Chapter 7
Analysing Symbolic Music with Probabilistic Grammars

Samer Abdallah, Nicolas Gold, and Alan Marsden

Abstract Recent developments in computational linguistics offer ways to approach the analysis of musical structure by inducing probabilistic models (in the form of grammars) over a corpus of music. These can produce idiomatic sentences from a probabilistic model of the musical language and thus offer explanations of the musical structures they model. This chapter surveys historical and current work in musical analysis using grammars, based on computational linguistic approaches. We outline the theory of probabilistic grammars and illustrate their implementation in Prolog using PRISM. Our experiments on learning the probabilities for simple grammars from pitch sequences in two kinds of symbolic musical corpora are summarized. The results support our claim that probabilistic grammars are a promising framework for computational music analysis, but also indicate that further work is required to establish their superiority over Markov models.

7.1 Introduction

Music is, arguably and amongst other things, *structured* sound. *Music analysis* is that branch of musicology which aims to explain the structure of pieces of music, in the sense of giving an account both of the relationships between different parts of the same piece, and the relationships between the piece and patterns common to other pieces of music. Theories of music analysis thus typically identify, classify, and relate musically meaningful parts of a work or works. These structural aspects of a piece of music, on the small and large scales, are crucial to its impact.

Samer Abdallah · Nicolas Gold
Department of Computer Science, University College London, London, UK,
e-mail: {s.abdallah, n.gold}@ucl.ac.uk

Alan Marsden
Lancaster Institute for the Contemporary Arts, Lancaster University, Lancaster, UK,
e-mail: a.marsden@lancaster.ac.uk

Music analysis, whether formal or informal, thus relies on "data": this includes the piece itself in sonic and/or symbolic form, but also, in most cases, a wealth of background knowledge and experience to make both the structure of a piece of music and the derivation of that structure explicit and so open to scrutiny.

We observe that listeners come to be able to perceive structure in pieces of music through mere exposure, though they might not be able to give an account of *how* they have come to perceive this structure, nor might they be able to justify it. In both these respects, music resembles language: speakers are able to understand and construct grammatical sentences, but they often require training to be able to explain what makes a sentence grammatical.

Shannon's (1948) work on information theory perhaps offers some explanation for this, in short, indicating that structure exists in any *departure from complete randomness* (this is explored further in the next section). Humans learn probabilities implicitly from exposure, and so effectively learn to perceive structure. This suggests that probabilistic modelling may offer a fruitful approach to music analysis (and one that likely demands a computational approach).

The field of computational linguistics has recently developed techniques that can be applied in the analysis of musical structure. These can induce models over a corpus of music. Such models take the form of probabilistic grammars from which idiomatic sentences can be produced. The derivation and application of the grammar rules thus offers an explanation of the musical structures modelled by them.

This chapter surveys historical and current research in computational linguistics applied to symbolic music analysis (by which we mean analysis of music in the form of symbolic, score-like data). The principles and operation of key developments are discussed, and we summarize our recent feasibility studies in using probabilistic programming techniques to provide tractable computation in this framework.

7.2 Information and Structure

Information theory provides a number of concepts that can help us to understand and quantify what we mean by 'structure'. Shannon's (1948) information theory is, to a large degree, concerned with the notion of *uncertainty*, quantified as *entropy*, and how the reduction of uncertainty can be considered a gain in information. Entropy is a function of probability distributions, and probability distributions can represent *subjective* beliefs, that is, the degrees of belief that an intelligent agent (whether biological like ourselves or artificial like our computers) places in a set of mutually exclusive propositions.[1] These degrees of belief can be based on any knowledge currently possessed by the agent in combination with any prior or innate dispositions it may have.

[1] Indeed, this *subjectivist* view of probability was espoused by, among others, de Finetti (1975), while Cox (1946) showed that, given certain reasonable assumptions, any system for reasoning consistently with numerical degrees of belief must be equivalent to probability theory.

Psychologists such as Attneave (1954) and Barlow (1961) proposed that perceptual systems in animals are attuned to the detection of *redundancy* in sensory signals, which means, essentially, the ability to predict one part of the sensory field from another. Often, the reason for such sensory regularities is the presence of coherent objects and processes in the outside world, so detecting and accounting for redundancy can enable the mind to see beyond the mass of sensory signals to relevant phenomena in the world. Redundancy of this sort ensues whenever different parts of the sensory field depart from complete statistical independence. This can be taken as a definition of what 'structure' is, and perceptual learning can be seen as a process of detecting statistical regularities on exposure to structured objects, and thereby making inferences about events in the world. This idea has successfully accounted for many features of perceptual processing (Knill and Pouget, 2004).

Thus, the use of probabilistic models, an approach on which we shall elaborate further in Sect. 7.4, is not merely a computational device: it is at the heart of our current understanding of human perception and cognition (Knill and Richards, 1996) and also addresses ideas about the role of uncertainty and expectation in music (Meyer, 1956).

A distinct, but related branch of information theory is *algorithmic information theory*, which is built on the idea that any object which can be represented as a sequence of symbols can also be represented by a computer program which outputs those symbols when run. Sometimes, a very short computer program can output a very long sequence of symbols, and hence, an ostensibly large object may be specified by a much smaller program. The *Kolmogorov complexity* (Li and Vitányi, 2009) of such an object is defined as the length of the smallest program that outputs the object when run, and can be considered a measure of the amount of information in the original object. As such, it is related to the *minimum description length* principle (Rissanen, 1978). Kolmogorov complexity also forms the basis of Martin-Löf's (1966) definition of a random object as one which does not admit of any smaller description and therefore cannot be compressed; this corresponds loosely with the notion of redundancy described above.

Given the close connection between predictability and compressibility (Cover and Thomas, 1991), and the probabilistic basis of optimal compression algorithms, we observe that in both theoretical frameworks (Shannon's information theory and algorithmic information theory), *structure* can be defined as that which enables an apparently large object to be represented more efficiently by a smaller description. The probabilistic approach and its subjectivist interpretation also emphasize that predictability (and hence compressibility) depend on the expectations of the observer, and thus can accommodate the subjectivity of perception in a natural manner. In addition, while the Kolmogorov complexity of an object is claimed to be an *objective* measure, in practice, it must be approximated by using compression programs that encode implicit or explicit assumptions about the data to be compressed and that are usually based on probabilistic models.

7.3 Theories of Structural Music Analysis

There are various established approaches to structure in music analysis, each of which gives quite a different account of what a musical structure is. A Schenkerian analysis (Schenker, 1935), for example, reveals the structure of a piece of music in hierarchical layers of reduction, resulting in something like a tree or some other kind of restricted directed graph (Fig. 7.1).[2] A paradigmatic and syntagmatic analysis in the manner of Nattiez (1975), on the other hand, divides the piece into units (syntagms) and groups those units by recurrence or resemblance into paradigms. At a higher level, then, the structure of the piece can be shown as a sequence of syntagms, each of which is an instance of a particular paradigm (Fig. 7.2). Theories of form, dating back to the nineteenth century, identified themes and other kinds of segments of music, assigning each to a role within one of a number of established formal patterns such as Sonata Form, Rondo or Binary.

Each of these analytical approaches has three common factors:

Segmentation	The music is divided into meaningful units.
Classification	Segments are assigned to classes.
Relation	Segments are related to each other according to their role in forming patterns of significance with other segments.

In the case of syntagmatic and paradigmatic analysis, the first two factors are clear, and the third often follows in a later stage of the analysis where patterns in sequences of instances of paradigms are identified, or a pattern in the occurrence of particular paradigms, or in their evolution. In the case of Schenkerian analysis, the first two factors work together, as groups of notes are identified as prolongations of notes at a

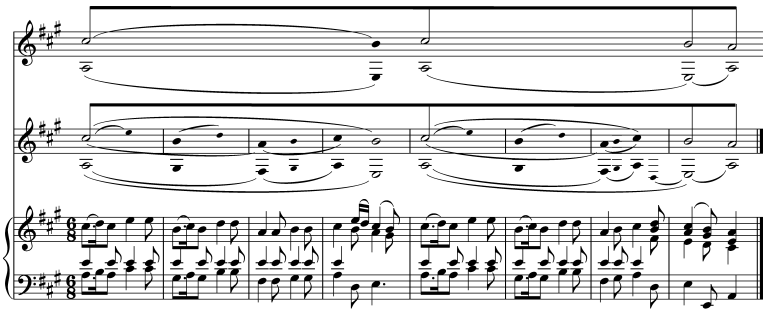


Fig. 7.1 An example Schenkerian analysis of the opening of Mozart's piano sonata in A major, K. 331. Notes on lower staves are considered to elaborate the structural notes with which they align or between which they occur on the staff above. Notes on the upper two staves with smaller noteheads are considered to elaborate notes with larger noteheads, and ones with solid noteheads to elaborate ones with open noteheads. The beam links the three notes forming the *Urlinie* which Schenker considered to constitute the fundamental line of every proper piece of tonal music

[2] For an explanation of the relation of Schenkerian analyses to trees and graphs, see Marsden (2005).

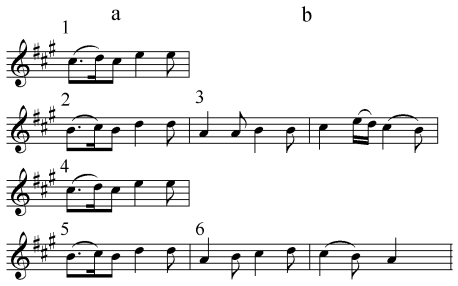


Fig. 7.2 An example syntagmatic and paradigmatic analysis of the opening of Mozart's piano sonata in A major, K. 331. The melody is divided into six syntagms, identified as 1–6, and organized into two paradigms, a and b

higher level. The patterns of those higher-level notes in turn govern the relations of the lower-level units. Even such an apparently different approach to musical structure as *set theory* (Forte, 1973) employs these three factors: groups of adjacent notes are separated from other notes (segmentation); the pitch-class set of each group is identified (classification); and the structure of the piece is considered to be determined in part by the relations between the pitch-class sets.

These three are also factors in language, and grammars exist to explain how a sentence falls into a relational structure of words (segments) according to the parts of speech (classes). We therefore consider the idea of grammar to be applicable to the analysis of musical structure not only because of the oft-remarked correspondence between Schenkerian reduction and parsing a sentence, but also because of a deeper correspondence between the factors determining structure in both music and language.

