HarmoGen: A Four-Part Harmony Generator

Alec LaLonde
AlecL@mail.rit.edu
Project Lead, Design Lead, Content Expert
6168 Donnattsburg Rd.
Glenfield, NY 13343

Jason Morrison
jpm4819@cs.rit.edu
Utility Developer, Tool Expert
115B Audino Lane
Rochester, NY 14624

ABSTRACT

HarmoGen is an automated four-part harmony generation tool. Provided with a line of melody, the system automatically determines a major or minor scale and picks chords according to several four-part harmony rules of music theory. Several randomizing factors are built in so a unique harmony is generated almost every time.

1. INTRODUCTION

Over the past few years, many individuals have created four-part harmony generation tools with varying degrees of success. Genetic algorithms and rules-based expert systems have been the predominant approaches to this harmonization problem. A 1988 expert system created by K. Ebcioglu constrained itself to harmonizing chorales in the style of J.S. Bach [1]. A later harmonization approach utilized genetic algorithms (GAs) solely [2]. The decision to pursue a more iterative rules-based approach over a fitness function was heavily influenced by another paper comparing the two [3]. It was found in these latter two papers that a GA was not the optimal choice for solving the harmonization problem. Building on these findings, HarmoGen was developed as a rules-based expert system for generating a technically sound (and therefore pleasant to the human ear) four-part harmony.

Another major goal for HarmoGen was for it to be easily available to the public as a free download. A thorough search in June 2004 did not reveal any other four-part harmony generators on the Internet, much less for free. Thus, a focus on usability strongly influenced each stage in HarmoGen's development.

2. SYSTEM ARCHITECTURE

After an appropriate harmonization approach was chosen, the next step for HarmoGen was laying out a high-level design, shown in Figure 1. A three-layer design was chosen to separate the input, output, and internal processing parts of the system:

The first layer represents the input functionality. Here we had to decide what types of input to accept, and how HarmoGen would convert different types of input into a single form. Simple note strings were chosen, where a note consists of its letter plus an octave integer (i.e. C6). MIDI compatibility was an obvious choice due to its widespread use in computer music, so a MIDI parser was created as well. Since MIDI files don't contain any inherent information about key, HarmoGen requires the user to input it manually. Three forms of input were decided on: Text files containing note strings, direct keyboard input of note strings, and type-0 MIDI files. It was in this stage too that we decided to ignore any information about rhythm and focus solely on the notes that make up a harmony.

The processing layer represents the core of HarmoGen. Here contains the knowledge of the system: the explicit rules for chord choice, cadence preferences, note/chord representations, and key information. It also contains the central find-best-chord algorithm used for choosing chords, as well as the steps for creating cadences. A much more detailed look at this algorithm is presented in section four. After the entire melody has been harmonized, HarmoGen passes four note strings to the output layer.

The output layer is what the user sees after a successful harmonization. A dialog is diplayed over HarmoGen's main screen containing the generated note strings. Three output options are available: Save the harmony as a MIDI file, play the melody, or save the note strings to a text file. HarmoGen needed a MIDI tool to support converting note strings to MIDI files to be played or saved. The JFugue library [4] was chosen due to its functionality and easy compatibility with HarmoGen.

3. HARMONIZATION ALGORITHM

At the core of HarmoGen is its algorithm for generating four-part harmonies, detailed in Figure 2. It combines an iterative approach with randomizing factors for unique, aesthetic harmonies.

