Unconventional Programming: Programming Non-programmable Systems

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik

der FRIEDRICH-SCHILLER-UNIVERSITÄT JENA

von Dipl. Bioinf. Gerd Grünert

geb. am 30.10.1981 in München

Gutachter:

1. Prof. Dr. Peter Dittrich, Jena
2. Dr. Klaus-Peter Zauner, Southampton
3. Prof. Dr. Wolfgang Weigand, Jena

Tag der öffentlichen Verteidigung: 09.12.2016

Summary

Unconventional and natural computing research offers controlled information modification processes in uncommon media, for example on the molecular scale or in bacteria colonies. Promising aspects of such systems are often the non-linear behavior and the high connectivity of the involved information processing components in analogy to neurons in the nervous system. Unfortunately, such properties make the system behavior hard to understand, hard to predict and thus also hard to program with common engineering principles like modularization and composition, leading to the term of non-programmable systems. In contrast to many unconventional computing works that are often focused on finding novel computing substrates and potential applications, unconventional programming approaches for such systems are the theme of this thesis: How can new programming concepts open up new perspectives for unconventional but hopefully also for traditional, digital computing systems?

Mostly based on a model of artificial wet chemical neurons, different unconventional programming approaches from evolutionary algorithms, information theory, self-organization and self-assembly are explored. A particular emphasis is given on the problem of symbol encodings: Often there are multiple or even an unlimited number of possibilities to encode information in the phase space of dynamical systems, e.g. spike frequencies or population coding in neural networks. But different encodings will probably be differently useful, dependent on the system properties, the information transformation task and the desired connectivity to other systems. Hence methods are investigated that can evaluate, analyse as well as identify suitable symbol encoding schemes.

Deutsche Zusammenfassung

Die Forschung aus dem Bereich der unkonventionellen und natürlichen Informationsverarbeitungssysteme verspricht kontrollierbare Rechenprozesse in ungewöhnlichen Medien zu realisieren, zum Beispiel auf der molekularen Ebene oder in Bakterienkolonien. Vielversprechende Eigenschaften dieser Systeme sind das nichtlineare Verhalten und der hohe Verknüpfungsgrad der beteiligten Komponenten in Analogie zu Neuronen im Gehirn. Da aber Programmierung meist auf Prinzipien wie Modularisierung, Kapselung und Vorhersagbarkeit beruht sind diese Systeme oft schwer- bzw. unprogrammierbar. Im Gegensatz zu vielen Arbeiten über unkonventionelle Rechensysteme soll in dieser Arbeit aber nicht hauptsächlich nach neuen rechnenden Systemen und Anwendungen dieser gesucht werden. Stattdessen konzentriert sich diese Dissertation auf unkonventionelle Programmieransätze, die sowohl für unkonventionelle Computer als auch für herkommliche digitale Rechner neue Perspektiven eröffnen sollen.

Hauptsächlich in Bezug auf ein Modell künstlicher chemischer Neuronen werden Ansätze für unkonventionelle Programmierverfahren, basierend auf Evolutionären Algorithmen, Informationstheorie und Selbstorganisation bis hin zur Selbstassemblierung untersucht. Ein spezielles Augenmerk liegt dabei auf dem Problem der Symbolkodierung: Oft gibt es mehrere oder sogar unendlich viele Möglichkeiten, Informationen in den Zuständen eines komplexen dynamischen Systems zu kodieren. In Neuronalen Netzen gibt es unter anderem die Spikefrequenz aber auch Populationskodes. In Abhängigkeit von den weiteren Eigenschaften des Systems, beispielsweise von der Informationsverarbeitungsaufgabe und dem gewünschten Eingabe-Ausgabeverhalten

dürften sich verschiedene Kodierungen als unterschiedlich nützlich erweisen. Daher werden hier Methoden betrachtet um die verschiedene Symbolkodierungmethoden zu evaluieren, zu analysieren und um nach neuen, geeigneten Kodierungen zu suchen.

Contents

Chapter 1

Introduction

1.1 Motivation

"Programming non-programmable systems" might sound like "flying pigs" and "three headed monkeys", like complete nonsense. Similarly, "programming" a birch tree or a tornado sounds absurd. But while no two birch trees and no two tornadoes look alike, there is a fundamental difference: The ensemble of tornado shapes is constrained only by the laws of physics and the local environment. But the ensemble of birch trees that could grow from a seed is additionally constrained by its genetic information. So if the range of possible shapes for those systems should be modified, altering the birch tree should still be more feasible than modifying the tornado shapes, because there is arbitrariness in the genomes of the former systems. Systems like tornadoes that cannot be modified other than by changing the laws of physics are not considered in this thesis and would neither be called programmable nor non-programmable. Nonetheless, the notion of "programming" still sounds absurd when considering genomes or brains, because we lack appropriate engineering principles to modify the genetic material or the brain connectivity in a predictable way. Hence these systems are called non-programmable as compared to digital computers and mechanic systems that are accessible to engineering principles. In this thesis, approaches towards opening also

non-programmable complex systems to means of guided modification, i.e. to programming, are explored.

Conrad argues that there is a trade off principle between (effective) programmability, computational efficiency and evolutionary adaptability [Con89, Con95b, Con95a]. This implies a difference between programmable and non-programmable systems. Examples for such non-programmable systems are biological organisms and the human brain with its huge number of neurons and synaptic connections, the simulation of which would currently require huge supercomputers with thousands of CPUs [Mar06]. To see the difference in programmability between brains and electronic computers, the definition of *programs* and *effective algorithms* are instructive:

A *program* is *"an expression of a computational method in a computer language"* [Knu73], where a "computational method" is used equivalently with the word *algorithm*. So a program is an algorithm that is translated into a formalization that can be understood by a machine. Probably one of the most central concepts in computer science, the *algorithm*, dates back more than 2000 years, as for example Euclid's Algorithm, originating earlier than 300 BC [Knu69]. Still, different notions instead of "algorithm" were used before the 1950s [Knu73]. The algorithm is further understood as a *"recipe, process, method, technique, procedure [or] routine"* with the following additional properties [Knu73, Rog87]:

discreteness The algorithm is composed from discrete instructions that can be followed step-by-step.

finiteness Both in its run time and in its description, the algorithm is to be finite, i.e., the algorithm is supposed to terminate after a finite number of steps and it is further supposed to be described by a finite number of instructions.

definiteness For each instruction of the algorithm, it ought to be indisputably clear what this instructions means and which effects it has.

effectiveness Each of the instructions of the algorithm has to be practically, almost mechanically, executable.

input / output Zero or more inputs can be supplied to the algorithm and one or more specified values in the system will be interpreted as the output of the algorithm.

Comparing biological cells or brains to the definition of algorithms given above, these systems typically do not follow such a clear "procedure" with several steps. Rather, stochastic noise commonly leads to variations in repetitions of the same information processing task. Also the purpose and effect of individual system components or of their interactions is not clearly defined. For example, in cells, the response of a set of signal transduction molecules to an external stimulus is a noise affected process and thus the result will vary every time the operation is being observed [Gil77]. Furthermore, there does not exist a precise algorithm for the signal processing taking place in our brains. At most, such an algorithm might be indirectly defined through the complex interactions of its components like different brain areas, tissues, their connectivity and the complex developmental process. Hence, recalling that programming means formalizing an algorithm, these natural information processing systems are not effectively programmable in Conrad's sense [Con95a], because there is typically no obvious algorithm of the definition given above.

But nevertheless, also biological information processing in genomes, cells, tissues, brains and whole organisms is highly organized. This makes it possible to find and analyze some functional patterns and principles [Bou02, DL06, Ibr08]. Apparently these systems are not purely self-organized by the physical properties of the systems like a Bénard cell [Bod00], a tornado or the Belousov-Zhabotinsky reaction [Zai70]. Instead, there is a lot of organization, parameterization and arbitrariness involved in natural information processing systems [Bar08, Gör11b, Gör13, Ibr13]. As Suzuki states, the genetic information shapes, orchestrates, controls, constrains, guides, harnesses or instructs the complex system of interactions [Suz13]. Yet clearly, neither every synapse and neuron of the brain nor every single signal receptor or kinase of a cell is described individually. Instead, the description in the genome is given in a compressed form, indirectly and in coordination with the environmental influences, in the system's own dynamics and in the developmental processes [Ste12]. Thus, the

"genetic program" is not an implementation of a discrete, finite, definite, deterministic and effective algorithm as defined above. But it shares the property of guiding the information processing that happens in the system. Nonetheless, while we would easily associate programming with computers, the notion of *programming* for most people has an uncomfortable tinge to it when it is applied to evolution, brains or other biological systems. Hence, the word *programming* will mostly be used in the sense of coarsely specifying and guiding the form or function of an information processing system here. This broad term includes both, using a formal algorithm that classical electronic computers are programmed with but also manipulating a dynamical system into a particular behavior, which can then be exploited for computation.

In this thesis, novel and unconventional programming paradigms are identified that could be used to conceive, develop and specify "programs"; potentially for programmable as well as for non-programmable information processing systems. As explored in the later chapters, these paradigms employ among others co-evolution schemes in evolutionary algorithms, information theory as fitness functions as well as self-organization and self-assembly processes as guiding principles. A further explanation, why new and unconventional approaches are useful and what we understand under the term "programming" will then be given in Section 1.2. One reason for investigating novel programming paradigms is the hope to harness the computational capacity of unconventional computing systems [Ada01, Ada06, Mat07, Teu08, Ste12] more efficiently: their low energy consumption, their efficiency in other problem domains and the possibility to compute in different media. Additionally, reusing these novel computing paradigms for generating problem solutions in classical electronic computers might be helpful to produce more efficient and more fault tolerant code. In Section 1.3, a more explicit formulation for such information processing systems, i.e., for the term "computation" will be given.

When looking at unconventional [Ada01, Ada06, Mat07, Teu08, Ste12] and natural computing systems [dC07, DC11] or biological information processing systems, these systems can sometimes be transformed into formal computing systems like Turing

Machines [Tur36, Pău06, Gru11b]. Yet for naturally occurring information processing systems like the brain, often we will not be able to find finite, definite and effective descriptions for the processes that happen there, at least not if the full computational potential of the systems is to be exploited. Thus, because unconventional computing systems are often similarly resistant to formal programming, searching for non-standard programming paradigms in these fields seems promising. In unconventional computing research, so far a lot of work has been focused on specialized applications and in particular on novel computing substrates. But also unconventional programming paradigms are investigated, as reviewed briefly in Section 1.4 and explained in more detail in [Ban05, Ste12].

1.2 Conventional Programming

In classical programming, the human programmer understands the capabilities and the potential of the computing environment and almost like an architect designs or constructs a problem solutions from the building blocks the programming language offers. This concept is satirized in the context of electronic circuit diagrams in Figure 1.1. For programming languages, such building blocks are mostly elementary data manipulation operations like simple arithmetics or control structures like loops, conditions, recursions or subroutines. Wing called this kind of human cognitive effort "computational thinking": *"it is not trying to get humans to think like computers. Computers are dull and boring; humans are clever and imaginative. We humans make computers exciting."*[Win06]

A typical aspect of programming seems to be the prediction of the macroscopic effects of microscopic program parts and from this, the construction of problem solutions [Con95b, Mat07] as formulated by Zauner: *"Programming is here equated with an engineering approach in which mental conception precedes physical creation [...]. It necessitates the possibility for the programmer to anticipate the actions of the available elementary operations. Only if the function of the elementary operations can be foreseen by the programmer can a desired input-output map be implemented by*

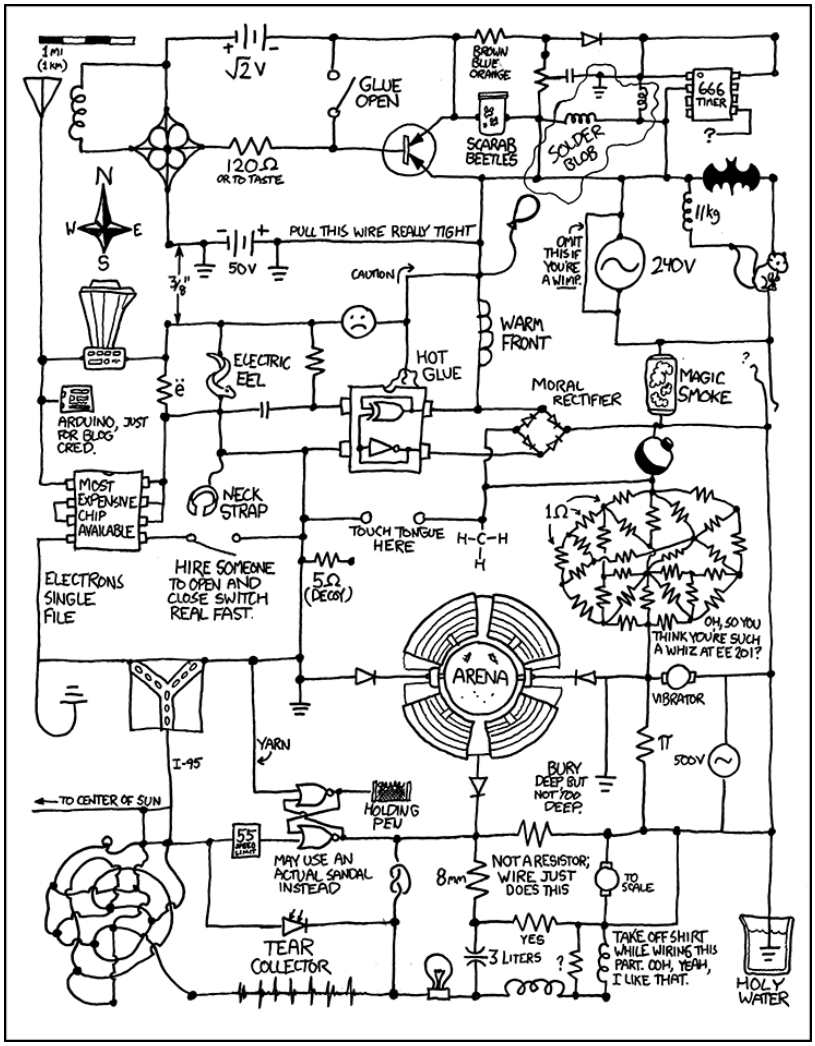


Figure 1.1: XKCD Comic strip on circuit diagrams by Randall Munroe (2010) from `https://xkcd.com/730/`.

incrementally composing a program. Accordingly, the machine's architecture has to adhere to a fixed, finite user manual to facilitate programming. To achieve this, numerous potential interactions among the components of the machine need to be suppressed."[Zau05a]

Thus, starting from an informal problem definition, the process of programming typically involves the intermediate steps of finding either informal problem solutions or formal problem definitions and then the generation of a formal problem solution in a

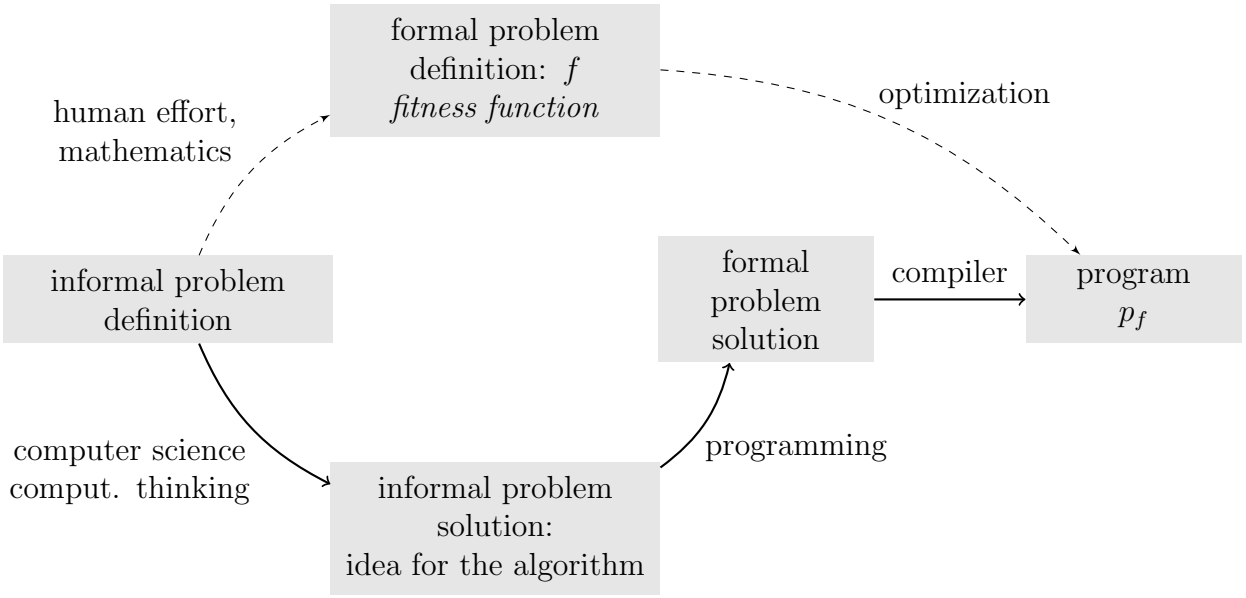


Figure 1.2: Classical vs. optimization approach to programming: In the classical programming process (lower, solid edges), an informal problem definition is solved by translating an informal problem solution, i.e., an idea for an algorithm, first to a formalized solution into a programming language and then automatically to a program p_f. The process of developing ideas for an algorithm, "computational thinking" [Win06], is typically supported by computer science knowledge that offers efficient standard methods for many typical applications. Drawn as dashed lines in the upper part of the diagram, an optimization approach is indicated where a programmer would first ignore solution strategies but focus on the formalization of the problem definition, resulting for example in a fitness function for an evolutionary algorithm [Fog66, Rec71, Sch75, Hol75, Koz89, Fog94, Bey02, Wei02, Eib08] or for other optimization systems.

higher programming language that is further compiled to a machine-readable binary code for a computer, as outlined in Figure 1.2. So in this thesis, conventional programming implies that humans use engineering principles, like understanding predictable systems, modularity, logic, composition and reasoning to first derive an informal problem solution strategy from the informal problem definition and then formalize this solution until the digital computer can execute the solution. There are many scientific results from computer science that suggest a set of potential building blocks for arriving at solution strategies and there are various techniques for engineering formalizations of these strategies [Knu73, Knu69, DC11]. Typical design decisions are the distinction between declarative and imperative programming paradigms and the use of modern software engineering design principles that allow for code-reuse like procedures, object orientation, data-driven programming, generics or aspect orientation as reviewed in [Har96, Kic97, Rob03, Mar10].

Even though it is sometimes claimed that computer science was a young discipline, our digital computers are being programmed in "higher programming languages" since the 1960s [Mar10], i.e. for around 50 years by now. But if we allow for a broader interpretation of the notion of programming, already the design of Babbage's mechanical difference engines [Bab89, Roe09], the beginning of which lay in the 1830s, could be considered programming in the sense of the definition given above: the mechanics of the difference engine were used to implement an algorithm from a finite, definite and effective set of components and their interactions. The Antikythera Mechanism, a complex mechanic astronomical calendar, even dates back more than 2000 years [Car14].

Because of this "long history" of algorithmics, a wide knowledge is established about the theoretical basis of algorithms, their worst case run times, their memory consumptions, the theoretical limits of computability, etc. [Knu73]. But even with good theoretical foundations, practically implementing these algorithms as programs is still a hard job that needs to be done by experts with many years of learning and experience, and typically entails many failures before computing systems behave in the

way they are intended to [Smi94, Jen02, How03, Rob03]. Scientific methods for producing secure and reliable code are for example static analysis [Cou77, Bes10], code verification [Hoh02] and unit testing systems [Che02].

In an ideal world however, programming would be simple. Science-fiction visions, e.g. in "star trek", also commonly feature interactions with computers on a verbal level, even though less ideal misconceptions there typically end in disaster. Approaches in this direction might be knowledge engines like Wolfram Alpha, Apple's Siri or IBM's Watson [Fer10, Jan12] system that process informal or even verbal requests. Nonetheless, this thesis aims lower than building the ultimate futuristic programming system. But still, obviously, there is much space for improvements in recent programming techniques. Hopefully studying novel programming paradigms, e.g. from unconventional computing and less rigorously defined information processing systems, will help to increase the efficiency, robustness and flexibility of our computing systems and simplify the process of programming them.

