcal{G})$	32	4	126	1	12	6.8

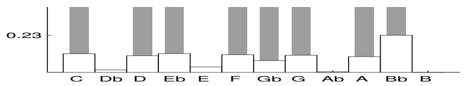
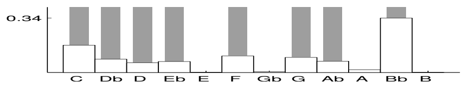
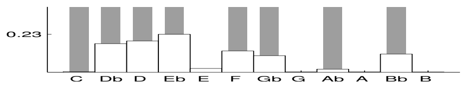
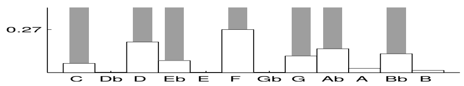
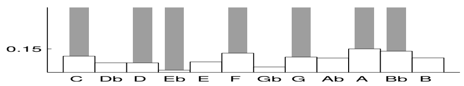
8. Evaluating User-Specific Perception

To demonstrate the quality and power of the abstractions that emerge when CGL histograms are clustered according to the vMn model, we now analyze what was learned for Parker's training set. Although a complete discussion concerning the harmonies, scales, and melodic styles of the Bebop genre are beyond this article's scope, some of these concepts are briefly mentioned because they lend musical credibility to the learned results. Such a domain-specific qualitative assessment is especially important because, with unsupervised training data, there is no other way to shed light upon the validity of what was learned. For a more detailed analysis of the $X(\mathcal{P})$ model and for an analysis of $X(\mathcal{G})$, see (Thom, 2001).

8.1. Tonal Learning

Table II presents the five pitch-class components that were learned for $X(\mathcal{P})^{\mathrm{P}}$. In the Size Column, two equivalent numbers are given, the number of bars in $X(\mathcal{P})^{\mathrm{P}}$ that were assigned to a component and the percentage of the training set that these bars occupy. The Musical Meaning Column summarizes the musical knowledge that is distributed within a component. BoB knows nothing about these musical scales; rather, it knows how probable each component's pitch classes are. It was the strong correlation between a component's more probable bins and a scale's pitch-class set (identified by the gray background columns) that led us to make these high-level assessments. Note that for Components ■, ■, and ▪, those assessments were immediately obvious; correlations are essentially one-to-one.

Table II. Learned Pitch-Class Components for Parker.

Sym	Pitch-Class Probabilities	Size	Musical Meaning
■		34/29%	B♭ Major Bebop Scale
■		40/34%	E♭ Bebop-7 Scale
□		10/8%	E♭ Min+7 Scale
▪		23/19%	E♭ Major Scale
▪		11/10%	B♭ Major Scale (with chromaticism)

The exciting fact is that musically-appropriate tonal abstractions are embedded within these parameters. For example, the Bebop Scales, which were formalized in part to explain the genre that Parker had a hand in inventing (Baker, 1983), account for 63% of the training set. Furthermore, given the B♭ blues progression, it makes perfect sense to improvise using B♭- and E♭-based Scales. These plots also poignantly demonstrate why this *distributed* representation, where each components' pitch classes are described by their own probability vectors, is more powerful than a symbolic, scale-based representation, especially when generation is the end goal. Whereas scales provide no information about *how* to realize a pitch sequence, BoB knows that, for example, when generating with ■, B♭ should occur about twice as often as any other probable pitch class.

Creating melody – an act as simple as improvised humming – is as much about one's ingenious use of exceptions as it is about adhering to certain rules (Johnson-Laird, 1991; Loy, 1991; Papadopoulos and Wiggins, 1999). Since the data set determines how probable a component's pitch classes are, BoB is free to construct a palette that accommodates both an artist's choice of scales and their subsequent, personalized modification. To illustrate this point, one can interpret Component □ as the following synthesis of musical ideas: improvise on the B♭ Major Scale, but prefer the Tritone (E) over the Scale's 4th (E♭) and add blues chromatic tones (the Minor-2nd, -3rd, and -7th) to the mix. While such concepts are roughly described in jazz texts (Russell, 1959; Coker, 1970; Ostransky, 1977; Baker, 1983; Aebersold, 1992), the idea of using a symbolic, scale-based framework to integrate such a wide variety of possibilities into a coherent model that automatically and intimately adapts to a particular user seems a daunting exercise at best, and at worst, a scenario whose very premise hinders the simulation of truly creative behavior.

In Figure 10, another view of what the tonal components have learned is presented. This figure displays $X(\mathcal{P})^{\mathrm{P}}$ as a temporal sequence of y^{P} abstractions. Each column of symbols occupies a similar (if not identical) chord in the underlying blues progression. This figure becomes impressive when one realizes that, in the blues, important harmony changes occur at Bars 1, 5, 7, 9 and 11. Even though BoB's learning algorithm knew nothing about the context, content, or structure of the harmony over which these improvisations transpired, these changes coincide with noticeable distinctions in BoB's perception of Parker's tonality:

– In Bar 1, Parker predominantly uses the □ Bebop Scale (70% of the time).
– In Bars 5 and 6, Parker almost exclusively uses the Bebop Scales (■ and □ occur 95% of the time).
– Bars 9 and 10 are the only place were Component ■ is used more than 50% of the time.
– Bar 12's pitch-class components distinguish between choruses that 'turn-back' (i.e., are repeated; choruses 1-4 and 6-9) and those in which Parker terminates his improvisation (choruses 5 and 10). In particular, all turn-backs (except Chorus 2) use the E♭ Scale, whereas at the end of the tune, solos resolve to the tonic B♭ Scale.

What we find especially encouraging about these results is that such important harmonic and scalar 'coincidences' were learned solely by considering the content of Parker's per-bar CGL histograms. Notably, these results emerged from within a model whose simplistic assumptions are clearly violated. At the same time, this simplicity is largely responsible for the model's ability to learn from a mere 119 histograms.

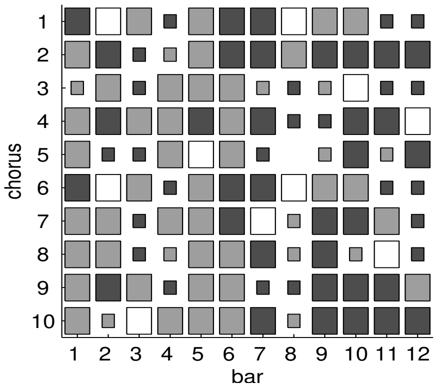
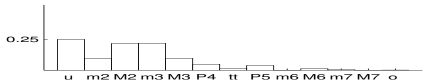
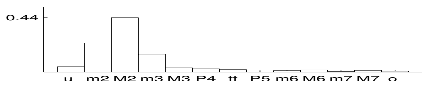
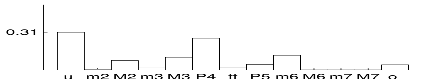
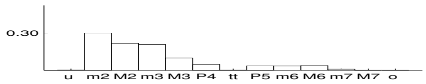


Figure 10. Training Data Pitch-Class Perception for Parker. The symbol in a given row and column identifies which Mnom component (as defined in Table II) was estimated as being most likely to have generated that bar's pitch-class histogram.

