Procedural Generation of Several Instrument Music Pieces with Hierarchical Wave Function Collapse

Raphael de Wolff[1]

Supervisor(s): Rafa Bidarra[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Raphael de Wolff
Final project course: CSE3000 Research Project
Thesis committee: Rafa Bidarra, Joana de Pinho Gonçalves

Procedural Generation of Several Instrument Music Pieces with Hierarchical Wave Function Collapse

Raphael de Wolff
TU Delft
Delft, The Netherlands

ABSTRACT

Wave Function Collapse (WFC) can be described as a family of algorithms, meant for content generation through constraint solving. One variant is Hierarchical WFC, where a hierarchical structure is given to the tileset used in WFC. This variant has seen use in a mixed-initiative procedural music generation model, where pieces of music simulating a single instrument playing chords and a melody are generated. In this paper, we explore how this model can be adapted to generate (coherent) pieces of music simulating several instruments playing in parallel. To achieve this, three model properties had to be defined: a new canvas structure, a set of constraints that instruments impose on each other, and the manner in which cells of all canvasses are collapsed. A hierarchical structure has been defined of a single section canvas, of which every cell has an inheriting chord canvas, of which every cell has an inheriting melody canvas for each instrument. The section cells impose constraints on inheriting chord and melody cells, and the chord cells impose constraints on inheriting melody cells. Besides this, melody cells impose constraints on cells belonging to other melody canvases (i.e. other instruments). With this canvas structure and these sets of constraints, the cells could be collapsed, such that coherent pieces of music simulating several instruments playing in parallel were generated.

CCS CONCEPTS

• **Applied computing → Sound and music computing**.

KEYWORDS

wave function collapse, procedural music generation, procedural classical music generation, constraint programming

ACM Reference Format:
Raphael de Wolff. 2024. Procedural Generation of Several Instrument Music Pieces with Hierarchical Wave Function Collapse. In *Proceedings of Research Project (CSE3000).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

CSE3000, June 23, 2024, Delft

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

1 INTRODUCTION

Since it's inception by M. Gumin in 2016, the Wave Function Collapse (WFC) algorithm [3] has been used for many instances of procedural content generation (PCG) [5]. This is most often involving the procedural generation of images or textures for video games. However, as proven by P. Varga and R. Bidarra, it is possible to use WFC for procedural music generation (PMG) [10]. Varga and Bidarra have developed a model for PMG using Hierarchical WFC (HWFC), a variant on the standard WFC algorithm initially researched in 2023 by Alaka and Bidarra [1]. This model is described in an as of yet unpublished paper made accessible to us [11]. This model procedurally generates a chord canvas and a melody canvas using HWFC, with the values representing a series of chords and notes played on a single instrument. If we look at man-made pieces of music, most of them are played with several instruments instead of with a single instrument, so now that it has been shown WFC can be used for PMG with a single instrument, it is a logical next step to check whether WFC can be used for PMG with several instruments.

We look at how HWFC can be used to procedurally generate pieces of music with multiple instruments playing in parallel. It should not just simulate several instruments playing random notes. Instead, there should be some form of coherence between the instruments, like in pieces of music composed fully by humans. We take the model described by Varga and Bidarra as a base, and attempts to extend this model. To do this, we study several subproblems: it looks at which constraints exist that can be enacted on the relation between several instruments' melodies. It looks at different ways to represent several instruments' sections, chords and melodies in a canvas structure, different ways of performing generation on these canvas structures and it assesses the quality of these different canvas structures and algorithm variations.

2 RELATED WORK

In this section we touch upon work done on WFC-based music composition, other work done on generative tools for music composition, and finally work done on the WFC algorithm itself which is relevant for our problem.

2.1 WFC-based music composition

As stated before, P. Varga and R. Bidarra have done, and are still doing, substantial work in HWFC-based music composition. They have proposed an HWFC-based model for composing music [10], and have an improved version of the model ready, which has been submitted for publication and made available to us [11]. It has a hierarchical structure of a section canvas, one or more chord canvases inheriting from the section cells, and melody canvases inheriting from the chord cells, where the chord canvases inherit

from the section cells, and the melody canvases inherit from the chord cells, as displayed in figure 1. Each canvas has its own set of constraints, and if it has a parent layer it also inherits relevant constraints from that layer.

