Exploring probabilistic grammars of symbolic music using PRISM

Samer A. Abdallah* and Nicolas E. Gold

{s.abdallah,n.gold}@ucl.ac.uk
Department of Computer Science,
University College London

Abstract. In this paper we describe how we used the logic-based probabilistic programming language PRISM to conduct a systematic comparison of several probabilistic models of symbolic music, including 0^{th} and 1^{st} order Markov models over pitches and intervals, and a probabilistic grammar with two parameterisations. Using PRISM allows us to take advantage of variational Bayesian methods for assessing the goodness of fit of the models. When applied to a corpus of Bach chorales and the Essen folk song collection, we found that, depending on various parameters, the probabilistic grammars sometimes but not always out-perform the simple Markov models. Examining how the models perform on smaller subsets of pieces, we find that the simpler Markov models do out-perform the best grammar-based model at the small end of the scale.

1 Introduction

Over the last 20 years or so, probabilistic modelling has become a key element in many fields, including machine learning, cognitive science, computational linguistics and music informatics. In our work, we are interested in the use of probabilistic models for analysing the structure of music, from both musicological and perceptual points of view, and in particular, the possibility that probabilistic grammars and their generalisations might help us understand multi-scale temporal structure. In this paper, we investigate the use of probabilistic programming languages for the systematic exploration of probabilistic models of symbolic music, focussing in particular on how the approach can support Bayesian model selection criteria for comparing the relative merits of alternative models.

In the following sections, we will discuss some of the issues around probabilistic modelling and model selection (§ 2) and how the probabilistic programming language PRISM can support the development of probabilistic grammars (§ 3). We then review previous grammar-based models of symbolic music in § 4, and describe the particular models we implemented, along with the results of fitting them in a number of variations to a corpus of symbolic music, in § 5, finally drawing conclusions in § 6.

* This work is supported by the EPRSC CREST Platform Grant [grant number EP/G060525/2]. Data may be obtained from Samer Abdallah.

2 Probabilistic modelling and model selection criteria

Let $\mathcal{M}$ denote a parametric probabilistic model over a domain $\mathcal{X}$, such that given some parameter setting θ, the probability[1] of observing $x \in \mathcal{X}$ is $P(x|\theta, \mathcal{M})$. A common task is to estimate θ given a sequence of independent observations $\mathcal{D} = \{x_1, \ldots, x_T\}$, either because the parameters themselves are of interest, or to make predictions about future observations. A related problem is to choose the 'best' model from a set of candidates $\{\mathcal{M}_1, \mathcal{M}_2, \ldots\}$.

The method of *maximum-likelihood* estimation involves finding a point estimate of the parameters $\hat{\theta} = \arg\max_\theta P(\mathcal{D}|\theta, \mathcal{M})$ which maximises the *likelihood*:

$$P(\mathcal{D}|\theta, \mathcal{M}) = \prod_{t \in 1..T} P(x_t|\theta, \mathcal{M}). \tag{1}$$

It is well known that if the model is complex (in the sense of having a large parameter space) and the available dataset is small, then the resulting model can suffer from over-fitting, leading to poor generalisation. Conversly, an overly simple model might not be capable of capturing regularities in the data that a better model would use to be less surprised by the data. Some way of managing the trade-off is required. The notion is often expressed as Ockam's razor—the principle that, when there are several possible explanations for some phenomenon, the simplest is to be preferred.