1.3 Definition of Computation

While the actual topic of this thesis is unconventional programming, the term *computation*, which is tightly interwoven with programming, should be concretized beforehand in this section. Computation might classically be defined as the execution of a formal algorithm, where computation is carried out error-free as a finite number of definite, effective and discrete, "mechanical" manipulation steps over a finite set of symbols [Knu73, Cop08]. For this thesis however, the notion of computation will deviate from this typical view: Instead, the perspectives of Zauner [Zau05a], MacLennan [Mac04], Suzuki [Suz13] or Lizier [Liz10] are adapted, who perceive computation as a physical process that can be used or harnessed for solving information processing problems. This concept is satirized in Figure 1.3. The definitions of the problems to be solved vary slightly from author to author: For Zauner these problems are defined as input-output mappings, for MacLennan the manipulation of abstract symbols,

Figure 1.3: SMBC Comic strip on computation: Drawing by Zach Weiner (2013) from `http://www.smbc-comics.com/?id=3054`.

whereas for Lizier the information-theoretic view of transmitting, storing and manipulating information is in the focus. Although the information-theoretic perspective will be further studied in Chapter 4, throughout this thesis the term *computation* will mostly be used in the sense of an abstract input-output mapping in combination with its realization. The physical process that eventually fulfills the abstract computation somehow happens in the real world and is typically noisy, continuous, asynchronously updated and infinite. This real-world process will here be called the implementation or synonymously the program p_f. So, in contrast to Turing Machines, we will not restrict our systems to using finite or discrete domains for the input, the output or the internal states of the computing systems.

Let $f : X \to Y$ be an abstract function that maps an input $x \in X$ to an output $y \in Y$. This abstract function f defines the abstract computation and somehow needs to be implemented in conventional or unconventional hardware as symbolized

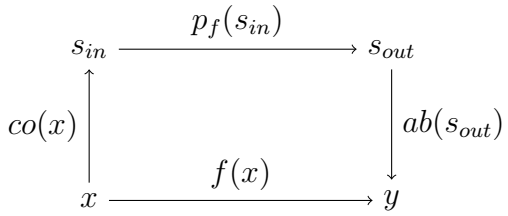


Figure 1.4: The concept of *computation* used in this thesis encompasses both, an abstract function $f : X \to Y$ and its implementation in a complex system or program $p_f : S_{in} \to S_{out}$. Implementation and program are used synonymously here. The computing device maps an abstract input symbol $x \in X$ in the form of a signal $s_{in} \in S_{in}$ to an abstract output $y \in Y$ in the form of an output signal $s_{out} \in S_{out}$. An abstraction function $ab : S_{out} \to Y$ and its analogue, the concretization function $co : X \to S_{in}$, are used to translate between the physical signals s_{in} and s_{out} and the abstract values x and y.

in Figure 1.4. Throughout this work, I will use the notion of a program $p_f : S_{in} \to S_{out}$ in the sense of such an implementation for the abstract computation f. While we now typically expect the computing machine to be independent of the actual program in the sense of the Universal Turing Machine [Tur36], the program p_f can as well be hard-coded in the computing dynamical system, as it is often the case with unconventional computing systems [Zau96]. Coarsely, *programming* is referred to in its broadest sense, as specifying the desired functionality of a computing device in contrast to the typically understood exact, algorithmic specification of digital data manipulation. Still, while the abstract function f maps abstract symbols X to Y, the real-world computation p_f produces an output signal $s_{out} \in S_{out}$, given an input signal $S_{in} \in S_{in}$. Thus, an abstraction function $ab : S_{out} \to Y$ and the related concretization function $co : X \to S_{in}$ are needed to translate between the abstract symbols and their correlating real-world signals. For example, in digital electronics, voltages around +3.3 V might be translated to a logical 1 while voltages around 0 V would be translated to a logical 0. In short, in this thesis, programming means the derivation of the program p_f, and, additionally, the abstraction and concretization functions $ab(x)$ and $co(s_{out})$ for a given abstract function f. In the next section, examples of unconventional programming approaches are discussed that might help or guide the search for such implementations p_f.

1.4 Unconventional Programming

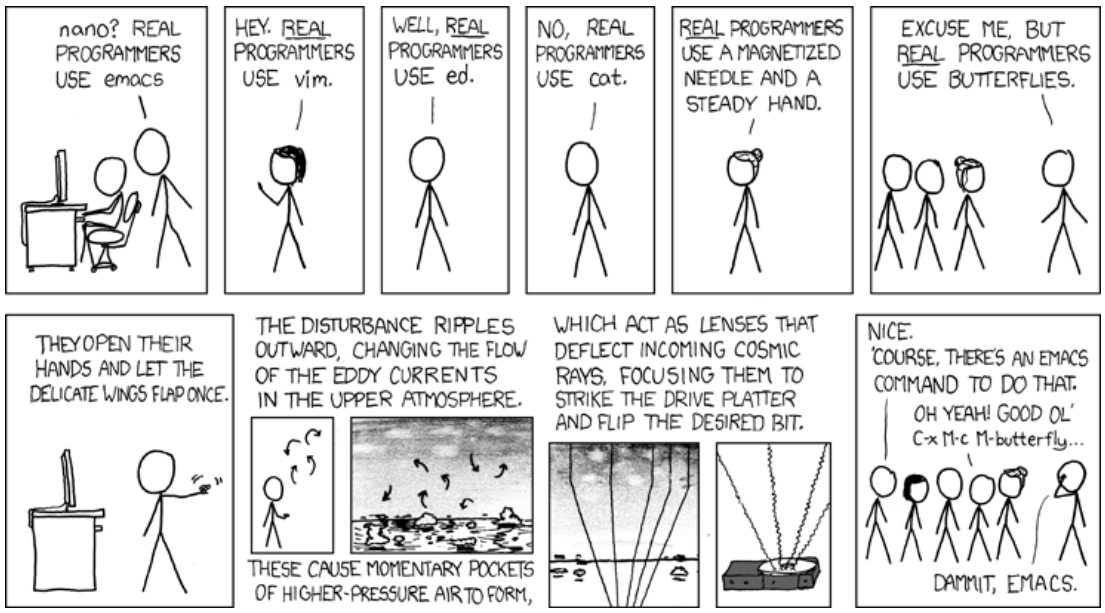


Figure 1.5: XKCD Comic strip on programming by Randall Munroe (2008) from `https://xkcd.com/378/https://xkcd.com/378/`.

In the following, a short overview on other unconventional programming and computing concepts will be given to set them apart from typical programming techniques that follow engineering principles, as satirized in Figure 1.5. Hopefully, these unconventional approaches allow for a more potent use of a medium's computational potential, e.g., for making possible a different set of abstract computations than traditional formalisms do: *"The set of compressible maps is a small subset of the potential input-output functions – most behaviors cannot be programmed."*[Zau05a]

Using Formal Problem Descriptions

Possibly on the borderline between conventional and unconventional programming paradigms, approaches that make use of a formal problem definition like mathematical optimization, constraint based programming [Jaf87], linear programming or evolutionary algorithms and genetic programming ([Fog66, Rec71, Sch75, Hol75, Koz89,

Fog94, Bey02, Wei02, Eib08], Chapters 3 and 5) allow for the automatic generation of more or less formalized but machine readable solutions: The formal problem solutions here could be fitness functions, optimality criteria or constraints. Note that evolutionary computation can also be implemented in a different medium as will be demonstrated in the Chapter 6, resulting in a simulation of embodied evolution [Wat02, Gru11b].

Example Driven Programming

The large field of machine learning is most prominently represented by example-driven, supervised learning techniques [Kot07]: Here, neither the problem nor the solution are formally defined. Only a set of learning test samples, which can even contain mistakes, is supplied together with an expected solution. Although machine learning is a well-established field of artificial intelligence, in our understanding of programming as defined in the last section, machine learning and classification can also be considered to be "programming", such that they implement the function f that maps high-dimensional inputs $x \in X$ to a much smaller space of output classes $y \in Y$. When the set Y of output classes is larger or more complex, various (symbolic) regression techniques can be used that are similarly based on a large set of samples and expected solutions. Another interesting combination of supervised learning and complex dynamical systems are liquid state machines [Maa02, Maa04], echo state networks [Jae01] or reservoir computing approaches [Luk09, Laz09, Ese14]. These approaches seem especially useful for generating complex, time-dependent output trajectories reacting on the input signals and might even have a biological neural analogue in the cerebellum structure [Yam07]. Similar to evolutionary computation, also machine learning techniques are not necessarily implemented on digital computers only. For example perceptron-like neural networks can also be implemented in chemical media [Hje91, Ban13]. In contrast to supervised learning techniques, for the class of reinforcement learning algorithms [Kae96], the information about the expected solution is not present. Instead, only the guesses made by the system are evaluated for their correctness. So while some problems are hard to express or to solve

in particular optimization or machine learning approaches [Min69, Wol97, Vap00], this "programming" via machine learning can be very powerful and has already be studied thoroughly [Kot07].

End-User Development

Another set of programming techniques are grouped together under the notion of end-user development [Fis04, Rep06]. These techniques are not designed to be used by computer experts, so only relatively informal methods are required to specify the problem and to find a solution strategy. Prominent examples are programming by example and programming by demonstration [Cyp93], where the user generates small training data sets or shows the behavior that is then abstracted by the system. Obviously, *programming by example* is overlapping with supervised machine learning techniques. Typical environments for this kind of programming concepts are spread sheet applications, visual programming systems [Gre96] like LabView or domain specific systems [Fow10].

Accessing and interlinking entries from large public databases by formulating the correct queries to knowledge engines [Jan12] can be seen as an act of programming as well, which was sought for in the "semantic web" [BL01] approaches, and has recently been advanced by combining it with natural language processing capabilities in products like Wolfram Alpha, Apple's Siri or the electronic "Jeopardy!" quiz show player "Watson" [Fer10].

Gap-Closing Techniques

As already discussed in Section 1.2, in classical computer programming, the programmer can envisage the macroscopic system behavior while writing microscopic instructions. In many unconventional computing systems [Ada01, Ada06, Mat07, Teu08, Ste12], e.g. in molecular computing, it is much harder to bridge the gap between the microscopic reaction rules and the macroscopic system behavior [Con95b, Zau05a,

Mat06a]. This offers a potential for novel analysis techniques that allow for the observing and understanding of dynamical systems from a different perspective and thus eventually also might lead to a different programming paradigm: Changes in the (typically implicit) system description are understood from different viewpoints, e.g., from the perspective of information dynamics [Sha48, Sch00, Wil10a, Wil10b, Liz10], organization theory [Dit07, Mat06a, Mat07], surrogate models [Pfa01, Ong03, Sim04, Cas10, Caw11], or by using genome parameters [Nic12]. This hopefully also leads to a new understanding of the basic modifications that can channel the system's behavior into desirable directions [Ban05, Teu08, Ste12].

For the visual programming languages mentioned earlier, the two-dimensional space only leads to a more convenient layout of the participating components of the work- or data flow. Yet the two-dimensional space and geometry can also be used to build an intuitive understanding of the macroscopic system behavior from the specified microscopic rules: For example amorphous computing or spatial computing paradigms [Abe00, Gia05, Bea05, Bea11] typically aim at finding relatively simple or compact conventional programs for a population of agents that lead to the desired emergent, macroscopic effects. When the spatial medium has less intrinsic computational capabilities, collision computing approaches [Ada01, Ada05, DLC11, Hol11b] can still exploit the geometric structure and compute Boolean logic or geometric problems, like Voronoi diagrams [Ada11c].

Linked to the problem of experimental design, in surrogate modeling or approximation modeling, evolutionary algorithms and other optimization techniques are coupled with a modeling framework to allow the optimizing system to simultaneously generate a simpler model of the full-detail model or of the real system. In particular, when real experiments are slow or expensive, the past experiments are reused to generate another (cheaper) model of the system. In the aforementioned micro to macro problem, this technique can be used to predict the effect of changes in the program, but also to scout for interesting experiments that would improve the surrogate model [Pfa01, Ong03, Sim04, Cas10, Caw11].

Especially when adopting the information-centric view on computation (cf. [Sha48,

Sch00, Wil10a, Wil10b, Liz10]), information-theory can play an interesting role as a symbol-encoding independent filter, so that the abstraction and concretization functions from Figure 1.4 do not have to be known in advance. In this thesis, this approach will further be studied in Chapters 4 and 5.

1.5 Summary

Throughout this thesis, unconventional methods for understanding and programming untypical computing systems will be explored using artificial chemical droplets that resemble biological neurons, which were designed in the NEUNEU EU project[1]. Networks of these droplets are introduced in Chapter 2, where models of this computing architecture are reviewed and extended on different scales. A system of lipid covered droplets containing reagents of the Belousov-Zhabotinsky (BZ) reaction is used in experiments as model system to study the signal transmission dynamics of chemical computers and their modeling. A chemical medium in sub-excitable, excitable or self-exciting (oscillating) regimes supports propagating excitation pulses. These pulses can be used for information coding and processing. Models that can be applied to describe the time evolution of a medium composed of droplets are discussed: a homogeneous differential equation model, a spatially extended partial differential equation model and a cellular automaton model of the chemical reaction are reviewed. Furthermore, a new high level modeling approach for the droplets is proposed, that discretizes the chemical states and considers stochasticity in the transition functions. It is demonstrated how the values of experimentally measured quantities like oscillation periods, diffusion coefficients and wave propagation speeds can be deduced from the lower level models.

Throughout this thesis, a special emphasis will be on the importance of suitable signal encodings for these systems. Obviously, different communication schemes between the elements of information processing systems may vary in their effectiveness, depending on the envisaged information processing task. Chapter 3 explores the idea of using

[1]project website: `http://neu-n.eu`

co-evolution schemes to design droplet networks together with appropriate signal encodings to communicate with droplet computers for simple Boolean and classification tasks. There, three different evolutionary set-ups are tested: Evolving network structures with fixed on/off rate coding signals, co-evolution of networks and signals, and network evolution with fixed but pre-evolved signals.

Subsequently, Chapter 4 will present the use of information-theory based measures to better understand the information flows in droplet networks but will also lead to symbol-encoding independent metrics for the quality estimation of droplet networks. The complex dynamics of unconventional computing devices like networks of droplets filled with the self-exciting Belousov-Zhabotinsky (BZ) reaction can be hard to track and to understand. Corresponding to recurrent neural networks, the flow of excitations in the network is not limited to a single direction in the droplets. Especially when unconventional computing systems are not engineered but evolved through genetic algorithms, the actual process of computation will often be incomprehensible. Several methods from Information Theory like Transfer Entropy, Information Dynamics, and Information Decomposition offer approaches for observing and analyzing computing systems on a higher level and allow for a better understanding of the involved data transferring and manipulation operations. In this work we show how to discretize the spike trains of BZ droplet networks and how to apply mutual information measures on the time series data of both physical implementations as well as on simulations.

Afterwards, in Chapter 5, a new method for using a system's own dynamics to find suitable symbol encodings, using so-called Tautological Loops, is introduced and studied. Therefore, a network of potential implementations of a function is simulated or built that feeds the calculated outputs back to other implementations of the same function as input. Hence no external output has to be specified and the dynamics can self-organize a state where suitable symbol encodings emerge that can be interpreted as well as produced by the function's implementation. Next to theoretical considerations about potential tautological loop architectures, the method is exemplified by finding a NOR gate built from the excitable droplets that will be introduced in Chapter 2.

While the next Chapters will mainly focus on information processing systems built from chemical droplets, the approaches introduced and described are not only meant to be applicable by this small group of unconventional computers. Instead, the methods will be generalizable for diverse unconventional or non-programmable systems and hopefully also lead to novel perspectives in programming and designing "conventional" computing systems. In this spirit, some experiments not involving chemical droplet systems were also conducted as explained in the later chapters: In Chapter 6), embodied evolution is simulated *in silico* in the form of an evolutionary algorithm that is completely implemented in SRSim [Gru10], a simulator for spatially structured, rule-based reaction systems. Then, in Chapter 7, the information processing perspective of biological, self-assembling structures like the Kinetochore [Tsc13, Gör13, Ibr13] and the yeast interphase chromatin organization [Geh12] are studied. A short summary and some ideas for further research are finally given in Chapter 8.

Chapter 2

Droplet Computers

Through the following chapters of this thesis, a compartmentalized, excitable chemical medium is used in experiment and simulation to demonstrate unconventional programming approaches. In this chapter, which is largely based on the paper [Gru13], we focus on modeling and simulation of this kind of medium, which can be used for computation. Chemical computers [Ada01, Ada05, Szy10, Iga11, Bul13] might be used in many fields of applications, ranging from controlling bio reactors to designing smart drugs, as reviewed in [Dit01, Dit05, Zau05b]. A nonlinear chemical medium, for example accommodating the Belousov-Zhabotinsky (BZ) reaction [Zai70, Noy72, Fie72, Gyo90], can dissipate the chemical energy showing various types of non equilibrium behavior like for example temporal oscillations or regular spatio-temporal structures. The type of non equilibrium evolution can be controlled by the initial concentrations of reagents used. A medium in the *excitable* regime, once it is properly stimulated, shows one cycle of oscillations, but it does not spontaneously enter a new oscillation cycle, but returns to the stable stationary state. *Self-exciting* medium, in contrary, will start oscillating spontaneously. However, this medium can still be used for information processing, since new oscillation cycles can also be triggered externally before the self-excitation happens. A more complicated behavior is shown by a *sub-excitable* medium, which is less excitable than the initially mentioned *excitable* medium. This means that an excitation wave entering a droplet from one

direction will not spread in any direction but will keep a "memory" of the original direction. Also, when two *sub-excitable* waves collide, the resulting wave can propagate into a new direction, rendering this medium suitable for collision-based computing [Ada02a, Ada04, Hol11a].

For the experiments presented here, the BZ medium is compartmentalized into small droplets [Agh08, Szy11] that form when the solution of reagents is dripped into oil. The compartments are stabilized against merging through lipid molecules that self-assemble at the border between the aqueous and the oil phase. Where two droplets meet, a lipid double layer membrane can be formed that still allows chemical reagents to pass through and to trigger an excitation in the neighboring droplet. Excitation waves can be transmitted through droplets but can also interfere with one another, dependent on their timing and on the chemical properties of the droplets and the medium within. Hence, droplets arranged in a network form a potential chemical computer [Gor03, Szy11, Ada11c, Ada11d, Hol11b]. An experimental implementation of such a droplet system is shown in Figure 2.1.

Based on a small experimental system of four droplets, we are giving an overview on the simulation techniques to describe and understand the behavior of different kinds of excitable media on different scales. After a short review of ordinary and partial differential equation and cellular automaton models, we introduce an event-based model that can be used to simulate a droplet network on large spatial and temporal scales. This droplet model will frequently be used throughout this thesis.

2.1 Basic Properties of the Belousov-Zhabotinsky Medium

At the proper concentrations of reagents the chemical Belousov-Zhabotinsky (BZ) medium can oscillate or can be triggered into an excitation.

In our experiments such medium was made by mixing the right amounts of water solutions of sulfuric acid, sodium bromate, malonic acid, potassium bromide and

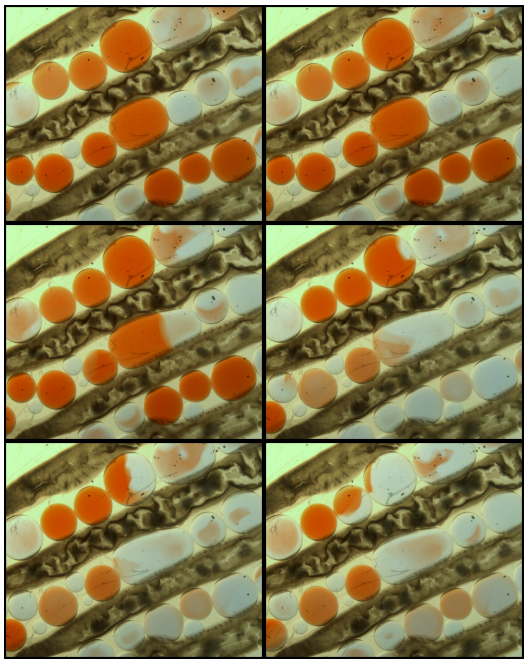


Figure 2.1: Experimental droplet system made from manually aligned droplets of a Belousov-Zhabotinsky medium. Time progression is shown from top left to bottom right. In the the bright red state, the medium is resting and transparent/white. Excitation waves propagating between droplets can be observed. Even though droplets remained relatively stable, a merger between the central two droplets is shown in the third frame. Pictures by Josephine Corsi (2011), Southampton.

ferroin [Gor12]. However, the resulting process is rather complicated: It has even been modeled using 26 chemical species and 80 reactions [Gyo90], so some kind of simplification is necessary for the practical use of the reaction models. We will first summarize those aspects of the BZ-reaction that we will focus on in this work.

