Harmony in Hierarchy: Mixed-Initiative Music Composition Inspired by WFC

Pál Patrik Varga[0009−0006−1794−2880] and Rafael Bidarra[0000−0003−4281−6019]

Delft University of Technology, Delft, The Netherlands
R.Bidarra@tudelft.nl

Abstract. Music has an important, but often subsidiary, role in most games and interactive experiences. Efforts to make generative music more accessible can potentially help numerous indie developers and designers. In this paper we explore the extension and application of the Wave Function Collapse (WFC) algorithm in the realm of music. Recognizing music's inherent complex, layered structure, we investigate the feasibility of adapting WFC, a method broadly used in procedural content generation, to handle musical semantics in a mixed-initiative setting. Our approach introduces a novel generative method utilizing multiple canvases and constraints, each defined with specific musical significance. At each level, new constraints are specified that influence the generator at the next levels. This approach is particularly fitting to capture the layered structure and complex patterns inherent to music, and it encourages a mixed-initiative, iterative composition process. We also provide a prototype implementation of the proposed algorithm in ProceduraLiszt, a declarative music editor designed to offer composers a platform for exploring highly-controllable generative composition. The effectiveness and creativity-support facilitated by this experimental setup are under evaluation and will be detailed in a future publication.

Keywords: procedural content generation, mixed-initiative authoring, wave function collapse, hierarchical wave function collapse, music composition

1 Introduction

Music is an essential element of many games and interactive experiences. However, it is often disregarded in such projects, due to expertise, budget or planning constraints. Therefore, for many indie projects, it makes sense to resort to a generative music approach, especially if it is accessible, responsive and iterative. Modelling music structurally and in terms of constraints for procedural generation is an idea that has been around since the early days of computers [3,12]. Many different models have been tried and evaluated, with various goals and degrees of flexibility [5].

Recent advancements in generative systems have demonstrated their significant potential in enhancing creative processes [13]. Among the promising algorithms in this domain is Wave Function Collapse (WFC) [4], renowned for

its simplicity and effectiveness in procedural content generation (PCG). Despite these advancements, application of WFC in the realm of music composition remains largely unexplored. This is a missed opportunity, because many strengths of WFC seem to directly encourage this application, including (i) its proven results in a mixed-initiative context, (ii) the simplicity of the algorithm, (iii) the direct interaction on its input analyzer to derive constraints, and (iv) the possibility of backtracking and giving explanations. This paper bridges this gap by exploring these features of WFC to tackle the challenges of modelling music, and presenting a novel approach to procedural music generation.

Our generative model builds upon an extension of WFC, and its music modeling capabilities feature a layered structure of multiple canvases and constraints, each imbued with distinct musical significance. This representation was designed to capture the complex layers and patterns inherent in music composition in a way that fits both a composer's mindset and our generative algorithm.

We implemented this generative approach in `ProceduraLiszt` [16], a declarative music editor that offers composers highly-controllable generative features. You can find more information about it in Appendix B.

2 Related work

Wave Function Collapse is a PCG algorithm inspired by a concept from quantum mechanics, where a wave function – representing a superposition of multiple different states – collapses into a definite state [4]. In the context of PCG, WFC uses this analogy to generate structured and diverse content. The algorithm operates by considering a superposition of possible states (or values) for each cell in a grid (which can represent e.g. pixels, tiles, or voxels). Iteratively, a cell gets collapsed to a single state, leading to a propagation of cell state changes based on rules derived from an input sample. Ultimately, if no rule conflict takes place, the algorithm outputs coherent and visually appealing content, that somehow resembles the input sample.

The original WFC algorithm, as conceptualized by Maxim Gumin [4], has inspired considerable PCG research. Karth and Smith [7] have examined the algorithm and the power that lies within it, supported by multiple in-the-wild use cases of WFC. Kim et al. [8] have demonstrated WFC's capabilities of solving constraint problems in non-grid-shaped settings.

