Cellular Symphonies

Jake Looney
Department of Electrical Engineering and Computer Sciences
University of Tennessee
Knoxville, United States
jlooney6@vols.utk.edu

***Abstract*—Cellular automata can display complex and chaotic behavior from only a small set of rules. Because of this, they are a good candidate for use in computer generated music. Cellular automata have been utilized to create music of all styles and genres in a variety of ways by a variety of authors. In this paper, I describe a method for creating orchestral scores and MIDI music utilizing cellular automata, including various design decisions, their benefits, and their drawbacks. I then describe the results of an experiment that correlates various properties of cellular automtata to how good I think music generated by them sounds, and expand upon these results to examine what makes music sound good in general. Finally, I conclude by looking at use cases and implications of cellular automata generated music, as well as other forms of computer generated music.**

***Index Terms*—cellular automata, music generation, score generation**

I. Introduction and Motivation

Cellular automata consist of a collection of cells of various states and a rule table dictating a cell's next state based off off their current state. These rule tables are usually very simple in nature, often only taking in information about a given cell's neighbors to determine its next state. However, despite the simple rules governing cellular automata, they can exhibit complex and chaotic behavior. Perhaps the most famous example of a cellular automata is Conway's Game of Life, a two dimensional automata that only has three rules. From these three rules, Conway's Game of Life is capable of general computation.

Because of the difference between the simple rules of a cellular automata and their complicated behaviors, they can often be useful in a variety of problems. One example of this is using them within creative and artistic projects. This is a natural use of cellular automata as they are similar to how humans attack creative projects – utilizing a simple set of rules and methods to build up and create something creatively complex.

In particular, it is easy to use automata to create music. Music can broadly be broken up into four attributes that we can modify: pitch, rhythm, harmony, and dynamics. By assigning an individual cell to control or modify one of these elements, with the cell's states determining exactly how an attribute is used or changed, we can create music. By repeating this for many instruments, we can create an entire orchestra of music using only cellular automata.

An individual cellular automata also has various ways of being described and categorized. One of these are Wolfram Classes, which generally determine the expected long term behavior of a cellular automata. Another example is values called Langton's Lambda and entropy, which characterize how energetic or disorderly a given automaton's rule table is. By generating musical outputs for a cellular automata as described in this paper, its possible to compare these classes and values with how an automata sounds.

Finally, by creating musical representations from cellular automata, it will be easier to look both at the potential use of cellular based music as well as to examine the implications of machine learning and biologically inspired music generation.

II. Related Works

As music is one of the most innate parts of the human experience, there is unsurprisingly no shortage in the literature of biologically-inspired musical creation and composition methods. Basically all biologically inspired computing methods have been applied to music in one way or another.

As discussed in class, Melomics is an example of a Lindemayer-System (L-Systems) that produces musical output [1] [2]. Melomics' music has even been demonstrated to be on par with conventional, human-composed music in the relaxing effect when used in music therapy [3]. Other older examples of using L-systems to create music also exist – as far back as 1986 L-systems were used for musical score generation [4].

Neural systems have also seen use in creating music. Liang and Ziang have used spiking neural networks to compose music based off of a given composer or genre, to favorable results when surveying both professional and non-professional subjects [5]. Recurrent neural networks have also been used to generate musical scores by Stanford students [6].

Finally, most relevant for us, cellular automata have no shortage of musical implementations. This can be seen with Greek composer Iannis Xenakis, who utilized cellular automata to help compose his 1986 piece *Horos*, which sounds like all of my greatest fears combined [7] [8]. A good sense of the breadth of use of automata can be seen in two surveys by Dave Burraston, Ernest Edmonds, and others on cerating MIDI from CA's. The first from 2004 gives a synopsis on some examples of cellular automata based music and goes into detail on different parameters – such as number cells, number of states, rule tables, and number of cellular automata used – for a brief number of implementations [9]. The second from 2005 gives a wider breadth of examples, as well as detailing instances of sonic-visual art utilizing cellular automata [10].

Two-dimensional automata such as Conway's Game of Life have also been utilized to create music, both in an interactive way [11] as well as more deterministic ways [12]. Finally, there is a web project called WolframTones that generates music in a very similar way to what I will lay out in this paper [13].

