xCoAx 2021 9th Conference on
Computation, Communication, Aesthetics & X

2021.xCoAx.org

Synthesis and Operation Flows

Keywords: Synthesis, Algorithms, Signal Flows, Sound Streams, Sound Transformation

Bjarni Gunnarsson
Gunnarssonb@koncon.nl
Institute of Sonology,
Royal Conservatoire

Digital sound synthesis can be described in terms of discrete operations applied to signals according to given criteria. The process of organising such operations can be subject to creative variation and generative algorithms. This article presents proposals for how a sound construction process can be thought of as a combination of synthesis and applied transformations. Operational spaces are examined and how process configuration and incompleteness can serve as creative models for synthesis. Properties of sound streams are also considered and how they can involve liveness and unpredictability. A software framework is finally introduced followed by a reflection on its use in creative practice.

1. Introduction

Algorithmic music procedures often involve generating data that requires mapping to a musical domain. Sound synthesis on the other hand usually concerns a more direct description of a signal flow. Instead of using predefined algorithms during composition what is proposed here is to use dynamic workflows inspired by data transformation pipelines and signal flows. Such processes are adapted in order to produce algorithmic sound based on synthesis, transformations and the manipulation of sound streams.

The goal here is to propose a model of a *synthetic sound stream* as a *pipeline of transformations* that can be freely manipulated or changed while they run. A related concern is how to generate transformation pipelines in order to produce evolving sounds. Focusing on synthesis and sequences of sound transformation reveals a working mode centred entirely around sound itself. It also introduces a functional strategy that concerns treating sound as an *operation sequence*. Finally, it questions how dynamic changes in synthetic sound streams can work as a flexible interaction mode and how the attitudes of evolving algorithms can be made audible through the resulting sound.

A software framework named OF (Operation Flows) will be introduced that implements some of the ideas covered here in this text. It consists of four main process categories:

1. Synthesis processes with dynamic waveform generation

2. Atomic sound transformations that adapt to incoming signals

3. Operation pipelines and algorithms to control and generate them

4. Direct access, observers and reactions for content-based triggers and actions

The framework allows for experimenting with synthesis and algorithmic pipelines of sound transformations. It is designed to be configured with common existing workflows and to be easily extensible. Examples that demonstrate the relation between the text and the system will be presented here below.

2. Background

2.1. Operational Spaces

Computer music processes consist of transformations that are applied to musical material in order to develop and extend it over time. Such transformations fluidly translate into algorithms allowing for automation, iteration or generative possibilities. The construction of automated processes, or the encoding of *"frozen thought"* (Chaitin 2005, 24), preferably involves transparent, elementary operations that are combined into precise steps and a clear sequence of actions. How mechanically executable tasks are defined and a chain of operations is arranged is something considered here as part of original sound synthesis and the becoming of electronic sound.

Creative intentions take shape during the activity of sound production and not only prior to it. By focusing on the operational as 'on-going', the aim here is to explore conditions where one continuously engages with generative sound processes and how they are constructed. The arrangement of synthetic sound streams and how connecting to them can be part of a creative process creates a bridge between algorithms and the operations of electronic music production. Focusing on the actions and algorithms behind sound streams allows for data processing to be considered as a fruitful model for sound composition.

In her book *Programmed vision* Wendy Hui Kyong Chun refers to the process of mechanization as *'Automatic programming'* (Kyong Chun 2011, 41). She argues that it contributes both to a certain deskilling but also to the forming of craftsmanship where *"through automation, expertise is both created and called into question"*. The particularity of a given task is highlighted here through repetition and purification of the steps it consists of. Kyong Chun also emphasises the specificity of the programmable medium where algorithms translate to "this thing called software—something theoretically (if not practically) iterable, repeatable, reusable, no matter who wrote it or what machine it was destined for".

Kyong Chun's vision brings about the idea of operational space, afforded by software, where elements enter into dynamic relations and display a rich possibility of connecting. This malleable nature of software allows for *"making possible the transformation of anything into anything else via the medium of information"* (Kyong Chun 2011, 57). It is particularly useful to compare sound creation to information processing in order to think of synthesis as the selection of operations that are combined from a larger set of possible actions. The idea of Action

space shaping (Kanervisto, Scheller and Hautamäki 2020) can be useful when addressing the arrangement of available operations. It refers to the modification of action space and how it can be refined within a given context. For example, how the number of potential actions can be restricted or how actions can be combined to better address any task at hand.