Most obviously, the BZ system visually indicates its state [Fie72, Gre78, Ger90, Ada10]. A chemical redox indicator like ferroin can provide information on the phase of the reaction system by switching from red to blue when a high concentration of reduced catalyst is outweighed by oxidized catalyst. Therefore the state of BZ-medium can be measured optically as the intensity of blue or red components on the color image of the medium.

These phases, as visualized in Figure 2.2, show up under well-stirred conditions or with spatial resolution. Beginning in the *responsive phase* when the medium looks red, it can be triggered into an excitation or self-excite, displaying an almost immediate change from red to blue color. When the fast change occurs, the medium is in the *excited* phase shortly and becomes *refractory* directly afterwards. In the *refractory* phase the medium is unresponsive against further stimulation. Then the system recovers its red color relatively slow and becomes ready for the next oscillation cycle. Once the BZ medium can be excited again, either by the excitation in the neighborhood or through external influences, we shall refer to the medium as *responsive* here. Dependent on the type of medium, the *responsive* phase can be quite stable and last until some perturbation occurs. In case of the *self-exciting* BZ mixture, in contrast, the *responsive* phase is instable such that a new oscillation cycle is started after the system was *responsive* for some time.

Excited droplets can influence their neighbors, but these can only react with a new excitation if they are in the *responsive* phase. When a droplet is *excited*, we assume its oscillation cannot be influenced by neighboring *excited* droplets. But even when a droplet triggers its neighbor into a new oscillation, there will be some delay between the excitation of one and the next droplet, since the BZ waves will first have to cover some distance from one BZ compartment through the membrane and into the other droplet.

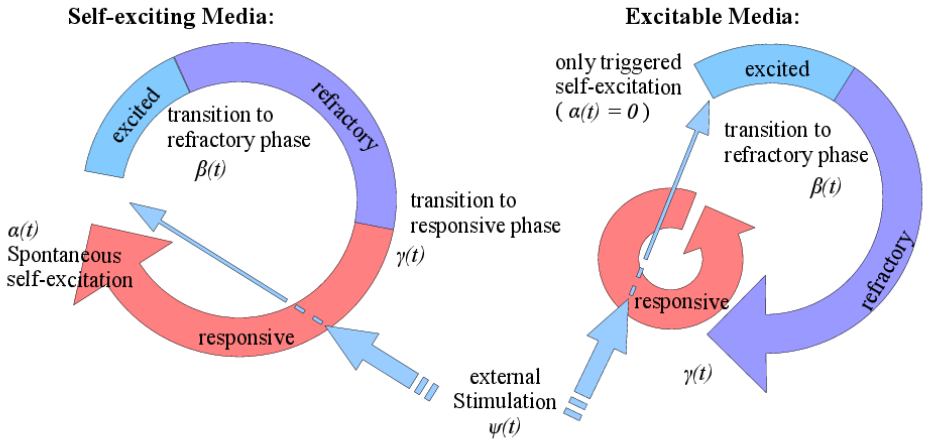


Figure 2.2: Abstract BZ Reaction cycle for self-exciting and excitable media. The complex Belousov-Zhabotinsky reaction can coarsely be described as a cyclic change of the state of the medium on the left side. For the *excitable* medium, the system can wait for external stimuli very long without going through a cycle by itself. The distribution functions $\alpha(t), \beta(t)$, $\gamma(t)$ and $\psi(t)$ describe the probability distributions for changing from one system state to another after staying there for time t.

We analyze the system states in different areas of an exemplary BZ system of four linearly arranged droplets in Section 2.3 and estimate their oscillation periods and wave propagation speeds. Even though a chemical reaction is an inherently stochastic process and subject to randomness at the molecular level, due to a large number of molecules involved in the experiments, the excitation periods of the droplets appear to be quite reliable. Nevertheless, there are qualitative changes happening in the medium that are beyond the scope of our modeling, e.g. the continuous degradation of the medium and the concomitant increase of the oscillation periods. Furthermore, spiral waves [Win72, Gre78] form and disappear, maybe due to gas bubbles or spatial changes in the droplet structure. Though our models do not cover the genesis of these phenomena, we were able to describe the situation once the special conditions are established.

2.2 Hypothetical Droplet Types

Like specialized electronic components on a circuit board, we suggest to utilize a variety of specialized droplet types. The typical droplets, that we were describing so far and that we are considering in this work's experiments and models, can be characterized as *Or* droplets: when we consider droplet "crossroads", where a droplet is connected to three or four other droplets, a signal arriving on one lane will spread out in all other directions as long as the other droplets are responsive. Nonetheless, we will introduce some hypothetical droplet types that can be of use when designing more complex droplet computers as pointed out in the Outlook Section 2.6.

Or The standard droplets as we are describing them in Sections 2.3 and 2.4. These droplets distribute incoming excitations to all other adjacent and responsive droplets. If there are two inputs to a droplet of this type, both inputs can equally lead to an excitation, such that the droplet behaves like a logical "And".

High Activity A droplet that is filled with quickly oscillating, self-excitatory medium might be used to supply a droplet network with continuous signals, e.g. acting as

pacemaker or timing signal. It is possible to manufacture this kind through a different BZ medium composition or by externally influencing the droplet, for example via optical stimulation.

And Droplets that can be stimulated by two or more synchronously arriving excitation waves, but not by a single one, might be build from droplets that contain less excitable BZ mixture. It is an open question, dependent on the further exploration of the droplets in laboratory experiments, how much synchronization between the two input signals is necessary to allow an activation of the And droplet.

Diode Another possibly valuable droplet type can propagate signals solely in a single direction while blocking signals arriving from the other direction. It could thus help to insulate some parts of a droplet network from the influences of other substructures and in general to have more control over the range of system dynamics. A chemical implementation can be achieved using diode membrane channels [Mag09], through differently sized droplets [Ste98, Szy11] and probably also with a different media composition.

Repeater To solve potential timing problems of the And droplets, we assume that we can manufacture a repeater droplet that will, once activated, repeat an excitation signal for longer than the typically short excited phase. This can be realized through droplets with different sizes [Ste98] or medium compositions [Gor11a], resulting in different oscillation periods.

Inhibition A droplet that, once activated, inhibits its adjacent droplets might prove extremely helpful. One could argue that the oscillation in the inhibitory droplet might consume the substrate of its neighbor droplets or that it might throw its neighbors into the refractory state, bypassing the excited state. Even though this type of droplet might be hard to produce, we will still consider the implications of the theoretical existence of such a droplet.

species	concentration
sulfuric acid	0.6 M
sodium bromate (NaBrO3)	0.45 M
malonic acid	0.35 M
potassium bromide (KBr)	0.06 M
ferroin	1.7 mM

Table 2.1: Composition of the BZ medium.

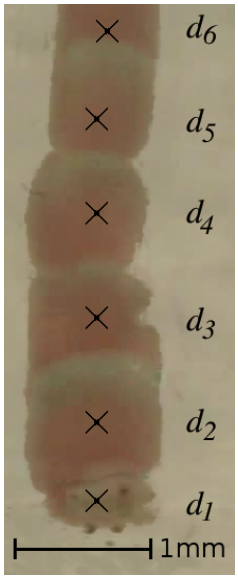
2.3 Exemplary Droplet System

We describe our modeling approaches by a system that consists of four stacked droplets of different sizes as displayed in Figure 2.3a. The initial concentrations of reagents are summarized in Table 2.1. For such concentrations the BZ-medium oscillates. The largest diameter of the droplets is approximately 10^{-3} m and the length of the droplet chain is approximately $6 * 10^{-3}$ m. Though the droplets in this system change their form slightly over time, the general structure and the droplets' linear arrangement stays constant over the experimental time of approx. 48 minutes or 2880 s (Supplementary movie[1]).

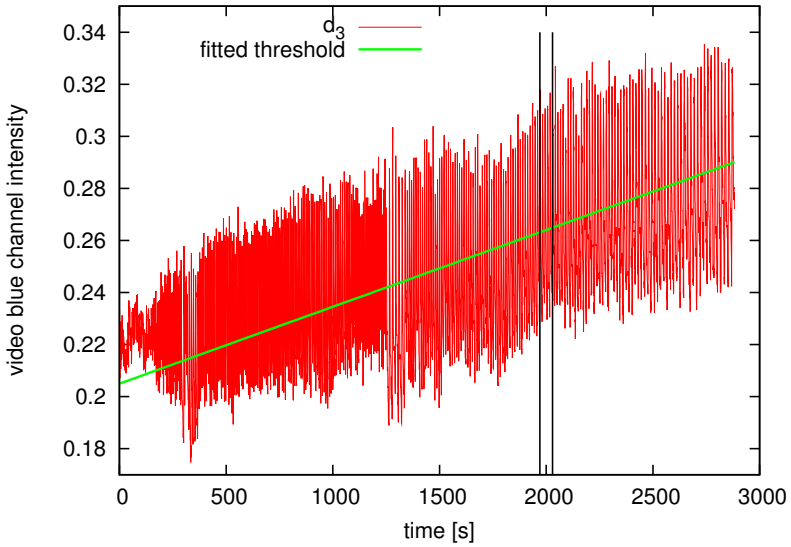
We reduced the description of coupled droplets to the observation of the time evolution of the phase at selected points located close to the droplet centers. The state of the medium is defined as the intensity of the blue channel video signal averaged over a 10 by 10 pixel rectangle around the selected points marked in Figure 2.3a. The blue color component of the video and thus the concentration of the oxidized catalyst over time is displayed in Figure 2.3b for the position d_3 only and in Figure 2.3c for all positions but over a smaller time interval. Probably due to impurities and due to the small size of the lowest droplet, we could not extract reliable oscillation data for the position d_1.

From the observations of the state of medium, we measured the oscillation periods that are displayed in Figure 2.4. The oscillation periods are determined by approximately overlaying a "threshold line" centrally over each of the oscillation plots analogous to

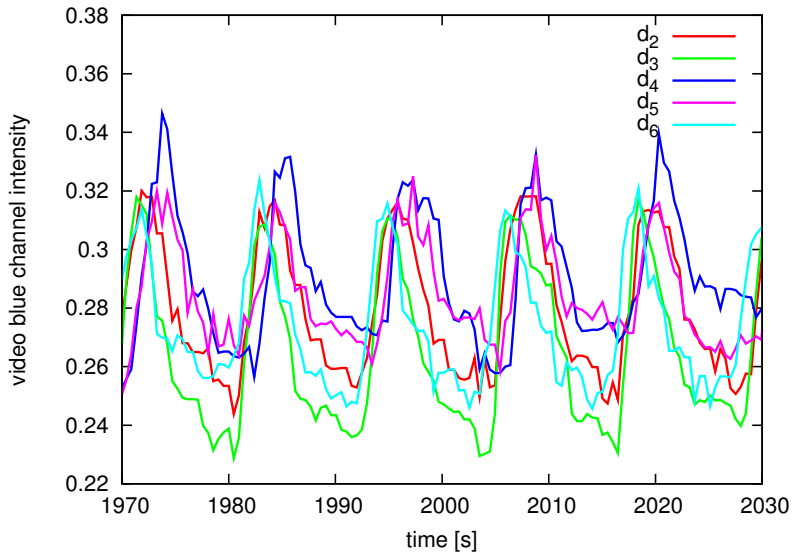
[1] `http://www.chemicalneuronalnet.uni-jena.de/Results/Project+Media.html`



(a) droplet system



(b) position d_3



(c) all droplets

Figure 2.3: Droplet system and extracted oscillation data. a) Different positions along the droplet system are marked with the symbols d_1 till d_6. b) The phase at the position d_3 is plotted over the whole experimental time. Additionally, a green line indicates the threshold that will later be used to extract oscillation periods from this data. Vertical black lines show the interval that is represented in Figure c). c) The intensities at the positions d_2 till d_6 are plotted over a small time window. Due to impurities and the small size of droplet d_1, we could not extract excitation data here.

the green line displayed in Figure 2.3b. Then the interval between the times when the threshold line is crossed upwards gives the oscillation periods for each position as displayed in Figure 2.4. In this figure, the main accumulation of oscillation periods increase linearly over the whole experimental time, starting from about 5 s and increasing to ca. 15 s after 3000 s. Some outlying points originated from the noise in the video data as seen in Figure 2.3c which leads to some positions being identified as rising oscillation flanks mistakenly. Likewise, some actually rising flanks may not be found, resulting in longer oscillation periods in the plot.

In a similar way, we calculate the interval between oscillations at neighboring positions to estimate the delay of the excitation wave between these positions as displayed in Figure 2.5. The time delay for the wave propagation is not equal between all positions pairs. Nonetheless, this does not imply that the wave travels through the medium at different speeds. Instead, the positions are not exactly the same distances apart. Additionally, crossing the lipid membrane, e.g. between the positions (d_1,d_2), (d_3,d_4) or (d_4,d_5), does take the excitation wave longer than travelling the same distance inside a droplet, e.g. between the positions (d_2,d_3) or (d_5,d_6).

The system is observed for 2880 s, counting around 300 oscillations during this time. The system's oscillation periods are increasing almost linearly over the experiment as shown in Figure 2.4. Nevertheless, we are using constant oscillation periods in all modeling scopes here even though we are working at incorporating the dynamically changing system behavior. Since the system showed a rich variety of different behaviors around the time $t_m = 1000\ s$, we chose this time for the modeling in the next section.

The lowest and smallest droplet at position d_1 with a diameter of about 0.6 mm oscillates the fastest and, for most of the time, controls the oscillations in the remaining system. Another interesting behavior of the observed droplets is a phase of slower self-excitation of the top droplet at positions d_5 and d_6 in a time window between simulation time 300 s and 800 s. In this phase, the top droplet self-excites with an oscillation period that is longer by a factor of about 2, compared to the oscillations induced by the lowest droplet at position d_1. There is also a short abnormal phase

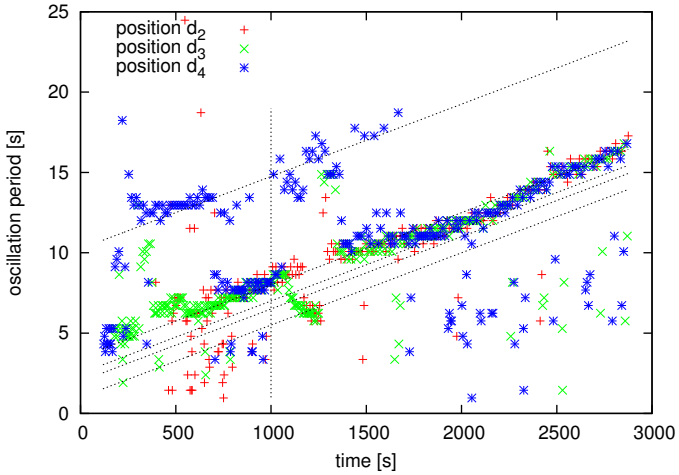
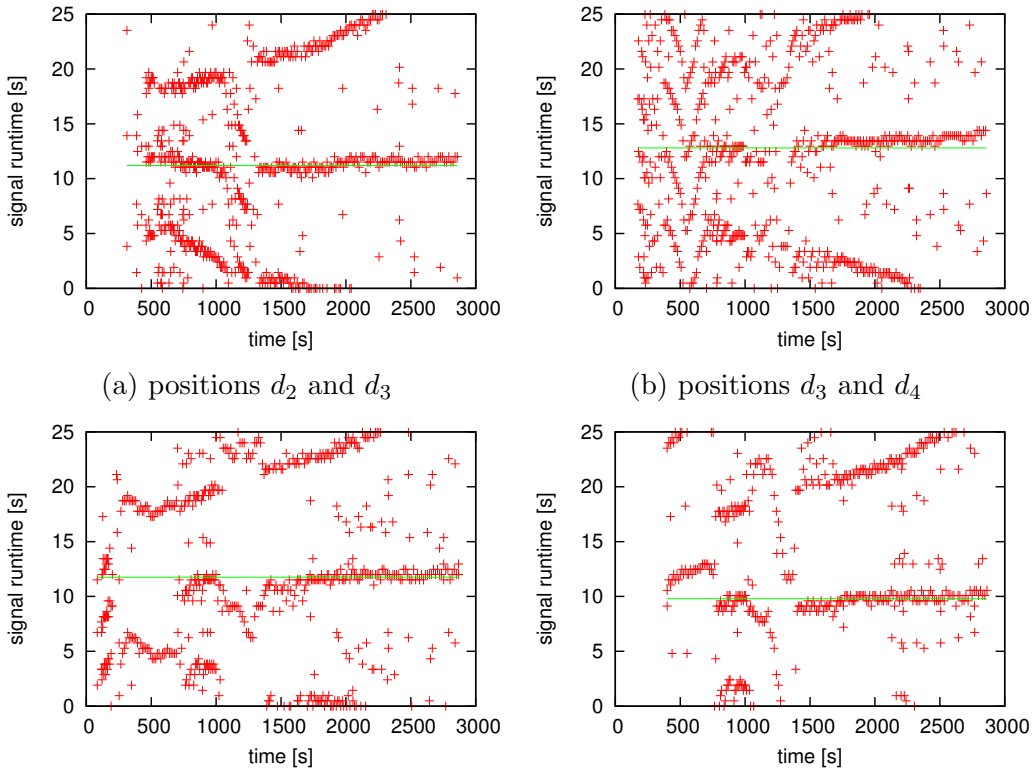


Figure 2.4: Oscillation periods extracted from experimental video. For simplicity, only positions d_2, d_3 and d_4 are shown in this plot. Generally, the oscillation periods increase over the experiment, following an approximately linear regime. A large cluster of oscillation periods is found on the line linearly climbing from 5 up to 15 s (second auxiliary line). These oscillations are results of trigger waves originated at position d_1. Most of the time, as can be observed in supplementary video 1, these waves spread out and dominate the whole system. A second cluster of oscillations on the line from 10 up to 25 s is mostly observed in the first half of the experiment (first auxiliary line). Here droplets self-excite because of broken influence of position d_1, supplying an estimate for the self-oscillation periods of droplets. Oscillation periods plotted below the lowest auxiliary line are most likely due to measurement and data extraction errors, such that two successive oscillations would often add up to elements on the main accumulations. As an exception, around the time 1200 s, a spiral wave pattern emerges close to position d_3 and leads to faster oscillations at position d_2 and d_3. Since these waves arrive with a high frequency, the BZ solution at the lipid bilayer between positions d_3 and d_4 does not have enough time to recover. Hence only every second wave can be transmitted over the gap between the droplets. Partially this long time between two oscillations already leads to self-excitations.