All of this is to say that the use of bio-inspired algorithms or cellular automata to create music is not a new concept. However, the exact method I take in this paper to create music may likely be unique. Further, from my cursory glimpse through the literature, I could not find an example of people performing table walk downs and determining musical fitness for each output as I do.

III. Methodology

A. Overview of Design

One-dimensional cellular automata can map very easily to musical output by dividing cells into musical instruments with cell states deciding some attribute of the music the instrument is playing. This can be done in a multitude of ways, depending on instruments selected, musical attributes used, and how cells are spatially arranged. For example, say a cell has a state value of 5. This could correspond to the clarinet part playing the 5^{th} scale degree of a key or could correspond to a trumpet playing at the 5^{th} loudest volume possible – all depending on the rules one uses to map from the automata. The MIDI 1.0 specification [14] defines 128 control channels (many of which do a lot more than others), and with time and pitch parameters, there are a plethora of ways to determine what a given cell does. For our purposes, we'll stick with only using pitch, velocity (loudness), and rhythm for what cell states can alter. This means each instrument in our ensemble will have three cells that control it.

For instrumentation, I elected to use 18 instrument parts (two violins, viola, cello, bass, flute, clarinet, oboe, alto saxophone, tenor saxophone, baritone saxophone, french horns, two trumpets, trombone, tuba, marimba, and piano) which would create a fairly reasonable orchestral instrumentation. This could obviously be altered, with General MIDI offering up to 128 performance patches, and other sound libraries offering far, far more.

Another important factor to consider is where instrument's and their cells are laid out spatially. I chose to group by attribute rather than group by instrument. That is, a certain attribute, say pitch, would have all instruments' pitch cells neighboring, and then followed by the next attribute. This in contrast where a certain instrument, say violin, would have all its attributes next to each other. Both methods of grouping – as well as other methods – make sense but will alter how music created sounds.

Finally, the last factor to consider is harmony. Harmony is important to decide what key the ensemble is playing in, which will determine the exact notes struck when coupled with the state of the pitch cells. Harmony is perhaps the most difficult to implement in way that sounds familiar to humans as harmony is heavily driven by context and cultural biases.

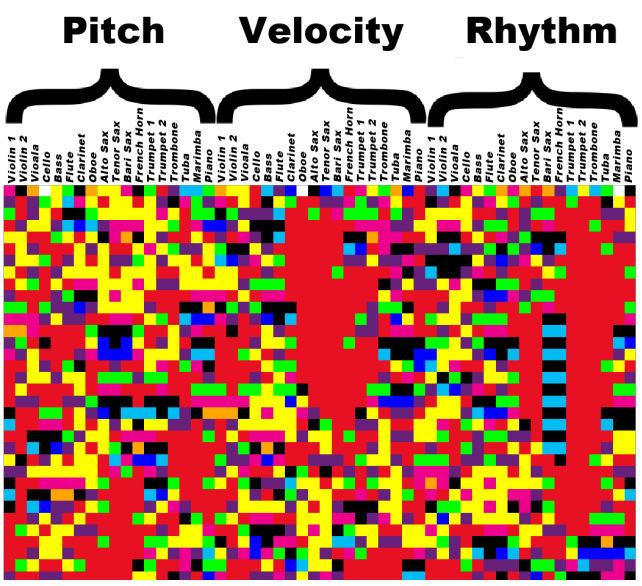

Fig. 1: A visual description of how a given automaton will map to musical output. Individual cells correspond to an instrument and a musical attribute with cell states (colors) corresponding to a value of those attributes.

While there are definitely ways I attempted to implement harmony within a cellular automaton – ranging from insanely alien and weird sounding to dull and boring – I ultimately elected to remove harmony from the automata and let it be decided problematically. This will be expanded upon later.

So, in all, our rules for creating musical output are best summed up in Figure 1. It is important to remember that the decisions taken to create these rules are somewhat arbitrary, and similarly musical results could arise from choosing a different mapping.

B. Building the Pipeline

The pipeline for our experiment will require a way to create multiple automata with the same starting seed and varying rule tables, and a way to take produce MIDI output for each automata.