For our current purposes, it is important to think of an action sequence as something that can be composed or transformed but that remains in a tight relationship with any material or data that it processes. It is also important to note that an operation selection is made from the action space afforded by available material. Operation sequences then execute over time and leave traces through the action space with a strong imprint on the resulting sound.

2.2. Configuration

For the composer Gottfried Michael Koenig the concept of musical material includes not only sounds but also the method by which they are treated. Koenig introduced compositional processes where musical form appears as the result of sound operations and the production technique of derived materials (Koenig 1987). Methods thus become structurally bound to material and provide a direction of how sounds are ordered through their operational history. Furthermore, formal relationships between sounds are highlighted by the way they have been treated where *"work-processes, composed in detail, are related to each other, and these relationships come into evidence at the surface of the sounds. Each sound is therefore not a blotch of colour but itself a form, as it owes its existence to a formal construction, to a form process."* (Koenig 1965, 8).

The point of view Koenig proposes emphasizes how a sound creation process becomes part of the structural possibilities the sound material offers. Extending this idea, the models presented here below combine synthesis with operations of transformation and control. How the operations are applied and the consequence that has on working with the generated sound becomes an important focus point for creative treatment.

Thinking material together with its treatment requires highlighting the concept of what it consists of and the element it connects to. In the context of human-machine interaction, Lucy Suchman proposes the term configuration to assist us in thinking about technological processes with a *"particular attention to the imaginaries and materialities that they join together"* (Suchman 2012, 48). The nature of applied methods plays a key role where, instead of existing as inde-

pendent of their application, *"they are figured within design and engineering discourses precisely not as already existing and independently agential, but as emerging from and dependent upon the actions of their (human) makers."* (Suchman 2012, 55).

Configuration brings elements together, it enables the separate existence of processes while extending the boundaries of what their combination can produce. By joining together, configuration contributes to the wholeness of technical processes.

On a similar thread, sociologist John Law proposes the notion of method assemblage that *"detect, resonate with, and amplify particular patterns of relations in the excessive and overwhelming fluxes of the real"* (Law 2004, 14). He believes that a research method is constructed in parallel with what it achieves. It allows for discovering realities but also to be produced along the way and to assist in the attempt to *"recognise and treat with the fluidities, leakages and entanglements that make up the hinterland of research"* (Law, 41). Sound synthesis can be considered for such a purpose as it offers ideal dimensions for investigating the imprints of automated operations and context-sensitive processing.

A tight coupling between method and material highlights the importance of direct access and situated actions (Suchman 2006). Instead of forming abstract plans, every action is situated around the context it will occur in. This could translate to investigate sound-itself as having potential characteristics of dynamic activities, continuous change, uncertainty and a tendency of becoming. The goal is to arrive at a contact point where sound synthesis, the construction of compositional action plans and development of formal ideas are all more or less the same thing. This bundling (of material and method) is achieved by constructing pipelines of transformations (discussed below) that are applied to synthetic sound.

Of great importance are the direct actions that can be applied once the pipeline executes and interact with the way it unfolds. Instead of applying musical algorithms as mechanical sequences of actions, the idea is rather to apply the transformations over time with a possibility of articulating key aspects of how each transformation becomes audible. Manual interaction with the automated pipeline pervades the operational configuration and allows for an enhanced engagement with its processes.

2.3. Incompleteness

In her book *Software Theory*, Federica Frabetti examines the ephemeral nature of software, the distinction between system conception and realisation, and the difference between *"software as a product and software as a process"* (Frabetti 2015, 104). She draws attention to the difficulty of detaching a system from its development and the importance of iteration where implementation informs specification while also being conditioned by it. This highlights the strange incompleteness of system building where *"writing, experimentation, "working out" are essential disciplines for the theoretician"*. It also questions the boundaries between intentions and results and how implementations can bring about unexpected consequences. Such repercussions (or accidents) can in turn *"'reveal' the underlying assumptions of software—the ones we rely upon in order to make software intelligible."* (Frabetti, 161).

