Agent-Based Rhythm Generation

November 2023

An artistic research project by Thomas Poeser

Abstract

This study examines systems of interconnected autonomous(1) agents that can each play a sound and over time create a rhythmic pattern, following simple rules based on delayed triggering of other agents.

The parameters of each system were defined in xml format.
Software applications to model and 'play' a system, and to generate five different visual representations were developed in Processing(2) (Java).

The aim of the study was to start at all an investigation into the possibilities of agent-based systems to generate rhythmic patterns, as well as testing the potential for rhythmic patterns to be represented in other ways (in contrast to common musical rhythm notation).

Four randomly generated systems with increasing structural complexity were examined.
Methods to remove redundancies from certain system configurations are presented.
The study also shows how directed graph diagrams can be restructured in ways that make it easier to identify loops in the system.
It becomes apparent, that any such system can be transformed into a linear trigger-chain structure. The new structure generates the same rhythmic pattern and, by abandoning inner loops, produces diagrams of reduced visual complexity. These diagrams can function as alternative notation systems.
All example systems generate stable rhythmic patterns. These loops differ in lengths as well as the time needed to reach stability within the delay & trigger-propagation system.
Links to sound-files that demonstrate the audible output of each example system are provided.

Further studies will focus on more structured and automated generation processes to find possible correlations of parameters as given in the definition data and the different forms of representation of the system, including its manifestation as an audible rhythm.

(1) 'autonomous' here refers to the agents not being centrally controlled, but each acting by its own rules upon incoming signals

(2) see: https://processing.org

Agent-Based Rhythm Generation

November 2023

An artistic research project by Thomas Poeser

The subject of investigation is a system of connected agents, each playing a sound and triggering connected agents after a fixed, agent-specific time delay from receiving a trigger signal themselves.
While in delay phase, new incoming trigger signals

- are ignored, (This seems to be working best; outcomes presented here are based on this behavior.)
- reset the delay time, or
- extend the delay time by a certain amount.

Self-loops (i.e. agents triggering themselves) are not allowed.

The system was modeled in the Creative Coding environment 'Processing', as were several additional applications to analyze graphs and generate diagrams.

Four simple examples of possible systems are shown in the following, starting with simple structures, then going toward increasing complexity. In terms of data structures, these systems are called 'directed graphs', where agents are represented as 'nodes' and arrowed 'edges' show the directed connections between them (i.e. which agent triggers which other agent).

Example 1: Simple 3-Node System

At the start of the program, the predefined 'seed_node' gets triggered. After its delay time passed, the agent plays its sound and triggers all connected 'to_nodes' (i.e. agents on the receiving end) which then enter a delay state of individual length in time (if they are not already in delay state). Consequently, these agents will play their own sound and trigger each of their own connections after their own delay time finished. And so on ...

```
<?xml version="1.0" encoding="UTF-8"?>
<graph seed_node_id="0">
  <node delay="4" file="perc_1.wav" id="0">
    <edge to_node_id="2"/>
  </node>
  <node delay="4" file="perc_2.wav" id="1">
    <edge to_node_id="0"/>
  </node>
  <node delay="5" file="perc_3.wav" id="2">
    <edge to_node_id="1"/>
  </node>
</graph>
```

xml definition of the directed graph

In Example 1 each node triggers the next. The last one triggers the first. In the corresponding 'score'(3) a rhythmic loop(4) evolves, right away. A resulting rhythmic loop indicates a loop in the directed graph.

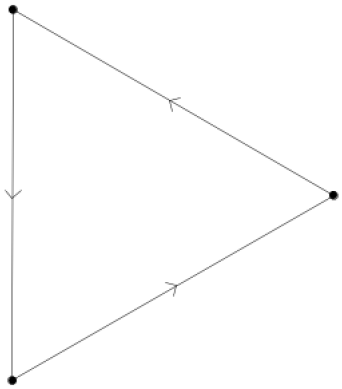
score of the resulting rhythmic sequence – the associated sound file is [s1]

(3) 'score' is similar to a time-line; showing which of the sounds –node ids are represented by the rows, or 'tracks'– gets played at what time; the discrete time unit used here is the 'tic' (a.k.a. 'beat' as in 'beats per minute')

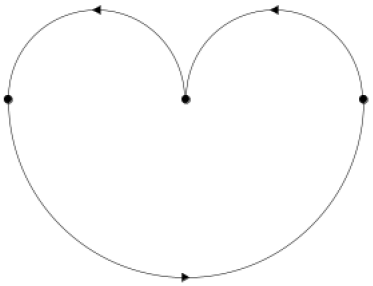
(4) i.e. a repeating sequence as indicated by the green bar in the score (loops are automatically detected by the software)

Example 1: Simple 3-Node System (continued)

In the visual representations of the graphs with either equally spaced or linear placed nodes, the loop behavior is clearly recognizable.

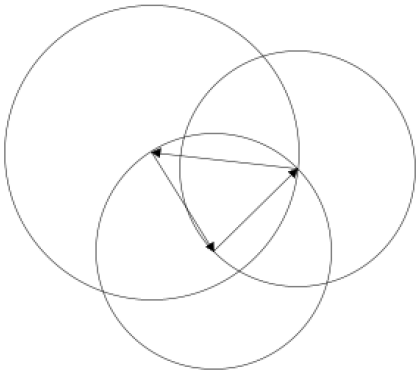


directed graph with nodes equally spaced

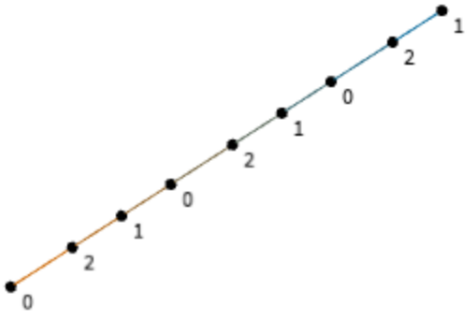


directed graph with linear placed nodes

A force-directed diagram representation approximates the connecting properties between nodes (in this case these are the individual delay times) as forces of imaginary springs between the nodes. Starting with a configuration with equally spaced nodes as above, a dynamic process leads (in most of the simple cases) to an equilibrium of forces, where the value of the connecting property corresponds to the length of the connecting arrow.



force-directed graph (shows delay time as distance)



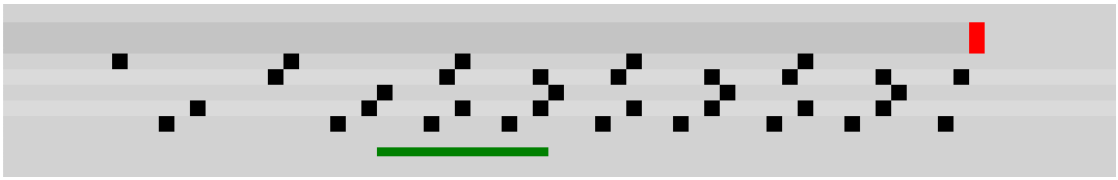
sequence diagram (shows delay time as distance)

The sequence diagram visualizes which agent triggers which other agent(s) at what time after getting triggered itself (here the series goes from yellow to blue in time; only a certain depth of steps is shown). In more complex systems where an agent triggers more than one other agent, the sequence will branch and the diagram will resemble a tree (or a root if presented the other way around). This can best be explored as a visualization in 3D space (which is not obvious in the case of example 1).