8.2. Melodic Continuity and Contour Learning

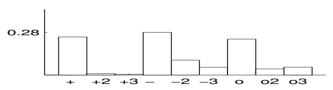
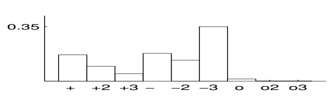
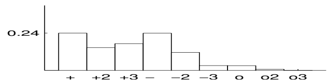
The four intervallic and three directional components that were learned for $X(\mathcal{P})^{\mathrm{I}}$ and $X(\mathcal{P})^{\mathrm{D}}$ are presented in Tables III and IV. Again, we assigned the meanings in the rightmost columns by interpreting the component probabilities from a musical perspective.

Table III. Learned Intervallic Components for Parker. Intervallic bins are labeled as described in Section 4.2.2.

Sym	Intervallic Probabilities	Size	Musical Meaning
A		51/43%	Expect relatively continuous, syncopated note sequences.
B		25/21%	Expect very continuous, non-syncopated note sequences.
C		9/8%	Expect syncopated, discontinuous note sequences.
D		34/29%	Expect relatively continuous, non-syncopated note sequences.

Our interpretations were based on two key insights. First, most of Parker's unison intervals (bin 'u') and the corresponding directional states (bins 'o' through 'o3') exist because of the way VLTs represent syncopated rhythms. Second, the interval bins, from left to right, proceed from smaller to larger melodic intervals,

Table IV. Learned Directional Components for Parker. Rather than showing all 27-dimensions of each DIR component, the stationary probability distribution associated with its underlying Markov chain is displayed. A chain's stationary distribution summarizes its asymptotic behavior. The probability associated with each bin reflects how often a particular state h is likely to occur within an infinitely long sequence of directions. Thus, for example, the '-' bin in Component 'a' reflects that approximately 0.28 isolated downwards motions will be seen in a very long pitch sequence. Similarly, '-2' refers to isolated pairs of downwards motions, and '-3' refers to three or more consecutive downwards motions.

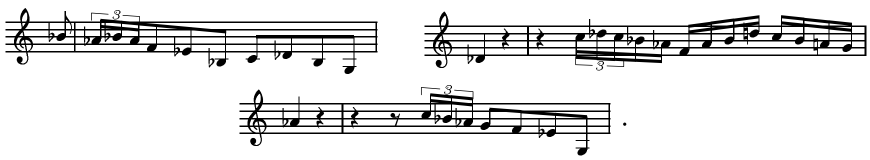
Sym	Directional Probabilities	Size	Musical Meaning
a		45/38%	Expect syncopation, interspersed by a single upwards or perhaps longer downwards motion.
b		33/28%	Expect downward motions, especially in runs of 3 or more, interspersed by shorter upward runs.
c		41/35%	Expect upwards motions, interspersed by shorter downward runs.

resulting in melodies with more (or less) continuity. Again, BoB only understands these concepts in a very low-level, distributed fashion, but these reflect musically powerful and appropriate notions, further demonstrating the usefulness of this learning approach.

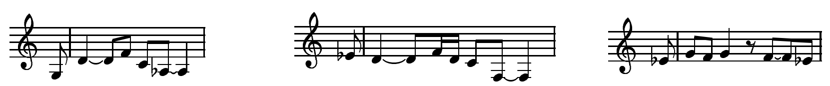
8.3. Some Conglomerative Perception Examples

To demonstrate the power and flexibility of BoB's conglomerative perception technique, we provide some examples of how training bars map into user playing modes. Although for each example, a bar's score and mrp are notated, the MRP only affects INT and DIR classifications.

Consider of the following examples, each mapping into playing mode ⟨■,B,b⟩:



Similarly, the following map into ⟨■,A,a⟩:



and these map into ⟨■,D,c⟩:


.

As one would expect, the component interpretations presented earlier coincide with the melodic trends shared among bars in the same mode. For instance, in ⟨■,B,b⟩, except for the A (middle example), all pitches coincide with tones in the E♭ Bebop-7 Scale. Melodies are also quite continuous, each example containing mainly Minor-2nd, Major-2nd, and Minor-3rd intervals. Finally, these three examples mostly consist of descending runs, interspersed by an occasional upwards motion or two, and no syncopation is present. To motivate such analysis, however, is not the main reason these examples were presented. Rather, we hope their availability will prompt the interested reader to *listen* to them.[19] The point is that, while in some ways the differences between per-row examples are significant, especially if one were to adopt an edit-based notion of similarity, in other ways they just seem to naturally 'fit together.' When played for audiences of both musically trained and untrained listeners, the elusive inter-cluster similarity and intra-cluster differences exhibited by these examples were appreciated.

These examples also illustrate the important role that histogram sparsity plays in the CGL classifier's generalization mechanism. In terms of Figure 7, the key point is that histograms with 'fairly different' looking shapes will map onto the same playing mode; the CGL examples presented here poignantly demonstrate this fact. Part of what enables these examples to look quite different, yet still belong to the same cluster, is that with such small sample sizes, only a few of a component's more probable bins are likely to be occupied at a given time. The components that BoB learned over the whole training set, however, promote many possible per-bar histogram instantiations. This observation naturally leads us to consider an exciting idea. Perhaps sparsity can be used to simulate musical creativity! In concrete terms, such an idea could be explored by investigating to what degree Parker's creativity can be 'explained' by considering which subsets of probable bins were occupied in specific musical contexts.

9. Evaluating User- and Context-Specific Generation

The performance of BoB's generation scheme is evaluated in three different ways. First, we investigate the degree to which the CGL heuristic helps to 'close the learning loop' by contrasting the performance of the transition probabilities specified in Equation 2 with those that contain no information. Next, the generation procedure's functionality and its generalization power and flexibility are demonstrated through

[19] MIDI files will be available at `http://www.cs.cmu.edu/~bthom` in the near future.

Table V. Simulated Generation Scenarios.