(a) positions d_2 and d_3

(b) positions d_3 and d_4

(c) positions d_4 and d_5

(d) positions d_5 and d_6

Figure 2.5: Signal propagation times, extracted from video data in Figure 2.3. The red crosses indicate the time delays between excitation of one droplet position and the following position. The green lines are fitted along the main aggregation of signal propagation times around 10 s and is used to parameterize the delay times of the discrete-event model in Section 2.4.4. The aggregations of delay times around 10 s are caused by a single excitation wave passing the system in forward direction, i.e. from d_i to d_{i+1}. The other two major aggregations, which are increasing and decreasing over the experimental time, are caused by waves that pass position d_i before or after the wave that is considered for position d_{i+1}. While the wave propagation speed stays approximately constant over the experiment, the delay between two successive waves at two distinct positions changes with the oscillation frequency over the experiment, leading to the increasing and decreasing secondary aggregations in the plots. In the time window between times 1000 s and 1500 s, the direction of the waves is partially inverted, leading to a change in the measured time delays as well. Furthermore, there are some delays marked far-off the main aggregations, which are mostly a result of mistakes at identifying the rising flanks of the oscillation due to noise.

system property	**observation time frame**	**position**	**fitted function**
trigger-wave from d_1 period	200 .. 2880 s	d_1 .. d_6	$\frac{9}{2000}t + 3.5\ s$
self-excitation period	500 .. 1500 s	d_2 .. d_6	$\frac{9}{2000}t + 10.25\ s$
refractory time, lipid bilayer	1200 s	d_3, d_4	$\frac{9}{2000}t + 2\ s$
oscillation period, spiral wave	1230 s	d_3, d_2	$\frac{9}{2000}t + 1\ s$

Table 2.2: Summary of the functions that are fitted based on Figure 2.4. The fit of the trigger waves' oscillation period, which control the system most of the time, is obvious. Fitting the self-excitation times that can be observed between times 500 till 1500 s is harder because of the smaller number of samples and a large variance therein. Choosing a linear function with the same slope as the trigger-wave function leads to an acceptable fit, at least in the small time window that the self-oscillations appear in. For the remaining parameters, the refractory times of droplet centers, droplet borders and the oscillation times of spiral waves, there is basically just one point along the time axis available. So following the previous two functions, we assumed the same slope again here. Nonetheless, for modeling the system behavior around time $t = 1000\ s$, the error should be relatively small because the underlying observations are made close to this time as well.

between the times 1050 s and 1250 s when a spiral forms in the lower middle droplet and leads to an oscillation period that is reduced by a factor of approximately 0.7, compared to the triggered oscillations. While these fast oscillations propagate into the lowest droplet after some time, they do not cross the membrane to position d_4 and above. These observations allow us to characterize the behavior of the droplets as summarized in Table 2.2.

2.4 Modeling Excitable and Self-Exciting Droplets

For model building, there are typically versatile approaches available that influence model characteristics like the level of detail, the accuracy, the simplicity, computability and other features. Differential equation approaches are often based on first principles, highly detailed but mostly also computationally demanding. They are probably best suited for understanding the behavior of the BZ reaction, especially when happening under uncommon conditions like inside a lipid membrane with its side effects.

This understanding is the basis for optimizing the chemical composition of medium, oil and lipids to produce a desired behavior.

But for figuring out which kinds of behavior could actually be of use to perform some kind of computation, faster and larger models including hundreds and more droplets become necessary. This is for example the case with cellular automaton or event-based simulation models of droplets that are simplified to use discrete states instead of continuous concentrations. In an even more abstract sense we will also need modeling techniques, such as high level programming languages to describe the function of complex droplet networks. We will give a short outlook into this direction in Section 2.6.

Naturally, the further one of these modeling viewpoints is apart from the first principles of the system, the harder will it be to obtain the necessary parameters. Ideally, we would like experiments and first principle models to parameterize the models on the next abstraction level. In contrast here, some of the higher level properties such as oscillation periods and wave propagation velocities can be observed directly in the experiment.

2.4.1 Well-Stirred Ordinary Differential Equation Model

Brusselator- and Oregonator-like [Pri68, Noy72, Fie72] models based on ordinary differential equations are commonly used to describe the time evolution of a perfectly stirred BZ-medium. We are using variations of such models by Szymanski and Gorecki, that allow obtaining the necessary system parameters directly from the experimentally used concentrations [Szy11, Gor11a, Gor12]. In the simplified two variable interpretation, the model describes the dynamics of one activating chemical species x and one inhibiting chemical species z. They correspond to bromous acid ($HBrO_2$) and ferroin ($Fe(phen)_3^{3+}$) concentrations, respectively. It reads as follows:

$$\frac{\partial x}{\partial t} = \epsilon_1 h_0 N x - \epsilon_2 h_0 x^2 - 2\alpha\epsilon_1 MK \left(\frac{1}{\beta} + q\frac{1}{h_0}\frac{z}{1-z}\right)\frac{x-\mu N}{x+\mu N} \tag{2.1}$$

$$\frac{\partial z}{\partial t} = \frac{h_0 N}{C} x - \alpha \frac{KM}{C h_0} \frac{z}{1-z} \tag{2.2}$$

The necessary model parameters are h_0: the Hammett acidity function of the solution and the concentrations of K : KBr, M : $CH_2(COOH)_2$, N : $NaBrO_3$ and the catalyst C : $[Fe(phen_3^{2+})] + [Fe(phen_3^{3+})]$ as listed in Table 2.1. Further fixed parameters are $\mu = 1.6 * 10^{-5}, \alpha = 2.6 * 10^{-4}, \epsilon_1 = 1200, \epsilon_2 = 6700, \beta = 1000$ and $q = 0.51$.

The oscillation cycle, cf. Figure 2.2, starts the excited phase with a moderate concentration of the activator x and a low concentration of the inhibitor z. Then the amount of inhibitor and activator rise rapidly and lead to a negative second derivative $\frac{dx}{dt}$ for the activator equation until x will finally drop back to its initial low value. From this point on, we call the BZ solution refractory with a low activator concentration and high but slowly decreasing inhibitor concentration z.

Now, if there was a small inflow of activator into the system, the BZ mixture could not be triggered into the next oscillation because the high inhibitor concentration would quickly degrade the activator. But when the inhibitor concentration z drops below a critical value, an inflow of activator x will not be degraded fast enough and the BZ mixture can be triggered into a new oscillation cycle. We define the responsive phase such that it begins when a specific activator inflow is sufficient to trigger the next excitation in the BZ mixture and lasts until the next excited state begins. Clearly, the begin of the responsive state and thus the threshold for the concentration of the inhibitor z depends on the amount of activator that flows into the system. This inflow would typically stem from an arriving excitation wave when considering spatially inhomogeneous BZ mixtures. Hence it is not possible to describe this point without a characterization of the membrane boundaries or the diffusion coefficients. In contrast, when using partial differential equations and appropriate diffusion coefficients in the next section, the inflow of activator from an arriving excitation wave is defined.

Dependent on the BZ system characteristics, the next responsive phase can last infinitely in the case of the excitable medium. Alternatively, the inhibitor can, in self-excitable medium, drop so low that the next excitation wave is triggered reliably after a certain interval.

2.4.2 Reaction-Diffusion Partial Differential Equation Model

For describing the spatial and temporal propagation of excitation waves in addition to the well-stirred dynamics of the BZ system, we need to include space. In the simplest form, this can be done by adding the Laplacian term into differential equation (2.1) of system variable x that is supposed to diffuse [Gor11a, Gor12], arriving at equation (2.3). We consider the molecules represented by system variable z to be larger, leading to diffusion on a slower timescale that will be ignored here. This change introduces the diffusion coefficient D_x of species x as new parameter, resulting in the equations:

$$\frac{\partial x}{\partial t} = \epsilon_1 h_0 N x - \epsilon_2 h_0 x^2 - 2\alpha\epsilon_1 MK \left(\frac{1}{\beta} + q\frac{1}{h_0}\frac{z}{1-z}\right)\frac{x-\mu N}{x+\mu N} + D_x \triangle x \tag{2.3}$$

$$\frac{\partial z}{\partial t} = \frac{h_0 N}{C}x - \alpha\frac{KM}{Ch_0}\frac{z}{1-z} \tag{2.4}$$

The partial differential equation models of BZ droplets proved their value for investigating the interaction between two droplets [Gor12] and, in a slightly modified form, for designing circuits or logic gates of about ten droplets for sub-excitable medium as we will show in Section 2.5.

Many parameters, such as diffusion coefficients for different species or at different locations in the droplet, might be hardly observable. Here, parameter fitting approaches are a practical way of deducing knowledge about the system from macroscopic properties like wave propagation velocities or oscillation periods [Gor12]. On the downside, these parameter fitting methods are indirect and results are not guaranteed to be unique.

However, the spatial expansion of the ordinary differential equation model of the last section is not straightforward. Causes are the immense computational efforts of the simulation, numerical instabilities, diffusion properties through the lipid bilayer that have to be determined in addition to coefficients in the medium and boundary conditions at the droplet borders. For example in supplementary movie 1, the waves'

passage over droplet borders and the resulting wave delay can be observed. Also, close to the droplet borders the BZ reaction can be inhibited [Ste98], whereas we assume a spatially homogeneous BZ medium in equations (2.3) and (2.4). Computations can be sped up using graphics card acceleration with CUDA [Jan10] and by specialized simulation techniques [Bar91]. Nonetheless, the numerical integration of such a partial differential equation system is computationally expensive and will mostly be used for small systems, considering few excitation waves.

2.4.3 Cellular Automaton Models

The problems of complex parameter sets and high computational expenses can at least partially be resolved in cellular automata [vN66, Wol83]. In this case, more phenomenological, ad hoc models are built, which rely on less unobservable parameters, but also less on first principles but rather on the observable properties of the system. Space is mostly discretized into a regular grid. Also time and cell states are discrete and update rules describe how each cell's state is calculated from the cell and its neighborhood's previous state. The discretization usually uses a varying number of states that fall into the classes of excited, refractory and resting or responsive phases. Examples for such automaton models could be four state systems [Gre78] or three state automaton models [Ada01, Ada10, Hüt12] that were already used to simulate systems of large droplet numbers.

Generally, cellular automata do not necessarily produce the same behavior for different spatial lattices [Shi03]. For instance, when simulated in two or three dimensions, the earlier cellular automaton models did not correctly reproduce the curvature that is observed in the real BZ medium. Also they did not account for the dispersion relation, meaning that the medium is less excitable shortly after being excited and thus transmits excitation waves slower. These effects were captured in more recent simulations [Ger90, Bar91].

Even though there are approaches of timed automata [Alu94], there is another challenge when modeling using cellular automata: the typically fixed time steps pose additional constraints in contrast to the irregular proportions between different phases

in the BZ system. This means that either the temporal resolution of the phases is coarse or that a large number of states has to be used for each phase to generate the observed timing fractions.

2.4.4 Discrete-Event Model for BZ Droplets

To follow the efficient simulation approach of cellular automata while including the possibility for exact timing, we propose a discrete-event based modeling and simulation approach [Fis01]. The parameterization of this model is relatively simple and can be derived from principal observations of the excitable medium. Furthermore, many droplet system can be simulated efficiently, i.e. many hundred excitations of systems of about 100 droplets can be simulated in less than a second. It uses coarse grained space and few discrete droplet states but continuous time. These states are the excited, the refractory and the responsive phase as introduced in Section 2.1 and as used by most cellular automaton approaches. Either deterministic or stochastic state transition functions can be applied, though we will only demonstrate deterministic state transitions in this work. We define the state transition behavior of the droplets with the following mathematic functions:

- $\alpha(\tau)$ Probability density function for the transition time from responsive state to the excited state. Hence this parameter describes the inclination of the system to self-excite.
- $\beta(\tau)$ Probability density function for the transition time from excited to the refractory phase. Hence the length of the excited phase.
- $\gamma(\tau)$ Probability density function for the transition time from refractory to responsive phase. Hence the length of the refractory phase.
- $\psi(\tau')$ Probability density function for the transmission time of a signal from one excited droplet to another responsive droplet taking the time τ'.

For these functions, τ is the time since entering the current state. For the transmission of excitation waves, in contrast, τ' is the time needed to propagate the excitation from

one droplet to another. The basic condition for transmission is that one droplet is excited while the other one is responsive. But since the spreading of waves takes some time, a delay τ' is sampled from the distribution function $\psi(\tau')$. So if one droplet is excited between the times t_1 and t_2, it can trigger an excitation in an adjacent droplet if this is responsive between $t_1 + \tau'$ and $t_2 + \tau'$.

Simulations

From an initial state of the system, all following events are sampled using the distribution functions $\alpha(\tau)$, $\beta(\tau)$, $\gamma(\tau)$ and $\psi(\tau')$. The times for these events are organized in a priority queue so that only changing droplets' states contributes to computational costs. At each simulation step, the earliest next event from the event list is selected, removed, verified and then executed by modifying the simulation data structure. Subsequently, all possible events for the modified droplet are inserted into the event queue. Events have to be verified to check if the situation of the droplet they concern has not changed in the time between issuing the event and the current simulation time. An event belonging to a droplet can for example become outdated if another droplet triggers it into an early excitation or if it is stimulated from outside the system to input data or to simulate noise. Therefore, less active or inactive parts of droplet networks do not require high computational efforts. The computational complexity of simulating a network of n droplets for a time t scales with $O(t \cdot n \cdot log(n))$. The factor n is due to the increase in the number of events with more droplets while the logarithmic factor results from modifying the priority queue. Our simulation software can be freely downloaded from the website[2].

Representation of Droplet Networks

Generally for our simulation system, an arbitrary graph structure can be used to define the connections between different droplets. This graph does not need to have a two or three dimensional embedding, such that we are free to explore arbitrary

[2] www.neu-n.eu

droplet network topologies.

For us, in analogy to an undirected graph $G = (V, E)$ which is defined by the tuple of its vertices and edges, a droplet network $N = (D, E)$ is defined by the tuple of its droplets $D = \{d_1, .., d_n\}$ and its connectivities or edges $E \subseteq D \times D$. Each droplet d_i can be one of the droplet types as defined in Section 2.2: $d_i \in \mathcal{D} = \{d^{Norm}, d^{LowEx}, d^{Slow}, d^{In}, d^{Out}\}$.

In simulation as well as in laboratory experiments, droplets will many times be arranged in two-dimensional $m \times n$ grids, such that a matrix-like description of the droplet network becomes convenient:

$$N = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,n} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m,1} & d_{m,2} & \cdots & d_{m,n} \end{pmatrix}$$

This description implies that adjacent droplets in either a Moore- or Von Neumann neighborhood are considered to be connected without explicitly mentioning the edges, i.e. $d_{i,j}$ is connected to another droplet $d_{k,l}$ iff $|i-k| + |j-l| \leq 1$ for a von Neumann neighborhood or iff $|i-k| \leq 1 \wedge |j-l| \leq 1$ for a Moore neighborhood.

Spatial Discretization

An intuitive discretization of space might be to choose one discrete droplet per position in Figure 2.3a, i.e. six droplets, such that the larger droplets are divided into smaller sub-compartments. In this case, this approach would not reproduce the observed wave dynamics:

Observing the experimental system in Section 2.3, the time required for excitation waves to pass from one position d_i to an adjacent one d_{i+1} is in the same order of magnitude as the oscillation period. More importantly, the wave propagation time is larger than the refractory period of about seven seconds. This means, the actually

travelling excitation waves can be hidden in the signal transmission times of the discrete model. Then, an excitation wave can pass from one sub-compartment to another via the signal transmission delay, even if there is another excitation wave moving in the opposite direction, which would actually annihilate the first wave in the real or differential equation systems. Furthermore, cyclic oscillations can form between two sub-compartments when two abstract oscillations move in opposite directions without canceling each other out. By subdividing droplets into smaller sub-compartments, we can choose the spatial discretization so fine that the signal propagation time τ' becomes smaller than half of the smallest refractory period. This resolves the problem: Any excitation started from sub-compartment d_i at time t_0 will not arrive at sub-compartment d_{i+1} before $t_0 + \tau'$. When sub-compartment d_{i+1} is now sending out an opposed excitation at $t_0 + \tau' - \epsilon$, shortly before the first wave hits, d_{i+1} will clearly be refractory. The opposed wave will hit the first sub-compartment d_i at the earliest at time $t_0 + 2\tau' - \epsilon$, which is still before the first sub-compartment d_i leaves the refractory state again.

So we divided the positions up into few smaller sub-compartments, such that the wave propagation times for the smaller compartments are shorter than half of the refractory times as displayed in Figure 2.6. Hence we represent the real system of four droplets with a system of 33 discrete sub-compartments. Each of the positions from Figure 2.3a is replaced by a set of four to six sub-compartments for the gradual signal transmission, another two to represent the signal transmission properties of the lipid bilayer and one more sub-compartment to record the excitations at the centers of d_1 till d_6. The lipid bilayer representing sub-compartment are shared by two adjacent positions and are replaced by sub-compartment of higher excitability in the case of the contact region between positions (d_2, d_3) and (d_5, d_6), where no bilayer is present in the experiment. For the sub-compartment representing medium, we used a refractory time of six seconds, while using seven seconds for the lipid bilayer representing sub-compartment.

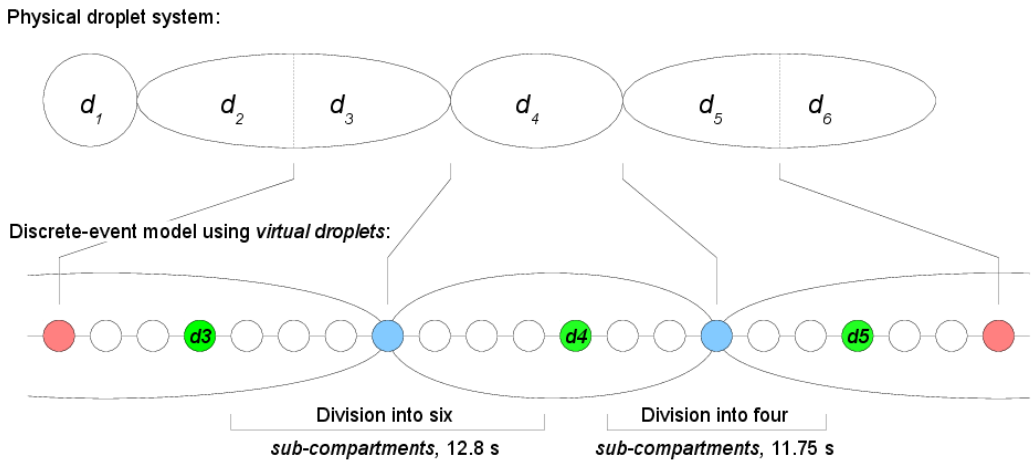


Figure 2.6: Mapping from the physical droplet system into the model composed of sub-compartments. While the upper part of the figure represents the physical droplet system as displayed in Figure 2.3, the lower part represents the discretization into homogeneous and discrete sub-compartments. Here, the white compartments represent the actual BZ medium between the observed positions and simulate the gradual propagation of excitation waves. The colored compartments are not delaying the wave propagation but are used for varying excitabilities and for observation. We use the blue sub-compartments to represent the droplet borders with their potentially higher excitability threshold that can lead to trigger waves spreading through the medium but not propagating over the lipid bilayers as observed in Section 2.3. The red droplets are the analogues to the blue droplets, representing the medium at the center of the large droplets. Green droplets represent the positions d_1 ... d_6 and are introduced into the model for observing the states at these positions.

Discrete-Event Model Instances

To demonstrate the capability of the discrete-event model to reproduce the qualitative system behavior, we show the two most prominent behaviors of the BZ droplet system presented in Section 2.3. To instantiate the discrete-event model in this case, we need to provide the distribution functions $\alpha(\tau)$, $\beta(\tau)$, $\gamma(\tau)$ and $\psi(\tau')$. In consistence with the differential equation models introduced before, we will only use deterministic transition functions by using the Dirac delta function $\delta(\tau - \tau_0)$. The τ_0 values will then be chosen as the timing parameters that are known from the experiments (c.f. Section 2.3) or could be determined from the ordinary differential equation models. In spite of the deterministic functions chosen here, it would also be possible to sample the phase transition times for example from Normal distributions, which would require the variances as further parameters though. We know that the concentration of the activating species $HBrO_2$ is noticeably high only for a very short period of time from the ordinary differential equation modeling in Section 2.4.1. Hence we simplify the excited phase to a single point in time, switching from the responsive to the excited and to the refractory behavior instantly by setting $\beta(\tau) = \delta(\tau - 0)$. The transition functions for the discrete-event model sub-compartments as we are using them here are summarized in Table 2.3.

First, we simulate the dominance of the trigger waves, generated at position d_1 around experimental time 1000 s, over the remaining system in Figure 2.7. In simulation and approximately also in the real system, all signals are oscillations with the same periodicity as d_1 but with a shifted phase.

