LIMITS OF CONTROL

Hanns Holger Rutz
Interdisciplinary Centre for Computer Music Research (ICCMR) – University of Plymouth
hanns.rutz@plymouth.ac.uk

ABSTRACT

We are analysing the implications of music composition through programming, in particular the possibilities and limitations of tracing the composition process through computer artefacts. The analysis is attached to the case study of a sound installation. This work was realised using a new programming system which is briefly introduced. Through these observations we are probing and adjusting a model of the composition process which draws ideas from systems theory, the experimental system of differential reproduction, and deconstructionism.

1. INTRODUCTION

> «But words are still the principal instruments of control»*

The term "computer music" can be used to denote a musical praxis and musical research that reflect the «profound influence of computer science on music»[1], and thus they substantially depend on the medial implications of the computer. The most crucial implication, as Loy and Curtis point out in a 1985 survey, is the formalisation of concepts through the use of programming languages. The involvement of programming languages distinguishes this form of computer music from other forms in which the musician takes the role of the user of readily available applications «in which inputs of a simple structure produce effects (such as outputs) desired by the user»[1]. Coincidentally, the sound transformation plug-ins offered by commercial music software are often called "effects"—they offer fully prescribed, and often standardised, tools to achieve well known goals.

The term 'goal' was early introduced in cybernetics (e.g. [2]), and is linked to the concept of control which is a regulatory mechanism to direct the system towards a goal. The feedback from the system's output to the regulator's input can be replaced by the human being listening to the sound produced by an "effect", and the regulator is the knob in the interface which also goes by the name of a "controller". By rotating the knob, the imaginated sound—the goal—is incrementally approached.

Obviously, the application of these terms seems easy in a rather mechanical case like this, but how is goal directedness translated when using a musical programming language, and how is control exercised with words instead of knobs? As Rosenblueth et al. carefully put it [2], purposefulness of a behaviour is an attribution resulting from an *interpretation*. In other words, an observer is needed to make this attribution.

2. DISSEMINATION

> «A basic impasse of all control machines is this: Control needs time in which to exercise control»

When using systems thinking as an approach it is important to remind oneself of its abstract nature. Although some authors give systems an ontological status and then deduce (observer specific) symbolic representations in the form of models as homomorphic mappings of the real world systems [3], we follow Checkland here in that systems themselves are just abstractions and epistemological tools [4]. One of the main abstractions, perhaps the most fundamental one, is that of the boundary between a system and its environment (cf. [5]): Where is the boundary between the computer music composer taking the role of a programmer and any other programmer (where is the boundary between programming and composing, between a programme and a composition)? Where does the process of composition begin and where does it end?

Light is shed on these questions with the help of a case study: ‹Dissemination›[1] is the title of an audio-visual installation by Hanns Holger Rutz and Nayarí Castillo, in which both media create a space by relying on and reflecting upon each other. It consists of horizontally and vertically suspended glass panels functioning both as a body for sound resonance and diffusion—through the attachment of sound transducers—as well as specimen holders carrying petri dishes filled with seeds (see fig. 1). Several tableaux are generated which unwind the temporal development of the generative sound composition in space. Flying seeds, which due to their natural features can be dispersed by wind, symbolise motion, traveling and migration. The act of dissemination is interpreted as an act of re-writing, leaving traces, producing movement instead of a fulfilment, strategy instead of finality.

Without going into many details, the concept of the sound installation was to create a generative real time composition with different temporal layers, some of which are

*All epigraphs from William S. Burroughs' essay *The Limits of Control*, but the last which is from the Jim Jarmusch movie of the same title.

[1] http://www.sciss.de/texts/ins_dissemination.html

Figure 1: Photos from ‹Dissemination›. On the left, the site (Gallery ESC Graz) is seen with the suspended glass panels pervading the space. The daylight is filtered with yellow gels. The right photo shows the pairing of a horizontal and vertical plate, the vertical plate being excited by the black sound transducer, the horizontal plate holding an ensemble of petri dishes. The middle photo shows a close up of a petri dish holding flying seeds.

cyclic and others which span an ongoing thread over the duration of the exhibition. Although the setup had been conceptualised back in September 2009, it was only in August of the following year that the sound composition was carried out. Within this period a framework for the description and connection of sound processes had been developed, and this was the second project in which it was put to practice.