	Parker						Grappelli				
	M	C	Succ	Fail	$rate\%$		M	C	Succ	Fail	$rate\%$
I			92	28	3.7%	V			66	63	5.2%
II		√	98	22	4.3%	VI		√	59	70	7.5%
III	√		120	0	40%	VII	√		122	7	45%
IV	√	√	120	0	47%	VIII	√	√	125	4	56%

some generative examples. Finally, we describe the impact that some solo-trading demos have had on live audiences.

9.1. Closing the Learning Loop

Using the call-and-response paradigm, BoB generated the eight experimental data sets listed in Table V. Experiments that were based on both the $X(\mathcal{P})$ and $X(\mathcal{G})$ datasets were run. This data was used to form a single very-long call, which the generator received a bar at a time. Experiments were further distinguished in two ways. A '$\surd$' in the M Column indicates that the model that was learned for the call data was also used to guide the search for each bar of response (i.e., the learning loop was closed). Otherwise, the CMC transition probabilities were uniformly weighted (i.e., contained no learned knowledge). Second, a '$\surd$' in the C Column indicates that the last pitch-value in each bar of response was clamped to be identical to the last pitch-value of each call, which ensures that both data sets contained the same sequence of MRPs; otherwise, only the very first pitch in the call and response data sets were necessarily identical. The Succ, Fail, and $rate\%$ Columns quantify how well the generator performed in each configuration, where the total number of successes and failures equals the number of bars in the call, and $rate$ is normalized by $num = 150$ tries per bar.

The results indicate that, without user customization (learning), many more bars failed, the worst case being experiment VI. In contrast, with customization, only a few of Grappelli's call bars could not be successfully inverted. Performing this inversion with the clamping constraints also tended to improve performance, eliminating 6 of the failures that occurred in experiment I and 3 in experiment VII. Since clamping more tightly couples the generator's search to the viable pitch sequence that the musician improvised in the first place, this behavior makes sense. This clamping scenario is also musically reasonable – it amounts to asking the agent to 'fill in the blanks' in the middle of a given melody while keeping the endpoints fixed.

Additional simulations are described in (Thom, 2001). For instance, therein, the data sets for experiment III, IV, VII, and VIII were used to relearn the vMn model. By comparing the learned model that drove the generation process with the model that was learned from this simulated data, a more convincing demonstration that the heuristic in Equation 2 effectively 'closed the learning loop' was obtained.

Other experiments we also conducted to verify that human listeners were able to distinguish between different playing modes and user models. These experiments relied on more complicated call-and-response scenarios (similar to what will be discussed in relation to Figure 11) and as such, demonstrated that, even when trends and conflicts were in competition, solutions could usually be found.

9.2. Some Generative Examples

Figure 11 presents several generation examples. Each bar of response was generated with its last pitch clamped to C:4. The MRPs that are notated were obtained from the call data and only impact the INT and DIR mappings. Solid arrows correspond to generation scenarios in which no stochastic rhythm transformation was allowed. The upper dashed arrow provides an example in which the generator was allowed to embellish the VLT's leaves, whereas the lower dashed arrow provides an example in which only rhythmic simplification was possible.

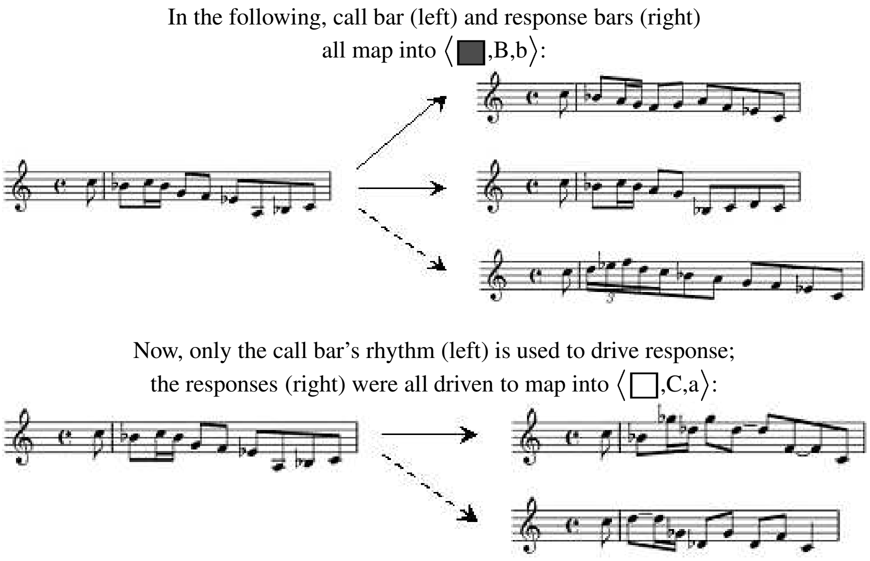


Figure 11. Generative Examples.

The two solid-line examples in the upper portion of Figure 11 demonstrate the generator's stochastic nature; in these cases, identical inputs produced different outputs. The playing modes associated with the upper and lower portions of this figure are quite different. ⟨■,B,b⟩ prefers all pitch classes *except* D♭, E, A♭, and B whereas ⟨□,C,a⟩ prefers pitch class D♭, as well as D, E♭, F, G♭, and B♭. For this reason, the D♭ tones produced with Goal ⟨□,C,a⟩ and the A tones produced with Goal =⟨■,B,b⟩ are noticeably present and distinctive. It is also interesting that, even though Goal ⟨□,C,a⟩'s tonality did not prefer pitch class C, it was able to achieve its goal even though its last tone was clamped to this value. This ability to successfully integrate goals and constraints has been witnessed in other experiments as well. In terms of intervals, $\langle\blacksquare$,B,b$\rangle$ prefers small intervals, especially the Major-2nd, but not the unison, whereas $\langle\square$,C,a$\rangle$ prefers the unison, Perfect-4th, and Minor-6th intervals. Whereas $\langle\blacksquare$,B,b$\rangle$ prefers runs of downwards motions, interspersed by an upwards interval or two, $\langle\square$,C,a$\rangle$ prefers single upwards and downwards steps, interspersed by syncopation, so that no obvious ascending or descending preference is evident. For these reasons, $\langle\square$,C,a$\rangle$'s responses are syncopated and jump around and $\langle\blacksquare$,B,b$\rangle$'s are continuous and lack syncopation.

