HO1

Automated Fugue Generation

by

Yu Yue Yue Yang

HO1

Advised by

Prof. Andrew Horner

Submitted in partial fulfillment
of the requirements for COMP398
in the
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
2010-2011

Date of submission: April 11, 2011

Table of Contents

1 Introduction

1.1 Overview

1.1.1 Introduction to Western Music

One interesting thing about music is that one can easily tell whether a piece of music is composed by a classical musician or randomly generated, even if he has no prior musical knowledge. Not everyone understands how to write a piece of music on a piece of paper, but this people do recognize repetition and sense direction of music without learning the concept of form and chord progression. All these examples suggest the existence of some characteristics in music, especially in classical Western music, which are essential to produce intellectual and emotional response. Composers have been developing various music theories from day one when music was produced. Particularly in western music, musicologists have coined terms like rhythm, melody and chord and describe the relationship between these basic elements. These efforts result in some general guidelines on composition, including compositional devices and techniques, and approval or disapproval of certain usage of musical elements.

1.1.2 Introduction to Computer Composition

When people have procedures and rules to generate music, it is natural to take advantage of computers to accelerate or even simulate the whole composition process. Depending on how and in which context computers are used in composition, such approaches can be named as computer composition, algorithmic composition, automatic music composition, etc.

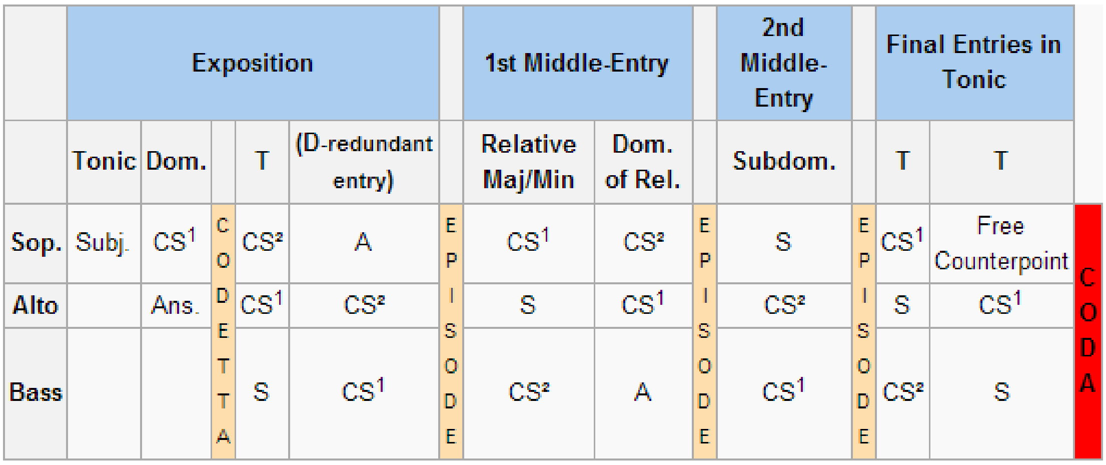
While those guidelines and concepts (such as counterpoint) are clearly defined, widely accepted and have been adopted for music composition education, it has never been easy to incorporate the musical knowledge into computer composition. This is because the vague guidelines are hard to be converted into precise code and they omit a large number of rules that most composers follow without realizing them. For instance, a classical musician will not add a non-diatonic note in his piece unless it can be convincingly explained by a grounded theory. Failing to state this point explicitly in a computer composition system might result in avant-garde masterpieces, as the system will hold the truth shared by Serialism composers to be self-evident that all pitch classes are created equal.

In addition, the developer of the computer composition system also needs to identify the basic units (chunks) that audiences perceive by chunking. Given that most people can only memorize 7 ± 2 chunks in short-term memory, a chunk usually contains several consecutive notes instead of one. Music that employs common harmony, regular rhythm and imitative melodies are much easier to listen to because the audiences can treat more notes together as one chunk, thus reducing their short-term memory demand. Of course the definition of chunks is highly subjective and often do not abide any notational boundaries like beats or the bar lines. This issue poses further challenges on the design of an encoding mechanism that is not only analogous to human perception of music but also convenient for meaningful manipulations.

1.1.3 Fugues and Inventions

A ***fugue*** is a polyphonic keyboard composition featuring imitative counterpoint based on one or more central and recurrent themes known as ***subjects***. Polyphony is to a type of musical texture where two or more independent voices retain their individual integrity while combining into a coherent musical unity. The science and art of how the voices with independent melodies and rhythms combine coherently is called counterpoint. In fugues this is achieved by ***imitation*** and the ***harmonic*** tension between voices

The general structure of a three-part fugue can be described as such (Fig 1-1):



	Exposition						1st Middle-Entry			2nd Middle-Entry		Final Entries in Tonic		
	Tonic	Dom.		T	(D-redundant entry)		Relative Maj/Min	Dom. of Rel.		Subdom.		T	T	
Sop.	Subj.	CS^1	CODETTA	CS^2	A	EPISODE	CS^1	CS^2	EPISODE	S	EPISODE	CS^1	Free Counterpoint	CODA
Alto		Ans.		CS^1	CS^2		S	CS^1		CS^2		S	CS^1	
Bass				S	CS^1		CS^2	A		CS^1		CS^2	S	

Fig. 1-1 Example of Key/Entry Structure in a Three-Voice Baroque Fugue

1) Exposition: At the beginning of the fugue, the subject is introduced in the tonic key by a single voice. Then it is reiterated by each of the voices in the pattern of subject, answer, subject, answer… until all the voices have been introduced. The ***answer*** has the same melody (sequence of melodic intervals) as the subject expect that it is transposed usually a fifth upward.

2) Development: A freer portion which generally avoids the tonic key. In particular, subjects almost always enter in the relative minor/major key and its dominant or subdominant key.

3) Recapitulation: Some reference to the subject in the tonic key near the end.

When a voice is active but not stating the subject, it either states the countersubject(s), or plays ***free counterpoint***, namely free material based on smaller chunks in the subjects. For some portion of the fugue, the subject may be absent and such a portion is called an episode.

An ***invention*** is like a fugue in that it is a polyphonic composition for a fixed number of voices featuring a basic recurring theme, counterpoint, episodes, a common set of transformations and the general artistic goals of creating a compelling narrative. Compared to a fugue, the recurring theme of an invention, named ***motive***, has simpler rhythm and implied harmonic background. In an invention, the imitative answer of the motive in an alternate voice occurs at the octave, instead of the fifth in a fugue. Thus the invention lacks a harmonic tension and complexity fundamental to the fugue, and is therefore easier to compose.

Since fugue and invention have rigorous contrapuntal and formal rules concerning imitation compared to other music styles, they are presumably suitable to be our foci of computer music generation.

1.1.4 Music Notation and Digital Symbolic Representation

1.1.4.1 Overview

The most common music notation system nowadays is sheet music (modern musical symbols structured by the five-line staves). People can be trained to interpret the content of the five-line staves, including but not limited to melody, rhythm and chord progression, within a reasonably short period of time.

When it comes to digital symbolic music representation whose purposes are playback, analysis and editing, using sheet music as media is apparently inappropriate. A variety of symbolic representation systems have been developed for different purposes.

1.1.4.2 MIDI

MIDI (Musical Instrument Digital Interface) is one of the most popular symbolic representations. Thanks to its binary format, a typical MIDI is usually compact and small. While compactness is critical for signal communication, MIDI requires extra libraries to read and write, and it fails to represent some of the musical information such as enharmonic equivalent tones, which are identical in sound yet different in terms of musical meaning. Additional functions are needed if manipulation of pitches is based on scale degree instead of semitones.

1.1.4.3 Humdrum and **kern Format

The *****kern*** format, a derivation of ***humdrum*** format, is a text-based musical representation originally designed for assisting in music research. **kern format is superior to MIDI for musical analysis because:

1) **kern format is a text-based musical representation. When the size of files is not the major concern for analysis, text-based format enables users to utilize text stream editors like ***grep***, ***gawk*** and ***sed***.
2) **kern format comes with a powerful utility set named ***humdrum toolkit***. With proper use of pipes and shell scripts, most of the analysis work can be done within a

hundred lines of code. This enables user to pick up statistically significant data for modeling.

3) **kern format lines up events that occur at the same time. This is particularly useful when users want to analyze the inter-voice attack patterns.

However, as stated by the author of humdrum toolkit, Prof. David Huron, Humdrum "is less well suited to individuals involved in creative (i.e. generative) musical activities -- such as composition". Without further enhancement it is impossible denote how a song is composed using small chunks of music segments.

1.1.4.4 Scheme Music Representation

The Scheme music representation is one of our major contributions in this project. Scheme is one of the two main dialects of the programming language Lisp. Its reliance on lists as data structures is particularly suitable for manipulation of musical chunks. In efforts to internalize the process of how the composer construct a piece of music by means of concatenation, repetition, inversion, transposition and modulation, Scheme music representation takes advantage of Scheme's list structure to show the relationship among all the chunks scattered throughout the music, as well as how the chunks are altered and reused.

This representation also leverages Scheme's ability to create and execute Scheme programs dynamically. An abstract structure of a musical segment, combined with atomic musical chunks, can be expanded into a normal **kern format file. In fact many different music representation formats can be defined in Scheme, of which our implementation is just an instance.

1.2 Objectives

This project aims to build a system that generates three-voice fugues or inventions, particularly in the style of the German composer Johann Sebastian **Bach** (1685–1750), one of the main composers of the Baroque period and one of the greatest composers of all time. The 48 fugues in Bach's ***Well-Tempered Clavier*** (WTC) will be our primary reference score because they are regarded as the greatest model of fugue by many composers and theorists. Although we constrain the musical style within Baroque, the user may alter the results to produce fugues with contemporary harmony by specifying other reference scores and chunks. Moreover, while this system is designed for fugues, some of the utilities developed in this project can be applied to other genres that feature imitations and counterpoint.

They system is modularized into two components: the analyzer and the generator. The analyzer reads in the user-provided score reference and extracts characteristics of the score reference in the form of statistics of musical information. The generator, based on these characteristics, constructs the form of the piece, generates the subject, incrementally fill in musical notes and eventually output the whole piece of polyphonic music.

1.3 Critical Review

1.3.1 Cornell’s Fugue Project

One predecessor of our project is an Artificial Intelligence Project from Cornell, which also a system to create original pieces of fugues from scratch. Cornell's procedure of fugal composition is two-fold: subject generation, and fugue generation from the subject.

A subject is encoded as a bit string where every 9 bits represents a sixteenth note. Each sixteenth note is encoded with pitch and articulation information. The evaluation functions for subject generation favor pitch smoothness, rhythm regularity, C Major scale tonality and moderate number and range of notes.

The rest of the fugue is the answer and the countersubjects. The answer is designed to be exactly the same as the subject but transposed a third downward. The countersubjects are subjects transposed a third upward, with its rhythm distorted to interleave interestingly with the subject rhythm.

1.3.2 Limitations of Cornell’s Project

In terms of data representation and regulations, Cornell's project is minimalistic: starting out with completely random notes in the initial population, it is not designed to rely on any concrete musical characteristics. The GA gene format and evaluators are very simple and non-specific to any particular style of music that exhibit imitative behavior.

However, this project from Cornell is largely incomplete from a musical perspective. Lacking the fractal structure or micro-macro relationship and a clear harmonic landscape, the results of Cornell's project can hardly be called a fugue. A Baroque fugue's answer is typically a fifth above the subject, not a third. Also, a fugue usually modulates to other related keys many times before finally in the tonic (home) key, stating the subjects in each local tonality. Staying mainly inside the C Major scale, Cornell's project does not cater the characteristics of tonal music such as chord progression, key center and modulation. Moreover, as we have attempted, genetic algorithms based on an unstructured sequence of notes fail to preserve any recurring musical phrases, which are essential for a fugue to exhibit musical meaning and thematic coherence.

2 Design

2.1 Fugue Generation

2.1.1 Framework

In our project, a fugue is divided into multiple bundles with equal time length. Each ***bundle*** consists of three melody segments that will sound simultaneously, corresponding to the three voices. The ***segments*** are single voice melodies of either the subject or free counterpoint. This equal-length segmentation is based on the assumption that

1) Every subject entry takes the same length, therefore the fugue will have no augmentation, diminuation or false entries;
2) Episodes, defined as the bridge portion between sequences of subjects, must take the same length as the subject; and
3) The subject must enter immediately after the completion of another subject entry or an episode.

2.1.2 Generation Procedure

We divide the problem of composing a fugue into multiple stages. Assuming a fixed form similar of a traditional three-voice fugue, we

1) Build the chord progression of the whole fugue marking where (in which bar, voice and key) the subject entries should be;
2) Find the subject with the best harmonic implication, melody and rhythm given the subject chord progression. Now that the subject is fixed, we replicate it at every subject entry position; and
3) Starting from the second bundle, incrementally complete each bundle taking into account the intra-voice quality of individual segment, harmonic coherency among segments and melodic connection to the previous bundle.

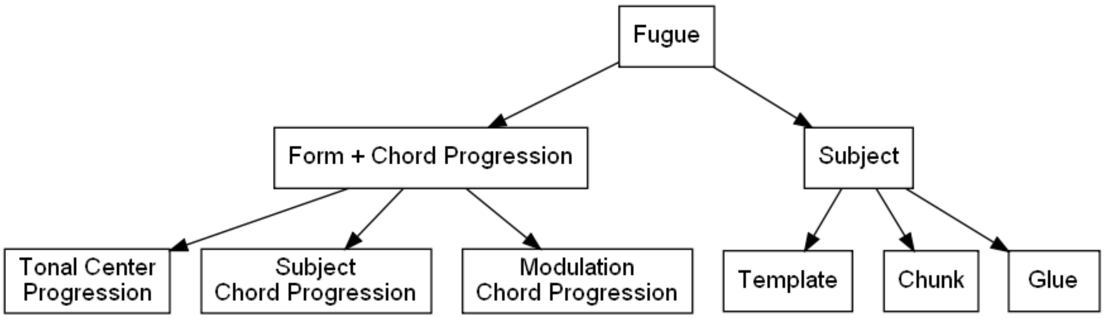


Fig. 2-1 Hierarchy of Musical Structures for Fugue Generation

Our decision to generate and fix the chord progression first and then generate the subject based on the chord progression can be justified by that

1) The chord progressions implied from Bach's fugues tends to stay within a rather restricted pattern, the most common progressions being a fifth upward, a fifth downward and a third downward. Subjects generated without having the chord progression in advance are often atonal, characterless or harmonically unsuitable for Baroque style fugues.