I was able to create cellular automata with a few modifications to the code from Lab 1 [15]. Automata size was limited to 54 states (18 instruments with 3 states each), number of states was increased to 10 states, and the number of time steps was limited to 512. The code was also modified to only produce one experiment at a time, which was required due to limitations in the MIDI library.

So, given an experiment, the program will perform a table walk down for a certain seed. It will create a rule table with the starting seed and then select a rule to zero for a given step. For each step, it will output a visual representation of the automata (similar to Lab 1) as well as store the board of the automata in order to later create MIDI output. Finally, it will create a csv file containing each steps values for λ, λ_T, H, and H_T.

In order to be able to write MIDI files, I searched for a MIDI python library and selected one called MIDIUtil [16]. This would be a mistake I would come to regret. MIDIUtil is a library that allows for a more natural way of writing to MIDI files than learning the MIDI specification, but has been abandoned since early 2018. As such, the library is full of countless and strange bugs that don't necessarily replicate themselves. Most noticeable of the bugs is that MIDIUtil would often just forget to end a note. The MIDI specification groups events into note on and note off events. MIDIUtil implements this by providing a single function – `MIDIFile.addNote()` – which takes a pitch, duration, and time. This function will add a MIDI note on event at the given time and create the note off event the given time plus duration. However, for reasons I still can not discern after rummaging through the code, MIDIUtil will just sometimes forget to add the note off. The result of this is an instrument continuing to play a note far longer than it should, sometimes up to several bars longer. This bug is also very inconsistent – most times, note off events are placed where they need to be, but every so often, an entire MIDI file will be missing several note off events. This became incredibly frustrating, and if given the chance, I would go back and create my own MIDI library.

After the automata ran for all steps in the rule table walk down, the resulting boards would be used to create MIDI output. To do this, four sequential values of a given instruments cells, as well as current chord and current time, would be passed to a function `WriteNote()`. Then, depending on the value in the rhythm cell, a subset of the passed values would be used to create the notes. For example, if the value in the rhythm cell corresponded to a triplet, then only the first three values of pitch and velocity passed would be used to create notes. If, however, the rhythm cell corresponded to sixteenth notes, then all four values of pitch and velocity passed would be used. A rhythm value corresponding to a rest would make the function return without writing anything. Checks were given such that if an instrument is currently playing a note (i.e. a half note or whole note), then it will note strike any other notes until finished.

The pitch of the notes struck would be determined based on the current key, the value of pitch cells, as well as a random number based on the experiments starting seed. This number would allow for instruments to play mostly within a given chord's chord tones (e.g for C major, most instruments would select pitches of either C, E, or G) with a small chance to play an tone from a scale rather than the chord (e.g. playing within the C major scale). This allows for a more diverse selection of pitches, and still remains deterministic as the probability is based off of the experiments starting seed. However, the consequences of this decision need to be remembered when analyzing the music generated.

Finally, the velocity of a given note is decided linearly by the velocity cell's value. A minimum velocity of 20 (corresponding to a cell state of 0) and a maximum velocity of 120 (corresponding to a cell state of 10) were chosen. The scaling of velocity values as well as the maximum and minimum velocities could easily be changed.

`Writenote()` is called once per beat(each time passing four values for each attribute) for 64 beats, resulting in 16 bars of music outputted. This is done for each instrument, with each instrument writing notes a different track of the MIDI output file. Finally, this is repeated for each step of the experiment, with a separate MIDI file created for each step.

As mentioned earlier, harmony was not determined by an automata and was instead decided probabilistically. This was done by using a random number generator seeded on the automata's starting seed. Then, for a given current chord, the output will change to a different chord depending on a random number. As a random number generator will always output the same numbers in the same order for a given seed, this still allows our musical output to be deterministic. The chord changes as well as their probabilities were determined by sitting down at a piano and thinking "this chord could move here about this likely". There are two reasons for using a random number generator to determine harmony instead of the automata. The first is that it was far easier to map my sense of "typical" harmony to a uniform probability distribution than the normal distribution that would arise from using the states of an automaton. The second is that I had tried to implement harmony into the automata, couldn't get it to work, was running out of time to start and finish the experiment, and so scrapped it. Different methods of implementing harmony are a topic of possible future work for this project.