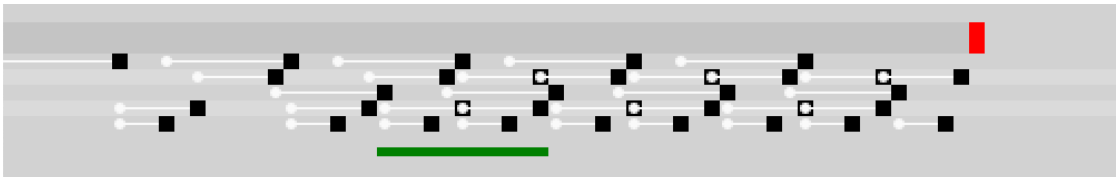
In the following, examples of higher complexity are presented in the same order of visual representations.

Example 2: 5-Node System

```
<?xml version="1.0" encoding="UTF-8"?>
<graph seed_node_id="0">
  <node delay="7" file="perc_1.wav" id="0">
    <edge to_node_id="3"/>
    <edge to_node_id="4"/>
  </node>
  <node delay="4" file="perc_2.wav" id="1">
    <edge to_node_id="2"/>
  </node>
  <node delay="6" file="perc_3.wav" id="2">
    <edge to_node_id="3"/>
    <edge to_node_id="4"/>
  </node>
  <node delay="4" file="perc_4.wav" id="3">
    <edge to_node_id="1"/>
  </node>
  <node delay="2" file="perc_5.wav" id="4">
    <edge to_node_id="0"/>
  </node>
</graph>
```

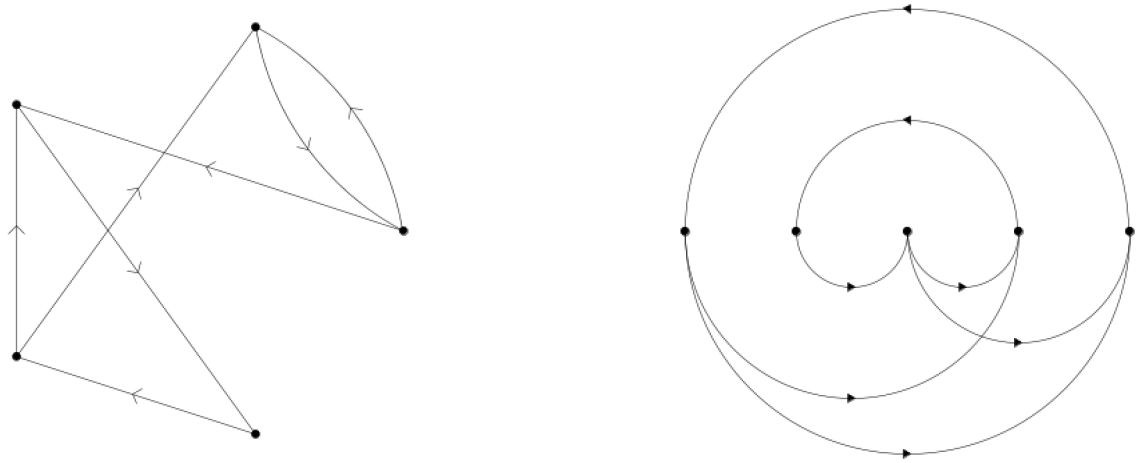


The loop manifests later in the score than in example 1 [s2]



Here, the score additionally shows delay times and trigger points (just when, but not by which agent)

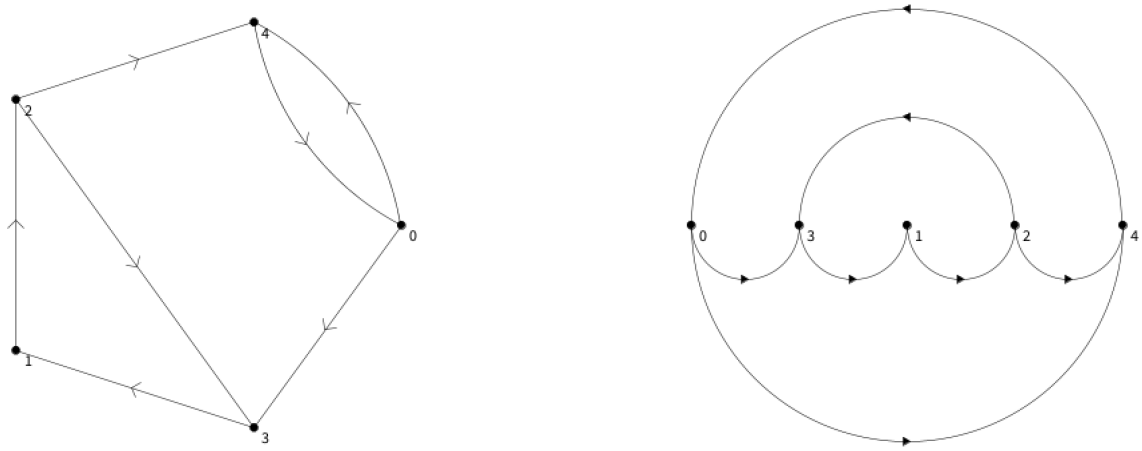
The visual representations of the graphs in example 2 initially (i.e. when placed in the order of their node ids) lack some clarity as connections seem tangled with arrows crossing each other.



tangled directed graphs with nodes placed according to node ids

Example 2: 5-Node System (continued)