Much of the work on WFC has addressed several of its original limitations, and extended it in a variety of directions, particularly regarding its accessibility to non-technical users. Recent advancements have focused on introducing mixed-initiative elements into WFC [9]. These enhancements allow for better user interaction and more control, enabling users to directly influence the generation process. A mixed-initiative approach not only makes WFC more intuitive for users but also expands its utility in creative domains such as game level design and graphic arts. By allowing for interactive steering of the algorithm, manual editing, and manipulation of generation parameters, these developments have

opened new avenues for creative expression and design flexibility in procedural content generation.

As detailed by Karth and Smith [6], WFC not only exemplifies the successful application of complex concepts like constraint solving and machine learning in PCG, but also highlights the significant potential of such algorithms beyond traditional academic research. This is evidenced by its diverse adaptations and extensions in various fields, ranging from game development to creative design, demonstrating a versatile and practical approach to algorithmic problem-solving and content creation.

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.

This paper is a successor of a short paper from 2023, written about the basic concept of using HWFC for procedural music generation, by the same authors [15].

3 A generative music model for WFC

Considering that WFC basically performs a constrained tiling on a canvas, one has to answer two questions: (i) which tileset is most appropriate for yielding a musical output, and (ii) what should the slots on our musical canvas stand for?

For this, it is tempting to think of the so-called *piano roll layout* as a natural layout. It visualizes music on a discrete, 2D grid, with the horizontal axis representing time and the vertical axis representing pitch. Although this resembles Maxim Gumin's original 2D canvas [4], there are multiple issues with this representation that make it unsuitable for WFC music generation purposes:

- WFC is explicitly about enforcing constraints in the vicinity of a grid slot. However, in music, notes that are strongly related are often not next to (i.e. not a half-step away from) each other. For example, thirds, fifths and octaves are all important concepts that our model should be able to work with easily.
- Chords and their duration become especially tricky. How can you specify how many slots an underlying chord should last, if WFC only works with local neighborhoods?
- For similar reasons, how can you guarantee that what is happening 'on top' (in the *melody*) is related to what is happening 'below' (in the *chords*)?
- Repeating musical characteristics (e.g. recurring themes or patterns) are essentially impossible to enforce.
- Similarly, enforcing a global characteristic throughout a piece of music (e.g. the key or the time signature) can be very cumbersome.

In conclusion, simply mixing the temporal and the pitch axes on the same 2D canvas is clearly impractical for music generation with conventional WFC. We posit that to overcome the drawbacks identified above, we need a model that properly captures the following music characteristics:

1. temporal character: notes are sequenced (yielding a melody), and every note has its own duration, possibly defining a rhythm for the sequence;
2. hierarchical structure: most music compositions clearly consist of distinct parts, each of them has its own meaning, and also a specific structure (e.g. strophe, chorus, etc.), usually with its own features;
3. pattern definitions: patterns (e.g. a theme, like the starting four notes of Beethoven's 5^{th} symphony) are what makes each music recognizable and distinguishable;
4. repetition of patterns: often, (a variation on) the same pattern occurs several times throughout a musical composition (e.g. a chorus inserted between strophes).

In this section we introduce a music model that contains all these features, and is therefore particularly suited for the intended extension of WFC to music generation.

3.1 Hierarchy for structure

From symphonies to movements, from phrases to measures, music is built from hierarchically layered structural elements. An inherent challenge of modeling music for procedural generation is defining these building blocks: we want a melody to be related to its underlying chord, and to the section of the piece it is in, and we want to be able to make these relationships explicit.

Conventional WFC cannot deal with hierarchical structures at all. To tackle them, we introduce a hierarchic model of inter-related, layered canvases. Currently, this hierarchy spans the following three layers:

- Sections: at the top level, the overarching structure of the piece is laid out. This is where, for example, an intro section is placed before the first verse, or the bridge is followed by one last chorus.
- Chords: for each section, a canvas of chords is available, each of its slots representing one chord. For the sake of simplicity, we can think of the duration of one chord as one measure in the piece.
- Melody: at the lowest level, for each measure, a canvas of notes is available, which will give us a short segment of the melody.

