

Multiple Viewpoint Systems: Time Complexity and the Construction of Domains for Complex Musical Viewpoints in the Harmonisation Problem

Raymond P. Whorley¹, Geraint A. Wiggins²,
Christophe Rhodes¹, and Marcus T. Pearce²

¹ Department of Computing
Goldsmiths, University of London
New Cross, London SE14 6NW, UK

² School of Electronic Engineering and Computer Science
Queen Mary, University of London
Mile End Road, London E1 4NS, UK

Abstract. We discuss the problem of automatic four-part harmonisation: given a soprano part, add alto, tenor and bass in accordance with the compositional practices of a particular musical era. In particular, we focus on the development of representational and modelling techniques, within the framework of *multiple viewpoint systems* and *Prediction by Partial Match* (PPM), for the creation of statistical models of four-part harmony by machine learning. Our ultimate goal is to create better models, according to the information theoretic measure *cross-entropy*, than have yet been produced. We use multiple viewpoint systems because of their ability to represent both surface and underlying musical structure, and because they have already been successfully applied to melodic modelling. To allow for the complexities of harmony, however, the framework must be extended; for example, we begin by predicting complete chords, and then extend the framework to allow part by part prediction. As the framework is extended and generalised, the viewpoints become more complex. This article discusses matters related to viewpoint *domains* (alphabets), such as their size and consequent effect on run time; and presents methods for their reliable construction. We also present an empirical analysis of the time complexity of our computer implementation.

1 Introduction

We are attempting to solve by computational means the problem of automatic four-part harmonisation: given a soprano part, add alto, tenor and bass in accordance with the compositional practices of a particular musical era. *Multiple viewpoint systems* were introduced by Conklin and Cleary (1988), who used them to hand-craft some simple statistical models of polyphony. Their application to melody (Conklin, 1990; Conklin and Witten, 1995) was more successful, culminating in the work of Pearce and Wiggins (2006, 2012), who demonstrated that

statistical models such as these can successfully model listeners' expectations. Conklin (2002) used viewpoints (although not multiple viewpoint systems *per se*) to discover repeatedly occurring harmonic patterns in a corpus of music. In this article we are concerned with the extension of this framework to the harmonic domain, given its promising melodic results and obvious potential in the field of harmony. In particular, we are focusing on the development of representational and modelling techniques, within the framework of multiple viewpoint systems and *Prediction by Partial Match* (PPM, Cleary and Witten, 1984), for the creation of statistical models of four-part harmony by machine learning. Our aim is to create better statistical models of harmony, according to the information theoretic measure *cross-entropy* (see §3.7), than have so far been produced. A *primitive viewpoint* describes a single feature of a sequence of musical objects; in this case, the objects are concurrently sounding notes. An example of a viewpoint employed in the modelling of music is `Pitch`. In this and earlier related research (Conklin and Witten, 1995; Pearce and Wiggins, 2004), the set of valid elements (or symbols, or values) for this viewpoint, its *domain*, comprises pitch values represented as MIDI numbers. This article addresses three fundamental issues relating to domains, illustrating the fact that extending the multiple viewpoint framework to harmony is non-trivial. One issue is the size of the `Pitch` domain; to predict all possible SATB note combinations, the domain is so large that prediction is excessively slow (we carry out a time complexity analysis in §9). We discuss principled ways of reducing the domain size, such that reasonable running times are made possible. Another issue is whether or not a domain can be fixed at the beginning of the prediction process, such that it can be used unchanged at all positions in a musical sequence. We explain why this is not, in general, possible. Finally, we see that the construction of domains for *linked viewpoints* (viewpoints formed by combining two or more primitive viewpoints) is far from straightforward. We show how to reliably construct linked viewpoint domains of various complexities, and explain why this is important.

§3 provides a brief description of multiple viewpoint systems and associated modelling techniques; §4 gives a very short description of our corpus and test data; §5 introduces domain-related issues by considering melodic viewpoint domains; and §§6–8 deal with the viewpoint domains of versions 1, 2 and 3 of our framework for modelling harmony respectively (having first introduced these versions). A procedure for the construction of complex viewpoint domains is outlined in the latter section, along with a detailed example. This procedure is formally presented as algorithms in Appendix A, which includes a line-by-line description following the same detailed example. There is an analysis of the time complexity of our computer model in §9, and in §10 we examine four hymn tune harmonisations automatically generated by our best model to date. Finally, we state our conclusions and indicate the direction of our future work in §11; but first, we review the relevant literature.

2 Previous Approaches to the Modelling of Harmony

There have been many attempts to model musical composition, or aspects of it, using AI techniques. Modelling four-part harmony is difficult: even if we were to encode an entire treatise on harmony into a computer system, the harmony produced by that system is unlikely to match that of an expert musician, who has the advantage of having actually experienced music; that is, music always goes beyond the rules that an analyst can synthesise from it. What we would ideally like to do is to find a way to model harmony such that the system is capable of producing consistently good harmony in any specified style. Previous approaches to the modelling of harmony are briefly outlined below.

Constraint-based Methods Ebcioğlu (1988) describes CHORAL, a rule-based *expert system* (using first-order logic), for the four-part harmonisation of chorale melodies in the style of J. S. Bach. The predicates are grouped in such a way that the music is observed from *multiple viewpoints* (the multiple viewpoints here are different from those used by Conklin, 1990, not least because no statistical modelling is involved; although they do appear to have been the original inspiration). Pachet and Roy (1998) focus on the exploitation of part-whole relations; for example, the relationship between note and chord. They use object-oriented techniques to build a structured representation of music, called the MusES system (Pachet et al., 1996), with which knowledge of harmony can be expressed.

Evolutionary Methods Phon-Amnuaisuk et al. (1999) use a *genetic algorithm* to generate homophonic four-part harmony for given melodies. Phon-Amnuaisuk and Wiggins (1999) compare the performance of this system with that of a rule-based system into which the same amount of musical knowledge has been encoded. An objective evaluation of the harmonisations clearly demonstrates the superiority of the rule-based approach, which is thought to be due to additional implicit knowledge in the form of a structured search mechanism.

Connectionist Methods Hild et al. (1992) have created a *neural network* system called HARMONET, which is capable of producing four-part harmonisations of chorale melodies in the style of J. S. Bach. A key feature of the architecture of this system is that the overall harmonisation task is divided into a number of subtasks.

Statistical Modelling Clement (1998) uses first-order *N-gram* models (see §3.3) to demonstrate that two distinct (though artificial) styles of harmonic progression can be learned. Ponsford et al. (1999) improve on this by using N-grams of up to third-order to create statistical models of underlying harmonic movement from a corpus of seventeenth century French *sarabandes*. Biyikoğlu (2003) uses N-gram models to study harmonic syntax and the relationship between melody and harmony. Allan (2002) suggests that melody can be properly taken into consideration in statistical models by using N-gram models in which a context of melody notes is added to the historical harmonic context.

Assayag et al. (1999) describe a *dictionary-based* approach to the machine learning of music. The models they construct are similar to N-gram models in many respects. To facilitate the modelling of polyphony, music is transformed into a sequence of discrete events using the *full expansion* technique. Special symbols indicate notes which continue to sound from event to event.

Allan (2002) describes a harmonisation model which uses *hidden Markov Models* (HMMs). Following Hild et al. (1992), the overall harmonisation task is divided into three subtasks (Allan and Williams, 2005, merge the first two subtasks in later work). The *Viterbi algorithm* (Viterbi, 1967) can be used to find the globally most probable sequence of hidden states from an observed event sequence. The model was evaluated by calculating cross-entropies (see §3.7).

Raphael and Stoddard (2003, p. 181) propose a *Bayesian network* model for harmonic analysis which “regards the data as a collection of voices where the evolution of each voice is conditionally independent of the others, given the harmonic state.”

3 The Multiple Viewpoint Framework

3.1 Motivation for the Use of Multiple Viewpoints to Model Harmony

Ebcioğlu (1988) was responsible for one of the more successful efforts at modelling harmony, producing an expert system which incorporated explicit hard-coded rules to encapsulate musical knowledge. There are problems associated with this sort of rule-based approach, however. Firstly, for any given style of harmonisation, there are many general rules with lots of exceptions. Formulating a theory of a style by creating a model in this way, therefore, requires expertise, is extremely time consuming, and results in a theory which is likely to be incomplete. Secondly, to formulate theories of different harmonic styles (*e.g.*, those of Tallis, Bach and Mozart), it is necessary to infer different sets of general rules and exceptions, which multiplies the time expended by the expert.

Machine learning has the potential to circumvent the above problems; the computer learns for itself how to harmonise in a particular style by constructing a model of harmony from a corpus of stylistically homogeneous music. Once the machine learning program has been written, and is demonstrably working satisfactorily, it should be able to model the style of harmony of any corpus presented to it. Providing that the resulting model is able to consistently generate harmonisations which are indistinguishable (by an expert; Pearce and Wiggins, 2007) in style from those in the corpus, then it is a theory of that style, containing structure equivalent to rules and exceptions.

Later work by Allan and Williams (2005) emphatically demonstrated the potential of a statistical machine learning approach: they used HMMs to create a model of four-part harmony from a corpus of chorale melodies harmonised by J. S. Bach. Harmonisations generated by their model convey a definite flavour of Bach’s style, and some of them are very good; but Allan and Williams (2005)

themselves recognise that better modelling, possibly within the framework of *Conditional Random Fields* (CRFs, Lafferty et al., 2001), could improve their harmonisations. They say that whereas their system represents chords simply as sets of intervals, the CRF framework would allow the inclusion of additional features such as position in bar and key signature.

An established means of representing music which, when combined with machine learning and modelling techniques, is able to incorporate such features (and many others beside), is a framework called *multiple viewpoint systems*, which was formulated in seminal work by Conklin (1990) and Conklin and Witten (1995). This not only directly represents basic musical attributes like duration and pitch, but also derived attributes such as intervals. The use of these derived attributes necessarily introduces a certain amount of low-level musical knowledge, which researchers such as Dixon and Cambouropoulos (2000) have found to be beneficial. Pearce (2005) has successfully used this framework to produce cognitive models of melodic expectancy, which in turn has led to models being developed to predict phrase segmentation (Pearce et al., 2008, 2010). On the basis of its past success in the musical sphere, we consider the multiple viewpoint framework to be the ideal representation scheme for the modelling of harmony.

Having motivated the representation, we now move on to the modelling. *Prediction by Partial Match* (PPM, Cleary and Witten, 1984) uses an *escape method* to create viewpoint prediction probability distributions from N-gram models of different order, starting with the highest order and backing off to lower orders (see §3.3 for more details). This is where viewpoint domains come in: if a distribution does not contain all possible predictions by the time the 0th-order model has been incorporated, it is completed by backing off to a uniform distribution comprising all of the members of the relevant domain.

We consider that the flexibility which the multiple viewpoint framework affords, both in terms of the number of viewpoints (and their combination) and the ability to utilise different context sizes, gives it an advantage over methods using a single or very few “viewpoints,” and/or methods using a single context size. HARMONET (Hild et al., 1992) effectively uses four “viewpoints” linked together in a predetermined way to form three different context sizes (in the advanced system). Similarly, the Bayesian network approach described in §2 (Raphael and Stoddard, 2003) uses a very limited number of “viewpoints,” and the models are no larger than 2nd-order. The multiple viewpoint framework, on the other hand, allows the use of a viewpoint selection algorithm to optimise the model by choosing which of many viewpoints should be in the system, and how they should be linked (see §3.6).

Having motivated the use of the multiple viewpoint framework, we first describe multiple viewpoint systems as applied to the modelling of melody.

3.2 Viewpoint Types

A *type* τ is an attribute or property of an *event* in a sequence (here a note in a melodic sequence). *Basic types* are the fundamental attributes of a note that are predicted or given. One that is predicted is `Pitch`, which is represented as

MIDI numbers in the current implementation; for example, the note G4 has a **Pitch** value of 67. Another is **Duration**, which is represented using units of one ninety-sixth of a semibreve (whole note); for example, a crotchet (quarter note) has a **Duration** value of 24. Some basic types that are given are: the start-time of a note (**Onset**); the number of flats or sharps in the key signature (**KeySig**), where for example 3 flats has a **KeySig** value of -3 , and 1 sharp a value of 1; whether the melody is in a major or a minor key (**Mode**), where major has a **Mode** value of 0, and minor a value of 9; and whether a note is at the beginning or end of a phrase, or otherwise (**Phrase**), which has possible values 1 (first note in phrase), -1 (last note in phrase) and 0 (otherwise). $[\tau]$ denotes the set of valid values, symbols or elements for a viewpoint type, which in the latter case is $\{-1, 0, 1\}$; this is called the *syntactic domain*, hereafter simply referred to as the *domain*. There are several more basic types; but if we assume that this is the full set of basic types, then the set of representable events (notes in the case of melody), called the *event space*, is

$$[\text{Onset}] \times [\text{Duration}] \times [\text{Pitch}] \times [\text{KeySig}] \times [\text{Mode}] \times [\text{Phrase}].$$

Derived types such as sequential pitch interval (**Interval**) and sequential duration ratio (**DurRatio**) are derived from, and can therefore predict, basic types (in this case **Pitch** and **Duration** respectively). **Interval** is the difference in pitch between two consecutive notes, measured in semitones (ascending intervals have positive values); and **DurRatio** is the ratio of the duration of a note to the duration of the immediately preceding note. Derived types are not necessarily defined at all positions in a sequence, with both **Interval** and **DurRatio** being undefined at the first note. An example of a viewpoint type derived from given attributes is the tonic of the relevant major or minor scale (**Tonic**), which is derived from **KeySig** and **Mode**; for example, when **KeySig** is 1 and **Mode** is 0 (major key with 1 sharp), **Tonic** is G. This is used in the determination of the pitch interval from the tonic (**ScaleDegree**, measured in semitones), which is primarily derived from, and can therefore predict, **Pitch**.

Threaded types are defined only at certain positions in a sequence, determined by Boolean test viewpoints such as **FirstInBar**; for example, **ScaleDegree** \ominus **FirstInBar** has a defined **ScaleDegree** value only for the first note in a bar. Threaded types are able to model longer range dependencies. Basic, derived, and in this research threaded types are collectively known as *primitive types*. Note that Conklin and Witten (1995) included a *timescale* with each threaded type, which, as Pearce (2005) pointed out, effectively meant that it was a linked type (see below). The timescale was required to predict attributes which are assumed to be given in this research.

A *linked type*, or *product type*, is the conjunction of two or more primitive viewpoints; for example, **DurRatio** \otimes **Interval**. If any of the constituent viewpoints are undefined, then the linked viewpoint is also undefined. The *type set* of a type τ is denoted by $\langle \tau \rangle$. This set comprises the basic types that a viewpoint is able to predict (*i.e.*, the basic types from which it is derived); therefore

$$\langle \text{DurRatio} \otimes \text{Interval} \rangle = \{\text{Duration}, \text{Pitch}\}.$$



Basic:	Pitch	67	67	71	69	67	69	69	71
	Duration	48	48	48	48	48	48	48	48
Derived:	Interval	\perp	0	4	-2	-2	2	0	2
	ScaleDegree	0	0	4	2	0	2	2	4
Threaded:	ScaleDegree \ominus FiB	\perp	0	\perp	\perp	\perp	2	\perp	\perp
Linked:	DurRatio \otimes Interval	\perp	$\langle 1, 0 \rangle$	$\langle 1, 4 \rangle$	$\langle 1, -2 \rangle$	$\langle 1, -2 \rangle$	$\langle 1, 2 \rangle$	$\langle 1, 0 \rangle$	$\langle 1, 2 \rangle$

Fig. 1. First phrase of hymn tune *Luther’s Hymn (Nun freut euch)* adapted from Vaughan Williams (1933), with sequences of viewpoint elements underneath. Viewpoint **FirstInBar** has been abbreviated to **FiB**. The symbol \perp means that a viewpoint is undefined.

Figure 1 shows the first phrase of hymn tune *Luther’s Hymn (Nun freut euch)* adapted from Vaughan Williams (1933), with example sequences of viewpoint elements underneath. See Table 1 for the subset of viewpoint types referred to in this article and their meanings. Other than for threaded types, viewpoint nomenclature in this article follows, and where necessary extends, that of Pearce and Wiggins (2006). See also Conklin and Witten (1995) and Pearce (2005) for more details.

3.3 N-gram Models

So far, *N-gram models* (or *context models*), which are Markov models employing subsequences of N symbols, have been the modelling method of choice when using multiple viewpoint systems (Conklin and Witten, 1995; Pearce, 2005). The idea is to predict the next symbol in a sequence by taking account of the immediately preceding defined symbols. The *transition probability* of the N^{th} symbol, the *prediction*, depends only upon the previous $N - 1$ symbols, the *context*. The number of symbols in the context is the *order* of the model. Transition probabilities are determined by *maximum likelihood estimation*: see Manning and Schütze (1999) for more details.

A *viewpoint model* is a weighted combination of various orders of N-gram model of a particular viewpoint type. The N-gram models can be combined by, for example, *Prediction by Partial Match* (PPM, Cleary and Witten, 1984). PPM makes use of a sequence of models, which we call a *back-off sequence*, for context matching and the construction of complete prediction probability distributions (*i.e.*, containing all possible predictions, however improbable). The back-off sequence begins with the highest order model, proceeds to the second-highest order, and so on. An *escape method* determines weights for prediction probabilities at each stage in this sequence, which are generally high for predictions appearing in high-order models, and *vice versa*. If necessary, a probability distribution is completed by backing off to a uniform distribution (where the probability mass is divided equally between all elements in the relevant domain). In this research,

Table 1. Viewpoint types τ referred to in this article.