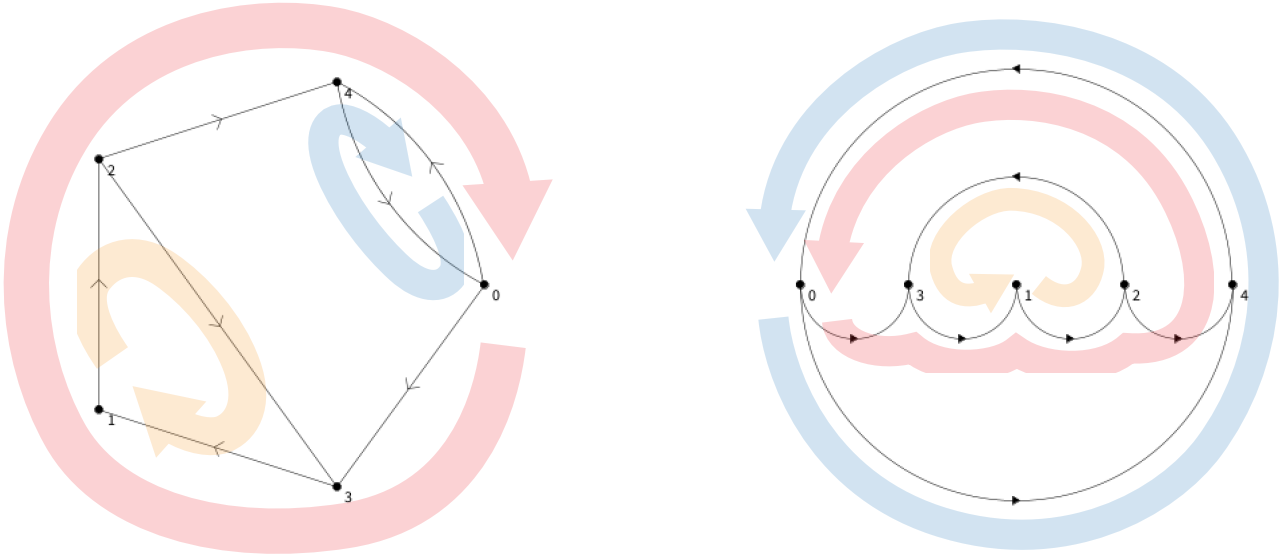
Swapping some node positions avoids crossing connections and helps visual clarity. Examples of the possible resulting diagrams are shown here, also stating the corresponding node ids:



un-tangled directed graphs with swapped node positions

In the un-tangled versions, the three loop configurations present in the system can be more easily identified. They are (by following node ids):

- 0 → 3 → 1 → 2 → 4 → 0 → ...
- 0 → 4 → 0 → ...
- [0 →] 3 → 1 → 2 → 3 → ...

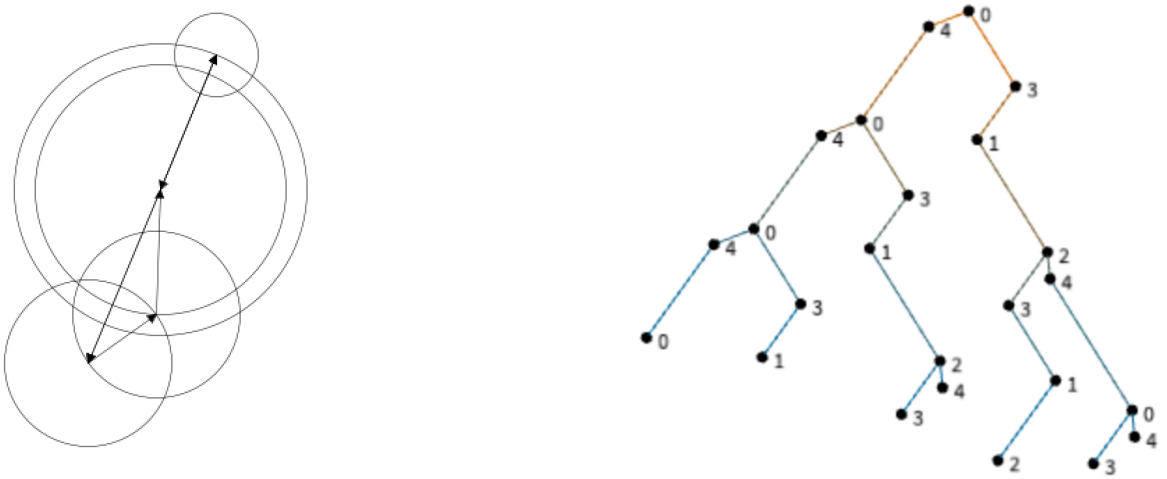


In how exactly the amount of loops, their lengths and inter-connectedness (forking, joining and sharing of paths) effect the resulting rhythmic pattern is elusive. Though, it is to be expected, that an increase in any of those properties will lead to higher complexity of both the graph system and resulting rhythmic pattern.

Example 2: 5-Node System (continued)

However, in the next chapter it will be shown that complex graph structures do not necessarily result in equally complex rhythmic patterns. The reason lies in that with the chosen behavior –to ignore incoming trigger signals when already in 'the delay-was-already-triggered' state– a trigger signal does not necessarily cause the receiving node to play its sound. If a particular connection never triggers the receiving end, that connection is redundant (which is not the case for any connection here).

The force-directed graph of the system in Example 2 does not seem to reveal any relevant relations between the underlying structure and its rhythmic manifestation. In a semiotic sense, it still serves well as a unique representation of the system (and as one of many possible ways of presenting the implied rhythmic pattern).



force-directed graph and sequence diagram for Example 2

The sequence diagram offers a window through which structures of ever repeating configurations can be observed. Though, detailed information about the timing of trigger events and sound play are hardly discernible (due to e.g. depth distortion). This especially holds true for still image representations. A better impression of shapes can be obtained by continuously rotating the tree-like diagram within the 3D space. An example video of the sequence graph as above, rotating around the vertical axis, is available [v2].

Random Generation and Redundancies

The system-configurations of all examples in this study were in their first instance generated within a framework of pre-defined parameters:

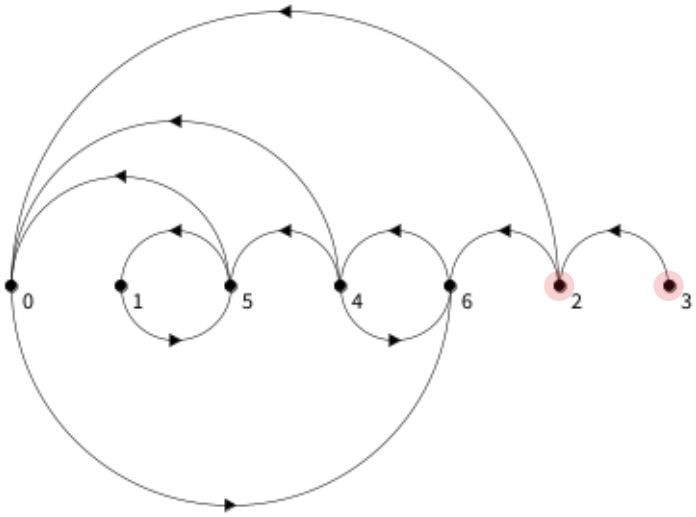
general:
- amount of nodes
- seed node (the node with id 0 was used throughout all examples)

per node:
- maximum amount of trigger target nodes
- range of delay time (in tics) before sound play / trigger propagation
- list of sound files (percussive sounds)

of which particulars for each specific node were randomly chosen.