9.3. Solo Trading Demos: Parker and Grappelli

The ultimate generation test is a live demo. On several occasions, we simulated and played back various trading-four exchanges for live audiences. These exchanges were configured by two parameters, which data set W was used for obtaining the musician's calls and which data set Z was used for training the user model. The four configurations that were simulated were obtained by assigning all possible combinations of values to W and Z from the set $\{X(\mathcal{P}), X(\mathcal{G})\}$. For a given configuration, solo-trading is simulated as follows:

1. Retrieve the learned model for data set Z.
2. Retrieve one or more choruses from data set W, and use the corresponding solo to simulate the musician's calls. This solo is broken into four-bar chunks, and every other chunk is changed into rests.
3. An independent module, which plays the musician's call VLT in real time, also feeds it into BoB's perception module.
4. Using the abstract cut-and-paste scenario, BoB transforms each VLT call into a response via the per-bar generation algorithm. As each bar of response is generated, it is in turn scheduled for real-time playback.

With $num = 50$ trials, all bars generated for all exchanges were successful. With respect to the Parker/Parker- and Grappelli/Grappelli-based demos, audiences could appreciate the degree to which BoB transformed and responded to the user's calls. With respect to the Parker/Grappelli- and Grappelli/Parker-based demos, it was generally agreed that these responses did not seem to fit with their calls as well as in the former demos, although an appreciation of the agent's 'consistent' musical behavior was often commented upon. These results are promising and encourage us to proceed with the natural next step: to evaluate live, interactive performance.

10. Related and Future Research and Limitations

By definition, IMC domains are interdisciplinary, drawing on techniques from a wide variety of fields: real-time systems, artificial intelligence, human-computer interaction, statistics, music analysis, psychology and cognition, etc. In this section, we outline some related research in two fields, interactive computer music and machine learning. This discussion leads us to consider the limitations of our approach, which motivates directions for future research.

10.1. Related Interactive Music systems

Most interactive music systems rely on an *author-the-aesthetics* paradigm, meaning that the artist, programmer, and/or user is expected to configure the system's inner workings, hand-customizing it until interesting musical aesthetics result.

For example, in many of Dannenberg's interactive performances, the computer is programmed to be an extension of his compositions, capable of responding to the real-time performance decisions and nuances of a live musician. In his performances, music perception is either hand-coded and/or trained using human-supervised learning data, and music generation is generally designed to carry out the composer's goals. More open-ended improvisational experiences are sought in Wessel's computer-assisted performance settings, but the primary focus is again on human-authored aesthetics. Towards this end, much of his work has involved creating technologies for organizing and controlling the access to musical material so that human-computer musicians can author meaningful musical experiences on-the-fly (Wessel et al., 1998; Wessel and Wright, 2000). Situated somewhere between these two approaches is Lewis' interactive *Voyager* system (Casserly, 1997; Roads, 1985). Improvisationally, *Voyager* is quite open-ended, and Lewis points out that the emergent properties that arise "when playing with the thing" are of primary interest to him. This interest closely coincides with our agenda, although Lewis, a practicing improviser, has spent many years hand-crafting his interactive software environment, whereas we hope to operationalize much of this process using ML.

Recently, ML techniques have been applied to reduce the authoring burden in interactive music systems, facilitating a more autonomous capture of music-aesthetic concepts. A few of these attempts are described below.

Improvisation Builder (IB). This system was developed in order to break out of the author-the-aesthetics camp (Walker, 1997). IB can play jazz with a musician without relying on any sort of lead sheet. Conversation analysis techniques are used to infer what the agent should do at a given time, either to solo or to accompany. As more sophisticated interaction scenarios are developed, it will make sense to consider adding this capability to BoB.

GenJam. This system is a genetic-algorithm based model of a jazz musician learning to improvise (Biles, 1994). GenJam is a concrete example of how one might automatically convert listeners' musical sensibilities into operational procedures by incorporating their opinions into fitness functions that drive the generation of the next population of solos. We did not try a similar approach because it seemed unreasonable to expect IMC users to listen to and rate many examples. The original system was not interactive, but in later work, GenJam was enhanced so that a live musician could trade fours with it (Biles, 1998).

Dannenberg. Although Dannenberg's agenda is quite different from ours, some of his research has influenced BoB's development. The improvised melody 'style detector' (Dannenberg et al., 1997) is one good example. In this research, a musician improvised while guided by a specific series of underlying intentions (e.g., bluesy, frantic, pointillistic, etc.). The data was then segmented and converted into histogram-based features, which a classifier was trained to distinguish accordingly. A natural next step would be to use this technology to direct a computer's generation of melody. Further consideration of this idea poignantly demonstrated the difficulty of closing such an abstract learning loop, and this primed our interest in transcription-based melody generation. Another example is the scoreless accompanist (Dannenberg and Mont-Reynaud, 1987), which is similar in spirit to BoB. The purpose of this system was to accompany a soloist that was freely improvising on top of a 12-bar blues. Within such a setting, data was collected and used to build a model that could estimate the musician's current location within the blues by analyzing their recent pitch-class usage. For each eighth-note location within the blues chorus, a *single* PC-based Mnom was estimated. The authors noted that a less disperse likelihood function would probably improve performance, and a mixture model distribution could directly address this point.

CHIME. This agent learns to trade solos with a jazz musician in two different phases Franklin (2001). CHIME represents melody as a sequence of pitches and rests with sixteenth-note durations (slurs are needed to create longer notes). In phase one, a recurrent neural network learns how to reproduce three jazz melodies from saxophonist Sonny Rollins' transcriptions. In phase two, a real-valued reinforcement learning (RL) algorithm is used in conjunction with a human performer's solos to refine the network's behavior. The reward signal, which was constructed by hand, rates the agent's solos in terms of simple jazz harmonic criteria. As Franklin noted, it is difficult to interpret why CHIME's networks produce specific outputs; to reason about BoB's musical behavior seems more straightforward. CHIME also implicitly couples its generator to this 'external critic' via the reward mechanism. In Bob, special emphasis is placed on exploring how much the desired goal can be *explicitly* embedded into its search for a solo. In the future, we look forward to empirically comparing and contrasting BoB's and CHIME's interactive musical behaviors.

10.2. Related Machine Learning Algorithms

Multinomial-based clustering has not received as much attention as Gaussian-based clustering (McLachlan and Basford, 1988). When algorithms do cluster using Mnom distributions, they often impose special restrictions upon the number of counts that data points may contain. For example, in *Autoclass* (Hanson et al., 1991), discrete variables are either present or not; no more than one count per bin is allowed. During the initial stages of our research, the histogram clustering algorithms that we found in the literature imposed such restrictions, which led us to develop our own algorithm. The result was vMn, which extends the Mnom mixture model so that it can take advantage of the additional information that larger sample sizes provide about a histogram's underlying generative character. Recently, Mnom clustering has received more attention, motivated in part by the increasingly widespread use of the Internet. For instance, variable-size labeled and unlabeled histograms were combined into a Mnom mixture model for classifying on-line documents (Nigam et al., 1998). In contrast to vMn, this model normalized histograms because it was felt that longer documents should not be more heavily considered.