τ	Meaning	Derived from
Basic		
Onset	start-time of event	
Duration	duration of event	
Cont	event continuation, or otherwise	
Pitch	chromatic pitch	
KeySig	number of flats or sharps	
Mode	major or minor key	
Phrase	event at start/end of phrase, or not	
Derived		
DurRatio	sequential duration ratio	Duration
Interval	sequential chromatic pitch interval	Pitch
Tonic	tonic of relevant major/minor scale	KeySig, Mode
ScaleDegree	chromatic pitch interval from tonic	Pitch
FirstInBar	first event in bar, or otherwise	Onset
LastInPhrase	last event in phrase, or otherwise	Phrase
Threaded		
ScaleDegree \ominus FirstInBar	ScaleDegree threaded at first in bar	Pitch, Onset

we are using escape method C (see Witten and Bell, 1989, for a review of this and other escape methods). We are also using a method known as *exclusion* in conjunction with escape to obtain more accurate estimates; see Cleary and Teahan (1997) for more details.

3.4 Multiple Viewpoints

A multiple viewpoint system comprises more than one viewpoint: indeed, usually many more. If we were to use only unlinked basic viewpoint types to predict existing data, it would be possible to back off under PPM until the required prediction (and its associated probability) was found, without needing to complete the distribution. In general, this is not the case, however; there may, for example, be derived viewpoints in the system. Backing off only until the required prediction is found will result in different sets of predictions in the respective distributions (once they have been converted into distributions over the domain of the basic type), which makes it impossible to properly combine the distributions so that the overall prediction probability can be found. This, fundamentally, is why distributions must be completed within this framework. The reliable construction of viewpoint domains, which is central to this article, ensures that distributions are completed properly. The first step in the combination process, then, is to convert the completed viewpoint prediction probability distributions into distributions over the domain of whichever basic type is being predicted at the time, for example `Pitch`. It is possible for a prediction in a derived viewpoint (*e.g.*, `ScaleDegree`) distribution to map onto more than one prediction in the basic viewpoint distribution; in this case, the probability of the derived

viewpoint prediction is divided equally between the corresponding basic viewpoint predictions. The resulting distributions are then combined by employing a weighted arithmetic (Conklin, 1990) or geometric (Pearce et al., 2005) combination technique.

3.5 Long-term and Short-term Models

Conklin (1990) introduced the idea of using a combination of a *long-term model* (LTM), which is a general model of a style derived from a corpus, and a *short-term model* (STM), which is constructed as a single piece of music is being predicted or generated. The latter aims to capture musical structure particular to that piece. The prediction probability distributions produced by the two models are combined using a weighted arithmetic or geometric combination technique to give an overall distribution. Note that prediction of test data means the assignment of probabilities to known events using the overall model, whereas the generation of a harmony given a melody is achieved by *random sampling* of the overall distributions.

3.6 Viewpoint Selection

Pearce (2005) introduced a feature selection algorithm, based on *forward stepwise selection* and *N-fold cross-validation*, which constructs multiple viewpoint systems according to objective criteria (such as cross-entropy, described below). A variant of this algorithm is used in this research, as follows. We start with a set comprising the basic viewpoints to be predicted. At each iteration, we try adding in turn all primitive viewpoints not already in the set, and linked viewpoints comprising a primitive viewpoint already in the set plus one other (again, avoiding duplicates). The viewpoint resulting in the largest improvement to the model is added to the set (multiple viewpoint system). At this point, all possible deletions are considered which leave the system fully able to predict data. When neither additions nor deletions are able to improve the model, the system is complete (although not necessarily globally optimal). See, for example, Aha and Bankert (1996) for the application of stepwise selection to machine learning.

3.7 Evaluation

An information theoretic measure, *cross-entropy*, is used to guide the construction of models, evaluate them, and compare generated harmonisations (since cross-entropy is measured in bits per symbol, it is possible to compare harmonisations of any length). Cross-entropy is an upper bound on the true entropy of the data; therefore the model assigning the lowest cross-entropy to a set of test data is the most accurate model of the data. See Manning and Schütze (1999) for more details.

3.8 Concluding Remarks

Our research has developed the multiple viewpoint/PPM framework to cope with the complexities of harmony, such that improved computational models of four-part harmonisation can be created (an early attempt to model polyphony using multiple viewpoints was made by Conklin and Cleary, 1988, where the models were hand-crafted from a very limited pool of viewpoints). Whereas a melody is a single sequence of notes, four-part harmony is composed of four interrelated sequences of notes; and a note in one part is not necessarily sounded at the same time as notes in the other parts (*i.e.*, harmony is rarely completely homophonic, and may be far from it). We have implemented three versions of the framework, beginning with a very strict application of the existing multiple viewpoint framework, and then extending and generalising. There are challenges with respect to viewpoint domains even in the first of these versions; but by the third version, the challenges are much greater.

4 Learning Corpus and Test Data

At present, 110 major key hymn tunes, with harmonisations found in Vaughan Williams (1933), can be distributed between the learning corpus and the test data. Most of the work we have done so far has involved the use of a learning corpus comprising 50 hymn tune harmonisations, and test data of 5 harmonisations. The raw data is in the form of MIDI files. This data must be preprocessed before it can be converted into sequences of viewpoint elements which are subsequently used to create N-gram models (see §3.3). One of the requirements is that each part must be in the form of an event sequence, where each event comprises a number of basic attribute values (*e.g.*, pitch and duration). Most of this information is derived from the MIDI data; but phrase boundaries must be indicated by hand. The issue of automatic segmentation, which employs computational techniques to group music into, for example, motifs, phrases and sections, is not addressed in this research. Another requirement is that when an event occurs in one part, simultaneous events must also occur in the other three parts. This is achieved by applying the *full expansion* technique used in Assayag et al. (1999), where the distinction between the start of a note and its continuation is made explicit (this is explained in more detail in §6.1). Rests are problematic, as they are not considered to be events within the current viewpoint formulation; this must be resolved as part of the ongoing research. For the time being, it is expedient to exclude from the corpus and test data the relatively small number of hymn tune harmonisations containing rests.

5 Melodic Viewpoint Domains

We can ease our way into consideration of issues relating to domains by considering melody alone. Melodic domains are very small compared with harmonic domains; for example, there are only 19 different chromatic pitches (B♭3 to E5)

in the soprano parts of our fifty-five hymn corpus plus test data, with MIDI values of 58 to 76 inclusive. Run time, therefore, is not a problem; but other issues do need to be tackled. First, though, recall from §3.2 that $[\tau]$ denotes the domain of τ , and that $\langle\tau\rangle$ is the type set of τ , where τ is the viewpoint type.

5.1 Can a Domain be Fixed?

The first of these issues, whether or not a domain can be fixed, has been addressed by Pearce (2005). The domain of a basic viewpoint, such as **Pitch**, “is predefined to be the set of viewpoint elements occurring in the corpus” (Pearce, 2005, p. 115). Basic viewpoint domains are therefore fixed; but let us consider what happens if we assume that a derived viewpoint, such as **Interval**, also has a domain fixed in the same way. A domain comprising all **Interval** values which occur in the corpus will have both positive and negative values; that is, it will contain both ascending and descending intervals. If the previous note had a pitch which was, for example, the lowest in the **Pitch** domain, then any negative values ending up in the **Interval** prediction probability distribution will predict **Pitch** values which are not in the **Pitch** domain. On the other hand, some of the higher values in the **Pitch** domain will not be predicted by anything in the **Interval** distribution. Since “[a] model m_τ must return a complete distribution over the basic attributes in $\langle\tau\rangle$ ” (Pearce, 2005, p. 115), something must be done. The solution Pearce (2005, p. 115) arrives at is as follows:

To address this problem, the domain of each derived type τ is set prior to prediction of each event such that there is a one-to-one correspondence between $[\tau]$ and the domain of the basic type $\tau_b \in \langle\tau\rangle$ currently being predicted.

Although a truly one-to-one correspondence between **Interval** and **Pitch** domains is achievable, this is not necessarily the case for other viewpoints. It is important to be clear what is really meant here by a one-to-one correspondence; the term is used in an informal rather than a strictly mathematical sense. Let us consider the case of **ScaleDegree** (chromatic pitch interval from tonic). Unless the **Pitch** domain is unrealistically small, the **ScaleDegree** domain is

$$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}.$$

For a **Pitch** domain covering more than an octave, therefore, the function from **Pitch** to **ScaleDegree** is not one-to-one, but *surjective*. This is also true of other derived viewpoints; so the solution to the problem should be clarified as follows. The domain of a derived type τ is determined prior to prediction of each event such that, between them, its members are able to predict all of, and only, the members of the domain of the basic type $\tau_b \in \langle\tau\rangle$ currently being predicted.

Irrespective of **ScaleDegree**, which (in theory and practice) has a fixed domain, there exist derived viewpoint domains which are not fixed or static, but dynamic with respect to sequence position (*i.e.*, we do not know *a priori* what the domains are). In the case of **Interval**, the domain comprises the intervals

between the pitch of the previous note in the sequence and each of the members of the `Pitch` domain in turn; that is, the domain is a partition of $[\text{Pitch}] \times [\text{Pitch}]$. In general, each element of the basic domain in turn is converted to an element of the derived domain, and the latter is added to the derived viewpoint domain if it is not already a member of the domain (alternatively, they can all be added to the domain initially, and duplicates removed afterwards).

5.2 Linked Viewpoint Domains

We now turn our attention to the construction of domains for linked viewpoints. Conklin and Witten (1995) state that the linked domain is the Cartesian product of the individual viewpoint domains, that is,

$$[\tau] = [\tau_1] \times \dots \times [\tau_n].$$

For links between functionally unrelated viewpoints, such as `Duration` and `Pitch`, this is undoubtedly true. Constructing linked domains for viewpoints capable of predicting the same basic viewpoint is rather different, however. Let us consider the linked viewpoint `Pitch` \otimes `Interval`. The majority of elements in the Cartesian product of the `Pitch` and `Interval` domains are nonsensical: this is because, as we have already established above, there is a true one-to-one correspondence between elements of the individual viewpoint domains in this case. Only links between corresponding elements make logical sense; therefore only these links should be included in the domain. Researchers have so far only used linked viewpoints comprising two individual viewpoints; but the methods outlined in this and following sections can easily be extended to the linking of more than two viewpoints (explicitly so in §8).

6 Version 1 Viewpoint Domains

6.1 Introduction to Version 1

The starting point for the definition of the strictest possible application of viewpoints to harmony is the formation of *vertical viewpoint elements* (Conklin, 2002). An example of such an element is $\langle 67, 62, 59, 43 \rangle$ (the first chord of Figure 2), where all of the values are from the domain of the same viewpoint (in this case `Pitch`), and all of the parts (soprano, alto, tenor and bass) are represented in that order. Similarly, a vertical element for linked viewpoint `Pitch` \otimes `ScaleDegree` may, for example, be $\langle \langle 67, 0 \rangle, \langle 62, 7 \rangle, \langle 59, 4 \rangle, \langle 43, 0 \rangle \rangle$ (also the first chord of Figure 2). This method reduces the entire set of parallel sequences to a single sequence, thus allowing an unchanged application of the multiple viewpoint framework, including its use of N-grams and PPM. To exemplify this, Figure 2 shows an incomplete harmonisation of the first phrase of hymn tune *Luther's Hymn (Nun freut euch)* adapted from Vaughan Williams (1933), with a sequence of `ScaleDegree` vertical viewpoint elements underneath (question marks represent unknown `ScaleDegree` values). The outlined elements

0	0	4	2	0	2	2	4	S
7	7	0	11	0	0	?	?	A
4	0	7	7	4	9	?	?	T
0	4	0	7	9	5	?	?	B

Fig. 2. Incomplete harmonisation of the first phrase of hymn tune *Luther's Hymn* (*Nun freut euch*) adapted from Vaughan Williams (1933), with a sequence of vertical **ScaleDegree** elements underneath. A trigram is shown predicting the penultimate element, in which question marks represent the unknown **ScaleDegree** values.

form a trigram, which is predicting the penultimate chord. This is the base-level version, which we have subsequently developed (and intend to further develop in future work) with the aim of substantially improving performance.

Parts are a kind of *layer* associated with a voice. Other layers are possible; for example, a layer for chord symbols such as Vb (a dominant chord in first inversion) can be envisaged. The soprano part is given; so we are really only predicting the alto, tenor and bass parts. This being the case, the given soprano part is termed a *support layer*, and the other three parts (to be predicted) are called *prediction layers*. Another way of thinking about vertical viewpoint elements is to see them as links between layers; after all, we already have linked viewpoints within layers. We call the linking of viewpoints between layers *inter-layer linking*, and the linking of viewpoints within layers *intra-layer linking*. The formal nomenclature for inter-layer linked viewpoints is similar to that for intra-layer linked viewpoints; but the layers are explicitly labelled, and a subscript p is used to indicate prediction layers. If we assume that the primitive viewpoint **ScaleDegree** is used on each layer, the name of the inter-layer linked viewpoint is:

$$(\text{ScaleDegree})_S \otimes (\text{ScaleDegree})_{Ap} \otimes (\text{ScaleDegree})_{Tp} \otimes (\text{ScaleDegree})_{Bp}.$$

Although this nomenclature is overkill for this version of the framework (and so will not be used for the remainder of this section), it allows us to make better comparisons with later versions. For version 1 only, we shall use names such as $(\text{ScaleDegree})_{SATB}$.

To be completely explicit about which layer is which in a vertical element, and which viewpoint is represented, a more sophisticated representation of viewpoint elements has been developed. In this, relative chord position is defined as the position in the viewpoint sequence relative to the prediction position. Each layer is represented in the following way:

$$\langle \text{layer, relative chord position, intra-layer linked viewpoint, symbol tuple} \rangle$$

(the reason for the viewpoint name appearing in every layer will become clear in §8.1). Note that a primitive viewpoint is treated as a special case of an intra-layer linked viewpoint, and that the symbol tuple comprises as many symbols as there are primitive viewpoints in the intra-layer linked viewpoint. An example of the new representation is

$$\begin{aligned} &\langle\langle S, 0, \text{Pitch} \otimes \text{ScaleDegree}, \langle 69, 0 \rangle \rangle, \\ &\langle A, 0, \text{Pitch} \otimes \text{ScaleDegree}, \langle 64, 7 \rangle \rangle, \\ &\langle T, 0, \text{Pitch} \otimes \text{ScaleDegree}, \langle 61, 4 \rangle \rangle, \\ &\langle B, 0, \text{Pitch} \otimes \text{ScaleDegree}, \langle 57, 0 \rangle \rangle \rangle. \end{aligned}$$

which is equivalent to $\langle\langle 69, 0 \rangle, \langle 64, 7 \rangle, \langle 61, 4 \rangle, \langle 57, 0 \rangle \rangle$ for viewpoint $(\text{Pitch} \otimes \text{ScaleDegree})_{SATB}$. The second element of the layer tuples, a zero, is the relative chord position. When this representation is used in the construction of contexts for N-gram modelling, this element becomes a negative integer. The more complex representation will be used in §8.4, where the use of the simpler representation would be unduly ambiguous.

Basic viewpoint **Cont** is introduced in this research specifically for use in the modelling of harmony: it is required because the corpus (like music in general) is not completely homophonic. We require *simultaneities* (concurrent events) to have a single value for each part, and a single duration overall; therefore *full expansion* is used to partition music in this way. This technique has already been used in conjunction with viewpoints (Conklin, 2002). Figure 3 shows the first three full bars of the harmonisation of hymn tune *Caton* (or *Rockingham*) in the top system, and its fully expanded form in the bottom system. Looking at the first full bar, we see that the semibreve (whole note) D in the alto has been split into two minims (half notes); and that where the tenor moves in crotchets (quarter notes) from A to G, the minims in the other three parts have been split into crotchets. To model harmony correctly, we need to know which notes have been split in this way, and which have not. To distinguish between the start of a note and its continuation, Assayag et al. (1999) used different pitch symbols; for example, ‘b’ for the start of a note and ‘**b**’ (in bold) for its continuation. Our preferred solution is viewpoint type **Cont**, which obviates the need to further increase the size of the **Pitch** domain (or any other viewpoint domain, since **Cont**, like any other basic viewpoint, is an attribute of a note as a whole). This type has the value *T* if a note is a continuation of the previous one in the same part (*i.e.*, it has the same pitch, and is not re-sounded), and the value *F* otherwise; therefore the vertical $(\text{Cont})_{SATB}$ elements for the first full bar of Figure 3 are

$$\langle F, F, F, F \rangle \langle F, T, F, F \rangle \langle F, F, F, F \rangle \langle T, T, F, T \rangle.$$

6.2 Domain Size and Run Time

In this section, we provide some numbers to give an idea of the run time problem. A detailed empirical analysis of time complexity is presented in §9. As



Fig. 3. The first four full bars of the harmonisation of hymn tune *Caton* (or *Rockingham*), adapted from Vaughan Williams (1933), are presented in the top system, and the full expansion of this excerpt is shown in the bottom system.

noted at the beginning of §5, there are 19 elements in the soprano *Pitch* domain. In addition, there are 18, 20 and 23 elements in the alto, tenor and bass *Pitch* domains respectively. To enable a probability distribution to predict any combination of these pitches, the *full* $(\text{Pitch})_{SATB}$ domain (and distributions based on it) must contain 157,320 vertical viewpoint elements; and since, for example, the $(\text{Duration} \otimes \text{Pitch})_{SATB}$ domain is the Cartesian product of the constituent viewpoint domains, there would be well over a million elements in the linked domain. There is also a total of 277 chords in our fully expanded test set of five hymn tune harmonisations, each chord having three attributes requiring prediction. If we assume a long-term model only, having a multiple viewpoint system with three viewpoints able to predict *Duration*, three able to predict *Cont* and four able to predict *Pitch*, then the total number of prediction probability distributions required is 2,770 (ignoring the fact that the viewpoints might not always be defined). If, in addition, we use a short-term model, this figure rises to 5,540. This may not seem too bad; but now consider what happens during viewpoint selection. Each multiple viewpoint system tried is evaluated using ten-fold cross-validation of the corpus, which means that fifty hymn tune harmonisations are predicted, rather than just five; and many systems need to be evaluated during viewpoint selection. In spite of a certain amount of caching of distributions, a typical viewpoint selection run (using a pool of 39 primitive viewpoints) for the optimisation of both a long-term and a short-term model typically requires about 3×10^6 distributions to be constructed. The need to repeatedly construct very large probability distributions definitely results in run time problems.