The resulting system was then checked for redundancies and all nodes and connections removed that do not influence the development of the rhythmic pattern. For instance, inspecting the following initial data structure we notice that the nodes with ids 2 and 3 are never triggered by any other nodes than by one triggering the other. These two nodes will never get triggered in the sequence that starts with the seed node (id 0). In the graph representations, they show up with dead-end paths that are also one-way-only – like a road that you are only allowed to drive out of, but which cannot be entered any other way. Since nodes 2 and 3 are never in use they can be left out of the system definition data (see the data basis in the next chapter: Example 3).

```
<?xml version="1.0" encoding="UTF-8"?>
<graph seed_node_id="0">
  <node delay="15" file="perc_1.wav" id="0">
    <edge to_node_id="6"/>
  </node>
  <node delay="16" file="perc_2.wav" id="1">
    <edge to_node_id="5"/>
  </node>
  <node delay="12" file="perc_6.wav" id="2">
    <edge to_node_id="0"/>
    <edge to_node_id="6"/>
  </node>
  <node delay="10" file="perc_7.wav" id="3">
    <edge to_node_id="2"/>
  </node>
  <node delay="18" file="perc_3.wav" id="4">
    <edge to_node_id="5"/>
    <edge to_node_id="0"/>
    <edge to_node_id="6"/>
  </node>
  <node delay="3" file="perc_4.wav" id="5">
    <edge to_node_id="1"/>
    <edge to_node_id="0"/>
  </node>
  <node delay="16" file="perc_5.wav" id="6">
    <edge to_node_id="4"/>
  </node>
</graph>
```



nodes 2 and 3 never get triggered by other nodes

In another example of redundancy, here, in the sequence of time, at every attempt by node 2 to trigger node 1 (every 4 tics), the latter is still in its delay-time of 9 tics and the incoming signal that node 2 sends is ignored. Since this happens in every instance, the connection ‘2 triggers 1’ is redundant and can be left out of the system definition data. Similar conditions can be found for the connections ‘1 triggers 0’ and ‘3 triggers 2’.

```
<?xml version="1.0" encoding="UTF-8"?>
<graph seed_node_id="0">
  <node delay="2" file="perc_1.wav" id="0">
    <edge to_node_id="2"/>
    <edge to_node_id="1"/>
  </node>
  <node delay="9" file="perc_2.wav" id="1">
    <edge to_node_id="3"/>
    <edge to_node_id="0"/>
  </node>
  <node delay="2" file="perc_3.wav" id="2">
    <edge to_node_id="1"/>
    <edge to_node_id="0"/>
  </node>
  <node delay="2" file="perc_4.wav" id="3">
    <edge to_node_id="2"/>
  </node>
</graph>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<graph seed_node_id="0">
  <node delay="2" file="perc_1.wav" id="0">
    <edge to_node_id="2"/>
    <edge to_node_id="1"/>
  </node>
  <node delay="9" file="perc_2.wav" id="1">
    <edge to_node_id="3"/>
  </node>
  <node delay="2" file="perc_3.wav" id="2">
    <edge to_node_id="0"/>
  </node>
  <node delay="2" file="perc_4.wav" id="3">
  </node>
</graph>
```

Random Generation and Redundancies (continued)

Redundant trigger connections might not be found straight-forwardly by examining the raw data or its visualizations in form of diagrams. The task is helped by running the system configuration (in the application that generates the ‘score’ and actually plays the sounds in real time) and noting down all successful trigger attempts of one node to another in the time sequence, until a steady loop gets detected. So in our last example, the attempts ‘2 triggers 1’, as well as ‘1 triggers 0’ and ‘3 triggers 2’ do not get noted down and can be removed from the definition data.

Here the corresponding protocol of successful trigger events (only unique pairs, i.e. doubles omitted, during the sequence starting with node 0, noted down until loop detection):

0 → 2
0 → 1
2 → 0
1 → 3 with ‘→’ for ‘triggers’

Example 3: More complex 5-Node System

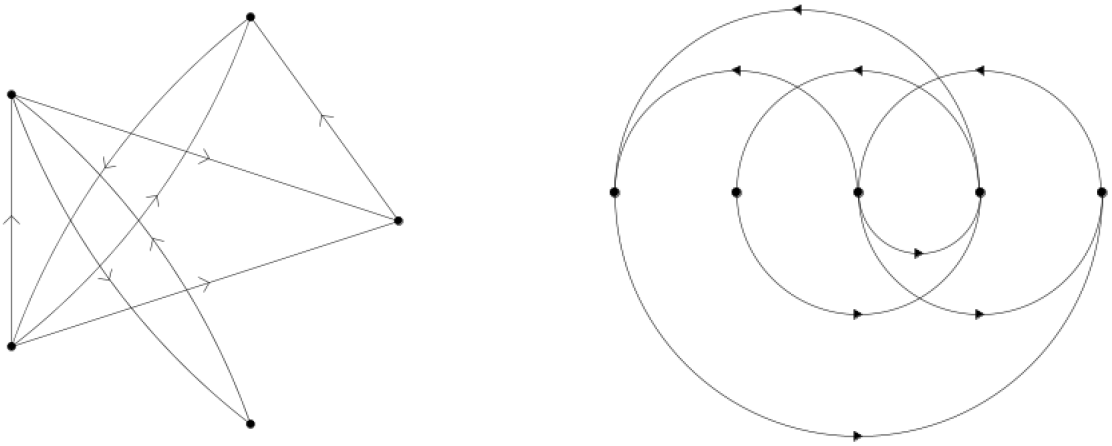
The data for this example stems from the first one in the previous chapter – this time, with the redundant nodes removed.

```
<?xml version="1.0" encoding="UTF-8"?>
<graph seed_node_id="0">
  <node delay="15" file="perc_1.wav" id="0">
    <edge to_node_id="6"/>
  </node>
  <node delay="16" file="perc_2.wav" id="1">
    <edge to_node_id="5"/>
  </node>
  <node delay="18" file="perc_3.wav" id="4">
    <edge to_node_id="0"/>
    <edge to_node_id="5"/>
    <edge to_node_id="6"/>
  </node>
  <node delay="3" file="perc_4.wav" id="5">
    <edge to_node_id="0"/>
    <edge to_node_id="1"/>
  </node>
  <node delay="16" file="perc_5.wav" id="6">
    <edge to_node_id="4"/>
  </node>
</graph>
```