As second instance of the discrete model, displayed in Figure 2.8, we decrease the self-excitation period of position d_3 down to 6.7 s. This value is close to the oscillation periods observed between experimental times 1000 and 1200 s when a spiral wave dominates position d_2 and d_3. Furthermore, the oscillation frequency is below the 7 s refractory period that we chose for lipid-bilayer droplets, resulting in blocking at least every second trigger wave. It means that the dominance of the trigger waves from position d_1 is interrupted for positions d_4 till d_6. Instead these upper droplets are controlled by every second spiral wave that can pass into the third droplet at

	Parameter τ_0 in $\delta(\tau - \tau_0)$ for distribution:			
	$\alpha(\tau)$	$\beta(\tau)$	$\gamma(\tau)$	$\psi(\tau)$
length of phase:	responsive	excited	refractory	transmission
d_1	0 s	0 s	8 s	0 s
d_2, d_3, d_5, d_6	9 s	0 s	6 s	0 s
d_3 normal	9 s	0 s	6 s	0 s
d_3 spiral	0.7 s	0 s	6 s	0 s
lipid bilayer droplet	8 s	0 s	7 s	0 s
medium d_1 - d_2	0 s	0 s	6 s	$\frac{10}{4}$ s
medium d_2 - d_3	9 s	0 s	6 s	$\frac{11.2}{4}$ s
medium d_3 - d_4	9 s	0 s	6 s	$\frac{12.8}{6}$ s
medium d_4 - d_5	9 s	0 s	6 s	$\frac{11.75}{4}$ s
medium d_5 - d_6	9 s	0 s	6 s	$\frac{9.8}{4}$ s

Table 2.3: Parameters for the discrete-event model used in the shown simulations. $\delta(\tau-\tau_0)$ denotes the Dirac delta function. The parameters are estimated from analyzing the experimental data and from ordinary differential equation modeling. Parameters for droplet d_1 are chosen with a zero responsive time since we only know that it triggers excitations every eight seconds, but not if it would be possible to excite it earlier. For the signal transmission times, fractions of four and six are chosen, because the experimental droplets are subdivided into four or six smaller sub-compartments droplets, to reduce the otherwise large signal transmission delay. For the transmission time between position d_1 and d_2, we chose 10 s since no measurements were available here. Membrane sub-compartments and the lipid bilayer sub-compartments are parameterized with different fractions between refractory and responsive time: The lipid bilayer area seems less excitable since trigger waves that propagated through the medium could not pass over the droplet borders in the experiments.

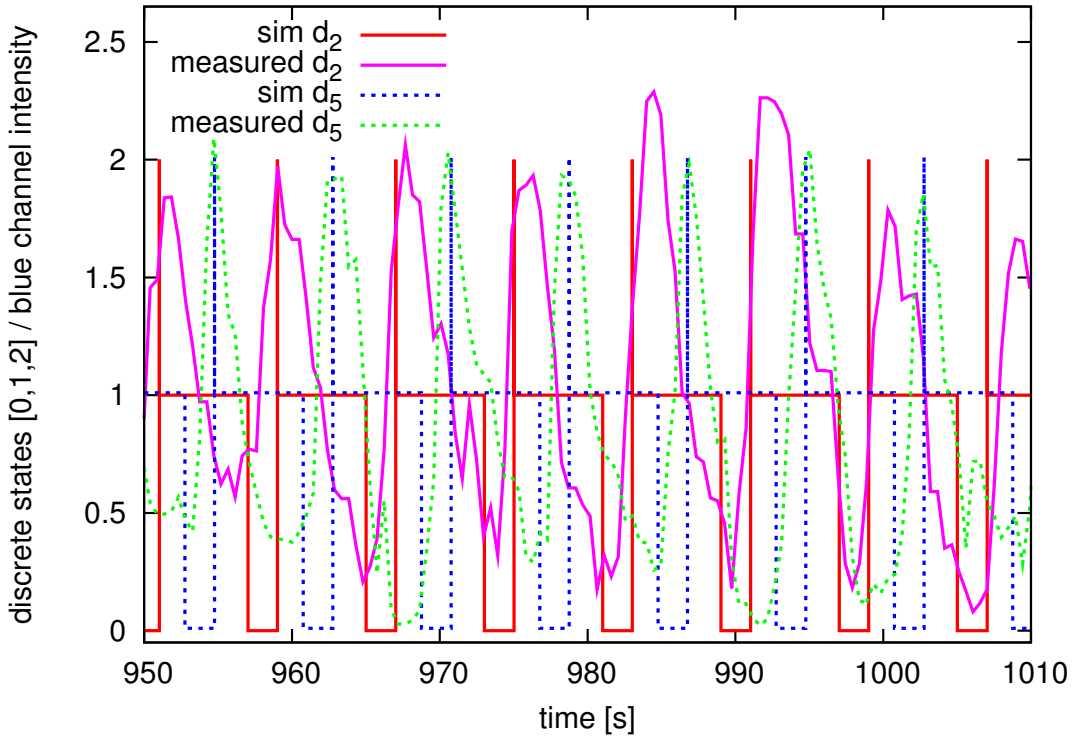


Figure 2.7: Comparison of the discrete model simulation for excitable droplets with the experimental system in positions d_2 and d_5 (cf.Fig. 2.3a). The data reflects the predominant control of the trigger waves generated by the first droplet at position d_1. The time axis for the experimental data is shifted with the function $t' = (t - 5\ \mathrm{s})$ to fit the curves of the discrete-event simulation. Furthermore, the experimental blue channel intensity is scaled to fit the interval $(0, 2)$ that is used by the discrete-event simulator to indicate the states responsive, refractory and excited with 0, 1 and 2 respectively. No further scaling of the time-axis was applied.

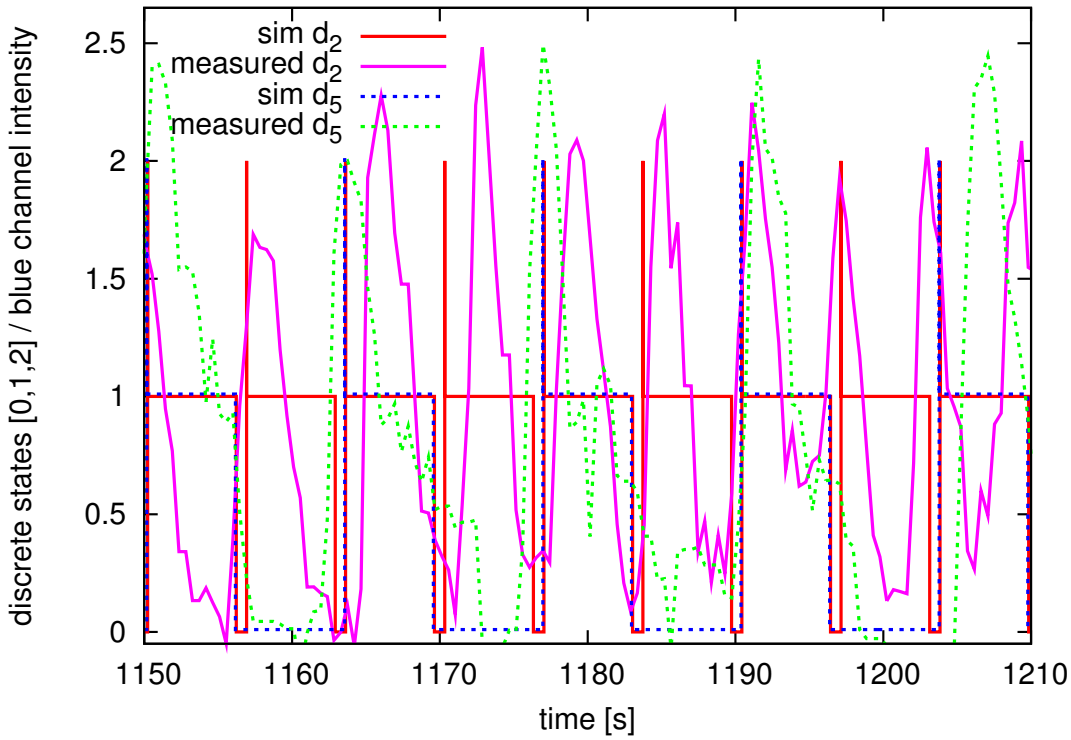


Figure 2.8: Comparison of the same simulation and experimental data as in the previous figure, but now at a later time-interval showing the effect of faster spiral waves in positions d_2 and d_3 on the remaining system. Since the system is modeled at time 1000 s but the related system behavior is found around time 1200 s, the time axis for the experimental data is rescaled with the function $t' = (t * 0.95 + 57.5)$ to compensate for the lower oscillation frequency at this slightly later time of the experiment.

position d_4. If the period of the spiral waves was a bit longer, every second wave would not have arrived fast enough to stop positions d_4 till d_6 from self-excitation. This effect can be observed in the video (Supplementary file 1) around time 1300, when self-excitation appears in the upper droplets.

2.5 Modeling of Sub-Excitable Droplets

Also sub-excitable BZ droplets were modeled using partial differential equations to design small networks implementing logic and arithmetic functions that could later be used as building blocks for larger systems [Hol11a]. Because the droplets explained here are experimentally realized as light projected in the shape of circles, they are referred to as discs in this section.

Logically symbolic waves are able to traverse the network modulated by interaction with pathways and other waves. The disc interior can be exploited for free space collision style reactions [Ada02b] where the pore loci and efficiency can compartmentalize the resulting reaction [Ada11b]. Circuits have been created from logical sub assemblies in orthogonal and hexagonal networks [Hol11a, Ada11b]. The functional density can be increased when including more variations in relative disc size, pore efficiency and connection angles [Hol11a].

Disc designs have been simulated on a two variable version of the Oregonator model [Noy72] as a model of the BZ reaction [Zai70, Zha73] adapted for photo-sensitive modulation of the Ru-catalyzed reaction [Kuh86] similar to the partial differential equation system described in Section 2.4.2 Use of a photo-sensitive adapted version of the Oregonator model permits a simple migration from simulation to experiment. Circuit designs from the simulation can be projected directly onto an actual photo-sensitive BZ medium [Hol11b]. Numerical simulations are achieved by integrating the equations using the Euler-ADI[3] method [Pre92].

Contrasting previous logic gates and composite circuits designs using the BZ substrate, for example [Tót95, Ste96, Mot99, Gor09], Figure 2.9 presents an example for

[3]Alternating direction implicit method.

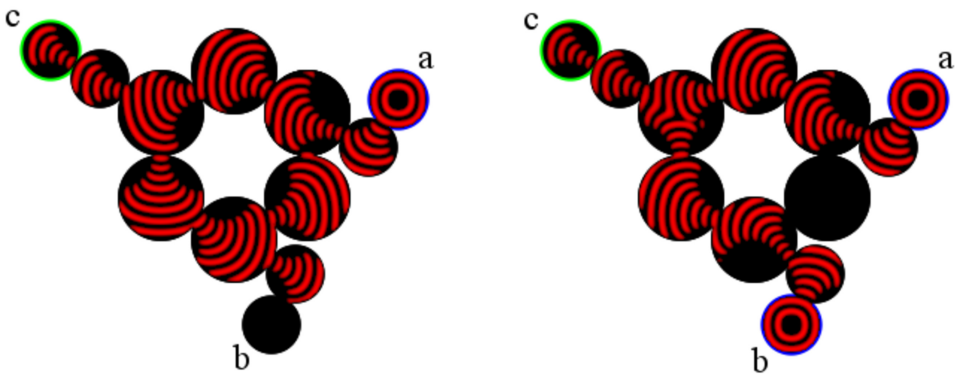
logic elements that can be created using nothing other than interconnected BZ discs. Wave fragment flow is represented by a series of superimposed time lapse images (unless stated otherwise). To improve clarity, only the activator wave front progression is recorded.

Next to logical AND, NAND and NOT gates, both for adaptive behavior in natural and synthetic computation is memory. It allows animals and machines to build an internal state independent from the current external world state. We present an example 1 bit volatile read write memory cell constructed entirely with BZ discs. Independent but similar to previous designs [Mot99, Mot01] in so much that the existence or absence of a rotating wave represents the setting or resetting of 1 bit of information.

When two BZ waves progress in opposite directions around an enclosed channel, loop or ring of connected discs, then at some point the two opposing wave fronts will meet and are always mutually annihilated. Nevertheless, if a unidirectional wave can be inserted into the loop then that wave front will rotate around the loop indefinitely[4]. Furthermore the rotating wave can be terminated by the injection of another asynchronous wave rotating in the opposite direction. Opposing inputs into a loop are analogous to a memory *set* or *reset*. Reading the state of the cell without changing the state can be achieved by connecting another output node where a stream of pulses can be directed to modulate other circuits [Gor03] (Figure 2.9).

The loop and a unidirectional gate (diode) are the two key constructions of this type of memory cell. Unidirectional gates in BZ media have previously been created by exploiting asymmetric geometries or chemistry on either side of a barrier [Agl96]. An alternative design is possible however using discs connected with different apertures. The operation relies on the relationship between the wave expansion and the angle of the connection. Fine control of the wave beam would in theory allow other angles of connectivity [Ada11a] and other functions.

[4]For as long as the chemical reagents can sustain the reaction.



(a) (b)

Figure 2.9: Memory cell with additional diodes on the cell inputs. Two additional angled diode junctions are added to each of the input discs (`a` & `b`). This prevents a reverse wave flow back down either of the inputs. An example output disc is also connected (top left). (a) Wave insertion at (top right) `a` input node results in a persistent counter-clockwise wave. Reverse wave flow down the opposing (bottom) input is blocked by an angled diode junction. (b) Simultaneous `a` & `b` inputs produce one output pulse `c` and annihilate wave rotation. Figures by Julian Holley (2011), Bristol.

2.6 Conclusions

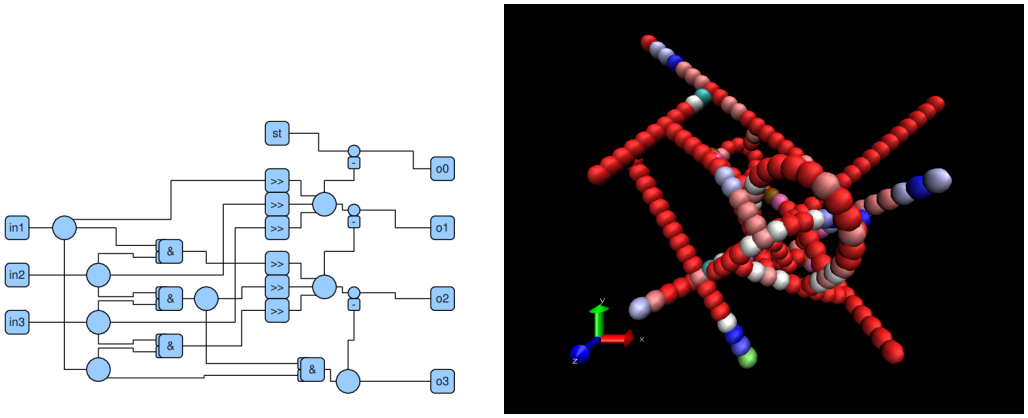
In this chapter, a summary on current techniques of modeling compartmentalized excitable media was presented and exemplified with lipid covered droplets of Belousov-Zhabotinsky BZ medium swimming in oil. The techniques are accompanied by an experimental system of four droplets and the analysis of their behavior over a time period of about 48 minutes, during which we registered about 300 oscillations waves. Starting from a non-spatial ordinary differential equation model, a spatial perspective is reviewed in the partial differential equation models. Then, spatial and temporal discretization to speed up simulation studies using cellular automata is discussed. Finally, in the Sections 2.4.4, an event-based model is proposed to achieve higher timing precision while preserving the efficiency for the investigation of large systems.

Even though we show the qualitative reproduction of the systems behavior with the

discrete-event model, there are some effects that are not yet covered by this approach. Most obviously, the real-world oscillation periods are increasing over the experimental times up to a factor of about three. This is neither covered by the differential equations nor by the discrete droplet models. While we are currently working on extending the differential equation systems to include this effect, it might in principle be easier to reproduce this effect in the more phenomenological event-based systems by adding the global time t as another parameter, such that the probability distribution functions look like $\alpha(t, \tau)$, $\beta(t, \tau)$, $\gamma(t, \tau)$ and $\psi(t, \tau')$. Furthermore, the dispersion relation [Kee86, Ger90] is another currently neglected effect, meaning that excitation waves move faster when the medium had more time to recover after the previous excitation. Here partial differential equation approaches already capture this effect but it would also be possible to have a signal transmission function $\psi(\tau')$ that would consider the droplets excitation history.

Observing the experimental data of Figure 2.3c, it becomes clear that a certain level of noise is part of the system, even though some quantum of the noise will also be due to the camera and digitization process. So we expect some uncertainty about the length of the oscillation period [Ali97], about the length of its phases, about the amplitudes, about the geometries of droplets, about the excitability of droplets and about the connectivity between droplets. These effects are probably rather troublesome properties of the system that will make the design of robustly working droplet computers more challenging. Consequently, a useful model of computing droplet systems will have to consider random perturbations. Though we do not show how to estimate the correct noise levels from the experiments here, especially the discrete models allow a parameterization of the transition functions, e.g. by representing the fluctuations as Gaussian probability density functions instead of the Dirac delta function.

The research on computing droplet systems should eventually lead to non-Boolean computing and information processing systems. But for both, unconventional as well as conventional computing approaches, the droplet models from this chapter constitute a foundation on which further experiments are built. Even though networks of conventional binary logic gates like in Section 2.5 and in the following Chapters



(a) Network Schematic (b) Network Simulation

Figure 2.10: A counting droplet network as design and in simulation. Dependent on the number of stimulated inputs, four different outputs will be excited maximally, representing the sum of the stimulated channels. a) Schematic of the droplet circuit using Or, High Activity and the postulated And, Repeater, Diode and Inhibition droplet types. b) The three dimensional implementation of the network design is shown with red droplets denoting responsive medium, while the white and blue tones represent the excited and refractory states.

might not be the ideal application for droplet based computing, the simplicity of writing down binary logic formulas might lead to some important applications such as smart drugs, which could already benefit from implementing even simple logics as hypothesized in Chapter 3.

As a simple example of a larger sized droplet system using binary logic, the “input counter” network is shown in Figure 2.10. It encodes inputs and outputs to and from the network through the absence or presence of excitations. It discriminates between zero, one, two or three inputs that are stimulated independently. In response, it should always stimulate exactly one out of four possible output lanes maximally, dependent on the number of activated inputs. The connectivity of the network is displayed in Figure 2.10a and leads to the 3d structure of the network that is displayed in Figure 2.10b when fed into the simulator. Noticeably the schematic graph uses droplet connections that cross over, effectively necessitating a three dimensional implementation of the droplet system that we cannot yet produce experimentally.

Chapter 3

Evolution of Droplet Computers and Signals

In this chapter, evolutionary algorithms [Fog66, Rec71, Sch75, Hol75, Koz89, Fog94, Bey02, Wei02, Eib08] are considered as a mechanism to generate droplet network designs as well as to infer adequate symbol representations when building logic gates or pattern discriminators from droplet networks. Given an optimization problem, an evolutionary algorithm selects good individuals in a population of solutions that changes over time via genetic operators. Starting from a randomly generated population and guided by the fitness function, after several generations, the evolutionary algorithm returns an approximating solution to the problem. The use of evolutionary algorithms to design logic gates and circuits has been studied in various contexts, e.g., in the context of genetic programming [Koz89], evolvable hardware [Mil00a] and Cartesian genetic programming (CGP, [Mil00b]). For generating artificial neural networks, evolutionary computation was already used to find suitable parameters and structures [Fog90, Bee92, Cli93, Yao97]. Also in the context of excitable media, 2D cellular automata have been evolved to fulfill binary logic fitness functions [Sto08]. These two dimensional designs are related to the mostly planar droplet network graph structures that can so far be generated in laboratory experiments, as reviewed in Chapter 2.