This is not to deny that there are also obvious and significant differences between language and music, in particular the often multidimensional (in the sense that a musical event can usually be characterized in multiple ways, such as pitch, duration, loudness, timbre, etc.) and polyphonic nature of music. These characteristics may necessitate the development of grammars that go beyond those commonly used in linguistics, though we note that even natural language can sometimes be described in a multidimensional way (Bilmes and Kirchhoff, 2003).

Furthermore, we find grammars particularly suited to computational analysis of music because (a) they are liable to implementation in computer software, and (b) they can be learned or derived from examples of sentences or pieces of music. The latter consideration is important, first, because it is evident that most people come to be able to perceive structure in music simply by exposure to it,[3] and second, because crafting a grammar 'by hand' is not always practical for music (as we discuss below).

[3] We do not claim, however, that people will necessarily come to be able to *produce* musical structures simply by exposure.

7.3.1 Links to Linguistics

It is in the field of computational linguistics that grammars have been best developed, and so we will briefly review some of the important developments in linguistics before narrowing our focus to grammars in music later in this section.

Languages and Grammars A *language* is, in theoretical terms, defined as follows: given a well-defined set of symbols, or *alphabet*, a *sentence* is a finite sequence of such symbols, and a language is a set of sentences. This set may be finite or infinite; for example, if we take as our alphabet the set of lower-case latin letters a, b, c etc., then

$$\{pat, pet, pit, pot, put\} \tag{L1}$$

is a finite language, while the languages

$$\{ab, abab, ababab, abababab, \ldots\} \tag{L2}$$

$$\{ab, aabb, aaabbb, aaaabbbb, \ldots\}\,, \tag{L3}$$

continued in the obvious way, are both infinite. A grammar specifies, by means of rewrite or production rules, a recursive generative process which produces such sentences and therefore defines a language. More formally, a grammar consists of a set of *terminal* symbols that will form the alphabet of the language, a disjoint set of *non-terminal* symbols, a set of production rules describing how certain sequences of symbols may be re-written, and a distinguished non-terminal called the start symbol (conventionally S). For example the language (L1) results from the following rules for non-terminals S and V:

$$\begin{aligned} S &\to pVt, \\ V &\to a \mid e \mid i \mid o \mid u\,, \end{aligned} \tag{G1}$$

where $V \to a \mid e \mid \ldots$ is shorthand for the multiple rules $V \to a, V \to e$, etc. Similarly, language (L2), sometimes referred to as $(ab)^n$, results from

$$S \to ab \mid abS \tag{G2}$$

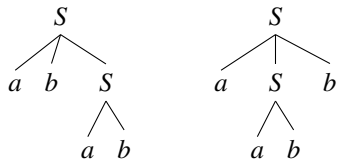
and (L3) (or $a^n b^n$) from

$$S \to ab \mid aSb\,. \tag{G3}$$

Note that the two infinite languages involve *recursive* production rules, that is, non-terminals that can expand to a sequence containing themselves.

As well as providing a recipe for *generating* strings from the language, a grammar also defines a way to *analyse* a given sentence in terms of the sequence of rule applications that could have been used to generate it, a process known as *parsing*. Thus, parsing seeks an *explanation* of the observed sequence in terms of possible generative structures and their semantic implications.

A particularly important class of grammars is that of the *context-free grammars*. A context-free grammar (or CFG) is a grammar in which only single non-terminal symbols appear on the left-hand side of production rules. All of the examples above are context-free grammars. Parsing a sequence using a CFG results in a *syntax tree*; for example, the sentences *abab* and *aabb* can be parsed using grammars (G2) and (G3) respectively as follows:



grammar (G2) grammar (G3)

Hierarchy of Grammars Chomsky (1957) classified grammars on the basis of their production rules into a hierarchy, with simple "regular" grammars at the bottom, followed by "context-free", "context-sensitive", and finally "unrestricted" grammars at the top. Each level up is more general than the level below. For example, grammars (G1) and (G2), as well as being context-free, are also members of the more restricted class of regular grammars. There is an intimate relationship between the levels of the hierarchy and the complexity of the computing machine, or *automaton*, that is required to judge whether or not a given sequence is grammatical. Grammars higher in the hierarchy can produce languages with more complex structures, but require more computational resources (time and memory) to process. For example, regular grammars require only a *finite state automaton* for recognition of grammatical sequences; CFGs require a *push-down* automaton, which essentially means an automaton with an arbitrarily deep stack in order to keep track of arbitrarily deep recursive structures; while unrestricted grammars require a Turing tape for recognition, or a Turing machine, which is the most general type of computing machine, for further processing. Given that natural languages are for human consumption, this places limits on the complexity of the grammars that might describe them, especially when the situation demands real-time production or understanding (e.g., during conversation). For this reason, the consensus seems to be that natural languages are "mildly" context-sensitive. Whether or not the same applies to music is an interesting question. Johnson-Laird (1991) suggests that, during jazz improvisation, limitations on working memory demand that improvised structures can be described by at most a regular grammar, whereas precomposed structures, like the chord sequence, may be described by a context-free grammar (CFG) or higher. Similarly, we might posit that, on a first listening, only structures describable by a regular grammar are accessible, but repeated listening (and analysis) might reveal more complex structures, the utility of which is to explain surface features which may have seemed surprising or arbitrary when analysed using the simpler grammar.

Formalisms A variety of grammar formalisms have been proposed in order to deal with linguistic structures that cannot be described simply in a context-free way; these include tree adjoining grammars (Joshi et al., 1975), definite clause grammars or

DCGs (Pereira and Warren, 1980), extraposition grammars (Pereira, 1981), a variety of unification based grammars (Shieber, 1985), and type-theoretical grammars such as Lambek calculus (Lambek, 1958) and combinatory categorical grammars or CCGs (Steedman, 2001; Steedman and Baldridge, 2011). A unifying idea behind all these grammar formalisms is that they attempt to provide a system, or meta-language, for specifying grammars, that is just powerful enough to describe linguistic structures in the domain of interest, but not so powerful that it leads to intractable computations. In the case of both natural languages and music, this seems to require something more than a (finite) CFG, but less than an unrestricted grammar. With the exception of the DCG formalism, which is Turing complete (i.e., capable of performing any computation performable by a Turing machine), the carefully limited complexity of these meta-languages means that generation and parsing is computationally tractable and so practically usable. Thus we may think of the grammar formalism as a kind of special purpose programming language, which is, in the terminology of programming language design, a *high-level*, *declarative*, *domain specific language* (DSL) designed to make sentence structure easy to describe.

Parsing A wide variety of parsing technologies have been studied. DCGs written in Prolog can be translated directly into an executable program that can both generate and recognize sentences from the language: Prolog's execution strategy results in top down (recursive descent) parsing; however, this can be inefficient for very ambiguous grammars. More efficient parsers can be written using dynamic programming techniques to avoid redundant computations; these *chart parsers* include the bottom-up CKY algorithm (Younger, 1967) and Earley's top-down parser (Earley, 1970). The close relationship between Earley's algorithm and tabling or memoization in general purpose logic programming (also known as Earley deduction) is well-known (Pereira and Warren, 1983; Porter, 1986), and can be carried over very elegantly to functional programming languages such as Scheme and Haskell using *memoizing parser combinators* (Frost and Hafiz, 2006; Johnson, 1995; Norvig, 1991).

Probabilistic Grammars When a grammar is extended to cover a wide corpus, then the number of production rules and/or the size of the lexical database means that parsing can become very expensive due to the resulting high degree of ambiguity. It was recognized that a majority of these alternative parses would be nonsensical, and that this notion could be characterized by noticing that certain words or constructs are much more likely to occur than others. Thus, though it is conceivable that "[the man] [saw [the dog with a telescope]]", it is much more likely that "[the man] [[saw [the dog]] [with a telescope]]". A grammar augmented with probabilities becomes a probabilistic language model, capable of being used in either direction: assigning probabilities to sentences and their syntax trees, or generating idiomatic as opposed to merely grammatical sentences.

Perhaps the simplest examples are probabilistic context-free grammars (PCFGs), which consist of a CFG supplemented with, for each non-terminal, a probability distribution over the possible expansions of that non-terminal. The *inside-out algorithm* (Baker, 1979) can compute the probabilities of alternative parses. Efficient algorithms for probabilistic parsing were developed during the 1990s (Abney, 1997; Collins,

1999; Goodman, 1998; Stolcke, 1995) and have gone on to revolutionize computational linguistics, resulting in the current state-of-the-art in parsing and natural language understanding.

Probabilistic grammars can support *grammar induction*, which is a form of inductive learning where the grammar rules are not given, but must be inferred from a collection of sample sentences. This is analogous to the situation in which a child finds itself in the first few years of life: beginning with no knowledge of nouns, verbs, etc., the child gains enough implicit grammar to produce (more or less) grammatical utterances simply from hearing spoken language. This process has been modelled in several ways; for example, Kurihara and Sato (2006) used rule splitting to explore the space of grammar rules and Bayesian model selection criteria to choose from the resulting grammars; Bod (2006) applied his *data oriented parsing* method, which builds a collection of commonly occurring syntactic sub-trees, to both language and music; and O'Donnell et al. (2009) used Bayesian nonparametric models to achieve a similar goal.

One family of probabilistic grammars is based on so-called 'log-linear' or 'undirected' models, which involve assigning numerical likelihoods to grammatical features in such a way that they need not sum to one, as probabilities are required to do (Abney, 1997; Charniak, 2000; Collins, 1997, 2003). These can yield flexible probability models, but, due to this lack of normalization, suffer from some technical difficulties in learning their parameters from a corpus. Goodman's (1998) *probabilistic feature grammars* retain much of the flexibility but avoid these problems by using directed dependencies to obtain a properly normalized probability model. Probabilistic extensions of CCG have been developed (Hockenmaier, 2001; Hockenmaier and Steedman, 2002), also using log-linear models, with statistical parsing and parameter learning algorithms (Clark and Curran, 2003, 2007).

7.3.2 Grammars in Musicology

While the parallels between music and linguistics go back much further (Powers, 1980), the application of formal grammars in music began in the 1960s. Winograd (1968) used a grammar formalism called "systemic grammar" to define a grammar for music, covering cadential and harmonic progression, chord voicing and voice leading, and presented a LISP implementation to parse a given fragment. Despite the use of heuristics to guide the search, high computational complexity meant that the system was limited to analysing relatively small fragments.