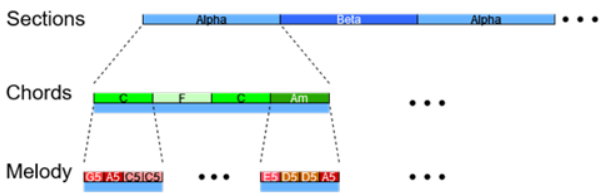

Figure 1: A diagram of an example composition using the model proposed by Varga and Bidarra, taken from the paper describing their model [11]. Only part of the structure has been displayed here, due to space constraints.

2.2 Generative music composition tools

For other generative music composition tools, we specifically look for those which generate several voices based on some user input.

Google's Creative Lab, in collaboration with Magenta, Google Brain's research project on music and art generation, introduced the Bach Doodle [4], a machine learning experiment that enables users to create music in the style of Johann Sebastian Bach. This tool leverages a machine learning model known as Coconet, which was trained on over 300 of Bach's chorale harmonizations. Users can input a melody, and the system generates a polyphonic composition in Bach's style, introducing them to the concept of counterpoint where multiple independent melody lines play together harmoniously.

Louie et al. [6] explored deep neural networks as a means to taming outputs of a generator based on Coconet. They introduced AI-steering tools, which, among other pieces of functionality, could: restrict voices to a given range, make the piece more/less similar to a given example piece, and adjust the piece semantically based on various emotion-based sliders. In their user study, they found that novices to composition found the generator more controllable, comprehensible and trustworthy when steered by their tools.

2.3 WFC extensions

The original WFC algorithm by M. Gumin [3] inspired much research into PCG with WFC. Most relevant to our paper is work related to HWFC, as this is the variant we are basing our model on.

The integration of a hierarchical tileset into the WFC algorithm has been shown to yield significant improvements in procedural content generation. Alaka and Bidarra [1] introduced Hierarchical WFC (HWFC), an extension to WFC featuring meta-tiles, i.e. intermediate elements (e.g. forest) which can stand for a group of possible tiles with a specific meaning (e.g. bush. pine, oak, grass...). Their work shows how the introduction of a hierarchy of tiles can improve interactivity and control in the design process. Similarly, Beukman et al. [2] reinforced this, proposing a different hierarchical approach to enhance the diversity and control of level generation. Of course the aforementioned Varga and Bidarra have made use of HWFC for procedural music generation [10][11].

3 METHODS

In this section we explain how we have designed and tested several HWFC-based model alternatives and a set of constraints, such that music pieces simulating several instruments playing in parallel can be procedurally generated. We will first touch upon the design of the models, then we will touch upon designing the constraints, and finally we will touch upon the testing process.

3.1 Designing the model

For designing the model, we have taken the model proposed by Varga and Bidarra discussed earlier as a baseline. Their model is designed for composition of music pieces with a singular melody played over a chord progression.

To avoid "reinventing the wheel", we have chosen to adjust their model and build upon it. First however, we make some assumptions. It is incredibly uncommon in music composition for different instruments to be playing different chord progressions. Furthermore, while one instrument is in a certain section (e.g. the 'verse' section), another instrument cannot be in a different section. Thus, the collapsed cells in the section canvas and chord canvases should hold for each instrument. Therefore we can say the section and chord layers stay the same for any models we design.

3.2 Designing the constraints

Varga and Bidarra [11] have created a set of constraints for each hierarchical level [9]. We will reuse this set of constraints. This set however is not sufficient for our problem, as there should be some cohesion between the instruments in music generated with our model. Therefore constraints should be added which cells belonging to different instruments can impose on one another.

As each instrument shall use the same section and chord layers, it is only the melody layer where cells will have some difference in terms of which instrument they belong to. Melody cells belonging to separate instruments should impose this new set of constraints on one another. This new set of constraints we will refer to as intermelody constraints.