One way of substantially improving run time is to cut the number of different combinations by placing into a basic viewpoint domain only those vertical elements which occur in the corpus, plus any others likely to occur. Since there is no easy way of identifying these additional elements, one solution is to simply include such elements which occur in nominally unseen test data (no statistics are collected from the test data; we are merely acknowledging the existence of vertical viewpoint elements which were not seen in the corpus). This has the practical advantage that probabilities can be assigned to all chords in the test data. There are currently 882 vertical elements in our *seen* $(\text{Pitch})_{SATB}$ domain. The basic viewpoints $(\text{Duration})_{SATB}$ and $(\text{Cont})_{SATB}$ have much smaller domains than $(\text{Pitch})_{SATB}$: only 11 vertical $(\text{Duration})_{SATB}$ elements appear in the data, while the $(\text{Cont})_{SATB}$ domain is limited to 15 (since $\langle T, T, T, T \rangle$ does not occur in the data).

If we were predicting or generating all four parts of the harmonic texture, this would be as far as we could go with respect to simplifying the basic domains. What we are particularly interested in, however, is the harmonisation of given melodies, which is a hard enough problem to be tackling for the time being (predicting all four parts is even more difficult, because it is a less constrained problem). In this case, since the soprano is given, and we do not wish to change it, only those vertical elements having a soprano pitch value the same as that given are allowed in the $(\text{Pitch})_{SATB}$ domain (and the resulting prediction probability distribution). In other words, domain elements contain a “must have” value from the support layer. This again greatly improves run time, since if the 882 vertical $(\text{Pitch})_{SATB}$ elements were divided equally amongst the 19 soprano $(\text{Pitch})_{SATB}$ elements, the size of the domain would be reduced to about 46. Of course, the distribution of elements is not really uniform; the true distribution is shown in Figure 4.

The use of the seen domain for the generation of test data melody harmonisations is not ideal, however. If the test data contains a single vertical element containing a soprano note not seen in the corpus, then that element has to be used to harmonise that soprano note. In addition, we know from Figure 10, which includes data from 110 hymns, that roughly 400 elements will be added to our 55-hymn seen $(\text{Pitch})_{SATB}$ domain by doubling the number of hymns. In other words, there are a great many possible chords which have not been seen by the machine learning program, but which could perfectly well be used to harmonise melodies. Our solution is simply to transpose chords which have been seen up and down, semitone by semitone, until one or other of the parts goes out of the range seen in the data. Such elements are added to the *augmented* $(\text{Pitch})_{SATB}$ domain, with the proviso that no duplicates are allowed. Obviously, these elements do not appear in the $(\text{Pitch})_{SATB}$ statistics gathered from the corpus, and so have very low prediction probabilities in $(\text{Pitch})_{SATB}$ distributions; but they can potentially have relatively high prediction probabilities in derived viewpoint distributions, such as those using $(\text{ScaleDegree})_{SATB}$. There are currently 5,040 vertical elements in our augmented $(\text{Pitch})_{SATB}$ domain. If these elements were divided equally amongst the 19 soprano Pitch elements, the

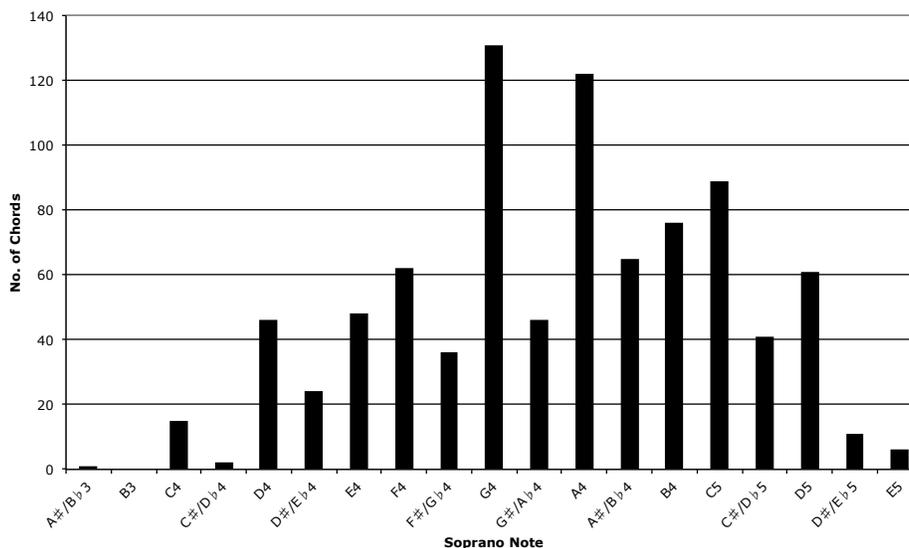


Fig. 4. Bar chart showing the number of different vertical elements in the seen $(Pitch)_{SATB}$ domain for each soprano note.

size of the domain for each prediction would be reduced to about 265. The true distribution of elements is shown in Figure 5. Note that the augmented domain is much closer in size to the seen domain than it is to the full domain. The seen domain is always a subset of the augmented domain, and for a large enough corpus could potentially closely approach the augmented domain in terms of size. Similarly, the augmented domain is always a subset of the full domain; but unless the corpus were to contain a large number of very strange chords, the size of the augmented domain could never approach that of the full domain.

6.3 Less Obvious Constraints

Following Conklin (1990) and Pearce (2005), we predict each basic attribute in turn at each chord prediction. In this research, prediction is carried out in the following order: **Duration**, **Cont**, **Pitch**. The reasoning is that **Duration** distributions were thought to have the lowest entropy, meaning that **Duration** is the most predictable attribute. Similarly, in general, **Pitch** distributions have the highest entropy, meaning that **Pitch** is usually the least predictable attribute. Following their prediction, known values of **Duration** and **Cont** (or viewpoint types derived from them) are used in linked viewpoints better to predict the (normally) more unpredictable **Pitch**. Having said all this, it was later found that, in fact, **Cont** distributions generally have the lowest entropy. Although it is usual in AI to deal with the most predictable attribute first, there is evidence from another area of our research to suggest that this will not necessarily produce

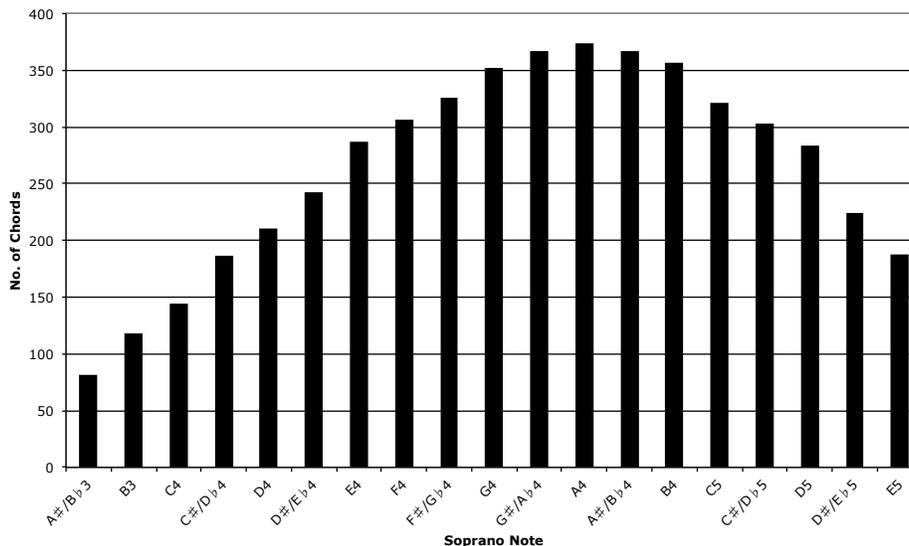


Fig. 5. Bar chart showing the number of different vertical elements in the augmented $(Pitch)_{SATB}$ domain for each soprano note.

the best results in the case of harmonisation. Our current $Duration$, $Cont$, $Pitch$ ordering is certainly not intrinsically bad. A detailed investigation into the effect of attribute prediction order is planned for future work.

Let us consider the basic viewpoint $(Duration)_{SATB}$ from the point of view of generation of a harmonisation. First of all, since a simultaneity has a single overall duration, a vertical viewpoint element always contains the same duration value for each part, such as $(48, 48, 48, 48)$. The domain comprises vertical elements containing durations which are less than or equal to the duration of the soprano note to be harmonised. This allows the possibility of passing and other unessential notes in the lower three parts. If, for example, the soprano note had a duration of 24 (a crotchet, or quarter note), and a duration of 12 (a quaver, or eighth note) was generated, the soprano note would be expanded, with the second quaver being assigned a $Cont$ value of T . The $(Cont)_{SATB}$ and $(Pitch)_{SATB}$ values of the first quaver are predicted before the focus is shifted to the second quaver, when prediction begins again with $(Duration)_{SATB}$. After the prediction of $(Duration)_{SATB}$, its domain contains only the predicted vertical viewpoint.

There are interactions between $Cont$ and $Pitch$, with different constraints operating depending on the attribute being predicted. During the prediction of $(Cont)_{SATB}$, the $(Cont)_{SATB}$ domain must not contain elements which would predict elements outside of the $(Pitch)_{SATB}$ domain. For example, let us assume that we have an E^b major chord, $(67, 63, 58, 51)$, followed by an $F4$ (MIDI value 65) in the soprano, as shown in the first bar of Figure 6. For simplicity, let us also



Fig. 6. The effect of viewpoint $(\text{Cont})_{SATB}$ on a chord progression. Bar 1 shows the soprano note to be harmonised and the preceding chord. Bars 2 to 5 illustrate the effect of vertical $(\text{Cont})_{SATB}$ elements $\langle F, F, F, F \rangle$, $\langle F, T, F, F \rangle$, $\langle F, F, T, F \rangle$ and $\langle F, T, T, F \rangle$ respectively on the chosen second chord.

assume that the only vertical element in the $(\text{Pitch})_{SATB}$ domain constrained by the soprano note is $\langle 65, 63, 58, 50 \rangle$. The set of vertical $(\text{Cont})_{SATB}$ elements compatible with this artificially small $(\text{Pitch})_{SATB}$ domain is

$$\{\langle F, F, F, F \rangle \langle F, T, F, F \rangle \langle F, F, T, F \rangle \langle F, T, T, F \rangle\}.$$

Any other $(\text{Cont})_{SATB}$ element, such as $\langle F, T, T, T \rangle$ or $\langle F, F, F, T \rangle$, would map onto a $(\text{Pitch})_{SATB}$ element which is not in the domain. Bars 2 to 5 of Figure 6 show the chord progression resulting from each of these $(\text{Cont})_{SATB}$ elements respectively, in conventional musical notation. Even realistically sized $(\text{Pitch})_{SATB}$ domains can restrict the $(\text{Cont})_{SATB}$ domain beyond what might be expected by constraining according to the Cont attribute of the soprano note. Once $(\text{Cont})_{SATB}$ has been predicted, its domain comprises a single vertical element, and the $(\text{Pitch})_{SATB}$ domain is further constrained to vertical elements which are compatible with the predicted vertical $(\text{Cont})_{SATB}$ element.

6.4 Construction of Derived and Linked Viewpoint Domains

Derived viewpoint domains are constructed by converting each vertical element of the relevant basic viewpoint domain (however it is currently constrained) into a vertical element of the derived domain. By constructing the domain in this way, we can be sure that between them, the members will be able to predict all of, and only, the members of the basic domain. As with melodic domains, after each conversion the vertical element is added to the derived domain only if it is not already a member (recalling that the conversion function is surjective). For example, assuming a key of A, vertical $(\text{Pitch})_{SATB}$ elements $\langle 69, 64, 61, 57 \rangle$ and $\langle 69, 64, 61, 45 \rangle$ both map to vertical $(\text{ScaleDegree})_{SATB}$ element $\langle 0, 7, 4, 0 \rangle$.

As with melodic domains, for links between functionally unrelated viewpoints, the linked domain is the Cartesian product of the individual viewpoint domains; and for links between viewpoints capable of predicting the same basic viewpoint, the informal unidirectional one-to-one correspondence between elements of the individual viewpoint domains means that only corresponding elements are included in the linked domain (there are also correspondences between Cont and Pitch , and viewpoints derived from it, as discussed above).

Continuing the example in the paragraph above, for a key of A, vertical elements $\langle\langle 69, 0 \rangle, \langle 64, 7 \rangle, \langle 61, 4 \rangle, \langle 57, 0 \rangle\rangle$ and $\langle\langle 69, 0 \rangle, \langle 64, 7 \rangle, \langle 61, 4 \rangle, \langle 45, 0 \rangle\rangle$ would both appear in the $(\text{Pitch} \otimes \text{ScaleDegree})_{SATB}$ domain; but $\langle\langle 69, 0 \rangle, \langle 64, 7 \rangle, \langle 61, 4 \rangle, \langle 55, 0 \rangle\rangle$ would not, since in the bass part the Pitch value of 55 corresponds to a G, and the ScaleDegree value of 0 corresponds to an A.

The reason that we must be able to reliably construct such domains is because prediction probability distributions are completed by backing off to uniform distributions based on these domains. The various viewpoint distributions are converted into basic viewpoint distributions prior to combination. To facilitate combination, the distributions are sorted into the same order. An unreliable domain construction procedure might, for example, result in one or two predictions being missing from some distributions, which means that at some point a succession of probabilities will be erroneously combined.

7 Version 2 Viewpoint Domains

7.1 Introduction to Version 2

In this version, it is hypothesised that predicting all unknown symbols in a vertical viewpoint element (as in version 1) at the same time is neither necessary nor desirable. It is anticipated that by dividing the overall harmonisation task into a number of subtasks (Allan and Williams, 2005; Hild et al., 1992), each modelled by its own multiple viewpoint system, an increase in performance can be achieved. For example, given a soprano line, the first subtask might be to predict the entire bass line. Since there is no supporting context in the alto and tenor layers yet, inter-layer linked viewpoints such as $(\text{ScaleDegree})_S \otimes (\text{ScaleDegree})_{Bp}$ are used (recall that subscript p indicates the *prediction* layer; that is, the layer to be predicted). Once the bass line has been predicted, it becomes a support layer in subsequent subtasks. This version allows us to experiment with different arrangements of subtasks. For example, having predicted the bass line, is it better to predict the alto and tenor lines together, or one before the other? Note that text books on harmonisation generally advocate the completion of the bass line first, in conjunction with harmonic function symbols such as Ib for tonic in first inversion. Alto and tenor notes are then added together (see Whorley et al. (2013) for an information theoretic investigation of this subject). Another alternative (not implemented) would be to predict all of the notes of a chord by the application of a sequence of subtask models before moving on to the next chord. As in version 1, vertical viewpoint elements are restricted to using the same viewpoint on each layer. The difference is that not all of the layers are now necessarily represented in a vertical viewpoint element; for example, for the prediction of bass given soprano, only the soprano and bass are represented, as shown in Figure 7.

7.2 Domains

Provided we always predict only one part at a time, and always constrain the Pitch domain as much as possible, we could reasonably construct full domains

7	7	0	11	0	0	?	?	A
4	0	7	7	4	9	?	?	T
43	47	43	50	52	48	50	43	B

Fig. 8. Incomplete harmonisation of the first phrase of hymn tune *Luther's Hymn* (*Nun freut euch*) adapted from Vaughan Williams (1933), with a sequence of $(\text{ScaleDegree})_{A_p} \otimes (\text{ScaleDegree})_{T_p} \otimes (\text{Pitch})_B$ elements underneath. A trigram is shown predicting the penultimate element, in which question marks represent the unknown **ScaleDegree** values.

the alto, tenor and bass, but not soprano, as shown in Figure 8. It should be noted, however, that even if a support layer is not represented in a linked viewpoint, the domain is still constrained by the given note at the prediction point of this layer. At present, for (relative) ease of implementation, prediction layers are assigned the same viewpoint, although support layers may have any combination of viewpoints (*c.f.* version 2, where the same viewpoint appears on all layers). A relaxation of the prediction layer restriction could conceivably result in better (but more complex) models; so we are contemplating a sub-version without this restriction. As usual, if at any point in the event sequence a derived viewpoint is undefined, an inter-layer linked viewpoint containing that viewpoint is also undefined.