Although other methods have been proposed, the Bayesian approach offers a theoretically and philosophically appealing solution to these problems [1]. This entails the consistent use of probabilities to represent uncertainty about *all* entities under consideration, including models and their parameters. For example, in considering a model $\mathcal{M}$ with parameters θ, we must represent our initial uncertainty about θ as a *prior* distribution $P(\theta|\mathcal{M})$. Then, on observing some data $\mathcal{D}$, we should update our 'belief state', giving a *posterior* distribution

$$P(\theta|\mathcal{D}, \mathcal{M}) = \frac{P(\mathcal{D}|\theta, \mathcal{M})P(\theta|\mathcal{M})}{P(\mathcal{D}|\mathcal{M})}, \tag{2}$$

which takes into account both the likelihood and the prior. The reason why this is the correct policy is that, in order to make the best possible prediction of a new datum x, given the model and observations so far, we must compute

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

The factorised form means that we can forget about the data $\mathcal{D}$ as long as we remember the posterior distribution $P(\theta|\mathcal{D}, \mathcal{M})$. The denominator in (2) is known as the *evidence* and can be expressed as

$$P(\mathcal{D}|\mathcal{M}) = \int P(\mathcal{D}|\theta, \mathcal{M})P(\theta|\mathcal{M}) \, \mathrm{d}\theta. \tag{3}$$

[1] We follow, for the sake of brevity, the convention of writing probability density and mass functions leaving the random variables X, Θ etc. implicit from the context.

When there are several candidate models $\mathcal{M}_1, \ldots, \mathcal{M}_N$, then the whole inferential process is lifted from distributions over parameters to distributions over models, with prior $P(\mathcal{M}_i)$ and posterior

$$P(\mathcal{M}_i|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{M}_i)P(\mathcal{M}_i)}{P(\mathcal{D})}. \tag{4}$$

We can see that the evidence $P(\mathcal{D}|\mathcal{M}_i)$ plays a key role in determining the relative plausibility of the models after the data has been observed. The committed Bayesian will use this posterior distribution to make predictions and decisions (this is *model averaging*), but forced to make a choice, perhaps because of limited computational resources, a reasonable policy, if the prior is flat, is to pick the model with the greatest evidence. If a model is too simple, then it may not assign sufficient probability to our given dataset $\mathcal{D}$, resulting in low evidence. If it is too complex, then it will assign significant probability to a greater variety of datasets and therefore less to any particular dataset, also reducing the evidence. Hence, using the evidence as the model selection criterion automatically invokes a form of Ockam's razor.

Representing uncertainty about model parameters and computing the evidence can be expensive operations computationally and approximations are often needed. For some models, *variational Bayesian* (VB) inference [2] can be good solution, combining an efficient representation of uncertainty about parameters with a tractable algorithm and delivering an estimate of the evidence. The algorithm minimises the *variational free energy* F, which is an upper bound on $-\log P(\mathcal{D}|\mathcal{M})$, and so, after the process is complete, F can be used instead of the true evidence for comparing models. For a given model, F is also a measure of the amount of information required to encode the data according to that model using the 'bits-back' coding scheme [3].

The choice of the variational free energy as a model selection criterion rests on our adoption of a Bayesian approach. Ultimately, this is something of a philosophical question; we refer the interested reader to some of the literature on these matters [4, 5].

3 Implementing probabilistic grammars in PRISM

Probabilistic programming languages aim to provide an environment where a wide variety of probabilistic models can be defined succinctly and in a relatively declarative way, by making available powerful constructs that are familiar from ordinary programming languages, such as recursion and structured data types. The system then provides functions for sampling and inference automatically, saving the programmer the effort of implementing special purpose algorithms for each model. To date, a variety of languages have been developed based on concepts from both logic programming [6–8] and functional programming [9–11].

The Prolog-based PRISM (PRogramming In Statistical Modelling) [7] is particularly attractive for the development of probabilistic grammars for several reasons: (a) it inherits Prolog's definite clause grammar (DCG) notation and

meta-programming facilities, so that grammars and grammar interpreters can be encoded very simply, (b) its use of tabling, provided by the underlying B-Prolog implementation, means that parsing mimics Earley's efficient chart parser without any special effort by the programmer [12], and (c) it provides an efficient implementation of variational Bayesian inference [13, 14]. PRISM has been used for implementing probabilistic gramars for natural languages and estimating their parameters [15] and for doing grammar induction using VB for model selection [13]. PRISM has also been used for music modelling, but using a hidden Markov model rather than a grammar-based model [16].