This hierarchical structure is depicted in Figure 1 for an example composition. Each section slot corresponds to a chord canvas underneath it, and each chord slot to a melody canvas (though not all are displayed here, for clarity).

At the melody level, music is represented by a 1D melody canvas depicting time, where the values taken at each slot represent the pitch(es) played at that given time. This model solves many of the issues raised by the 2D canvas: relationships that were difficult to model, e.g. large leaps in the melody, become

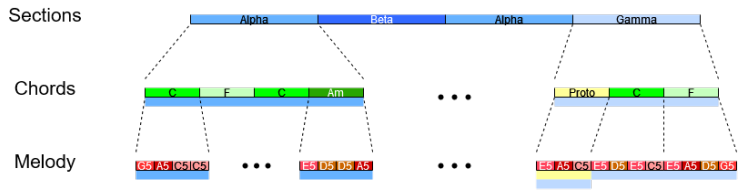

Fig. 1. Diagram of a composition example using our multi-canvas model. Each section and chord slot has a canvas in the layer underneath it, though not all are displayed here due to space constraints. The color directly underneath a given canvas represent the constraints it inherits from the parent canvas, above it in the hierarchy. This output could have originated from the following specification: the default number of chords per section, and of notes per chord, is 4; section *Alpha* requires that the first and last notes in a measure are the fifth and the root of the chord in the given measure, respectively. On the right hand, a simple example of the use of a prototype. Section *Gamma* overrides the global specification, by requiring only 3 measures on its chord canvas. In turn, the chord prototype *Proto* also overrides the global measure size to fit only 3 notes on its melody canvas. This measure is therefore inheriting constraints from both the section and the chord prototypes above it.

simple neighbor-constraints, and enforcing a key becomes restricting the array of (pitch) options for a (sequence of) slot(s).

One layer up, the canvas for chords is (similarly) a 1D canvas, but the possible values now stand for every chord in the chromatic scale. Again, at this level, a key can be enforced by constraining the chords a slot can collapse to.

The Chord and Melody layers directly relate to the actual output of the generator: their canvases feature a sequence of notes (each with pitch, time and duration) that are playable. The Section layer also features a simple 1D canvas, indicating the sequence of distinct parts in the composition. However, each slot in that canvas is slightly less tangible: it does not directly translate into notes, but stands for more abstract music features like the key, the time signature, the rhythm and other similar properties. Semantically, a section can stand for a recurring piece of melody, or a unique chord progression, or any other part of the composition piece with a meaningful role (e.g. an Intro or a Chorus). In other words, when a composer creates a given section, they want the canvases 'underneath' it to have specific requirements *imposed onto them*. The idea that music can be segmented based on such musical properties is well-established [11,18].

This representation – despite its simplicity – is suitable to model many music pieces with a clear chord progression and a prominent main melody. However, the above hierarchy alone does not solve other issues regarding e.g. time signatures, and especially underlying chords and recurring themes. Solving these requires other model elements, as presented in the next subsection.

3.2 Prototypes

As presented above, a section essentially bundles a number of properties and constraints and imposes them on the underlying (chord and melody) canvases. As such, it is a particular example of what we can call more generally a prototype, defined as *a template which (i) acts as a meta-tile that can be applied on a slot of a canvas, and (ii) bundles a set of parameter values and constraints that operate both on that slot and on its underlying canvas.*

The convenience of this notion is that it combines the advantages of being a tile (i.e. it can be used to collapse a slot into) with the fact that it propagates a predefined set of constraints down the hierarchy. So, referring back to Figure 1, section prototype Alpha could e.g. specify that it should be in the key of *C*, consist of 4 chords (i.e. spanning 4 measures in the respective chord canvas), and have the last of these chords be an *A minor*. These constraints are then imposed onto the chord canvas, before WFC goes on to collapse its slots. Notice that this prototype is being used twice in the section canvas, exemplifying the case for reusing its bundled constraints and exploring thematic coherence.