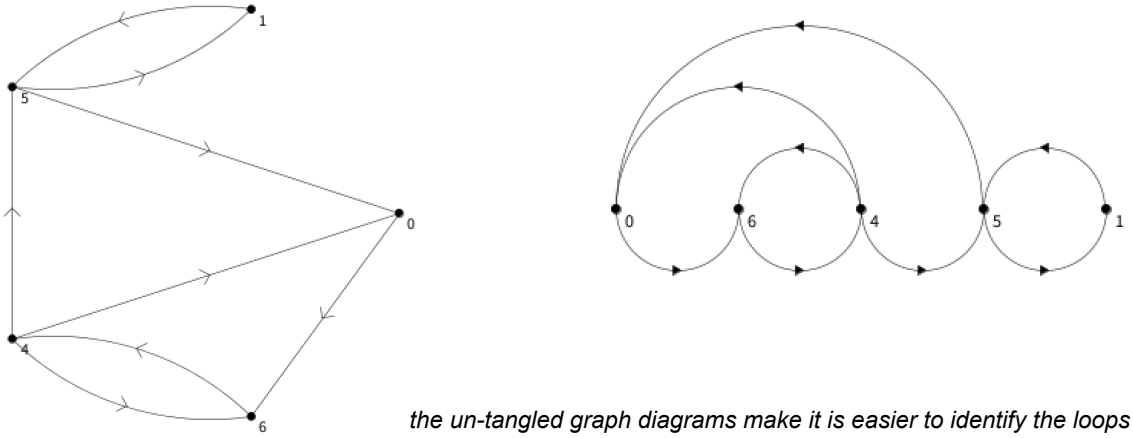
The loop manifests very late in the score – it starts at tic 134 and is 21 tics long [s3]

It becomes apparent that systems differ in the time needed to reach an equilibrium, i.e. to get into a stable rhythmic repetition / loop. The process could be described as the system going through a 'swinging-in' phase. In cybernetics, it would probably reasoned to be caused by ‘negative feedback’, i.e. loops that take momentum out of the system. Apart from an intuitive guess that short delay times, a low number of nodes and connections could result in a short swinging-in time, the exact influence of parameters in more complex systems is elusive.

Example 3: More complex 5-Node System (continued)



the initial graph diagrams are convoluted



the un-tangled graph diagrams make it is easier to identify the loops

Unfortunately, attempts to get a stable force-directed diagram were not successful with this system. It seems to oscillate between several similar configurations and would serve better as a dynamic representation. An image is omitted here, but a video is available to show the process [v1].

The following two diagrams may convey the increase in visual complexity that is added by each step in the sequence. Repeating structures correlate with loops. The diagram tree can be shown with any node as selected starting point, progressing any defined amount of steps (within reasonable bounds) from there.

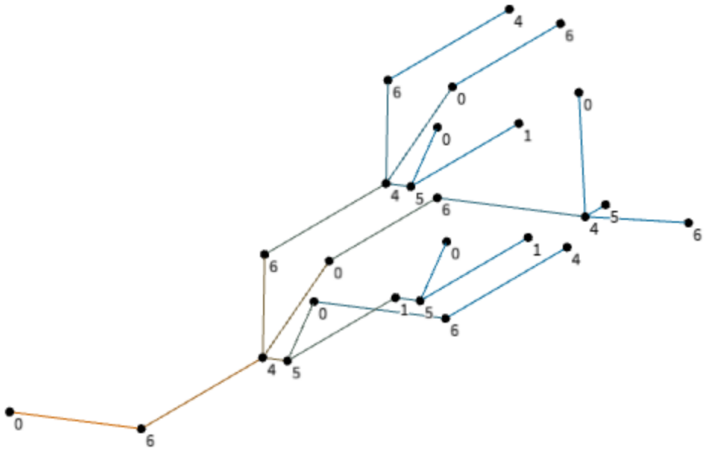
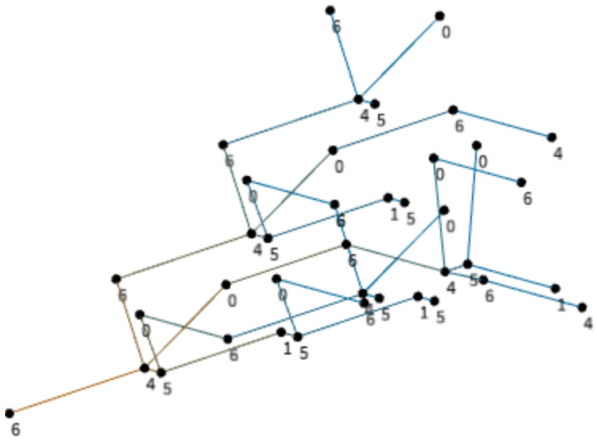


diagram shows the first 6 steps of the sequence



6 steps, starting from node 6 (with slightly altered viewing parameters)

Simplified Generation and Reproduction

It should be noted that every looping system of the kind that is studied here, can be translated into a ‘trigger-chain’ configuration, where each sound (or group of simultaneous sounds) is triggered by the previous. Such a structure can be observed in Example 1. Within this configuration, where every sound event in the loop is represented by its own node (well, in some cases, it might be possible to identify and form sub-structures that can be re-used and fewer nodes are needed in total), the swinging-in time can be reduced to a minimum.

Let us have a look at our last example, specifically its score:

The score of Example 3 – it features 5 tracks; the looped sequence is 21 tics long, in which 6 sound events occur

The sound loop as shown on the score (indicated by the green bar) can be translated into the following data structure:

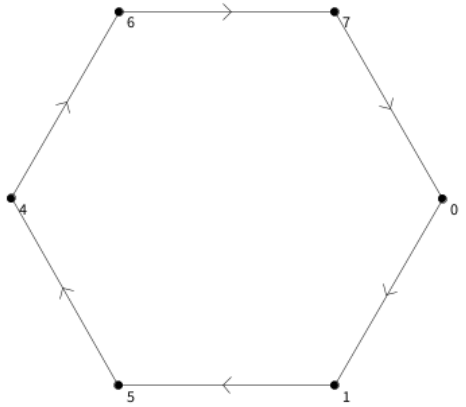
```
<?xml version="1.0" encoding="UTF-8"?>
<graph seed_node_id="0">
  <node delay="1" file="perc_1.wav" id="0">
    <edge to_node_id="1"/>
  </node>
  <node delay="0" file="perc_2.wav" id="1">
    <edge to_node_id="5"/>
  </node>
  <node delay="9" file="perc_3.wav" id="4">
    <edge to_node_id="6"/>
  </node>
  <node delay="3" file="perc_4.wav" id="5">
    <edge to_node_id="4"/>
  </node>
  <node delay="1" file="perc_5.wav" id="6">
    <edge to_node_id="7"/>
  </node>
  <node delay="1" file="perc_4.wav" id="7">
    <edge to_node_id="0"/>
  </node>
</graph>
```