Kassler (1967) made steps toward formalizing Schenker's theory , encoding recursive functions to describe the process of elaboration from *Ursatz* to middleground for the top and bass voices in a polyphonic score, showing that the resulting grammar was decidable, meaning there exists an algorithm (given in the paper) which can determine in a finite number of steps whether or not a given musical sequence is a member of the language defined by the grammar. Software, using backtracking search and written in APL, was presented in later work (Kassler, 1976, 1987).

Another system notable for its wide scope and technological sophistication was Ebcioğlu's (1987) CHORAL, which was implemented in a custom logic programming language and included nondeterminism and heuristically-guided "best-first" backtracking search. It was applied to harmonizing chorales in the style of J. S. Bach.

Other researchers have focused on narrower goals, such as melody analysis (Baroni et al., 1983; Lindblom and Sundberg, 1970), jazz chord analysis (Pachet, 2000; Pachet et al., 1996; Steedman, 1984; Ulrich, 1977) and grammars for melodic improvisation (Johnson-Laird, 1991). Smoliar's (1980) system was notable in that it did not attempt to do Schenkerian analysis *automatically*, but instead provided the analyst with a collection of tools to facilitate the process.

The *Generative Theory of Tonal Music* (GTTM) of Lerdahl and Jackendoff (1983) was perhaps one of the more complete attempts to account for structure in music, including melodic phrase structure, metrical structure, and hierarchical reduction similar to that described by Schenker. Lerdahl and Jackendoff expressed their theory in generative rules, but their approach was not computational. Subsequent attempts to implement the theory (e.g., Hamanaka et al., 2006, 2007, see also Chaps. 9 and 10, this volume) have encountered difficulty with their 'preference rule' concept.

More recent years have seen developments in computational linguistics, especially probabilistic grammars, filtering back into musicological work. For example, Steedman's chord grammar has inspired a number of researchers to apply more sophisticated grammar formalisms to more general harmonic models (Granroth-Wilding, 2013; Granroth-Wilding and Steedman, 2012; Rohrmeier, 2006, 2011; Steedman, 2003).

On the melodic side, Mavromatis and Brown (2004) reported that they had been able to design a grammar for Schenkerian analysis as a Prolog DCG. Schenker's melodic elaborations are similar to grammar production rules but because some of them, such as the introduction of neighbour notes or passing notes, depend on *two* adjacent notes, they cannot be written as a *context-free* grammar if the melody is represented as a sequence of pitches. Instead, Mavromatis and Brown represented melodic sequences as sequences of pitch *intervals* and were thus able to devise a CFG to embody melodic elaborations. However, practical obstacles to a working implementation were encountered, due to the large number of rules required.

Gilbert and Conklin (2007) also adopted this approach and designed a PCFG to model both melodic and rhythmic elaborations. Their grammar included five basic melodic elaborations (including the repeat, neighbour, passing, and escape types of ornamentation used in our experiments, described in Sect. 7.5). Figure 7.3 shows how a short melodic sequence can be represented as a sequence of pitch intervals and described in terms of a syntax tree using two of the elaboration rules (the *term* rule simply emits a terminal symbol). We will build on this grammar later in Sect. 7.5.

Issues of how to represent tree-like elaboration structures to enable a grammar-based analysis were examined by Marsden (2005), who showed that adopting an interval-based encoding, though sufficient to allow a context-free description for some types of melodic elaboration, is not sufficient in general to cope with other musical structures such as suspensions and anticipations which are (locally) context-dependent in a fundamental way. He went on to develop software for the analysis of

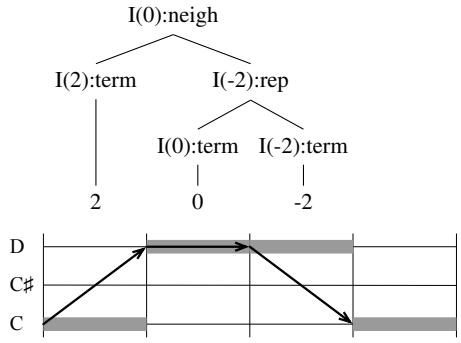


Fig. 7.3 An example of a syntax tree for a short sequence of melodic intervals. The grid below is a piano-roll representation of the notes with the arrows showing the pitch (in semitones) and time intervals between successive notes

short fragments based on bottom-up chart parsing, heuristics for ranking the quality of analyses, and pruning (similar to beam search) to limit the search space (Marsden, 2007, 2010, 2011).

Kirlin and Jensen (2011) and Kirlin (2014) also base their probabilistic model of musical hierarchies on the elaboration of intervals, adopting Yust's (2009) triangulated graphs as their structured representation, rather than the trees of conventional grammatical analysis. Another recent grammar-based approach to music analysis is that of Sidorov et al. (2014), who implement a non-probabilistic form of grammar induction.

In the literature on music theory, Temperley (2007) is the most prominent application of a probabilistic approach. Most of the book is concerned with recognition of high-level features such as metre and key rather than structures at the level of phrases and notes, but it does include a discussion of a possible approach to modelling Schenkerian analysis through a probabilistic grammar (pp. 172–179). This chapter could be seen in part as a response to Temperley's challenge to Schenkerian theorists to demonstrate how the theory 'reduces the uncertainty of tonal music' (p. 179).

7.4 Probabilistic Models

A probabilistic model (henceforth, simply 'model') of a domain is essentially an assignment of probabilities to things in that domain. It is these probabilities which determine how surprising a thing is and, in the case of a partially observed temporally unfolding object such as a piece of music, how it might continue.

Although models and data may seem rather abstracted from issues of music, if one considers 'data' to be one or many musical works in symbolic form and a 'model' as, for example, a structural analysis of a single piece, or a music theory for a large corpus of works, the application of these concepts becomes clearer.

7.4.1 Model Selection Criteria

The most effective models are adaptable to new situations (such as new musical styles) on the basis of observations. Even when listening to an individual piece, our expectations are fluid and adaptable: a theme or motif is less surprising when heard a second time or in some subsequent variation.

In all practical cases, the data is finite. Unless we have very specific information about how the data was generated, it will not be possible to determine a single 'correct' probabilistic model. For one thing, it is not possible to extract the infinite amount of information required to determine real-valued parameters from data that contains only a finite amount of information about them; for example, neither the mean and variance of a Gaussian distribution, nor the probability of a biased die rolling a six can be determined with certainty from a finite sequence of observations. In another, deeper sense, if the amount if data is limited in an essential way, such that there *cannot be* any more data (for example, the set of pieces composed by Ravel) then it can be argued that an objective probability distribution describing this set does not exist. To give another example, in a certain literal sense, the only pieces in the style of Mozart are the ones he actually wrote. However, musicians regularly use this concept, and broadly mean 'if Mozart had written another piece, then it might have been like this'. This of course is contrafactual, making clear why an objectively 'correct' model is not possible. Thus candidate models must be evaluated to determine which is the most plausible given the data available.

A number of familiar machine-learning issues arise in this problem of 'model selection'. One is *over-fitting*, where an overly complex model becomes too tightly coupled to incidental rather than essential features of the data, leading to poor *generalization*. Conversely, an overly simple model may *under-fit*, and not capture enough of the regularities that are in the data.

Bayesian model selection criteria offer a theoretically and philosophically appealing approach to addressing these issues (Dowe et al., 2007; Kass and Raftery, 1995). Bayesian inference is underpinned by the consistent use of probability to capture the inferrer's uncertainty about everything under consideration, including the models themselves (and their parameters). An inferring agent with model $\mathcal{M}$ with parameters $\theta \in \Theta$ (where Θ is the parameter space) can initially only set those parameters as an uncertain probability distribution (the *prior* distribution $P(\theta|\mathcal{M})$). After observing some data $\mathcal{D}$, the agent will update the state of its belief, giving a *posterior* distribution:

$$P(\theta|\mathcal{D},\mathcal{M}) = \frac{P(\mathcal{D}|\theta,\mathcal{M})P(\theta|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})}\,, \tag{7.1}$$

This accounts for the prior and the likelihood that the model with parameters θ could have produced the data. To best predict a new item of data d, given the model and observations so far, the agent needs to compute

$$P(d|\mathcal{D},\mathcal{M}) = \int_{\Theta} P(d|\theta,\mathcal{M})P(\theta|\mathcal{D},\mathcal{M})\,\mathrm{d}\theta\ .$$

As long as the agent remembers the posterior distribution $P(\theta|\mathcal{D},\mathcal{M})$, the data is no longer needed. The *evidence* (the denominator in (7.1)) is computed thus

$$P(\mathcal{D}|\mathcal{M}) = \int_{\Theta} P(\mathcal{D}|\theta,\mathcal{M})P(\theta|\mathcal{M})\,\mathrm{d}\theta\ . \tag{7.2}$$

With multiple candidate models, inference takes place on distributions over those models (rather than parameters) with prior $P(\mathcal{M}_i)$ and posterior

$$P(\mathcal{M}_i|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{M}_i)P(\mathcal{M}_i)}{P(\mathcal{D})}\ . \tag{7.3}$$

If the prior over models $P(\mathcal{M}_i)$ is relatively flat (i.e., no model is distinctly more probable than others), then the evidence is the key determinant of the relative plausibility of each model after data observation. The posterior distribution can be used to make predictions and decisions (*model averaging*) and if a decision is required, the model with the greatest evidence can be selected. While it is clear that the evidence $P(\mathcal{D}|\mathcal{M})$ rewards models that are able to fit the data well in some part of their parameter space—since $P(\mathcal{D}|\theta,\mathcal{M})$ will be large in this region—it also penalizes models that are more complex than the data can support.

To see why this is so, consider that, for a given size of dataset, $P(\mathcal{D}|\mathcal{M})$ is a probability distribution over the space of complete datasets and therefore only has a finite amount of probability mass to spread out over that space. Simple models are only able to assign significant probability mass to a small set of datasets, while complex models betray their complexity by being able to assign probability mass to a much larger space of datasets. Figure 7.4 shows how this results in trade-off when we wish to choose a model for a particular observed dataset $\mathcal{D}_1$. In short, model $\mathcal{M}_1$ is too inflexible and a poor fit, while $\mathcal{M}_3$ is flexible enough but unnecessarily complex. Model $\mathcal{M}_2$ out-performs both.

This automatic penalty for overly-complex models gives us a formal expression of Ockham's razor, the philosophical principle that, other things being equal, we should choose the *simplest* explanation for observations.