It is necessary to modify the viewpoint selection algorithm described in §3.6. We start with a set consisting of the basic viewpoints to be predicted, which now comprise prediction layers only. Assuming that we are given the soprano and bass, and are predicting the alto and tenor, one of the viewpoints in this initial set is $(\text{Duration})_{A_p} \otimes (\text{Duration})_{T_p}$. At each iteration, we try adding in turn viewpoints such as $(\text{ScaleDegree})_{A_p} \otimes (\text{ScaleDegree})_{T_p}$; and viewpoints involving incremental intra- or inter-layer linkages with viewpoints already in the set, such as $(\text{Duration} \otimes \text{ScaleDegree})_{A_p} \otimes (\text{Duration} \otimes \text{ScaleDegree})_{T_p}$ or $(\text{Duration})_{A_p} \otimes (\text{Duration})_{T_p} \otimes (\text{ScaleDegree})_B$. The rest of the algorithm is the same as before.

8.2 Domain Construction Issues

Version 3 viewpoints can be much more complex than any we have seen before: especially when three parts are given and we are predicting the remaining part. In this case, each of the four parts could have a different viewpoint; but let us begin with something far more simple. During prediction of bass given soprano, we may use the viewpoint $(\text{ScaleDegree})_S \otimes (\text{Pitch})_{B_p}$. Although the constituent

viewpoints are able to predict the same basic viewpoint, the fact that they are assigned to different layers means that there are no correspondences which need to be taken into account; therefore taking the Cartesian product of the individual layer domains (as constrained by the given soprano note) is a perfectly acceptable way of constructing the inter-layer linked domain. We would not wish to do so in practice, however, for reasons outlined above. As before, we would place elements into the domain which correspond to `Pitch` combinations found in the corpus and test data (and optionally, transpositions of these combinations).

Unfortunately, most version 3 viewpoints are not as straightforward as this. We are predicting basic viewpoints `Duration`, `Cont` and `Pitch`, each of which has its own domain of a particular size; there are other basic viewpoints which we assume to be given; there are many derived viewpoints, most of them derived from `Pitch`, having domains of various sizes; and there are up to four layers represented in any viewpoint. A general method for reliably constructing domains for these complex viewpoints is required. As usual, the domains must be able to predict all of, and only, the members of the basic viewpoint domain(s). It is possible for each layer to have a different viewpoint; therefore it is expedient to deal with each layer in turn. This being the case, to achieve precisely the correct combinations of viewpoint elements, we need to know in advance how many elements there are in the provisional inter-layer linked viewpoint domain (which, as we shall see, may end up containing undefined or duplicate elements which must be removed). To calculate this number, we also need to know the set of basic viewpoints that the constituent viewpoints are derived from, that is, the type set $\langle\tau\rangle$ (note that the set of basic viewpoints that the inter-layer linked viewpoint is able to predict is a subset of $\langle\tau\rangle$, since only constituent viewpoints of the prediction layers should be considered); since we can determine what the domains of these basic viewpoints are for the combination of layers in question, we are easily able to ascertain their sizes. For the purposes of constructing a provisional inter-layer linked domain, we assume that the size of each derived domain (covering all of the parts represented in the viewpoint) is the same as that of the relevant basic domain. This means that we can relatively easily assign *inner-* and *outer-multipliers* to each basic or derived constituent viewpoint prior to construction of the inter-layer linked viewpoint. These multipliers respectively determine the number of times a primitive domain element is repeated prior to moving on to the next, and the number of times the entire primitive domain is repeated (along with any internal repeats).

As a simple example, let us assume that there are only two elements (A and B) in the primitive domain, and that the inner- and outer-multipliers have values of 3 and 2 respectively. Figure 9 illustrates the effect of the multipliers on the primitive domain. The elements of this domain are each shown as the root of a tree. The outer-multiplier is 2; therefore each root divides into two branches, resulting in the primitive domain being shown twice. The inner-multiplier is 3; so each of the four nodes splits into three, thereby duplicating elements within each of the copies of the primitive domain. The number of elements in the provisional

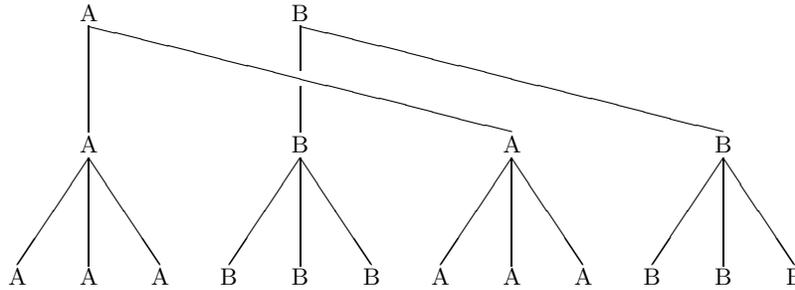


Fig. 9. Trees illustrating the effect of multipliers on primitive domain $\{A, B\}$. Branching from the roots is due to the outer-multiplier (2); and further branching to produce the leaves is due to the inner-multiplier (3).

inter-layer linked domain is the same as the total number of leaves on the trees, which is the product of the primitive domain size and the two multipliers.

8.3 Determination of Multipliers

In calculating the multipliers, we assume that the basic viewpoints are in a particular order: **Duration**, **Cont**, **Pitch**, followed by other basic viewpoints. The order of the other basic viewpoints is not important, as in our research they are given and therefore effectively have a domain containing only one element. Once we know which of **Duration**, **Cont** and **Pitch** the constituent viewpoints are derived from, we can assign multipliers according to their relative positions in the ordered list. For a particular basic viewpoint, the inner- and outer-multipliers are the product of the domain sizes of any basic viewpoints occurring after it, and before it, respectively. The default value of both is 1. For example, consider a linked viewpoint with constituents derived from **Duration**, **Cont** and **Pitch**, and which also contains viewpoint **LastInPhrase** (derived from basic viewpoint **Phrase**). If we assume that **Duration**, **Cont** and **Pitch** have domain sizes of 5, 10 and 40 respectively, then for any constituent viewpoint able to predict **Duration**, the inner- and outer-multipliers are 400 and 1; for any constituent viewpoint able to predict **Cont**, the inner- and outer-multipliers are 40 and 5; for any constituent viewpoint able to predict **Pitch**, the inner- and outer-multipliers are 1 and 50; and for any other constituent viewpoint (*e.g.*, **LastInPhrase**), the inner- and outer-multipliers are 1 and 2000 respectively. Generalised algorithms for the determination of multipliers can be found in Algorithms 1, 2 and 3 in Appendix A, along with a line-by-line description of their use in the above example.

8.4 Domain Construction Procedure

This procedure will be illustrated by a modified real example: only a small subset of the actual **Pitch** domain is used, comprising note names such as **Ab4** rather

than MIDI numbers, in order to make the illustration more readily intelligible. We are predicting bass given soprano using, amongst other viewpoints,

$$(\text{Duration} \otimes \text{Pitch})_S \otimes (\text{Cont} \otimes \text{ScaleDegree})_{Bp}.$$

The set of basic viewpoints (type set $\langle \tau \rangle$) that the above's constituent viewpoints are derived from is $\{\text{Duration}, \text{Cont}, \text{Pitch}\}$. The **Duration** attribute of a bass note has already been predicted; therefore the **Duration** domain contains only one element:

$$\{\langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \langle B, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle\}.$$

The next attribute to be predicted is **Cont**. The given soprano note has a **Cont** attribute of F ; therefore the domain is constrained to two elements:

$$\{\langle \langle S, 0, \text{Cont}, \langle F \rangle \rangle \langle B, 0, \text{Cont}, \langle T \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Cont}, \langle F \rangle \rangle \langle B, 0, \text{Cont}, \langle F \rangle \rangle \rangle\}.$$

The 5-element **Pitch** domain (constrained to have a soprano $\text{Ab}4$) is

$$\{\langle \langle S, 0, \text{Pitch}, \langle \text{Ab}4 \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{F}2 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Pitch}, \langle \text{Ab}4 \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{Eb}3 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Pitch}, \langle \text{Ab}4 \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{E}3 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Pitch}, \langle \text{Ab}4 \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{F}3 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Pitch}, \langle \text{Ab}4 \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{F}\sharp 3 \rangle \rangle \rangle\}.$$

The highest part (in terms of pitch) is dealt with first. If the viewpoint on this layer is, or contains, a basic viewpoint, each of the symbols or values belonging to that layer in the basic viewpoint domain is added, in turn, to what will become the inter-layer linked viewpoint domain, with repeats determined by the inner- and outer-multipliers. If the viewpoint is derived, the procedure is the same except that each basic viewpoint symbol is converted to a derived viewpoint symbol. In the example, viewpoint **Duration** in the soprano part is dealt with first. Its inner-multiplier is the product of the **Cont** and **Pitch** domain sizes, which is 10, and its outer-multiplier is 1. The provisional linked domain therefore starts off as

$$\{\langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle, \langle \langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \rangle\}.$$

At this point, the provisional inter-layer linked viewpoint domain contains all of the elements we need (and probably more), albeit that they are incomplete. If there is a second basic or derived viewpoint associated with this layer (*i.e.*, a constituent of an intra-layer linked viewpoint), then the same procedure is followed except that the symbols are linked with the symbols already in the provisional domain, in the same order as the original additions to the domain.

In the example, viewpoint `Pitch` in the soprano part is dealt with next. Its inner-multiplier is 1, and its outer-multiplier is the product of the `Duration` and `Cont` domain sizes, which is 2. The provisional linked domain then becomes

$$\{\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle\}.$$

We then move on to the next layer, again adding to the elements already in the provisional domain, and so on, until all the layers have been dealt with. There is only one other layer in the example, the bass part, which contains the intra-layer linked viewpoint `Cont` \otimes `ScaleDegree`. Primitive viewpoint `Cont` is dealt with first; its inner-multiplier is 5 (the size of the `Pitch` domain), and its outer-multiplier is 1 (the size of the `Duration` domain). The provisional linked domain then becomes

$$\{\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle\}.$$

The final viewpoint to be added is `ScaleDegree` in the bass part; because it predicts `Pitch`, its inner-multiplier is 1, and its outer-multiplier is 2. There is a particular problem to be overcome with respect to the intra-layer linking of `Cont` with `Pitch`, or viewpoints derived from it, however. As we have seen in §6.3, there is an interaction between the `Cont` and `Pitch` domains; therefore the Cartesian product would contain pairings making no logical sense. In this case, as the `Pitch` domain is traversed, the `Pitch` symbol is checked against the relevant `Cont` symbol (which is already in the provisional domain): if the pairing makes logical sense, the `Pitch` symbol, or a symbol derived from it, is added to the element in the provisional domain as usual; if not, the element is tagged as undefined. The previous note was F3 (MIDI value 53); therefore 53 is the only `Pitch` value which can be sensibly paired with a `Cont` value of `T`. The tonic is

Eb; therefore F has a `ScaleDegree` value of 2. The provisional linked domain then becomes

```
{⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, undef⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, undef⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, undef⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨T, 2⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, undef⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 2⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 0⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 1⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 2⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 3⟩⟩⟩}.
```

The provisional domain, constructed in this way in order to ensure that symbols which do not correspond with each other are not linked, may contain elements tagged as undefined, and may also contain duplicate elements; therefore the final inter-layer linked viewpoint domain is achieved once undefined and duplicate elements have been removed:

```
{⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨T, 2⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 0⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 1⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 2⟩⟩⟩,
  ⟨⟨S, 0, Duration ⊗ Pitch, ⟨24, Ab4⟩⟩, ⟨B, 0, Cont ⊗ ScaleDegree, ⟨F, 3⟩⟩⟩}.
```

A generalised algorithm for the construction of version 3 inter-layer linked viewpoint domains can be found in Algorithm 4 in Appendix A, along with a line-by-line description of its use in the above example.

9 Time Complexity Analysis

To demonstrate the utility of reducing domain size, we have carried out an empirical time complexity analysis on version 1 prediction runs. This involved running the computer model with different numbers of viewpoints, different sizes of corpus, and different types of $(Pitch)_{SATB}$ domain (seen, augmented and full). Maximum N-gram order also influences run time, but we have not yet investigated it; all runs used a maximum N-gram order of 3. Seen $(Duration)_{SATB}$ and $(Cont)_{SATB}$ domains are used throughout, which are small enough to be neglected for the purposes of this analysis. During the course of this exercise, we have also determined how $(Pitch)_{SATB}$ domain size varies with corpus/test data size for the seen, augmented and full domain cases. It is with this that we shall begin.

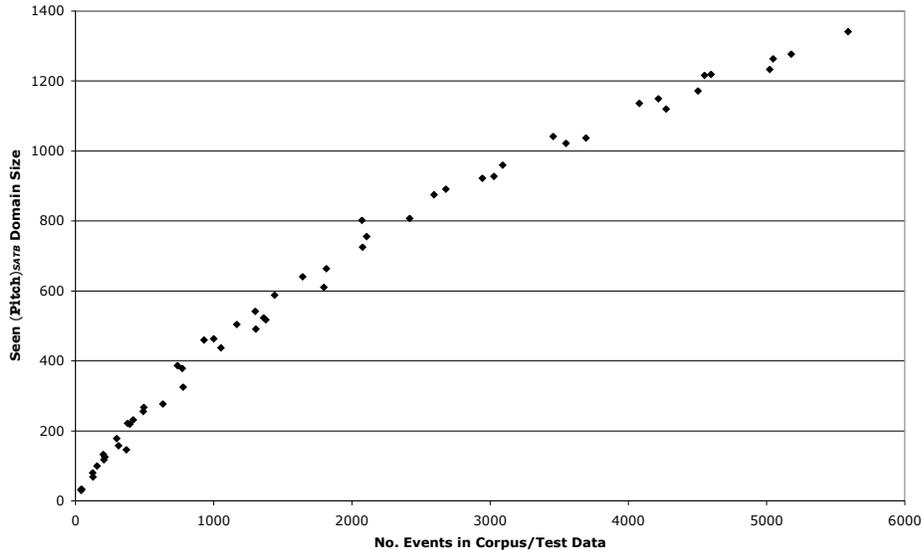


Fig. 10. Plot of number of events in the corpus/test data against seen $(\text{Pitch})_{SATB}$ domain size.

9.1 Variation of Domain Size with Corpus/Test Data Size

Seen, augmented and full $(\text{Pitch})_{SATB}$ domains were computed using 19 different combined domain and test data sizes between 1 and 110 hymns. For each of these sizes (except 110, which comprised all available data), there were three randomly selected sets of hymns; since the hymns vary in length, this greatly increased the number of different corpus/test data sizes in terms of the number of events. A plot of number of events in the corpus/test data against seen $(\text{Pitch})_{SATB}$ domain size (see Figure 10) shows a rapid discovery of novel chords at very low corpus/test data sizes. The discovery rate tails off with increasing corpus/test data size until the size reaches about 2,000 events, above which there appears to be a linear relation. Given sufficiently large corpus/test data sizes, we would expect the discovery rate to decline, leading to the relationship becoming asymptotic.

A similar plot for the augmented $(\text{Pitch})_{SATB}$ domain has much the same shape (see Figure 11); but the data points are more dispersed, since the domain size depends on both the number of novel chords and the domain sizes of the individual parts (SATB). The augmented domain is about five times larger than the seen domain. We would expect the augmented domain relationship to become asymptotic at lower corpus/test data sizes than that for the seen domain.

Finally, Figure 12 is a similar plot for the full $(\text{Pitch})_{SATB}$ domain. In this case, there is initially an extremely rapid increase in domain size with corpus/test data size, followed by a rapid decline in the increase, followed by a long tail with relatively little increase (*i.e.*, the relationship is already close to asymptotic).

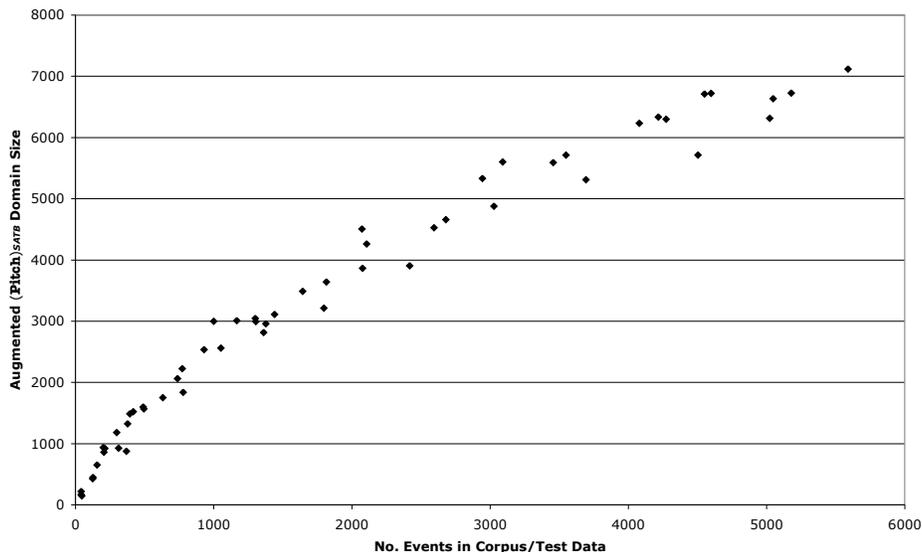


Fig. 11. Plot of number of events in the corpus/test data against augmented $(\text{Pitch})_{SATB}$ domain size.