The code utilized in this project can be viewed at https://github.com/6a6c/cellularSymphonies.

C. The Experiment

In order to find patterns in the musical output of automata, 30 experiments as described above were computed. For each experiment, I would open MIDI output in MuseScore 4 [17], allowing me to listen to it using synthesized instruments. After listening to a musical output, I would then assign it a "fitness" value within $[0, 1]$. Then, after listening to and assigning fitness for all experiments, I would visually inspect the automata to determine Wolfram Class. This would be repeated for every experiment.

A given output's "fitness" was determined, in general, based on how much I liked listening to it. Some attempts to be objective were taken, but the ultimate fitness value was still subjective. I would attempt to try to "give points" to an output if it had sections within it that sounded good (or even listenable), even if the things surrounding a section of music did not sound good. I would also rank automata that did not play much music very low, which would have an effect most examples of Wolfram Class I behavior as we will see. Finally, as tonality and harmony were not determined by the automata and were instead probabilistic, I tried as best I could to not take them into account when assigning fitness. This still left a lot of room for determining how an automata sounded, as variations in voice leading, dynamics, and rhythm constitute a large part of what we see as "good" music. However, the

effect of harmony was likely not fully ignored, and should be considered when analyzing results.

Musical outputs were also varied in their tempo to allow for more diversity, with ten experiments at 80 beats per minute, ten a 100 beats per minute, and ten at 120 beats per minute. Following the fitness assignment and Wolfram classification of every experiments, I was left with 840 points of data with which to help characterize the listen-ability of music generated by cellular automata.

IV. Results

A. Qualitative Results

Above all, the music generated by our cellular automata can best be described as "not as bad as you think it would be".

The music is often chaotic and just somewhat off sounding. Especially for Class III and Class IV behavior, the rapid and noisy changes in a cells states can cause the music to sound like everyone in the ensemble is just banging on their instruments. However, there are many instances where the stars align and everything comes together for a few bars. The result creates a somewhat uncanny feeling in the listener, almost is if a human sat down and wrote two bars of music before handing it back off to random noise. And even still, there were some instances where the music would stick together and sound "normal" for far longer.

If you yourself would like to listen to some selected pieces generated by cellular automata, you can visit https://soundcloud.com/cellularsymphonies

B. Effects of Lambda and Entropy on Fitness

For each rule table in the experiments, values of Langton's Lambda as well as table entropy were computed in both a totalistic and non-totalsitc way. These values are given by the following equations:

$$\lambda = \frac{T - n_q}{T} \tag{1}$$

$$\lambda_T = 1 - \frac{m_0}{S} \tag{2}$$

$$H = -\sum_s \frac{n_s}{T} log_2 \frac{n_s}{T} \tag{3}$$

$$H_T = -\sum_s \frac{m_s}{S} log_2 \frac{m_s}{S} \tag{4}$$

where

- n_s is the number of states that map to a state s
- T is the number of entries in the rule table
- m_s is the number of entries in a totalistic rule table that map to a state s
- S is the size of the totalistic rule table

From my experiments, there are definite peaks in the correlation between fitness and Langton's Lambda values, with less evident peaks in the correlation between fitness and entropy values. These can be seen in Figure 2a. Figure 2b suggests that λ and λ_T may follow a cubic fit while H and H_T may follow a quartic fit.

These trends in the fitness make some intuitive sense. Langton's Lambda is, in a sense, a measure of the excitability of a rule table for an automata. $\lambda = 1$ entails an automata that is highly excitable and will never move to the quiescent state (0). Meanwhile, $\lambda = 0$ entails every rule in an automata's rule table will lead to zero. And, by the way we created music for the automata, a state value of zero entails either tonic pitch, low velocity, or no rhythm. So, a high value λ automata will have few states that maps to zero, meaning every instrument is going to be playing basically all the time. Meanwhile, a low lambda automata will have almost all states map to zero, and so will be playing basically none of the time.

Similarly, H and H_T measure the entropy, or disorder, of a given automata. A more entropic automaton is likely to to map to higher value cell states, which include higher velocity and faster rhythms. Meanwhile, a low entropy automaton will map more states to either zero or lower cell state values, which correspond to quieter and less rhythmically intense music.