Frabetti reflects on how unexpected system behaviour contributes to a perceived sense of separation through the novelty it creates where at *"the moment when we are 'surprised' by software—is the moment when we form a 'point of view' on software that aporetically separates ourselves from it."* (Frabetti, 160). Both her 'assumptions' and 'surprises' can be understood as processes emerging from systems (or software) without having been part of their initial specification. Assembling automated processes can introduce a sense of incompleteness that can bring about side effects, expanding the potential behaviour realised with such systems.

For many musical works based on algorithmic processes, large-scale form is still done manually and without any computer assistance (Eigenfeldt 2014). The ephemeral nature of creative generative applications involves uncertainty and incompleteness. This often makes it ideal for local experimentation but difficult for any kind of global organisation. Perhaps that is the nature of electronic music somehow. Still, models can be developed that take advantage of such tendencies. Instead of composing with fixed systems, one can imagine working out musical material while setting up the operations it will be organised through. Instead of having positive accidents occurring during development, they become part of an automated synthesis process that can then be executed in various ways.

Another aspect of local contact between methods and material is how difficult it is to predict the long-term results of an action procedure. Although extended sequences of operations can easily be calculated, their relevance tends to stay

the strongest around the local conditions from where they originate. However, events that are generated further away in time remain more uncertain to be of any use. A tension, therefore, emerges between perceived local sound and the direction of a technical process that has been pre-computed and put in motion. Addressing such tensions highlights the importance of an interactive relationship with algorithms on different levels.

3. Streams

3.1. Flow

A characteristic of raw electronic sound is how it unfolds like an endless stream. Unlike most naturally caused sounds, the synthetic ones last forever. Stopping a sound or making it disappear becomes something that requires decision making instead of being caused by properties of a sound-producing object. Synthetic sound, therefore, exists as a continuous flow waiting to be further transformed without any "limitations of human performance" (Holmes 2015, 123). For sound synthesis, this can mean to apply operations that disturb or halt a sound stream. The same principle can be applied to how a sound starts or changes.

Sound composition becomes the activity of managing a sonic flow from which things appear and later disappear. Streams of sound are characterised by their behaviour over time and the way they start and stop.

In his study of flow in television, media scholar *Raymond Williams* proposes a three-tier categorisation based on the order of detail. He claims that flow takes place over an evening programme, between events or on the detailed level of a particular movement where *"the characteristic experience, is one of sequence or flow"* (Williams 2003, 86). A flow consists of discrete items that are designed to appear in a sequence to bring about the flow and perception of progress. How the succession of the elements is designed is what contributes to their flow, motion and continuity. His investigation highlights how switching between various sources can in itself contribute to a flowing quality. A televised flow does not begin or end but takes shape once one engages with it while it unfolds.

Flow sequences can be extended to sound synthesis where transformation pipelines are activated but take place in time through programmatic switching. Algorithmic sound reflects the struggle of putting together discrete elements as unified and contributing to a coherent flow. A technical process contains a distinct flow, (or will) and directionality that potentially *"wants to sort itself out,*

to self-assemble into hierarchical levels [...] to perpetuate itself, to keep itself going. And as it grows, those inherent wants are gaining in complexity and force." (Kelly 2010, 25). Tuning into televised flow bears a similarity of connecting to information streams that possess their own will, goals and direction. Coupling such streams with operations such as switching, following or influencing can be a way of turning their treatment into a shared creation process.

3.2. Liveness

The audible streams discussed here are constructed in real-time and through attaching methods that transform a sound flow. An essential aspect is the 'liveness 'of the processes and algorithms that execute the synthesis and transformations. This occurs not only as part of the final outcome in a piece or a performance but through the liveness of the software operations themselves that are executed while a sound is being put together or refined. Context-sensitive sound transformations become concrete while operating and belong to what Wolfgang Ernst labels as time-critical actions (Ernst 2013, 143). They allow for experimentation, the building of actions chains but are also *"sensitive to micro-temporal intrusion, irritation and manipulation"* (Parikka 2018). Algorithmic sound makes the impact of its constituting operations audible. The liveness of synthetic sounds takes shape during the activation of the real-time operations it consists of. Listening to the influence these operations have highlights the technical causalities responsible for their behaviour.

