Generating Coherent Melodies by Modeling Four Part Harmony Rules

Brian Eubanks
University of Texas at Austin
brian.eubanks@utexas.edu

Christine Phan
University of Texas at Austin
christine_phan@utexas.edu

ABSTRACT

With the increasing popularity of machine learning and generative AI, music artists have attempted to generate melodies that are pleasing to the listener, but with mixed results. For instance, ChatGPT or any other generative AI can be prompted to provide a melody, but how can it be verified that the output melody is valid? One way to address the correctness of AI-generated music is to apply constraints to generate melodies and harmonies that are more likely to make coherent sense by implementing some rules for part writing.

Four-part writing style is a commonly used method for writing harmonies that includes four parts, soprano, alto, tenor, and bass. Four-part writing style imposes rules that must be adhered to in order to create coherent melodies.

Music21 is a Python toolkit that provides tools for composing and analyzing music. It provides musicians with a simple way to write and analyze music in symbolic forms and enables modularity and reusability.

This project aims to model a few of the four-part writing rules in Alloy and utilize the Music21 Python library to visualize and playback the generated melodies from Alloy. This approach can provide a potential framework and guardrails for verifying that automatically generated melodies are valid.

KEYWORDS

AI, testing, model generation, alloy, music21

ACM Reference Format:
Brian Eubanks and Christine Phan. 2024. Generating Coherent Melodies by Modeling Four Part Harmony Rules. In *Proceedings of Verification and Validation (Verification and Validation '24).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Artificial Intelligence is emerging as a new major advancement and has become an integral part of society. One notable model that has become increasingly popular due to its vast applicability is ChatGPT. A key question that arises in many machine learning applications is how correct is the given output? To determine correctness, a set of specifications must be defined to determine what constitutes a correct or incorrect output.

Verification and Validation '24, August, 2024, Austin, TX

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In the context of writing music, music theory presents practices that composers use to create music. Although not absolute guidelines, the practices can be thought of as a set of rules or specifications to follow when making music. [6]

This paper presents a model of some four-part harmony rules in Alloy, specifically the rules of no parallel fifth movements and no parallel octave movements. To limit the model to the music it can generate, a constraint of triads, or chords, is also added to the output space.

To verify that the four-part rules are modeled accurately, the Alloy instances are verified by a series of Python unit tests to ensure that the generated music follows the rules. The Alloy instances are parsed and passed as inputs to the Python unit tests.

An additional tool that has aided in the visualization and playback of the generated melodies is the Music21 Python library, a toolkit for computer-aided musical analysis licensed by MIT. [5]

Lastly, this paper presents a comparison between four-part harmonies generated by Alloy and ChatGPT to determine their correctness. The ChatGPT generated harmonies are parsed and used as inputs to the Python unit tests, similar to the Alloy instances.

2 FOUR-PART HARMONY

Four-part harmony is the study and application of tonality through harmonic progressions in four voices or parts. [3] These parts are known as soprano, alto, tenor, and bass, or SATB. The parts range relatively in pitch, starting with soprano at the highest pitch and bass at the lowest pitch. The way the four parts move and connect with one another to form a chord progression is known as a musical phrase.

When writing music, composers have historically followed best practices to achieve the most desirable sounds. These rules are well-defined and constrained. A few of the rules explored in this paper are the following:

- No Consecutive/Parallel Fifths
- No Consecutive/Parallel Octaves
- Smooth Voice Leading
- Double the Root and Bass Voice on the Root

2.1 No Consecutive/Parallel Fifths

When voices move in the same direction by the same interval, it is called a parallel movement. A semitone, or half step, is the smallest movement in music. Visually on a piano, it can be seen as two adjacent keys. A fifth is when two pitches, or two parts, are seven semitones apart. This rule states that voices must not move in parallel fifths. In other words, for two parts that begin a fifth apart, their next notes may not be a fifth apart. [2]

2.2 No Consecutive/Parallel Octaves

Similar to the parallel fifths, a parallel octave is when two parts are eight notes apart, or 12 semitones apart. This rule states that voices must not move in parallel octaves. In other words, for two parts that begin an octave apart, their next notes may not be an octave apart. [2]

2.3 Smooth Voice Leading

Voice leading is the process where the challenge is to create musical phrases in the four parts that are both harmonically coherent and logical within the key in which the phrase is written, and are melodically smooth. A key is the set of notes and chords that the piece of music is constrained to. [3] In order to create smooth voice leading, the parts must not move in leaps of large intervals, and the parts must not cross each other. For example, the Soprano shall always be pitched above the Alto, the Alto will always be pitched above the Tenor, and the Tenor will always be pitched above the Bass.