Thus, when looking at the programming, an arbitrary line must be drawn between the preparatory work and the composition in the narrow sense. Since both the framework and the composition have been managed using a versioning system, we are able to look at the development over time. The repository containing the actual composition was opened in August 2010 and is visualised in figure 2. The lines of code written in the Scala programming language have been manually categorised as belonging either to the technical infrastructure of the system or carrying actual musical meaning. The piece was exhibited twice, with the premiere taking place on September 18, and the second exhibition beginning on October 20.

It can be seen that in the beginning more time is spent with the programming of the infrastructure than the actual composition. Since the framework was rather new, various adaptations were required to realise the ideas for the project. Infrastructure code generation decreases to nearly zero towards the end of the first composition cycle with another final spike due to preparing the work for autonomous operation during the exhibition [2].

The other distinction made here is between newly written code and code derived from previous projects or from older stages of the same project. This follows from a recursion model of the composition process developed in [6] which proposes that this process is driven by material injected from outside the system as well as (re-)transformation of material already inside the system. Finding this pattern in manifest condensations—computer artefacts such as the code—could indicate that something similar is happening in the psychic system of the composer. This is confirmed by other studies, for example Collins in his case study of a composer working on instrumental staff based music identifies a combination of both linear and recursive motions in the development of the piece [7].

Infrastructure code copied from previous projects indicates lack of modularisation, but can be mostly explained by the time pressure factor in the realisation of a work[3]. On the other hand, the adaptation of existing musical code mostly amounts to sound synthesis and transformation instruments which are reused. In the second half, self referential composition code—code which is derived from code within the same project—begins to increase and finally amounts to roughly half of the additionally produced code volume. Musically, this can be interpreted as variation: More sound processes are introduced which share structural similarities with previously created processes but are then differentiated—for example by using other sound files, other parametrisation, probabilities, spatialisation etc. Figure 2 therefore seems to support the model of an increasingly recursive behaviour of the composition process.

[2] A bug was found that caused the system to become unstable after a couple of hours running, so several measures had to be taken to work around it.

[3] Copying code in the short term is a much faster measure than refactoring into reusable external modules which pays off in the long term.

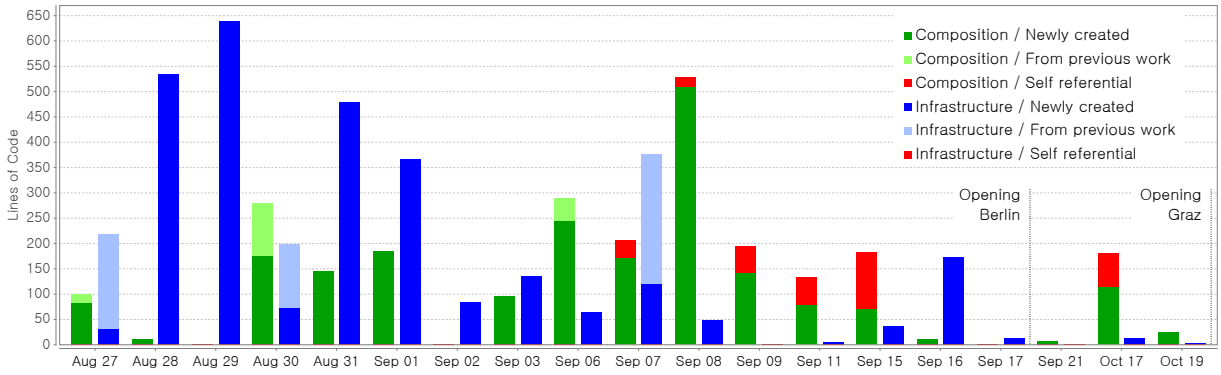

Figure 2: Code commits to the GIT repositories of the composition and frameworks it depends on. Line count includes lines created, lines edited and lines deleted. Multiple commits per day are integrated. Code carrying musical meaning in the narrow sense is shown in green and contrasted with code used to build the infrastructure, used for debugging, and so forth, which is shown in blue. Bars are split to distinguish newly created parts from parts derived from previous work.