Implementing a PDCG in PRISM A context free grammar (CFG) consists of a set of *terminal* symbols, a set of *non-terminal* symbols, a set of production rules describing how each non-terminal can be rewritten as a sequence of terminals or non-terminals, and a distinguished non-terminal called the start symbol. A probabilistic CFG (PCFG) adds to this probability distributions over the possible expansions of each non-terminal.

A PCFG can be easily implemented in PRISM by writing an interpreter with probabilistic choice between alternative expansions for each non-terminal. In ordinary Prolog, DCG rules (non-terminals) can be parameterised and arguments used to represent linguistic phenomena such as number or tense agreement. As is well known, an appropriately written DCG can be used to generate strings as well as parse them, but turning a DCG into a *probabilistic* generative model presents some difficulties. The problem is that constraints represented by rule head unification, or embedded Prolog goals (wrapped in braces), can result in failure, and introducing failure into a probabilistic program results in a significant complication of inference and learning [17].

For our purposes, the problem of failure can be avoided by structuring the rule expansion process so that no probabilistic choice can result in failure. Each production rule is associated with a unique label, and written in one of two forms:

$$\mathit{Head} :: \mathit{Label} \implies \mathit{Body}.$$
$$\mathit{Head} :: \mathit{Label} \implies \mathit{Guard} \,|\, \mathit{Body}.$$

Guard is an ordinary Prolog goal, which, along with pattern matching against any arguments in *Head*, determines whether or not that rule is applicable given a particular instantiation of *Head*. If selected, *Body* is not allowed to fail. This means that when expanding a non-terminal, the labels of applicable rules can be collected and a label sampled from an associated distribution, represented as a PRISM 'switch'. The selected rule body is expanded as in an ordinary Prolog DCG, except that we use $+X$ instead of $[X]$ to emit a terminal symbol, and *nil* instead of [] for an empty production. Non-failing Prolog/PRISM goals embedded in braces are permitted, and for convenience, a PRISM switch S can be sampled using $X \sim S$, which is equivalent to the PRISM goal $msw(S, V)$.

All of this can be illustrated with reference to the program in fig. 2. The neighbour note rule (labelled *neigh*) applies to the expansion of a non-terminal $i(P)$, where P is a pitch interval in semitones, but only when P=0. The deviation *P1* to the neighbour note is sampled from a random switch called *step*, and is

between -4 and 4. The rule labelled *term* shows how a non-terminal $i(P)$ can produce a terminal symbol, in this case, the integer P.

Since PRISM switches are associated with learnable probability distributions, these can all be estimated from a given dataset. They also support the specification of a Dirichlet prior, which is used in variational Bayesian inference.

Parameterisation of label distributions Programs written in our DCG language define the permissible expansions for parameterised non-terminals, but not how the probability distributions over those expansions are parameterised. We implemented the following two approaches. The first, most straightforward solution is to treat each ground instance of each rule (that is, with definite values for all variables) as an independent PCFG rule with its own distribution that can be learned from examples. Such an encoding is sometimes referred to as a 'rule schema', and we will refer to it as the 'ground head' parameterisation.

An alternative is to collect together all rule heads with the same functor and arity *and* which lead to the same set of applicable expansions, and have them share a single probability distribution. For example, in fig. 2, all non-terminals of the form $i(P)$ where the absolute value of P is between 6 and 16 share the functor i with arity 1 and can be expanded using the rules *term*, *rep* and *esc*. Thus, under this 'head functor' parameterisation, they share the same probability distribution over those three expansion rules. This approach generally produces a model with fewer parameters, which could potentially result in better performance on small datasets.