The goal of intermelody constraints is to have the melody canvasses follow certain rules, such that the melodies sound good together. We chose to look for rules in classical music composition, and specifically four part composition. Four part composition is the composing of music with four different melodic parts (also referred to as voices), and its ruleset is perfect for creating a set of intermelody constraints. For the ruleset we use a lesson created by Dave Smey in 2004 [8].

The lesson assumes four different parts which play notes of different height relative to each other. The parts from low to high are: bass, tenor, alto and soprano. Parts which are not the bass are also referred to as the "upper voices". The lesson can be reduced to the following set of rules and good practices:

- **Proper spacing**: Upper voices are not allowed to be more than an octave apart.
- **No parallel fifths**: Any two parts that make a fifth, are not allowed to immediately make another fifth.

- **No parallel eights**: Any two parts that make an octave, are not allowed to immediately make another octave. Playing the same note in the same octave also counts as making an octave here.
- **No direct octaves/fifths**: If the soprano and bass parts make an octave or a fifth, the previous notes they played cannot both be higher or lower than the notes they are currently playing, unless the previous soprano note is just one step higher or lower than the current note.
- **No large leaps**: A part should not transition to a note that is a 7th or further than an octave away from its current note.

4 PROPOSED MODELS

This section outlines the proposed models. It first outlines two proposed canvas structure, and argues why one is better than the other. Then it explains several different ways of applying WFC to the canvas structure.

4.1 Canvas structure

As discussed before, our canvas structure has the same section and chord layers as Varga's model. The difference comes in in the melody layer. Each instrument plays a separate melody, and thus should have some set of cells representing the notes they play.

One important choice influencing the design of this melody layer, is in what way intermelody constraints are applied. We have chosen that intermelody constraints are applied by melody cells belonging to separate instruments based on the time the notes in the cells are played. If two cells would be played at the same time, intermelody constraints apply between these two cells.

One initial idea is for the total melody played by all instruments to be represented on a 2-dimensional canvas, where one dimension represents the order of notes, and the other by what instrument a note is played. However, for the manner in which we wish to apply intermelody constraints, this structure is unnecessarily awkward.

A second, better idea, is for each instrument to have their own one dimensional melody canvas. This method is in terms of constraint applying the same as the first idea, except that the structure is more clear. Furthermore, the better separation of instrument canvasses also makes different ways of applying WFC (i.e. order in which to collapse different instruments' cells) much more clear. The total structure is displayed in figure 2.

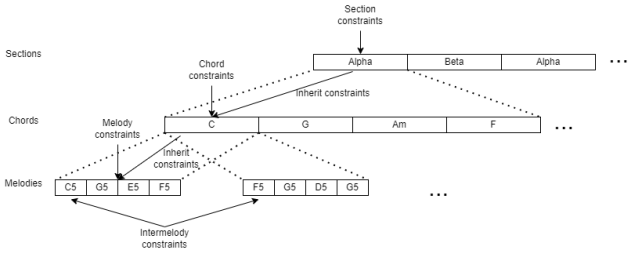

Figure 2: The canvas structure proposed by us. Not all parts of the structure are displayed due to space constraints.

4.2 Applying WFC

After defining the structure of our canvas, we still need to define how we will apply WFC to it. The basis of the algorithm is clear: First we create a section canvas and collapse all of its cells, based on section constraints defined by the composer. Then, for each section cell, we create a chord canvas inheriting from that cell, and we collapse the cells of this canvas according to the inherited constraints and the chord constraints defined by the composer. Then, for each chord cell, we create a melody canvas inheriting from that cell for each instrument.

Cells from such a melody canvas should be collapsed according to inherited constraints and melody constraints defined by the composer, but also according to intermelody constraints defined by the composer. Intermelody constraints, as stated before, should be imposed between melody canvasses. When a melody cell is collapsed, it represents a note being played at some time. The cell then looks for melody cells belonging to other instruments, which represent the note or range of notes being played at that time by that instrument. Based on the value to which the melody cell has collapsed and our intermelody constraints, the set of possible values these other cells can collapse to are updated.