Together, these two hint at the idea that music sounds the best to humans in some balance between boring and all over the place. Music with higher values of λ, λ_T, H, H_T is more likely to be too much to handle and will sound like unconstrained noise. Meanwhile, music with lower values will be boring. Thus, an optimum between the two makes some intuitive sense.

Personally, I was a bit surprised by how far right the peaks were for these values, especially for H and H_T. However, this can potentially be explained by the way the music is generated in the automata and how most Class II behavior occurred. Most Class II behavior presented itself in vertical oscillating patterns, as opposed to patterns that striped across the entire space. As such, if this vertical oscillating pattern does not occur at least in part of the section of the board that dictates rhythm, then all instruments will be tacit. Thus, Class II behavior, which has lower values of H and H_T, often had lower fitness than if we determined musical output in another way.

C. Effects of Wolfram Class on Fitness

The effects of Wolfram Class on fitness follow a trend that again aligns with intuitive sense. For each automata rule table, a visual representation was outputted as well as the audio representation. From this visual representation, I was able to determine what Wolfram Class [18] a given automata belongs to. Classes were defined as Class I being a uniform limit, Class II being an oscillating limit, Class III being chaotic, and Class IV being the edge of chaos. The correlations between Wolfram Class and fitness is shown in Figure 3.

There is likely some degree of unreliability in my classifications. As the board was only 54 cells across, determining differences between Class III and Class IV behavior was difficult. While I did my best to discern carefully, there were some five to ten instances where I was on the cusp between chaotic and edge of chaotic behavior.

Overall, however, these again hint at the idea from above – that music that is enjoyable to humans exists in a balance