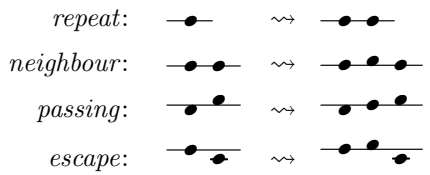
4 Modelling symbolic music

Probabilistic models of symbolic music can, to a large extent, be divided into two broad classes: those based on Markov (or n-gram) models, and those based on grammars. While fixed-order Markov models have problems avoiding over-simplicity for low n and over-fitting for high n, *variable* order Markov models have been used successfully to model monophonic melodic structure [18] and chord sequences [19].

Grammar-based models Although grammars have been applied in computational musicology since the late 1960s [20–22], resulting in some influential theories [23, 24], *probabilistic* grammar-based models of music are a relatively recent development. They can broadly be divided into models of *harmonic* sequence [25, 26] and models of *melodic* sequence [27–29]. We will focus on melodic models only in this paper.

Gilbert and Conklin [28] applied a PCFG to melodic structure analysis, drawing parallels between their approach and the hierarchical graphs of Schenkerian analysis [30], which attempts to account for the details of musical structure in terms of elaborations of simpler underlying forms, but in an unformalised way. Schenker's 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, Gilbert and Conklin represented the melody as a sequence of pitch *intervals* and were thus able to devise a CFG that embodies four types of melodic elaboration, which we can illustrate (following figure 1 of [28]) as follows:



Prior to this, Mavromatis and Brown [31] reported that they had been able to implement a non-probabilistic grammar (in Prolog) for Schenkerian analysis by adopting the same policy of elaborating sequences of intervals, though we have been unable to find any details about this grammar for a comparison with ours. Despite the problems that an interval-based encoding can lead to if the grammar is developed further, for example, to model durations or cover more types of elaboration [32], we will adopt it for the grammars tested in this paper.

Markov *vs* Grammar-based models This division between n-gram based models and grammar-based models echoes a similar division in computational linguistics, where probabilistic grammars parsing are widely used for tasks where a hierarchical syntactic analysis is required, such as language understanding, but as far as assigning probabilities to sentences is concerned, do not do as well as n-gram models. The issue was discussed by Brill *et al* [33], but while both n-gram and grammar based models have advanced since then, variable order Markov models continue to out-perform grammatical models (e.g. [34]). In computational musicology, the situation is less clear: whilst research activity around probabilistic grammars is increasing, a systematic framework for comparison across all models is yet to be established.

we propose that Bayesian model selection criteria, implemented using a probabilistic programming language that can support a wide variety of models, can provide such a framework, and that PRISM's support for variational Bayesian inference and efficient parsing makes it a good place to start. Although we have begun with small number of relatively simplistic models, the framework will support an exploration of increasingly sophisticated models in such a way that robust musicological conclusions can be drawn about the relative merits of such models as applied to a variety of musical corpora.

5 Experiments

We conducted an experiment to compare the performance of several models on a corpus of monophonic melodies. We used a collection of scores in Humdrum/Kern format, comprising three datasets, all available from the Kern Scores website at *http://kern.humdrum.org*. The first is a set of 185 Bach chorales, and is the dataset that was used by Gilbert and Conklin (henceforth referred to as G&C).

(a)

```
values(nnum, X) :- numlist(40,100,X).
values(tr(_),  X) :- get_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~tr(Y), +X, s1(X).
```

(b)

```
values(ival,  X) :- numlist(-20,20,X).
values(tr(_),  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~tr(Y), +X, s1(X).
```

Fig. 1. PDCG for 0^{th} and 1^{st} order Markov chains over (a) pitch (encoded as MIDI note number) and (b) pitch interval to the next note in semitones.

The second is a larger set of 370 Bach chorales. The third is the Essen folk song collection, containing 6174 scores. Because the full Essen collection was too large to process with the grammars on our test computer (an Apple laptop with 8 GB of memory), we took two random subsets of 1000 scores each. These datasets will be referred to as *chorales*, *chorales371*, *essen1000a* and *essen1000b* respectively.