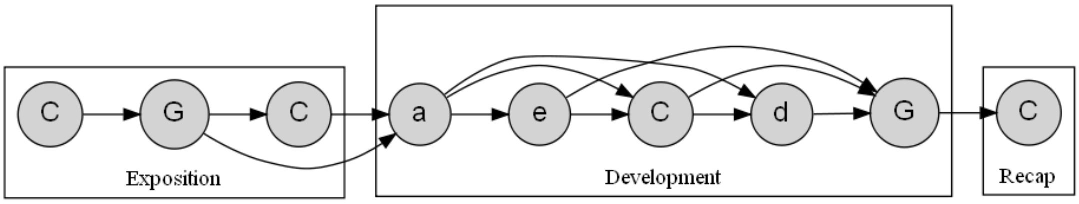
2) Many subjects composed by Bach have ambiguous harmonic implications: having no definite underlying chord, they can smoothly accommodate different chord progressions throughout the fugue. A good example is the subject of WTC Book I Fugue No.1 (BWV846). Composing the chord progression first provides the opportunity to design subjects that are compatible with multiple chord progressions.

2.1.3 Form and Chord Progression

The form and chord progression is the canvas of the fugue, which indicate the placeholders for subject entries and episodes and also provide harmonic context for the generation of melodies.

We structure the harmony, or chord progression of our fugue by the progression of ***tonal center*** (local tonic key). The chord progression generator is analogous to a Finite State Machine (FSM) where each state is a tonal center (Fig. 2-2). When the FSM arrives at a state (a tonal center), it emits the chord progression for a subject or the "subject – answer – subject" sequence. When the FSM transits to the next state, the transition emits the chord progression for an episode which modulates to the next tonal center. The resulting chord progression for the fugue is the sequence of chords emitted along the path from the beginning to the end.

2.1.3.1 Tonal center progression



on of the fugue.
' center.

Generalized from many of Bach's fugues in Major keys, we obtained a graph of the key centers Bach's fugue typically go through: Exposition starts from tonic and dominant, development starts from relative minor and its dominant and/or subdominant, and eventually the key center resolves back down along the circle of fifths to tonic. For simplicity we managed to arrange the graph as a 1D list with only forward links and no backward links, and the transition probability is evenly distributed at every state.

2.1.3.2 Subject chord progression

One hallmark of a good subject for a fugue is its rich harmonic implications. Our chord progression analysis of all fugues in major keys and simple meters in WTC

(Table 2-1) shows that eight out of the nineteen subjects share the harmonic implication of "I IV V I", and the others are variants that revolve around I and V.

For the sake of simplicity, we restrict the subject chord progression to be "I IV V I" ("i iv V i" in minor key centers) everywhere in the fugue, although generalization to other subject chord progressions is straightforward.

WTC Book	Fugue No.	BWV	Subject Chord Progression				
1	1	846		IV	I	V	I
1	3	848	I	V	I	V	I
1	5	850		**I**	**IV**	**V**	**I**
1	7	852		I	II	V	
1	9	854				V	I
1	11	856		IV	I	V	I
1	13	858		**I**	**IV**	**V**	**I**
1	17	862		I	IV	V	
1	23	868		V	I	V	I
2	1	870		I	IV		I
2	3	872		I		V	I
2	5	874		**I**	**IV**	**V**	**I**
2	7	876		**I**	**IV**	**V**	**I**
2	9	878		I		V	I
2	11	880		**I**	**IV**	**V**	**I**
2	13	882		**I_7**	**IV**	**V**	**I**
2	17	886	I	V	I	V	I
2	19	888		**I**	**IV**	**V**	**I**
2	23	892		**I**	**IV**	**V**	**I**

Table 2-1 Chord progressions of all major key, simple meter fugues in WTC.

2.1.3.3 Modulation chord progression

Bach's modulation techniques typically involve a combination of melodic sequence and common chord progressions. For each possible modulation (link between tonal centers), a few possible chord progressions are given with equal probabilities. As an example, in WTC Book I Fugue No.3 (BWV848), the recurring melodic sequence modulates from I to V by I – #iv°– iii – II – V, which is read as IV – vii°– vi – V_7 – I in the dominant key.

2.1.4 Subject

The length of subjects varies among the 48 fugues in WTC Book I & II. For simplicity we assume a subject is a special kind of segment, which is always two-bar long.

The subject is structured by the template, which instructs the sequence and repetition of chunks.

2.1.4.1 Template

When it comes to representation of music, it is natural to think of music as a sequence of notes. However, audience does not interpret what they hear in units of notes, nor do the composers think in this way. In order to raise audience's interest, it is imperative to use repetition at all levels throughout a piece, be it in big scale like ternary form or Rondo form, or just small scale repetition of motives.

A template indicates the form of a piece of music. In this project we propose a template representation called **Graph Meta Structure** using Scheme syntax, which is covered below.

2.1.4.2 Chunk

When a piece is moderately fast, note is a unit too small for human comprehension. This can be explained by a well-known fact that the average of short-term memory capacity is about seven chunks of information. If audience intend to memorize every note when they listen to fast-paced music, it is impossible to for them to follow the music without training.

Instead of notes, we use **chunk** as our basic unit of building blocks. In our project, a chunk is a group of 4 units of elements. Each element can be either a note or a rest, and shares the same duration. All the chunks are extracted from WTC Book I & II.

2.1.4.3 Glue

Glue is introduced to link up two adjacent chunks in order to sound like continuous. One of the easiest ways is to mandate the last attack of former chunk to be one degree higher or lower than the first attack of the latter chunk. In Music Analysis Section, we show that this assumption is actually championed by statistical data in WTC.

2.1.5 Bundle

A **Bundle** is a set of segments in different voices that are played in the same time. In this way we can examine if there is any infringement of counterpoint, dissonant interval, etc.

2.2 Enhanced Framework in Scheme Approach

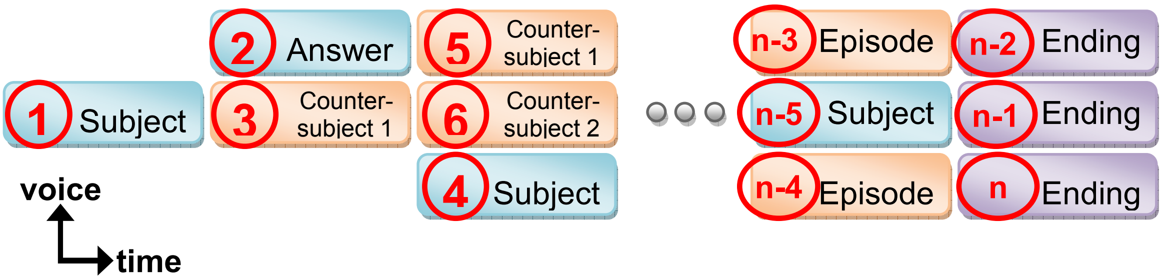


Fig. 2-3 Sequential Approach

We started from the sequential approach that composes one segment at a time, as shown in .The segment-based framework is a greedy approach that tries to maximize current segment without considerations of other segments that have not been computed but will influence the total score. For instance, in Fig. 2-3, steps 4, 5 and 6 should be regulated by counterpoint, yet over-emphasis on the firstly computed Segment 4 might leave not much room for other two segments to produce decent score in terms of counterpoint. This shortcoming is inherited from greedy algorithm when not all local solutions within a optimal global solution is optimal.

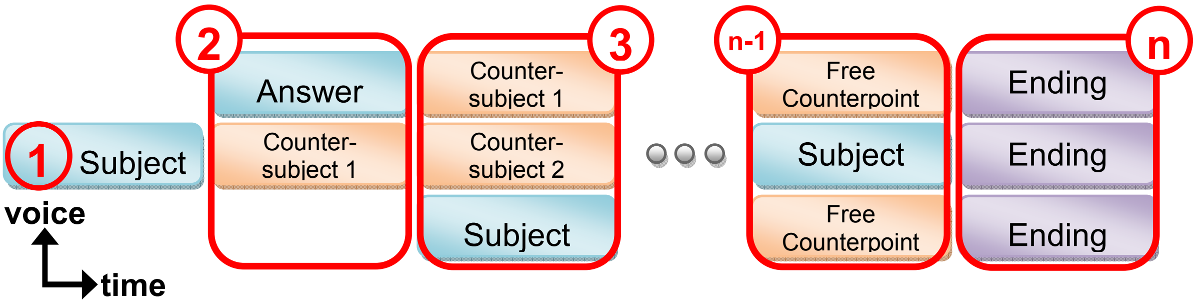


Fig. 2-4 Enhanced Framework

To address the problem, Scheme approach binds segments which are played at the same time into a bundle before submitting them to genetic algorithm. It is still a greedy approach, but the side-effect of greedy algorithm on counterpoint is reduced (see Fig. 2-4).

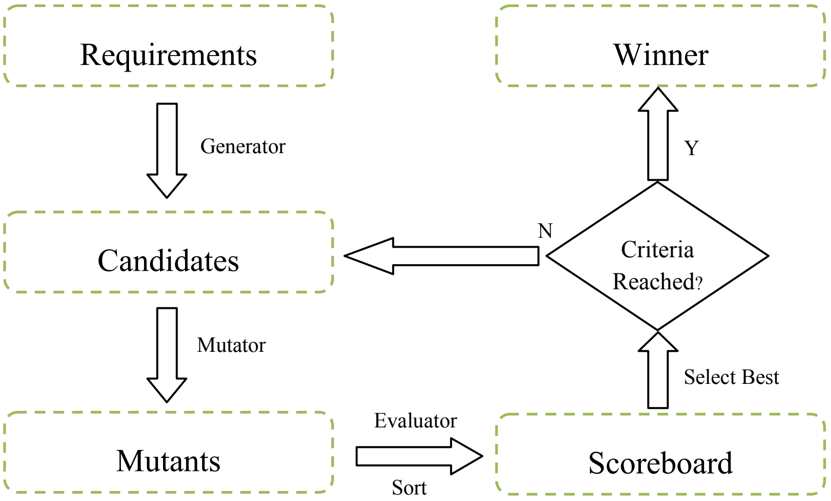
The framework is enhanced at the cost of dramatic increase of demand of computational time, as the number of possibilities that genetic algorithm need to handle are squared or cubed if two or three segments are bound as a bundle, respectively. While the running time is still acceptable for scheme approach, shell script approach does not benefit from the enhancement because the time consumption is too high.

2.3 Genetic Algorithm

For each musical segment (or segment bundle) generated in the process, a genetic algorithm is applied in order to obtain better, if not optimal result.

The context of the segment, i.e. the current tonal center, current chord progression, segments in other voices, or whether it is a subject, may have a major effect on how the segment is generated, mutated and evaluated. Therefore, the generator, mutator as well as evaluator will take an extra parameter containing all information related to the generation process.

Once the generator has obtained a fixed number of candidates, the mutator alters each candidate and generates mutants accordingly. The performance of each mutant is evaluated and compared, while the best of them are used as the candidates of next generation. The genetic algorithm process continues unless certain criteria are reached, such as maximum generation or high score obtained. In the end, the best candidate is selected as the winner.



2.4 Music Representation Formats

2.4.1 **kern Format

***kern format* is the central format of musical data in Humdrum Toolkit.

In **kern format, each line represents an instant and each column (named spine) represents a voice. Tokens (attacks, rests, articulations and other ornaments) in the same line denote concurrent events. If there is no attack (or beginning of a rest) in a voice, it will be denoted as a dot. A line that contains nothing but dots (an empty, uneventful line) is usually omitted. But still, if desirable, empty lines can be regenerated to uniformly resolve the musical data to certain time units (typically an eighth, a sixteenth or a thirty-second) simply by issuing ***timebase*** command from Humdrum Toolkit.

In **kern format, there are three predefined tokens: notes, rests and bar lines. A note is denoted as its duration and pitch. For instance, "**8c**" means an eighth note with a pitch of C4. The number of pitch class as well as capitalization indicates the octave. In addition, "**#**", "**n**" and "**-**" are used to represent accidentals. Rests follow the convention of note representation, and are represented as duration plus letter "**r**", such as "**16r**". A bar line is formatted as "**=x**", where "**x**" indicates the number of bar.

A typical kern file is as follows:

```
!! J.S. Bach, Fugue 2 WTC Book I
!! (3 parts), in c minor; BWV 847b
```

```
**kern **kern **kern
*M4/4  *M4/4  *M4/4
*MM72  *MM72  *MM72
*k[b-e-a-] *k[b-e-a-] *k[b-e-a-]
*c: *c: *c:
=1  =1  =1
1r  8r  1r
.   16cc    .
.   16bn    .
.   8cc .
.   8g  .
.   8a- .
.   16cc    .
.   16b .
.   8cc .
.   8dd .
=2  =2  =2
1r  8g  1r
.   16cc    .
.   16bn    .
.   8cc .
.   8dd .
.   16f .
.   16g .
.   4a- .
.   16g .
.   16f .
=3  =3  =3
1r  16e-    8r
.   16cc    .
.   16bn    16gg
.   16an    16ff#
.   16g 8gg
.   16fn    .
.   16e-    8cc
.   16d .
.   8c  8ee-
.   8ee-    16gg
.   .   16ff#
.   8dd 8gg
.   8cc 8aan
=4  =4  =4
*-  *-  *-
```

Which is an excerpt from BWV 847 By J.S. Bach:

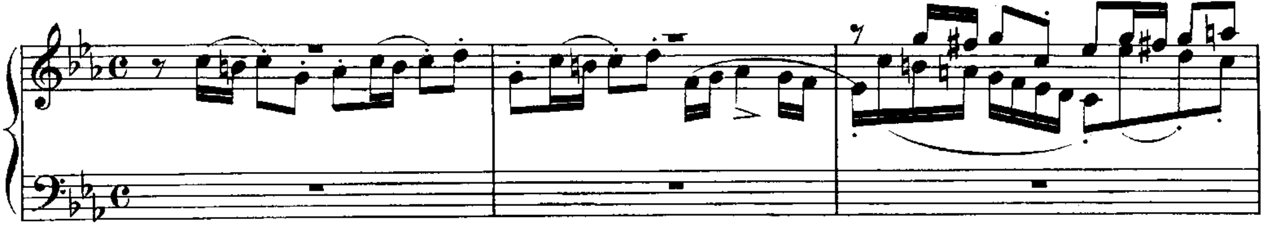


Fig. 2-5 Corresponding score of example Kern file (J.S. Bach, Fugue 2 WTC Book I)

A comprehensive introduction to ***kern format* can be found at http://www.lib.virginia.edu/artsandmedia/dmmc/Music/Humdrum/kern_hlp.html .