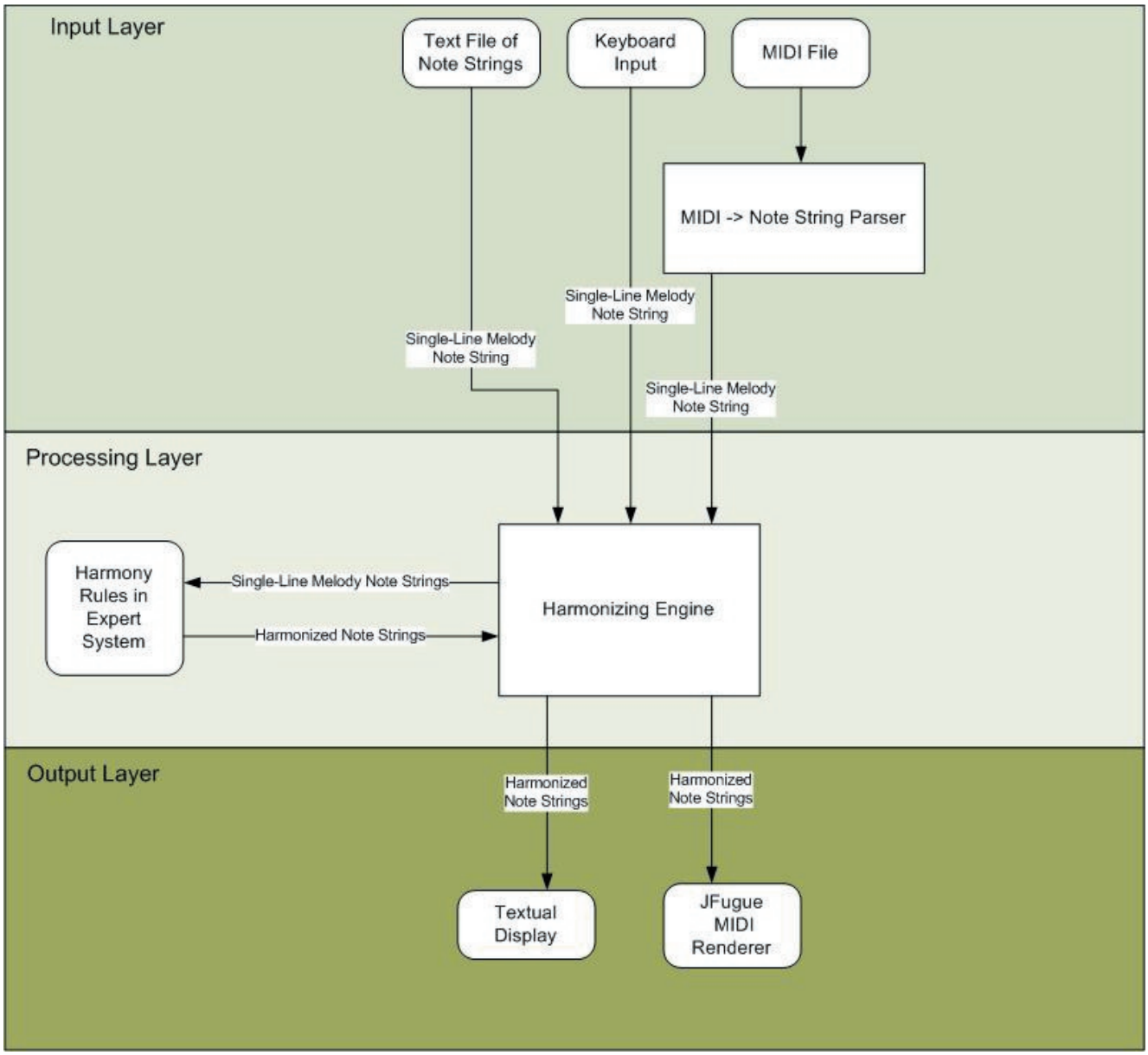

Figure 1. *HarmoGen's high-level design.*

3.1 CADENCE

First, a proper cadence is generated based on the given key and melody. A cadence is defined as the ending phrase of a melodic idea. It usually consists of two chords that provide a feeling of finality for the melody. However, different cadences can be utilized to provide a less conclusive feel to the melodic phrase, such as a half or deceptive cadence. These cadences are automatically generated in the current version of HarmoGen, but future releases may include user cadence selection. The order of preference for cadences are: Authentic, Plagal, Half, Deceptive. The first cadences are the most conclusive, while the last cadences are much less final. HarmoGen emphasizes the cadence first because it is the most significant and memorable part of the harmony.