Here, it is not the aim to generate a single droplet network design that could act as a universal computer, solving any kind of computable problem. Instead, it appears feasible and useful to build droplet devices that compute results for particular problem instances in the sense of Zauner's anti-universal machines [Zau96]. Nonetheless, given a problem instance, input data needs to be specified in some way. This could either happen through the initial state of the droplet system or during the run time, most probably through external stimulation of certain droplets. As outlined in Chapter 1 in Figure 1.4, it is an important design decision which encoding, i.e. which abstraction and concretization function, is used to feed inputs into the droplet network. Most probably the optimal encoding will depend on a number of factors like the type of task, the number of used symbols, parameters of the computing substrate, the applied quality measure, and how much computation can be done outside the droplet network to generate and to interpret the encoding. Finding the symbol encoding together with a suitable droplet network will be the focus of the following Section 3.1 that coarsely follows the publications [Gru12, Esc13]. Then, the later Section 3.2 will summarize finding an implicit generative encoding [Hor01] for the physical design of a droplet network with a fixed input encoding, resulting from Alexandra K. Diem's Master's thesis [Die12a].

3.1 Network - Symbol Co-Evolution

Several evolutionary computation approaches are investigated as a mechanism to "program" networks of excitable chemical droplets. For this kind of system, a task is assigned while concentrating on the characteristics of the signals representing the input and output symbols. Given a Boolean function as target functionality, 2D networks composed of 10×10 droplets were considered in our simulations. Three different set-ups were tested: Evolving network structures with fixed on/off rate coding signals, co-evolution of networks and signals, and network evolution with fixed but pre-evolved signals. Evolutionary computation served in this work not only for designing droplet networks and input signals but also to estimate the quality of a symbol

representation: It is assumed that a signal leading to faster evolution of a successful network for a given task is better suited for the droplet computing infrastructure. Results show that complicated functions like XOR can evolve using only rate coding and simple droplet types, while other functions involving negations like the NAND or the XNOR function evolved slower using rate coding. Furthermore, symbol representations were discovered that performed better than the straight forward on/off rate coding signals for the XNOR and AND Boolean functions. To conclude, this approach is suitable for the exploration of signal encoding in networks of excitable droplets.

While it is clearly possible to rebuild basic and combined logical gates in excitable chemical media [Hol11b], this might not necessarily use the capabilities of unconventional computers to their greatest extent [Gen12, Ste12]. Nonetheless, in this section, we will evolve droplet networks fulfilling basic logical functions because of their simplicity while exploring the impact of different input symbol encodings. So the focus is not mainly on the evolved functionality but rather on the varying difficulty of the evolution process. Additionally we also investigate the symbol representation in order to discover an adequate and efficient interpretation for them.

Since we cannot influence the amplitude of the excitation spikes, a list of times at which we excite particular droplets should contain all the information that is available to the computing droplet system. Nonetheless, different features of this list of excitations might be of more or less importance. From neuroscience we know for example the coding techniques rate coding, population coding and temporal coding [Bro04]. In the case of rate coding, the (averaged) oscillation frequency is used to distinguish different meanings while the exact timing of the spikes would be ignored. Population coding on the other hand would mean that the activity of different sub-populations of droplets denoted different meanings. For temporal coding, the precise timing differences between excitation spikes are utilized as information carrier. These coding schemes might be candidates for excitable droplets as well.

3.1.1 Methods

Self-Exciting Droplets

In this study we use the droplet simulation model that was explained in Section 2.4.4 with the following parameters: normal droplets d_{Norm} as well as input and output droplets are modeled with an expected oscillation period of 16 s, which is composed of 10 s responsive time τ_{res}, 1 s excited time τ_{ex} and 5 s refractory time τ_{ref}. Signal propagation delays τ_{prop} are 1 s. The exact timing parameters for each phase are sampled using normal distributions with a standard deviation of 0.05 s around the mean values given before. Less excitable droplets d_{LowEx} use the same timing distributions but require at least two adjacent droplets to be excited at the same time to trigger an excitation.

We also use variations of the model for less excitable droplet types, such that at least two concurrent excitations are necessary to trigger a droplet into an earlier excitation. Furthermore, to allow for a richer dynamic behavior, we decided to include one more droplet type that would oscillate slower. The different oscillation period can be achieved by differently composed BZ mixtures. In this case, all timing mean values as well as the standard deviations are multiplied by an arbitrary factor of $\frac{3}{2}$.

Droplet Networks

We perform *in silico* experiments of droplet networks in a 10×10 grid of simulated droplets that are connected in a Moore neighborhood of radius one, such that all directly adjacent cells can excite each other as explained in Section 2.4.4. These geometric properties of the networks were chosen based on the size of networks that can presumably be achieved experimentally by our collaboration partners in the near future. Up to four different kinds of cells are used, which represent empty cells, normal droplets, droplets of lower excitability and droplets with longer oscillation periods. Furthermore, there are two fixed input droplets and two fixed output droplets defined on the network grid. They can be used to dynamically feed a stream of excitations into and out of the droplet network. The positions of the input and output droplets

are fixed to arbitrary values, coarsely in the middle of the left and right hand sides of the grid, as visualized in Figure 3.1 (a).

Signal Encoding

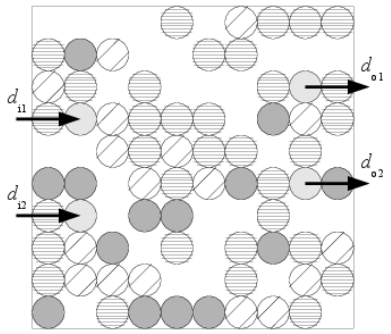
When representing binary signals by rate coding, we stimulate droplets as much as possible for a symbol '1' and not at all for a symbol '0'. When droplets are maximally stimulated, the oscillation time will be $\tau_{ex} + \tau_{ref} = 6$ s. Normal droplets that are left alone do not stop oscillating but their frequencies are lower with periods of $\tau_{ex} + \tau_{ref} + \tau_{res} = 16$ s.

To allow more complex symbol representations, we use a timing pattern that determines which input droplet is stimulated from the outside at which times as visualized in Figure 3.1 (b). We divide the length T of the stimulation pattern up into m small intervals $\{I_1 ... I_m\}$, each of the length $\triangle t = \frac{T}{m}$. Hence, interval I_j is defined between the times $(j-1) \cdot \Delta t$ and $j \cdot \Delta t$. Considering a single droplet only, we define a pattern $S_{(1)}$ as a Boolean vector, which states if the droplet is stimulated in the interval I_j or not. To describe meaningful symbols, Δt should be small in comparison to a droplet's oscillation period, resulting in a fine temporal resolution. Meanwhile, the total length T of the symbol should probably be long in comparison to the droplet's oscillation to allow symbols to consist of more than a single excitation.

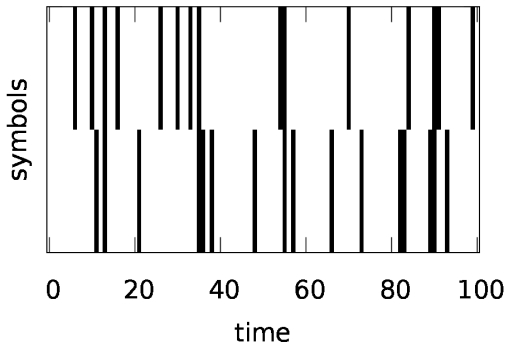
$$S_{(1)} = (a_{I_1}, a_{I_2}, ..., a_{I_m})$$

$$a_i \in \{0,1\}, \quad S_{(1)} \in \{0,1\}^m, \quad m \cdot \Delta t = T$$

Because typically more than one input will be used, multiple droplets will have to be stimulated, e.g., both inputs for an XOR gate. Furthermore, thinking about population coding, a single symbol like a logical '1' could affect multiple droplets with individual stimulation patterns. In contrast, for the sake of redundancy, a common stimulation pattern might be supplied to multiple droplets. Here we use the notion of the droplet channel c_i to signify a set of droplets that receives the same stimulation pattern $S^{c_i}_{(1)}$. Two droplet channels c_i and c_j could now either be used as components



(a) Droplet Network



(b) Input Symbol Representation

Figure 3.1: Illustration of network- and symbol individuals used in the evolutionary algorithm: (a) Rendering of an evolved 10×10 droplet network instance. Each square represents a droplet on a two dimensional array. Not all positions of the array are filled with droplets. Horizontally, diagonally and dark gray circles represent normal, less excitable and long period droplets, respectively. The input droplets (d_{i1}, d_{i2}) and output droplets (d_{o1}, d_{o2}) on arbitrarily fixed positions are indicated by arrows. Touching droplets can excite each other, defining the connectivity for the droplet simulation. (b) Example of two symbols that evolved together with a network instance to realize the XOR function. The lower row of the image represents symbol '0' while the upper row represents symbol '1'. Time advances left to right over 100 frames with a time step of 0.5 s, leading to a total length of 50 s per symbol. The input droplets are stimulated only in the intervals that are represented by black vertical bars and are left alone where the white vertical bars are rendered. The symbols are fed into the droplet network repeatedly, recapitulating the stimulation pattern every 50 s. At least three oscillation cycles are completed per symbol repetition because the simulated droplets' self-excitation periods are around 16 s. Since droplets are modeled with refractory times, not every white stimulation bar will actually lead to an excitation in the droplet but can as well be disregarded in the droplets refractory phases, especially when two excitations follow each other closely.

		Input			
	Task	**0 0**	**0 1**	**1 0**	**1 1**
	Identity	0 0	0 1	1 0	1 1
	OR	0	1	1	1
Expected	AND	0	0	0	1
Output	NAND	1	1	1	0
$\tilde{o}_{cp}$	XOR	0	1	1	0
	XNOR	1	0	0	1
	Half-adder	0 0	1 0	1 0	0 1

Table 3.1: Boolean functions that were used as fitness criteria in evolution. Two input and up to two output channels were used.

of a single symbol or as independent inputs. Nonetheless, in the experiments shown in this work, a symbol will only consist of a single droplet channel.

For stimulation patterns that are composed of many channels $C = \{c_1, c_2, ..., c_{|C|}\}$, we can extend the pattern definition $S_{(1)}$ to an array $S_{(|C|)}$ that stores the activation state a_{c_i,I_j} of each channel $c_i \in C$ for each interval I_j:

$$S_{(|C|)} = \begin{pmatrix} a_{c_1,I_1} & a_{c_1,I_2} & \cdots & a_{c_1,I_m} \\ a_{c_2,I_1} & a_{c_2,I_2} & \cdots & a_{c_2,I_m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{c_{|C|},I_1} & a_{c_{|C|},I_2} & \cdots & a_{c_{|C|},I_m} \end{pmatrix}$$

Task Definition

To evaluate the quality of a droplet network and of different symbol encodings, we define Boolean functions that should be fulfilled in terms of their truth tables. As displayed in Table 3.1, we tested seven different functions with up to two input and output channels.

Fitness Evaluation

Ultimately, the aim of these experiments is to find symbols that can be used by the network internally as input as well as for output. But to evaluate the fitness of a droplet network for binary operations using arbitrary symbols, a metric that determines the similarity between an input symbol and a recorded output excitation stream would be necessary. As discussed later in Section 3.3, choosing an appropriate metric is not trivial. Consequently, we are evolving complex symbol representations to feed into the network but we do not yet expect the network to reproduce these complicated symbols as outputs. Instead we use simple rate coding for the outputs: high activity is interpreted as symbol '1' and low activity as symbol '0'. Here again, as for the rate coding input, high activity means droplets are entering the next oscillation cycle very shortly after leaving the refractory phase, resulting in a high spike frequency. Low activity, in contrast, means that droplets are rarely triggered into early excitations by their neighbors and mostly self-excite, resulting in a low spike frequency.

The evaluation is divided into distinct phases p by assigning each combination of input symbols to one phase, resulting in four phases for two binary inputs. Then, for each phase, we analyze the output droplet channels, i.e., the activity on the designated output droplets, for their similarity to an expected output: for each phase p, the system is simulated with the appropriate input signals for a fixed time and the number of received excitations at the droplets of output channel c are stored in o_{cp}. We denote the maximal and minimal counted peak numbers as o_{max} and o_{min}. The symbol that is expected at the output droplets for the channel-phase pair (c, p) is referenced as $\tilde{o}_{cp} \in \{0, 1\}$ instead.

The final fitness F is influenced by two different aspects, F_1 and F_2, of the output behavior. First, the normed difference between highly activated and less activated channel-phase pairs should be maximized to allow some kind of discrimination. We define the difference between the maximum and minimum peak numbers divided by the maximum peak number as F_1. F_1 is zero if all peak numbers are equal and at most one when the minimum value is zero. Second, the truth table should be fulfilled, leading to a function F_2. Here, the worst case channel-phase pair defines the overall

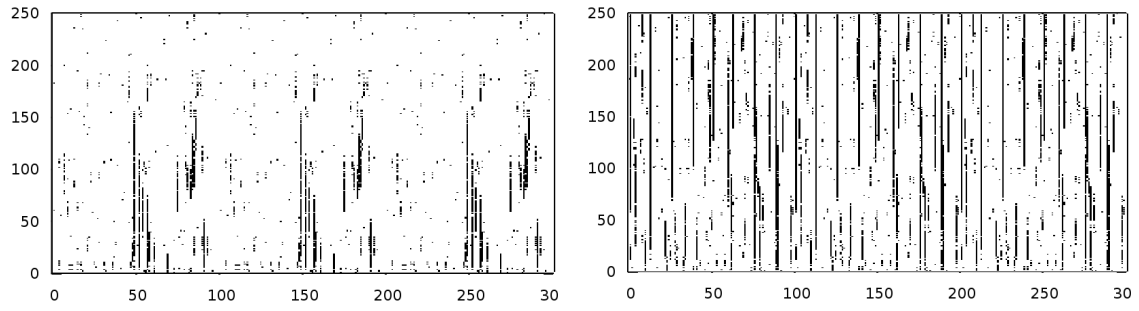
fitness. Each channel-phase pair peak number should lie as close as possible to the minimum or maximum peak number, dependent on the expected output $\tilde{o}_{cp}$. Finally, if a minimum discriminability is exceeded and also the Boolean function is fulfilled, the distance between minimum and maximum rates should further be expanded.

$$F = \begin{cases} F_1 & \text{if } F_1 < 0.2 \\ F_2 + 1.0 & \text{if } F_1 \geq 0.2 \text{ and } F_2 < 0.9 \\ F_1 + 2.0 & \text{if } F_1 \geq 0.2 \text{ and } F_2 \geq 0.9 \end{cases} \tag{3.1}$$

$$F_1 = \frac{o_{max} - o_{min}}{o_{max}} \tag{3.2}$$

$$F_2 = \min_{c,p} \begin{cases} 1 - \frac{o_{cp} - o_{min}}{o_{max} - o_{min}} & \text{if } \tilde{o}_{cp} = 0 \\ \frac{o_{cp} - o_{min}}{o_{max} - o_{min}} & \text{if } \tilde{o}_{cp} = 1 \end{cases} \tag{3.3}$$

Experimental Set-Up



(a) Symbol '0' (b) Symbol '1'

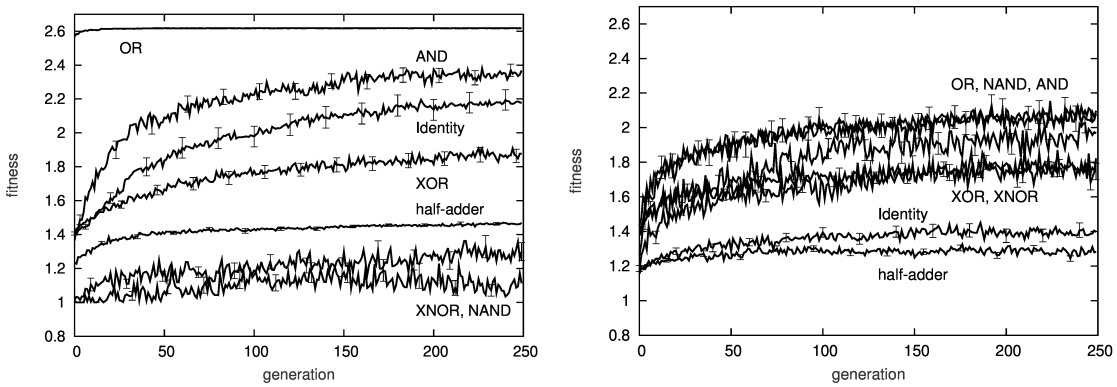
Figure 3.2: Evolutionary trajectory of two symbol representations over 250 generations co-evolution with a droplet network. The y-axis denotes the evolutionary generation while the x-axis represents the stimulation interval for each fitness evaluation similar to the signal plot in Figure 3.1(b). The regularities that can be observed along the x-axis in both graphics are not evolved regularities but result from the repetition of the pattern: As the pattern of 100 intervals is fed into the simulator during fitness evaluation in a repeated manner, three repetitions of the input signal are plotted over 300 time frames.

We employed an evolution strategy of the type $(8/2, 30) - ES$, meaning a comma strategy with 8 parents and 30 children, running for 250 generations where the parents of each generation are discarded. Two parents are recombined to produce each child. The best symbol representation of each generation of a single experiment is displayed in Figure 3.2. For each experiment, we ran a batch of 50 evolutionary optimizations to build mean values. In total, we conducted 35 experiments for all the combinations of the seven target functions from Table 3.1 and the five experimental variations: Network only evolution with three or four droplet types, network and signal co-evolution with three or four droplet types and network only evolution with pre-evolved symbol representations. The symbol representation for the pre-evolved signals was taken from the co-evolution experiment that achieved the best fitness. Using four droplet types means using empty droplets, normal droplets, less excitable droplets and long period droplets, while the latter is discarded for the three droplet type experiments.

For mutating the droplet network, the probability of switching an arbitrary position is 0.05. When using four droplet types, the probabilities for changing to an empty cell, to a normal droplet, to a low-excitability droplet and to a long-period droplet are 0.4, 0.4, 0.1 and 0.1 respectively. For the runs without the long period droplet type, the remaining probabilities read $\frac{4}{9}$, $\frac{4}{9}$ and $\frac{1}{9}$. Single point crossover recombination is applied with an uniformly chosen position in the row-by-row linearized representation of the droplet network. For the input signal, the probability of switching an arbitrary position is 0.025. When a mutation occurs, the probability for generating a '1' is 0.1 while a '0' is generated with probability 0.9. Single point crossover recombination is applied with an uniformly chosen position.

3.1.2 Results

Small droplet systems of up to 100 droplets were arranged by means of evolutionary algorithms to satisfy the Boolean functions Identity, OR, AND, NAND, XOR, XNOR and half-adder. Based on differentially fast fitness increase, some target functions are easier to evolve than others (cf. Figure 3.3). As observed in [Ada09], the reason for



(a) Network only Evolution (b) Network and Signal Co-evolution

Figure 3.3: Average fitness of population's best individual over 50 experiments for evolving different target functions from Table 3.1. Error bars indicate the standard error of the mean. Generally, all fitness values are lower for the signal and network co-evolution because of the higher dimensional search space. Exceptions are those functions that benefit from a simple swapping of rate coding signals, i.e. the NAND and XNOR functions.

this is partially the different problem complexity and partially the properties of the computing substrate that favor and disfavor certain kinds of tasks. In the case of our droplet computing, using rate coding only, the OR and AND functions evolve fastest, followed by Identity, XOR, the half-adder, the XNOR, and the NAND function. Nonetheless, there is a strong qualitative transition between the XOR and the half-adder function. The mediocre fitness of the XOR network is based on the some few evolution runs that produced high fitness XOR networks and many non-functional ones. The XNOR and NAND evolutions using rate coding as well as the half-adder with any coding on the other side did not lead to a single evolution run producing a functional network.