2.4.2 Scheme Music Representation

2.4.2.1 Vertex, Edge, Ring and Graph

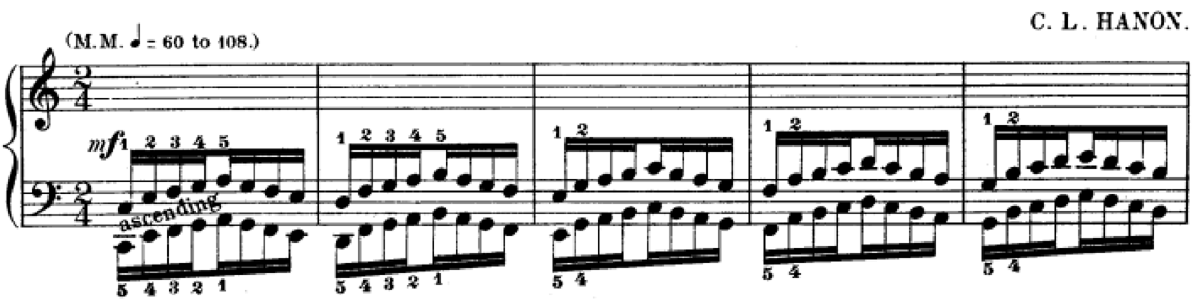
In Scheme music representation, a ***Graph*** is a list of events records what happens in a specific time frame, as well as the relationship between current time frame and others. There are four predefined statuses:

1. Symbol ***k*** represents an attack whose pitch height is unknown.
2. Symbol ***r*** represents a non-attack, either a rest or an elongation of a previous note.
3. Symbol ***?*** represents complete uncertainty at that timestep. The ***?*** symbols typically act as placeholders padding segments that do not begin or end on beats.
4. A string (e.g. ***"cc#"***) represents an attack with a confirmed pitch. In scheme representation, **kern representation of pitch is adopted.

All these four statuses are called ***Vertex*** in scheme representation. The uncertain property of ***k*** and ***?*** provides flexibility for pattern matching.
In order to denote how a note has an influence on another note, ***Edge*** is introduced. A vertex can have none, one or multiple Edges. The first element is a positive integer indicating the offset of the affected Vertex (except when it is 1, it is defined to be the next attack rather than next Vertex), and the second element indicates the offset in terms of degree. A tuple that contains exactly one Vertex and multiple Edges is called a ***Ring***. A tuple comprised of multiple Rings is called a ***Graph***.
For example, the following Graph:
((***k*** (2 1) (4 1)) (***r***) (***k***) (***r***))
indicates a relationship between the first Vertex and the third Vertex. In particular, the pitch of the third Vertex should be one degree higher than the first one. Since the Graph in the example has only 4 Rings, the second Edge (4 1) of the first Ring does not have any effect. Such Edge is called an external Edge in comparison with internal Edge, where the edge is effective within a Graph.

2.4.2.2 Graph Structure

A ***Graph Structure*** is an algebra system of Graph representation. Consider the following excerpt from The Virtuoso-Pianist No. 1, By C.L. Hanon:



An experienced music student may immediately discover that every bar is a simple transposition of the previous bar. This can be described in a Graph:

((***k*** (1 2) (8 1)) (***k*** (1 1)) (***k*** (1 1)) (***k*** (1 1)) (***k*** (1 -1)) (***k*** (1 -1)) (***k*** (1 -1)) (***k***))

The Edge “(8 1)” in the first Ring indicates how it is transposed. At this point an operator “**G+**”, whose primary functionality is simply putting several Graphs into one, is introduced. To denote the right hand part in this excerpt, we have:

```
(G+
  '((k (1 2) (8 1)) (k (1 1)) (k (1 1)) (k (1 1)) (k (1 -1)) (k (1 -1)) (k (1 -1)) (k))
  '((k (1 2) (8 1)) (k (1 1)) (k (1 1)) (k (1 1)) (k (1 -1)) (k (1 -1)) (k (1 -1)) (k))
  '((k (1 2) (8 1)) (k (1 1)) (k (1 1)) (k (1 1)) (k (1 -1)) (k (1 -1)) (k (1 -1)) (k))
  '((k (1 2) (8 1)) (k (1 1)) (k (1 1)) (k (1 1)) (k (1 -1)) (k (1 -1)) (k (1 -1)) (k))
  '((k (1 2) (8 1)) (k (1 1)) (k (1 1)) (k (1 1)) (k (1 -1)) (k (1 -1)) (k (1 -1)) (k)))
```

which is redundant and clumsy. To simplify the notation, we introduce multiplication operator “**G***”, which gives us equivalent but much shorter formula:

```
(G*
  '((k (1 2) (8 1)) (k (1 1)) (k (1 1)) (k (1 1)) (k (1 -1)) (k (1 -1)) (k (1 -1)) (k)) 5)
```

In order to be more automated, notice the number “8” in the transposition edge is calculated from the length of the Graph. Therefore ***Gmf*** operator is introduced to add an transposition edge:

```
(G* (Gmf
  '((k (1 2)) (k (1 1)) (k (1 1)) (k (1 1)) (k (1 -1)) (k (1 -1)) (k (1 -1)) (k)) 1) 5)
```

More documentation is available in appendices. However, the examples above should have demonstrated how to use Graph Structure representation to minimize the number of variables without compromising the possibility of meaningful music excerpts.

2.4.3 Genetic Algorithm Data Format

Neither of the two musical formats mentioned above is easy to manipulate in genetic algorithm. The main function of genetic algorithm needs to locate the variables in the text, identify the type of the variable, and get the current value of the variable for mutation purpose.

Hence, it is imperative to separate the original file and variable settings. In shell script approach two type of files are used: the template file and the variable file. Scheme approach simply splits these two types of data and stores them into one tuple.

A template and variable couple example is as followed:

Template File

```
**kern
*C:
=1
16cn
16r
16r
16<VAR00001>
16r
16<VAR00002>
16r
16<VAR00003>
16r
16<VAR00004>
16r
16r
16r
16<VAR00005>
16r
16<VAR00006>
=2
16r
16<VAR00007>
16<VAR00008>
16<VAR00009>
16<VAR00010>
16r
16<VAR00011>
16r
16<VAR00012>
16r
16<VAR00013>
16<VAR00014>
16<VAR00015>
16<VAR00016>
16<VAR00017>
16r
=3
16cn
==
```

Variable File

```
VAR00001      $DATABASE/scripts//generators/pitchMutate     c
VAR00002      $DATABASE/scripts//generators/pitchMutate     e
VAR00003      $DATABASE/scripts//generators/pitchMutate     c
VAR00004      $DATABASE/scripts//generators/pitchMutate     B
VAR00005      $DATABASE/scripts//generators/pitchMutate     d
```

```
VAR00006      $DATABASE/scripts//generators/pitchMutate     B
VAR00007      $DATABASE/scripts//generators/pitchMutate     e
VAR00008      $DATABASE/scripts//generators/pitchMutate     g
VAR00009      $DATABASE/scripts//generators/pitchMutate     a
VAR00010      $DATABASE/scripts//generators/pitchMutate     cc
VAR00011      $DATABASE/scripts//generators/pitchMutate     a
VAR00012      $DATABASE/scripts//generators/pitchMutate     d
VAR00013      $DATABASE/scripts//generators/pitchMutate     c
VAR00014      $DATABASE/scripts//generators/pitchMutate     A
VAR00015      $DATABASE/scripts//generators/pitchMutate     B
VAR00016      $DATABASE/scripts//generators/pitchMutate     G
VAR00017      $DATABASE/scripts//generators/pitchMutate     e
```

In scheme representation it is very similar to shell scripts approach. Below is a template and variable tuple (hereinafter ***Graph Meta Structure*** or ***GMS***) example:

```
(
  (Gdx (G+
          (Gml
            (Gdx (g*
                    (var4 (Gdx (Gdx (G* (var2 (Gdx var1) var3)
                                            2)))
                           var5)
                    3))
            var6)
          (Gdx (G*
                        (var8 (Gdx var7) var9)
                        2))))

  (

    (var1   atm4      '((k (1 2)) (k (1 -1)) (k (1 -1)) (k)))
    (var2   glu-type  Gml)
    (var3   rep-ofst  1)
    (var4   glu-type  Gml)
    (var5   rep-ofst  1)
    (var6   glu-ofst  -1)
    (var7   atm4      '((k (1 -1)) (k (1 -1)) (k) (r)))
    (var8   glu-type  Gml)
    (var9   rep-ofst  -1)))
```

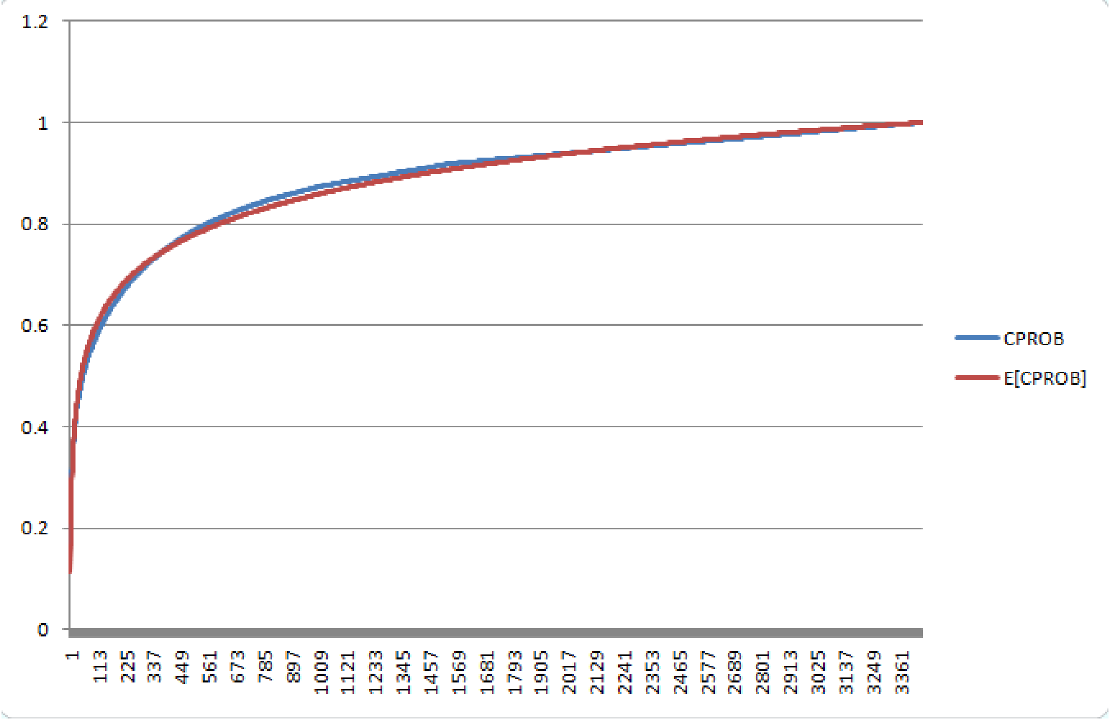
Notice that a GMS can be converted into a Graph Structure by substitution.

3 Music Analysis

In this project, we primarily use WTC Book I & II as our database. We have identified several critical characteristics on how J.S. Bach tends to write a fugue, some of which are not available in books of related topics.

3.1 Zipf’s Law

One of the curious discoveries is that statistics of many musical properties, however implicit, meet Zipf's law astoundingly well. Zipf's law is a strong assertion that many physics or social science data can be approximated by generalized harmonic number. In particular, when we investigate the distribution of chunks in WTC Book I & II, the distribution is almost identical to Zipfian distribution:



fugues

The y-axis in the figure shows the cumulative probability of chunks, and the x-axis represents ranking. Red curve represents a standard Zipfian distribution while the blue one is the chunk data.

Under this law, a stochastic generator of chunk based on cumulative distribution function is very easy to implement. The generator may simulate Zipfian distribution by selecting the $(n+1)^r$-th item, where n is the cardinality of the library and r is a random number between 0 and 1. A function **zipf-cond** in this project takes advantage of the convenience provided by Zipf's Law.

3.2 Intra-Voice Characteristics

3.2.1 Probability Distribution of Scale Degrees

One of the major characteristics of classical music is tonality. Tonality is the hierarchical structure of the twelve tones in an octave; it gives the sense of direction toward certain notes or chords in tonal music. Since all the fugues have been normalized to C Major / a minor, the profile of solfège frequency depicts Bach's preference of notes. From Fig. 3-2 we observe that notes in the C Major scale dominate in the distribution. Bach rarely use accidentals without compelling reasons:

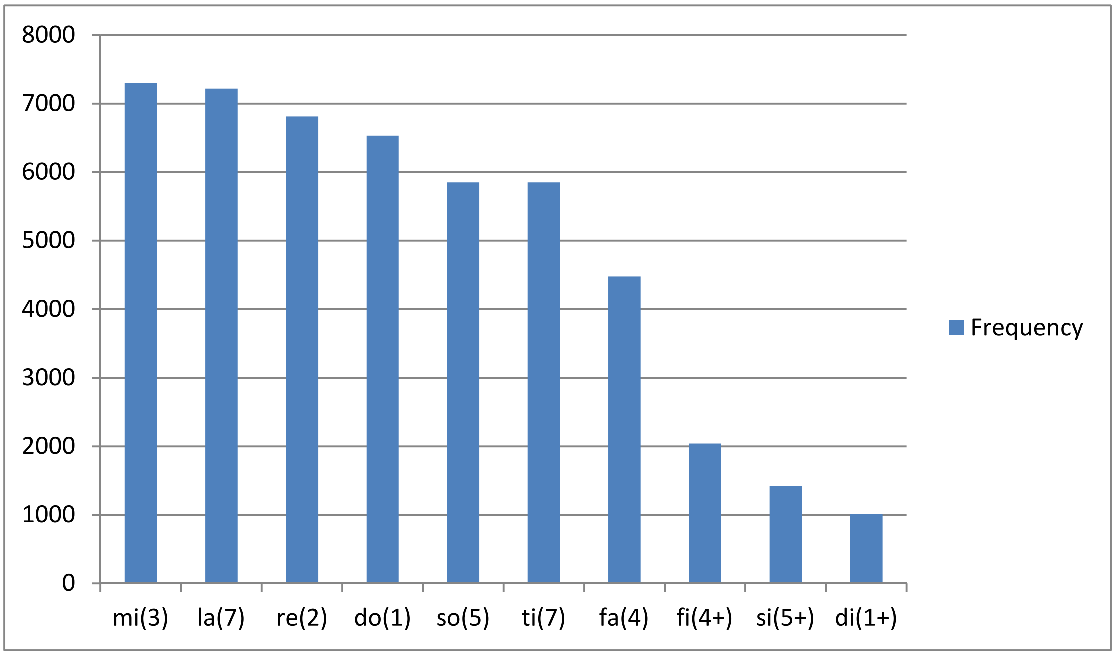


Fig. 3-2 Frequency of Solfège (scale degree) unigram in WTC fugues

The scale degrees G#, F# and C# are the leading tones of the harmonic minor scale, the dominant key and the secondary dominant key, respectively; in WTC they are often deliberately employed for the purpose of modulation and establishing new tonal centers.