Methods Six probabilistic models were implemented. The pitch-based models assume the input is encoded as a list of MIDI note numbers, while the interval-based models assume an input encoded as a list of integers followed by the atom *end*: each note is represented as the pitch interval to the *following* note, while the last note has no following note.[2] The models, with their short names and approximate numbers of parameters, are:

1. 0^{th} order Markov model over pitches (*p1gram*, 61).
2. 1^{st} order Markov model over pitches (*p2gram*, 3721).
3. 0^{th} order Markov model over intervals (*i1gram*, 41).
4. 1^{st} order Markov model over intervals (*i2gram*, 1681).
5. Modified G&C grammar, ground-head parameterisation (*gilbert2*, 233).
6. Modified G&C grammar, head-functor parameterisation (*gilbert3*, 102).

The DCG rules for all of these models are shown in fig. 1 (*p1gram* and *p2gram*, *i1gram* and *i2gram*), and fig. 2 (*gilbert2* and *gilbert3* share the same rules and differ only in their parameterisation). All the Markov models use the head functor parameterisation, so for *s0* and *s1*(_), there is only one distribution over the labels [*tail*, *cons*] which determines whether or not the chain is terminated or continues. We have omitted some supplementary code for intialising the switch probabilities and other ancillary tasks, as well as the DCG interpreter itself.

[2] When, in future, we extend the model to handle other musical dimensions such as duration, metrical strength, articulation etc., the attributes of the last note can be associated with the *end* symbol.

```
values(step,   X) :- numlist(−4,4,X).
values(leap,   X) :- numlist(−20,20,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  ⟹ passable(P)  |(P1,P2)~passing(P),  i(P1),  i(P2).
i(P) ::  esc   ⟹ escapable(P) |(P1,P2)~escape(P),  i(P1),  i(P2).

passable(P) :- abs_between(2,5,P).
escapable(P) :- abs_between(1,16,P).
abs_between(L,U,X) :- Y is abs(X), between(L,U,Y).
```

Fig. 2. A grammar based on Gilbert and Conklin's [28], 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.

Our version of G&C's grammar differs slightly from the original for reasons of which lack of space precludes a fuller discussion, though partly they stem from our decisions to represent each note by the pitch interval to the *following* note, rather than the preceeding note, as G&C do. Also, the numerical ranges of steps, leaps, and the limits for allowing passing or escape note introductions were chosen arbitrarily as this information was not given in G&C's paper.

A number of hyperparameters are available to control the Dirichlet priors over the probability distributions for each switch. These affect the shape of distributions, such as those over intervals in semitones or absolute pitches as MIDI note numbers, that might reasonably be expected to be weighted towards a central value, such as zero in the case of pitch intervals, or some central register for absolute pitches. The expected shape for such distributions is encoded as a weighted sum of binomial and uniform distributions. The hyperparameter *prior_weight* affects all distribution priors, and determines how much data is required to effectively override prior beliefs about the switch distribution. We omit further details about the hyperparameters due to lack of space, but, for completeness, list them below along with the sets to which their values belong.

$$
\begin{aligned}
\textit{prior\_weight} &: \{0.3,1,3,10,30\}. \quad \% \textbf{ all models} \\
\textit{prior\_shape} &: \textit{shape\_spec} \ \% \textbf{ for markov models} \\
\textit{leap\_shape} &: \textit{shape\_spec} \ \% \textbf{ for grammar models} \\
\textit{pass\_shape} &: \textit{shape\_spec} \ \% \textbf{ for grammar models} \\
\textit{shape\_spec} &= \{\textit{binomial}, \textit{uniform}, \textit{binomial+uniform}\} \\
&\quad \cup \{\textit{binomial} + K{*}\textit{uniform} \mid K \textit{ in } \{0.1,0.3\}\}
\end{aligned}
$$

To assess the effect of dataset size on model performance and to look for evidence of over-fitting of complex models to small datasets, subsets of pieces were extracted from each dataset. Subsets of varying sizes, from 1 to 30 were considered; in each case, 20 subsets were chosen at random from the full dataset. The same subsets were supplied to all models for training. Thus, each subset can be identified by a dataset name, a size from 1 to 30, and an index from 1 to 20.