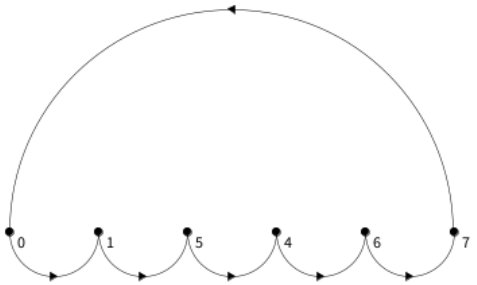
which generates the following score:

The sound pattern/loop that is generated from the new data structure is a one-to-one reproduction of the former sound pattern/loop. The score now has 6 tracks since there are 6 unique nodes, though only 5 different sounds occur as nodes 5 and 7 play the same sound (these are on positions 3 and 6 in the loop sequence as shown on the score).

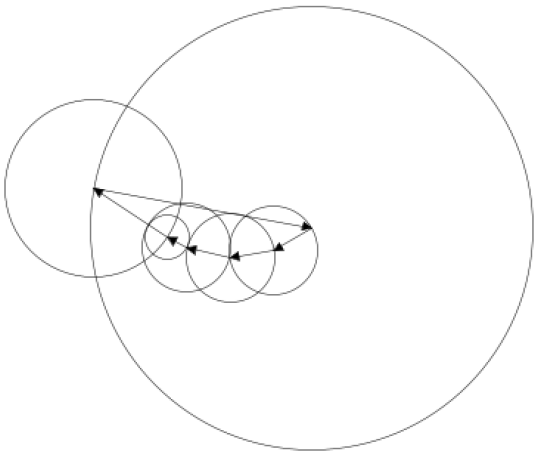
A remark: a delay time of zero (above for node 1) means that both, playing the sound and trigger propagation, are initiated on the following tic. As zero values often cause problems in visualizations of actually present entities, for the following diagrams that represent delay times (force-directed graph & sequence diagrams) all delay times were increased by one – which is just a different interpretation of the time scale in the score.

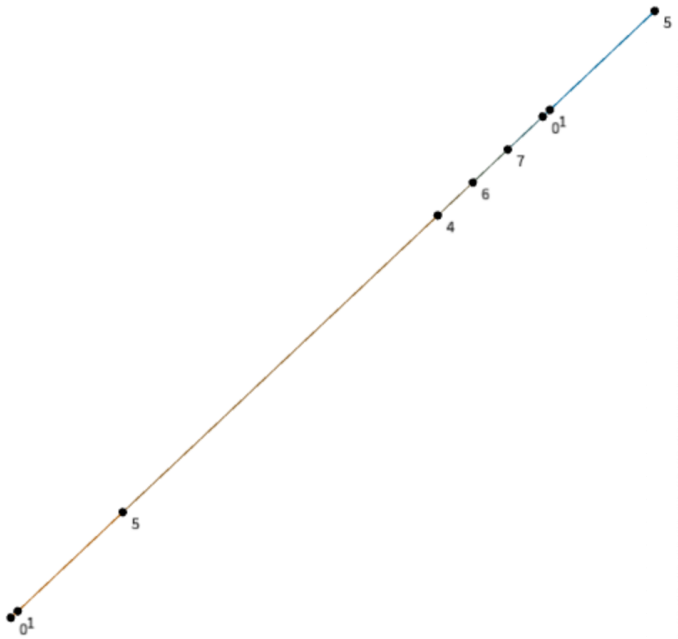
Since, in this example, no more than one sound is played at any given time (i.e. it is a monophonic pattern), the graphs have a single strand of progression (no forking – the system describes a closed loop where each node has exactly one incoming and one outgoing connection).





simple step-by-step triggering





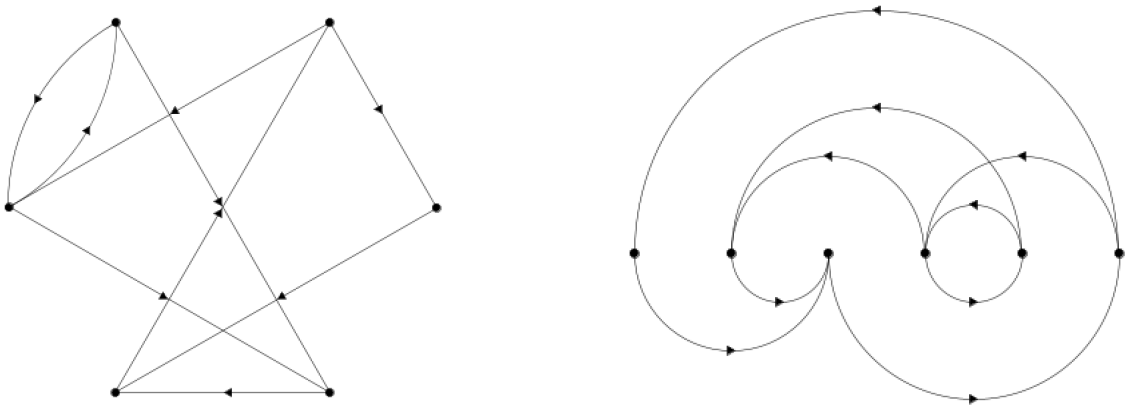
all delay times were increased by one to show distinct nodes

The conceptual simplification leads to diagrams that are more intuitive to ‘read’. Depending on the amount of sound events within the looped sequence, the reduction of visual complexity will have to be paid for by an increase in nodes to be defined in the data representation, followed by an equivalent increase in tracks on the score.