The malleable behaviour of sounding algorithms can be thought of in terms of liveness but also as an example of performative action. Highlighting how they perform, Parisi & Portanova suggest that we can look at algorithms as continuously emerging from within computational processes instead of as a pre-defined collection of instructions (Parisi & Portanova 2011). Technical operations perform through the material they operate on and leave traces of how they execute. Andrew Murphie suggests that performant computational procedures can be studied on the lower-level of signal processing allowing to think *"performance more generally in terms of signal flows and breaks, signal events and signal work, relations and varying intensities."* (Murphy 2013, 2). Performance should be understood here as working with flows, including signal flows, intensities and the tendencies that generate signals. Thinking sound development through operations that deal with signals allows for approaching liveness and performability through the different levels where sound synthesis takes place.

3.3. Unpredictability

Artist-researcher Winnie Soon proposes to investigate the live dimension of code inter-action in software through the three vectors of "unpredictability, temporality and automation". (Soon, 2016, 115). She underlines how unpredictability entails the possibility of disruptions but also the uncertainty it brings. Her view resonates well with the synthesis models proposed here below where unpredictability occurs on various levels. For example how the stochastic sound-producing methods remain difficult to predict or how processes can be activated or ordered through unpredictable methods. Another aspect relates to the development of the synthesis pipelines in time. The choice of operations, their change, external influence and behaviour takes place through declarative descriptions of the processes that are made before they run. As they run, the context will always have slightly changed causing unforeseen consequences.

Unpredictability is a big factor when putting pipelines together that reveals itself when they run and become audible. This speculative aspect is fundamental to the project as it demonstrates the many experimental aspects a computational system has where many extend beyond properties of algorithms. Configurations of operations and synthesis combine, where instead of being pre-programmed they can be subject to conditions or triggering behaviour of the running pipelines. Control then moves from being imposed to occurring through the balance of system components. Automation thus contributes to situations of unpredictability through the complexity of component interactions it causes.

4. System

OF (https://github.com/bjarnig/OF) is a software framework realised as part of this project and enables the construction of sound streams and operation pipelines. OF is implemented in the SuperCollider (McCartney 2002) environment and builds on the conventions of the JITLib (Rohrhuber, de Campo, and Wieser 2005) paradigm. The models for shaping algorithmic processes and sharing information are partially inspired by data transformation pipelines and signal flows. Related approaches for transforming signal flows and streams exist such as Faust for digital sound synthesis (Orlarey, Fober and Letz 2009) or Haskell for dataflow programming (Uustalu and Vene 2005).

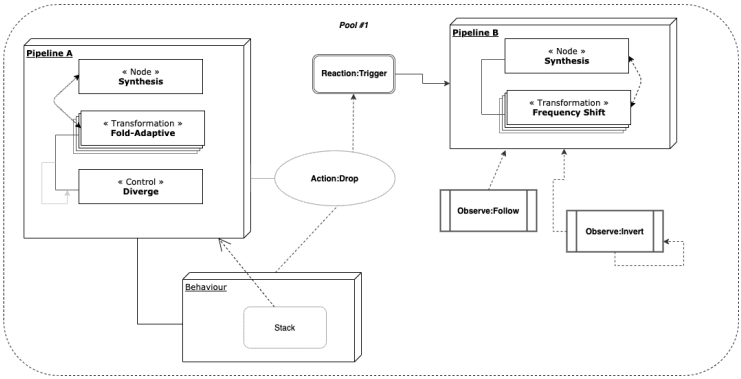

Fig. 1. *OF* Process Example.

The central element in OF is the *pipeline* that encapsulates a *synthesis* process, multiple sound *transformations* and various parameter *controls*. The pipeline is executed through entities called *behaviours* that define how a pipeline is played. A behaviour takes care of how the control and transformation operations are activated and the synthesis is made audible. The behaviours also contain different methods that a pipeline can be played with. This usually means further audio treatment of the pipeline as a whole. In OF there can be many pipelines, each with its own synthesis and associated operations. A collection of pipelines is managed through a *pool*. Pipeline pools can be treated in many similar ways as individual ones. A pipeline can also be subject to intervention or functionality change through *actions*. A pipeline can be *observed* for creating activity based on how it develops and includes *reactions* that trigger when certain conditions are met once it executes.