Simplicity can also be expressed in terms of the *minimum message length* principle (Wallace and Boulton, 1968) and the related *minimum description length* principle (Rissanen, 1978). These state that models that produce the shortest possible description of the data (including the description of any necessary model parameters) should be selected. The close relationship between compression and probabilistic structure means that this is essentially the same as the Bayesian approach (MacKay, 2003), with some minor differences (Baxter and Oliver, 1994).

The relationship between message length, the Bayesian evidence, and model complexity can be illuminated further by recalling that, for optimal compression, an object x should be encoded by a message of length $\lceil -\log P(x) \rceil$ bits (assuming logarithms to the base 2), where $P(x)$ is the probability that both sender and receiver assign to x and $\lceil \cdot \rceil$ is the operation of rounding-up to an integer value. If the ob-

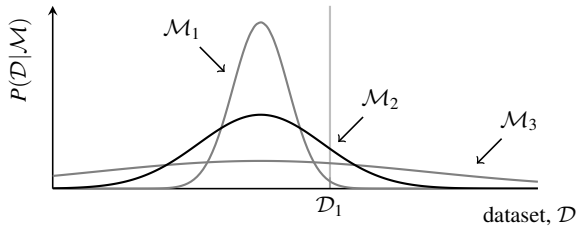


Fig. 7.4 An illustration of how model complexity and data-fit interact to determine the Bayesian evidence. The x-axis represents the space of all possible datasets of a given size (in most cases this will be a very high-dimensional space, but one dimension is sufficient to illustrate the principle). The curves are the probability distributions over datasets assigned by each of three models, so the total area under each curve is the same. $\mathcal{M}_1$ is not flexible enough to assign much probability to the observed dataset $\mathcal{D}_1$, while $\mathcal{M}_3$ is flexible enough to assign probability to a great variety of datasets, including $\mathcal{D}_1$. In doing so, however, it must spread out its probability mass more thinly. By being just flexible enough, $\mathcal{M}_2$ receives greater evidence than either $\mathcal{M}_1$ or $\mathcal{M}_3$

ject to be transmitted is a dataset $\mathcal{D}$, then clearly, the model $\mathcal{M}_i$ with the greatest evidence $P(\mathcal{D}|\mathcal{M}_i)$ will yield the shortest message, encoding the entire dataset in approximately $-\log P(\mathcal{D}|\mathcal{M}_i)$ bits.

Looking more closely at how the evidence relates to the model parameters θ, we can reason that a well-fitting model would assign a high probability to the data for a certain optimal value of θ, say $\hat{\theta}$. Given that $P(\mathcal{D}|\hat{\theta},\mathcal{M})$ is relatively high, a relatively short message of length approximately $-\log P(\mathcal{D}|\hat{\theta},\mathcal{M})$ bits could be sent to describe the dataset, but only if sender and receiver had agreed on $\hat{\theta}$ beforehand. Since $\hat{\theta}$ depends on the data that only the sender has access to, this is not possible, so the sender must send a message approximately $-\log P(\hat{\theta}|\mathcal{M})$ bits long describing $\hat{\theta}$ first. If the model is complex in the sense of having a large parameter space, this message may be large enough to offset any gains obtained from using a complex model to increase $P(\mathcal{D}|\hat{\theta},\mathcal{M})$.

The above considerations mean that the length of the message describing a dataset $\mathcal{D}$ will be approximately

$$-\log P(\mathcal{D}|\hat{\theta},\mathcal{M}) - \log P(\hat{\theta}|\mathcal{M}) = -\log P(\mathcal{D},\hat{\theta}|\mathcal{M})\,. \tag{7.4}$$

This is not the same as the $-\log P(\mathcal{D}|\mathcal{M})$ bits we estimated before considering the parameters. How can we resolve this discrepancy?

Bits Back Coding Let us look again at the length of the message required to send $\mathcal{D}$ and θ: this depends on $P(\mathcal{D},\theta|\mathcal{M})$ which, for a fixed dataset, is proportional to $P(\theta|\mathcal{D},\mathcal{M})$, the posterior distribution over the parameters given the data. If this distribution is relatively broad, it defines a *region* of Θ over which the total coding cost is approximately minimal. If, instead of sending the optimal value $\hat{\theta}$, the sender sends a value chosen *at random* from the posterior distribution, the coding cost will remain about the same, but this extra freedom of choice can allow an additional message to be sent at no extra cost. This is the basis of the 'bits back' coding scheme,

first described by Wallace (1990), but so named by Hinton and van Camp (1993). We can see how this accounts for the discrepancy noted above as follows:

$$\begin{aligned} P(\mathcal{D},\theta|\mathcal{M}) &= P(\theta|\mathcal{D},\mathcal{M})P(\mathcal{D}|\mathcal{M}) \,, \\ \log P(\mathcal{D},\theta|\mathcal{M}) &= \log P(\theta|\mathcal{D},\mathcal{M}) + \log P(\mathcal{D}|\mathcal{M}) \,, \\ \log P(\mathcal{D}|\mathcal{M}) &= \langle \log P(\mathcal{D},\theta|\mathcal{M})\rangle - \langle \log P(\theta|\mathcal{D},\mathcal{M})\rangle \,, \\ -\log P(\mathcal{D}|\mathcal{M}) &= \langle -\log P(\mathcal{D},\theta|\mathcal{M})\rangle - H(P_{\theta|\mathcal{D},\mathcal{M}}) \,, \end{aligned} \tag{7.5}$$

where $\langle\cdot\rangle$ denotes an expectation with respect to the posterior $P(\theta|\mathcal{D},\mathcal{M})$, and $H(\cdot)$ is the entropy of the given probability distribution. Thus, the difference between the theoretical message length $-\log P(\mathcal{D}|\mathcal{M})$ and the average message length for sending both $\mathcal{D}$ and θ when θ is chosen randomly from the posterior $P(\theta|\mathcal{D},\mathcal{M})$ is the entropy of the posterior.

Variational Free Energy Representing uncertainty about model parameters, computing the evidence and doing model averaging can be expensive operations computationally and approximations are often needed. For some models, *variational Bayesian learning* (Jordan et al., 1998; MacKay, 1997) can be a good solution, combining an efficient representation of uncertainty about parameters with a tractable learning algorithm, delivering an estimate of the evidence as a function of the *variational free energy* F. As with the bits back coding scheme, it too is defined by focusing on the posterior distribution $P(\theta|\mathcal{D},\mathcal{M})$. However, instead of trying to work with the exact posterior, we choose to approximate it with a distribution $Q(\theta)$ chosen from a more tractable class of distributions. The free energy is then defined as a function of this variational distribution Q:

$$F(Q) = -\langle \log P(\mathcal{D},\theta|\mathcal{M})\rangle_Q + \langle \log Q(\theta)\rangle_Q \tag{7.6}$$

where $\langle\cdot\rangle_Q$ denotes an expectation with respect to θ drawn from the variational distribution $Q(\theta)$. Note that this is the same, except for the reversal of sign, as the third line of (7.5) with Q replacing the true posterior $P_{\theta|\mathcal{D},\mathcal{M}}$. This tells us that $F(Q)$ is the effective number of bits required to transmit the dataset after using the bits back coding scheme to recover an extra $H(Q)$ bits encoded in a random choice of θ from $Q(\theta)$. It can be shown that F is an upper bound on $-\log P(\mathcal{D}|\mathcal{M})$, minimized when Q is as close as possible to the true posterior in a certain sense:

$$F(Q) = -\log P(\mathcal{D}|\mathcal{M}) + D(Q||P_{\theta|\mathcal{D},\mathcal{M}}), \tag{7.7}$$

where $D(\cdot||\cdot)$ is the Kullback–Leibler divergence, a measure of distance between two probability distributions, and non-negative. Variational Bayesian methods search the chosen space of variational distributions to find a Q that minimizes the free energy, and so, after the procedure is complete, we can use $F(Q)$ instead of the true evidence for model comparisons.

Thus, we come to the methodology we adopt for our subsequent modelling experiments: given a dataset and a number of candidate models, we fit each model

using variational Bayesian learning and use the variational free energy to compare them—the lower the free energy, the better the model. The variational free energy itself indicates how much information is needed to transmit the dataset using the bits back coding scheme with the optimal variational distribution Q.

7.4.2 Probabilistic Programming

While probabilistic modelling is a powerful technique for solving problems involving uncertainty, the translation of a model into a working computer program can become laborious if all the mathematical machinery has to be implemented from scratch. The development of *graphical models* or *Bayesian networks* (Pearl, 1988) made it possible to provide software libraries that require only a high level description of the model as a network of nodes and edges. However, the graph notation has limitations when it comes to models with an unknown or unbounded number of objects in the domain of interest. This motivated the development of more flexible languages for specifying and working with a broad class of probabilistic models, including grammars, Bayesian networks and probabilistic process models. The differences between language and music suggest that we will need such flexibility to go beyond the capabilities of linguistic grammar formalisms to model musical structure.

The idea behind probabilistic programming is to simplify this process by combining the flexibility of general purpose programming constructs (such as structured data types, recursion, functions, etc.) with probabilistic primitives in a high-level programming language, allowing the description of probabilistic models at a high level of abstraction and having the underlying framework provide the appropriate inference and learning functions *automatically*.

This is ideal for problems where some unobserved underlying structure is thought to give rise to observable consequences and an estimate of the underlying structure is desired. The probabilistic expression of the problem implies a resulting 'posterior' probability distribution over underlying structures and well-known general methods such as Markov chain Monte Carlo algorithms can be used to direct and give structure to the search over the solution space. It also means there is a clear separation between the exact statement of the problem and any approximations invoked to solve it.

Probabilistic programming is currently an active research topic in both artificial intelligence and cognitive modelling and marries the power of Bayesian methods with the flexibility and computational completeness of general programming languages.

Many probabilistic programming languages have been proposed and developed, including probabilistic Horn abduction (Poole, 1991, 1993), stochastic logic programming (Muggleton, 1996), PRISM (Sato and Kameya, 1997), Markov logic (Domingos and Richardson, 2007), IBAL (Pfeffer, 2001), Church (Goodman et al., 2008), and Hansei (Kiselyov and Shan, 2009).