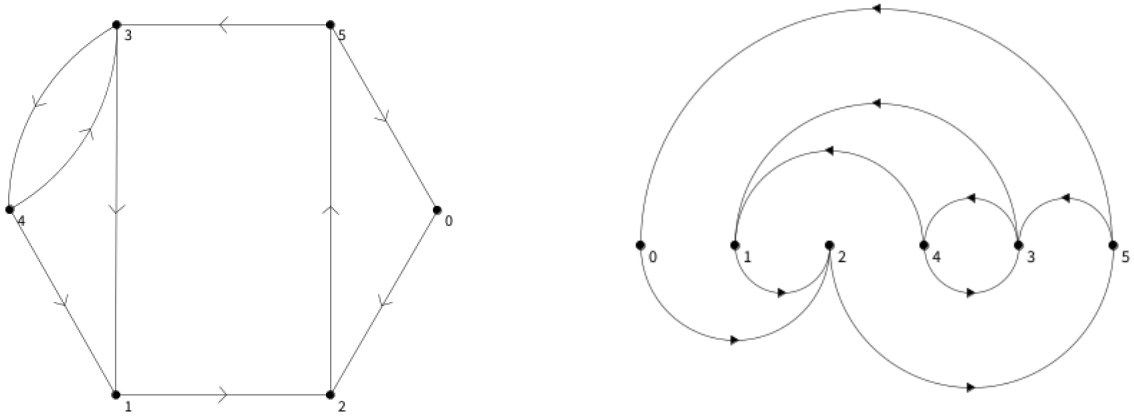
Example 4: 6-Node System

```
<?xml version="1.0" encoding="UTF-8"?>
<graph seed_node_id="0">
  <node delay="19" file="perc_1.wav" id="0">
    <edge to_node_id="2"/>
  </node>
  <node delay="7" file="perc_2.wav" id="1">
    <edge to_node_id="2"/>
  </node>
  <node delay="10" file="perc_3.wav" id="2">
    <edge to_node_id="5"/>
  </node>
  <node delay="2" file="perc_4.wav" id="3">
    <edge to_node_id="1"/>
    <edge to_node_id="4"/>
  </node>
  <node delay="19" file="perc_5.wav" id="4">
    <edge to_node_id="1"/>
    <edge to_node_id="3"/>
  </node>
  <node delay="6" file="perc_6.wav" id="5">
    <edge to_node_id="0"/>
    <edge to_node_id="3"/>
  </node>
</graph>
```

The loop is hard to recognize, due to its length in time – it is 87 tics long and starts at tic 45 [s4]

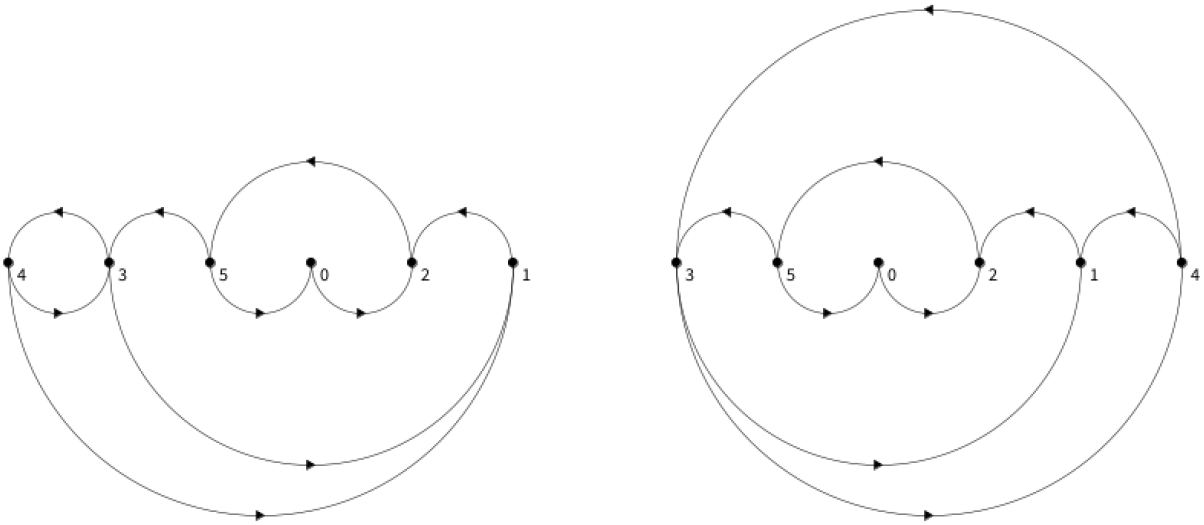


tangled representations

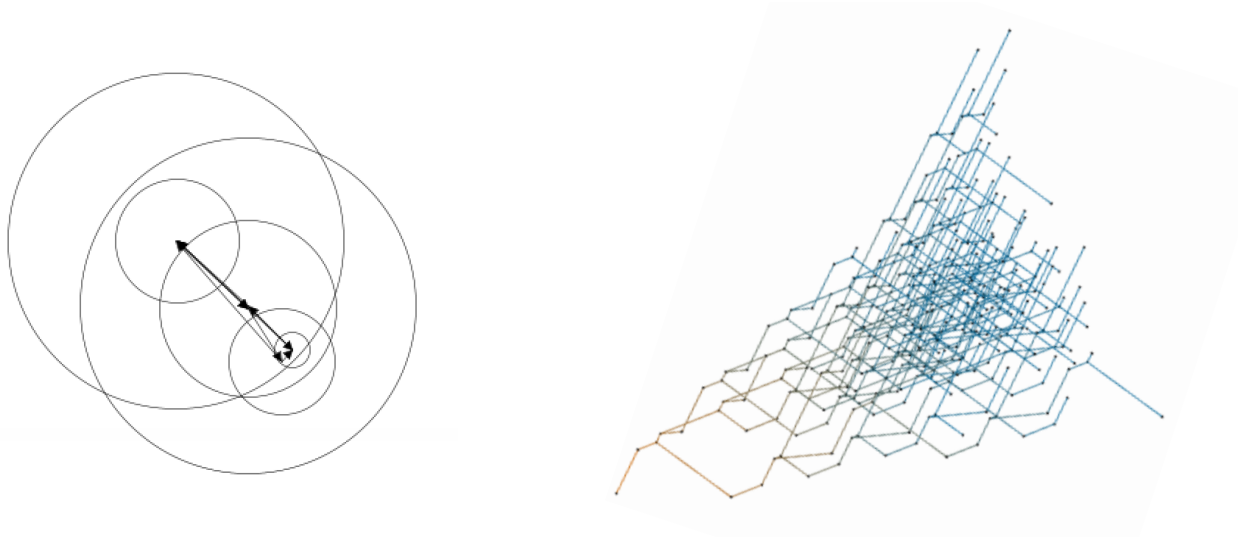


un-tangled representations

Example 4: 6-Node System (continued)



alternative un-tangled representations



force-directed graph and sequence diagrams, representative but also vague in 'meaning'

Ideas and Outlook

The two main themes that I would like to explore further within this framework are 1) systems that manifest in a chaotic way, i.e. that show non-stabilizing behavior and 2) the potential of expressiveness that force-directed graph representations can offer (more from an artistic viewpoint than expecting structural clarity).

According to present cellular automaton(5) theory there should be the possibility of system configurations that never reach a stable repetition (Stephen Wolfram [6] is a good source of information in regards to research on cellular automata).