It's unclear in what order we want to collapse the cells of the melody canvasses. We have designed 4 different ways of collapsing these canvasses, outlined below.

Naive collapse. Choose a melody canvas to collapse, and collapse all its cells. Repeat this process until each melody canvas is fully collapsed.

Random collapse. Choose a melody canvas to collapse one cell of. Collapse one cell of this melody canvas. Repeat this process until each melody canvas is fully collapsed.

Random k-collapse. Choose a melody canvas to collapse k cells of. Collapse k cells of this canvas. Repeat this process until each melody canvas is fully collapsed.

Jam collapse. The concept this method is trying to replicate, is that one instrument leads the melody and the rest follow this leader. Thus also the name *Jam* collapse. The method goes as follows: First choose a melody canvas to collapse, and collapse all its cells. Then, apply either *Random collapse* or *Random k-collapse* on the rest of the melody canvasses.

5 PROPOSED CONSTRAINTS

This section outlines constraints chosen for the proposed models. It first touches upon hierarchical section, chord and melody constraints, borrowed from the model proposed by Varga. Then it outlines the list of intermelody constraints designed by us.

5.1 Hierarchical constraints

As stated before, we reuse constraints designed by Varga and Bidarra, which can be seen in the tool they have developed as proof of their model's functionality [9]. We have made an adjustment to the limits of not-inherited constraints applied to the melody layer. Since we have a separate melody canvasses for each instrument, and each instrument has the same section and chord canvas, melody constraints not inherited from the section or chord canvasses should be able to be different for each instrument. Thus in our proposed

model, melody canvasses belonging to separate instruments are able to have different, non-inherited melody constraints.

It should be noted that the *No large leaps* rule identified in the 4-part composition ruleset earlier can be applied using the *Melody absolute step size* constraint designed by Varga and Bidarra.

5.2 Intermelody constraints

As stated before, intermelody constraints are applied between melody cells belonging to separate instruments, which represent notes being played at the same time. If some instrument's melody cell is collapsed, the value of this cell is used in combination with the intermelody constraints to update the set of values other instruments' relevant cells can collapse to. We have converted all rules identified from the 4-part composition ruleset to intermelody constraints, except for the *No large leaps* rule, as this as stated before can be represented better as a hierarchical constraint. A boolean-esque description of each of these intermelody constraints is given below.

- **Proper spacing**: Holds if one of the cells belongs to the first (lowest) instrument, or if the cells have collapsed to notes no further than an octave apart.
- **No parallel fifths**: Holds if the cells have collapsed to notes that do not make a fifth, or if the next and previous cells of both canvasses have not yet all collapsed, or have collapsed to notes which do not make a fifth.
- **No parallel eights**: Holds if the cells have collapsed to notes that do not make a octave or play the same note, or if the next and previous cells of both canvasses have not yet all collapsed, or have collapsed to notes which do nott make a octave or play the same note.
- **No direct octaves/fifths**: Hold if the cells do not belong to the first (lowest) and last (highest) instrument, or if the cells have collapsed to notes which do not make a fifth or an octave, or if the previous cells of both canvasses have not yet all collapsed, or they have collapsed to values which are not both higher or lower than the values the current cells have collapsed to.

6 IMPLEMENTATION

Varga and Bidarra's model, which we heavily build upon in our model and constraint design, has been implemented in the application *ProceduraLiszt*[9]. The repository has been made available to us, and thus we built upon its source code to implement a proof of concept of our proposed models and constraints. This implementation is also what we have used in the next section to evaluate our designs.

7 EVALUATION

This section first touches on the procedure used to evaluate our proposed models and constraints. Then it outlines the results of this performing these tests.

7.1 Procedure

We test the set of constraints together with each model variant on both efficiency and failure rate, and we attempt to assess some qualitative features of each model variant.

The failure rate entails how often WFC generation of a piece of music fails (i.e. a conflict is found), and the efficiency metric entails how fast a successful generation is on average.