PRISM (Sato and Kameya, 1997) supports efficient exact inference for a wide class of discrete-valued models, with PRISM equivalents of standard models such as hidden Markov models and PCFGs resulting in the equivalent of efficient standard

inference and learning algorithms for these models. PRISM also supports a limited form of Bayesian learning for a wide class of discrete-valued models including hidden Markov models and probabilistic context-free grammars (PCFGs), resulting in the equivalent of efficient standard inference and learning algorithms for these models. PRISM has been used to implement probabilistic grammars for natural languages and estimate their parameters (Sato et al., 2001) and for grammar induction using Variational Bayes for model selection (Kurihara and Sato, 2006). It has already been used for music modelling (Sneyers et al., 2006), and as the basis for a probabilistic constraint logic programming (CLP) system which was also used for music modelling (Sneyers et al., 2009).

MIT Church (Goodman et al., 2008) on the other hand, uses approximate, random-sampling based inference mechanisms and can support a wider class of models, at the expense of greater computational requirements as compared with exact inference algorithms when these exist. Alternative implementations of Church have also been developed that address exact inference for some programs (Stuhlmüller and Goodman, 2012; Wingate et al., 2011a,b).

Hansei (Kiselyov and Shan, 2009) is an example of the embedded DSL approach, where an existing language (OCaml) is augmented with probabilistic primitives. It inherits from the host language a sophisticated type system, higher order programming facilities, a module system and libraries, all of which aid in the development of robust and flexible models.

7.4.3 Building Probabilistic Grammars in PRISM

PRISM (PRogramming In Statistical Modelling) (Sato and Kameya, 1997) is an attractive approach for the development of probabilistic grammars for several reasons: grammars and interpreters can be encoded very simply by virtue of its inheritance of Prolog's definite clause grammar (DCG) notation and meta-programming facilities, it can mimic Earley's efficient chart parsing without work from the programmer (using tabling provided by the underlying B-Prolog implementation), and it includes an efficient implementation of variational Bayesian inference (Kurihara and Sato, 2006; Sato et al., 2008).

To give a flavour of how PRISM can be used to encode probabilistic grammars, we will implement the grammar (G3) from Sect. 7.3.1. The non-probabilistic form of the grammar can be implemented trivially in Prolog using the DCG notation as:

$ab \longrightarrow [a,b].$
$ab \longrightarrow [a],\ ab,\ [b].$

The non-terminal *S* is represented by the DCG goal named *ab* with zero arguments, referred to as *ab*//0. (Some notes about Prolog syntax are provided in the appendix.) Given this program, Prolog environments are capable of answering queries about whether or not a given string is grammatical, and of generating strings from the language:

```
?- phrase(ab,[a,a,b,b]).
yes
?- phrase(ab,[a,a,a,b]).
no
?- phrase(ab,X).
X = [a,b] ?;
X = [a,a,b,b] ?;
% etc...
```

To develop this into a PCFG, we must use a PRISM 'switch' to encode probability distributions over the alternative expansions of each non-terminal. This can be done in more than one way, but in this example, we will assign a label to each expansion rule. Then we will define a switch with the same name as the associated non-terminal and ranging over the labels of the expansion rules of that non-terminal. Finally, we modify the definition of *ab* to include an explicit random choice using the *msw*/2 PRISM primitive, followed by a call to DCG goal (::)//2 (written without parentheses using :: as a binary operator) which takes two parameters: the name of the non-terminal and the name of the chosen rule. The complete program is

:− **op**(500,*xfx*,::).
values(*ab*, [*stop*, *recurse*]).

$ab \longrightarrow \{msw(ab,Label)\}, ab::Label.$
$ab :: stop \longrightarrow [a, b].$
$ab :: recurse \longrightarrow [a], ab, [b].$

Once loaded, the grammar can be used in several ways: (a) for generatively sampling from the implied probability distribution over sentences; (b) for analytically computing the probability of observing a given sentence; and (c) for estimating the parameters from a list of samples using variational Bayesian learning (note that the variational free energy quoted by PRISM is the negative of the free energy as usually defined):

```
?- sample(ab(X)).
X = [a,a,b,b]
yes
?- prob(ab([a,a,b,b]),P).
P = 0.25
yes
?- set_prism_flag(learn_model,vb).
yes
?- get_samples(50,ab(X),Data), learn(Data).
  % ...various statistics about learning...
  Final variational free energy: -78.533991360
  % ...more statistics about learning...
Data = [ab([a,a,b,b]),ab([a,b]),ab([a,a,b,b])|...]
yes
```

Now that the grammar is embedded in a Turing complete probabilistic programming language, extending it beyond what is possible using a CFG is relatively straightforward. For example, we can write a probabilistic DCG for the language $a^n b^n c^n$ by

using a parameterized non-terminal *ab*//1 similar to *ab*//0, but returning the number[4] of repeats of *a* and *b*, and then adding a parameterized recursive non-terminal *c*//1 to produce that many copies of *c*:

$values(ab,\ [stop,\ \ recurse]).$

$abc \longrightarrow ab(N),\ c(N).$
$ab(N) \longrightarrow \{msw(ab,Label)\},\ ab(N) ::\ \ Label.$
$ab(zero) ::\ \ stop \longrightarrow [].$
$ab(succ(N)) ::\ recurse \longrightarrow [a],\ ab(N),\ [b].$
$c(zero) \longrightarrow [].$
$c(succ(N)) \longrightarrow [c].$

This example demonstrates the 'bidirectional' nature of Prolog logic variables: the argument to *ab*//1 is an *output* value, while the argument to *c*//1 is an *input*. However, if an output argument is partially or fully instantiated on input (representing a constraint on its value) sampling execution of the program may fail, resulting in a form of rejection sampling: the sequence of random choices involved in sampling a sentence from the grammar can result in the violation of a constraint, requiring that that sample be thrown away and a new one started. This example was written carefully to avoid such an eventuality, but in general, when creating a probabilistic version of a DCG, the possibility of failure results in a significant complication of the algorithms required for learning and inference (Sato et al., 2005).

For our purposes, failure can be avoided by structuring the process of rule expansion to prevent any probabilistic choice resulting in failure. Instead of using B-Prolog's built-in DCG compiler, we wrote a DCG meta-interpreter (in PRISM) taking production rules written in one of two forms:

$Head :: Label \implies Body.$
$Head :: Label \implies Guard\ |Body.$

Guard is an ordinary Prolog goal, which, combined with pattern matching against arguments in *Head*, determines the rule's applicability given a particular *Head*. *Body* is not allowed to fail if the rule is selected, meaning that during non-terminal expansion a label (from the collection of applicable rules) is sampled from its associated switch (defined automatically by the interpreter). The body of the selected rule is interpreted as an ordinary DCG (with the exception of using +*X* instead of [*X*] for the emission of terminal symbols and *nil* instead of [] for empty productions). Non-failing Prolog/PRISM goals can be included in braces and $X \sim S$ (equivalent to $\{msw(S,X)\}$) can be used to sample a PRISM switch *S*.

Consider the program in Fig. 7.7. *neigh* denotes the neighbour note rule which expands a non-terminal *i*(*P*) (where *P* is a pitch interval in semitones, but only when *P*=0). The random switch *step* provides a sample (between -4 and 4) resulting in the deviation *P1* to the neighbour note. *term* defines how the non-terminal *i*(*P*) produces the integer *P*: a terminal symbol.

[4] Represented algebraically where zero is *zero*, one is *succ(zero)*, two is *succ(succ(zero))*, etc.

7.5 Experiments with Probabilistic Music Models

The techniques described above can be used to develop a range of probabilistic models on various corpora of music. We summarize our experiments (Abdallah and Gold, 2014a,b) to compare the performance of several models on a corpus of monophonic melodies in Humdrum/Kern format. This comprised four datasets from the KernScores website at http://kern.humdrum.org: 185 Bach chorales (BWV 253–438 excluding BWV 279), which is the same dataset used by Gilbert and Conklin (2007), a larger set of 370 Bach chorales, and two 1000-element random subsets of the Essen folk song collection. In the description below, these datasets are referred to as *chorales*, *chorales371*, *essen1000a* and *essen1000b* respectively.

Several probabilistic models were implemented as PDCGs, of which six are described here. The models, with their short names, are:

p1gram	0^{th} order Markov model over pitches
p2gram	1^{st} order Markov model over pitches
phmm	1^{st} order hidden Markov model over pitches
i1gram	0^{th} order Markov model over intervals
i2gram	1^{st} order Markov model over intervals
gilbert2	Modified Gilbert and Conklin grammar

DCG rules for these models are presented in Fig. 7.5 (*p1gram*, *p2gram* and *phmm*), Fig. 7.6 (*i1gram* and *i2gram*), and Fig. 7.7 (*gilbert2*).

---

values(*nnum*, *X*) :− *numlist*(40,100,*X*).
values(*mc*(_), *X*) :− *values*(*nnum*,*X*).
values(*hmc*(_), *X*) :− *num_states*(*N*), *numlist*(1,*N*,*X*).
values(*obs*(_), *X*) :− *values*(*nnum*,*X*).

% start symbol for p1gram
s0 :: *tail* ⟹ *nil*.
s0 :: *cons* ⟹ *X*∼ *nnum*, +*X*, *s0*.

% start symbol for p2gram
s1(_) :: *tail* ⟹ *nil*.
s1(*Y*) :: *cons* ⟹ *X*∼ *mc*(*Y*), +*X*, *s1*(*X*).

% start symbol for phmm
sh(_) :: *tail* ⟹ *nil*.
sh(*Y*) :: *cons* ⟹ *X*∼ *hmc*(*Y*), *Z*∼ *obs*(*X*), +*Z*, *sh*(*X*).

Fig. 7.5 PDCGs for zeroth- and first-order Markov chains and first-order HMMs over pitch (encoded as MIDI note number). The number of states in the HMM is a parameter of the model

```
values(ival,   X) :- numlist(−20,20,X).
values(mc(_),  X) :- get_values(ival,X).

% start symbol for i1gram
s0 ::  tail  ⟹ +end.
s0 ::  cons  ⟹ X∼ ival, +X, s0.

% start symbol for i2gram
s1(_) ::  tail  ⟹ +end.
s1(Y) ::  cons  ⟹ X∼ mc(Y), +X, s1(X).
```

Fig. 7.6 PDCG for zeroth- and first-order Markov chains over pitch interval to next note in semitones

We encode pitch using MIDI note numbers and intervals as integers, in a sequence terminated by a Prolog *end* atom (notes are represented as a pitch interval in semitones to the following note, but the final note element has no subsequent pitch to encode). In richer models including other musical attributes such as duration or metrical stress, the attributes of the final note could be associated with the terminating symbol.