2.4 Double the Root and Bass Voice on the Root

A chord, or triad, consists of three notes stacked in consecutive thirds. The lowest note of a chord is known as the root. The middle note is the third, and the highest note is the fifth. Each of these notes must be covered by a part, and one part will double the root. This rule states that the bass voice must always cover the root of the chord, or the lowest note. [4]

Additionally, all notes of the triad

3 ALLOY MODELING FOUR PART HARMONY

The Alloy Analyzer provides a way to generate models for a given problem. The applications of the generated model instances are vast, but one useful application is to use the generated instances to generate test cases. [1]

3.1 Musical Staff and Parts as a Linked List

To begin, the Alloy model must define a measure of music. It is easy to visualize a single musical line as a linked list, where each node represents a note. Each note has a pitch value and a pointer to the next note. To account for four parts, each node will contain four pitch values, one each for Soprano, Alto, Tenor, and Bass.

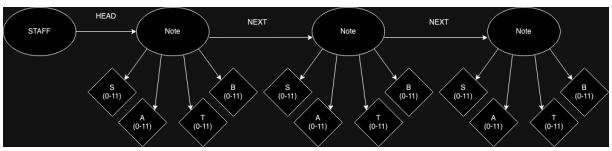

Figure 1: Visualization of Staff Data Structure as a linked list of Notes. Each note has a Soprano, Alto, Tenor, and Bass value to represent the pitch for each part.

For simplicity, this model ignores any rhythm values, meaning that all notes will be of the same duration, and each part will move together. Although it simplifies the model, it is not uncommon with four-part writing. Allowing for subdivisions of time would not only make the model more complicated, but it also makes the final analysis more complicated, as it would need to consider which notes were important to the chord and which were simply melodic passing tones. Figure 1 shows the staff data structure defined in Alloy. Figure 2 shows the signature of a Note in the Alloy model.

```
one sig Staff {
  head: one Note
}


sig Note {
  s: one Int,
  a: one Int,
  t: one Int,
  b: one Int,
  next: lone Note
}
```

Figure 2: Alloy Model of Staff with Notes.

3.2 Constraining Parts to Notes

Alloy defaults to using a 4-bit Integer to generate Int values. This limits the range of Int values to -8 to 7. Since there are only 12 possible note letters in an octave, the 4-bit Int range could have been used to represent pitches in the model. However, we decided to increase the Integer size to 5 bits so we could use Integers 0 to 11 to represent the 12 chromatic notes in an octave. Figure 3 shows a predicate in Alloy that constrains the note pitches from Integers 0 to 11.

```
//
// Set of Possible notes 1 - 11
//
pred irange(n:Int) {
   n = {0} or
                n = {1} or
               n = {2} or
                n = {3} or
                n = {4} or
                n = {5} or
                n = {6} or
                n = {7} or
                n = {8} or
                n = {9} or
               n = {10} or
                n = {11}
}
fact prange {
  all n: Note | irange[n.s] and irange[n.a] and irange[n.t] and irange[n.b]
}
```

Figure 3: Alloy Set of Possible Pitches represented as Ints.

3.3 Four-Part Rules: No Parallel Fifths or Octaves

In order to generate valid four-part harmony test cases, the first constraint implemented was to prevent parallel fifths. The original intent was to model the 12 chromatic notes, which move in half-step intervals, however we decided to model 12 diatonic notes instead. This further allowed the model to generate more realistic and pleasing harmonies, as well as allow a better implementation of rules such as parallel octaves, since we were no longer constrained to pitches within a single octave. Figures 4 and 5 show Alloy predicates for the parallel fifth and parallel octave intervals.

```
//
// Set of 5ths for Parallel 5ths
//
pred ser5[n1,n2: Int] {
  (n1 = {0} and n2 = {4} ) or
                    (n1 = {1} and n2 = {5} ) or
                    (n1 = {2} and n2 = {6} ) or
                    (n1 = {3} and n2 = {7} ) or
                    (n1 = {4} and n2 = {8} ) or
                    (n1 = {5} and n2 = {9} ) or
                    (n1 = {6} and n2 = {10} ) or
                    (n1 = {7} and n2 = {11} ) or
                    (n1 = {8} and n2 = {5} ) or
                    (n1 = {9} and n2 = {6} ) or
                    (n1 = {10} and n2 = {7} ) or
                    (n1 = {11} and n2 = {8} )
}
pred par5 {
  some n1,n2: Note | n1 !=n2 and n1.next = n2 and
                     ( (ser5[n1.s,n1.a] and ser5[n2.s,n2.a]) or
                       (ser5[n1.s,n1.t] and ser5[n2.s,n2.t]) or
                       (ser5[n1.t,n1.a] and ser5[n2.t,n2.a]) or
                       (ser5[n1.s,n1.b] and ser5[n2.s,n2.b]) or
                       (ser5[n1.b,n1.t] and ser5[n2.b,n2.t]) or
                       (ser5[n1.b,n1.a] and ser5[n2.b,n2.a]) )
}
```