3.2.2 Probability Distribution of Melodic Intervals

The statistics of single-voice melodic intervals (disregarding interval quality and up/down direction) in Bach's WTC Book I & II fugues bring a striking revelation:

From Fig. 3-3, we see among all the melodies of voices of the fugues, 68% of melodic intervals are seconds, namely stepwise motions. Unisons, corresponding to repeated attacks, are extremely rare (4%) compared to thirds (12%), namely skips. Indeed, Bach exploited the smoothing value of stepwise motions: the dominance of second intervals greatly reduces the amount of information given to the audiences by going smoothly up and down a musical scale, while scales are a wonderful opportunities for distraction to audiences, harmonic ambiguity and modulation in Bach's fugues.

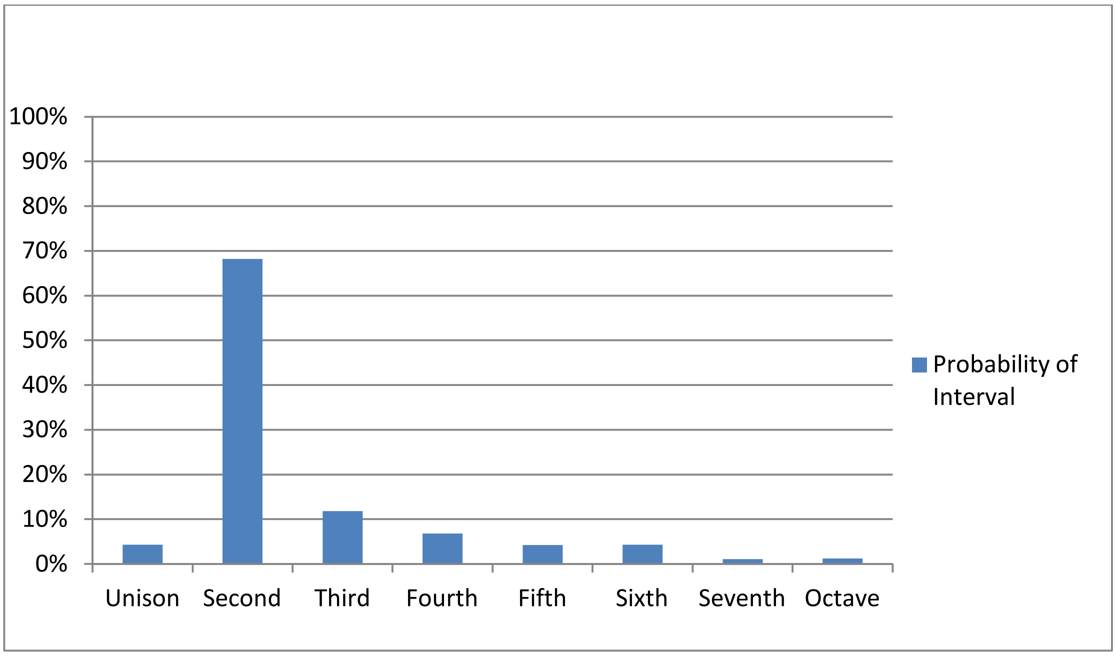


Fig. 3-3 Frequency of Melodic Interval unigram in WTC fugues

3.3 Inter-Voice Characteristics

While maintaining integrity of each of the three voices, Bach managed to reduce the listeners' memory burden by reducing the amount of concurrent attacks, which is his art of rhythmic counterpoint. From the statistics of 3-voice joint rhythm pattern below we clearly see that Bach tried to contrast note density between voices, the most common joint rhythm being repetition of ratio 1:2:4. Many other interleaving joint patterns have high robability rankings.

3-voice Joint Rhythm Pattern	Negative Logarithm
xooo xoxo xxxx	3.476
ooxo xooo xxxx	4.174
xooo xooo xoxo	4.384
xoxo xoxo xxxx	4.471
ooxo xoxo xxxx	4.481
ooxo xooo xooo	4.543
oxxx xooo xoxo	4.630
ooxo xooo xoxo	4.652
xooo xoxo xoxo	4.900

ooxx xooo xxxx	4.997

Table 3-1 Top ten 3-voice Rhythm Pattern in Well-Tempered Clavier I & II Fugues

Each “xxxx” represent a pattern in one voice. “x” represents an attack, “o” is a non-attack.

4 Implementation

4.1 Analyzer: Humdrum Toolkit on Shell Scripts

4.1.1 General Information

Humdrum toolkit is used to analyze WTC Book I & II. In order to analyze some musical properties that Humdrum toolkit does not provide corresponding tools to calculate, some extra extensions have been implemented as well.

4.1.2 Preprocessing

Although the scores of WTC fugues in the original database downloaded from [CITE] are already in **kern format, additional cleansing is required because Bach tends to increment the number of voices at the end of the fugue, which complicates analysis and processing of the files.

In addition, all the fugues are transposed into C Major / a minor. This has several advantages:

1) Scale degree system and solfège system are unified both in Major and minor pieces of music (for example, do=1, re=1, mi=3, fi=4+, te=7-). Humdrum toolkit provides extra functionality in program ***degree*** compared to program ***solfa***.
2) Pitch statistics (compared to pitch class statistics) now reflect the tonality of all the fugues.

Finally, for intra-voice statistics, programs run much faster if multiple voice songs are split into files that contains single spine. Hence a single-voice library of the database is created for cache use.

4.1.3 Statistics Generator

To extract principal characteristics of Bach’s fugual compositional style, the occurring **probabilities** of certain musical elements and patterns are calculated from the preprocessed data, saved into negative logarithm (amount of information) and then sorted in descending probabilities.

The general process is:

1) using ***solfa***, ***degree***, ***timebase***, and other utilities to generate desired unigram sequences of solfège, scale degree, rhythm and their combinations;
2) using ***context*** to generate N-gram; and
3) using ***infot*** to calculate the distribution.

The following data have been calculated:

1) Pitch / Pitch Class N-gram
2) Melodic Interval N-gram
3) Rhythm Block N-gram
4) (Melodic Interval + Rhythm) Block N-gram
5) Degree Inter-voice Unigram
6) Rhythm Block Inter-voice Unigram
7) Downbeat degree Inter-voice Bigram

Below is an example of pitch class unigram statistics.

Degree	Negative Logarithm
3	2.783
6	2.799
2	2.883
1	2.943
5	3.102
7	3.103
4	3.488
4+	4.620
5+	5.147
1+	5.628

Table 4-1 Top 10 Degree Distribution in Well-Tempered Clavier I & II Fugues

4.2 Note-Based Generators with Shell Scripts

As continuation of the efforts of analysis based on Humdrum toolkit and shell scripts, we implement generator on the same platform. We soon find out that building a generator on shell scripts is not cost-effective in terms of implementation time and performance. Text-based format complicates the process of data manipulation of high degree because of lack of abstraction. For example, to implement a function that can transpose notes in text-based format requires several days of work if use other than common GNU utilities is not allowed.

We also encounter unexpected problems when we are using Humdrum toolkit, for the toolkit was implemented in 1990s, and the programs it relies on have changed a lot in the past decade. One of the bugs we identify and fix is that the regular expression in ***awk*** behaves differently in the mainstream machines in 1990s compared to now, as Unicode was not common in that period, contrary to its common presence in *nix machines nowadays.

Finally, shell scripts are extremely slow. This is partially due to its nature as an interpreted language. However, compared to our final resort, scheme, which is also an interpreted language, the performance of shell scripts is still frustrating. Since the core of our algorithm is Genetic Algorithm, whose principle is to trade large amount of computation and time for quality, Shell script is by no means an acceptable choice.

4.3 Chunk-Based Generator with Scheme

4.3.1 Procedure of generation

After withdrawal from Shell scripts, we turn to scheme for an all-in-one solution to fugue generation, including representation and generation. As stated in Section 2.2, the procedure of generation is a bundle-by-bundle approach. At the very beginning the global environment of the fugue, including tonal center progression, chord progression as well as subject is defined, followed by calculating an optimal bundle that fits its predecessor best one by one, until hitting the coda of the fugue. It is greedy to the extent that every bundle is fixed once it is calculated, but the algorithm for searching the optimal solution to every bundle is genetic algorithm due to massive search space.

4.3.2 General Information

Currently we restrict our fugal composition to have the home key of C Major and a simple binary meter (4/4 or 2/4). It will be straightforward to generalize to all Major and minor keys through direct transposition and the relative minor keys, while generalization to compound meters like 3/4, 6/8, 9/8, etc. requires additional work on the capture of rhythmic characteristics.

4.3.3 Subject Generator

As a genetic algorithm approach, we randomly generate 100 **Graph Meta Structures (GMS)** for selection. A mutator alters some of the variables in the GMS and release 1000 mutants. Each mutant is evaluated according to its pitch span, pitch range, density of attacks, fitness to its corresponding chord progression, etc. 100 best mutants are selected after the evaluation and used as the next generation of genetic algorithm. After 10 generations the best candidate is picked up as the subject.

4.3.4 Chord Progression Generator

As stated in Section 2.1.3, the chord progression generator is a Finite State Machine (FSM) where each state is a tonal center (referring back to Fig. 2-2). Each state emits subject's chord progression in that tonal center while each transition emits the chord progression for an episode which modulates to the next tonal center. Both the tonal center progression and the modulation chord progression are random progression of hard-coded possible values.

Modulation of key center			**Possible chord progressions**											
From		**To**												
C	➔	**G**	C	Am	D	D7	C	D7	Em	D7				
Am	➔	**Em**	Am	D7	B	B7	Am	D	C	B7				
C	➔	**Dm**	C	G	A	A7	C	F	A	A7	C	F	C7	A7
G	➔	**Am**	G	D	E	E7	G	C	E	E7	G	C	G7	E7
Am	➔	**Dm**	Am	E	A	A7	Am	C	C7	A7				
Dm	➔	**G**	F	A	D	D7	Dm	Am	D	D7				
G	➔	**C**	G	A7	D	G7	G	G7	Dm	G7				
Am	➔	**C**	Am	D	G	G7	Am	Dm	G	G7				
Em	➔	**G**	Em	A	D	D7	Em	Am	D	D7				
C	➔	**Am**	C	G	D	E7	C	D	E	E7				
Em	➔	**C**	Em	A	D	G7	C7	A7	Dm	G7				

Table 4-2 Modulation chord progression (emission of transitions between key centers)

Below is the chord progression generated, tagged with subject or free counterpoint, key center, voice index (0 for Bass, 1 for Alto, 2 for Soprano). Scale degree 0 = do, 1 = re, 2 = mi, …, 6 = ti. The #\# and #\- denote sharp and flat of the scale degree it is following.

```
((("subject" "C" 1) ((0 2 4) ()) ((3 5 0) ()) ((4 6 1) ()) ((0 2 4) ()))
 (("subject" "G" 2) ((4 6 1) ())
                    ((0 2 4) ())
                    ((1 3 5) ((3 #\#)))
                    ((4 6 1) ()))
 (("subject" "C" 0) ((0 2 4) ()) ((3 5 0) ()) ((4 6 1) ()) ((0 2 4) ()))
 (("free" "G") ((0 2 4) ((3 #\#)))
               ((1 3 5 0) ((3 #\#)))
               ((2 4 6) ((3 #\#)))
               ((1 3 5 0) ((3 #\#))))
 (("subject" "G" 1) ((4 6 1) ((3 #\#)))
                    ((0 2 4) ((3 #\#)))
                    ((1 3 5) ((3 #\#)))
                    ((4 6 1) ((3 #\#))))
 (("free" "C") ((4 6 1) ())
               ((5 0 2 4) ((0 #\#)))
               ((1 3 5) ((3 #\#)))
               ((4 6 1 3) ()))
 (("subject" "C" 2) ((0 2 4) ()) ((3 5 0) ()) ((4 6 1) ()) ((0 2 4) ()))
 (("free" "Am") ((0 2 4) ())
                ((4 6 1) ())
                ((1 3 5) ((3 #\#)))
                ((2 4 6 1) ((3 #\#) (4 #\#))))
 (("subject" "Am" 0) ((5 0 2) ())
                     ((1 3 5) ())
                     ((2 4 6) ((4 #\#)))
                     ((5 0 2) ()))
 (("subject" "Em" 1) ((2 4 6) ())
                     ((5 0 2) ())
                     ((6 1 3) ((1 #\#) (3 #\#)))
                     ((2 4 6) ()))
 (("subject" "Am" 2) ((5 0 2) ())
                     ((1 3 5) ())
                     ((2 4 6) ((4 #\#)))
                     ((5 0 2) ()))
 (("free" "Em") ((5 0 2) ((3 #\#)))
                ((1 3 5 0) ((3 #\#)))
                ((6 1 3) ((3 #\#) (1 #\#)))
                ((6 1 3 5) ((3 #\#) (0 #\#) (1 #\#))))
 (("subject" "Em" 0) ((2 4 6) ((3 #\#)))
                     ((5 0 2) ((3 #\#)))
                     ((6 1 3) ((3 #\#) (1 #\#)))
                     ((2 4 6) ((3 #\#))))
 (("free" "G") ((2 4 6) ((3 #\#)))
               ((5 0 2) ((3 #\#)))
               ((1 3 5) ((3 #\#)))
               ((1 3 5 0) ((3 #\#))))
 (("subject" "G" 1) ((4 6 1) ((3 #\#)))
                    ((0 2 4) ((3 #\#)))
                    ((1 3 5) ((3 #\#)))
                    ((4 6 1) ((3 #\#))))
 (("free" "C") ((4 6 1) ()) ((4 6 1 3) ()) ((1 3 5) ()) ((4 6 1 3) ()))
 (("subject" "C" 2) ((0 2 4) ()) ((3 5 0) ()) ((4 6 1) ()) ((0 2 4) ()))
 (("free" "C") ((3 5 0) ()) ((4 6 1) ()) ((0 2 4) ()) ((0 2 4) ())))
```

This format is then compiled into degree score and key sig progression, where in each bundle, each scale degree (from 0 to 6, or do to ti) is associated with a score that will be used for melody generation.