Other implementation details To create a productive environment for conducting the experiments, we wrote a library for SWI Prolog that allows PRISM to be used as a sub-process. This enabled us to take advantage of existing SWI Prolog libraries and provided a degree of robustness, since the PRISM process can place heavy demands on the operating system and occasionally crashes. SWI Prolog, on the other hand, is very stable, and can be used to 'supervise' the PRISM process, restarting it and restoring its state when necessary.

We also implemented SWI Prolog libraries for reading musical scores in Humdrum/Kern format and for the persistent memoisation of computation results. The latter is available as an SWI Prolog 'pack' called *memo*. We plan to release the PRISM and Humdrum libraries in future, but in the meantime, they are available from the authors on request.

Results Fig. 3(a) shows a summary of the overall performace of each model against each dataset, over a range of hyperparameter values. In order to compare performance between datasets of different sizes, the variational free energy in each case is divided by the total number of notes in the dataset and displayed in 'bits per note' (bpn), to give a sense of the amount of information required to encode each note under that model, using the 'bits back' coding scheme [3]—the lower this is, the better the model. The data for *p1gram*, the 0^{th} order Markov model over pitches, is not shown, as in every case it was far below the other models, achieving, at best around 3.7 bpn.

Of those remaining, we will now consider the models in order of increasing performance, beginning with *i1gram*. This performs worse than *p2gram*, which is consistent with the the fact that a pair of consecutive pitches contains all the information in one pitch interval, plus information about absolute pitch which is not available to *i1gram*. Thus, the class of *i1gram* models is contained within the class of *p2gram* models. Proceeding onwards, there is some overlap between *p2gram* and *gilbert3* for the chorales, but the optimum for *p2gram* is better than the optimum for *gilbert3* for all datasets. For all datasets, the best two models are *i2gram* and *gilbert2*. The latter acheives approximately 2.68 bpn on the *chorales* dataset with the hyperparameter settings *prior_weight* : 3, *leap_shape* : *binomial* + 0.1∗*uniform* and, *pass_shape* : *binomial*. This is comparable with

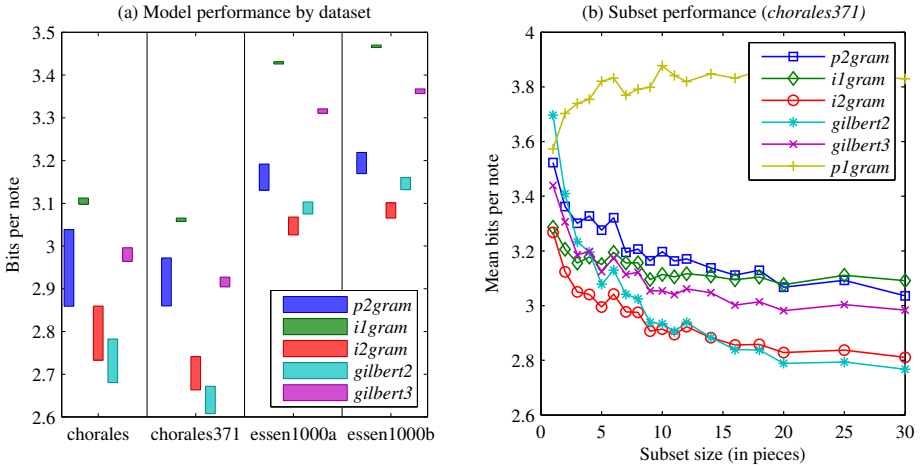


Fig. 3. (a) Overall performance, in bits per note, of each model against each dataset (lower is better). For each model/dataset pair, the bar shows the range values obtained over the various hyperparameter combinations described in the text. (b) Variation of model performance with dataset size. For each subset size K, 20 random subsets of size K were extracted from the *chorales* dataset. Then, for each model, the hyperparameter settings obtaining the best mean performance over the 20 subsets were determined, and this mean performance was plotted against the subset size. Though there is some variation due to the random subset selection mechanism, the graph clearly shows how the most complex model, *gilbert2*, performs the worst for the smallest datasets, even compared with *p1gram*.