Figure 4: Alloy Predicate containing the set of all possible fifth intervals between notes, and a predicate preventing two consecutive notes from being fifths.

3.4 Voicing Rules

Due to the small size and limitations of the model, more constraints on voicing rules were not implemented. This would involve ensuring that the motion of each voice is constrained to a set of all intervals below a Perfect Fourth. An additional constraint would also be needed to ensure the the Soprano is always a higher pitch than the Alto, the Alto is always higher pitch than the Tenor, and the Tenor is always above the Bass. Since the model only considers a small range of pitches, an octave and a half, it would severely limit the number of instances the model could generate. In order to implement these rules, the range of pitches would at least have to double to include multiple octaves and to account for the range between Bass and Soprano. Another concern is the time for Alloy to parse and solve for all of these constraints. Increasing the pitch range would drastically increase the total space and require significantly more time to reduce this total space to a valid input.

```
//
// Set of Octaves for Parallel 8ths
//
pred ser8[n1,n2: Int] {
  (n1 = {0} and n2 = {7} ) or
                    (n1 = {1} and n2 = {8} ) or
                    (n1 = {2} and n2 = {9} ) or
                    (n1 = {3} and n2 = {10} ) or
                    (n1 = {4} and n2 = {11} ) or
                    //(n1 = {5} and n2 = {0} ) or
                    //(n1 = {6} and n2 = {1} ) or
                    (n1 = {7} and n2 = {0} ) or
                    (n1 = {8} and n2 = {1} ) or
                    (n1 = {9} and n2 = {2} ) or
                    (n1 = {10} and n2 = {3} ) or
                    (n1 = {11} and n2 = {4} )
}
pred par8 {
  some n1,n2: Note | n1 !=n2 and n1.next = n2 and
                     ( (ser8[n1.s,n1.a] and ser8[n2.s,n2.a]) or
                       (ser8[n1.s,n1.t] and ser8[n2.s,n2.t]) or
                       (ser8[n1.t,n1.a] and ser8[n2.t,n2.a]) or
                       (ser8[n1.s,n1.b] and ser8[n2.s,n2.b]) or
                       (ser8[n1.b,n1.t] and ser8[n2.b,n2.t]) or
                       (ser8[n1.b,n1.a] and ser8[n2.b,n2.a]) )
}
```

Figure 5: Alloy Predicate containing the set of all possible eighth intervals between notes, and a predicate preventing two consecutive notes from being eigths.

3.5 Triad Rules and Bass Voicing

To limit the number of harmonies the Alloy model generated, triad rules and bass voicing were other constraints added. The triad rules generate harmonies where the soprano and alto parts double the root. The bass voicing constraint ensures that the bass part is always on the root of the triad. Figure 6 shows an Alloy predicate that constrains a set of possible triads and ensures the bass is on the root note and the other voices double the root.

4 CHATGPT FOUR PART HARMONY

To serve as a comparison for the Alloy-generated instances, we wanted to compare with four-part harmonies generated by ChatGPT. We asked ChatGPT to generate one measure of SATB harmony in quarter notes. The first output was correct, however it did not attempt to follow four part harmony rules. We asked it to do the same progression, but follow four part harmony rules. To generate more instances we simply asked it to generate another instance, and it would provide a new chord progression that it claimed followed the harmony rules.