We modify Gilbert and Conklin's (2007) original grammar in two ways. Firstly, a different mechanism is used for introducing new intervals not captured by the elaboration rules: the *s* non-terminal, instead of being parameterized by the interval covered by the entire sequence, simply expands into a sequence of $i(P)$ non-terminals with the P sampled independently from the *leap* distribution. Secondly, because a note is represented by the pitch interval to the *following* note, the $i(P)$ :: *rep* rule has $i(0)$, $i(P)$ on the right-hand side instead of $i(P)$, $i(0)$ as in Gilbert and Conklin's grammar (see Fig. 7.3 for an illustration of this). Additionally, we chose the numerical ranges of the various operations (steps, leaps, and limits for passing and escape note introduction) as these were not specified by Gilbert and Conklin.

7.5.1 Results

The models obtained in the first part of the experiment can be used to obtain probabilistic parses of melodic sequences. Examples of these are shown in Figs. 7.8 and 7.9, illustrating two analyses of individual phrases from one of the Bach chorales (BWV 270) in the *chorales* dataset. In the first example, note that the high values of relative probability (RP) on the first two parses show that this is a relatively unambiguous parse. Note also that all four parses share the same first and last subtrees below the *s* non-terminal. In the second example, the lower values of RP indicate that this parse is much more ambiguous than that of the first phrase. Still, there is much common structure between these parses, so we may conclude that this grammar is fairly confident in its analysis of, for example, the final repeated note.

```
values(step,   X) :- numlist(−4,4,X).
values(leap,   X) :- numlist(−16,16,X).
values(passing(N),   Vals)  :−
    ( N>0 → M is N−1, numlist(1,M,I1)
    ; N<0 → M is N+1, numlist(M,−1,I1)
    ),
    maplist(N1,  (N1,N2),N2 is   N−N1,I1,Vals).

values(escape(N),   Vals)  :−
    ( N<0 → I1 = [1,2,3,4]
    ; N>0 → I1 = [−1,−2,−3,−4]
    ),
    maplist(N1,(N1,N2),N2 is   N−N1,I1,Vals).

% start symbol
s ::  last   ⟹ i(end).
s ::  grow ⟹ P∼ leap, i(P), s.

i(P)  ::  term   ⟹ +P.
i(P)  ::  rep    ⟹ i(0),   i(P).
i(P)  ::  neigh  ⟹ P=0 |
                     P1∼ step, {P2 is −P1}, i(P1),  i(P2).
i(P)  ::  pass   ⟹ abs_between(2,5,P)   |
                     (P1,P2)∼ passing(P), i(P1),  i(P2).
i(P)  ::  esc    ⟹ abs_between(1,16,P)  |
                     (P1,P2)∼ escape(P), i(P1),  i(P2).
abs_between(L,U,X) :− Y is  abs(X),  between(L,U,Y).
```

Fig. 7.7 Extract from a grammar (Abdallah and Gold, 2014b) modelled on Gilbert and Conklin's (Gilbert and Conklin, 2007), written in a DCG language defined in PRISM. *maplist/5* and *between/3* are standard B-Prolog predicates and *numlist(L,U,X)* is true when *X* is a list of consecutive integers from *L* to *U*. Code to initialize the switch probabilities and perform ancillary tasks is omitted

Figure 7.10 summarizes the performance of all the models per dataset over a range of parameter values. To normalize the comparison of differently-sized datasets, the variational free energy was divided by the total number of notes in the dataset to give the 'bits per note' (bpn), offering a sense of the amount of information required to encode each note under that model using the 'bits back' coding scheme (Honkela and Valpola, 2004)—the lower this is, the better the model. Data for *p1gram* (the zeroth-order Markov model over pitches) is not shown as its best-case performance (3.7 bpn on the chorales) was consistently lower than all the other models.

We observe that the relative performance of *i1gram* and *p2gram* is what we would expect, since a pair of consecutive pitches (a 2-gram) contains information about both pitch interval *and* absolute pitch, while the latter is not available to *i1gram*.

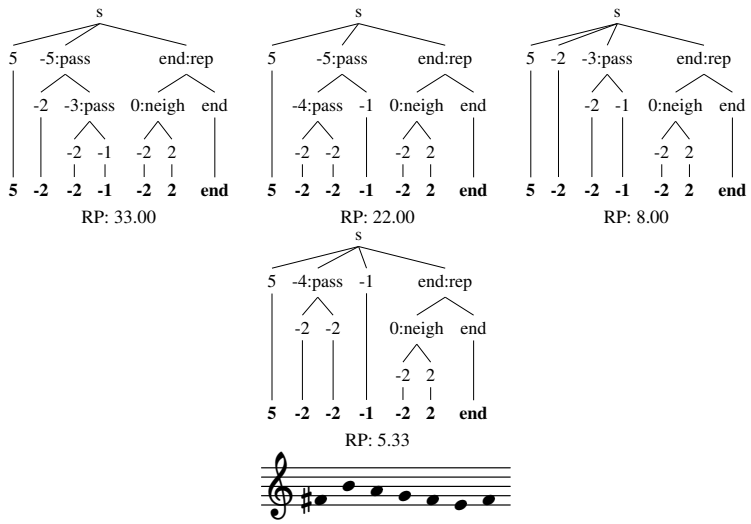


Fig. 7.8 Bach chorale BWV 270 ('Befiehl du deine Wege'), phrase #1, top 4 parses. Nodes representing non-terminals of the form *i(N)* are labelled with *N* and name of the rule used to expand them, while the terminal symbols are written in boldface aligned on the bottom row. Also, the top level sequence of *i(_)* non-terminals produced by recursive expansion of *s* using the *grow* rule have been collapsed into a single level below the *s* node, as suggested by Gilbert and Conklin (2007). The values labelled 'RP' are the posterior probabilities of each of the parses relative to that of the most likely parse not displayed, in this case, the 5th one

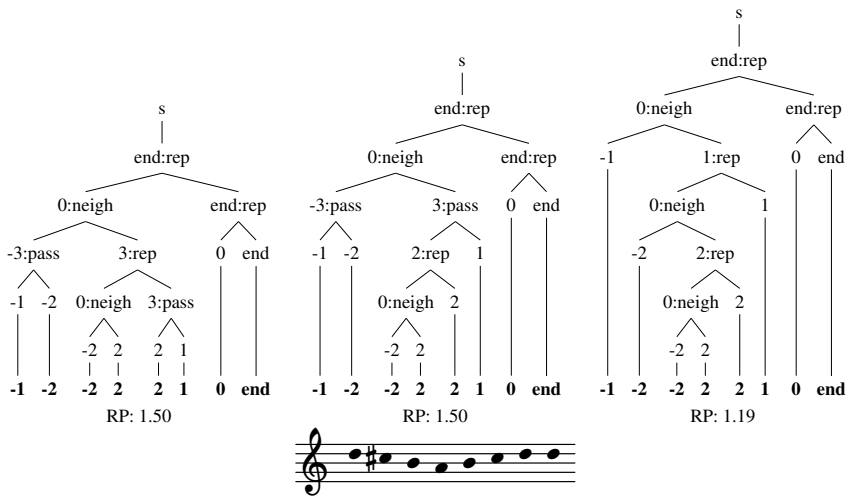


Fig. 7.9 Bach chorale BWV 270, phrase #3, top 3 parses. See Fig. 7.8 for an explanation of the RP values—here, the values are relative to the probability of the 4th most probable parse

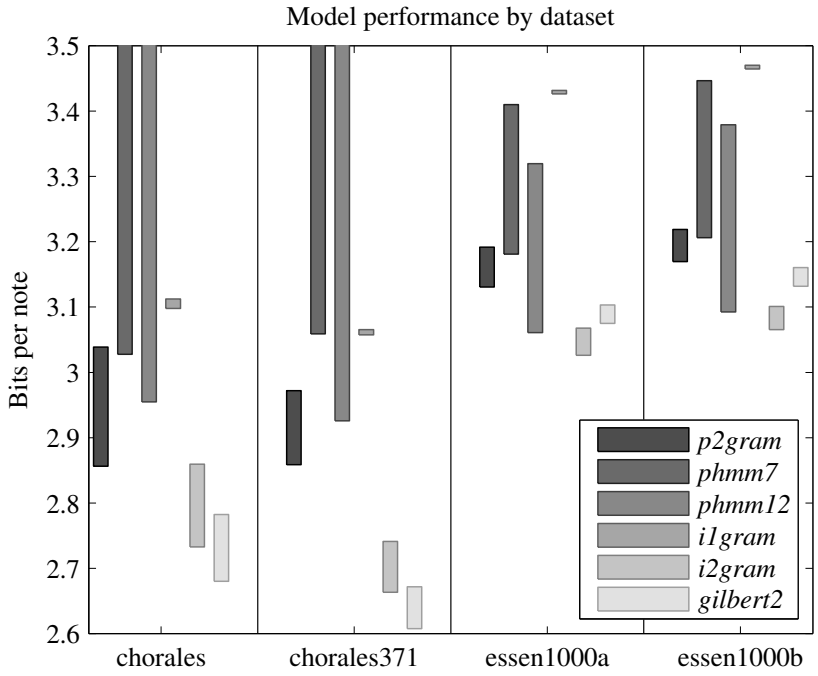


Fig. 7.10 Chart showing the overall performance of each model against each dataset (Abdallah and Gold, 2014b), measured in bits per note (lower bpn means better performance). For each model and dataset pair, the bar shows the range of values obtained using various parameter combinations. The chorales data extends beyond the top of the chart. Bars labelled as *phmmN* represent the HMMs with *N* states

We also note that the HMMs have the widest range of results, most likely due to the learning algorithm getting stuck in local optima (if this is the case, standard techniques (multiple restarts, simulated annealing) available within PRISM might be used to alleviate the problem).

Across all the datasets, the *gilbert2* and *i2gram* models perform consistently well, with the larger HMMs also performing competitively on the Essen collection. The *i2gram* model achieves approximately 2.68 bpn on the *chorales* dataset with its best parameter settings, comparable with the 2.67 bpn reported by Gilbert and Conklin (2007).