The data points are very dispersed, due to the fact that domain size depends solely on the domain sizes of the individual parts. A log curve seems to fit this data best, which is not the case for the seen and augmented data.

9.2 Effect of Domain Size on Program Running Time

Figure 13 is a log-log plot of $(\text{Pitch})_{SATB}$ domain size against time for the learning phase of the program, which was run using a corpus of 30 hymns and multiple viewpoint systems comprising 2 and 10 viewpoints. There are three domain sizes, corresponding to the seen, augmented and full domains. The seen domain causes the program to run about two orders of magnitude faster than the full domain. Encouragingly, the use of the augmented domain results in a running time which is not too much slower than that of the seen domain.

For the prediction phase of the program (Duration , Cont and Pitch prediction), the relative differences in running time are even greater: the seen domain causes the program to run about three orders of magnitude faster than the full domain (see Figure 14). The running time of the program using the augmented domain is still fairly close to that using the seen domain.

We have demonstrated the utility of reducing the $(\text{Pitch})_{SATB}$ domain size. The full $(\text{Pitch})_{SATB}$ domain causes the program to run very slowly even for the prediction of only one short harmonisation. For viewpoint selection runs, which involve ten-fold cross-validation of the corpus, the situation is far worse. During the course of such a run, many multiple viewpoint systems are tried. For each of

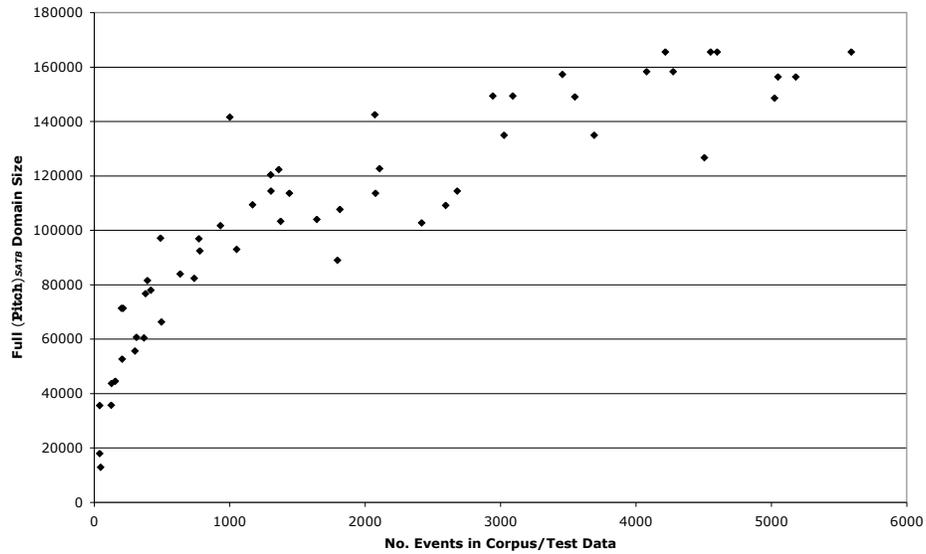


Fig. 12. Plot of number of events in the corpus/test data against full $(\text{Pitch})_{\text{SATB}}$ domain size.

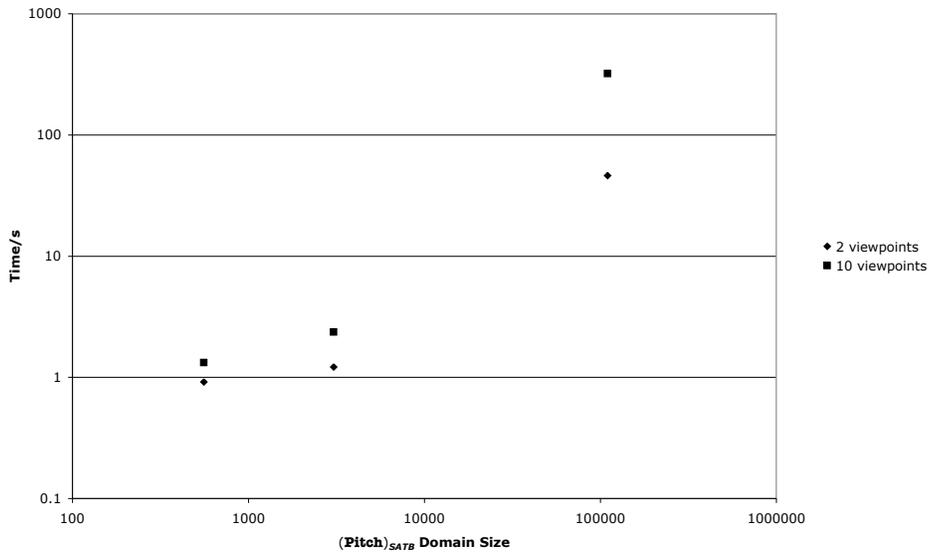


Fig. 13. Log-log plot of $(\text{Pitch})_{\text{SATB}}$ domain size against time for the learning phase of the program, which was run using a corpus of 30 hymns and multiple viewpoint systems comprising 2 and 10 viewpoints.

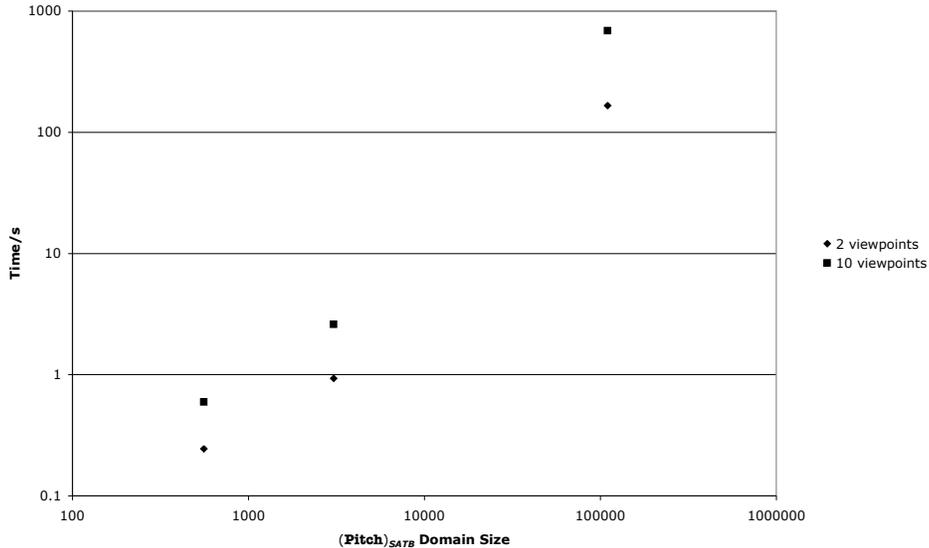


Fig. 14. Log-log plot of $(\text{Pitch})_{\text{SATB}}$ domain size against time for the prediction phase of the program, which was run using a corpus of 30 hymns and multiple viewpoint systems comprising 2 and 10 viewpoints. A single hymn tune harmonisation comprising 33 events was predicted.

these systems, the learning phase is run ten times, and every harmonisation in the corpus is predicted. Since it is clearly not practical to use the full domain, further analysis will concentrate on the use of the seen and augmented domains. Time complexities will be derived in terms of number of viewpoints, size of corpus and size of test data, to give prospective developers of similar programs an idea of their effect on running time.

9.3 Empirical Time Complexity Analysis Using Seen and Augmented Domains

In this more detailed analysis, the learning phase is split into domain construction and model construction phases. First of all, it is appropriate to explain why we have decided to derive time complexities empirically rather than analytically. The domain construction phase should be the easiest to analyse, since the number of viewpoints is irrelevant: we are only constructing the three basic domains during this phase. We can simplify things further by neglecting the *Duration* and *Cont* domains, which are very small in comparison with that of *Pitch*. The procedure is to traverse the corpus and test data, adding previously unseen elements to the *Pitch* domain. The traversal time, ignoring domain processing time, is proportional to the number of events in the corpus and test data; but we cannot ignore domain processing. For each event in the corpus and test data,

its `Pitch` element must, in the worst case, be compared with every element in the `Pitch` domain, which increases in size throughout this process. The rate of increase and ultimate size is much greater for the augmented domain than for the seen domain. In neither case, however, can this information be accurately determined *a priori* for any given corpus and test data (although inspection of Figures 10 and 11 can give a rough idea for this style of music); we consider it better, therefore, to derive the time complexity empirically.

Having said that, we know that in this particular case there are only 882 elements in the seen domain, compared with 2,601 events in the corpus and test data (a ratio of 0.34). If we define n_c to be the number of events in the corpus and n_{td} to be the number of events in the test data, at worst the time complexity for the seen domain is $O((n_c + n_{td})^{1.34})$. On the other hand, there are 5,040 elements in the augmented domain; therefore in this case we would expect a complexity of $O((n_c + n_{td})^{2.94})$ at worst.

Since time complexities for the model construction and prediction phases are even more difficult to derive analytically (*e.g.*, we cannot determine the size and structure of the viewpoint models *a priori*), we have made no attempt to do so. Let us now look at the empirical analyses.

Domain Construction Phase Figure 15 shows a plot of number of viewpoints against time for the seen domain construction phase of the program, which was run using corpora of 5, 10, 15, 20, 25 and 30 hymns. Each data point has a mean time of ten runs, each run using a different randomly selected multiple viewpoint system capable of predicting `Duration`, `Cont` and `Pitch` (the ten runs vary widely in duration). Straight lines fit the data reasonably well. All six lines are close to horizontal; therefore it has been concluded that, in reality, the run times are constant with respect to the number of viewpoints. This makes sense, considering that only the three basic viewpoint domains are constructed during this phase. A similar plot for the augmented domain (see Figure 16) also leads us to the conclusion that the times are constant with respect to the number of viewpoints. The times are taken to be those in the centre of the fitted lines; that is, at the 6 viewpoint mark. Two other data sets were also generated and analysed, using different sets of hymns for each corpus size (resulting in different numbers of corpus events). The graphs, which are not shown here, are similar to those for the first data set.

Bearing in mind that the test data (in addition to the corpus) is taken into account when constructing domains, we are now in a position to plot the number of events in the corpus and test data against time; see Figure 17, which makes use of all three of the generated data sets. The data is rather sparse and scattered; but a straight line seems to produce a reasonable fit for the seen domain, and a quadratic curve a reasonable fit for the augmented domain (both fits are constrained to extend to the origin). The fact that time increases more rapidly with increasing number of events for the augmented domain is not unreasonable, since there is additional processing involved in its construction.

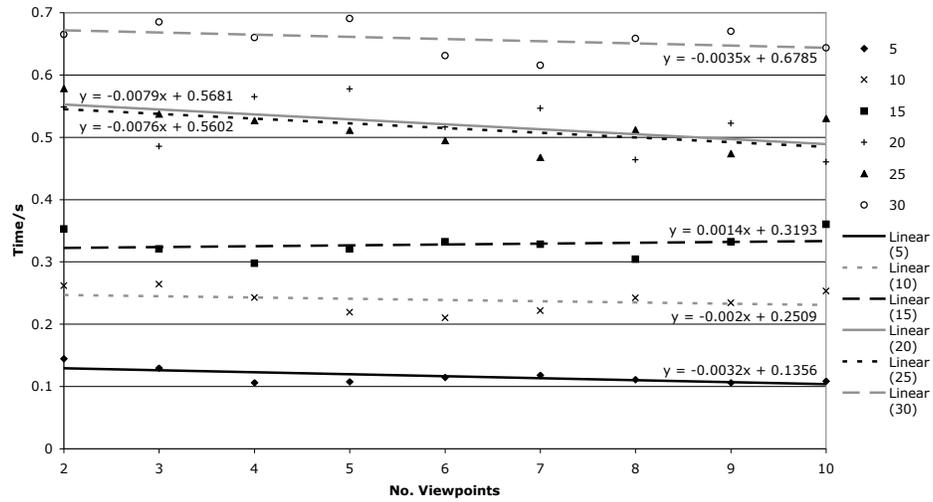


Fig. 15. Plot of number of viewpoints against time for the seen domain construction phase of the program, which was run using corpora of 5, 10, 15, 20, 25 and 30 hymns.

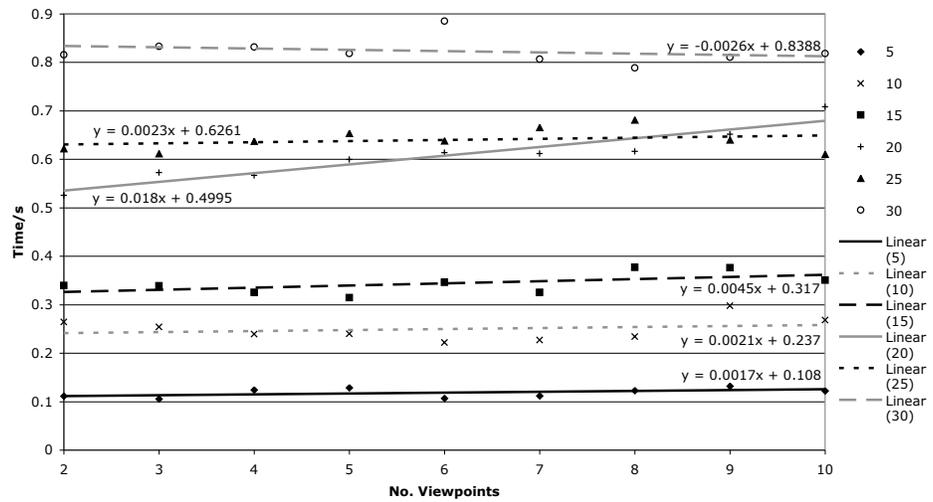


Fig. 16. Plot of number of viewpoints against time for the augmented domain construction phase of the program, which was run using corpora of 5, 10, 15, 20, 25 and 30 hymns.

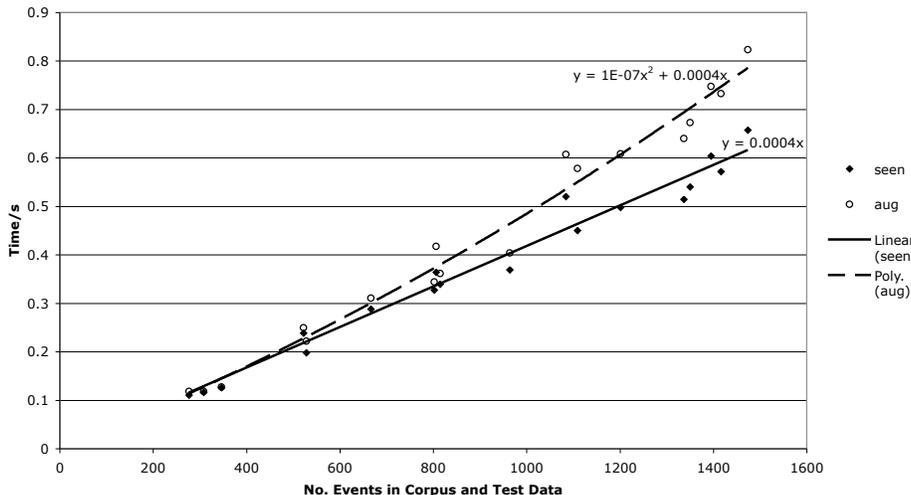


Fig. 17. Plot of number of events in the corpus and test data against time for the seen and augmented domain construction phases of the program.

From the foregoing we gather that the time complexity of the domain construction phase of the program is at worst $O((n_c + n_{td})^2)$. In other words, run time is proportional to the square of the number of events in the corpus and test data. This is better than the semi-analytically derived worst case complexity of $O((n_c + n_{td})^{2.94})$, which necessarily made use of simplifying assumptions.

Model Construction Phase Figure 18 shows a plot of number of viewpoints against time for the model construction phase of the program, which was run using the seen $(\text{Pitch})_{SATB}$ domain and corpora of 5, 10, 15, 20, 25 and 30 hymns. Again, straight lines fit the data reasonably well; but in this case there is a definite increase in run time with increasing number of viewpoints, which is due to the fact that a separate model needs to be built for each viewpoint. A similar plot for the augmented $(\text{Pitch})_{SATB}$ domain (see Figure 19) shows a larger increase in run time with increasing number of viewpoints. This is almost certainly not due to an increase in the time required to construct the models from the corpus *per se*, but rather to the time needed for other processing involving domains in this part of the program. Again, two other data sets were also generated and analysed, with similar results.