3.2 INITIAL CHORD

Next, the first chord is generated. It is almost always the tonic for the key, in order to introduce the key right from the beginning. If the melody doesn't allow for a I chord, another chord is selected and the key is established in the next few chords instead.

3.3 BODY CHORDS

Body chords are the chords between the first chord and the final cadence. A "find-best-chord" algorithm was

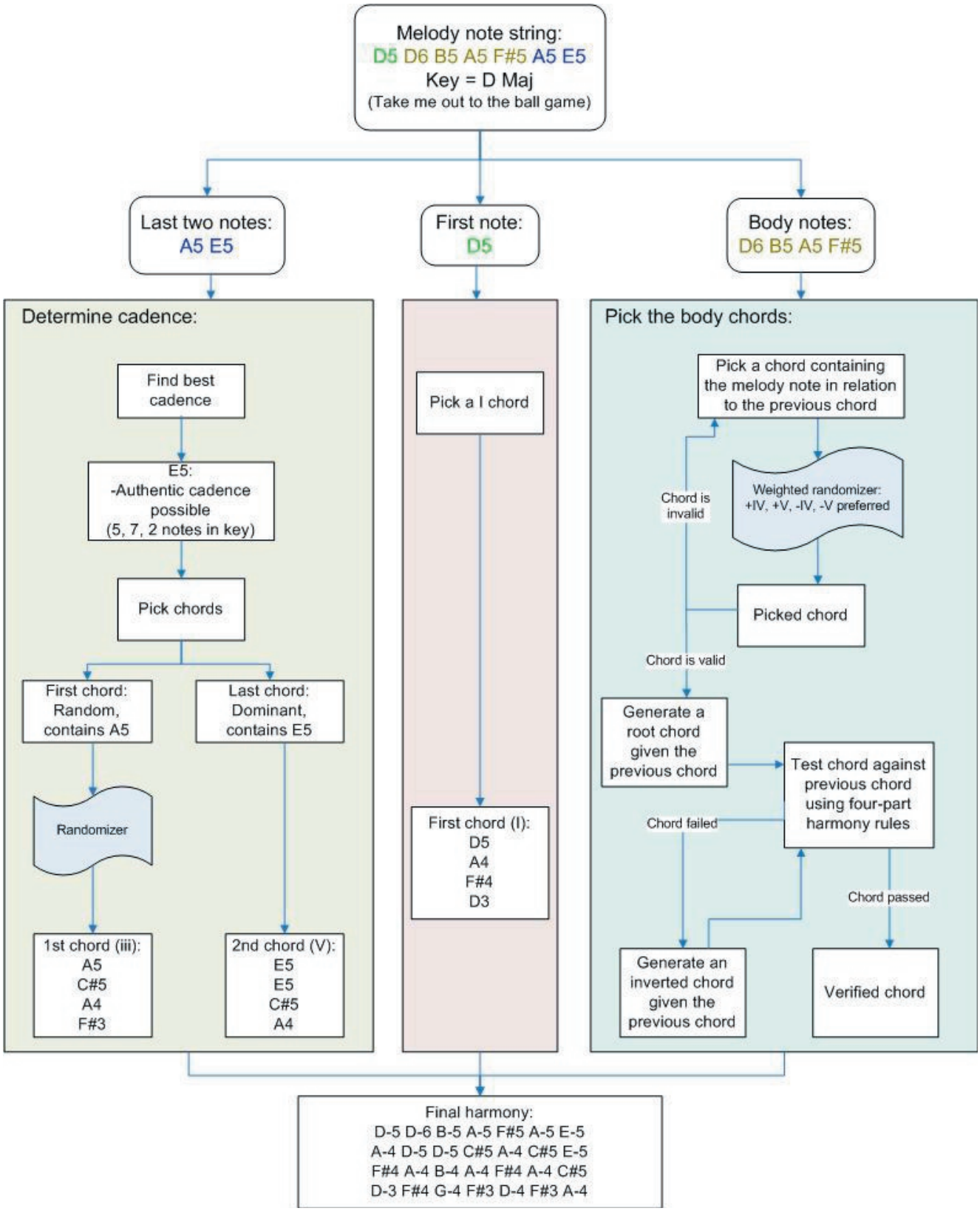

Figure 2. *HarmoGen's harmonization algorithm.*

developed to determine these chords iteratively, starting with the 2nd chord of the harmony. This algorithm uses a predetermined set of four-part harmony rules:

3.3.1 FOUR-PART HARMONY RULES

- Stepwise motion ideal
- Prefer doubling of tonic and third, not fifth
- Keep common tones whenever possible
- Bass should not leap greater than a perfect fifth, but octaves are ok
- No going outside of voice range
- No part crossing
- No parallel fifths
- No parallel octaves
- No gaps larger than an octave between voices (except for bass - tenor)
- Prefer start on I chord, root position
- Cadence preference order:
  1) V - I (Authentic)
  2) IV - I (Plagal)
  3) ? - V (Half)
  4) ? - vi (Deceptive)

3.3.2 FIND-BEST-CHORD ALGORITHM

This algorithm is the basis for selecting body chords based on the provided melody, key, previous chord, and rules declared above. The algorithm is the core piece of HarmoGen and consequently is being tuned constantly to produce better harmonies. The original algorithm is as follows:

1. Pick a random chord in relation to the previous chord (+V, -ii, +vii)
2. Determine notes of the selected chord
3. Generate chord (alto, tenor, bass notes) where the bass is the root of the chord
   a. If chord breaks a rule, generate inverted chord (tenor or alto is the root note)
   b. If inverted chord breaks a rule, repeat from beginning