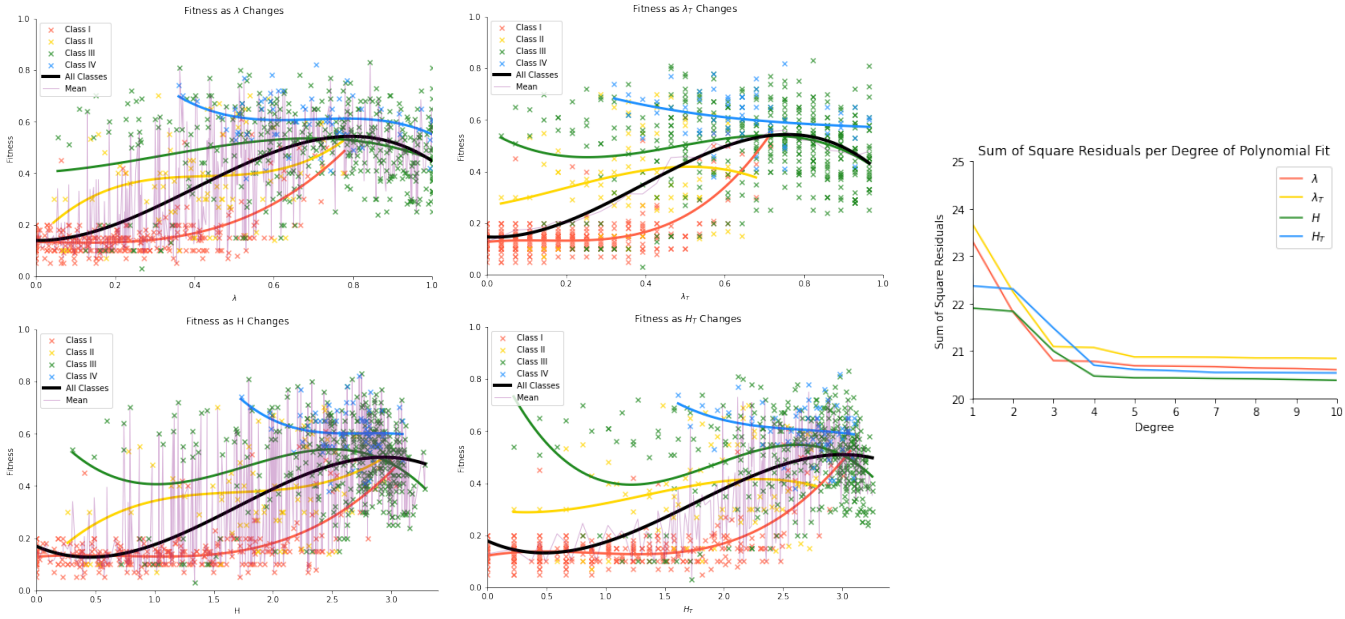

Fig. 2: On the left, we see the correlation between our values λ, λ_T, H, and H_T. Different classes are colored by different points and fits, and the mean for a given value is shown by the faint purple line. On the right, we have the sum of square residuals for each value per degree of polynomial fit.

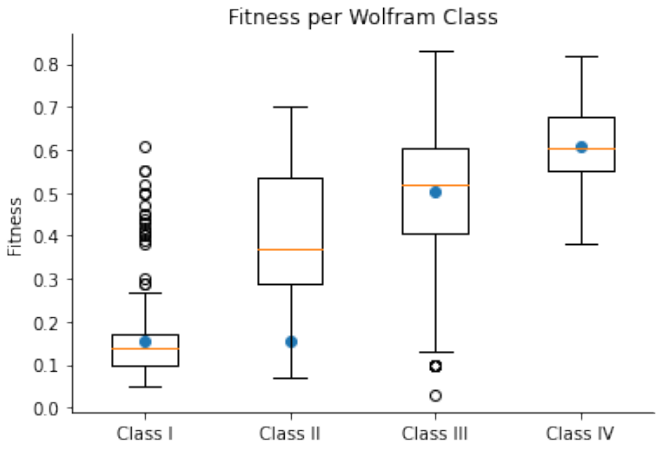

Fig. 3: A box and whisker plot showing the values of fitness for different Wolfram Classes. The mean for a given class is shown by a blue dot.

between chaos and boring. This is perhaps most evident in Class IV – literally the edge of chaos – which has the highest median and mean fitness. Class IV is significantly higher than I thought it would be. This might be a sampling issue, as far fewer examples of Class IV were observed than any other class. More experiments would be needed to more properly determine fitness for Class IV.

Class III had the highest individual fitness score, as well as having the largest range of values. This makes some degree of sense, as chaotic systems could hit all the values needed for music to sound good while still being chaotic. However, many chaotic systems will just be noise and thus will not translate into music that sounds good, leading to a wide range of possible values.

Class II is again lower than I thought it would be. This is likely for the reason above, where much of Class II behavior sounds like quiescent Class I behavior by the nature of how the musical representations are determined. Still, I was expecting more examples of simple yet entertaining outputs from Class II that I never really saw.

Finally, Class I unsurprisingly was low, as most Class I behavior played no music at all. The outliers of Class I are likely from transient behavior before ultimately going to the quiescent state.

V. Discussion and Conclusion

A. Future Work

This project is definitely far from done and could benefit from future work. One thing in particular that should be implemented is automaton controlled harmony. As discussed, harmony in the current implementation is probabilistic and only uses a random number generator seeded on the starting seed of the automaton to create harmony. This does somewhat remove the total power of cellular automata created music. By adding some method of including harmony in the cellular automaton, the number of cells would likely grow. This would likely result in an increased number of Class IV behavior that could manifest.

Another factor of this project that could use expansion is the experiment itself. As I was the only person to listen to and rank all the examples of music, the sample size for the results is quite small. As such, conclusions drawn are likely less

indicative about music in general and only about my taste in music. Further experiments with a wider sample of participants is likely to help expand the results. These experiments would bring with them their own challenges. For example, blind trials without prefacing the expectations of listeners for the music they are about to here could result in a large amount of very low scores. Similarly, getting a random participant to do as many examples as I did (which took about 6 to 7 hours) is unreasonable. Problems like these and others would need to be considered when expanding the scope of the experiment.

B. Potential Use

Personally, I only see my implementation potentially being used as a composition and production complement. After spending hours listening to music generated by this thing, I do not foresee it possibly being used exclusively to generate long-from orchestral music. The music is just too off or chaotic sounding. However, I do see the potential of this being used along side an actual human for composition or production. There were many examples I found of music that scratched a certain itch in my brain for a brief second. By surveying various outputs from automata, a musician could sample – either in score or directly – parts of the output for use in other musical works. However, I do not see many use cases beyond that.