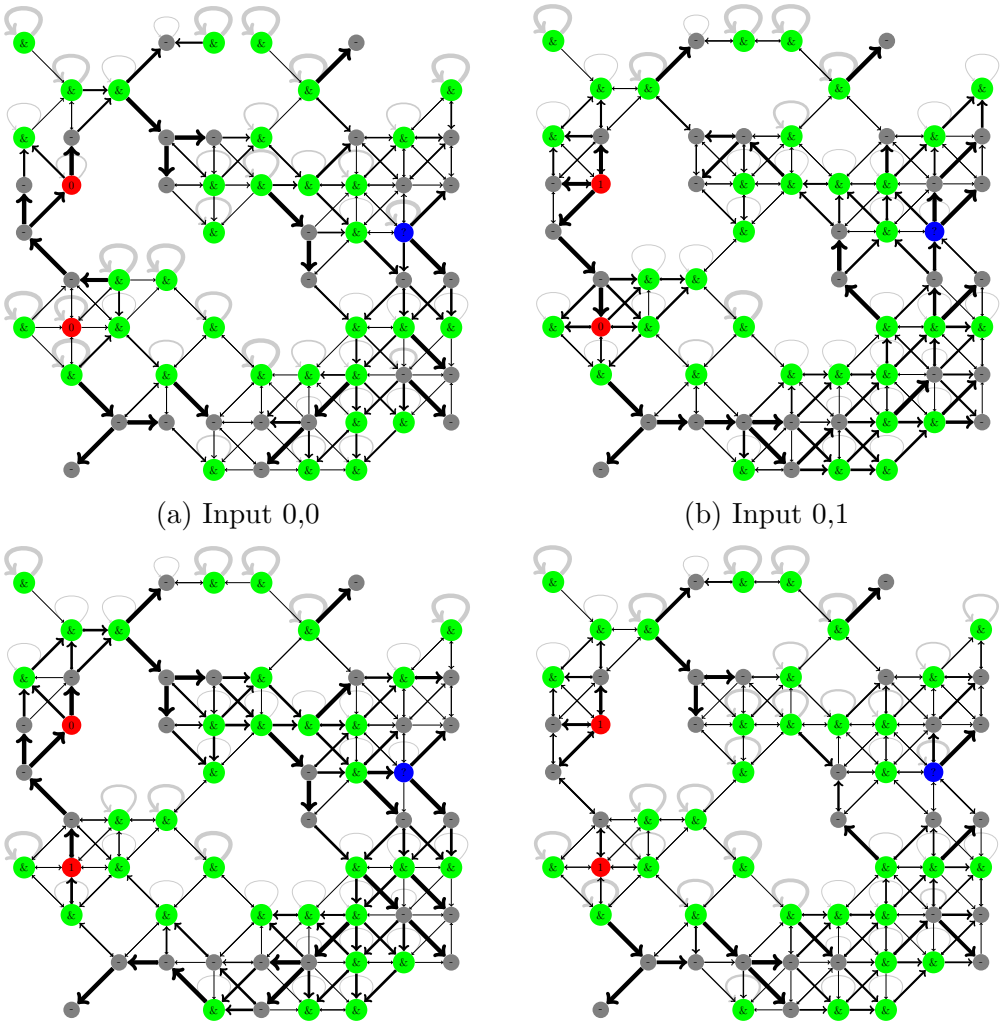
Despite these difficulties, even a complicated function like XOR was evolved, even for single channel rate coding signal inputs, albeit not as fast as a simple OR or AND function (cf. Figure 3.4). Interestingly, the identity function, meaning a mere connection between both inputs and outputs, is not a simple task compared to AND or XOR when co-evolving input signals (cf. Figure 3.3). Apparently co-evolving networks and symbol representations for the identity function is almost as hard as

evolving the half-adder. While using rate coding, in contrast, the identity function evolved faster than the XOR function. Evolution with and without the third droplet type with long oscillation periods did not significantly change the speed or final quality of the evolution process.

A network successfully implementing the half-adder functionality did not evolve in our experiments so far. The reason for this is most probably the difficulty of crossing over two connections in the two dimensional lattice of droplets. A half-adder network could be implemented by an XOR gate together with an AND gate, each of which evolved comparatively easy. But to construct the half-adder from the two gates, both inputs would have to be available to each of the gates. Thus at least one of the input signals would need to cross over another one. While we do not exclude the possibility of a signal crossing over another one given a suitable construction of a droplet system and a fitting symbol encoding, we did not observe such a system in our experiments. At least when trying to evolve a rate coding identity function with two inputs and two outputs while crossing over the outputs, the fitness dropped dramatically, such that no satisfying solution has evolved.

Shown in Figure 3.5, at least in the case of the AND and XNOR functions, pre-evolved signals exist (cf. Figure 3.6) that are clearly leading to a faster evolution of droplet networks than simple rate coding. Here droplet networks and signals were originally co-evolved. Then, one of the best evolved symbol representations was used consistently through a full network-only evolution run.

The evolved symbols look similar (cf. Figure 3.6 (a)) to rate coding signals but most probably allow for a better synchronization of arriving spikes. While a single activation peak remained for the '0' symbol, it had no obvious influence on the fitness of the symbol. The synchronization of spikes seems to be important considering that a low excitability droplet is only activated by other droplets, when two spikes arrive in a narrow time window. The length of the window used in our experiments was one second. So while a constant activation, using rate coding symbols, leads to the highest frequency of spikes in the input droplets, the phase of both input droplet oscillations can randomly drift and is dependent on the initial conditions.



(a) Input 0,0 (b) Input 0,1

(c) Input 1,0 (d) Input 1,1

Figure 3.4: Activity plot of an evolved XOR network using different stimulations at the red input droplets on the left side of the network. The nodes in the network represent input droplets (red), output droplets (blue), normal droplets (gray) and less excitable droplets (green). The arrows in the picture describe the impact of other droplets for the excitations of each droplet. While a strong loop arrow on top of a droplet means that the droplet mostly self-excited, an arrow from a neighboring droplet means that it was excited by this neighbor many times. While an input of (0,1) or (1,0) leads to a cyclic propagation of pulses through the network in either clockwise or counter-clockwise direction, no excitation at all (0,0) or full stimulation (1,1) results in pulses that propagate from the left to the right. The cyclic propagation results in a higher total spike rate arriving at the blue output droplet as thinner self-excitation loop on top of the output droplet can be observed. Hence, the droplet network fulfills the function of an XOR gate when used with rate coding inputs.

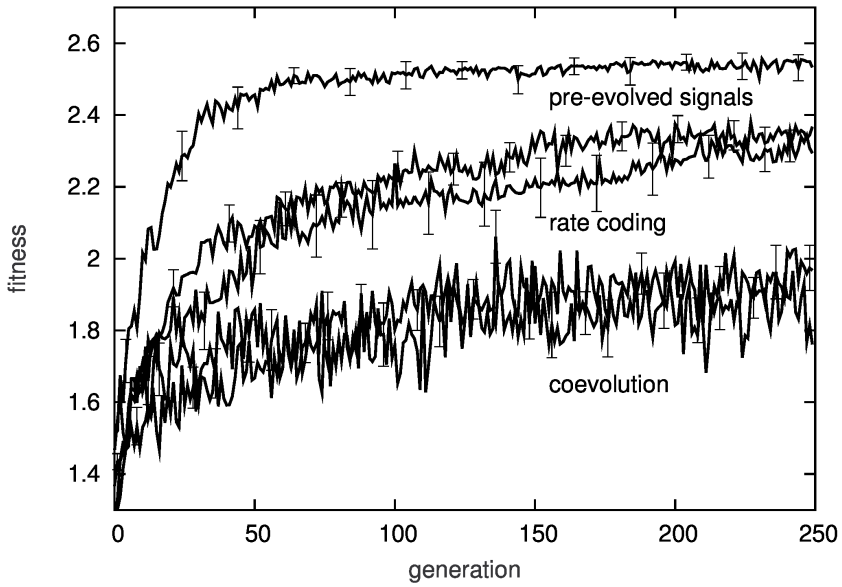
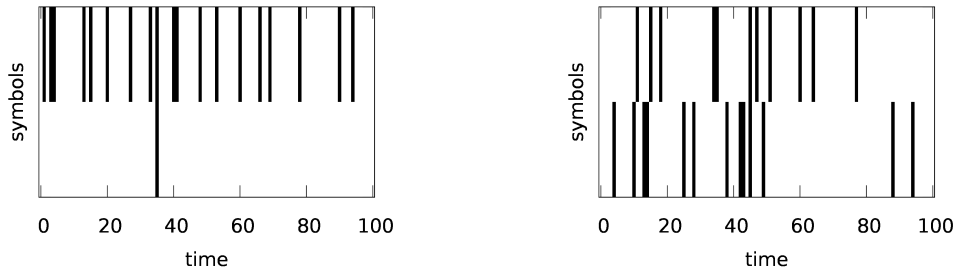


Figure 3.5: Average fitness of populations' best individual over 50 experiments for evolving the AND function using rate coding, co-evolution and pre-evolved symbol representation. Error bars indicate the standard error of the mean. For the rate coding and co-evolution experiments, two curves are plotted: The corresponding simulations ran with and without the long period droplet types, but no significant difference was observed.



(a) Evolved AND Symbols (b) Evolved XNOR Symbols

Figure 3.6: Evolved symbol representations for the AND and the XNOR functions that performed better than rate coding. (a) While the AND symbol looks very similar to rate coding symbols, there is one peak included for symbol '0' that might serve as a helper for synchronization. (b) For the XNOR signals, both symbols are represented by a series of about 30 seconds activation followed by ca. 20 s rest. The difference between both symbol representations could be either in the shift of the active phases of about 10-20 s or in the exact pattern of each signal.

When using a slightly lower activation rate instead, the phases of both input droplets are controlled by the stimulation, leading to a higher number of concurrent spikes arriving at low excitability droplets. This, in turn, leads to a higher influence of the low excitability droplets on further droplets in the network. We tested an evolved droplet network and stimulated it with a rate coding symbol, a co-evolved symbol and with an additionally engineered symbol. The engineered symbol includes no stimulated at all for symbol '0' and regular spike every seven seconds for symbol '1'. With this spike pattern, the engineered symbol reaches very similar input and output average spike rates compared to the evolved symbol. The measured spike frequencies are summarized in Table 3.2.

A further extreme rise in evolution efficiency was observed for the NAND function. However, this is most probably only due to a crosswise substitution of the signals for symbols '0' and '1', such that the problem is reduced to a rate coded OR function. Functions that involve a mapping from symbol '0' to low activity like the XNOR, and NAND functions seemed more difficult with pure rate coding. This problem

	Average Spike Frequency [spikes/second]		
	Rate Coding	Evolved Symbol	Engineered Symbol
Symbol '0' Input Droplet	0.067	0.067	0.067
Symbol '1' Input Droplet	0.167	0.140	0.143
Supposedly Active Output Droplet	0.086	0.139	0.138
Supposedly Inactive Output Droplet	0.064 - 0.067	0.64	0.064 - 0.066

Table 3.2: Summary of the spike frequencies measured in an evolved AND network when stimulated with rate coding, evolved and an engineered symbol input. The engineered symbol did not produce any stimulation for symbol '0' and a regular spike pattern of a spike every seven seconds for symbol '1'.

of inverting signals should easily be resolved when using multi-channel symbol representations that would supply a high and a low activity channel for each symbol. Problems that did not benefit significantly from pre-evolved symbol representations were the OR, the XOR, the Identity function and the half-adder. Nonetheless, the pre-evolved symbols never led to worse evolution trajectories in our experiments.

3.2 Self-Assembly of Droplet Computers

Using the architecture of computing droplets that was introduced in Chapter 2 and the idea of DNA single strands attached to lipid vesicles to mediate the vesicles' interactions [Had10], we studied how to derive simple self-assembly *programs* p_{f_c} for computing droplet networks via evolutionary algorithms [Fog66, Rec71, Sch75, Hol75, Koz89, Fog94, Bey02, Wei02, Eib08]. Here we use an implicit encoding of the droplet networks that should combine the potential advantages of generative encodings [Hor01] with the long-term objective of finding computing structures that can self-assemble without external help that would place specific droplets in specific

positions. Most parts of these results were generated in the scope of the Alexandra K. Diem's Master's thesis [Die12a] and presented at the ECCS conference [Die12b].

Following the notation from Section 1.3 in Figure 1.4, the abstract computation f_c that should be realized here is a classification task of a real-valued input vector from $\mathbb{R}^8$ to a binary *malign/benign* state. Training and verification data is taken from the "Breast Cancer Wisconsin" data set of the *Proben1* benchmark problem set [Pre94].

Such a computation f_c will be implemented by first assembling a two dimensional network of computing droplets following an evolved set of rules, by then simulating the resulting droplet network in our simulation environment *DropSim* as explained in Section 2.4.4 and by finally interpreting the activity in a particular droplet as the frequency coded output. The implementation of the computation, p_{f_c}, is specified by the set of droplet types, their quantities and the list of possible interactions. Hence the assembly step is tightly interwoven with the actual signal processing function of the network.

The assembly of a network N is conducted on an $m \times m$ rectangular grid, where $m = 20$, from a given set of droplets $\mathcal{D}$ and the empty droplet d_ϵ as explained in Section 2.4.4. This leads to the huge search space of $|\mathcal{D} + 1|^{400}$ possible networks.

$$N \in (\mathcal{D} \cup d_\epsilon)^{(20\times 20)}$$

An instance of these networks is chosen by "growing" a network from a random seed droplet $d_0 \in \mathcal{D}$ and by then successively adding adjacent droplets d_i from $\mathcal{D}$. The assembly is controlled by an evolved table of rules $A \in \{0,1\}^{|\mathcal{D}|\times|\mathcal{D}|}$, where $A_{ij} = 1$ iff droplet type d_i can attach to droplet type d_j. Because $A_{ij} = A_{ji}$, the size of the genotype is then $\frac{1}{2}\ |\mathcal{D}|\ (|\mathcal{D}| + 1)$. An exemplary instance of a set of assembly rules and the resulting droplet network is shown in Table 3.3 and Figure 3.7. Please note that, even if $A_{ij} = 0$, the final graph may contain adjacent droplet types d_i and d_j, if other droplets in the proximity lead to the attachment of d_i or d_j.

After the assembly, the grid of droplets is simulated in our droplet simulation environment *DropSim* as explained in Section 2.4.4. Stimulation patterns from the

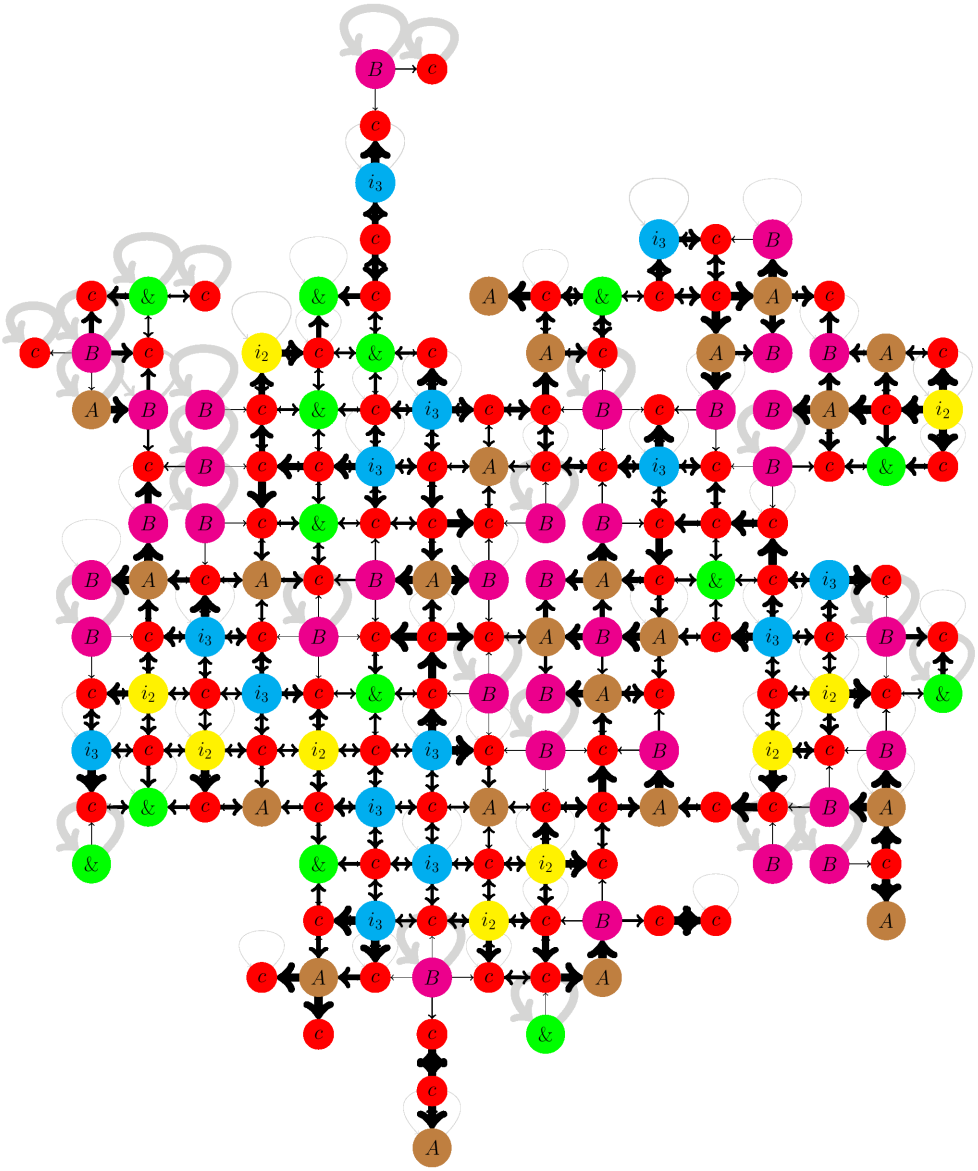


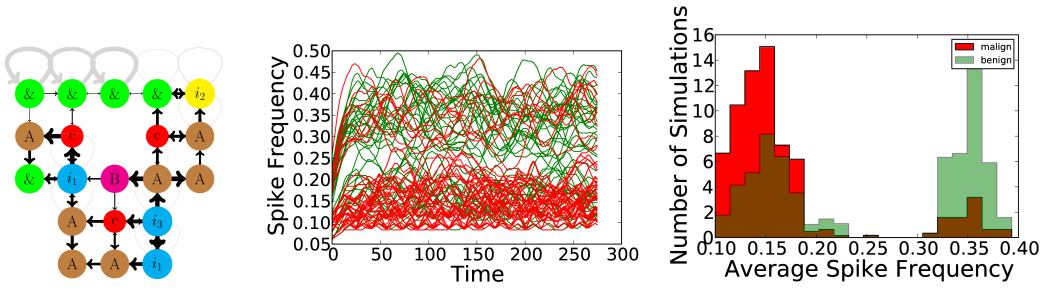
Figure 3.7: Example of a set of assembly rules and the resulting grown network structure. The Arrows indicate the typical flow of excitations from one droplet to another. Gray loops indicate that a droplet was self-excited before another excitation from an adjacent droplet triggered it. The thickness of the arrays and loops indicates the probabilities for each type excitation.

	d^{Norm}	d^{DirA}	d^{DirB}	d^{LowEx}	d^{In0}	d^{In2}	d^{In2}
d^{Norm}	0	1	1	1	1	1	1
d^{DirA}		1	0	0	0	0	0
d^{DirB}			0	0	0	0	0
d^{LowEx}				0	0	0	0
d^{In0}					0	0	0
d^{In1}						0	0
d^{In2}							0

Table 3.3: Assembly rules for an exemplary individual. The different droplet types fulfill the following functions. CON: connector droplet, simple forwarding of signals, DIR: one-way forwarding of signals (direction A or B), AND: forwarding of signals only if two signals arrive at the same time, IN: input droplets (types 0, 1, 2).

Proben1 data set are fed into the system and the system's final fitness is evaluated dependent on the rate-coded proximity to the expected output of the classification task. Out of the nine inputs, the third, fourth and seventh were selected by principal component analysis [Jol02] to reduce the number of input droplets. For reading out the results, instead of fixing a particular droplet type or position in the grid as the output, we evaluate each droplet for being a potential output node. The averaged fitness of the fittest node is then reported as the final network fitness. Similar to the fitness function from Section 3.1.1, each node's fitness is evaluated by counting the number of spikes per simulation in comparison to the other input cases. The fitness then is the fraction of input cases that can be correctly classified using a threshold spike frequency o_{th}.

From the experiments we learned that networks can be assembled that are relatively fit to calculate our classification task f_c with an accuracy of ca. 72% as shown in Figure 3.8. The winning network is relatively small when compared to many other networks on the 20×20 grid as shown in Figure 3.7. Furthermore, a given genotype, i.e. a set of assembly rules, is not guaranteed in our approach to generate fit network instances only. Further experiments will have to show the fitness distribution of grown networks for a given set of assembly rules.



(a) The assembled network. (b) Spike activity time course. (c) Average spike frequency.