3. PERMEABLE BOUNDARIES

«Concession is another control bind»

The ambitious goal of "strategy instead of finality" was to reflect the recursive model of human activity in the generative structure of the piece itself. After all, «Programming is not about doing; it's about causing the doing»[8]. Following the technical agnosticity of a frameworks's application programming interface (API) as to whether a call is issued by direct human action or deferred action in the form of an algorithm (the "caused doing"), both direct and indirect actions can be collapsed in the notion of an abstract writing process as outlined by Derrida [9] (cf. [6]). A data structure which would allow the generative parts of the piece to inscribe their actions into the piece itself was proposed in [10], but unfortunately had not been ready for production by the time ‹Dissemination› was created.

As a result, we can observe how the practice of the art production ignores "impossibilities" imposed by the systems at hand and transcends their boundaries: The idea of a persistent trace was upheld by another type of observer: One of the main algorithms, named *Plates* (see figure 3), instead of being able to read traces left inside a data structure, uses audio signal feature extraction to gain insight into the instantaneous state of the composition. It tries to maintain a sort of energy balance, reacting by generation of new material or withdrawal of current material. The production of new material is accomplished by recording parts of the installation's output signal and feeding it into a set of signal transformation processes, eventually re-injecting the transformed material into the piece. The material thus generated remains on the hard disk of the installation computer and leaves a persistent trace over the period of the exhibition. This pool of material is periodically thinned out so that the second function of memory, forgetting, is included.

However, the assumption that, had the persistence layer been developed to planned extent, the whole composition would have been confined to the boundaries of this layer and hence be fully traceable, is illusionary. This is because our framework is a designed system and as such has a prescribed model and a prescribed purpose (allowing a traceable form of composition), and it would consequently fall into the category of a "taciturn system" according to Pask [11]. On the other hand, we offer the composer the programming language which falls into the second category of "language oriented systems"—these exhibit contingency since the composer can change her mind and instruct it to do other things than before. The composer-observer resides inside the boundaries of the language oriented system, so to say, but outside the boundaries of the technological observer, the domain specific language (DSL), the framework subset of the language.

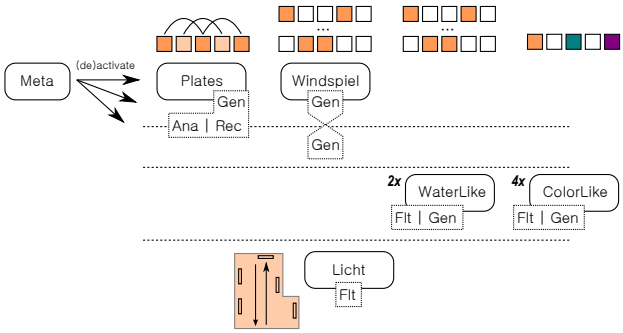

Figure 3: Schema of the constituent sound processes in ‹Dissemination›. Vertical lines partition sound layers where lower layers can filter or shadow upper layers. Orange squares indicate spatialisation modes, corresponding to the five channel diffusion in the first exhibition, and with the mode for *Licht* showing the floor plan in Berlin. Each process provides different components which can act as sound generators, filters or analysing and recording stages.

This is illustrated in the top most diagram of figure 4. The framework's language contains a notion of sound processes which are described in terms of input and output signals, control parameters, resources such as sound buffers and sound files, as well as a function which generates a graph of unit generators responsible for the actual sound analysis, transformation and synthesis, using the SUPERCOLLIDER server. A simplified example from the *Plates*

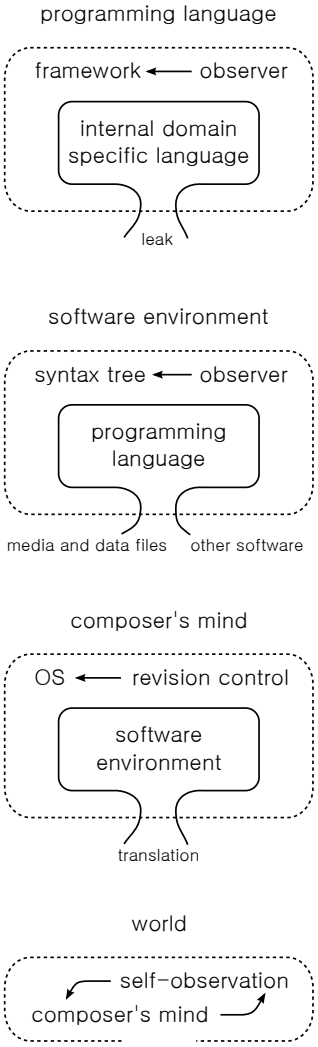

Figure 4: The idea of a composition system as observing (tracing) the composer's traces, and how it is undermined.

algorithm is shown in figure 5.