We test different combinations of model variant, amount of instruments (2, 3 and 4), and set of constraints. Each combination shall have a BPM of 120, 1 section, 2 chords, a melody length of 4, and a key of C Major. Furthermore, each chord canvas shall have the *Chord in key* and *Chord absolute step size: Everything but 0* constraints, and each melody canvas shall have the *Melody in key* and *Melody range: C5-B6* constraints. The different sets of constraints added on top of these settings will be:

(1) Basic constraints.
(2) All 4-part composition related constraints (all intermelody constraints, and the *Melody absolute step size* constraint with as input the range [0, 12]).
(3) All 4-part composition related constraints, and different melody constraints for each instrument, namely *Descending melody* for instrument 1, *Ascending melody* for instrument 2, *Melody starts on root of chord* for instrument 3, and *Melody ends on root of chord* for instrument 4.

We also test the influence of the number of intermelody constraints used on the efficiency and failure rate. We use the same basic parameters for the previous setup, and then we test each model variant with 3 instruments and an increasing number of intermelody constraints. We add the intermelody constraints in the order they are listed in section 5.2, so we first use *Proper spacing*, then we use *Proper spacing* and *No parallel fifths* and so on.

For both tests, we generate each combination 10000 times and measure the portion of failures, and the speed of successful generations.

Besides efficiency and failure rate, we also test our models on quality. We listen to some of the successful generations from the first testing setup above, and we comment how they sound, and other features the different model variants appear to have. As these are our own interpretations which can not be represented in an objective manner, these interpretations are not touched upon in section 7.2, however they are discussed in section 8.

7.2 Results

The results of the first test in the evaluation procedure, applied to our designed model variants and set of constraints, are displayed in Table 1 below.

The average time it takes to generate a piece of music ('Avg. efficiency (ms)') is quite low for all combinations, but does go up with increased complexity.

Conflicts do not occur for the basic constraint set. They do start occurring when 4-part composition related constraints are introduced, and they occur over 98% of the time for the combination of 4-part composition related constraints and 4 instruments.

The results of the second test in our evalution procedure, which takes a closer look on the influence of the amount of intermelody constraints on the efficiency and failure rate for the different model variants, are displayed in Table 2 below.

Table 1: Efficiency and failure rates of different instrument amount, model variant and constraint combinations

Variant	Constraint set	No. instruments	Avg. efficiency (ms)	Failure %
Naive collapse	Basic	2	1.490	0
		3	2.170	0
		4	2.683	0
	All 4-part composition	2	1.835	60.1
		3	2.708	94.7
		4	3.478	99.8
	All 4-part composition, some melody	2	2.029	51.5
		3	2.820	96.7
		4	N/A	100.0
Random collapse	Basic	2	1.521	0
		3	2.162	0
		4	2.709	0
	All 4-part composition	2	1.858	67.2
		3	2.721	88.1
		4	3.648	99.1
	All 4-part composition, some melody	2	2.070	65.7
		3	2.884	91.5
		4	3.696	98.9
Random k-collapse (k=2)	Basic	2	1.531	0
		3	2.164	0
		4	2.732	0
	All 4-part composition	2	1.855	62.8
		3	2.709	87.1
		4	3.635	98.6
	All 4-part composition, some melody	2	2.023	69.7
		3	2.895	90.8
		4	3.674	99.6
Jam collapse (k=2)	Basic	2	1.534	0
		3	2.172	0
		4	2.708	0
	All 4-part composition	2	1.845	71.3
		3	2.694	90.7
		4	3.600	99.7
	All 4-part composition, some melody	2	2.078	54.8
		3	2.891	91.3
		4	3.700	99.9

The average time to generate a piece of music is again quite low, but does go up with added complexity (i.e. more intermelody constraints).

The failure rate generally goes up with added complexity. The failure rate is highest for *Naive collapse* and *Jam collapse*, and lowest for *Random collapse* and *Random k-collapse*.