In an analogous way, and perfectly in agreement with the prototype definition above, we can also define a chord prototype one level down: it bundles a group of parameters and constraints that determine how the melody canvas underlying it should be handled. As a meta-tile, a chord prototype assigned to a slot behaves as intended: WFC will end up collapsing it into one concrete chord, subject to the constraints defined at that level for generating the chord progression. However, once that has taken place and it is time for the prototype to initialize the underlying canvas for the melody, this layer will inherit those constraints unique to the chord prototype.

Among other advantages, prototypes allow a composer to (i) convey musical meaning that is not easily represented as a concrete pattern of notes in the composition, but rather at a higher level, and (ii) preserve, tweak and replicate instances of these local pieces with the same basic semantics. Both features are extremely favorable in music composition.

Some constraints can be made to depend on parameters defined by values on layers above. As a simple example, imagine we want the piece of melody in a given measure to start on the root of the chord on that measure. But at the time of defining the constraints, the value of the chord (and so its root) is not known yet. We can simply define a 'grabber' (in the implementation, this takes the form of a callback function), which can grab the root of the chord once it is already there, and provide its value to the constraint that can then use it.

Finally, although a composer might have set some global composition constraints (e.g. 'Melody starts on the root of the chord', applicable by default to every melody canvas), a prototype can still be able to override and replace them with its own constraints. Global constraints for this include the key, the tempo, the time signature, the number of measures per section, and the rhythmic characteristics: for each of them, a composer can define its default value and, when using a prototype, choose whether to inherit that value or to override it.

The interaction between composer and computer and the main steps of the composition and generation process are described in detail in Appendix A.

4 Discussion

The music model presented in this paper and implemented in `ProceduraLiszt` has shown to support the mixed-initiative generation of a wide variety of compositions. This conclusion was also confirmed in a variety of informal sessions where we let people with knowledge of music make their own creations. We can therefore conclude that the proposed multi-canvas layered model for music is successful towards the stated goal of extending WFC for music generation. More importantly, each layer in this model has clear semantics that appeal to composers, and uses a meaningful vocabulary for them to express their creative intent. In this sense, there is little of the usual *black box* inconvenience shown by many other generative systems.

At each layer, this model is able to support the WFC algorithm with its essential elements: a tileset, a canvas, and a constraint set. Moreover, the vertical propagation of constraints between layers effectively leads to translate high-level design intent down to the melody level in a structured manner. In addition, the introduction of section and chord prototypes is a powerful mechanism to explore the controlled generation and repetition of patterns. The fact that the canvases are 1D and, in general, the tilesets strongly restricted due to musical constraints, contributes to a very efficient generation.

In the spirit of the original WFC algorithm, one might argue that it would be very convenient to be able to extract a set of constraints from an input piece of music, thus actually learning them from an example. However, it is unclear even what that set could consist of, let alone how to distinguish its essential features from other accidental occurrences, which could lead the learned model to overfit the data. This is a formidable endeavor, deserving a whole new project, its biggest challenge being to capture the appropriate constraints at the right level.

We are currently exploring how to implement backtracking over the various canvases. Without it, it may occasionally happen that the WFC algorithm runs into a constraint conflict, typically at the Melody canvas. In principle, it should be possible to undo both slot collapses and constraint propagation for each iteration, but it is still unclear what their computational cost can be. In any case, it would be very desirable to at least provide proper feedback on the causes of a conflict, whenever it takes place.

Finally, inspired by editing features of proven mixed-initiative WFC implementations [9,1], it would be very convenient to implement a mechanism of partial locking of a canvas on demand, so that composers can preserve satisfactory parts of the output, and focus on refining and regenerating the remainder of it.

5 Conclusions

We presented a novel generative approach to assist music composition. It combines a layered model of music, very suitable for capturing a composer's intent, with a multi-canvas extension to WFC. Each layer has a clear and meaningful musical connotation, and the WFC algorithm is tasked with solving its respective canvas, subject to both the local constraints and the constraints inherited from upper layers. This representation is able to conveniently capture the complex structure and patterns inherent to music.