HarmoGen first looks at the previous chord and determines its type in relation to the key (I, V, vii, etc.). It then picks a number at random in relation to this key type. The notes of this newly chosen chord are then found based on the key type. After the notes have been chosen, logic is passed to the chord generator. The chord generator picks the notes on the staff based on the notes chosen previously. For example, if a V chord is chosen in a G major key, the notes would be D, F#, and A. The chord generator would pick the octave for each note as well as which voice would play (sing) each note. The probability is such that the bass would sing the D, tenor the F#, and alto the A. Since the soprano part is melody the user has already provided it.

Each time a chord is generated it is checked to be sure it has passed each four-part harmony rule. If it fails any rule, another chord is generated until the chord passes. A limiting factor is built in so the system doesn't infinitely generate chords if no correct chord can possibly be generated. In this case, HarmoGen tries to generate an inverted chord. In the rare occasion that both types of chord fail, a different type of chord is selected based on the previous chord.

Randomization factors are built into each step. For instance, the chord picked in step 1 has the greatest probability of being either a fourth or fifth up or down since these chords usually represent strong, desired progressions. Doubled notes, parts of inverted chords, and octaves are all randomly generated; however, in most cases one choice will break a rule, so improper combinations are weeded out.

4. PERFORMANCE

The find-best-chord algorithm yields harmonies that seldom break the designated rules. However, parts are sometimes crossed when the given melodies constrict the ranges of the voices. Also, large gaps (jumps of an octave plus) in the bass part sometimes occur for similar reasons. Performing one of HarmoGen's four-part harmonies in a chorus would usually require an exceptionally talented Bass with a fantastic range.

Repetition is probably the greatest problem in HarmoGen's harmonies. The nature of the algorithm causes jumps of a fourth or fifth up or down most of the time, which often leads to less desirable chords and repetitive phrases. Fitness functions could be built in to detect undesirable chords as well as broken rules, so more suitable chord types could be chosen instead of purely random ones.