The grammar-based model gives the best fit on both chorales datasets, although it is beaten by the Markov model on the larger Essen datasets. The causes of this are unclear from the current data but investigation of the learned parameters would likely offer some explanation (some of these are illustrated in Fig. 7.11). One possibility is that the Essen dataset is relying proportionately more on the *s*::*grow* rule of the grammar to introduce new intervals. In the limit, this would reduce to a zeroth-order Markov model equivalent to *i1gram*, which, as we have seen, performs much worse than *i2gram*. Considering that higher-order Markov models would be likely to

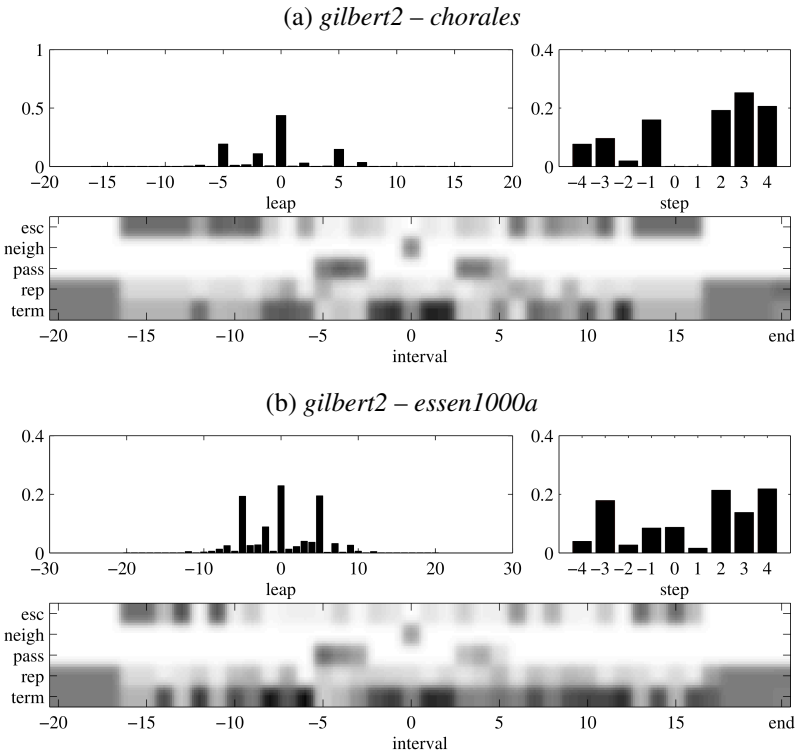


Fig. 7.11 Some of the switch distributions learned by fitting the *gilbert2* model to the *chorales* and *essen1000a* datasets. The *leap* distribution is used when introducing a new interval with the *s::grow* rule. The *step* distribution is used when introducing a neighbour note. The *intervals* greyscale images show the probability distributions over expansions of the non-terminal $i(N)$ for different values of N, bearing in mind that some rules are inapplicable for certain values. The Essen-fitted model appears to have a higher entropy *leap* distribution, makes less use of the *neigh* and *pass* rules, and has some curious structure in the *term* and *esc* probabilities for large intervals, all of which suggest that the grammar as designed is not such a good fit for it

perform better still, we might also conclude that designing *by hand* a probabilistic grammar (that is, the structure of the rules rather than the numerical probabilities which have been optimized in this experiment) capable of out-performing variable order Markov models would be a non-trivial task. Inducing grammar models over a corpus of works using probabilistic methods like those explored here is likely to make this more tractable and is desirable since variable-order Markov models cannot offer the explanatory power of a grammatical analysis.

7.6 Conclusions

This chapter has reviewed the principles, implementation, and application of grammar-based models in structural music analysis. It presented relevant aspects of the underpinning principles of information theory and probabilistic inference, showing how these can be applied to music.

The chapter concluded by showing that various probabilistic models of symbolic music can be implemented and applied to collections of Bach chorales and the Essen folk song collection (Abdallah and Gold, 2014b). A probabilistic grammar based on that of Gilbert and Conklin (2007) performed best (in the sense of allowing efficient representation of the collections) on the Bach chorales, but a more parsimonious parameterization of the same grammar performed worse than a first-order Markov model over pitch intervals.

In contrast to Markov models, however, grammars effect in a simple fashion the three factors identified above as common in music analysis: segmentation, classification and relation. Segmentation is embodied in the sequences of symbols on the right-hand side of rules, classification in the left-hand sides, and relation in the structure of the resultant parse tree. Putting this with the results of our experiments leads us to conclude that probabilistic grammars are a promising foundation for further developments in computational music analysis.

Acknowledgements This work was partially supported by the EPSRC CREST Platform grant [grant number EP/G060525/2] and the AHRC Digital Music Lab and ASyMMuS projects [grant numbers AH/L01016X/1 and AH/M002454/1]. Parts of this chapter (in particular, Fig. 7.5, Fig. 7.6, Fig. 7.7, Fig. 7.10 and the text of the Appendix) have previously appeared in Abdallah and Gold (2014a) and Abdallah and Gold (2014b), in the latter case used under the terms of the Creative Commons Attribution 3.0 Unported License (http://creativecommons.org/licenses/by/3.0/).

Supplementary Material Source code and data are available on request from Samer Abdallah (subject to copyright restrictions).

Appendix

Notes on Prolog Syntax Prolog code and data consist of *terms* built from a *functor* and a number of arguments; e.g., $a(10,b,X)$ is a term with a head functor $a/3$ (because it has three arguments), and arguments 10 (an integer), b (an atom or symbol), and X (a logic variable). Atoms and functor names start with a lower-case letter, while variable names start with an upper-case letter or underscore. A solitary underscore (_) stands for a variable whose value is not needed. Functors can be declared as prefix, infix, or suffix operators, for example, we declare $\sim$ to be an infix operator, so the head functor of $P\sim leap$ is $\sim/2$. The definite clause grammar (DCG) notation allows grammar rules to be defined using clauses of the form $Head \longrightarrow Body$, where $Head$ is a term and $Body$ is a list of one or more comma separated DCG goals. Within

the body, a list [*X*,*Y*,...] represents a sequence of terminals, while a term enclosed in braces {*Goal*} is interpreted as an ordinary Prolog goal.

References

Abdallah, S. A. and Gold, N. E. (2014a). Exploring probabilistic grammars of symbolic music using PRISM. In *Proceedings of the First Workshop on Probabilistic Logic Programming*, Vienna, Austria.

Abdallah, S. A. and Gold, N. E. (2014b). Comparing models of symbolic music using probabilistic grammars and probabilistic programming. In *Proceedings of the 2014 International Computer Music Conference (ICMC/SMC 2014)*, pages 1524–1531, Athens, Greece.

Abney, S. P. (1997). Stochastic attribute-value grammars. *Computational Linguistics*, 23(4):597–618.

Attneave, F. (1954). Some informational aspects of visual perception. *Pyschological Review*, 61(3):183–193.

Baker, J. K. (1979). Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America*, 65(S1):S132.

Barlow, H. B. (1961). Possible principles underlying the transformation of sensory messages. In Rosenblith, W. A., editor, *Sensory Communication*, pages 217–234. MIT Press.

Baroni, M., Maguire, S., and Drabkin, W. (1983). The concept of musical grammar. *Music Analysis*, 2(2):175–208.

Baxter, R. A. and Oliver, J. J. (1994). MDL and MML: Similarities and differences. Technical Report 207, Department of Computer Science, Monash University.

Bilmes, J. A. and Kirchhoff, K. (2003). Factored language models and generalized parallel backoff. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Companion Volume of the Proceedings of HLT-NAACL 2003, Short Papers, Volume 2*, pages 4–6. Association for Computational Linguistics.

Bod, R. (2006). An all-subtrees approach to unsupervised parsing. In *Proc. 21st Intl. Conf. on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872. Association for Computational Linguistics.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the First North American Chapter of the Association for Computational Linguistics Conference*, pages 132–139. Association for Computational Linguistics.

Chomsky, N. (1957). *Syntactic Structures*. Mouton de Gruyter.

Clark, S. and Curran, J. R. (2003). Log-linear models for wide-coverage CCG parsing. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 97–104.

Clark, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 16–23.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. Wiley.

Cox, R. T. (1946). Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13.

de Finetti, B. (1975). *Theory of Probability*. Wiley.

Domingos, P. and Richardson, M. (2007). Markov logic: A unifying framework for statistical relational learning. In Getoor, L. and Taskar, B., editors, *Introduction to statistical relational learning*, chapter 12, pages 339–372. MIT Press.

Dowe, D. L., Gardner, S., and Oppy, G. (2007). Bayes not bust! Why simplicity is no problem for Bayesians. *The British Journal for the Philosophy of Science*, 58(4):709–754.

Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Ebcioğlu, K. (1987). Report on the CHORAL project: An expert system for chorale harmonization. Technical Report RC 12628, IBM, Thomas J. Watson Research Center, Yorktown Heights, NY.

Forte, A. (1973). *The Structure of Atonal Music*. Yale University Press.

Frost, R. A. and Hafiz, R. (2006). A new top-down parsing algorithm to accommodate ambiguity and left recursion in polynomial time. *ACM SIGPLAN Notices*, 41(5):46–54.

Gilbert, É. and Conklin, D. (2007). A probabilistic context-free grammar for melodic reduction. In *International Joint Conference on Artificial Intelligence (Workshop on Artificial Intelligence and Music) (IJCAI-07)*, Hyderabad, India.

Goodman, J. (1998). *Parsing inside-out*. PhD thesis, Division of Engineering and Applied Sciences, Harvard University.

Goodman, N., Mansinghka, V., Roy, D., Bonawitz, K., and Tenenbaum, J. (2008). Church: a language for generative models. In *Proceedings of the Twenty-Fourth Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pages 220–229, Corvallis, Oregon. AUAI Press.

Granroth-Wilding, M. (2013). *Harmonic analysis of music using combinatory categorial grammar*. PhD thesis, School of Informatics, University of Edinburgh.

Granroth-Wilding, M. and Steedman, M. (2012). Harmonic analysis of jazz MIDI files using statistical parsing. In *Fifth International Workshop on Machine Learning and Music*, Edinburgh, UK.

Hamanaka, M., Hirata, K., and Tojo, S. (2006). Implementing "A Generative Theory of Tonal music". *Journal of New Music Research*, 35(4):249–277.

Hamanaka, M., Hirata, K., and Tojo, S. (2007). FATTA: Full automatic time-span tree analyzer. In *Proc. Intl. Computer Music Conference (ICMC2007), Copenhagen*, volume 1, pages 153–156.

Hinton, G. E. and van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning theory*, pages 5–13.