The OF architecture has been designed to support an open configuration where many kinds of SuperCollider objects such as *NodeProxies* and *Patterns* can be used in addition to those included. The framework supports an interactive working-mode for the creation of pipelines, how they are arranged in time and the direct access to sound streams while they execute. The pipelines follow a modular construction approach that allows addressing each of its components individually. For example to replace or modify operations or events while a pipeline runs. The pipeline operations are divided into phases in order to change between them and to support multiple starting points. Each phase represents a certain state of the pipeline that can in itself be extended in different directions. An important requirement is to always provide a minimum level of development and continuous change. To provide varying states while being capable of focusing on a specific area until the next incoming pipeline change.

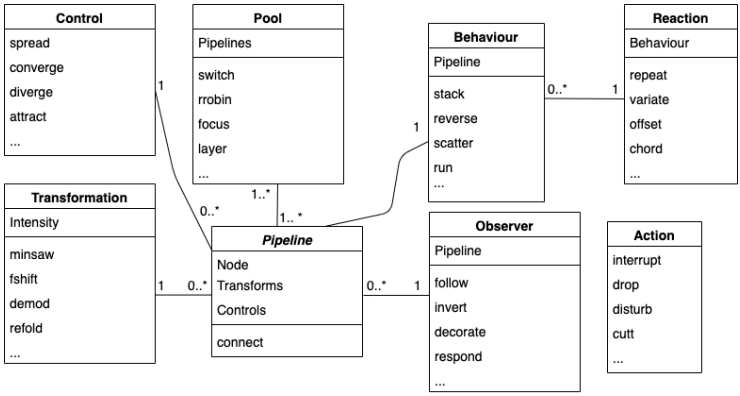

Fig. 2. *OF* Entity Diagram.

4.1. Automation

Using *OF*, development takes place through sequences of transformation that are performed on initial input. Instead of expressing how a certain algorithm operates, it can be described in a declarative manner, stating what it will do (not how). The *OF* pipelines follow a declarative convention which makes them easy to define. Configuring pipelines, their structure and behaviour and how they will be executed are fundamental operations when using the framework. Through automation, the processes become fluid and autonomous and by iterating a sequence of transformation a sound shape is produced. Most of the transformations that are included with OF simply are turned on or off but are dynamic by design. Their transformation behaviour occurs by generative principles, or by analysing the incoming sound stream. By minimising the need to change the parameters of a process while it runs, a sharper focus can be put on how operations are turned on and off and the sequence they appear in. Instead of working with parametric spaces, the idea is to allow for the opening of operational spaces that are then traversed in various ways.

An operation sequence starts on the basis of a sound synthesis algorithm. *OF* contains a small collection of sound synthesis methods that all generate waveforms that are continuously changing as they play and vary from within by definition. Both initial and runtime parameters are supported and a waveform can always be regenerated by invoking the constructor function with different initial inputs. The resulting sound streams are full in terms of spectrum and ideal for raw, rough or gritty sonorities. The audio transformations that are applied to those are of various kinds, modifying both amplitude and frequency through techniques such as waveshaping, clipping, amplitude modulation and frequency

shifting. Thinking transformations as operations sequences enforces a kind of mechanical workflow. Sound comes about through chains of behaviour, operations sequences and tight automation.

Fig. 3. *OF* Code with Pipeline GUI.

4.2. Directness

An important part of the *OF* framework is how pipelines are manipulated while under operation. Actions can be applied during runtime that can, for example, halt, or disturb a running pipeline. The working mode then changes from specifying how things happen to prevent them from doing so. From internal refinement to external forcing. Such intervention offers a very direct contact that allows focusing on the minute detail of a dynamic sound instead of the factors making it change in time. Algorithmic sound streams offer rich possibilities for new sounds (or those requiring more attention) in which case the framework allows for actions to halt sequences and highlight desired aspects of a complex sound.