More recently, Mnom-based mixtures have been used to segregate sequences into their most likely Markov chain generators (Smyth, 1999; Cadez et al., 2000; Ramoni et al., 2000; Thom, 2000b). For music generation, such models are especially interesting because the chains that are learned can be used to generate new musical sequences. Unfortunately, Markov-chain-based music generation often suffers from a lack of coherence because, for example, the sampler can get ‘lost’ within its random walk. In other words, there is not enough context around in which to keep track of necessary higher-level structural aspects. Learning algorithms that can automatically extend context as needed provide one such mechanism for dealing with this issue, and such techniques have already been successfully applied to music generation (Lartillot et al., 2001). In BoB, this issue was addressed by introducing a heuristic for combining Markov chains of multiple orders into a unified sampling procedure. Notably, this technique facilitates learning even when little training data is available. Although still in a preliminary stage, to combine trends with different Markovian orders into an exact mathematical sampling procedure is also being investigated by Steinsaltz and Wessel.

A particularly relevant melody generation system was proposed in Hörnel (1998). This system was based on neural network technology and ‘closed its learning loop’ by learning two types of networks: those that could cluster melodies into motives and those that could produce new cluster-specific motives. By-hand, the number of motive clusters was chosen so as to improve the degree to which each motive-generator could produce new representative material. This system dealt dealt with the lack of coherence by learning sequences on multiple time scales. In particular, the system could predict a motive-based sequence, which was then used to guide the generation of actual note sequences. In the future, we plan to employ a similar methodology by learning to predict the musician’s abstract playing-mode sequences.[20] Such functionality would certainly allow BoB to interactively participate on a deeper level.

10.3. LIMITATIONS AND FUTURE RESEARCH

The most pressing limitation is BoB’s inability to play with live performers and is due to BoB’s reliance on transcribed input. This issue is so important that we have already begun to realize the plans outlined in Section 1.3.2.

Once live interaction is possible, we will begin exploring how to best author domain knowledge. One promising idea is to clamp pitches that lie on strong metric boundaries to chord tones. This mechanism, which incorporates both harmonic

[20] Perhaps by applying some variant of the algorithm used in Lartillot et al. (2001).

and metric knowledge, might significantly reduce the number of ‘bad notes’ that BoB generates. We are also eager to have as many musicians play with the system as possible. By carefully analyzing the corpus of interactions that result, a comprehensive list of serious errors can be compiled and used to incrementally improve BoB’s model, guiding the addition of new features, the modification of existing features, and the use of constraints. Toiviainen’s research might be very valuable in this stage, for he developed an algorithm that could predict key ‘target notes’ in a jazz solo, and these could perhaps be used to constrain BoB’s random walks in productive ways.

An obvious limitation is BoB’s mechanical, unexpressive playback. In addition to significantly bolstering a system’s believability, expression plays an important part in communicating emotional meaning (Bresin and Friberg, 2000). With the recent strides that have been made in automating musical expression, integrating such a capability into BoB is worth considering. Dannenberg and Derenyi have developed a technique that integrates synthesis with a low-level performance model. Their research focuses on standard trumpet-based performance practices, requiring less symbolic knowledge than some other methods. Since their performance model was tuned for classical music transcriptions and required minimal phrasing information, integrating this model into BoB would require some adaptation, but the endeavor looks promising. Another possibility is the *Saxex* model (Arcos et al., 1998), which uses case-based reasoning techniques to find synthesis parameters that are ‘appropriate’ for expressing a score. Given that Saxex was crafted for jazz ballads, an interesting research question is how well their approach could perform in more general improvised settings. Integration with BoB is straightforward in that Saxex operates upon note sequences in specific harmonic contexts, but more difficult issues remain. For example, Saxex does not run in real-time, and the current system operates upon ‘expressivizing’ an audio signal (versus MIDI). Fortunately, providing an interface that would allow the system to operate directly with MIDI data is part of Arcos et al.’s research agenda.

11. Concluding Remarks

This article introduced a novel domain for user modeling – building agents that provide a musician with personalized, improvisational companionship. An implementation of BoB, an agent that was designed to trade live, customized solos with an improvising musician, was also presented. BoB’s primary technical challenge – to automatically customize a real-time model of improvisation to its user – was examined in detail, and UL techniques were developed to configure this model. The performance of this customization was evaluated in two ways. First, simulations were used to quantify how well particular algorithms performed their assigned tasks. Second, the abstractions that were learned and the resulting solos that these abstractions could be used to generate were qualitatively examined from both a musician specific and a more general commonsense point of view. Evaluation took place in two real-time simulated contexts, Charlie Parker's and Stephane Grappelli's improvisations.

BoB's implementation introduced a novel melody representation scheme in which solos were segmented per bar, providing a highly localized melodic view. Segments were further decomposed into a set of views that explicitly captured salient aspects of their rhythmic-, tonal-, intervallic-, and contour-based content. The power of this representation lies in its hierarchical, musically-meaningful, multifaceted nature. In particular, VLTs are used to hierarchically encode rhythms; the VLT's leaves are used to construct PC, INT, and DIR histograms; these histograms are used to construct the CGL feature vector; and this vector is mapped onto a particular user-specific playing mode. While the more abstract views ensure that BoB can see 'beyond a solo's immediate melodic surface,' the highly-localized segments provide the flexibility needed to intimately adapt and respond to the musician. This representation's musically-meaningful handling of rhythm- and pitch-based content is also a big part of why BoB's solo-trading model is easy to understand and interpret. These qualities will make it easy to incorporate higher-level domain knowledge and user desires into the generation procedure in the future.

Customized abstract perception was achieved using unsupervised learning. A mixture of Mnom components was fit to the CGL feature set constructed from the user's warmup improvisation. This model mapped each bar of solo onto one of a discrete number of user-playing modes. These modes were inferred by teasing out some different ways in which the musician used pitch-class, intervallic, and directional tends when spontaneously creating pitch sequences. When trained on saxophonist Charlie Parker, impressive abstracts emerged. For example, even though the system knew nothing about the Bebop genre that Parker had a hand in developing, relevant Bebop Scales were distributed within several of the PC Mnom components.