8 DISCUSSION

In this section we discuss the results outlined in the previous section. We make some notes on the raw speed of each model variant, on the failure rates of each model variant and some qualitative features of each model variant.

First, in terms of raw speed, all the models are quite fast. Computation time goes up with the amount of constraints and instruments, but since the goal of this model is to generate just one piece of music, and not thousands of pieces of music in one go, each model variant is fast enough.

Second, the failure rates show some more interesting differences between the models. The failure rates start going above 0 as 4-part composition related constraints are introduced, and then they go up steeply for more than 2 instruments. When the model fails to find a solution, it means it made some decision in the past which ended up making it impossible to find a valid value for some tile. As we introduce more complicated constraints, or more instruments, the set of valid solutions becomes smaller, and thus the chance of finding a valid solution in one go becomes smaller. Thus it makes sense that the failure rate goes up quite steeply.

Table 2: Efficiency and failure rates with 3 instruments, for different model variants and a different amount of inter-melody constraints

Variant	No. intermel. con.	Avg. eff. (ms)	Failure %
Naive collapse	1	2.155	94.1
	2	2.238	94.4
	3	2.329	94.4
	4	2.351	96.2
Random collapse	1	2.162	85.8
	2	2.318	86.5
	3	2.366	87.2
	4	2.473	93.2
Random k-collapse (k=2)	1	2.172	86.3
	2	2.335	86.5
	3	2.365	86.6
	4	2.491	92.3
Jam collapse (k=2)	1	2.175	94.1
	2	2.298	95.0
	3	2.326	94.4
	4	2.415	95.6

It should be noted that there is some variance in failure rates, and it is highest for combinations which have 2 instruments and include all 4-part composition related constraints. This is evident from the *Naive collapse* and *Jam collapse* results in Table 1, where the combinations with 2 instruments which include 4-part composition constraints see their failure rates go down when some melody constraints are introduced. Seeing the failure rate go down with added complexity is unexpected, but can be explained by a high variance.

The algorithm which handles higher complexity the worst appears to be *Naive collapse*, as its failure rate is 100% for the most complicated combination of parameters. *Random collapse* appears to narrowly be the best at handling high complexities, although this could also be due to variance. Another explanation could be that the most complicated set of parameters has constraints that restrict certain tiles to single values (*Melody starts/ends on root of chord*), and *Random collapse* has the highest chance of collapsing these tiles first.

We have also listened to some of the successful generations of the tests used to generate the results represented in Table 1. Music pieces generated by each model variant do sound like attempts at classical music, and there is definitely some coherence. Music quality is subjective, but it does seem clear that the quality of these pieces of music is still much lower than pieces of classical music composed by professional composers, or pieces of classical music composed by state of the art models such as the Bach Doodle [4].

The model variants which appear to generate the most coherent music are the *Random k-collapse* and *Jam collapse* variants. The *Naive collapse* variant appears to be biased towards generating music playing notes in the higher end of the melody range. This behaviour can be explained by *Naive collapse* collapsing the lowest instrument's melody canvas first, which would likely restrict the other instrument's melody ranges to the higher notes of the melody range. The *Random collapse* variant appears to generate the most random pieces of music.

9 CONCLUSION

We have designed a model which is able to generate pieces of music simulating several instruments playing in parallel. Due to having constraints restricting the model to 4-part composition rules, the model can produce coherent music. All variants of the model are sufficiently fast. For more complicated sets of constraints, the *Random collapse* variant of the model has the lowest rate of failure, however the *Random k-collapse* and *Jam collapse* variants of the model appear to produce the most coherent pieces of music. Thus it is unsure which model variant can be deemed the best.

The failure rate for complicated sets of constraints is still quite high, and thus the model would heavily benefit from integrating backtracking into the model.

10 RESPONSIBLE RESEARCH

This section will first touch upon ethical concerns considered during this research, after which it will look at the reproducibility of our work.

10.1 Ethical concerns

For ethical concerns regarding PMG using HWFC, we have looked at the ethical considerations with regards to AI raised by Olaoye and Potter [7].