the 2.67 bpn reported by G&C, but it should be born in mind that the variational free energy includes a model complexity penalty.

Fig. 3(b) shows how the models perform relative to each other on much smaller datasets obtained by extracting random subsets, in this case, from the *chorales* dataset. The graph shows how, for smaller datasets, the Markov models out-perform the grammar-based models, with *gilbert2* only emerging as best with datasets of 20 or more pieces.

6 Conclusions

We have shown how a variety of probabilistic models of symbolic music can be implemented in the probabilistic programming language PRISM, and applied these models to collections of Bach chorales and the Essen folk song collection.

While it is interesting to note that a model based on Gilbert and Conklin's grammar gave the best account of both chorales datasets, it was beaten by a first-order Markov model on the much larger Essen datasets. Bearing in mind that variable-order Markov models are likely to perform better still, this shows that designing by hand a probabilistic grammar capable of out-performing state-of-the-art Markov models is not a trivial task. The alternative, more parsimonious

head functor parameterisation of the same grammar performed quite badly in comparison with the other models. All of these observations show that is not safe to assume that a grammar-based model, though seemingly more sophisticated, will always out-perform even a first-order Markov model.

In an investigation of the models' performance on smaller datasets, we found that, as the size of the dataset was decreased, the 1^{st} Markov model over intervals began to out-perform the best grammar-based model on the chorales dataset, until, for very small datasets, the 0^{th} order Markov model over intervals performed best. This highlights the need to consider modelling assumptions carefully when dealing with small collections of music, which may often be the case when analysing certain stylistically related pieces, of which only a small number may exist.

Looking forward, it is likely that variable-order Markov models will improve significantly on the performance of *i2gram*, challenging us to develop better grammar based models. Overall, we hope that our framework and initial set of models will form a basis for such developments and a systematic exploration of probabilistic models of music and their use in analysis and the development of music theory.

References

1. Kass, R.E., Raftery, A.E.: Bayes factors. Journal of the American Statistical Association **90** (1995) 773–795
2. Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. In Jordan, M.I., ed.: Learning in Graphical Models. MIT Press, Cambridge, MA (1998) 105–161
3. Honkela, A., Valpola, H.: Variational learning and bits-back coding: an information-theoretic view to bayesian learning. Neural Networks, IEEE Transactions on **15** (2004) 800–810
4. Guyon, I., Saffari, A., Dror, G., Cawley, G.: Model selection: Beyond the bayesian/frequentist divide. The Journal of Machine Learning Research **11** (2010) 61–87
5. Gelman, A., Shalizi, C.R.: Philosophy and the practice of bayesian statistics. British Journal of Mathematical and Statistical Psychology **66** (2013) 8–38
6. Poole, D.: Representing Bayesian networks within probabilistic Horn abduction. In: Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc. (1991) 271–278
7. Sato, T., Kameya, Y.: PRISM: a language for symbolic-statistical modeling. In: Proc. 15th Intl. Joint Conf. on Artifical Intelligence (IJCAI). Volume 2. (1997) 1330–1335
8. Muggleton, S.: Stochastic logic programs. Advances in inductive logic programming **32** (1996) 254–264
9. Pfeffer, A.: IBAL: A probabilistic rational programming language. In: IJCAI, Citeseer (2001) 733–740
10. Goodman, N., Mansinghka, V., Roy, D.M., Bonawitz, K., Tenenbaum, J.: Church: a language for generative models. In: Uncertainty in Artificial Intelligence. (2008)
11. Kiselyov, O., Shan, C.C.: Embedded probabilistic programming. In: Domain-specific languages, Springer (2009) 360–384
12. Porter, H.H.: Earley deduction. Technical report, Oregon Graduate Center (1986)