A two-part algorithm was introduced for transforming a user's solo segment into a customized response. First, a stochastic method for transforming the call's rhythm was mentioned. Second, a pitch sequence was assigned to this rhythm's durations by 'closing the learning loop,' wherein a stochastic sampling method for inverting the pitch sequence to playing-mode abstraction was developed. The power and flexibility of this sampling method lies in its ability to integrate both trends that are based on different (perhaps conflicting) Markovian orders and a fairly general set of constraints. Quantitatively, the feasibility of such integration was demonstrated in realistic, simulated musical contexts. The generator's solo-trading performance was also qualitatively evaluated, further demonstrating the power of its user- and context-specific functionality.

Before closing, a few remarks about the call-and-response metaphor that was used to ground human-computer interaction are in order. It is easy to argue that call and response is, in essence, a toy scenario, and if taken literally, this is probably true. However, this scenario has only been used to *situate* IMCs, and from within this paradigm, much more interesting modes of interaction can be imagined. For example, BoB will ultimately need to move beyond abstract cut-and-paste, and this will require that, at an abstract, user-specific level, the agent be able to predict what should happen next. The model we have proposed is flexible enough to accommodate such learned knowledge. From an authoring perspective, however, the crucial requirement is that the model incorporate such abstraction into an operational procedure, i.e., the agent needs to be able to generate new note sequences that exhibit particular user-specific properties on demand. BoB presents one example of such a working system, and with this agent in hand, meaningful and interesting applications can follow.

Author's vita

Dr. Belinda Thom

Harvey Mudd College, Dept. of Computer Science, 301 E. 12th Street, Claremont, CA, USA 91711

Dr. Thom received her B.S. in Mechanical Engineering from the University of California at Berkeley and her M.S. and Ph.D. in Computer Science from Carnegie Mellon University. Throughout her life, she has played the violin in both classical and improvised settings. In July 2001, after having defended her doctoral research, she was awarded a year-long American Fulbright Fellowship in Germany, where she is collaborating with the Information Structures in Music group within the Institut für Logik, Komplexität und Deduktionssysteme at the Universität Karlsruhe. Her current research is focused on real-time rhythm quantization and melodic segmentation. In the Fall of 2002, she will begin her appointment as Assistant Professor of Computer Science at Harvey Mudd College, where she plans to enjoy teaching undergraduates, using every opportunity to collaborate with them on new music- and art-based AI research.

Acknowledgments

Research leading to the preparation of this article was supported by the German-American Fulbright Commission and the generous support of the Computer Science Dept. at Carnegie Mellon University. For their invaluable guidance, thanks to the author's thesis advisor, M. Veloso, and the rest of her thesis committee: R. Dannenberg, T. Mitchell, and D. Wessel. Sincere thanks also goes out to K. Höthker, M. Mateas, W. Menzel, M. Waldrop, R. Warner, and the three anonymous reviewers whose constructive comments helped to improve this paper.

References

Aebersold, J.: 1992, *How to Play Jazz and Improvise*. New Albany, NJ: Jamey Aebersold.

Arcos, J., R. L. de Mantaras, and X. Serra: 1998, 'Saxex: A Case-Based Reasoning system for generating expressive musical performances'. *Journal of New Music Research* **27**(3), 194–210.

Bailey, D.: 1992, *Improvisation, Its Nature & Practice in Music*. Cambridge, MA: Da Capo Press.

Baker, D.: 1983, *Jazz improvisation : A Comprehensive Method of Study For All Players*. Bloomington, IN: Frangipani Press.

Bartlett, J. C.: 1993, 'Tonal Structure of Melodies'. In: T. J. Tighe and W. J. Dowling (eds.): *Psychology & Music: the Understanding of Melody & Rhythm*. Manwah, NJ: Lawrence Erlbaum Associates, Inc.

Bates, J.: 1992, 'The Nature of Character in Interactive Worlds'. Technical Report CMU-CS-92-200, School of Computer Science, Carnegie Mellon University.

Berliner, P. F.: 1994, *Thinking in Jazz, The Infinite Art of Improvisation*. Chicago, IL: University of Chicago Press.

Biles, J.: 1994, 'GenJam: A genetic algorithm for generating jazz solos'. In: *Proceedings of the International Computer Music Conference (ICMC'94)*, Aarhus, Denmark, pp. 131–137.

Biles, J.: 1998, 'Interactive GenJam: Integrating Real-time Performance with a Genetic Algorithm'. In: *Proceedings of the International Computer Music Conference (ICMC'98)*, Ann Arbor, USA, pp. 232–235.

Bod, R.: 2001, 'Memory-based models of melodic analysis: challenging the gestalt principles'. *Journal of New Music Research* **30**(3), In Print.

Bresin, R. and A. Friberg: 2000, 'Emotional Coloring of Computer-Controlled Music Performances'. *Computer Music Journal* **24**(4), 44–63.

Cadez, I., S. Gaffney, and P. Smyth: 2000, 'A Generalized Probabilistic Framework for Clustering Individuals'. Technical Report UCI-ICS-00-09, University of California, Irvine, Information and Computer Science.

Cambouropoulos, E.: 2001, 'The Local Boundary Detection Model (LBDM) and its Application in the Study of Expressive Timing'. In: *Proceedings of the ICMC 2001*.

Cambouropoulos, E., T. Crawford, and C. S. Iliopoulos: 1999, 'Pattern Processing in Melodic Sequences: Challenges, Caveats & Prospects'. In: *Proceedings from the AISB'99 Symposium on Musical Creativity*. pp. 42–47.

Cassell, J. and T. Bickmore: In Print, 'Negotiated Collusion: Modeling Social Language and its Relationship Effects in Intelligent Agents'. *User Modeling and User-Adapted Interaction, Special Issue on User Modeling and Intelligent Agents*. In this issue.

Casserly, L.: 1997, 'Person to ... person?'. *Resonance Magazine* **6**(1).

Cemgil, A. and B. Kappen: 2001, 'Tempo Tracking and Rhythm Quantization by Sequential Monte Carlo'. In: *Advances in Neural Information Processing Systems 14*.

Cemgil, A. T., P. Desain, and B. Kappen: 2000, 'Rhythm Quantization for Transcription'. *Computer Music Journal* **24**(2), 60–76.

Coker, J.: 1970, *The Jazz Idiom*. Englewood Cliffs, NJ: Prentice-Hall Inc.

Cope, D.: 1992, 'Computer modeling of musical intelligence in EMI'. *Computer Music Journal* **16**(2), 69–83.