We can now plot the number of of events in the corpus against time; see Figure 20. The times are taken from the fitted lines on the plots described above rather than from the original data points. In this case, the best fit to the data is a quadratic curve (constrained to extend to the origin); but the trend is close to linear for the range of data analysed. As expected, run time increases with the number of events in the corpus: there is a more rapid increase in time with

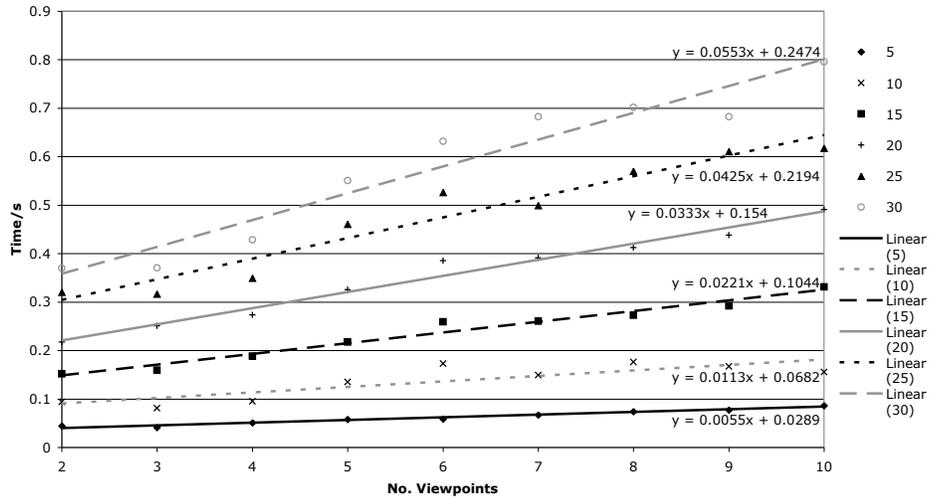


Fig. 18. Plot of number of viewpoints against time for the model construction phase of the program, which was run using the seen $(Pitch)_{SATB}$ domain and corpora of 5, 10, 15, 20, 25 and 30 hymns.

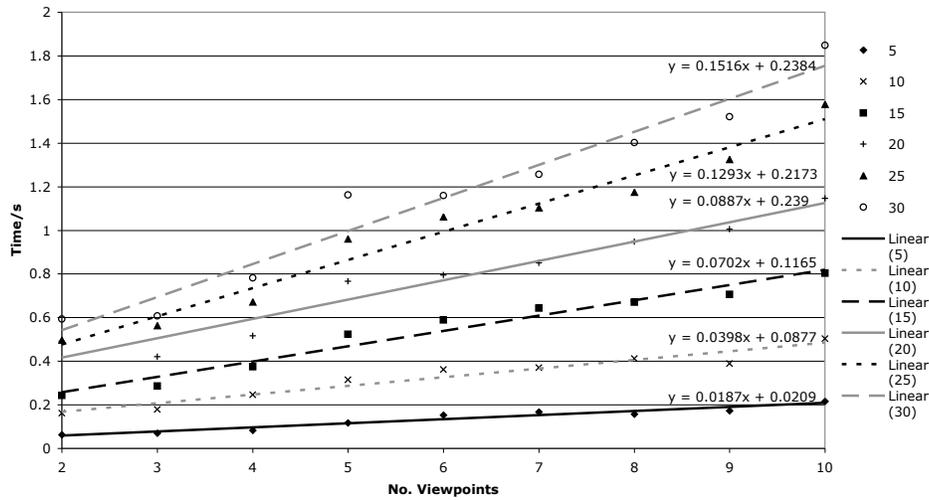


Fig. 19. Plot of number of viewpoints against time for the model construction phase of the program, which was run using the augmented $(Pitch)_{SATB}$ domain and corpora of 5, 10, 15, 20, 25 and 30 hymns.

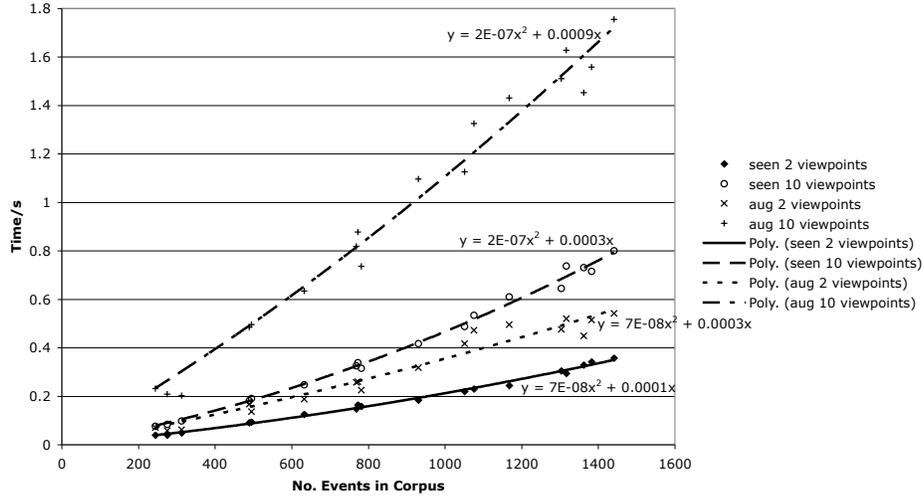


Fig. 20. Plot of number of events in the corpus against time for the model construction phase of the program, using seen and augmented (Pitch) $_{SATB}$ domains and multiple viewpoint system sizes of 2 and 10 viewpoints.

increasing number of viewpoints, and the augmented domain gives rise to longer run times than the seen domain.

If we define v to be the number of viewpoints, then from the above we conclude that the time complexity of the model construction phase of the program is $O(vn_c^2)$. In other words, run time is proportional to the number of viewpoints, and also proportional to the square of the number of events in the corpus. For the range of data analysed here, however, we can say that run time is approximately proportional to n_c .

Prediction Phase Figure 21 shows a plot of number of viewpoints against time for the prediction phase of the program, which was run using the seen (Pitch) $_{SATB}$ domain and corpora of 5, 10, 15, 20, 25 and 30 hymns. Once more, there is a reasonably good linear fit to the data, with increasing run time resulting from increasing number of viewpoints. The reason for this is that more prediction probability distributions need to be constructed and combined as the number of viewpoints increases. A similar plot for the augmented (Pitch) $_{SATB}$ domain (see Figure 22) shows a larger increase in run time with increasing number of viewpoints. Once more, two other data sets were also generated and analysed, with similar results.

Figure 23 shows a plot of the number of events in the the corpus against time (taken from the fitted lines on the plots described above). The data is best fitted using straight lines, which have been constrained to extend to the origin. Again, run time increases with the number of events in the corpus, and the

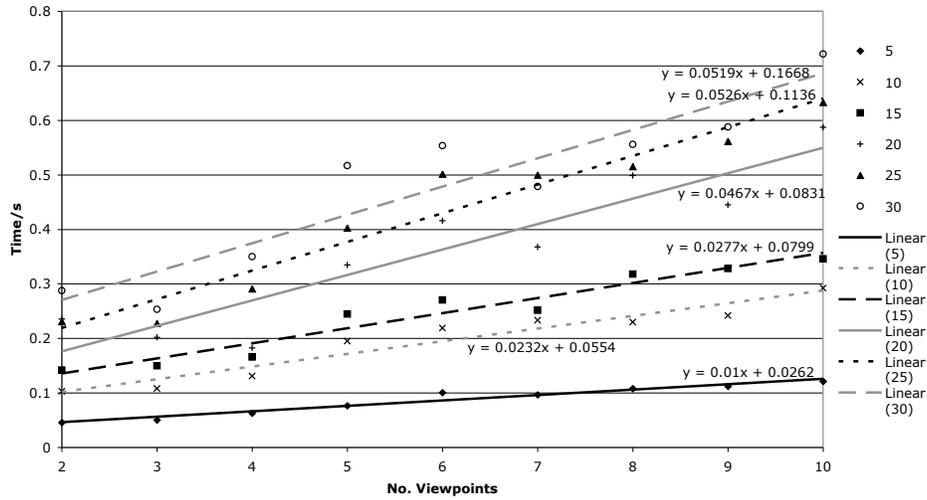


Fig. 21. Plot of number of viewpoints against time for the prediction phase of the program, which was run using the seen $(Pitch)_{SATB}$ domain and corpora of 5, 10, 15, 20, 25 and 30 hymns.

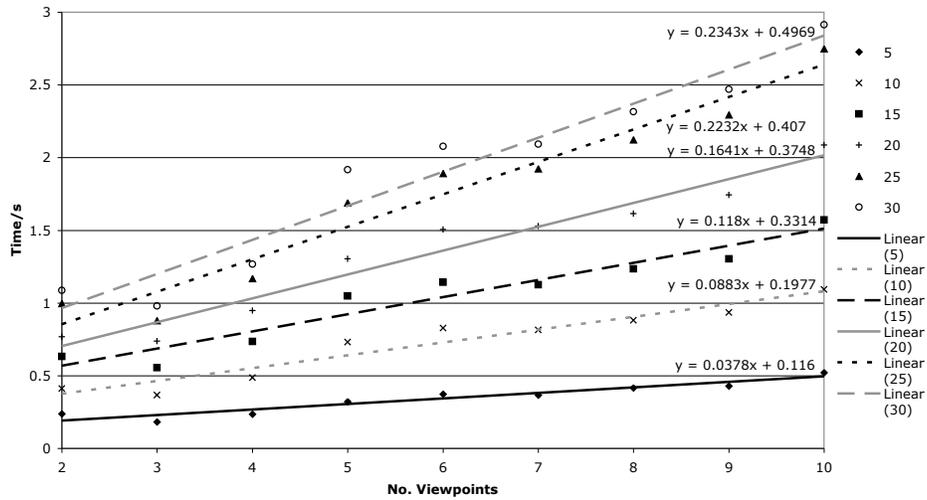


Fig. 22. Plot of number of viewpoints against time for the prediction phase of the program, which was run using the augmented $(Pitch)_{SATB}$ domain and corpora of 5, 10, 15, 20, 25 and 30 hymns.

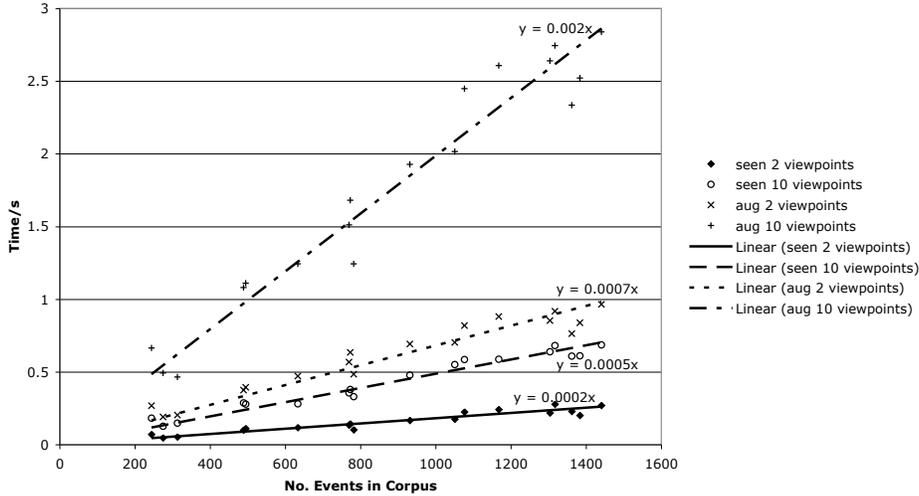


Fig. 23. Plot of number of events in the corpus against time for the prediction phase of the program, using seen and augmented ($Pitch$) $_{SATB}$ domains and multiple viewpoint system sizes of 2 and 10 viewpoints.

effect of domain type and number of viewpoints is similar to that in the model construction phase.

We can be fairly certain, without doing experiments, that the run time of this phase is proportional to the number of events in the test data being predicted. This, combined with the above analysis, results in a time complexity of $O(vn_{c}n_{td})$ for the prediction phase of the program. In other words, run time is proportional to the number of viewpoints, the size of the corpus, and the size of the test data.

Complete Prediction Run The time complexity of a complete prediction run is the same as the worst complexity of the three constituent phases, which is $O(vn_c^2)$ as for model construction.

9.4 Time complexity Analysis of Viewpoint Selection

For viewpoint selection, an important factor in determining run time is the number of primitive viewpoints v_p in a pool to be used in the stepwise optimisation of the multiple viewpoint system. At any viewpoint addition stage, single additions are made to the current multiple viewpoint system, and each of these new systems is evaluated using ten-fold cross-validation. In the limiting case, each primitive viewpoint in turn is added to the current system on its own and as part of a link with primitive viewpoints already in the system. This means that an evaluation is run the same number of times (at most) for each primitive

viewpoint in the pool, which in turn means that run time is proportional to v_p (with a very large coefficient). This is only approximately true, however, since the optimisation path can change dramatically with the addition of a primitive viewpoint to the pool; it could, for example, result in a different number of viewpoint addition stages, which cannot be predicted in advance. With this proviso in mind, the time complexity of a viewpoint selection run is $O(v_p v n_c^2)$, where v is the number of viewpoints in the ultimately selected multiple viewpoint system.

9.5 Time Complexity of Versions 2 and 3

Although no analysis has yet been done of the time complexity of versions 2 and 3, we anticipate that it will be broadly the same as for version 1 except for the additional factor of prediction in stages. If, for example, the alto, tenor and bass parts are predicted one after the other, the version 2 run time is likely to be approximately three times that for prediction of ATB together. For version 3, each successive prediction stage will take longer than the preceding one. Other than that, coefficients and constants are likely to be higher in versions 2 and 3 (especially 3), thereby contributing to longer run times.

10 Harmony Produced by Our Best Model to Date

Although the focus of this article is the alleviation of time complexity, and in particular its consequences for the reliable and efficient construction of domains, we have included four automatically generated hymn tune harmonisations to give a flavour of what the multiple viewpoint models can do. They were deliberately chosen to demonstrate a range of quality. All of these harmonisations were generated using the lowest cross-entropy model found so far, which is a version 3 long- and short-term model using the augmented `Pitch` domain generating B given S followed by AT given SB. Although a fifty-hymn corpus was used during viewpoint selection (for time complexity reasons), generation was undertaken using a hundred-hymn corpus.

Before looking at the harmonisations, we need to explain the concept of *probability thresholds*. The generation of harmony relies on the random sampling of prediction probability distributions. It was quickly very obvious to us, judging by the quality (or rather, the lack of it) of generated harmonisations, that a modification to the generation procedure would be required to produce reasonably good harmony. The problem was that random sampling sometimes generated a chord of very low probability, which was bad in itself because it was likely to be inappropriate in its context; but also bad because it then formed part of the next chord's context, which had probably rarely or never been seen in the corpus. This led to the generation of more low probability chords. The solution we proposed was to disallow the use of predictions below a chosen value, the probability threshold, defined as a fraction of the highest prediction probability in a given distribution. This definition ensures that there is always at least one usable prediction in the distribution, however high the threshold. Bearing in mind that

an expert musician faced with the task of harmonising a melody would consider only a limited number of the more likely options for each chord position, we considered the removal of low probability predictions to be a reasonable solution to the problem (unfortunately, low probability predictions known to be acceptable are disallowed along with those known to be unacceptable; better solutions will be sought in future work). The following harmonisations were all generated using $(\text{Dur})_{SB}$, $(\text{Dur})_{SATB}$, $(\text{Cont})_{SB}$ and $(\text{Cont})_{SATB}$ probability thresholds of 0.5; a $(\text{Pitch})_{SB}$ threshold of 0.2; and a $(\text{Pitch})_{SATB}$ threshold of 0.4.

A harmonisation of hymn tune *Bromsgrove* (Vaughan Williams 1933, hymn no. 144) is shown in Figure 24. Let us first examine what is wrong with this least successful of the four harmonisations. There are four parallel octaves within the first three bars, and another straddling bars 8 and 9. The change of harmony is too rapid in the first half of the second bar. There are unnecessary repeated notes in bars 5, 6, 7, 14 and 17; and the last chords of bars 13 and 16 are discordant. On the third beat of bar 5 the alto and tenor notes could be better chosen to make smoother chord progressions. In bar 11 there is a plagal cadence in the key of F major with an incorrect strong-to-weak metrical arrangement; and the piece does not end on a recognisable cadence. Although the harmony is generally weak, there are some good points. The modulation to the relative minor in the first phrase is interesting, although the appearance of an $F\sharp$ would have made it more convincing. The modulation to F major in the second phrase is reasonably well executed, ending with a solid ii^7b-V-I .

Figure 25 shows a slightly more successful harmonisation of hymn tune *Das ist meine Freude* (Vaughan Williams 1933, hymn no. 97). Although the harmony sounds more convincing, there is still plenty wrong with it. There is a parallel fifth in bar 12 and a parallel octave in bar 13 (both between tenor and bass). Similarly, there is a parallel unison straddling bars 5 and 6. There are unnecessary repeated notes in bars 4, 5, 8 and 12; and the last chord of bar 9 is discordant. There are unessential notes which are not properly resolved in bars 1 and 11. The soprano note on the first beat of bar 2 is below the preceding alto note; that is, the parts overlap, which is frowned upon. There is similar overlapping between tenor and bass in bars 7, 8, 9 and 11. The last chord of bar 13 (ignoring the auxiliary note) fulfils some of the criteria for a passing $\frac{6}{4}$, in that it is a metrically weak Ic approached by step; importantly, however, it is not quitted by step. There is not a proper cadence at the end of bar 6, and the final cadence is weak. On the other hand, most others are strong perfect cadences, with an imperfect cadence occurring at the end of bar 10. The ii^7b-I (in C major) at the end of bar 8 has the flavour of a plagal cadence, since the first chord can be heard as a subdominant chord with an added sixth. A well executed passing note at the end of bar 3 goes some way to disguising a parallel octave and a false relation. The use of modulation makes the harmony interesting. Starting in the tonic, a modulation to the dominant is established in bar 4. The music reverts to the tonic in bar 9, before modulating to the dominant again in bar 12. The piece ends in the tonic, as usual.



The image displays a musical score for the hymn tune 'Bromsgrove' in G minor, 3/4 time. The score is presented in three systems, each with a treble and bass staff. The first system (measures 1-6) features a melodic line in the treble staff and a bass line in the bass staff. The second system (measures 7-12) continues the melody and bass line. The third system (measures 13-18) concludes the piece with a double bar line. The key signature has one flat (Bb), and the time signature is 3/4. The melody consists of eighth and quarter notes, while the bass line is primarily composed of quarter and eighth notes.