Figure 3 displays one of HarmoGen's renditions of "Jingle Bells." Included in the notation is each chord type picked by HarmoGen. It should be noted that doubled notes are only represented with one flag rather than two. In this example only the alto/soprano parts are occasionally doubled.

The most noticeable aspect of this harmony is the bass part. It tends to jump octaves at will. Since there is no restriction for the gap between a tenor and bass part, the bass tends to jump way down into a low register much more often than the other parts. Disallowing jumps of over

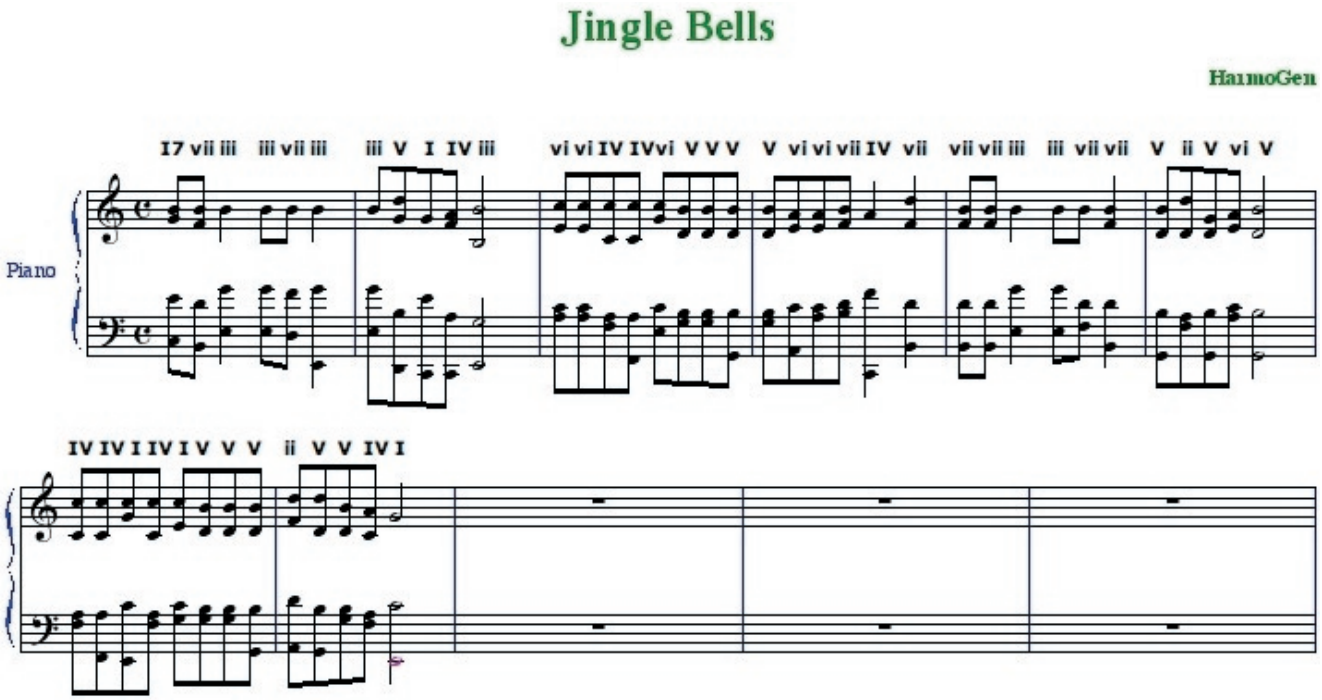

Figure 3. *A generated four-part harmony for "Jingle Bells."*

an octave, or even a fifth, would eliminate these inconsistencies.

It should be noticed that each four-part harmony rule defined in the system is followed throughout this example. Again, incorporating more rules and guidelines would produce harmonies of greater quality.