Example of degree score and key sig progression:

```
((("subject" "C" 1) (((0 1) (1 0) (2 0) (3 4) (4 0) (5 3) (6 0)) ())
                 (((0 -3) (1 0) (2 -3) (3 4) (4 -1) (5 3) (6 0)) ())
                 (((0 -1) (1 1) (2 0) (3 -4) (4 4) (5 -3) (6 3)) ())
                 (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ()))
 (("subject" "G" 2)
  (((0 -4) (1 1) (2 -3) (3 0) (4 3) (5 0) (6 3)) ())
  (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ())
  (((0 -4) (1 4) (2 -3) (3 3) (4 -1) (5 1) (6 0)) ((3 #\#)))
  (((0 0) (1 -3) (2 0) (3 0) (4 4) (5 -1) (6 3)) ()))
 (("subject" "C" 0) (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ())
                 (((0 -3) (1 0) (2 -3) (3 4) (4 -1) (5 3) (6 0)) ())
                 (((0 -1) (1 1) (2 0) (3 -4) (4 4) (5 -3) (6 3)) ())
                 (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ()))
 (("free" "G")
  (((0 4) (1 0) (2 3) (3 0) (4 1) (5 0) (6 0)) ((3 #\#)))
  (((0 -2) (1 4) (2 -3) (3 3) (4 -1) (5 1) (6 0)) ((3 #\#)))
  (((0 -2) (1 -4) (2 4) (3 -3) (4 3) (5 -1) (6 1)) ((3 #\#)))
  (((0 2) (1 4) (2 -4) (3 3) (4 -3) (5 1) (6 -1)) ((3 #\#))))
 (("subject" "G" 1)
  (((0 -2) (1 -3) (2 0) (3 -3) (4 4) (5 -1) (6 3)) ((3 #\#)))
  (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ((3 #\#)))
  (((0 -4) (1 4) (2 -3) (3 3) (4 -1) (5 1) (6 0)) ((3 #\#)))
  (((0 0) (1 -3) (2 0) (3 -3) (4 4) (5 -1) (6 3)) ((3 #\#))))
 (("free" "C")
  (((0 0) (1 1) (2 0) (3 0) (4 4) (5 0) (6 3)) ())
  (((0 3) (1 -1) (2 1) (3 0) (4 -2) (5 4) (6 -3)) ((0 #\#)))
  (((0 0) (1 4) (2 -1) (3 3) (4 -2) (5 -3) (6 0)) ((3 #\#)))
  (((0 0) (1 -3) (2 0) (3 2) (4 4) (5 -1) (6 3)) ()))
 (("subject" "C" 2) (((0 4) (1 -1) (2 3) (3 -2) (4 -3) (5 0) (6 -3)) ())
                 (((0 -3) (1 0) (2 -3) (3 4) (4 -1) (5 3) (6 0)) ())
                 (((0 -1) (1 1) (2 0) (3 -4) (4 4) (5 -3) (6 3)) ())
                 (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ()))
 (("free" "Am")
  (((0 4) (1 0) (2 3) (3 0) (4 1) (5 0) (6 0)) ())
  (((0 -4) (1 1) (2 -3) (3 0) (4 3) (5 0) (6 3)) ())
  (((0 0) (1 3) (2 0) (3 3) (4 -4) (5 1) (6 -3)) ((3 #\#)))
  (((0 0) (1 -2) (2 4) (3 -3) (4 3) (5 -1) (6 1)) ((3 #\#) (4 #\#))))
 (("subject" "Am" 0)
  (((0 3) (1 -2) (2 -3) (3 0) (4 0) (5 4) (6 -1)) ())
  (((0 -3) (1 4) (2 -1) (3 3) (4 0) (5 -3) (6 0)) ())
  (((0 0) (1 -4) (2 4) (3 -3) (4 3) (5 -1) (6 1)) ((4 #\#)))
  (((0 3) (1 0) (2 -3) (3 0) (4 0) (5 4) (6 -1)) ()))
 (("subject" "Em" 1)
  (((0 -3) (1 0) (2 3) (3 0) (4 3) (5 -4) (6 1)) ())
  (((0 3) (1 0) (2 -3) (3 0) (4 -3) (5 4) (6 -1)) ())
  (((0 -3) (1 3) (2 -1) (3 1) (4 0) (5 -4) (6 4)) ((1 #\#) (3 #\#)))
  (((0 0) (1 0) (2 4) (3 0) (4 3) (5 0) (6 -3)) ()))
 (("subject" "Am" 2)
  (((0 3) (1 0) (2 -3) (3 0) (4 -3) (5 4) (6 -1)) ())
  (((0 -3) (1 4) (2 -1) (3 3) (4 0) (5 -3) (6 0)) ())
  (((0 0) (1 -4) (2 4) (3 -3) (4 3) (5 -1) (6 1)) ((4 #\#)))
  (((0 3) (1 0) (2 -3) (3 0) (4 0) (5 4) (6 -1)) ()))
 (("free" "Em")
  (((0 3) (1 0) (2 1) (3 0) (4 0) (5 4) (6 0)) ((3 #\#)))
  (((0 -1) (1 4) (2 -1) (3 3) (4 0) (5 -3) (6 0)) ((3 #\#)))
  (((0 -2) (1 3) (2 0) (3 -2) (4 0) (5 -1) (6 4)) ((3 #\#) (1 #\#)))
  (((0 0) (1 3) (2 0) (3 1) (4 0) (5 2) (6 4))
   ((3 #\#) (0 #\#) (1 #\#))))
 (("subject" "Em" 0)
  (((0 0) (1 0) (2 4) (3 -1) (4 3) (5 -2) (6 -3)) ((3 #\#)))
  (((0 3) (1 0) (2 -3) (3 0) (4 -3) (5 4) (6 -1)) ((3 #\#)))
  (((0 -3) (1 3) (2 -1) (3 1) (4 0) (5 -4) (6 4)) ((3 #\#) (1 #\#)))
  (((0 0) (1 0) (2 4) (3 -1) (4 3) (5 0) (6 -3)) ((3 #\#))))
 (("free" "G")
  (((0 0) (1 0) (2 4) (3 0) (4 3) (5 0) (6 1)) ((3 #\#)))
  (((0 3) (1 0) (2 -3) (3 0) (4 -3) (5 4) (6 -1)) ((3 #\#)))
  (((0 -3) (1 4) (2 -1) (3 3) (4 0) (5 -3) (6 0)) ((3 #\#)))
  (((0 2) (1 4) (2 0) (3 3) (4 0) (5 1) (6 0)) ((3 #\#))))
 (("subject" "G" 1)
  (((0 -2) (1 -3) (2 0) (3 -3) (4 4) (5 -1) (6 3)) ((3 #\#)))
  (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ((3 #\#)))
  (((0 -4) (1 4) (2 -3) (3 3) (4 -1) (5 1) (6 0)) ((3 #\#)))
  (((0 0) (1 -3) (2 0) (3 -3) (4 4) (5 -1) (6 3)) ((3 #\#))))
 (("free" "C") (((0 0) (1 1) (2 0) (3 0) (4 4) (5 0) (6 3)) ())
               (((0 0) (1 1) (2 0) (3 2) (4 4) (5 0) (6 3)) ())
               (((0 0) (1 3) (2 0) (3 1) (4 -4) (5 1) (6 -3)) ())
               (((0 0) (1 -3) (2 0) (3 -1) (4 4) (5 -1) (6 3)) ()))
 (("subject" "C" 2) (((0 4) (1 -1) (2 3) (3 -2) (4 -3) (5 0) (6 -3)) ())
                    (((0 -3) (1 0) (2 -3) (3 4) (4 -1) (5 3) (6 0)) ())
                    (((0 -1) (1 1) (2 0) (3 -4) (4 4) (5 -3) (6 3)) ())
                    (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ()))
 (("free" "C") (((0 -3) (1 0) (2 -3) (3 4) (4 -1) (5 3) (6 0)) ())
               (((0 -1) (1 1) (2 0) (3 -4) (4 4) (5 -3) (6 3)) ())
               (((0 4) (1 -1) (2 3) (3 0) (4 -3) (5 0) (6 -3)) ())
               (((0 4) (1 0) (2 3) (3 0) (4 1) (5 0) (6 0)) ())))
```

4.3.5 Melody Generator

Melody generator is almost identical to subject generator except it is restricted by more regulations including intra-voice and inter-voice factors. Intra-voice (per-segment) evaluators favor fitness to the chord progression, moderate density of notes, moderate pitch span, and smooth linkage with previous segment in the timeline. Inter-voice evaluators favor the lack dissonant clash of intervals of seconds, sevenths and ninth, and the lack of clash between chord-tone attacks and non-chord-tone attacks. See appendix for GMS.

5 Evaluation

We have spent enormous time in fine-tuning the parameters of evaluators. Before a fix is implemented which reduces the time of fugue generation by half, it is estimated that a minimum of two hours is required to generate a fugue provided genetic algorithm is fully utilized. Belated feedback is the main issue that hampers the process of fine-tuning.

In addition, the subjectivity of music evaluation is also a problem. While not everyone appreciates the beauty of fugues, we are fortunate to have many friends who have prior music study experience to give us feedback.

The evaluators used in Genetic Algorithm have been changed several times after the first version. The oldest evaluator focuses on fitness to chord as well as note density. Until now, several other evaluators have been added, including pitch span, linkage, range etc.. Inter-voice evaluators are also used prevent cross-voice dissonant interval as well as other counterpoint taboos.

One of the most difficult challenges is to guide the genetic algorithm to optimal solution. A cock-tail master evaluator is prone to please the evaluator, say Evaluator A, that is willing to give high score at the very beginning, but in some special cases such practice is hazardous because mutants need to bypass an area where Evaluator A is not that into before reaching a place where both Evaluator A and Evaluator B are happy about the result. We manage to solve this problem by suppressing the grading from Evaluator A when the grading of form Evaluator B is too low. Such incremental evaluators contribute to positive results especially when we prioritize chord evaluators.

However, if the criteria of suppression are too strict, there are cases that the master evaluator is stuck in suppressed state for generations and never find a way out. This situation can be relieved by lowering the criteria or restart the genetic algorithm process.

While we are still improving our evaluators, the performance of the system is already satisfying. In the appendices we attach a score of sample composition as well as its corresponding background information, including chord progression and Graph Meta Structure bundles.

6 Conclusion and Future Work

In general we have successfully demonstrated a system that can generate polyphonic music similar to fugues and inventions, while its similarity to Bach's work is left to reader's discretion. In terms of level of easy-listening, most of the audience give positive response. In addition, the authors of the report consider some excerpts from the fugues are not easy to compose even for themselves.

However, there are several aspects where the system can be improved. While in most cases the melodies suit the chord pretty well, sometimes the fitness is at the expense of unacceptable of pitch range violation, voice crossing and overlap. When we prioritize prohibition of these violations, the system tends to keep silent in order to circumvent penalties. It will be a great improvement if high score of these two kinds of evaluators can be achieved simultaneously.

7 References

[1] David Huron, The Humdrum Toolkit: Software for Music Research [online], http://www.musiccog.ohio-state.edu/Humdrum/, [Accessed: 2011.1.28]

[2] David Huron, Humdrum Toolkit FAQ

[online],http://www.musiccog.ohio-state.edu/Humdrum/FAQ.html#Who_Benefit, [Accessed: 2011.1.28]

[3] Paul Mark Walker, Theories of Fugue from the Age of Josquin to the Age of Bach (Rochester: University of Rochester Press, 2000), 2.

[4] CCARH@Stanford, CSML@OSU, ThemeFinder[online], www.themefinder.org [Accessed: 2010.9.21]

[5] Horner, A., and Goldberg, 1991. Genetic Algorithms and Computer-Assisted Music Composition, in proceedings of the 1991 International Conference on Genetic Algorithms and Their Applications, pp. 337-441.

[6] Eric Milkie, Joel Chestnutt, Fugue Generation with Genetic Algorithms [online], http://www.cs.cornell.edu/boom/2001sp/Milkie/index.html, [Accessed: 2010.9.19]

[7] Tom Pankhurst, Tom Pankhurst's Guide to Schenkerian Analysis [online], http://www.schenkerguide.com/, [Accessed: 2010.9.19]

[8] Kent Kennan, Counterpoint, New Jersey: Prentice-Hall, 1987.