Sound streams and associated operations can be combined with others of a similar kind. Instead of making links between sounds that happen simultaneously, relations unfold through the pipelines themselves and how their transformations take place. Each pipeline is isolated and unaware of others. The act of binding them together can take place by switching between streams or blending them in various ways. The way the output is treated is therefore unrelated to the pipeline behaviour itself since it occurs on the output-level only. Switching still has plenty of creative potentials and can, for example, include other actions that trigger once a switching occurs. Sound stream continuity comes to the fore-

ground where many pipelines can be activated but only a few that are audible. The creative problem then revolves around how one connects, interprets and makes audible a running process. If the pipelines are programmed to execute, the switching can also be done manually, introducing liveness, and a possibly more sensitive approach to micro-temporal details. Finally, algorithmic switching can be introduced for further exploration, for example through methods using probabilities or permutation principles.

4.3. Boundaries

The workflows described here serve as an attempt to combine attitudes of system building, sound synthesis and composing with process. The pipelines function as configurations, binding together synthesis and processing while also delineating the whole audio flow as an entity. Pipelines can be combined in sequence or in parallel, allowing for transitions from one to the next in various ways. They can run on their own, or be set in relations to others. Pipelines can also be algorithmically generated. For example by using different selections from the transformations or using stochastic processes to order them. The same applies to their duration and the delay between each transformation. One can think of the operation sequences as a driving force behind generative processes and their outcome. They shape the whole ensemble and the distribution of information. They resemble algorithms, consist of precise, programmed steps and are enacted by different software components.

Collective behaviour can also arise once several pipelines are running at the same time that are possibly influencing each other. Registering some as observing or reacting can lead to a deferral of any centralised control and move things more towards distributed chains of behaviour. Music then occurs as a result of how relations are arranged. The system develops according to local changes and can even be thought of as an emergent complex system. Such attempts deserve further investigation. However, the fundamental *OF* features are the most notable. To set up audible transformations pipelines, to manipulate those algorithmically and to manipulate sound streams using audio domain processes. A certain workflow often appears, but also a sonic imprint, a characteristic behaviour that emerges from the basic principles the framework is built around.

The *OF* software framework and the ideas that have been covered in this text are grounded in the author's creative practice and approach to synthesis and composition. The framework reflects a process-oriented attitude and should not be considered as something fixed but rather as evolving and context-sensitive.

The composition *'Drain'* (2021) is a recent practical example. The piece explores synthetic sound streams developed in-time through transformational pipelines and various behaviours for activating those. The pipelines are pre-configured and then activated. The live element evolves around switching, processing and modifying the pipelines while they run.

Many processes are continuously running, even in the background and later receive focus by being switched on (or off). Such manual actions often introduce new events and other processes and so the piece unfolds. Somehow the idea is of an operator connecting to an autonomous process and interpreting it while it runs.

Recorded examples from *OF* and the piece can be found here:

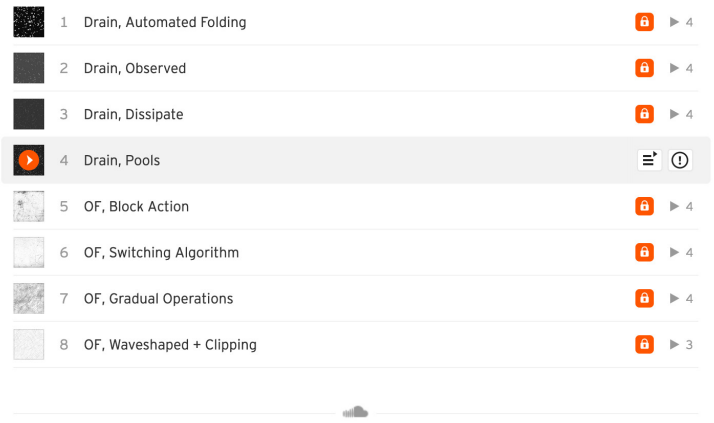

Fig. 4. *OF* Sound Examples. https://soundcloud.com/bjarni/sets/research

5. Conclusion

This article has presented an approach to synthesis and composition based on sound streams, operation pipelines and data processing. Pipelines have been defined as bundles of synthesis and transformation and how they bring about possibilities for process and development has been a central concern. Operational spaces, process configuration, automation, direct access, liveness, incompleteness and unpredictability have all been discussed as possible dimensions that augment a creative approach to sound composition. A software framework, *OF*, was introduced that implements many of concepts that have been covered and its software architecture was demonstrated.