5. APPLICATIONS

A discussion of HarmoGen would not be complete without indicating some possible applications for the system. Admittedly, the system is still in a relatively early stage of development and could use a few tweaks before its harmonies could be easily performed by an amateur chorus or choir. However, the system already could be studied by itself for educational purposes.

Many Music Theory professors encourage students to create their own melodies that follow a few preset rules. These melodies, if in a soprano's range, could be inputted into HarmoGen directly. The resulting generated harmony could serve as an introduction to four-part harmony, an integral aspect of any course in basic Music Theory. Students could analyze the outputted harmonies, adding a chord analysis or identifying any errors after sufficiently studying the rules and guidelines for four-part harmony.

A more obvious application of HarmoGen could be to generate harmonies for a chorus. A conductor or soprano could write his or her own melody, and HarmoGen would eliminate the time-consuming task of writing parts for the other three voices. Further refinement would be necessary, however, for HarmoGen to create truly usable four-part harmonies.

6. EVOLUTION AND FUTURE GOALS

HarmoGen's core algorithm has been constantly tweaked since its original implementation in August, 2004. The original system used a command-line interface as well, which was replaced in early 2005 by a complete graphical user interface. However, these improvements represented only a few of the creators' goals for HarmoGen's future:

6.1 RHYTHM SUPPORT

HarmoGen currently ignores any and all aspects of rhythm. Manual or text-file input doesn't allow for any addition of rhythm information, while MIDI files are stripped of all rhythm information on input. Potential solutions could include:

- Harmonizing each note, keeping the original rhythm. Some sort of duration information would have to be inputted with each note. Then, the system would simply create three additional notes for every single note of the melody, keeping the same duration for each note in a chord.
- Harmonizing on certain beats of a measure. A user could indicate how often a chord is desired, i.e. on each beat, every other beat or each measure.

6.2 MULTI-VOICE MELODIES

HarmoGen currently only recognizes melodies within a soprano's range. It wouldn't be overly difficult to accept melodies for any voice, as long as the user indicated which voice the melody was for. The system would simply generate parts for the other voices.

6.3 TRANSITION TONE SUPPORT

Support for the automation or customization of escape tones, passing tones, neighboring tones, etc. could be added. Either HarmoGen would automatically add these tones or the user could have control over their placement, frequency, and/or duration.

6.4 NOTATION SUPPORT

Hooking into a third-party notation tool would greatly increase HarmoGen's potential for educational use. Generated harmonies could be displayed on a grand staff instead of the current clumsy text-based form. This would be a huge undertaking to do from scratch, however, due to the complexity of dynamic graphic displays; using a third-party notation tool would be essential.

7. CONCLUSION

HarmoGen was designed to provide complete, aesthetic four-part harmonies for a given melody. It has thus far completed this objective. Its I/O capabilities and MIDI support greatly enhance its usefulness, and further refinements are in the works. While the idea of an automated four-part harmony generator is not new, an available system comparable to HarmoGen has yet to be found.

HarmoGen can be freely downloaded at *http://www.rit.edu/~arl2548/HarmoGen* and used according to the instructions provided on the web site. Please direct any comments, complaints or suggestions to *AlecL@mail.rit.edu.*

8. REFERENCES

[1] EBCIOGLU, K. 1988. An expert system for harmonizing four-part chorales. *Computer Music Journal, 12*

[2] PHON-AMNUAISUK, S., TUSON, A., AND WIGGINS, G. 1999. Evolving Musical Harmonisation. In *Proceedings of Fourth International Conference on Neural Networks and Genetic Algorithms, ICANNGA 99*, Slovenia.

[3] PHON-AMNUAISUK, S., AND WIGGINS, G. *The Four-Part Harmonisation Problem: A comparison between Genetic Algorithms and a Rule-Based System.* Music Informatics Research Group, Division of Informatics, University of Edinburgh, UK.

[4] KOELLE, D. 2004. *JFugue – A Java API for Music Programming.* http://www.jfugue.org.