One of the problems—defining a stable reference point in the sound stream which could be analysed despite all processes appearing and disappearing dynamically—was solved within the framework: Special collector nodes are created which are maintained throughout the installation. The two collectors *pColl1* and *pColl2* correspond to the two upmost dotted horizontal lines in figure 3. This allows for example another process, *Windspiel*, to decide whether it wants its sound output be picked up by *Plates'* analysis (using level *pColl1*) or not (using level *pColl2*).

Notwithstanding, the real problem is the highlighted line in figure 5: Once every second the smoothed out analysis data is sampled and a subroutine *newAnalysis* called, and this is were the code escapes the intended traceability of the persistence framework. One might well be able to catch and represent the graph function within this framework, but the client code looking at the analysis data and making decisions about starting or stopping sound processes leaks outside. Code and data fall apart, and there are only two possible solutions to this dilemma: Either the approach of developing the composition within a DSL em-

```
val fColl = filter("+") { graph { in=>in }}
val pColl1 = fColl.make
val pColl2 = fColl.make
pColl1 ~> pColl2

val pAna = (diff("ana") { graph { in =>
   val bufID   = bufEmpty(1024).id
   val chain   = FFT(bufID, Mix(in))
   val loud    = Loudness.kr(chain)
   val centr   = SpecPcile.kr(chain)
   val flat    = SpecFlatness.kr(chain)
   val compound= List(loud, centr, flat)
   val smooth  = Lag.kr(compound, 10)
   1.react(smooth)(plate.newAnalysis(_))
}}).make

pColl1 ~> pAna
pColl1 ~> pRec
pAna.play
```

Figure 5: Example of an analysis sound process code from the *Plates* component. Also shown is the creation and connection of collector nodes which establish stable references in the conceived level structure.

bedded in a general purpose language is given up in favour of a more rigid environment—perhaps one in which it is still possible to write functions for the synthesis graph, but such that they are embedded in hidden glue code that prevents the composer from escaping a given scope—, or we move the observer outside the language, dealing not directly with the framework anymore but the abstract syntax tree (AST) representation of the language. This latter solution corresponds with the second diagram in figure 4. In the argumentation of soft systems approaches—which seem adequate when dealing with language oriented systems—, any system is open by definition, since it effects and is affected by its environment. The "leak" can be recursively closed: «Any open system can always be reframed as closed by expanding the system boundaries to include its environment.»[12].

Figure 4 shows this recursive closure with the software observer eventually being replaced by the composer observing herself, meaning «the hermeneutic circle of interpretation-action, on which all human activity is based.»[13] Each stage bears new problems. In the AST analysis, it becomes unfeasible to trace musical intentions in a fine grained way, it also places a serious technical hurdle by the need to represent multiple versions of code fragments within the same class loader. On the other hand, the composition may easily integrate data which is not part of the host language. In the diagram, this is indicated by "media and data files" and "other software". In ‹Dissemination›, several sound files were incorporated and meta files in the form of segmentation data for particular sounds. Furthermore, a separate software FSCAPE was used from various processes to render sound transformations while the installation is running. The next logical boundary would thus be a file revision control system that could observe all data involved. Again, if the fine levels of granularity shall not be lost, this must be combined with observers of the previous levels. The third stage definitely escapes the ability to monitor this purely in software, as the actual piece in-

volves decision making in the composer's head which may not be directly projected onto the software. A simple reminder is that this is an audio-visual installation, so the composition of the physical components, the arrangement and conditioning of the space are not covered, neither are notes taken in sketchbooks and so forth.