C. What Are We Even Doing With All This Stuff Anyway?

Finally I'd like to conclude by looking at the current state of computer generated music. A few weeks ago, a fake song utilizing AI text-to-speech was created mimicking the vocal styles of Drake and The Weeknd [19]. This is, as far as I know, the first instance that saw record labels respond to AI assisted music with legal action.

While the music that AI Drake and The Weeknd were rapping over was not computer generated, computer generated music – either biologically inspired or traditional – does not seem far off from being able to fake a Drake beat. There already are examples of machine learning being used to separate music into stems and AI being able to determine melody and lyrics of a vocal performance. I've seen many computer science people on Twitter who want to combine these principles, to be able to have any song faked by any artist.

Personally, this drive by many in the field seems scary and myopic to me. Art such as music, poetry, are visual art are an innate things humans do. In our modern world, art is often an escape for people, something they have limited time to do but do because it is something they love. Yet somehow, many people, both in and out of the field of computer science, seem eager to create a future where art is something machines do.

In conclusion, I guess I wish more computer solutions to artistic problems were like this – more complementary and less replacing. I think many computer scientists, myself included, need to be more cognisant of the ethical effects and impact that research in the field can bring, for better or worse, to a variety of other fields. Because the future will likely be heavily shaped by the work and research of computer scientists in the coming decades, and if we don't reconcile with that, the results could be disastrous.

REFERENCES

[1] F. Vico, D. Albarracin-Molina, G. Diaz-Jerez, and L. Manzoni, *Automatic Music Composition with Evolutionary Algorithms: Digging into the Roots of Biological Creativity*, pp. 455–483. Cham: Springer International Publishing, 2021.

[2] C. Schuman, "Developmental systems," Apr 2023.

[3] A. Raglio, P. Baiardi, G. Vizzari, M. Imbriani, M. Castelli, S. Manzoni, F. Vico, and L. Manzoni, "Algorithmic music for therapy: Effectiveness and perspectives," *Applied Sciences*, vol. 11, no. 19, 2021.

[4] P. Prusinkiewicz, "Score generation with l-systems," in *ICMC*, pp. 455–457, Ann Arbor, MI, 1986.

[5] Q. Liang and Y. Zeng, "Stylistic composition of melodies based on a brain-inspired spiking neural network," *Frontiers in Systems Neuroscience*, vol. 15, 2021.

[6] N. Agarwala, Y. Inoue, and A. Sly.

[7] M. Solomos, "Cellular automata in Xenakis's music. Theory and Practice," in *International Symposium Iannis Xenakis (Athens, May 2005)* (M. Solomos, A. Georgaki, and G. Z. (eds.), eds.), (Greece), p. 11 p., 2005.

[8] I. X. on YouTube https://www.youtube.com/watch?v=9aYsh8SRB c, Apr 2022.

[9] D. Burraston, E. Edmonds, D. Livingston, and E. R. Miranda, "Cellular automata in midi based computer music," in *Proceedings of the 2004 International Computer Music Conference*, International Computer Music Association, 2004.

[10] D. Burraston and E. Edmonds, "Cellular automata in generative electronic music and sonic art: a historical and technical review," *Digital Creativity*, vol. 16, no. 3, pp. 165–185, 2005.

[11] K. Ogawa and Y. Kuhara, "Life game orchestra as an interactive music composition system translating cellular patterns of automata into musical scales.," in *NIME*, pp. 50–51, 2009.

[12] E. R. Miranda and A. Kirke, *Game of Life Music*, pp. 489–501. London: Springer London, 2010.

[13] W. T. https://tones.wolfram.com, 2005.

[14] T. M. Association, "Official midi specifications."

[15] C. Schuman, "Cellular automata part 2," Feb 2023.

[16] M. C. Wirt, Sep 2016.

[17] MuseScore, Dec 2022.

[18] S. Wolfram, *A new kind of science*. Wolfram Media, 2021.

[19] J. Coscarelli, "An a.i. hit of fake "drake" and "the weeknd" rattles the music world," Apr 2023.