5 MUSIC21

Music21 is a Python library from MIT that provides definitions for representing, manipulating, and analyzing musical data structures. It has vast applications, but for the purposes of this paper, it is used for the visualization and playback of the music generated by the Alloy model and ChatGPT. It has support for four-part writing, which is the key focus of this project. [5]

```
//
// Set of Triads
//   Double root
//   Root in Bass
//
// Positions of perspective T,A,S. Ignoring Bass,
// n4 = B, n1 = S
//
pred triad[n4,n3,n2,n1: Int] {

// 2nd Position
                    (n1 = {0} and n2 = {2} and n3 = {4} and n4 = {7}) or
                    (n1 = {1} and n2 = {3} and n3 = {5} and n4 = {8}) or
                    (n1 = {2} and n2 = {4} and n3 = {6} and n4 = {9}) or
                    (n1 = {3} and n2 = {5} and n3 = {7} and n4 = {10}) or
                    (n1 = {4} and n2 = {6} and n3 = {8} and n4 = {11}) or
                    (n1 = {5} and n2 = {7} and n3 = {9} and n4 = {5}) or
                    (n1 = {6} and n2 = {8} and n3 = {10} and n4 = {6}) or

                    (n1 = {0} and n2 = {9} and n3 = {11} and n4 = {7}) or
                    (n1 = {1} and n2 = {10} and n3 = {5} and n4 = {8}) or
                    (n1 = {2} and n2 = {11} and n3 = {6} and n4 = {9}) or
                    (n1 = {3} and n2 = {5} and n3 = {7} and n4 = {10}) or
                    (n1 = {4} and n2 = {6} and n3 = {8} and n4 = {11}) or
// 3rd Position
                    (n1 = {0} and n2 = {4} and n3 = {7} and n4 = {2}) or
                    (n1 = {1} and n2 = {5} and n3 = {8} and n4 = {3}) or
                    (n1 = {2} and n2 = {6} and n3 = {9} and n4 = {4}) or
                    (n1 = {3} and n2 = {7} and n3 = {10} and n4 = {5}) or
                    (n1 = {4} and n2 = {8} and n3 = {11} and n4 = {6}) or
                    (n1 = {5} and n2 = {9} and n3 = {5} and n4 = {7}) or
                    (n1 = {6} and n2 = {10} and n3 = {6} and n4 = {8}) or

                    (n1 = {0} and n2 = {11} and n3 = {7} and n4 = {9}) or
                    (n1 = {1} and n2 = {5} and n3 = {8} and n4 = {10}) or
                    (n1 = {2} and n2 = {6} and n3 = {9} and n4 = {11}) or
                    (n1 = {3} and n2 = {7} and n3 = {10} and n4 = {5}) or
                    (n1 = {4} and n2 = {8} and n3 = {11} and n4 = {6})
}

pred allTriad{
  all n: Note | triad[n.s,n.a,n.t,n.b]
}
```

Figure 6: Alloy Predicate containing the Set of Possible Triads with the Bass always covering the root and one other voice doubles the root. Contains combinations where the Soprano and Alto double the root. Missing 1st Position chords where the Tenor doubles the root.

5.1 Importing Data from Alloy or ChatGPT

To import the instances from Alloy, the Alloy text representation was parsed and converted to the note range defined in the Alloy model. Then, the integer notes were converted to the note format supported by Music21. The notes for each part were extracted and their order was preserved.

Similarly for ChatGPT, the chat log was parsed for each part, and the notes for each part were extracted and their order was preserved. There was no need to convert the notes from the ChatGPT text output to be compatible with Music21.

For both sets of data, the music generated contained four pitch values in a single measure of music. The data can now be rendered with Music21. A stream was created for each part, and each stream was appended to a score.

5.2 Unit Tests

To test the accuracy of the four-part harmony pieces generated by the Alloy model and ChatGPT, the various outputs of both were parsed and used as inputs to a series of Python unit tests to verify that the rules were followed. The Python unit tests check if the harmonies violate the no parallel fifth or no parallel eighth rules.

5.3 Visualising Measures

Music21 allows us to visualize and playback the generated measures. This lets us hear how realistic or normal the test cases sound. Figure 7 shows a rendering of an Alloy-generated music instance rendered with Music21. Most chord progressions from Alloy were more 'strange' or non-standard since we did not constrain the progressions. We found that ChatGPT measures seemed to give a more standard chord progression. Visualizing also makes it easy to search for any violations of the four-part harmony rules. Figure 8 shows a rendering of an ChatGPT-generated music instance rendered with Music21.

6 RESULTS

We generated a total of 16 Alloy instances and 6 ChatGPT instances. We rendered these instances with Music21 and ran our tests on them to see how closely they followed the Four-Part Harmony Rules.

6.1 Alloy Results

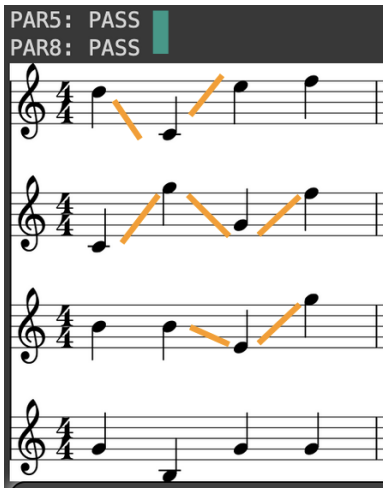

Figure 7: Rendering of an Alloy Generated instance. Passes Parallel 5ths and Parallel 8ths Test. Fails Voicing rules, shown with the yellow line.