4. SOLUTION SPACES

> «All control systems try to make control as tight as possible, but at the same time, if they succeeded completely, there would be nothing left to control»

It can only be concluded that any observation system is limited and should be modest and candid about its limitations. It goes without saying that this does not imply that such systems are not useful, but a clarification of the purpose of this observation is required. It is not about the establishing of a trace as an empirical fact pointing back to an arche-trace, as there is no such thing as an arche-trace (cf. [9]). The transportation of code from previous projects and frameworks into a "new" project has already indicated that, and figure 6 underlines it even further, as it shows how DSP processes, sound files and concepts from previous projects form an important part of the piece. The DSP processes depicted have been developed by the author over a period of ten years, and some of them are highly idiosyncratic and unfold their potential when connected with the other processes, so they are considered essential compositional elements and not just "effects" (infrastructure). The sound files, too, span almost a decade. Some of them had been used in previous projects, some had never been used in a piece, others have been used in a completely different manner before. Finally ‹Kalligraphie› and ‹Amplifikation› are two sound installations which had a profound influence on this new one: They established the idea of a sound mobile constructed from semi-independent processes and driven by a dedicated meta process, the physical setup of glass plates, transducers and daylight colourisation, and even specific processes such as *Licht*—a process which takes the frequency response of the glass plates and constructs an inverse filter, imposing a moving gesture of immateriality onto the "unfiltered" sounds—which was enhanced from the original version in ‹Amplifikation›.

The boundary between this piece and all previous pieces seems to correlate with the performances and exhibitions of the respective pieces, but it is just as conventional as any system boundary and may not help in the analysis of the composition process.

Observation as drawing-a-distinction is not so much different from writing, especially when the observer is located within the system, as this type of observer not only describes the system, but is also a relation, a determining component of the system, and the very description of the system dynamically changes the subject of description [13]. Thus, when we construct a computer music composition system with an observable, traceable data structure, we aim at changing the notion of composition altogether, and by doing that we hope to contribute to a form of computer music which is truly depending on the computer as medium. By offering the trace of the composition process as an access point to recursive transformation, we are not supporting the closure of a problem space but the opening of a solution space (cf. [7]).

This is best illustrated with a classic concept from control theory, Ashby's law of requisite variety [14]. It states that goal-directed systems pursue a state of equilibrium, and that in order to shield them from disturbances from the environment—which would deter the system from its goal—the regulator must provide counter actions which have a variety that is at least as great as the variety of disturbances. The function of the regulator thus is to minimise variety in the output. Obviously this does not resonate with language oriented systems which deal with communication and wish to maximise the variety of possible expressions. These two types of systems do not contradict each other, as one can easily specify expressiveness as the goal of a system, so that the regulator would minimise disturbances which inhibit expressiveness. Heylighen and Joslyn portrait this duality as interaction between two systems with complementary goals [3].

Another way to see this complementarity is the experimental system described by Rheinberger which is based on differential reproduction [15]. It is the modus operandi of the scientific system but can be equally applied to arts. Differential reproduction is a process of repetition and not replication which means that the goal of identity is substituted for one of variation. In each iteration of an experiment a form of cohesion must be established that allows it to be compared to the previous experiment, but at the same time the system must be open for disturbances from the environment so that unforeseen things can happen. Unforeseen things *should* happen because the scientific system aims at creating new knowledge, similarly the arts aim at creating novel experiences. While technology serves as a background layer and is a form of answering-machine—problem closure, Pask's taciturn system—, the epistemic object is a question-generating machine—opening the solution space, the language oriented system.

5. CONCLUSIONS

> «. . . musical instruments . . . still sound even when not being played, have a memory; every note that has been played once with them, still there, inside, is resonating in the molecules of the wood»

We have identified computer music (a better term would be computer sound art) as a form which is intrinsically inspired by the computer as medium and not just a workbench with tools. Systems theory and cybernetics were evaluated for the usefulness and applicability of their concepts regarding the computer music composition process. The distinction of language oriented from taciturn systems and the identification of the former with computer music programming languages opened the discussion of specific terms such as 'goal', 'boundary' and 'observer'.

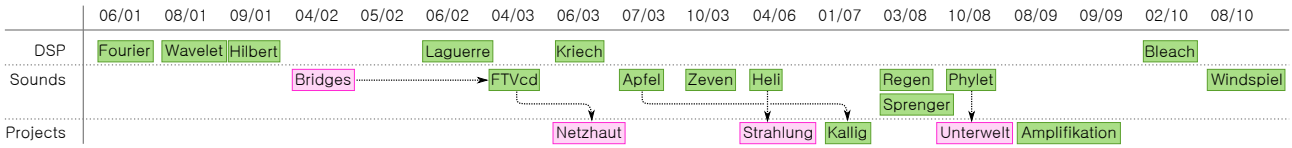

Figure 6: Selection of external references for the piece and their establishment over time: Special digital signal processing algorithms, sound files, and previous works. Pink colour items do not have a direct link with ‹Dissemination›, arrows indicate dependencies between these references.