Firstly, it is important that the model we have described is transparent and explainable. We think our model qualifies, as the algorithm can be easily followed by anyone, even with tools as simple as a pen and a piece of paper.

The second metric is fairness and bias. As our model does not deal with data related to persons, there can be no notion of discrimination or inequality with relation to individuals, and thus we think our model satisfies this metric.

The third metric is privacy and data usage. As the values used to generate pieces of music are merely primitive musical notes, which are not owned by anyone, our model does not use privately owned data. Therefore our model satisfies this metric aswell.

The fourth metric is accountability and responsibility. This metric could be applicable to our model, if a user were to use it to compose a piece of music so similar to some existing song, that the user suffers negative consequences related to copyright laws or something similar. The chance that a piece of music is produced that is nearly exactly similar to an existing song is incredibly low. Furthermore, our model is mixed-initiative, so the human composer still plays a large part in the composition process, and thus bears part of the responsibility for not publishing copyrighted music. Therefore, we deem that this is in fact not a concern, and that our model satisfies the accountability and responsibility metric.

The fifth and final metric is robustness and safety. Our model has a defined set of inputs, namely the amount of instruments, and the set of constraints. Our model can handle this set of inputs, i.e. it can generate a piece of music, or indicates it when it is not possible

to generate a piece of music. Therefore, our model satisfies this metric aswell.

10.2 Reproducibility

We feel that we've thoroughly described our model, and thus anyone with some experience in programming can create an experimental setup similar to ours. The reproduction might not have a sophisticated GUI that we had the opportunity to incorporate, as we were extending an already existing webapp. It will however be able to perform the same experiment that we have done. Therefore, we feel our work is reproducable.

REFERENCES

[1] Shaad Alaka and Rafael Bidarra. 2023. Hierarchical Semantic Wave Function Collapse. In *Proceedings of the 18th International Conference on the Foundations of Digital Games* (Lisbon, Portugal) *(FDG '23)*. Association for Computing Machinery, New York, NY, USA, Article 68, 10 pages. https://doi.org/10.1145/3582437.3587209

[2] Michael Beukman, Branden Ingram, Ireton Liu, and Benjamin Rosman. 2023. Hierarchical WaveFunction Collapse. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 19, 1 (Oct. 2023), 23–33. https://doi.org/10.1609/aiide.v19i1.27498

[3] Maxim Gumin. 2016. *Wave Function Collapse Algorithm.* https://github.com/mxgmn/WaveFunctionCollapse

[4] Cheng-Zhi Anna Huang, Curtis Hawthorne, Adam Roberts, Monica Dinculescu, James Wexler, Leon Hong, and Jacob Howcroft. 2019. The Bach Doodle: Approachable music composition with machine learning at scale. arXiv:1907.06637 [cs.SD]

[5] Isaac Karth and Adam M. Smith. 2022. WaveFunctionCollapse: Content Generation via Constraint Solving and Machine Learning. *IEEE Transactions on Games* 14, 3 (2022), 364–376. https://doi.org/10.1109/TG.2021.3076368

[6] Ryan Louie, Andy Coenen, Cheng Zhi Huang, Michael Terry, and Carrie J. Cai. 2020. Novice-AI Music Co-Creation via AI-Steering Tools for Deep Generative Models. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (, Honolulu, HI, USA,) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376739

[7] Favour Olaoye and Kaledio Potter. 2024. Ethical Considerations in Artificial Intelligence. *Machine Learning* (03 2024).

[8] Dave Smey. 2004. Important Rules for 4-Part Progressions. https://davesmey.com/theory/partwritingrules.pdf

[9] Patrik Pál Varga. 2023. ProceduraLiszt. https://bit.ly/proceduraliszt_app

[10] Pál Patrik Varga and Rafael Bidarra. 2023. Procedural mixed-initiative music composition with hierarchical Wave Function Collapse. (2023).

[11] Patrik Pal Varga and Rafa Bidarra. 2024. Harmony in Hierarchy: Mixed-Initiatie Music Composition Inspired by WFC.