Fig. 24. Automatically generated harmonisation of hymn tune *Bromsgrove* (Vaughan Williams 1933, hymn no. 144).

The image displays a musical score for the hymn tune "Das ist meine Freude" by Vaughan Williams (1933, hymn no. 97). The score is presented in a grand staff format, consisting of five systems of two staves each (treble and bass clefs). The key signature is one flat (B-flat major or D minor), and the time signature is common time (C). The melody is written in the treble clef, and the accompaniment is in the bass clef. The score is divided into measures, with measure numbers 4, 7, 10, and 13 indicated at the beginning of their respective systems. The music features a simple, homophonic texture with a clear harmonic structure.

Fig. 25. Automatically generated harmonisation of hymn tune *Das ist meine Freude* (Vaughan Williams 1933, hymn no. 97).

Fig. 26. Automatically generated harmonisation of hymn tune *Das walt' Gott Vater* (Vaughan Williams 1933, hymn no. 36).

Figure 26 shows an even more successful harmonisation, this time of hymn tune *Das walt' Gott Vater* (Vaughan Williams 1933, hymn no. 36). It is still far from perfect, however. There are four parallel unisons between the middle of bar 5 and the beginning of bar 7, and a parallel fifth between the second and third chords of bar 6. The alto and tenor notes cross unnecessarily in the fourth chord of bar 1; and there are two part overlaps in the second and third chords of bar 3. Notes are unnecessarily repeated at the beginning of bar 2 and the end of bar 6. An opportunity to modulate to the dominant is missed in the second phrase, which instead ends with an imperfect cadence. On the positive side, there is a good example of a passing note between the second and third beats of bar 2, which helps to justify the doubled major third (often frowned upon) on the third beat. The harmony is generally strong, albeit completely in the tonic, with two perfect and two imperfect cadences.

Figure 27 shows a reasonably good harmonisation (arguably the best of the four) of hymn tune *Innocents* (Vaughan Williams 1933, hymn no. 37), albeit that there are still some imperfections. There is a parallel octave between the second and third chords of bar 5, and another straddling bars 5 and 6. The doubled major third in the last chord of bar 5 is better avoided; and the last quaver (eighth note) of bar 6 is discordant. There are overlapping parts in bar 1 caused by the B3 in the bass (a B2 would solve the problem). Doubled major thirds



Fig. 27. Automatically generated harmonisation of hymn tune *Innocents* (Vaughan Williams 1933, hymn no. 37).

in the soprano and bass are fairly common in the corpus, as appear in the last chord of bar 3; but the following chord would sound better with an A3 in the bass. Much of this harmony comprises good, strong chord progressions (including cadences). It is made more interesting by a modulation to the dominant in the first phrase (although one would normally expect the tonic to be established over a longer period before modulating). There is a good example of a passing note in the alto in bar 6.

These are just a few examples of the many harmonisations that we have produced, using gradually improving models, during the course our research. These harmonisations vary widely in quality; but we have ideas for further improving the models, which we are working on. In this way, we hope to be able to produce consistently high quality harmony.

11 Conclusions and Future Work

The goal of extending the multiple viewpoint framework to the harmonic sphere is hampered by the computational complexity arising from the increased number of possible different musical events to be predicted. We have developed and tested a series of measures for making the problem tractable, allowing the implementation of new model features which contribute to the generation of improved harmonisations. In the context of the multiple viewpoint framework, a viewpoint domain is the set of valid elements (or symbols, or values) for a viewpoint, which is a means of representing a musical attribute such as pitch. We have discussed three issues affecting domains, and described how to construct them for increasingly complex viewpoints, culminating in a formal procedure for the construction of the most complex viewpoint domains.

Firstly, we have seen in §6.2 that very large domains, like that of $(\text{Pitch})_{SATB}$ in version 1, can be vastly reduced in size by only including elements seen in the corpus and test data, thereby greatly reducing run time. To take better account of elements as yet unseen, this domain can be augmented by chord transpositions occurring within the bounds of the known part ranges. If a part is given, such as the soprano in version 1, the domain is further constrained by the attributes of the given note.

Secondly, it has been known for some time with respect to melodic modelling that it is not possible in general to fix derived viewpoint domains: they need to be specially constructed before each prediction such that between them, the members of the domain are able to predict all of, and only, the members of the basic viewpoint domain (see §5.1). This article discusses the extension and implementation of this principle for harmony.

Thirdly, we have shown in §5.2 that linked viewpoint domains are not, in general, constructed simply by taking the Cartesian product of the constituent viewpoint domains. This is due to the correspondences between basic and derived viewpoints, and between the basic viewpoints Cont and Pitch (see §6.3): many of the tuples in the Cartesian product make no logical sense and can therefore be excluded.

In taking account of the above three issues whilst constructing domains, we must be careful not to introduce errors for the reasons outlined in §6.4. We have therefore described how to reliably construct domains for melody alone, and for three versions of the framework for representing and modelling harmony that we have developed (motivated by our aim to take better account of the complexities of four-part harmony). The domain construction method outlined for the most complex version has been implemented, and so far found to be suitably robust. The method can easily be extended beyond the prediction of three basic viewpoints.

The empirical analysis of the time complexity of version 1 of our computer model has demonstrated beyond doubt the utility, from a run time point of view, of reducing the Pitch domain from full to seen or augmented (see §9.2). In the initial domain construction phase (for basic viewpoints only), run time is proportional to the number of events in the corpus and test data for the seen domain, whereas there is a quadratic relationship for the augmented domain. This is because for each event in the corpus and test data, its Pitch element must, in the worst case, be compared with every element in the Pitch domain; and the size of the augmented domain increases at a much faster rate than that of the seen domain as the corpus and test data is traversed. The semitone by semitone transposition of newly seen chords further adds to the processing requirements for the augmented domain.

In the model construction phase, run time is proportional to the number of viewpoints, and is quadratic with respect to the number of events in the corpus, for both the seen and the augmented domains. The viewpoint relationship is entirely expected; but the behaviour with respect to the size of the corpus needs some explanation. To create a viewpoint model, an N-gram sized window incre-

mentally traverses the corpus. At each event, the partially constructed model must be traversed to check whether or not the N-gram context has been seen before. If it has, its count is updated; and if it has not, it is added to the model. The larger the context the rarer it will be, leading to a larger model (possibly approaching the size of the corpus); hence the relationship is bound to be worse than linear.

In the prediction phase, run time is proportional to the number of viewpoints, the size of the corpus and the size of the test data. For each event in the test data, each viewpoint model is traversed to find a match with the context, and a prediction probability distribution of (constrained) domain size is constructed. Domain size increases with corpus size (actually corpus size plus test data size; but we assume the test data size to be negligible) according to a sub-linear relationship (see Figures 10 and 11), and we would expect a similar relationship with respect to model size (context discovery is similar to domain element discovery). Since model traversal and domain construction occur in series, run time should be at worst linear with respect to corpus size.

In the limit, overall prediction run time (all three phases) is proportional to the number of viewpoints and the square of the number of elements in the corpus. Viewpoint selection run time is also proportional (with a large coefficient) to the number of primitive viewpoints employed in the stepwise optimisation of the multiple viewpoint system (see §9.4).

Finally, we have drawn attention to the strengths and weaknesses of a few example harmonisations produced by our best model to date. In the immediate future, we intend to implement other versions which push the development of the multiple viewpoint/PPM framework further, which we hope will enable us to produce consistently high quality harmony which is recognisably in the style of the learning corpus. In the light of the work described here, we do not anticipate any significant problems with the construction of domains for these proposed versions.

Bibliography

- Aha, D. W. and Bankert, R. L. (1996). A comparative evaluation of sequential feature selection algorithms. In D. Fisher and H. J. Lenz, editors, *Learning from Data: AI and Statistics V*, pages 199–206. Springer, New York.
- Allan, M. (2002). Harmonising chorales in the style of Johann Sebastian Bach. Master’s thesis, School of Informatics, University of Edinburgh.
- Allan, M. and Williams, C. K. I. (2005). Harmonising chorales by probabilistic inference. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press.
- Assayag, G., Dubnov, S., and Delerue, O. (1999). Guessing the composer’s mind: Applying universal prediction to musical style. In *Proceedings of the 1999 International Computer Music Conference*, pages 496–499, San Francisco. ICMA.
- Biyikoğlu, K. M. (2003). A Markov model for chorale harmonization. In R. Kopiez, A. C. Lehmann, I. Wolther, and C. Wolf, editors, *Proceedings of the 5th Triennial European Society for the Cognitive Sciences of Music (ESCOM) Conference*, pages 81–84.
- Cleary, J. G. and Teahan, W. J. (1997). Unbounded length contexts for PPM. *The Computer Journal*, 40(2/3):67–75.
- Cleary, J. G. and Witten, I. H. (1984). Data compression using adaptive coding and partial string matching. *IEEE Trans Communications*, COM-32(4):396–402.
- Clement, B. J. (1998). Learning harmonic progression using Markov models. EECS545 Project, University of Michigan.
- Conklin, D. (1990). Prediction and entropy of music. Master’s thesis, Department of Computer Science, University of Calgary, Canada.
- Conklin, D. (2002). Representation and discovery of vertical patterns in music. In C. Anagnostopoulou, M. Ferrand, and A. Smaill, editors, *Music and Artificial Intelligence: Proc. ICMAI 2002, LNAI 2445*, pages 32–42. Springer-Verlag.
- Conklin, D. and Cleary, J. G. (1988). Modelling and generating music using multiple viewpoints. In *Proceedings of the First Workshop on AI and Music*, pages 125–137. The American Association for Artificial Intelligence.
- Conklin, D. and Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73.
- Corman, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms*. MIT Press, second edition.
- Dixon, S. and Cambouropoulos, E. (2000). Beat tracking with musical knowledge. In W. Horn, editor, *Proceedings of the 14th Biennial European Conference on Artificial Intelligence (ECAI)*, pages 626–630, Amsterdam. IOS Press.
- Ebcioğlu, K. (1988). An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12(3):43–51.
- Hild, H., Feulner, J., and Menzel, W. (1992). Harmonet: A neural net for harmonizing chorales in the style of J. S. Bach. In R. P. Lippmann, J. E. Moody, and

- D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 4, pages 267–274. Morgan Kaufmann.
- Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc ICML*, pages 282–289.
- Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. MIT Press.
- 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.
- Pachet, F. and Roy, P. (1998). Formulating constraint satisfaction problems on part-whole relations: The case of automatic musical harmonization. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence (ECAI) Workshop: Constraint techniques for artistic applications*, Brighton, UK.
- Pearce, M. T. (2005). *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. PhD thesis, Department of Computing, City University, London.
- Pearce, M. T., Conklin, D., and Wiggins, G. A. (2005). Methods for combining statistical models of music. In Wiil, U. K., editor, *Computer Music Modelling and Retrieval*, pages 295–312. Springer, Berlin.
- Pearce, M. T., Müllensiefen, D., and Wiggins, G. A. (2008). A comparison of statistical and rule-based models of melodic segmentation. In *Proceedings of the 9th International Conference on Music Information Retrieval (ISMIR)*, pages 89–94, Philadelphia, USA. Drexel University.
- Pearce, M. T., Müllensiefen, D., and Wiggins, G. A. (2010). Melodic grouping in music information retrieval: New methods and applications. In Ras, Z. W. and Wieczorkowska, A., editors, *Advances in Music Information Retrieval*. Springer, Berlin.
- Pearce, M. T. and Wiggins, G. A. (2004). Improved methods for statistical modelling of monophonic music. *Journal of New Music Research*, 33(4):367–385.
- Pearce, M. T. and Wiggins, G. A. (2006). Expectation in melody: The influence of context and learning. *Music Perception*, 23(5):377–405.
- Pearce, M. T. and Wiggins, G. A. (2007). Evaluating cognitive models of musical composition. In Cardoso, A. and Wiggins, G. A., editors, *Proceedings of the 4th International Joint Workshop on Computational Creativity*.
- Pearce, M. T. and Wiggins, G. A. (2012). Auditory expectation: The information dynamics of music perception and cognition. *Topics in Cognitive Science*, 4(4):625–652.
- Phon-Amnuaisuk, S., Tuson, A., and Wiggins, G. A. (1999). Evolving musical harmonisation. In *Proceedings of the Fourth International Conference on Neural Networks and Genetic Algorithms (ICANN'99)*, Slovenia. Springer-Verlag.
- Phon-Amnuaisuk, S. and Wiggins, G. A. (1999). The four-part harmonisation problem: A comparison between genetic algorithms and a rule-based system.

- In *Proceedings of the AISB'99 Symposium on Musical Creativity*, Edinburgh. AISB Symposium.
- Ponsford, D., Wiggins, G. A., and Mellish, C. (1999). Statistical learning of harmonic movement. *Journal of New Music Research*, 28(2):150–177.
- Raphael, C. and Stoddard, J. (2003). Harmonic analysis with probabilistic graphical models. In *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR)*, pages 177–181, Baltimore, USA.
- Vaughan Williams, R., editor (1933). *The English Hymnal*. Oxford University Press.
- Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.
- Whorley, R. P., Rhodes, C., Wiggins, G. A., and Pearce, M. T. (2013). Harmonising melodies: Why do we add the bass line first? In Maher, M. L., Veale, A., Saunders, R., and Bown, O., editors, *Proceedings of the Fourth International Conference on Computational Creativity*, pages 79–86, Sydney, Australia.
- Witten, I. H. and Bell, T. C. (1989). The zero frequency problem: Estimating the probability of novel events in adaptive text compression. Technical Report 89/347/09, Department of Computer Science, The University of Calgary.

A Domain Construction Algorithms

A.1 Determination of Multipliers

Generalised algorithms for the determination of multipliers can be found in Algorithms 1, 2 and 3. As an example, consider a linked viewpoint with constituents derived from **Duration**, **Cont** and **Pitch**, and which also contains viewpoint **LastInPhrase** (derived from basic viewpoint **Phrase**). Algorithm input parameter *basic-type* is an array equivalent to the type set $\langle \tau \rangle$, which is $\{\text{Duration}, \text{Cont}, \text{Pitch}, \text{Phrase}\}$; and input parameter *basic-domain* is an array of corresponding domains. Let us assume that **Duration**, **Cont** and **Pitch** have domain sizes of 5, 10 and 40 respectively, and that we wish to determine the multipliers relating to **Duration** in the first instance. In this case, the input parameter k (the *basic-type* index for the current constituent viewpoint) has a value of 0. The inner-multiplier is obtained from Algorithm 2. In line 1, variable n is set to 3, which is the highest *basic-type* index. The predicate in line 2 is false; therefore line 4 is executed next. Algorithm 1 is called in this line, with input parameters *start* and *finish* set to 1 and 3 respectively. On line 1, variable *multiplier* is set to 1; and in the **for** loop beginning on line 2, counter i is initially set to 1. In line 3, there are 10 elements in the **Cont** domain; therefore *multiplier* is reassigned to 10. The **for** loop is executed again with i set to 2. In line 3, there are 40 elements in the **Pitch** domain; therefore *multiplier* becomes 400. The **for** loop is executed once more with i set to 3. In line 3, there is only one element in the **Phrase** domain (since **Phrase** attribute values are given); therefore the value of *multiplier* remains at 400. This value is returned to line 4 of Algorithm 2, where

Algorithm 1 Generalised algorithm for the determination of multipliers, in the style of Corman et al. (2001). It is used in both Algorithm 2 and Algorithm 3. A domain is an array of all viewpoint elements which could possibly be predicted. Input parameter *basic-domain* is an array of domains corresponding to the basic types in type set $\langle\tau\rangle$. Procedure SIZE() returns the number of elements in an array.

```

GET-MULTIPLIER(start, finish, basic-domain)
1  multiplier  $\leftarrow$  1
2  for i  $\leftarrow$  start to finish
3      multiplier  $\leftarrow$  multiplier  $\times$  SIZE(basic-domain[i])
4  return multiplier

```

Algorithm 2 Algorithm for the determination of inner-multipliers, in the style of Corman et al. (2001). A domain is an array of all viewpoint elements which could possibly be predicted. Input parameter *basic-type* is an array equivalent to the type set $\langle\tau\rangle$; input parameter *basic-domain* is an array of corresponding domains; and input parameter *k* is the *basic-type* index for the current constituent viewpoint. Procedure SIZE() returns the number of elements in an array; and procedure GET-MULTIPLIER() can be found in Algorithm 1.

```

GET-INNER-MULTIPLIER(k, basic-type, basic-domain)
1  n  $\leftarrow$  SIZE(basic-type) - 1
2  if basic-type[k] = basic-type[n]
3      then inner-multiplier  $\leftarrow$  1
4      else inner-multiplier  $\leftarrow$  GET-MULTIPLIER(k + 1, n, basic-domain)
5  return inner-multiplier

```

is assigned to variable *inner-multiplier*. The value 400 is returned in line 5. The outer-multiplier is obtained from Algorithm 3. The predicate in line 1 is true; therefore line 2 is executed next. In this line, variable *outer-multiplier* is set to 1. This value is returned in line 4.