This composition approach was exemplified by an audio-visual installation work. Using data from a versioning system we found indications that the composition process can be modelled as a recursive system which takes input both from new ideas but equally from previous projects and then increasingly relies on transformed material fed back from previous iterations within the same process. Despite being reinforced by other studies of composer observation, further research is needed to turn these indications into generalised claims about computer music composition.

While the software framework used was not yet capable of tracing the writing of the composition, strategies have been employed to overcome this limitation. A memory was realised as recorded and rendered sound files and self observation was realised using audio feature extraction. Elaborating on the hypothetical implementation of the persistent observing system, it was found that any such observation system is inherently limited in scope and that it lies in the nature of the programming language approach—maximising expressiveness and communication with its environment—that system bounds become poriferous.

Hitting the limits of objective or "unconcerned" observation, we introduced a turn in the purpose of observation. A "concerned" observation is actively transforming the compositional process by facilitating a recursive self-reflection and focusing on its creative potential rather than an archival idea of establishing the origins or roots of a composition. This self-reflection can be interpreted as a strategy for maintaining and increasing a solution space—for example by allowing the composer to establish new connections and relations between existing elements of the composition cutting across different versions in time (the ‘meld’ operation of confluent persistence)—while at the same time allowing for a cohesion between past and future states of a composition. The oscillation between cohesion and openness to disturbance from the environment is what classifies this process as a differential reproduction, supporting the emergence of new forms of expressions and thereby supporting a core interest of the arts.

6. REFERENCES

[1] G. Loy and C. Abbott, “Programming Languages for Computer Music Synthesis, Performance, and Composition,” *ACM Computing Surveys (CSUR)*, vol. 17, no. 2, pp. 235–265, 1985.

[2] A. Rosenblueth, N. Wiener, and J. Bigelow, “Behavior, Purpose and Teleology,” *Philosophy of Science*, pp. 18–24, 1943.

[3] F. Heylighen and C. Joslyn, “Cybernetics and Second Order Cybernetics,” in *Encyclopedia of Physical Science and Technology*, R. A. Meyers, Ed. Academic Press, 2001, vol. 4, pp. 155–170.

[4] P. Checkland, “Systems Thinking,” in *Rethinking Management Information Systems*, W. L. Currie and R. Galliers, Eds. New York: Oxford University Press, 1999, pp. 45–56.

[5] R. L. Flood, “Unleashing the ‘Open System’ Metaphor,” *Systemic Practice and Action Research*, vol. 1, no. 3, pp. 313–318, 1988.

[6] H. H. Rutz, E. Miranda, and G. Eckel, “Reproducibility and Random Access in Sound Synthesis,” in *Proceedings of the International Computer Music Conference*, 2011.

[7] D. Collins, “A synthesis process model of creative thinking in music composition,” *Psychology of Music*, vol. 33, no. 2, pp. 193–216, 2005.

[8] D. Harel, “Can Programming Be Liberated, Period?” *Computer (IEEE)*, vol. 41, no. 1, pp. 28–37, 2008.

[9] J. Derrida, *Of Grammatology*. Baltimore: Johns Hopkins University Press, 1997 (1967), trans. Gayatri Chakravorty Spivak.

[10] H. H. Rutz, E. Miranda, and G. Eckel, “On the Traceability of the Compositional Process,” in *Proceedings of the Sound an Music Computing Conference*, 2010, pp. 38:1–38:7.

[11] G. Pask, “The meaning of cybernetics in the behavioural sciences (The cybernetics of behaviour and cognition; extending the meaning of ‘goal’),” *Progress of Cybernetics*, vol. 1, pp. 15–44, 1969.

[12] A. Ryan, “What is a Systems Approach?” *Arxiv preprint arXiv:0809.1698*, 2008.

[13] F. J. Varela, “Autonomy and Autopoiesis,” in *Self-organizing Systems: An Interdisciplinary Approach*, G. Roth and H. Schwegler, Eds. Frankfurt and New York: Campus Verlag, 1981, pp. 14–23.

[14] W. R. Ashby, *An introduction to Cybernetics*. London: Chapman & Hall, 1956.

[15] H.-J. Rheinberger, *Toward a history of epistemic things: Synthesizing proteins in the test tube*. Palo Alto: Stanford University Press, 1997.