13. Kurihara, K., Sato, T.: Variational Bayesian grammar induction for natural language. In: Grammatical Inference: Algorithms and Applications. Springer (2006) 84–96
14. Sato, T., Kameya, Y., Kurihara, K.: Variational Bayes via propositionalized probability computation in prism. Annals of Mathematics and Artificial Intelligence **54** (2008) 135–158
15. Sato, T., Abe, S., Kameya, Y., Shirai, K.: A separate-and-learn approach to EM learning of PCFGs. In: NLPRS. (2001) 255–262
16. Sneyers, J., Vennekens, J., De Schreye, D.: Probabilistic-logical modeling of music. In: Practical Aspects of Declarative Languages. Springer (2006) 60–72
17. Sato, T., Kameya, Y., Zhou, N.F.: Generative modeling with failure in PRISM. In: IJCAI. (2005) 847–852
18. Pearce, M.T.: The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition. PhD thesis, Department of Computing, City University, London (2005)
19. Yoshii, K., Goto, M.: A vocabulary-free infinity-gram model for nonparametric Bayesian chord progression analysis. In: ISMIR. (2011) 645–650
20. Winograd, T.: Linguistics and the computer analysis of tonal harmony. Journal of Music Theory **12** (1968) 2–49
21. Kassler, M.: A trinity of essays. PhD thesis, Princeton University (1967)
22. Lindblom, B., Sundberg, J.: Towards a generative theory of melody. Department of Phonetics, Institute of Linguistics, University of Stockholm (1970)
23. Steedman, M.J.: A generative grammar for jazz chord sequences. Music Perception **2** (1984) 52–77
24. Lerdahl, F., Jackendoff, R.: A Generative Theory of Tonal Music. MIT Press, Cambridge, MA (1983)
25. Rohrmeier, M.: Towards a generative syntax of tonal harmony. Journal of Mathematics and Music **5** (2011) 35–53
26. Granroth-Wilding, M.: Harmonic Analysis of Music Using Combinatory Categorial Grammar. PhD thesis, School of Informatics, University of Edinburgh (2013)
27. Bod, R.: Probabilistic grammars for music. In: Belgian-Dutch Conference on Artificial Intelligence (BNAIC). (2001)
28. Gilbert, É., Conklin, D.: A probabilistic context-free grammar for melodic reduction. In: International Workshop on Artificial Intelligence and Music, IJCAI-07, Hyderabad, India. (2007)
29. Kirlin, P.B., Jensen, D.D.: Probabilistic modeling of hierarchical music analysis. Analysis **1** (2011) 15
30. Schenker, H.: Der freie Satz. Universal Edition, Vienna (1935) (Published in English as E. Oster (trans., ed.) *Free Composition*, Longman, New York, 1979.).
31. Mavromatis, P., Brown, M.: Parsing context-free grammars for music: A computational model of Schenkerian analysis. In: Proc. 8th Intl. Conf. on Music Perception and Cognition, Evanston, USA. (2004) 414–415
32. Marsden, A.: Schenkerian analysis by computer: A proof of concept. Journal of New Music Research **39** (2010) 269–289
33. Brill, E., Florian, R., Henderson, J.C., Mangu, L.: Beyond n-grams: Can linguistic sophistication improve language modeling? In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1, Association for Computational Linguistics (1998) 186–190
34. Wood, F., Archambeau, C., Gasthaus, J., James, L., Teh, Y.W.: A stochastic memoizer for sequence data. In: Proceedings of the 26th Annual International Conference on Machine Learning, ACM (2009) 1129–1136