Dannenberg, R. B. and I. Derenyi: 1998, 'Combining Instrument and Performance Models for High-Quality Music Synthesis'. *Journal of New Music Research* **27**(3), 211–238.

Dannenberg, R. B. and B. Mont-Reynaud: 1987, 'Following an Improvisation in Real Time'. In: *Proceedings of the International Computer Music Conference (ICMC'87)*, Champagne/Urbana, USA, pp. 241–248.

Dannenberg, R. B., B. Thom, and D. Watson: 1997, 'A Machine Learning Approach to Musical Style Recognition'. In: *Proceedings of the International Computer Music Conference (ICMC'97)*, Thessaloniki, Greece, pp. 344–347.

Dempster, A. P., N. M. Laird, and D. B. Rubin: 1977, 'Maximum Likelihood from Incomplete Data via the EM Algorithm'. *Journal of the Royal Statistical Society B* **39**, 1–38.

Eliassi-Rad, T. and J. Shelvik: In Print, 'A System for Building Intelligent Agents that Learn to Retrieve and Extract Information'. *User Modeling and User-Adapted Interaction, Special Issue on User Modeling and Intelligent Agents*. In this issue.

Elliot, C. and J. Brzezinski: 1998, 'Autonomous Agents as Synthetic Characters'. *AI Magazine* pp. 13–30.

Franklin, J. A.: 2001, 'Multi-Phase Learning for Jazz Improvisation and Interaction'. In: *Proceedings of the Eighth Biennial Symposium for Arts & Technology*, Connecticut, USA.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin: 1996, *Bayesian Data Analysis*. New York, NY: Chapman & Hall.

Glaser, M. and S. Grappelli: 1981, *Jazz Violin*. New York, NY: Oak Publications.

Goldsen, M. H. (ed.): 1978, *Charlie Parker Omnibook: For C Instruments*. Atlantic Music Corp.

Gridley, M. C.: 1988, *Jazz Styles, History and Analysis*. Upper Saddle River, NJ: Prentice Hall, third edition.

Hanson, R., J. Stutz, and P. Cheeseman: 1991, 'Bayesian Classification Theory'. Technical Report FIA-90-12-7-01, NASA Ames Research Center.

Hörnel, D.: 1998, 'A Multi-scale Neural-Network Mode for Learning and Reproducing Chorale Variations'. In: W. B. Hewlett and E. Selfridge-Field (eds.): *Melodic Similarity; Concepts, Procedures, and Applications*. Cambridge, MA: The MIT Press, pp. 141–158.

Hörnel, D., J. Langnickel, B. Sandberger, and B. Sieling: 1999, 'Statistical vs. Connectionist Models of Bebop Improvisation'. In: *Proceedings of the International Computer Music Conference (ICMC'99)*, Beijing, China, pp. 244–247.

Hörnel, D. and W. Menzel: 1998, 'Learning Musical Structure & Style with Neural Networks'. *Computer Music Journal* **22**(4), 44–62.

Johnson-Laird, P. N.: 1991, 'Jazz Improvisation: A Theory at the Computational Level'. In: P. Howell (ed.): *Representing Musical Structure*. Orlando, FL: Academic Press, Inc., pp. 291–325.

Krumhansl, C. M.: 1990, *Cognitive Foundations of Musical Pitch*. Oxford, UK: Oxford University Press.

Lartillot, O., S. Dubnov, G. Assayag, and G. Bejerano: 2001, 'Automatic Modeling of Musical Style'. In: *Proceedings of the International Computer Music Conference (ICMC'2001)*, Havana, Cuba, pp. 447–454.

Levitt, D.: 1981, 'A Melody Description System for Jazz Improvisation'. Master's thesis, Massachusetts Institute of Technology.

Loy, G.: 1991, 'Connectionism and Musiconomy'. In: P. M. Todd and D. G. Loy (eds.): *Music and Connectionism*. Cambridge, MA: MIT Press, pp. 23–36.

Marsden, A.: 1999, 'Representation of Melodies as Elaboration Trees'. In: *Proceedings from the AISB'99 Symposium on Musical Creativity*, Edinburgh, Scotland, pp. 70–75.

Mateas, M.: 1999, 'An Oz-Centric Review of Interactive Drama & Believable Agents'. In: M. J. Wooldridge and M. M. Veloso (eds.): *Lecture Notes in Artificial Intelligence*, Vol. 1600. Heidelberg: Springer-Verlag, pp. 297–329.

Mateas, M.: 2001, 'Expressive AI'. *Leonardo: The Journal of the International Society of the Arts, Sciences and Technology* **34**(2), 147–153.

McLachlan, G. J. and K. E. Basford: 1988, *Mixture Models: Inference & Applications to Clustering*. New York, NY: Marcel Dekker.

Mitchell, T. M.: 1997, *Machine Learning*. New York, NY: McGraw-Hill.

Mongeau, M. and D. Sankoff: 1990, 'Comparison of Musical Sequences'. *Computers and the Humanities* **24**, 161–175.

Mukherjee, R., P. S. Dutta, N. Sajja, and S. Sen: In Print, 'A Movie Recommendation System – an Application of Voting Theory to User Modeling'. *User Modeling and User-Adapted Interaction, Special Issue on User Modeling and Intelligent Agents*. In this issue.

Nigam, K., A. McCallum, S. Thrun, and T. Mitchell: 1998, 'Learning to Classify Text from Labeled & Unlabeled Documents'. In: *Proceedings of the American Association of Artificial Intelligence (AAAI'98)*, Madison, USA, pp. 792–799.

Ostransky, L.: 1977, *Understanding Jazz*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Pachet, F.: 2000, 'Computer Analysis of Jazz Chord Sequence: Is *Solar* a Blues?'. In: E. R. Miranda (ed.): *Readings in Music and Artificial Intelligence*. London, UK: Harwood Academic Publishers.

Papadopoulos, G. and G. Wiggins: 1999, 'AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects'. In: *Proceedings from the AISB'99 Symposium on Musical Creativity*, Edinburgh, Scotland, pp. 110–117.

Pennycook, B. and D. Stammen: 1993, 'Real-time Recognition of Melodic Fragments Using the Dynamic Timewarp Algorithm'. In: *Proceedings of the International Computer Music Conference (ICMC'93)*, Japan, pp. 232–235.