[9] CCARH@Stanford, KernScore[online], http://kern.humdrum.org/ , [Accessed: 2010.9.21]

8 Appendices

8.1 GMS for the composition

```
((('((r) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r)
    (r) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r))
   ())
  ((gf=
    (gdx
     (g+
      (gml
       (g+
        (gml
         (gdx
          (g*
           (gml
            (gdx
             (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                  (gdx '((r) (r) (r)
(r)))))
            1)
           2))
         -1)
        (gdx
         (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
             (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
       1)
      (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
    var1)
   ((var1 fixed-subj-first-note
"c")))
  ('((r) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r)
    (r) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r))
   ()))
 (('((r) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r)
    (r) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r))
   ())
  ((gf=
    (gdx
     (g+
      (var16
       (g+
        (var10
         (g+ (var8 (gdx (g+ (var3 (gdx
var1) var4) (gdx var2))) var9)
             (gdx (gdx (g* (var6 (gdx
var5) var7) 2))))
         var11)
        (gdx (g+ (var3 (gdx var1) var4)
(gdx var2))))
       var17)
      (gdx (g+ (var14 (gdx var12)
var15) (gdx var13)))))
    var18)
   ((var1 atm4 '((k (1 -1)) (r) (k (1
-1)) (k)))
    (var2 atm4 '((r) (r) (k) (r)))
    (var3 glu-type gml)
    (var4 glu-ofst 2)
    (var5 atm4 '((k (1 2)) (r) (k) (r)))
    (var6 glu-type gml)
    (var7 rep-ofst 0)
    (var8 glu-type gml)
    (var9 glu-ofst -1)
    (var10 glu-type gml)
    (var11 glu-ofst 1)
    (var12 atm4 '((k (1 1)) (k (1 -1))
(k (1 -1)) (k)))
    (var13 atm4 '((k) (r) (r) (r)))
    (var14 glu-type gmf)
    (var15 glu-ofst 2)
    (var16 glu-type gmf)
    (var17 glu-ofst -2)
    (var18 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "b")))
  ((gf=
    (gdx
     (g+
      (gml
       (g+
        (gml
         (gdx
          (g*
           (gml
            (gdx
             (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                  (gdx '((r) (r) (r)
(r)))))
            1)
           2))
         -1)
        (gdx
         (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
             (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
       1)
      (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
    var1)
   ((var1 ignored "g"))))
 (((gf=
    (gdx
    (g+
     (gml
      (g+
       (gml
        (gdx
         (g*
          (gml
           (gdx
            (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                (gdx '((r) (r) (r)
(r)))))
          1)
         2))
       -1)
       (gdx
        (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
           (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
     1)
     (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
   var1)
  ((var1 ignored "C")))
 ((gf=
   (gdx
    (g+
     (var10 (gdx (gdx (g* (var2 (gdx
var1) var3) 2))) var11)
     (gdx (g* (var8 (gdx (g+ (var6
(gdx var4) var7) (gdx var5))) var9)
3))))
   var12)
  ((var1 atm4 '((r) (r) (k) (r)))
   (var2 glu-type gml)
   (var3 rep-ofst 0)
   (var4 atm4 '((k (1 1)) (r) (r) (k
(1 1))))
   (var5 atm4 '((k) (r) (r) (r)))
   (var6 glu-type gml)
   (var7 glu-ofst 1)
   (var8 glu-type gml)
   (var9 rep-ofst -1)
   (var10 glu-type gml)
   (var11 glu-ofst 3)
   (var12 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "c")))
 ((gf=
   (gdx
    (g+
     (var10
      (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
3))
      var11)
     (gdx (gdx (g* (var8 (gdx var7)
var9) 2)))))
   var12)
  ((var1 atm4 '((k (1 1)) (k (1 1))
(k (1 -2)) (k)))
   (var2 atm4 '((k) (r) (r) (r)))
   (var3 glu-type gml)
   (var4 glu-ofst 2)
   (var5 glu-type gml)
   (var6 rep-ofst 1)
   (var7 atm4 '((k (1 0)) (r) (k) (r)))
   (var8 glu-type gml)
   (var9 rep-ofst 0)
   (var10 glu-type gml)
   (var11 glu-ofst -1)
   (var12 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
         "c"))))
 (((gf=
   (gdx
    (g+
     (var10
      (gdx (g* (var4 (gdx (gdx (g*
(var2 (gdx var1) var3) 2))) var5) 3))
      var11)
     (gdx (g+ (var8 (gdx var6) var9)
(gdx var7)))))
   var12)
  ((var1 atm4 '((k (1 2)) (r) (k) (r)))
   (var2 glu-type gmf)
   (var3 rep-ofst 0)
   (var4 glu-type gml)
   (var5 rep-ofst -1)
   (var6 atm4 '((k (1 3)) (r) (k) (r)))
   (var7 atm4 '((k (1 1)) (k (1 -1))
(k (1 -1)) (k)))
   (var8 glu-type gml)
   (var9 glu-ofst -1)
   (var10 glu-type gml)
   (var11 glu-ofst 3)
   (var12
    ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
    "C")))
  ((gf=
   (gdx
    (g+
     (var16
      (g+
       (var11
        (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
2))
        var12)
       (gdx (g+ (var9 (gdx var7)
var10) (gdx var8))))
      var17)
     (gdx (gdx (g* (var14 (gdx var13)
var15) 2)))))
   var18)
  ((var1 atm4 '((k (1 -1)) (k (1 -1))
(k (1 -1)) (k)))
   (var2 atm4 '((k (1 -1)) (k (1 -1))
(k) (r)))
   (var3 glu-type gml)
   (var4 glu-ofst 1)
```

```
    (var5 glu-type gmf)
    (var6 rep-ofst 0)
    (var7 atm4 '((r) (k (1 1)) (k (1
1)) (k (1 1))))
    (var8 atm4 '((k (1 -4)) (r) (k)
(r)))
    (var9 glu-type gml)
    (var10 glu-ofst -1)
    (var11 glu-type gmf)
    (var12 glu-ofst -1)
    (var13 atm4 '((k (1 -1)) (k (1 1))
(k (1 1)) (k)))
    (var14 glu-type gml)
    (var15 rep-ofst 1)
    (var16 glu-type gmf)
    (var17 glu-ofst -2)
    (var18 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "A")))
  ((gf=
    (gdx
     (g+
      (var9 (gdx (g* (var4 (gdx (gdx
(g* (var2 (gdx var1) var3) 2))) var5)
3))
           var10)
      (gdx (gdx (g* (var7 (gdx var6)
var8) 2)))))
    var11)
   ((var1 atm4 '((k (1 -7)) (r) (k)
(r)))
    (var2 glu-type gmf)
    (var3 rep-ofst 0)
    (var4 glu-type gmf)
    (var5 rep-ofst 1)
    (var6 atm4 '((k (1 1)) (k (1 1))
(k (1 1)) (k)))
    (var7 glu-type gml)
    (var8 rep-ofst -1)
    (var9 glu-type gml)
    (var10 glu-ofst 1)
    (var11 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
          "cc"))))
 (((gf=
    (gdx
     (g+
      (var16
       (g+
        (var10
         (g+ (var8 (gdx (gdx (g* (var2
(gdx var1) var3) 2))) var9)
             (gdx (g+ (var6 (gdx var4)
var7) (gdx var5))))
        var11)
        (gdx (gdx (g* (var2 (gdx var1)
var3) 2))))
       var17)
      (gdx (g+ (var14 (gdx var12)
var15) (gdx var13)))))
    var18)
   ((var1 atm4 '((k (1 1)) (r) (k (1
1)) (k (1 1))))
    (var2 glu-type gml)
    (var3 rep-ofst 0)
    (var4 atm4 '((k (1 1)) (k (1 -1))
(k (1 -1)) (k)))
    (var5 atm4 '((k (1 -1)) (r) (k (1
-1)) (k)))
    (var6 glu-type gml)
    (var7 glu-ofst 1)
    (var8 glu-type gml)
    (var9 glu-ofst 1)
    (var10 glu-type gml)
    (var11 glu-ofst 1)
    (var12 atm4 '((k) (r) (r) (r)))
    (var13 atm4 '((r) (r) (r) (k)))
    (var14 glu-type gml)
    (var15 glu-ofst 1)
    (var16 glu-type gml)
    (var17 glu-ofst -1)
    (var18
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "G")))
  ((gf=
    (gdx
     (g+
      (gml
       (g+
        (gml
         (gdx
          (g*
           (gml
            (gdx
             (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                (gdx '((r) (r) (r)
(r)))))
           1)
          2))
        -1)
       (gdx
        (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
            (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
      1)
     (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
    var1)
   ((var1 ignored "G")))
  ((gf=
    (gdx
     (g+
      (var10
       (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
3))
       var11)
      (gdx (gdx (g* (var8 (gdx var7)
var9) 2)))))
    var12)
   ((var1 atm4 '((k (1 2)) (r) (k) (r)))
    (var2 atm4 '((k (1 -2)) (k (1 1))
(k (1 2)) (k)))
    (var3 glu-type gml)
    (var4 glu-ofst -2)
    (var5 glu-type gmf)
    (var6 rep-ofst 1)
    (var7 atm4 '((k (1 -2)) (r) (k)
(r)))
    (var8 glu-type gml)
    (var9 rep-ofst 0)
    (var10 glu-type gml)
    (var11 glu-ofst 1)
    (var12 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
         "b"))))
 (((gf=
    (gdx
     (g+
      (var15
       (g+
        (var9
         (gdx (g* (var4 (gdx (gdx (g*
(var2 (gdx var1) var3) 2))) var5) 2))
         var10)
        (gdx (gdx (g* (var7 (gdx var6)
var8) 2))))
       var16)
      (gdx (g+ (var13 (gdx var11)
var14) (gdx var12)))))
    var17)
   ((var1 atm4 '((k (1 -1)) (r) (r)
(k)))
    (var2 glu-type gml)
    (var3 rep-ofst -1)
    (var4 glu-type gml)
    (var5 rep-ofst -1)
    (var6 atm4 '((r) (k (1 -1)) (k (1
-1)) (k)))
    (var7 glu-type gmf)
    (var8 rep-ofst -1)
    (var9 glu-type gmf)
    (var10 glu-ofst 2)
    (var11 atm4 '((k (1 1)) (r) (k)
(r)))
    (var12 atm4 '((k) (r) (r) (r)))
    (var13 glu-type gml)
    (var14 glu-ofst 0)
    (var15 glu-type gmf)
    (var16 glu-ofst -1)
    (var17
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "G")))
  ((gf=
    (gdx
     (g+
      (var16
       (g+ (var8 (gdx (g+ (var3 (gdx
var1) var4) (gdx var2))) var9)
         (gdx (gdx (g* (var6 (gdx
```

```
var5) var7) 2))))
      var17)
     (gdx
      (g* (var14 (gdx (g+ (var12 (gdx
var10) var13) (gdx var11))) var15)
        2))))
    var18)
   ((var1 atm4 '((k) (r) (r) (r)))
    (var2 atm4 '((r) (r) (r) (r)))
    (var3 glu-type gml)
    (var4 glu-ofst 1)
    (var5 atm4 '((k (1 1)) (k (1 1))
(k) (r)))
    (var6 glu-type gml)
    (var7 rep-ofst 0)
    (var8 glu-type gml)
    (var9 glu-ofst -1)
    (var10 atm4 '((k (1 1)) (r) (k (1
1)) (k (1 1))))
    (var11 atm4 '((r) (r) (r) (r)))
    (var12 glu-type gml)
    (var13 glu-ofst 2)
    (var14 glu-type gml)
    (var15 rep-ofst -1)
    (var16 glu-type gml)
    (var17 glu-ofst -1)
    (var18 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "b")))
  ((gf=
    (gdx
     (g+
      (var15
       (g+
        (var10
         (gdx (g* (var4 (gdx (gdx (g*
(var2 (gdx var1) var3) 2))) var5) 2))
         var11)
        (gdx (g+ (var8 (gdx var6) var9)
(gdx var7))))
       var16)
      (gdx (gdx (g* (var13 (gdx var12)
var14) 2)))))
    var17)
   ((var1 atm4 '((k (1 2)) (r) (k) (r)))
    (var2 glu-type gml)
    (var3 rep-ofst 0)
    (var4 glu-type gmf)
    (var5 rep-ofst 1)
    (var6 atm4 '((k (1 1)) (r) (k) (r)))
    (var7 atm4 '((r) (r) (k) (r)))
    (var8 glu-type gml)
    (var9 glu-ofst 1)
    (var10 glu-type gml)
    (var11 glu-ofst 1)
    (var12 atm4 '((k (1 1)) (k (1 1))
(k (1 -2)) (k)))
    (var13 glu-type gml)
    (var14 rep-ofst -1)
    (var15 glu-type gmf)
    (var16 glu-ofst 1)
    (var17 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
         "e"))))
 (((gf=
   (gdx
    (g+
     (var12
      (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
2))
      var13)
     (gdx (g* (var10 (gdx (gdx (g*
(var8 (gdx var7) var9) 2))) var11)
2))))
   var14)
  ((var1 atm4 '((k (1 5)) (r) (k) (r)))
   (var2 atm4 '((k (1 1)) (r) (k (1
1)) (k (1 1))))
   (var3 glu-type gmf)
   (var4 glu-ofst 0)
   (var5 glu-type gml)
   (var6 rep-ofst -1)
   (var7 atm4 '((k (1 0)) (r) (k (1
-1)) (k)))
   (var8 glu-type gml)
   (var9 rep-ofst 1)
   (var10 glu-type gml)
   (var11 rep-ofst -1)
   (var12 glu-type gml)
   (var13 glu-ofst -1)
   (var14
    ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
    "E")))
  ((gf=
   (gdx
    (g+
     (var17
      (g+
       (var11
        (g+ (var9 (gdx (g+ (var3 (gdx
var1) var4) (gdx var2))) var10)
            (gdx (g+ (var7 (gdx var5)
var8) (gdx var6))))
        var12)
       (gdx (g+ (var3 (gdx var1) var4)
(gdx var2))))
      var18)
     (gdx (g+ (var15 (gdx var13)
var16) (gdx var14)))))
   var19)
  ((var1 atm4 '((k (1 1)) (k (1 -1))
(k (1 -1)) (k)))
   (var2 atm4 '((k (1 0)) (r) (k) (r)))
   (var3 glu-type gml)
   (var4 glu-ofst 1)
   (var5 atm4 '((k (1 -1)) (r) (k)
(r)))
   (var6 atm4 '((k) (r) (r) (r)))
   (var7 glu-type gml)
   (var8 glu-ofst 1)
   (var9 glu-type gml)