By the continued application of these algorithms, for this example we can see that for any constituent viewpoint able to predict *Duration*, the inner- and outer-multipliers are 400 and 1; for any constituent viewpoint able to predict *Cont*, the inner- and outer-multipliers are 40 and 5; for any constituent viewpoint able to predict *Pitch*, the inner- and outer-multipliers are 1 and 50; and for any other constituent viewpoint (*LastInPhrase* in this case), the inner- and outer-multipliers are 1 and 2000 respectively.

Algorithm 3 Algorithm for the determination of outer-multipliers, in the style of Corman et al. (2001). A domain is an array of all viewpoint elements which could possibly be predicted. Input parameter *basic-type* is an array equivalent to the type set $\langle\tau\rangle$; input parameter *basic-domain* is an array of corresponding domains; and input parameter *k* is the *basic-type* index for the current constituent viewpoint. Procedure GET-MULTIPLIER() can be found in Algorithm 1.

```

GET-OUTER-MULTIPLIER(k, basic-type, basic-domain)
1  if basic-type[k] = basic-type[0]
2    then outer-multiplier  $\leftarrow$  1
3    else outer-multiplier  $\leftarrow$  GET-MULTIPLIER(0, k - 1, basic-domain)
4  return outer-multiplier

```

A.2 Domain Construction Procedure

A generalised algorithm for the construction of version 3 inter-layer linked viewpoint domains can be found in Algorithm 4. This algorithm will be illustrated by a modified real example: only a small subset of the actual **Pitch** domain is used, comprising note names such as Ab4 rather than MIDI numbers, in order to make the illustration more readily intelligible. We are predicting bass given soprano using, amongst other viewpoints,

$$(\text{Duration} \otimes \text{Pitch})_S \otimes (\text{Cont} \otimes \text{ScaleDegree})_{Bp}.$$

Input parameter *viewpoint-type* is an array of layers, which in turn are arrays of primitive viewpoints; *viewpoint-type* is therefore the 2-dimensional array

$$\begin{array}{rcc}
& & \begin{array}{cc} 0 & 1 \end{array} \\
\begin{array}{l} S \\ B \end{array} & \begin{array}{cc} 0 & 1 \end{array} & \begin{array}{cc} \text{Duration} & \text{Pitch} \\ \text{Cont} & \text{ScaleDegree} \end{array}
\end{array}$$

The set of basic viewpoints that the above's constituent viewpoints are derived from (type set $\langle\tau\rangle$) is $\{\text{Duration}, \text{Cont}, \text{Pitch}\}$. Input parameter *basic-type* is an array containing these three viewpoints in the order given, with indices 0, 1 and 2 respectively. For the purposes of this example, we assume that the **Duration** attribute of a bass note has already been predicted; therefore the **Duration** domain contains only one element:

$$\{\langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle \langle B, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle\}.$$

The next attribute to be predicted is **Cont**. The given soprano note has a **Cont** attribute of *F*; therefore the domain is constrained to two elements:

$$\{\langle\langle S, 0, \text{Cont}, \langle F \rangle \rangle \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle\},$$

$$\langle\langle S, 0, \text{Cont}, \langle F \rangle \rangle \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle\}.$$

The 5-element *Pitch* domain (constrained to have a soprano Ab4) is

$$\begin{aligned} &\{\langle\langle S, 0, \text{Pitch}, \langle \text{Ab4} \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{F2} \rangle \rangle\rangle, \\ &\quad \langle\langle S, 0, \text{Pitch}, \langle \text{Ab4} \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{Eb3} \rangle \rangle\rangle, \\ &\quad \langle\langle S, 0, \text{Pitch}, \langle \text{Ab4} \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{E3} \rangle \rangle\rangle, \\ &\quad \langle\langle S, 0, \text{Pitch}, \langle \text{Ab4} \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{F3} \rangle \rangle\rangle, \\ &\quad \langle\langle S, 0, \text{Pitch}, \langle \text{Ab4} \rangle \rangle \langle B, 0, \text{Pitch}, \langle \text{F\#3} \rangle \rangle\rangle\}. \end{aligned}$$

Input parameter *basic-domain* is an array containing the above three domains in the order given, with indices 0, 1 and 2 respectively.

The highest part (in terms of pitch) is dealt with first. If the viewpoint on this layer is, or contains, a basic viewpoint, each of the symbols or values belonging to that layer in the basic viewpoint domain is added, in turn, to what will become the inter-layer linked viewpoint domain, with repeats determined by the inner- and outer-multipliers. If the viewpoint is derived, the procedure is the same except that each basic viewpoint symbol is converted to the relevant derived viewpoint symbol.

In the **for** loop beginning on line 3, variable *layer* is initially assigned the array of viewpoints in the soprano layer. Having been initialised as -1 in line 2, counter *p* (an index indicating the layer) is incremented to 0 in line 5. In the **for** loop beginning on line 6, variable *constituent-viewpoint* is initially assigned *layer* element **Duration**. Having been initialised as -1 in line 4, *layer-constituent-count* is incremented to 0 in line 7. In line 8, variable *i* is assigned the *basic-type* index for **Duration**, which is 0. The return value of Algorithm 2, in this case, is the product of the **Cont** and **Pitch** domain sizes, which is 10. This value is assigned to variable *inner-multiplier* on line 9. Algorithm 3 returns the default value 1, which is assigned to variable *outer-multiplier* on line 10. In line 12, variable *outer-counter* is set to 1; and since there is only one element in *basic-domain*[0] (the **Duration** domain), variable *j* is set to 0 in line 13. Variable *basic-element* is assigned the single **Duration** element

$$\langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle, \langle B, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle$$

in line 14. Since the constituent viewpoint is **Duration**, in line 15 variable *derived-element* is assigned precisely the same element as *basic-element*. In line 16, in the first instance, *inner-counter* is set to 1. Having been initialised as -1 in line 11, variable *e* (an index indicating the inter-layer linked viewpoint domain element) is incremented to 0 in line 17. The predicate in line 18 is true; therefore line 19 is executed, which assigns $\langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle$ to array element *domain*[0][0]. Lines 16 – 19 (inclusive) are executed a further nine times, giving a provisional linked domain of

$$\begin{aligned} &\{\langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \\ &\quad \langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \\ &\quad \langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \end{aligned}$$

Algorithm 4 Generalised algorithm for the construction of version 3 inter-layer linked viewpoint domains, in the style of Corman et al. (2001). A domain is an array of all viewpoint elements which could possibly be predicted, which in turn are arrays of layers. Input parameter *viewpoint-type* is an array of layers, which in turn are arrays of primitive viewpoint types; input parameter *basic-type* is an array equivalent to the type set $\langle \tau \rangle$; and input parameter *basic-domain* is an array of corresponding domains. Procedure SIZE() returns the number of elements in an array; procedure GET-BASIC-TYPE-INDEX() returns the *basic-type* index for the current constituent viewpoint; procedures GET-INNER-MULTIPLIER() and GET-OUTER-MULTIPLIER can be found in Algorithms 2 and 3 respectively; procedure GET-DERIVED-ELEMENT() converts a basic viewpoint element into a derived viewpoint element; procedure INTRA-LAYER-ELEMENTS-COMPATIBLE() returns TRUE if a viewpoint symbol to be added to a layer portion of a domain element is logically compatible with a symbol (or symbols) already in it; and procedure MERGE-DOMAIN-ELEMENT-LAYER() links a viewpoint type and symbol with a type and symbol already in a layer portion of a domain element. Procedures REMOVE-UNDEFINED-ELEMENTS() and REMOVE-DUPLICATE-ELEMENTS() are self-explanatory.

```

CONSTRUCT-DOMAIN(viewpoint-type, basic-type, basic-domain)
1  domain  $\leftarrow$  initialise array
2  p  $\leftarrow$  -1
3  for layer  $\leftarrow$  viewpoint-type[0] to viewpoint-type[SIZE(viewpoint-type) - 1]
4    layer-constituent-count  $\leftarrow$  -1
5    p  $\leftarrow$  p + 1
6    for constituent-viewpoint  $\leftarrow$  layer[0] to layer[SIZE(layer) - 1]
7      layer-constituent-count  $\leftarrow$  layer-constituent-count + 1
8      i  $\leftarrow$  GET-BASIC-TYPE-INDEX(constituent-viewpoint)
9      inner-multiplier  $\leftarrow$  GET-INNER-MULTIPLIER(i, basic-type,
                                                    basic-domain)
10     outer-multiplier  $\leftarrow$  GET-OUTER-MULTIPLIER(i, basic-type,
                                                    basic-domain)
11     e  $\leftarrow$  -1
12     for outer-counter  $\leftarrow$  1 to outer-multiplier
13       j  $\leftarrow$  SIZE(basic-domain[i]) - 1
14       for basic-element  $\leftarrow$  basic-domain[i][0] to basic-domain[i][j]
15         derived-element  $\leftarrow$  GET-DERIVED-ELEMENT(constituent-
                                                         viewpoint, basic-element)
16         for inner-counter  $\leftarrow$  1 to inner-multiplier
17           e  $\leftarrow$  e + 1
18           if layer-constituent-count = 0
19             then domain[e][p]  $\leftarrow$  derived-element[p]
20             else if INTRA-LAYER-ELEMENTS-COMPATIBLE(
                domain[e][p], basic-element[p]) = TRUE
21               then domain[e][p]  $\leftarrow$  MERGE-DOMAIN-
                ELEMENT-LAYER(
                domain[e][p], derived-element[p])
22               else domain[e][p]  $\leftarrow$  "undef"
23   domain  $\leftarrow$  REMOVE-UNDEFINED-ELEMENTS(domain)
24   domain  $\leftarrow$  REMOVE-DUPLICATE-ELEMENTS(domain)
25   return domain

```

$$\langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle, \langle\langle S, 0, \text{Duration}, \langle 24 \rangle \rangle\rangle\}.$$

At this point, the provisional inter-layer linked viewpoint domain contains all of the elements we need (and probably more), albeit that they are incomplete. If there is a second basic or derived viewpoint associated with this layer (*i.e.*, a constituent of an intra-layer linked viewpoint), then the same procedure is followed except that the symbols are linked with the symbols already in the provisional domain, in the same order as the original additions to the domain.

The **for** loop beginning on line 6 is executed again, with *constituent-viewpoint* set to **Pitch**. In line 7, *layer-constituent-count* is incremented to 1; and in line 8, *i* is assigned the value 2 (the *basic-type* index for **Pitch**). Algorithm 2 returns the default value 1, which is assigned to *inner-multiplier* on line 9. The return value of Algorithm 3 is the product of the **Duration** and **Cont** domain sizes, which is 2. This value is assigned to *outer-multiplier* on line 10. In line 12, *outer-counter* is initially set to 1; and since there are five elements in *basic-domain*[2] (the **Pitch** domain), *j* is set to 4 in line 13. Variable *basic-element* is initially assigned the **Pitch** element

$$\langle\langle S, 0, \text{Pitch}, \langle \text{Ab4} \rangle \rangle\rangle, \langle\langle B, 0, \text{Pitch}, \langle \text{F2} \rangle \rangle\rangle$$

in line 14. Since the constituent viewpoint is **Pitch**, in line 15 *derived-element* is assigned precisely the same element as *basic-element*. In line 16, *inner-counter* is set to 1; and having been initialised as -1 in line 11, *e* is incremented to 0 in line 17. The predicate in line 18 is false; therefore line 20 is executed next. The predicate in this line can only possibly be false if the current layer contains both **Cont** and a viewpoint derived from **Pitch**. In this case, the predicate is true; therefore line 21 is executed, which replaces the contents of *domain*[0][0] with $\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle$. The **for** loop beginning on line 14 is executed a further four times (thereby running through the rest of the **Pitch** domain) before executing the **for** loop beginning on line 12 one more time. At this stage the soprano layer has been completed, giving a provisional linked domain of

$$\{\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle, \\ \langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle\rangle\}.$$

We then move on to the next layer, again adding to the elements already in the provisional domain, and so on, until all the layers have been dealt with.

There is only one other layer in the example, the bass part, which contains the intra-layer linked viewpoint $\text{Cont} \otimes \text{ScaleDegree}$.

The **for** loop beginning on line 3 is executed again, with *layer* assigned the array of viewpoints in the bass layer. Counter *p* is incremented to 1 in line 5; and in the **for** loop beginning on line 6, *constituent-viewpoint* is initially assigned Cont . Having been initialised as -1 in line 4, *layer-constituent-count* is incremented to 0 in line 7; and in line 8, *i* is assigned the value 1 (the *basic-type* index for Cont). The return value of Algorithm 2 is the Pitch domain size, which is 5. This value is assigned to variable *inner-multiplier* on line 9. Algorithm 3 returns 1 (the size of the Duration domain), which is assigned to variable *outer-multiplier* on line 10. In line 12, *outer-counter* is set to 1; and since there are two elements in *basic-domain*[1] (the Cont domain), *j* is set to 1 in line 13. Variable *basic-element* is initially assigned the Cont element

$$\langle\langle S, 0, \text{Cont}, \langle F \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle$$

in line 14. Since the constituent viewpoint is Cont , in line 15 *derived-element* is assigned precisely the same element as *basic-element*. In line 16, in the first instance, *inner-counter* is set to 1; and having been initialised as -1 in line 11, *e* is incremented to 0 in line 17. The predicate in line 18 is true; therefore line 19 is executed, which assigns $\langle B, 0, \text{Cont}, \langle T \rangle \rangle$ to array element *domain*[0][1]. Lines 16 – 19 (inclusive) are executed a further four times, and then the **for** loop beginning on line 14 is executed again (thereby processing the remaining Cont domain element). The provisional linked domain then becomes

$$\begin{aligned} &\{\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle T \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont}, \langle F \rangle \rangle\rangle\}. \end{aligned}$$

The final viewpoint to be added is ScaleDegree in the bass part; because it predicts Pitch , its inner-multiplier is 1, and its outer-multiplier is 2. There is a particular problem to be overcome with respect to the intra-layer linking of Cont with Pitch , or viewpoints derived from it, however. As we have seen in §6.3, there is an interaction between the Cont and Pitch domains; therefore the Cartesian product would contain pairings making no logical sense. In this case, as the Pitch domain is traversed, the Pitch symbol is checked against the relevant Cont symbol (which is already in the provisional domain): if the pairing makes logical sense, the Pitch symbol, or a symbol derived from it, is added to the element in the provisional domain as usual; if not, the element is tagged as

undefined. The previous note was F3 (MIDI value 53); therefore 53 is the only `Pitch` value which can be sensibly paired with a `Cont` value of T . The tonic is Eb; therefore F has a `ScaleDegree` value of 2.

The `for` loop beginning on line 6 is executed again, with *constituent-viewpoint* set to `ScaleDegree`. In line 7, *layer-constituent-count* is incremented to 1; and in line 8, i is assigned the value 2 (the *basic-type* index for `Pitch`). Algorithm 2 returns the default value 1, which is assigned to *inner-multiplier* on line 9. The return value of Algorithm 3 is the product of the `Duration` and `Cont` domain sizes, which is 2. This value is assigned to *outer-multiplier* on line 10. In line 12, *outer-counter* is initially set to 1; and since there are five elements in *basic-domain*[2] (the `Pitch` domain), j is set to 4 in line 13. Variable *basic-element* is initially assigned the `Pitch` element

$$\langle\langle S, 0, \text{Pitch}, \langle \text{Ab}4 \rangle \rangle, \langle B, 0, \text{Pitch}, \langle \text{F}2 \rangle \rangle\rangle$$

in line 14. This time, since the constituent viewpoint is `ScaleDegree`, in line 15 *derived-element* is assigned the element

$$\langle\langle S, 0, \text{ScaleDegree}, \langle 5 \rangle \rangle, \langle B, 0, \text{ScaleDegree}, \langle 2 \rangle \rangle\rangle.$$

In line 16, *inner-counter* is set to 1; and having been initialised as -1 in line 11, e is incremented to 0 in line 17. The predicate in line 18 is false; therefore line 20 is executed next. The predicate in this line is false, because F3 followed by F2 is not compatible with a `Cont` value of T (it is assumed that previous predictions have global scope; that is, they can be accessed from anywhere); therefore line 22 is executed, in which the contents of *domain*[0][1] are replaced with *undef*. The `for` loop beginning on line 14 is executed a further four times, thereby running through the rest of the `Pitch` domain. The predicate in line 20 is true on only one of these passes (when F3 is followed by F3), in which case the contents of *domain*[3][1] are replaced with $\langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle T, 2 \rangle \rangle$. The `for` loop beginning on line 12 is executed one more time, giving a provisional linked domain of

$$\begin{aligned} &\{\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \text{undef} \rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \text{undef} \rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \text{undef} \rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle T, 2 \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \text{undef} \rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 2 \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 0 \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 1 \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 2 \rangle \rangle\rangle, \\ &\langle\langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab}4 \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 3 \rangle \rangle\rangle\}. \end{aligned}$$

The provisional domain, constructed in this way in order to ensure that symbols which do not correspond with each other are not linked, may contain elements tagged as undefined, and may also contain duplicate elements; therefore

the final inter-layer linked viewpoint domain is achieved once undefined and duplicate elements have been removed (lines 23 and 24):

$$\{ \langle \langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle T, 2 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 0 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 1 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 2 \rangle \rangle \rangle, \\ \langle \langle S, 0, \text{Duration} \otimes \text{Pitch}, \langle 24, \text{Ab4} \rangle \rangle, \langle B, 0, \text{Cont} \otimes \text{ScaleDegree}, \langle F, 3 \rangle \rangle \rangle \}.$$