We implemented this generative composition method in `ProceduraLiszt`, our prototype mixed-initiative music editor. This online system offers composers a highly-controllable tool to declaratively create their own music through the incremental expression of their artistic ideas.

We are continuing our research on how WFC can be further expanded and deployed to enhance music composition, both with new layers and parameters in the music model, and with useful editing features in our `ProceduraLiszt` editor. Among many other ideas, we are investigating a more generic and powerful model for rhythm, as well as how to optimally profit from a smarter WFC backtracking mechanism. A library of prototypes, as well as of configuration parameters, would also be a very convenient productivity feature. Finally, `ProceduraLiszt`'s current GUI is designed for composers with some understanding of music theory. We would like to investigate what would it take to develop a GUI for non-savvy music fans [14].

In recent years, generative AI systems have been presented that yield amazing, although often not surprisingly creative, high-quality content. So far, music generation has been lagging in this context. We believe the most powerful creativity can be unlocked by the synergy between a human artist and an intelligent generative assistant [10]. We therefore expect that increasingly more research efforts will be focusing on exploring the challenges of this creative collaboration. Our work, proposing a structured music model usable in a mixed-initiative WFC setting, is a step in that promising direction.

References

1. Alaka, S., Bidarra, R.: Hierarchical semantic wave function collapse. In: Proceedings of the 18th International Conference on the Foundations of Digital Games. FDG '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3582437.3587209
2. Beukman, M., Ingram, B., Liu, I., Rosman, B.: Hierarchical wavefunction collapse. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment **19**(1), 23–33 (Oct 2023). https://doi.org/10.1609/aiide.v19i1.27498
3. Brooks, F.P., Hopkins, A.L., Neumann, P.G., Wright, W.V.: An experiment in musical composition. IRE Transactions on Electronic Computers **EC-6**(3), 175–182 (1957). https://doi.org/10.1109/TEC.1957.5222016
4. Gumin, M.: Wave Function Collapse Algorithm (9 2016), https://github.com/mxgmn/WaveFunctionCollapse

5. Herremans, D., Chuan, C.H., Chew, E.: A functional taxonomy of music generation systems. ACM Comput. Surv. **50**(5) (sep 2017). https://doi.org/10.1145/3108242
6. Karth, I., Smith, A.: WaveFunctionCollapse: Content generation via constraint solving and machine learning. IEEE Transactions on Games **14**, 364–376 (2021). https://doi.org/10.1109/TG.2021.3076368
7. Karth, I., Smith, A.M.: WaveFunctionCollapse is constraint solving in the wild. In: Proceedings of the 12th International Conference on the Foundations of Digital Games. FDG '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3102071.3110566
8. Kim, H., Lee, S., Lee, H., Hahn, T., Kang, S.: Automatic generation of game content using a graph-based wave function collapse algorithm. In: 2019 IEEE Conference on Games. pp. 1–4. IEEE, London, UK (2019)
9. Langendam, T.S., Bidarra, R.: miWFC - designer empowerment through mixed-initiative wave function collapse. In: Proceedings of the 17th International Conference on the Foundations of Digital Games. FDG '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3555858.3563266
10. Liapis, A., Yannakakis, G.N., Nelson, M.J., Preuss, M., Bidarra, R.: Orchestrating game generation. IEEE Transactions on Games **11**(1), 48–68 (2019), https://doi.org/10.1109/TG.2018.2870876
11. McCallum, M.C.: Unsupervised learning of deep features for music segmentation. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 346–350 (2019). https://doi.org/10.1109/ICASSP.2019.8683407
12. Moorer, J.A.: Music and computer composition. Commun. ACM **15**(2), 104–113 (feb 1972). https://doi.org/10.1145/361254.361265
13. van den Oever, M., Saunders, R., Jordanous, A.: Co-creativity between music producers and smart versus naive generative systems in a melody composition task. In: 14th International Conference on Computational Creativity. Associations for Computational Creativity, Waterloo, Canada (June 2023), https://kar.kent.ac.uk/101968/
14. van Dongen, J.: Robo maestro (1 2023), https://store.steampowered.com/app/\1808490/Robo_Maestro/
15. Varga, P.P., Bidarra, R.: Procedural mixed-initiative music composition with hierarchical wave function collapse. In: Proceedings of PCG 2023 - Workshop on Procedural Content Generation for Games. pp. 1–3 (Apr 2023), http://graphics.tudelft.nl/Publications-new/2023/VB23
16. Varga, P.P., Bidarra, R.: ProceduraLiszt (2024), https://proceduralisztdevs.github.io/proceduraliszt/#/proceduraliszt/
17. Varga, P.P., Bidarra, R.: ProceduraLiszt results (2024), https://proceduralisztdevs.github.io/proceduraliszt/#/proceduraliszt/results/
18. Vatolkin, I., Koch, M., Müller, M.: A multi-objective evolutionary approach to identify relevant audio features for music segmentation. In: Romero, J., Martins, T., Rodríguez-Fernández, N. (eds.) Artificial Intelligence in Music, Sound, Art and Design. pp. 327–343. Springer International Publishing, Cham (2021)