Our initial results show that the Alloy-generated instances always pass parallel fifth and parallel octave tests. This is expected and encouraging that the model is successfully generating models that pass these tests. It can also be seen by manually analyzing the generated measures that all parts of the triad are covered, with the Bass voice always covering the root, and the root always doubled in another voice. However, by visual inspection, we can also see these instances do not pass the voicing rules. All generated instances contain large leaps between notes and often the voices cross over each other.

This is due to a limitation in our model that the pitches defined were only in a small range. As stated earlier, it would take a much

larger model to use the full range to reasonably add these voicing constraints to Alloy.

6.2 ChatGPT Results

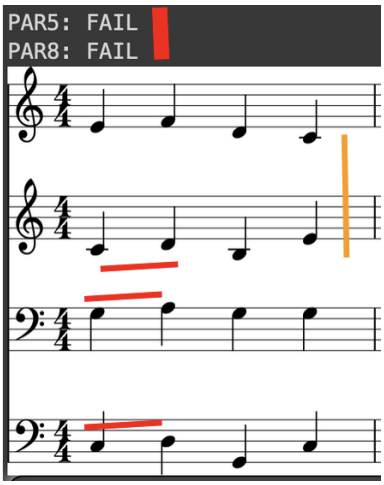

Figure 8: Rendering of a ChatGPT Generated instance. Fails both the Parallel 5ths and Parallel 8ths Test, shown with the red lines. Fails Voicing rules, shown with the yellow lines.

Our initial results with ChatGPT give interesting results. Despite confidently claiming to generate melodies that follow the Four Part Rules, these models clearly do not follow the rules. Five of the six generated instances fail either parallel fifths or parallel octaves, and the last one fails both. One generated instance passes the parallel fifth and parallel octave test, but it is seen to be in violation of the voicing rule by crossing voices.

7 CONCLUSION AND FUTURE WORK

Alloy has proven to be a useful tool for implementing music theory rules and generating valid four-part harmonies. Since the Alloy model was constrained to create triads, the instances it generated were limited but sufficient to conclude that Alloy is a useful tool for implementing part-writing rules to generate coherent melodies.

Compared to ChatGPT, the Alloy model produced more accurate results for the four-part harmonies, as evident from the Alloy instances passing more unit test cases than the ChatGPT output.

Although the Alloy model passed the tests for the parallel fifth and parallel octave tests, it was not in full compliance with all the four-part writing rules. To improve the generated music, the Alloy model could be easily expanded to model more four-part writing rules, such as avoiding leaps and using proper spacing for parts. With the addition of more part-writing rules, the constraint of generating chords can be stricter to only create certain chord progressions or loosened to create a variety of melodies.

Future research can also be conducted to see if there are other tools to generate more test cases or model more rules. Once more rules are implemented, the Python unit tests can be expanded to test the accuracy of the generated melodies.

8 RESOURCES

Below is a link to a Github repository containing several Alloy models showing the incremental development of the model, as well as the ChatGPT text log of inputs and outputs. It also contains a Python notebook containing all of the Python code for importing the Alloy and ChatGPT inputs for test cases. This Python notebook also contains the unit tests for parallel fifths and octaves and allows for rendering and playback of generated models.

The models were written with Alloy Analyzer 6.10.0.

The ChatGPT test inputs were from OpenAI's free version of ChatGPT.

(1) https://github.com/BrianEubanks/FourPartHarmony

REFERENCES

[1] Alloy [n. d.]. About Alloy. https://alloytools.org/
[2] Important Rules for 4-Part Progressions [n. d.]. https://davesmey.com/theory/partwritingrules.pdf
[3] Elliott Miles McKinley. 2022. *COMPOSING MUSIC: FROM THEORY TO PRACTICE.* Elliott Miles McKinley.
[4] Music Theory for the 21st-Century Classroom [n. d.]. https://musictheory.pugetsound.edu/mt21c/TriadsIntroduction.html
[5] Music21 2024. music21 Documentation. https://www.music21.org/music21docs/
[6] Wikipedia 2024. Music theory. In Wikipedia. https://en.wikipedia.org/wiki/Music_theory