```

```
   (var10 glu-ofst 1)
   (var11 glu-type gml)
   (var12 glu-ofst 1)
   (var13 atm4 '((k (1 1)) (k (1 1))
(k (1 1)) (k)))
   (var14 atm4 '((r) (r) (k) (r)))
   (var15 glu-type gml)
   (var16 glu-ofst 1)
   (var17 glu-type gml)
   (var18 glu-ofst -1)
   (var19 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "e")))
  ((gf=
   (gdx
    (g+
     (gml
      (g+
       (gml
        (gdx
         (g*
          (gml
           (gdx
            (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                (gdx '((r) (r) (r)
(r)))))
           1)
          2))
        -1)
       (gdx
        (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
            (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
      1)
     (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
   var1)
  ((var1 ignored "cc"))))
 (((gf=
   (gdx
    (g+
     (var17
      (g+
       (var11
        (g+ (var9 (gdx (g+ (var3 (gdx
var1) var4) (gdx var2))) var10)
            (gdx (g+ (var7 (gdx var5)
var8) (gdx var6))))
        var12)
       (gdx (g+ (var3 (gdx var1) var4)
(gdx var2))))
      var18)
     (gdx (g+ (var15 (gdx var13)
var16) (gdx var14)))))
   var19)
  ((var1 atm4 '((k (1 0)) (r) (k (1
1)) (k)))
   (var2 atm4 '((k (1 -1)) (r) (k (1
-1)) (k)))
   (var3 glu-type gml)
```

```
    (var4 glu-ofst -1)
    (var5 atm4 '((r) (r) (k) (r)))
    (var6 atm4 '((k (1 2)) (r) (k) (r)))
    (var7 glu-type gmf)
    (var8 glu-ofst -1)
    (var9 glu-type gml)
    (var10 glu-ofst -2)
    (var11 glu-type gml)
    (var12 glu-ofst 1)
    (var13 atm4 '((k) (r) (r) (r)))
    (var14 atm4 '((r) (r) (r) (r)))
    (var15 glu-type gml)
    (var16 glu-ofst -1)
    (var17 glu-type gml)
    (var18 glu-ofst 1)
    (var19
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "c")))
  ((gf=
    (gdx
     (g+
      (var10
       (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
3))
       var11)
      (gdx (gdx (g* (var8 (gdx var7)
var9) 2)))))
   var12)
   ((var1 atm4 '((k) (r) (r) (r)))
    (var2 atm4 '((k) (r) (r) (r)))
    (var3 glu-type gml)
    (var4 glu-ofst -2)
    (var5 glu-type gml)
    (var6 rep-ofst -1)
    (var7 atm4 '((k (1 -1)) (r) (r)
(k)))
    (var8 glu-type gml)
    (var9 rep-ofst 1)
    (var10 glu-type gml)
    (var11 glu-ofst 1)
    (var12 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "E")))
  ((gf=
    (gdx
     (g+
      (var12
       (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
2))
       var13)
      (gdx (g* (var10 (gdx (gdx (g*
(var8 (gdx var7) var9) 2))) var11)
2))))
   var14)
   ((var1 atm4 '((k (1 -3)) (r) (k)
(r)))
    (var2 atm4 '((k (1 1)) (r) (k) (r)))
    (var3 glu-type gmf)
    (var4 glu-ofst 2)
```

```
    (var5 glu-type gml)
    (var6 rep-ofst 1)
    (var7 atm4 '((k) (r) (r) (r)))
    (var8 glu-type gmf)
    (var9 rep-ofst 0)
    (var10 glu-type gml)
    (var11 rep-ofst -1)
    (var12 glu-type gml)
    (var13 glu-ofst -2)
    (var14 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
         "a"))))
 (((gf=
    (gdx
     (g+
      (gml
       (g+
        (gml
         (gdx
          (g*
           (gml
            (gdx
             (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
              (gdx '((r) (r) (r)
(r)))))
          1)
         2))
       -1)
      (gdx
       (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
          (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
      1)
     (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
   var1)
   ((var1 ignored "AA")))
  ((gf=
    (gdx
     (g+
      (var14
       (g+
        (var9
         (gdx (g* (var4 (gdx (gdx (g*
(var2 (gdx var1) var3) 2))) var5) 2))
         var10)
        (gdx (gdx (g* (var7 (gdx var6)
var8) 2))))
       var15)
      (gdx (gdx (g* (var12 (gdx var11)
var13) 2)))))
   var16)
   ((var1 atm4 '((r) (r) (k) (r)))
    (var2 glu-type gmf)
    (var3 rep-ofst 0)
    (var4 glu-type gml)
    (var5 rep-ofst 1)
    (var6 atm4 '((k (1 3)) (r) (k) (r)))
    (var7 glu-type gmf)
    (var8 rep-ofst 0)
    (var9 glu-type gml)
    (var10 glu-ofst 1)
    (var11 atm4 '((k (1 2)) (k (1 -1))
(k (1 -1)) (k)))
    (var12 glu-type gml)
    (var13 rep-ofst 0)
    (var14 glu-type gmf)
    (var15 glu-ofst -1)
    (var16 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "C")))
  ((gf=
    (gdx
     (g+
      (var15
       (g+
        (var10
         (g+ (var8 (gdx (gdx (g* (var2
(gdx var1) var3) 2))) var9)
             (gdx (g+ (var6 (gdx var4)
var7) (gdx var5))))
         var11)
        (gdx (gdx (g* (var2 (gdx var1)
var3) 2))))
       var16)
      (gdx (gdx (g* (var13 (gdx var12)
var14) 2)))))
    var17)
   ((var1 atm4 '((k (1 1)) (r) (r) (k
(1 1))))
    (var2 glu-type gmf)
    (var3 rep-ofst 0)
    (var4 atm4 '((k (1 2)) (r) (k) (r)))
    (var5 atm4 '((r) (r) (k (1 -1))
(k)))
    (var6 glu-type gml)
    (var7 glu-ofst 1)
    (var8 glu-type gml)
    (var9 glu-ofst 2)
    (var10 glu-type gml)
    (var11 glu-ofst 1)
    (var12 atm4 '((k (1 -2)) (r) (k)
(r)))
    (var13 glu-type gml)
    (var14 rep-ofst -2)
    (var15 glu-type gml)
    (var16 glu-ofst 1)
    (var17 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
          "aa"))))
 (((gf=
    (gdx
     (g+
      (var11
       (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
3))
       var12)
      (gdx (g+ (var9 (gdx var7) var10)
(gdx var8)))))
    var13)
```

```
   ((var1 atm4 '((k (1 2)) (r) (k) (r)))
    (var2 atm4 '((k (1 1)) (k (1 1))
(k (1 -1)) (k)))
    (var3 glu-type gml)
    (var4 glu-ofst -1)
    (var5 glu-type gml)
    (var6 rep-ofst 1)
    (var7 atm4 '((k (1 -1)) (r) (k)
(r)))
    (var8 atm4 '((k (1 -1)) (r) (k (1
-1)) (k)))
    (var9 glu-type gml)
    (var10 glu-ofst 1)
    (var11 glu-type gml)
    (var12 glu-ofst 1)
    (var13
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "E")))
  ((gf=
    (gdx
     (g+
      (gml
       (g+
        (gml
         (gdx
          (g*
           (gml
            (gdx
             (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                 (gdx '((r) (r) (r)
(r)))))
           1)
          2))
        -1)
        (gdx
         (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
             (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
       1)
      (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
    var1)
   ((var1 ignored "e")))
  ((gf=
    (gdx
     (g+
      (var16
       (g+
        (var11
         (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
2))
        var12)
        (gdx (g+ (var9 (gdx var7)
var10) (gdx var8))))
       var17)
      (gdx (gdx (g* (var14 (gdx var13)
var15) 2)))))
```

```
   var18)
  ((var1 atm4 '((k (1 -1)) (r) (k (1
-1)) (k)))
   (var2 atm4 '((k (1 3)) (r) (k) (r)))
   (var3 glu-type gml)
   (var4 glu-ofst 1)
   (var5 glu-type gml)
   (var6 rep-ofst 0)
   (var7 atm4 '((k (1 0)) (r) (k) (r)))
   (var8 atm4 '((k (1 1)) (k (1 1))
(k) (r)))
   (var9 glu-type gml)
   (var10 glu-ofst -2)
   (var11 glu-type gml)
   (var12 glu-ofst 0)
   (var13 atm4 '((k (1 -1)) (r) (r)
(k)))
   (var14 glu-type gml)
   (var15 rep-ofst 1)
   (var16 glu-type gml)
   (var17 glu-ofst 1)
   (var18 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
         "gg"))))
 (((gf=
   (gdx
    (g+
     (var12
      (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
2))
      var13)
     (gdx (g* (var10 (gdx (gdx (g*
(var8 (gdx var7) var9) 2))) var11)
2))))
   var14)
  ((var1 atm4 '((k) (r) (r) (r)))
   (var2 atm4 '((r) (k (1 1)) (k (1
1)) (k (1 1))))
   (var3 glu-type gml)
   (var4 glu-ofst 1)
   (var5 glu-type gml)
   (var6 rep-ofst 0)
   (var7 atm4 '((k (1 -1)) (k (1 -1))
(k (1 2)) (k)))
   (var8 glu-type gml)
   (var9 rep-ofst 0)
   (var10 glu-type gml)
   (var11 rep-ofst 1)
   (var12 glu-type gml)
   (var13 glu-ofst 2)
   (var14
    ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
    "AA")))
  ((gf=
   (gdx
    (g+
     (var15
      (g+
       (var10
```

```
       (gdx (g* (var4 (gdx (gdx (g*
(var2 (gdx var1) var3) 2))) var5) 2))
       var11)
      (gdx (g+ (var8 (gdx var6) var9)
(gdx var7))))
     var16)
    (gdx (gdx (g* (var13 (gdx var12)
var14) 2)))))
   var17)
  ((var1 atm4 '((r) (r) (k) (r)))
   (var2 glu-type gmf)
   (var3 rep-ofst 0)
   (var4 glu-type gml)
   (var5 rep-ofst 1)
   (var6 atm4 '((k (1 3)) (r) (k) (r)))
   (var7 atm4 '((k (1 5)) (r) (k) (r)))
   (var8 glu-type gml)
   (var9 glu-ofst -1)
   (var10 glu-type gml)
   (var11 glu-ofst 1)
   (var12 atm4 '((k) (r) (r) (r)))
   (var13 glu-type gmf)
   (var14 rep-ofst 1)
   (var15 glu-type gml)
   (var16 glu-ofst -1)
   (var17 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "C")))
  ((gf=
   (gdx
    (g+
     (gml
      (g+
       (gml
        (gdx
         (g*
          (gml
           (gdx
            (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                (gdx '((r) (r) (r)
(r)))))
           1)
          2))
        -1)
       (gdx
        (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
            (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
      1)
     (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
   var1)
  ((var1 ignored "a"))))
 (((gf=
   (gdx
    (g+
     (var15
      (g+
       (var10
        (g+ (var8 (gdx (gdx (g* (var2
(gdx var1) var3) 2))) var9)
                  (gdx (g+ (var6 (gdx var4)
var7) (gdx var5))))
            var11)
           (gdx (gdx (g* (var2 (gdx var1)
var3) 2))))
         var16)
        (gdx (gdx (g* (var13 (gdx var12)
var14) 2)))))
     var17)
    ((var1 atm4 '((k (1 -2)) (r) (k)
(r)))
      (var2 glu-type gml)
      (var3 rep-ofst -1)
      (var4 atm4 '((k (1 3)) (r) (k) (r)))
      (var5 atm4 '((k (1 1)) (r) (r) (k
(1 1))))
      (var6 glu-type gml)
      (var7 glu-ofst -1)
      (var8 glu-type gml)
      (var9 glu-ofst -3)
      (var10 glu-type gml)
      (var11 glu-ofst -1)
      (var12 atm4 '((k (1 1)) (k (1 1))
(k (1 1)) (k)))
      (var13 glu-type gml)
      (var14 rep-ofst 1)
      (var15 glu-type gmf)
      (var16 glu-ofst 1)
      (var17
       ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
       "AA")))
   ((gf=
      (gdx
       (g+
        (var11
          (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
3))
         var12)
        (gdx (g+ (var9 (gdx var7) var10)
(gdx var8)))))
     var13)
    ((var1 atm4 '((k (1 2)) (r) (k (1
-1)) (k)))
      (var2 atm4 '((k (1 0)) (r) (r) (k)))
      (var3 glu-type gml)
      (var4 glu-ofst 1)
      (var5 glu-type gml)
      (var6 rep-ofst 1)
      (var7 atm4 '((k (1 1)) (r) (k) (r)))
      (var8 atm4 '((r) (k (1 1)) (k) (r)))
      (var9 glu-type gml)
      (var10 glu-ofst 2)
      (var11 glu-type gmf)
      (var12 glu-ofst 1)
      (var13 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "A")))
   ((gf=
      (gdx
```

```
       (g+
        (var15
          (g+
           (var10
             (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
2))
            var11)
           (gdx (gdx (g* (var8 (gdx var7)
var9) 2))))
          var16)
         (gdx (gdx (g* (var13 (gdx var12)
var14) 2)))))
     var17)
    ((var1 atm4 '((k (1 2)) (r) (k) (r)))
      (var2 atm4 '((k) (r) (r) (r)))
      (var3 glu-type gmf)
      (var4 glu-ofst 2)
      (var5 glu-type gml)
      (var6 rep-ofst 1)
      (var7 atm4 '((k (1 -1)) (k (1 -1))
(k) (r)))
      (var8 glu-type gmf)
      (var9 rep-ofst 1)
      (var10 glu-type gml)
      (var11 glu-ofst -2)
      (var12 atm4 '((k (1 3)) (r) (k)
(r)))
      (var13 glu-type gml)
      (var14 rep-ofst -1)
      (var15 glu-type gml)
      (var16 glu-ofst 1)
      (var17 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
               "a"))))
  (((gf=
      (gdx
       (g+
        (gml
          (g+
           (gml
             (gdx
              (g*
               (gml
                 (gdx
                  (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                       (gdx '((r) (r) (r)
(r)))))
                1)
               2))
            -1)
           (gdx