Hockenmaier, J. (2001). Statistical parsing for CCG with simple generative models. In *Association for Computational Linguistics, 39th Annual Meeting and 10th Conference of the European Chapter, Companion Volume to the Proceedings of the Conference: Proceedings of the Student Research Workshop and Tutorial Abstracts*, pages 7–12, Toulouse, France.

Hockenmaier, J. and Steedman, M. (2002). Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342.

Honkela, A. and Valpola, H. (2004). Variational learning and bits-back coding: An information-theoretic view to Bayesian learning. *IEEE Transactions on Neural Networks*, 15(4):800–810.

Johnson, M. (1995). Memoization in top-down parsing. *Computational Linguistics*, 21(3):405–417.

Johnson-Laird, P. N. (1991). Jazz improvisation: a theory at the computational level. In Howell, P., West, R., and Cross, I., editors, *Representing Musical Structure*, pages 291–325. Academic Press.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1998). An introduction to variational methods for graphical models. In Jordan, M. I., editor, *Learning in Graphical Models*, pages 105–161. MIT Press.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975). Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1):136–163.

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795.

Kassler, M. (1967). *A trinity of essays*. PhD thesis, Princeton University.

Kassler, M. (1976). The decidability of languages that assert music. *Perspectives of New Music*, 14/15:249–251.

Kassler, M. (1987). APL applied in music theory. *ACM SIGAPL APL Quote Quad*, 18(2):209–214.

Kirlin, P. B. (2014). *A probabilistic model of hierarchical music analysis*. PhD thesis, University of Massachusetts Amherst.

Kirlin, P. B. and Jensen, D. D. (2011). Probabilistic modeling of hierarchical music analysis. In *12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, pages 393–398.

Kiselyov, O. and Shan, C.-C. (2009). Embedded probabilistic programming. In Taha, W. M., editor, *Domain-Specific Languages*, pages 360–384. Springer.

Knill, D. C. and Pouget, A. (2004). The Bayesian brain: the role of uncertainty in neural coding and computation. *TRENDS in Neurosciences*, 27(12):712–719.

Knill, D. C. and Richards, W., editors (1996). *Perception as Bayesian inference*. Cambridge University Press.

Kurihara, K. and Sato, T. (2006). Variational Bayesian grammar induction for natural language. In *Grammatical Inference: Algorithms and Applications*, volume 4201 of *Lecture Notes in Artificial Intelligence*, pages 84–96. Springer.

Lambek, J. (1958). The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170.

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. MIT Press.

Li, M. and Vitányi, P. M. B. (2009). *An Introduction to Kolmogorov Complexity and its Applications*. Springer.

Lindblom, B. and Sundberg, J. (1970). *Towards a generative theory of melody*. Department of Phonetics, Institute of Linguistics, University of Stockholm.

MacKay, D. J. C. (1997). Ensemble learning for hidden Markov models. Technical report, Cavendish Laboratory, Cambridge University.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Marsden, A. (2005). Generative structural representation of tonal music. *Journal of New Music Research*, 34(4):409–428.

Marsden, A. (2007). Automatic derivation of musical structure: A tool for research on Schenkerian analysis. In *International Conference on Music Information Retrieval (ISMIR 2007)*, pages 55–58, Vienna, Austria.

Marsden, A. (2010). Schenkerian analysis by computer: A proof of concept. *Journal of New Music Research*, 39(3):269–289.

Marsden, A. (2011). Software for Schenkerian analysis. In *Proceedings of the 2011 International Computer Music Conference (ICMC2011)*, pages 673–676, Huddersfield, UK.

Martin-Löf, P. (1966). The definition of random sequences. *Information and Control*, 9(6):602–619.

Mavromatis, P. and Brown, M. (2004). Parsing context-free grammars for music: A computational model of Schenkerian analysis. In *Proceedings of the Eighth International Conference on Music Perception and Cognition*, pages 414–415, Evanston, IL.

Meyer, L. B. (1956). *Emotion and Meaning in Music*. University of Chicago Press.

Muggleton, S. (1996). Stochastic logic programs. In de Raedt, L., editor, *Advances in Inductive Logic Programming*, volume 32, pages 254–264. IOS Press.

Nattiez, J.-J. (1975). *Fondements d'une sémiologie de la musique*. Union Générale d'Editions.

Norvig, P. (1991). Techniques for automatic memoization with applications to context-free parsing. *Computational Linguistics*, 17(1):91–98.

O'Donnell, T. J., Tenenbaum, J. B., and Goodman, N. D. (2009). Fragment grammars: Exploring computation and reuse in language. Technical Report MIT-CSAIL-TR-2009-013, MIT.

Pachet, F. (2000). Computer analysis of jazz chord sequence: Is *Solar* a blues? In Miranda, E. R., editor, *Readings in Music and Artificial Intelligence*, pages 85–114. Routledge.

Pachet, F., Ramalho, G., and Carrive, J. (1996). Representing temporal musical objects and reasoning in the MusES system. *Journal of New Music Research*, 25(3):252–275.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Pereira, F. (1981). Extraposition grammars. *Computational Linguistics*, 7(4):243–256.

Pereira, F. C. and Warren, D. H. (1980). Definite clause grammars for language analysis—a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3):231–278.

Pereira, F. C. and Warren, D. H. (1983). Parsing as deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, pages 137–144.

Pfeffer, A. (2001). IBAL: A probabilistic rational programming language. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 733–740, Seattle, WA.

Poole, D. (1991). Representing Bayesian networks within probabilistic Horn abduction. In *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*, pages 271–278, Los Angeles, CA.

Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, 64(1):81–129.

Porter, H. H. (1986). Earley deduction. Technical report, Oregon Graduate Center.

Powers, H. S. (1980). Language models and musical analysis. *Ethnomusicology*, 24(1):1–60.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14(5):465–471.

Rohrmeier, M. (2006). Towards modelling harmonic movement in music: Analysing properties and dynamic aspects of pc set sequences in Bach's chorales. Technical Report DCRR-004, Darwin College, University of Cambridge.

Rohrmeier, M. (2011). Towards a generative syntax of tonal harmony. *Journal of Mathematics and Music*, 5(1):35–53.

Sato, T., Abe, S., Kameya, Y., and Shirai, K. (2001). A separate-and-learn approach to EM learning of PCFGs. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, pages 255–262, Tokyo, Japan.

Sato, T. and Kameya, Y. (1997). PRISM: a language for symbolic-statistical modeling. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, volume 2, pages 1330–1335, Nagoya, Aichi, Japan.

Sato, T., Kameya, Y., and Kurihara, K. (2008). Variational Bayes via propositionalized probability computation in PRISM. *Annals of Mathematics and Artificial Intelligence*, 54(1–3):135–158.

Sato, T., Kameya, Y., and Zhou, N.-F. (2005). Generative modeling with failure in PRISM. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 847–852, Edinburgh, UK.

Schenker, H. (1935). *Der freie Satz*. Universal Edition. (Published in English as E. Oster (trans., ed.) *Free Composition*, Longman, New York, 1979.).

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27(3–4):379–423,623–656.

Shieber, S. M. (1985). Criteria for designing computer facilities for linguistic analysis. *Linguistics*, 23(2):189–212.

Sidorov, K., Jones, A., and Marshall, D. (2014). Music analysis as a smallest grammar problem. In *Proceedings of the Fifteenth International Society for Music Information Retrieval Conference (ISMIR)*, pages 301–306, Taipei, Taiwan.

Smoliar, S. W. (1980). A computer aid for Schenkerian analysis. *Computer Music Journal*, 4(2):41–59.

Sneyers, J., Meert, W., and Vennekens, J. (2009). CHRiSM: Chance rules induce statistical models. In *Proceedings of the Sixth International Workshop on Constraint Handling Rules (CHR'09)*, pages 62–76, Pasadena, CA.

Sneyers, J., Vennekens, J., and De Schreye, D. (2006). Probabilistic-logical modeling of music. In van Hentenryck, P., editor, *Practical Aspects of Declarative Languages: 8th International Symposium, PADL 2006, Charleston, SC, USA, January 9–10, 2006, Proceedings*, volume 3819 of *Lecture Notes in Computer Science*, pages 60–72. Springer.

Steedman, M. (2001). *The Syntactic Process*. MIT Press.

Steedman, M. (2003). Formal grammars for computational musical analysis. In *INFORMS Annual Meeting*, Atlanta, GA.

Steedman, M. and Baldridge, J. (2011). Combinatory categorial grammar. In Borsley, R. D. and Börjars, K., editors, *Non-Transformational Syntax: Formal and explicit models of grammar*, pages 181–224. Wiley-Blackwell.

Steedman, M. J. (1984). A generative grammar for jazz chord sequences. *Music Perception*, 2(1):52–77.

Stolcke, A. (1995). An efficient probabilistic context-free parsing algorithm that computes prefix probabilities. *Computational Linguistics*, 21(2):165–201.

Stuhlmüller, A. and Goodman, N. D. (2012). A dynamic programming algorithm for inference in recursive probabilistic programs. *arXiv preprint arXiv:1206.3555*.

Temperley, D. (2007). *Music and Probability*. MIT Press.

Ulrich, J. W. (1977). The analysis and synthesis of jazz by computer. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence (IJCAI-77)*, pages 865–872, Cambridge, MA.

Wallace, C. S. (1990). Classification by minimum-message-length inference. In Akl, S. G., Fiala, F., and Koczkodaj, W. W., editors, *Advances in Computing and Information—ICCI'90*, volume 468 of *Lecture Notes in Computer Science*, pages 72–81. Springer.

Wallace, C. S. and Boulton, D. M. (1968). An information measure for classification. *The Computer Journal*, 11(2):185–194.

Wingate, D., Goodman, N., Stuhlmueller, A., and Siskind, J. M. (2011a). Nonstandard interpretations of probabilistic programs for efficient inference. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24 (NIPS 2011)*, pages 1152–1160, Granada, Spain.

Wingate, D., Stuhlmüller, A., and Goodman, N. D. (2011b). Lightweight implementations of probabilistic programming languages via transformational compilation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, pages 770–778, Ft. Lauderdale, FL.

Winograd, T. (1968). Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory*, 12(1):2–49.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Yust, J. (2009). The geometry of melodic, harmonic, and metrical hierarchy. In Chew, E., Childs, A., and Chuan, C.-H., editors, *Mathematics and Computation in Music: Second International Conference, MCM 2009, New Haven, CT, USA, June 19–22, 2009. Proceedings*, volume 38 of *Communications in Computer and Information Science*, pages 180–192. Springer.