A Algorithm description

The composition process Most previously mentioned WFC approaches work with constraint sets which are set once in the beginning of the process, the most common approach is extracting the allowed neighbor-relations from a fully collapsed example canvas, commonly known as Constraint Discovery. However, with our multi-layered model, feature extraction from music is non-trivial and is a topic for future research.

It is also clear that it is precisely the constraints that define the unique quality and character that the generated music will hold. Therefore, in our approach, the main mechanic of composing a piece of music is to cleverly define the constraints by which each canvas operates, and structuring the prototypes that encapsulate these constraints on various levels. Comparably to actual music composition, this generative composition process becomes an iterative loop of defining constraints, listening to the output, and then tweaking those constraints again, as illustrated in Figure 2.

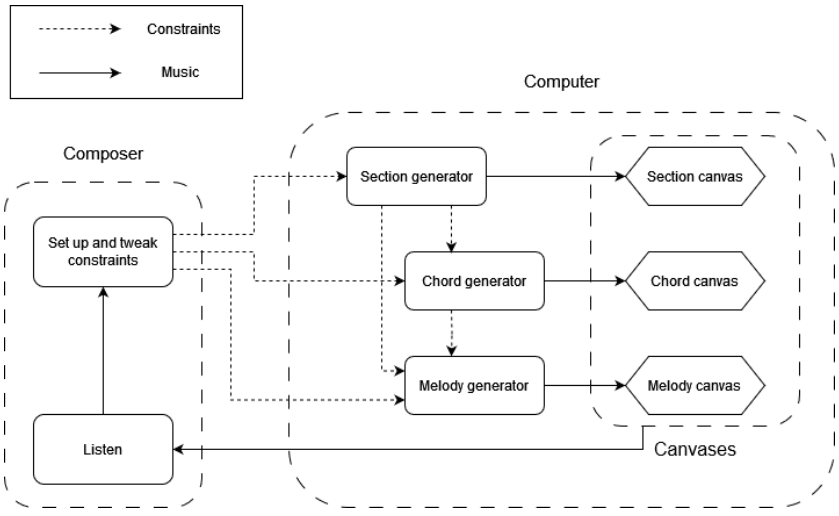

Fig. 2. Mixed-initiative cycle of the composition process and the generative stage.

The generative stage Upon receiving the constraints and prototypes defined by the composer, our model generates an output from the top down:

1. The canvas of sections is filled according to the section constraints. Each section tile takes the value of one of the prototypes defined by the composer.
2. For each section, a canvas of chords is created. The constraints on this canvas will be the globally defined chord constraints, merged with the constraints inherited from the section that was chosen on the level above.