PGMusic: 1999, *Band-In-a-Box*. `http://www.pgmusic.com`.
Ramoni, M., P. Sebastiani, and P. R. Cohen: 2000, 'Multivariate Clustering by Dynamics'. In: *Proceedings of the American Association of Artificial Intelligence (AAAI'2000)*, Austin, USA, pp. 633–638.
Raphael, C.: 1999, 'Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models'. *IEEE Transactions on Pattern Analysis & Machine Intelligence* **10**(4), 360–370.
Roads, C.: 1985, 'Improvisation with George Lewis'. In: C. Roads (ed.): *Composers and the Computer*. Los Altos, CA: William Kaufmann, Inc., pp. 75–88.
Rolland, P. and J. Ganascia: 1996, 'Automated Motive-Oriented Analysis of Musical Corpuses: a Jazz Case Study'. In: *Proceedings of the International Computer Music Conference (ICMC'96)*, Hong Kong, pp. 240–243.
Rowe, R.: 1993, *Interactive Music Systems : Machine Listening & Composing*. Cambridge, MA: MIT Press.
Russell, G.: 1959, *The Lydian Chromatic Concept of Tonal Organization for Improvisation: For All Instruments*. Cambridge, MA: Concept Publishing.
Sengers, P.: 1998, 'Anti-Boxology: Agent Design in a Cultural Context'. Ph.D. thesis, Computer Science Department, Carnegie Mellon University.
Smyth, P.: 1999, 'Probabilistic, Model-Based Clustering of Multivariate and Sequential Data'. In: *Proceedings of the 7th International Workshop on AI and Statistics*.
Steinsaltz, D. and D. Wessel: In Progress, 'The Markov Melody Engine: Generating Random Melodies With Two-Step Markov Chains'. Technical report, Department of Statistics, University of California at Berkeley.
Sudnow, D.: 1978, *Ways of the Hand, the Organization of Improvised Conduct*. Cambridge, MA: Harvard University Press.
Thom, B.: 2000a, 'Artificial Intelligence and Real-Time Interactive Improvisation'. In: *Proceedings from the AAAI-2000 Music and AI Workshop*, Austin, USA, pp. 35–39.
Thom, B.: 2000b, 'Unsupervised Learning and Interactive Jazz/Blues Improvisation'. In: *Proceedings of the American Association of Artificial Intelligence (AAAI'2000)*, Austin, USA, pp. 652–657.
Thom, B.: July 2001, 'BoB: An Improvisational Music Companion'. Ph.D. thesis, Computer Science Department, Carnegie Mellon University.
Toiviainen, P.: 1995, 'Modeling the target-note technique of bebop-style improvisation'. *Music Perception* **12**(4), 399–413.
Walker, W.: 1997, 'A Computer Participant in Musical Improvisation'. In: *CHI 97 Electronic Publications*.
Wessel, D. and M. Wright: 2000, 'Problems and Prospects for Intimate Musical Control of Computers'. In: *CHI '01: Workshop New Interfaces for Musical Expression*, Seattle, USA.
Wessel, D., M. Wright, and S. A. Kahn: 1998, 'Preparation for Improvised Performance in Collaboration with a Khyal Singer'. In: *Proceedings of the International Computer Music Conference (ICMC'98)*, Ann Arbor, USA, pp. 497–503.
Winkler, T.: 1998, *Composing Interactive Music: Techniques and Ideas Using MAX*. Cambridge, MA: MIT Press.

Appendix

A. Music Terminology

Most of the musical terms defined in this appendix were adapted from Gridley (1988) and Johnson-Laird (1991).

MIDI: Musical Instrument Digital Interface.

bar of solo: a spontaneously improvised melody that occupies one metrical group of beats.

beat: a song's underlying pulse (one often taps their foot to the beat).

chord: three or more musical tones sounded simultaneously.

chromatic scale: the pitches that result when an octave is broken up into twelve logarithmically equidistant parts.

chromaticism: refers to the inclusion of tones outside of a particular key or scale.

duration: refers to how much time a sound event occupies.

enharmonic distinction: theoretically, there is a difference between a sharped and a flatted pitch, e.g., $E\flat \neq D\#$; in BoB (see the definition of pitch class), this distinction is ignored.

harmony: the structure of music with respect to the progression of its chords.

interval: the distance between any two adjacent pitches in a melody.

key / modulation / tonality: these terms are explained together because their definitions are interrelated. Key relates to the predominant Major or Minor Scale associated with a tonal piece of music. Modulation relates to the change from one key or 'tonal center' to another within a piece of music (in jazz, modulation is very common). Tonality refers to the organization of a melody and/or harmony in relation to a tonic, which affects which tones will seem more consonant/stable or dissonant/instable.

melody: a temporally-ordered, monophonic sound event sequence.

meter: a song's predominant repetitive beat grouping (e.g., in rock-n-roll, every fourth beat typically receives the most emphasis).

modal: with respect to improvisation, this term usually indicates that the underlying harmony only contains several chords that do not change very quickly.

monophonic: music composed of a single voice.

octave: the relationship that occurs when the ratio between two pitches' fundamental frequencies is two.

pitch: a discrete, perceptual value associated with a tone's fundamental frequency.

pitch class: by ignoring in which octave a tone resides, pitch class maps a tone onto one of the twelve semitones that are contained within each octave; pitch class values include: $\{C, D\flat, D, E\flat, E, F, G\flat, G, A\flat, A, B\flat, B\}$.

pitch tracking / rhythmic quantization / tempo tracking: these terms are explained together because their definitions are interrelated. Pitch tracking is the conversion an audio signal into a stream of discrete pitches and rests. Tempo tracking is the corresponding segmentation of this stream into beat locations. Rhythmic quantization describes the mapping from arbitrary inter-beat locations onto discrete, hierarchical rhythms.

rest: a silence (no pitch is sounding).

riff / lick / motive: each of these terms to refer to short, melody fragments.

rhythm: refers to the arrangement of sound events hierarchically in time and relates to the discrete perception of rhythm (e.g., when a musician plays two pitches within a single beat, together they might be perceived as dividing the beat in half, even though their performed durations might not be identical).

scale: refers to a specific, ordered set of pitch classes (e.g., the B♭-Major Scale is comprised of the set $\{B\flat, C, D, E\flat, F, G, A\}$).

semitone: the distance between any two adjacent pitches in the chromatic scale.

solo: a consecutive sequence of bars of solo.

sound event: a pitch or rest and its corresponding duration.

syncopation: this term refers to rhythms whose durations span beat boundaries.

tempo: the rate at which beats pass.

tonic: relating to or based on the first tone of a scale (e.g., in the B♭-Major Scale, the tonic is B♭).

transcription: amounts to converting an audio signal into a musical score.