             (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
                  (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
          1)
         (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
     var1)
```

```
   ((var1 ignored "E")))
  ((gf=
    (gdx
     (g+
      (var9 (gdx (g* (var4 (gdx (gdx
(g* (var2 (gdx var1) var3) 2))) var5)
3))
          var10)
      (gdx (gdx (g* (var7 (gdx var6)
var8) 2)))))
    var11)
   ((var1 atm4 '((k (1 2)) (r) (k) (r)))
    (var2 glu-type gml)
    (var3 rep-ofst 0)
    (var4 glu-type gmf)
    (var5 rep-ofst 1)
    (var6 atm4 '((r) (r) (r) (r)))
    (var7 glu-type gml)
    (var8 rep-ofst -2)
    (var9 glu-type gml)
    (var10 glu-ofst 1)
    (var11 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "E")))
  ((gf=
    (gdx
     (g+
      (var10
       (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
3))
       var11)
      (gdx (gdx (g* (var8 (gdx var7)
var9) 2)))))
    var12)
   ((var1 atm4 '((k (1 -2)) (r) (k)
(r)))
    (var2 atm4 '((k (1 1)) (r) (k (1
1)) (k (1 1))))
    (var3 glu-type gml)
    (var4 glu-ofst 2)
    (var5 glu-type gmf)
    (var6 rep-ofst 1)
    (var7 atm4 '((r) (k (1 -1)) (k (1
2)) (k)))
    (var8 glu-type gml)
    (var9 rep-ofst -1)
    (var10 glu-type gml)
    (var11 glu-ofst 1)
    (var12 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
         "g"))))
 (((gf=
    (gdx
     (g+
      (var12
       (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
2))
       var13)
      (gdx (g* (var10 (gdx (gdx (g*
(var8 (gdx var7) var9) 2))) var11)
2))))
    var14)
   ((var1 atm4 '((k (1 0)) (r) (k) (r)))
    (var2 atm4 '((r) (r) (k (1 -1))
(k)))
    (var3 glu-type gml)
    (var4 glu-ofst 3)
    (var5 glu-type gml)
    (var6 rep-ofst 1)
    (var7 atm4 '((k) (r) (r) (r)))
    (var8 glu-type gml)
    (var9 rep-ofst -1)
    (var10 glu-type gml)
    (var11 rep-ofst 1)
    (var12 glu-type gmf)
    (var13 glu-ofst 2)
    (var14
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "E")))
  ((gf=
    (gdx
     (g+
      (var17
       (g+
        (var11
         (g+ (var9 (gdx (g+ (var3 (gdx
var1) var4) (gdx var2))) var10)
           (gdx (g+ (var7 (gdx var5)
var8) (gdx var6))))
         var12)
        (gdx (g+ (var3 (gdx var1) var4)
(gdx var2))))
       var18)
      (gdx (g+ (var15 (gdx var13)
var16) (gdx var14)))))
    var19)
   ((var1 atm4 '((k (1 -2)) (r) (k)
(r)))
    (var2 atm4 '((k (1 2)) (r) (k) (r)))
    (var3 glu-type gml)
    (var4 glu-ofst 2)
    (var5 atm4 '((k (1 3)) (r) (k (1
-1)) (k)))
    (var6 atm4 '((k (1 1)) (r) (k) (r)))
    (var7 glu-type gml)
    (var8 glu-ofst -1)
    (var9 glu-type gml)
    (var10 glu-ofst -1)
    (var11 glu-type gml)
    (var12 glu-ofst 1)
    (var13 atm4 '((k (1 -1)) (k (1 -1))
(k (1 2)) (k)))
    (var14 atm4 '((r) (r) (r) (r)))
    (var15 glu-type gml)
    (var16 glu-ofst 1)
    (var17 glu-type gml)
    (var18 glu-ofst 1)
    (var19 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "G")))
  ((gf=
    (gdx
     (g+
      (var16
       (g+
        (var11
         (g+ (var9 (gdx (g+ (var3 (gdx
var1) var4) (gdx var2))) var10)
             (gdx (g+ (var7 (gdx var5)
var8) (gdx var6))))
         var12)
        (gdx (g+ (var3 (gdx var1) var4)
(gdx var2))))
       var17)
      (gdx (gdx (g* (var14 (gdx var13)
var15) 2)))))
    var18)
   ((var1 atm4 '((k) (r) (r) (r)))
    (var2 atm4 '((r) (r) (r) (r)))
    (var3 glu-type gml)
    (var4 glu-ofst 1)
    (var5 atm4 '((k) (r) (r) (r)))
    (var6 atm4 '((k (1 -2)) (r) (k)
(r)))
    (var7 glu-type gml)
    (var8 glu-ofst -2)
    (var9 glu-type gml)
    (var10 glu-ofst -1)
    (var11 glu-type gmf)
    (var12 glu-ofst 0)
    (var13 atm4 '((k (1 1)) (r) (k)
(r)))
    (var14 glu-type gmf)
    (var15 rep-ofst -1)
    (var16 glu-type gml)
    (var17 glu-ofst -1)
    (var18 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
           "dd"))))
 (((gf=
    (gdx
     (g+
      (var17
       (g+
        (var11
         (g+ (var9 (gdx (g+ (var3 (gdx
var1) var4) (gdx var2))) var10)
             (gdx (g+ (var7 (gdx var5)
var8) (gdx var6))))
         var12)
        (gdx (g+ (var3 (gdx var1) var4)
(gdx var2))))
       var18)
      (gdx (g+ (var15 (gdx var13)
var16) (gdx var14)))))
    var19)
   ((var1 atm4 '((r) (k (1 -1)) (k (1
-1)) (k)))
    (var2 atm4 '((k) (r) (r) (r)))
    (var3 glu-type gml)
    (var4 glu-ofst -1)
    (var5 atm4 '((k (1 -1)) (k (1 1))
(k) (r)))
    (var6 atm4 '((r) (r) (k) (r)))
    (var7 glu-type gml)
    (var8 glu-ofst -2)
    (var9 glu-type gml)
    (var10 glu-ofst 2)
    (var11 glu-type gml)
    (var12 glu-ofst 1)
    (var13 atm4 '((k (1 7)) (r) (k (1
-1)) (k)))
    (var14 atm4 '((k) (r) (r) (r)))
    (var15 glu-type gml)
    (var16 glu-ofst 1)
    (var17 glu-type gml)
    (var18 glu-ofst 1)
    (var19
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "D")))
  ((gf=
    (gdx
     (g+
      (gml
       (g+
        (gml
         (gdx
          (g*
           (gml
            (gdx
             (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                 (gdx '((r) (r) (r)
(r))))))
            1)
           2))
         -1)
        (gdx
         (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
             (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
       1)
      (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
    var1)
   ((var1 ignored "G")))
  ((gf=
    (gdx
     (g+
      (var16
       (g+
        (var10
         (g+ (var8 (gdx (gdx (g* (var2
(gdx var1) var3) 2))) var9)
             (gdx (g+ (var6 (gdx var4)
var7) (gdx var5))))
         var11)
        (gdx (gdx (g* (var2 (gdx var1)
var3) 2))))
       var17)
      (gdx (g+ (var14 (gdx var12)
var15) (gdx var13)))))
    var18)
   ((var1 atm4 '((k (1 1)) (k (1 -2))
(k (1 1)) (k)))
    (var2 glu-type gml)
    (var3 rep-ofst -1)
    (var4 atm4 '((k (1 5)) (r) (k) (r)))
    (var5 atm4 '((k) (r) (r) (r)))
    (var6 glu-type gml)
    (var7 glu-ofst 2)
    (var8 glu-type gml)
    (var9 glu-ofst -2)
    (var10 glu-type gmf)
    (var11 glu-ofst 1)
    (var12 atm4 '((k) (r) (r) (r)))
    (var13 atm4 '((r) (r) (k) (r)))
    (var14 glu-type gml)
    (var15 glu-ofst -2)
    (var16 glu-type gmf)
    (var17 glu-ofst 1)
    (var18 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
          "aa"))))
 (((gf=
    (gdx
     (gdx
      (g*
       (var10
        (gdx
         (g+
          (var8
           (g+ (var5 (gdx (g* (var2
(gdx var1) var3) 2)) var6) (gdx var4))
          var9)
          (gdx var7)))
        var11)
       2)))
    var12)
   ((var1 atm4 '((k (1 0)) (r) (k) (r)))
    (var2 glu-type gml)
    (var3 rep-ofst 0)
    (var4 atm4 '((k (1 -1)) (k (1 1))
(k (1 1)) (k)))
    (var5 glu-type gml)
    (var6 glu-ofst -1)
    (var7 atm4 '((k (1 -4)) (r) (k)
(r)))
    (var8 glu-type gml)
    (var9 glu-ofst 0)
    (var10 glu-type gml)
    (var11 rep-ofst 1)
    (var12
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "G")))
  ((gf=
    (gdx
     (g+
      (var9 (gdx (g* (var4 (gdx (gdx
(g* (var2 (gdx var1) var3) 2))) var5)
3))
```

```
          var10)
     (gdx (gdx (g* (var7 (gdx var6)
var8) 2)))))
    var11)
   ((var1 atm4 '((k (1 3)) (r) (k) (r)))
    (var2 glu-type gml)
    (var3 rep-ofst 1)
    (var4 glu-type gml)
    (var5 rep-ofst -1)
    (var6 atm4 '((r) (r) (r) (r)))
    (var7 glu-type gml)
    (var8 rep-ofst 1)
    (var9 glu-type gmf)
    (var10 glu-ofst -1)
    (var11 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "g")))
  ((gf=
    (gdx
     (g+
      (var15
       (g+
        (var10
         (gdx (g* (var4 (gdx (gdx (g*
(var2 (gdx var1) var3) 2))) var5) 2))
         var11)
        (gdx (g+ (var8 (gdx var6) var9)
(gdx var7))))
      var16)
     (gdx (gdx (g* (var13 (gdx var12)
var14) 2)))))
    var17)
   ((var1 atm4 '((k (1 0)) (r) (k) (r)))
    (var2 glu-type gml)
    (var3 rep-ofst 0)
    (var4 glu-type gml)
    (var5 rep-ofst 0)
    (var6 atm4 '((k (1 2)) (k (1 1))
(k (1 1)) (k)))
    (var7 atm4 '((k (1 -1)) (r) (k (1
1)) (k)))
    (var8 glu-type gml)
    (var9 glu-ofst -1)
    (var10 glu-type gmf)
    (var11 glu-ofst 1)
    (var12 atm4 '((k (1 -1)) (k (1 1))
(k (1 1)) (k)))
    (var13 glu-type gmf)
    (var14 rep-ofst 0)
    (var15 glu-type gml)
    (var16 glu-ofst 1)
    (var17 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
         "b"))))
 (((gf=
    (gdx
     (g+
      (var9 (gdx (g* (var4 (gdx (gdx
(g* (var2 (gdx var1) var3) 2))) var5)
3))
          var10)
     (gdx (gdx (g* (var7 (gdx var6)
var8) 2)))))
    var11)
   ((var1 atm4 '((k (1 2)) (r) (k) (r)))
    (var2 glu-type gml)
    (var3 rep-ofst 0)
    (var4 glu-type gmf)
    (var5 rep-ofst 1)
    (var6 atm4 '((k (1 -1)) (r) (r)
(k)))
    (var7 glu-type gml)
    (var8 rep-ofst 1)
    (var9 glu-type gml)
    (var10 glu-ofst 1)
    (var11
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "CC")))
  ((gf=
    (gdx
     (g+
      (var15
       (g+
        (var10
         (g+ (var8 (gdx (gdx (g* (var2
(gdx var1) var3) 2))) var9)
             (gdx (g+ (var6 (gdx var4)
var7) (gdx var5))))
         var11)
        (gdx (gdx (g* (var2 (gdx var1)
var3) 2))))
       var16)
      (gdx (gdx (g* (var13 (gdx var12)
var14) 2)))))
    var17)
   ((var1 atm4 '((k (1 -1)) (r) (r)
(k)))
    (var2 glu-type gml)
    (var3 rep-ofst 1)
    (var4 atm4 '((k (1 -1)) (r) (k)
(r)))
    (var5 atm4 '((k (1 -7)) (r) (k)
(r)))
    (var6 glu-type gml)
    (var7 glu-ofst 1)
    (var8 glu-type gml)
    (var9 glu-ofst -1)
    (var10 glu-type gml)
    (var11 glu-ofst 1)
    (var12 atm4 '((r) (r) (r) (r)))
    (var13 glu-type gml)
    (var14 rep-ofst 0)
    (var15 glu-type gml)
    (var16 glu-ofst -1)
    (var17 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "c")))
  ((gf=
    (gdx
     (g+
      (gml
       (g+
        (gml
```

```
         (gdx
          (g*
           (gml
            (gdx
             (g+ (gml (gdx '((k (1 2))
(r) (k) (r))) 1)
                 (gdx '((r) (r) (r)
(r)))))
            1)
           2))
         -1)
        (gdx
         (g+ (gml (gdx '((k (1 0)) (r)
(k) (r))) 2)
             (gdx '((k (1 1)) (k (1 1))
(k (1 -2)) (k))))))
       1)
      (gdx (gdx (g* (gml (gdx '((k) (r)
(r) (r))) 0) 2)))))
    var1)
   ((var1 ignored "cc"))))
 (((gf=
    (g+
     (var10
      (gdx
       (g+
        (var8
         (g+ (var5 (g+ (var3 (gdx var1)
var4) (gdx var2)) var6) (gdx var1))
         var9)
        (gdx var7)))
      var11)
     '((k) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r)))
    var12)
   ((var1 atm4 '((k (1 0)) (r) (k) (r)))
    (var2 atm4 '((r) (k (1 -1)) (k (1
-1)) (k)))
    (var3 glu-type gml)
    (var4 glu-ofst 2)
    (var5 glu-type gml)
    (var6 glu-ofst 1)
    (var7 atm4 '((k (1 -7)) (r) (k)
(r)))
    (var8 glu-type gml)
    (var9 glu-ofst -2)
    (var10 glu-type gmf)
    (var11 glu-ofst -1)
    (var12
     ("CC" "DD" "EE" "FF" "GG" "AA"
"BB" "C" "D" "E" "F" "G" "A" "B" "c")
     "F")))
  ((gf=
    (g+
     (var7
      (gdx
       (gdx (g* (var5 (gdx (g+ (var3
(gdx var1) var4) (gdx var2))) var6)
2)))
      var8)
     '((k) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r)))
    var9)
   ((var1 atm4 '((k (1 -1)) (k (1 1))
(k (1 1)) (k)))
    (var2 atm4 '((k) (r) (r) (r)))
    (var3 glu-type gml)
    (var4 glu-ofst 1)
    (var5 glu-type gml)
    (var6 rep-ofst 1)
    (var7 glu-type gml)
    (var8 glu-ofst 1)
    (var9 ("C" "D" "E" "F" "G" "A" "B"
"c" "d" "e" "f" "g" "a" "b") "f")))
  ((gf=
    (g+
     (var7
      (gdx (g+ (var5 (gdx (g* (var2
(gdx var1) var3) 3)) var6) (gdx var4)))
        var8)
       '((k) (r) (r) (r) (r) (r) (r) (r)
(r) (r) (r) (r) (r) (r) (r) (r)))
    var9)
   ((var1 atm4 '((r) (k (1 1)) (k (1
1)) (k (1 1))))
    (var2 glu-type gml)
    (var3 rep-ofst -2)
    (var4 atm4 '((k (1 5)) (r) (k) (r)))
    (var5 glu-type gml)
    (var6 glu-ofst -1)
    (var7 glu-type gml)
    (var8 glu-ofst 1)
    (var9 ("c" "d" "e" "f" "g" "a" "b"
"cc" "dd" "ee" "ff" "gg" "aa" "bb")
         "aa")))))
```

8.2 Music score of the composition