If there are system configurations that cause infinite, never repeating output, how are they structured, what exactly makes them unstable, and first of all, how can they be found? One thing is already given: their graph

(5) The presented system shows similarities with different model definitions that involve networked entities, like (Augmented) Transitional Networks / Petri Nets, Multi-Agent Networks, and some forms of Neural Networks, though it cannot be clearly categorized as a specific type. The closest relations I was able to identify are with Douglas Hofstadter's definition of an 'actor model' [3], which, as agent-based systems in general, shares behavioral characteristics with the conceptually related cellular automata [1] [4].

should also have a loop, since that is a pre-condition for infinite running of a finite system (i.e a system with a limited amount of nodes) – just that this loop does not cause a repetitive pattern. But on which other factors does (in-)stability of the running system depend? In trying to find unstable systems, one strategy would be to study the influence of increasing grades of complexity on the stabilization process. A different approach would be to try to find ways to sabotage the self-regulating process of a stable system.

Many possible alterations of system behavior are possible. These could, for example, be directed towards introducing non-deterministic processes(6), reacting to changes in a physical environment(7), including the active manipulation of rules and parameters during play(8). It should also be pointed out that the inner mechanisms of any such system could –within a certain frame– be performed by humans(9).

Whilst force-directed graphs seem to work well in depicting simple configurations, with higher complexity, it gets harder to draw any relevant information from those diagrams. In the best case, we get a diagram similar to the graph with equally distanced nodes, though exactly not equally apart, but with the distances matching each node's delay time when seen in relation to each other.

It is, for example, possible to generate a simple, accurate representation of the timing as shown on the score by translating the sequence of sound events into a trigger-chain system (see chapter "Simplified Generation and Reproduction"). On the corresponding force directed graph all time-related parameters are now more easy to discern. This diagram constitutes one of the possible alternative notations of the particular sound pattern. Though, in the most likely case, that in which the translation into a trigger-chain format creates a new system (i.e. one that is structurally different from the system which initially generated the score), it is impossible to draw any but the most trivial inferences from the new diagram about any original underlying system architecture.

From a semiotic point of view, the force-directed graph is a sufficiently representational sign for (or signifier of) a system – any two systems can be differentiated by comparing the structure of their force-directed graphs (geometric mirroring of structures aside). If the 'seed/start node' gets marked (i.e. so it can be identified; this is missing in the diagrams presented here), all of the system parameters are translated into distinct visual features. In many cases, the sequential / in-time behavior is difficult to identify and a direct relation to the rhythmic behavior is hard to establish. However, the presented diagrams could be used as abstract scores or rough guides for improvisation in a performative context (performers would interpret the inter-relations of the graph-representation at their own will).

Once the underlying workings of these systems are uncovered (at least partially and much better than at present), it might be possible to involve the graph structures in the composition of rhythms in a directed way(10).

(6) Non-deterministic behavior can be achieved though built-in, probabilistic parameter selection processes (for single agents, initiated by incoming trigger signals). This brings the system closer to –and may even transform the system into– a Markov Chain.

(7) Each agent could react to changes in its wider environment by altering its parameter settings. Sensor data would influence the delay time, choose which sound to play or take decisions on which other agent(s) to trigger. Such a configuration offers interaction with humans (i.e. a potential use as some form of instrument) and / or a link to more general ambient conditions.

(8) A performer may actively change the system configuration, initiating transitory rhythmic disruptions. The undefined rhythmic sequence during the time the system needs to self-organize (before a new rhythm 'emerges') might be interesting to listeners. The length of the adaptive process is unpredictable for switches other than those in between two direct trigger-chains.

(9) The system is, in its algorithmic workings, substrate independent (or substrate neutral, see [2]). It is convenient to use a computer program to model the system and exercise the sequential application of rules, though the underlying functions could in theory be performed by humans, as well (through signalling, counting beats and invoking a percussive sound).

(10) An interesting compositional approach would be to draw or construct a system diagram, which then functions as the basis for the generation of the rhythm. The algorithmic generative software 'Nodal' [5] works in a similar fashion: there, a map of pathways and nodes is constructed on a grid, resulting in a network that 'performing agents' traverse in timed intervals, invoking certain actions when encountering a node.

Ideas and Outlook (continued)

Next step here will be to generate many more examples to find out, if there exists a form of correlation, i.e. a meta-pattern of rhythmic behavior generating parameters, and how intuitive those are discernible in force-directed graphs. Exact measures aside, the diagrams are definitely a suitable artistic representation. In my eyes, they form an aesthetically pleasing abstract work (maybe interesting to show whilst playing the sound).

For both goals, it will be helpful to establish a way of automating the generation of varying graphs and their different forms of representation in conjunction with a more structured approach of choosing distinct system parameters. Through serial works where only few of the same parameters are changed in value from one to the next (i.e. in a progression of parameter values) some correlations might become apparent.

Literature and References

[1] Roger T. Dean (ed.), Alex McLean (ed.): Oxford Handbook for Algorithmic Composition, chapter: Alice Eldridge, Oliver Brown: Biologically Inspired and Agent-Based Algorithms for Music, sub-chapter: 13.3.1 Agent-Based Modelling

[2] Daniel C. Dennett: Darwin's Dangerous Idea, 1995 Simon & Schuster, p.50-51

[3] Douglas R. Hofstadter: Gödel, Escher, Bach. New York : Basic Books, 1979, p. 662

[4] Gerhard Nierhaus: Algorithmic Composition. Springer Vienna, 2009, chapter: 10.1.6 Agents, p. 254

[5] https://nodalmusic.com

[6] Stephen Wolfram: A New Kind of Science. Wolfram Media, Inc., 2002

Additional Media

videos: [v1] https://www.tp23.co.uk/projects/sagent_net/example_3-untangled-fd_graph.mov (8,5 MB)
dynamics of the non-stable process whilst attempting to establish a balanced force-directed graph

[v2] https://www.tp23.co.uk/projects/sagent_net/example_2-3d_sequence.mov (11,6 MB)
sequence diagram (3D), rotating around the vertical axis

sound: [s1] https://www.tp23.co.uk/projects/sagent_net/example_1.mp3 (224 kB)
[s2] https://www.tp23.co.uk/projects/sagent_net/example_2.mp3 (296 kB)
[s3] https://www.tp23.co.uk/projects/sagent_net/example_3.mp3 (749 kB)
[s4] https://www.tp23.co.uk/projects/sagent_net/example_4.mp3 (1,1 MB)
recordings of the score-generating application’s sound output
with limited amount of sounds shared by several nodes (for a more rounded rhythmic expression)