The OF components should be extended in various ways to better address the framework purpose. The synthesis modules are similar in nature and broadening their scope will allow for enacting the operations in more ways. The behaviours are also rather narrow and given the tight relationship between the synthesis and processing, adding more experimental behaviours is needed in order to fully explore the pipeline possibilities. Finally, the transformations (and algorithms that manipulate them) offer great potential for further development. Processing based on self-analysis and time-domain distortions seems to be a fruitful area to explore. In addition to the component improvements, an investigation is needed for how interaction occurs within the framework, for example how pipelines are made to start, stop or change. Concerns that further question the boundaries of sound, treatment and algorithm.

References

Chaitin, Gregory.
2006. *Meta Math! The Quest for Omega*. New York, NY: Vintage Books.

Chun, Wendy Hui Kyong.
2011. *Programmed Visions : Software and Memory*. Cambridge, MA: The MIT Press.

Eigenfeldt, Arne.
2014. *Generating Structure – Towards Large-Scale Formal Generation*. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 10(1).

Ernst, Wolfgang.
2013. *Digital Memory and the Archive*. Edited by Parikka Jussi. University of Minnesota Press.

Frabetti, Federica.
2015. *Software Theory: A Cultural and Philosophical Study*, London: Rowman and Littlefield International.

Holmes, Thom.
2015. *Electronic and Experimental Music: Technology, Music, and Culture*. London. Routledge.

Kanervisto, Scheller and Hautamäki.
2020. *Action Space Shaping in Deep Reinforcement Learning*. EEE Conference on Games 2020. arXiv:2004.00980.

Kelly, Kevin.
2010. *What Technology Wants*. New York, NY: Viking Press.

Koenig, Gottfried Michael.
1965. The Second Phase of Electronic Music. http://koenigproject.nl/download-pdfs/ (accessed 10 Jan 2021).

Koenig, Gottfried Michael.
1987. *Genesis of Form in Technically Conditioned Environments*. Interface 16, no.3. 165-175.

Law, John.
2004. *After Method*. New York, NY: Routledge.

McCartney, James.
2002. *Rethinking the Computer Music Language: SuperCollider*. In Computer Music Journal, 26(4), 61-68. Cambridge, MA: MIT Press.

Murphie, Andrew.
2018. *Convolving Signals Thinking the performance of computational processes*. Performance Paradigm, (9). http://www.performanceparadigm.net/index.php/journal/article/view/135 (accessed 11 Dec 2020).

Parikka, Jussi.
2018. *Ernst on Time-Critical Media: A mini-interview*. Machinology. https://jussiparikka.net/2013/03/18/ernst-on-microtemporality-a-mini-interview/ (accessed 22 Jan 2021).

Parisi, Luciana and Stamatia Portanova. 2011. *Soft thought (in architecture and choreography)*. Computational Culture 1 (November 2011). http://computationalculture.net/soft-thought/. (accessed 18 Dec 2020).

Rohrhuber, Julian, Alberto de Campo Alberto, and Renate Wieser. 2005. *Algorithms Today: Notes On Language Design For Just In Time Programming.* International Computer Music Conference. Volume 2005.

Soon, Winnie. 2016. *Executing Liveness: An Examination of the Live Dimension of Code Interactions in Software (Art) Practice*. PhD dissertation, Aarhus University. http://siusoon.net/home/me/doc/soon_PhD_FINAL.pdf (accessed 24 Nov 2020)

Suchman, Lucy. 2006. *Human-Machine Reconfigurations: Plans and Situated Actions* (2nd ed., Learning in Doing: Social, Cognitive and Computational Perspectives). Cambridge: Cambridge University Press.

Suchman, Lucy. 2012. Configuration. In Celia Lury and Nina Wakeford (Eds.), *Inventive Methods: The happening of the social*. London. Routledge. 48–60.

Uustalu, Tarmo, and Varmo Vene. 2005. *The Essence of Dataflow Programming*. Central European Functional Programming School, First Summer School, CEFP 2005, Budapest, Hungary.

Williams, Raymond. 2003. *Television : Technology and Cultural Form*. New York :Schocken Books.

Yann Orlarey, Dominique Fober, and Stéphane Letz. 2009. Faust : *r*. Editions Delatour France. New Computational Paradigms For Computer Music, 65-96. Hal-02159014.