Figure 3.8: Best evolved and assembled network design for the cancer data set: In panel (a), the assembled network is displayed together with the arrows indicating the predominant direction of signal propagation. Differently colored droplets indicated different chemical compositions as explained in Section 2.2. The top right yellow droplet is used as the output droplet for the next panels. Panel (b) shows the spike activity time courses of different simulation experiments using different stimulation patterns from the malign (green) as well from the benign (red) classes. Summing up the the spike activity for each experiment, the average spike frequency histogram for both output classes is shown in Panel (c).

	$fr < 0.16$	$fr \geq 0.16$	Σ
benign	869 (TN)	251 (FP)	1120
malign	172 (FN)	258 (TP)	430
Σ	1041	509	1550

Table 3.4: Confusion Matrix showing the true and false positives and negatives dependent on the average spike frequency fr. $fr < 0.16$ is interpreted as a malign signal (positives), $fr \geq 0.16$ as a benign signal (negatives). This results in an accuracy of 0.72, a sensitivity of 0.6, a specificity of 0.77 and a precision of 0.5. We calculated the accuracy as (TP+TN)/(TP+TN+FN+FP), the sensitivity as TP / (TP+FN), the specificity as TN / (FP+TN) and the precision as TP / (TP+FP) .

3.3 Discussion

In this chapter, the evolutionary design of droplet networks and appropriate signal encodings was exemplified. To this end, droplet networks were evolved that show the functionality of Boolean AND, OR, NAND, XOR, XNOR and Identity gates, while the evolution of a complete half-adder network did not succeed. We demonstrated, that the evolution of some of the functions, i.e., XNOR and NAND networks, is only possible when symbol encodings are co-evolved together with the network topology. We showed that the suitable co-evolved symbols, as compared to naive rate-coding symbols, simplify the task of finding a droplet network structure for Boolean gates, especially in the case of an AND network.

Besides designing droplet network structures and symbol encodings, evolutionary algorithms also served another purpose in this work: To some extent, evolutionary algorithms also offer a measure of complexity, telling us whether a problem is simple or hard to solve [Ada09]. Or, given two distinct symbol encodings, which of them makes searching for a solving network structure easier.

A straight forward construction of two adequate symbols, representing '0' and '1', might be to maximize the distance between them. The problem here is to define the distance metric that would heavily influence the result of the maximization. Ideally these experiments would only depend on the properties of the computing substrate itself and not on arbitrary definitions that are put in from the outside. But most metrics like the Hamming distance or the spike train similarity measures from neuroscience research [Dau09] seem sensible but artificial with respect to the computing droplet substrate. A meaningful alternative would be to run a nested evolution of a droplet network simulation as distance metric - the easier it is to evolve a network that discriminates both signals, the larger the distance between both symbols. Still, the computational efforts for a single evaluation of this kind of fitness function were immense. This led us to the different approach of co-evolving signals and droplet networks for simple binary problems at first. In Chapter 4 then, information theory will be proposed as useful metric for distinguishable symbol encodings.

Even though simple logic functions were evolved here, the automatic construction of larger, more complex systems might be hard, especially when fitness functions cannot provide enough gradient for the optimization algorithm to follow. The "multi-step" fitness functions that we used in Equation 3.1 tries to focus on different aspects of generating the network functions at different times, dependent on how close to perfect the solution is. Additionally, since it is not generally possible to find all non-dominated solution candidates by mapping multiple fitness criteria onto a single scalar value, using Pareto optimization for future experiments might be an option [Sch85, Zit04].

Generally the influence of the droplet network dimensions should be interesting - especially how few droplets can generate the sought-after behavior, what number of droplet species are essential, is there a preferential length for droplet signal patterns and how many input channels should be used per symbol? Also the aspect of robustness has not yet been in the focus of this chapter: Even with multiple simulation runs per individual, we could not completely eliminate the possibility that in some evolution runs high fitness scores were achieved accidentally, which could then not be sustained under different initial conditions and with noise.

Chapter 4

Information Theory Based Methods

In this chapter, general methods are presented that can be used to explore the information transformation process of a computing substrate, for example of a medium composed of oscillating self-exciting droplets as introduced in Chapter 2. Networks of Belousov-Zhabotinsky (BZ) droplets seem especially interesting as chemical reaction-diffusion computers because their time evolution is qualitatively similar to neural network activity. Moreover, such networks can be self-generated in microfluidic reactors. However, it is hard to track and to understand the function performed by a medium composed of droplets due to its complex dynamics. Corresponding to recurrent neural networks, the flow of excitations in a network of droplets is not limited to a single direction and spreads throughout the whole medium. In this chapter, the operation performed by droplet systems is analyzed by monitoring the information flow as explained in [Gru14] and [Giz16]. This is achieved by measuring mutual information and time delayed mutual information of the discretized time evolution of individual droplets. To link the model with reality, experimental results are used to estimate the parameters of droplet interactions. For example, an evolutionary generated droplet structure is investigated that operates as a NOR gate. The presented methods can be applied to networks composed of at least hundreds of droplets.

4.1 Introduction

Using the interactions between droplets, encoding information in excitation pulses, classical Boolean gate logic can be constructed in experiment and simulation [Ada02a, Hol11a, Hol11b], but also non-Boolean, e.g. graph-theory based [Ada11c] applications are being explored in the context of unconventional computing [Ada01, Gor05, Ste12, Ban13]. Most of these information processing devices were constructed using a bottom-up approach by a human designer and can in principle be assembled in microfluidic reactors [Thu13]. On the other hand, various networks of BZ droplets can be generated by evolutionary algorithms [Fog66, Rec71, Sch75, Hol75, Koz89, Fog94, Bey02, Wei02, Eib08] following a chosen fitness function, but then their information processing regime is not known [Esc13]. The methods described in this article can be used to predict the information processing potential of a given droplet structure or to suggest a structure that is more suitable for a specific task.

Due to the similarities between real and artificial neurons, we also resort to established analysis techniques known from neuroscience [Bor99, Bro04, Qui09, Vid11, Wib11] and other fields involving complex networks, e.g. atmosphere chemistry [Sol04], ecosystems [Sug12] or biological signaling networks [Pah08] to understand the reason for their computational efficiency and their modes of operation. In return, any innovation in understanding either of these systems might also be applied in the other complex networks related domains.

4.1.1 Challenges

Typically the dynamics of a possibly recurrent network with a vast amount of interconnected components are quite rich and complex [Maa02, Dit05]. This implies, that our common, intuitive, modular, engineering perspective of the system will not be successful any more [Ste12, Zau05b].

In contrast to human-engineered artifacts, in biological systems not even the purpose of a particular system is known, even less the process and function how such a purpose is realized. In biological systems, a network function is determined by evolution and

cognitive learning. In both cases this function is hidden in the network geometry and the interactions between its elements. In order to understand the system behavior we require a set of filters that show and highlight different aspects of the systems. Dynamical systems theory [Jet89, Wil10a] is one of these filters.

From a different perspective, computation equals at least for a deterministic process "information destruction" as the data processing inequality [Lat09] in information theory suggests. An algorithm p_f that transforms the signals S_{in}, representing the abstract inputs X, to the outputs S_{out} cannot generate new information that was not already present in the input data: $I(X, S_{in}) \geq I(X, p_f(S_{in}))$, where I is the mutual information between two variables. Consequently, computing might essentially be information destruction [Lan61, Ben82, Zau05a]. Hence, another useful filter for understanding computation that should be elaborated in this chapter is information theory and information dynamics, for identifying the key components of information processing: information transmission, modification and storage [Sha48, Sch00, Per05, Liz08, Wil10a, Wil10b, Ros14]. While finding the places in the network where information storage is exploited to produce results certainly is of key importance, this concept is not in the scope of this chapter. Instead, the focus is on information transmission and information modification here.

One further problem of unknown dynamic systems is to fix the symbol encoding [Fri14, Sha14] when the systems should be used for computation: Different kinds of symbol encodings can be more or less useful for information modification, storage and transmission operations in a particular system [Esc13, Gor14]. For example in case of Boolean operations in networks of BZ droplets, it sounds natural to assign a high spike frequency to the symbol '1' and a low spike frequency to the symbol '0' [Gor14]. Such information coding is simple and straightforward but the other encoding strategies, like exploiting more subtle differences in the time delay between two neighboring droplets or the activity patterns of a whole set of droplets can be more efficient. Nonetheless information theory allows to track information by abstracting away the particular encoding.

Furthermore, when measuring the dependency between any two droplets, the general

problem arises to distinguish between correlation that is generated because droplets are independently oscillating with the same frequency but without influencing each other, and on the other side, those droplets that are correlated because they are actively influencing each other's dynamics. Using simple mutual information, we would still measure a strong statistical dependency between unconnected droplets of the same frequency. As a trivial solution, we can combine measurements of many experiments with different initial conditions, such that a fixed phase shift dependency is broken. But especially in the case of chemical laboratory experiments, some of them are hardly repeatable. Also, the mutual information is still a symmetric value, such that we cannot determine the direction of the influence. For these reasons, we will use time-delayed mutual information [Fra86, Per05, Jin10] instead to determine the direction and time delay of the propagating signals, as explained in Section 4.2.3. An alternative would be to use transfer entropy [Sch00, Sta08, Wib11], that is measured when the past observations of a single variable do not suffice to explain its future trajectory and when, thus, a second variable's influence is required.

4.1.2 Overview on this Chapter

In this chapter, we show how to analyze an experimental five droplet system as well as simulated droplet systems in the order of magnitude of 100 droplets by following the information flow. This means on the one hand the information that is directly transmitted from one droplet to another by looking at the time-delayed mutual information between spike patterns in both droplets. On the other hand, we can investigate the mutual information between spike patterns in particular droplets and external values like specific input classes, their combinations as well as expected outputs of a computation.

In particular with the latter approach, we show how to highlight the synergy between different inputs with the mutual information to the expected output of a computation. Typically some positions in the droplet network start showing more information about the output than all their neighbors that supply this droplet with signals. In this case, information is combined and the synergy of the sources is exploited for computation.

4.2 Methods

The video data from experimental observations as well as simulation records can be analyzed with a unified information theory based analysis method. Our basic assumption is, that the times of spikes contain all information on the droplet activity, *i.e.* the amplitude of the spike and the behavior of the chemical medium between the spikes is irrelevant and abstracted away. Thus we extract the spike times from the video data of chemical experiments and, analogously, remove all remaining information that is available from simulation experiments except for the spike times.

To begin with, we will describe our chemical experimentation setup in the next Section, then our simulation of droplets and then the information theory based analysis method.

4.2.1 Experimental Droplet System

In the experiments, performed in cooperation with Konrad Gizynski and Prof. Jerzy Gorecki, we used commercially available analytical grade reagents without further purification. The concentrations of BZ reagents were: 0.3 M sulfuric acid (H_2SO_4), 0.375 M sodium bromate ($NaBrO_3$), 0.125 M malonic acid ($CH_2(COOH)_2$) and 0.04 M potassium bromide (KBr). In the experiment we used a mixture of catalysts. The bathoferroin ($[Fe(batho)_3]$), which concentration was 0.0015M played the major catalytic role. To make the medium photosensitive a small amount of ruthenium complex ($Ru(bpy)_3Cl_2$) (0.00021M) was added.

We controlled oscillations in the droplets by illumination with blue LEDs connected to a PC. Plastic optical fibers (1.5 mm diameter) attached to the diodes allowed to illuminate each droplet individually. Light emitting fiber tips were positioned centrally below the corresponding droplets to limit the amount of light transferred to their neighbors. The experimental system is drawn schematically in Figure 4.1. The light intensity was adapted during the experiments to obtain required frequency of oscillations for the droplets.

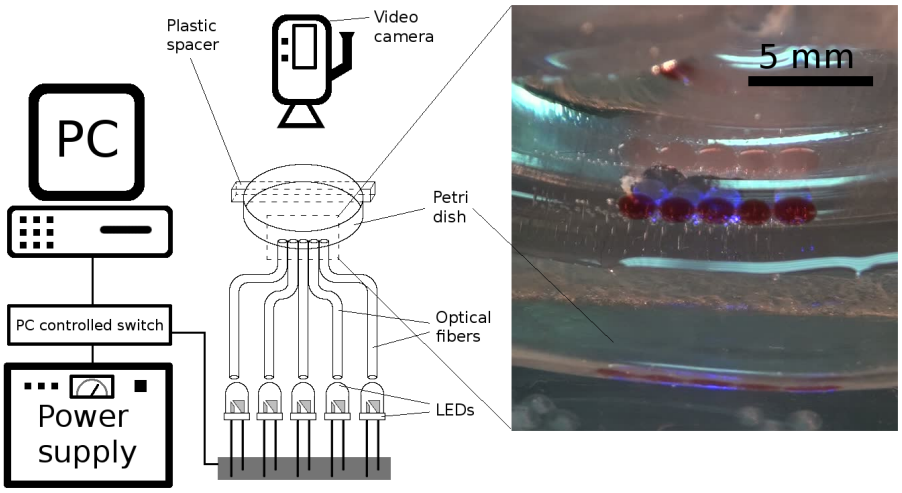


Figure 4.1: The experimental setup of PC controlled diodes emitting blue light (λ=462 nm), which is transferred to each droplet separately through optical fibers. The illumination is used to slow down the BZ oscillation in the droplets up to the point where no oscillation occurs. Furthermore, the illumination can be used to set the oscillation phase of droplets and thus also to synchronize multiple droplets. A snapshot showing droplets in a Petri dish is shown on the right. Drawing and Picture by Konrad Gizynski and Jerzy Gorecki (2014), Warsaw. [Gru14, Giz16]

Blue light interacts with the ruthenium complex and produces Br^- ions that inhibit the BZ reaction. As a result, at a low illumination level the oscillation period is longer than in a dark medium and at high illumination the oscillations are suppressed.

4.2.2 Simulated Droplet System

For the *in silico* studies of droplet systems, we used the stochastic, continuous time, discrete event simulation system described in Ref. [Gru13]. In this approach, the behavior of the chemical droplets is structured into the three phases denoted the excited, the refractory and the responsive phase. Only in the very short excited phase, a droplet can influence its neighbors if they are in the responsive phase. In this work we restrict the influence to the nearest 4 neighbors on a square lattice. A signal propagation delay is included to account for the speed of the BZ wave expansion. In simulations we typically used oscillating droplets with period 16 s (displayed in red).

For these droplets excited, refractory and responsive phases were 1, 5 and 10 s long respectively. Beside of these normal droplets we considered slow and fast oscillating droplets marked with brown color and letters (s) and (f) with periods 12.8 and 20 seconds. Moreover the studied networks included less excitable droplets (green ones), marked with '&' that can be triggered only by two concurrent excitations at their neighbors.

When considering a complex and interacting system of droplets, we will refer to individual droplets as d_i, and to the whole droplet system as $D = (d_1, d_2, ..., d_w)$. Because a two-dimensional structure will probably be reproduced earlier in laboratory experimental setups, we will typically use planar graph structures for the droplets networks here.

4.2.3 Information Theoretic Approach

Symbol Representation

Our basic assumption for measuring the droplet network information transmission is that all information on the dynamics of the droplet system D is present in the sequence of spike times. Therefore, for a given droplet system D, the dynamics can be represented by a set of times $T_{s_d} = \{t_1, t_2, ..., t_n\}$ for each droplet $d \in D$ when it becomes excited. We assume that, the amplitude and the exact shape of the excitations can be neglected.

For the measurement of information, we partition the continuous-timed spike train data into discrete symbols [Str98] as shown in Figure 4.2. Let us select the time discretization length Δt and define a series of $m = \lceil t_n/\Delta t \rceil$ time frames $F_1, F_2, ..., F_m$, where $F_i = [(i-1) \cdot \Delta t, i \cdot \Delta t]$, for $i = 1, 2, ..., m$. For a given droplet d the binary number a_{F_i} is defined as

$$a_{F_i} = \begin{cases} 1, & \text{if } \exists\ t_j \in T_{s_d} \text{ such that } t_j \in F_i \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

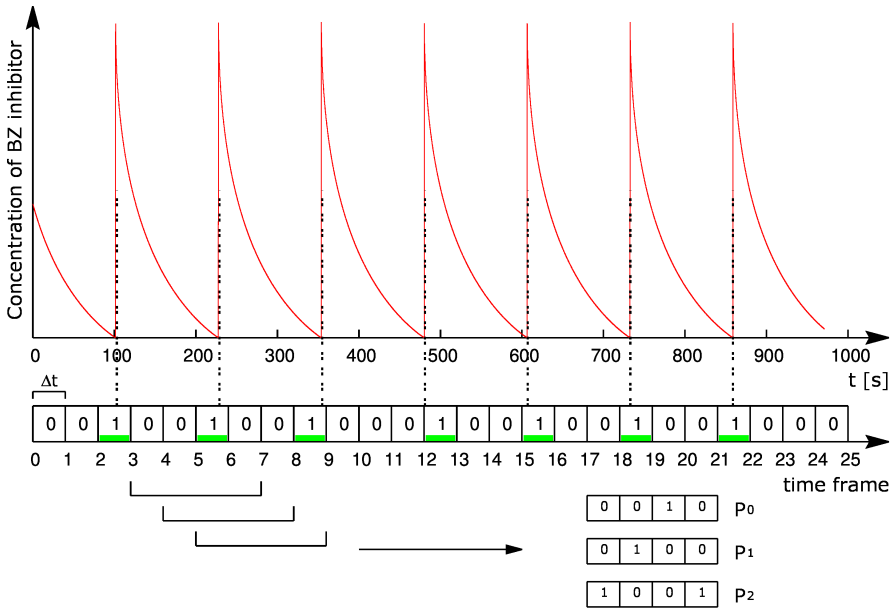


Figure 4.2: Time discretization and extraction of spike patterns from an exemplary spike train using the frame size $\Delta t = 40$ s and pattern length $l = 4$. Diagram by Konrad Gizynski (2014), Warsaw. [Gru14, Giz16].

Now we convert the continuous spike train T_{s_d} of a droplet d into a binary sequence of either zeroes or ones, resulting in the discretized spike train $S_d = (a_{F_1}, a_{F_2}, ..., a_{F_m}) \in \mathcal{S} = \{0,1\}^m$.

Then, from whole discretized spike train S_d with m frames, we generate the patterns $P_i \in \{0,1\}^l$ of the pattern length $l << m$. The multiset of discrete patterns for droplet d, $\mathcal{P}_d = \{P_1, P_2, ..., P_{m-l+1}\}$, is acquired by sliding a window of length l along S_d, resulting in $m - l + 1$ discrete spike patterns. The spike pattern $P_i \in \mathcal{P}_d$ is generated from the time frames $F_i, F_{i+1}, ..., F_{i+l-1}$ as $P_i = (a_{F_i}, a_{F_{i+1}}, \cdots, a_{F_{i+l-1}})$.

For our information theoretic approach, we consider the spike patterns P_i to be the basic symbols. That means we estimate the probability distribution over all the patterns appearing in experiment or simulation as well as the joint distributions of the patterns together with different external inputs or expected outputs of the computations. From a whole recorded or simulated spike train we estimate the probability distribution $p(P_i)$ for all possible spike patterns $P_i \in \mathcal{P}_d$. For the pattern length l

there are in principle 2^l different patterns, but only a small subset of all possible spike patterns do actually appear: given that Δt is small in comparison to the oscillation period of the droplets, most time frames a_{F_i} are zero.

In this article, spike patterns are composed from the excitations of a single droplet d only. But we can trivially generalize the concept of spike patterns to any subset $A \subseteq D$. The corresponding spike patterns for the subset A would then be generated by building a new discretized spike train S_A, such that each time frame F_i would not point to a single value $a_{F_i}^{(d)} \in \{0, 1\}$ for a single droplet d but to a binary number for all the droplets in A, leading to the vector $a_{F_i}^{(A)} \in \{0, 1\}^{|A|}$.

Spike Train Entropy

For a given discretization raster Δt and for a pattern length l, the entropy of a discretized spike train for a droplet d can be measured by the standard Shannon entropy formulation [Sha48] over the distribution of spike patterns found in the spike train:

$$H(\mathcal{P}_d) = -\sum_{P_i \in \mathcal{P}_d} p(P_i)\ \log_2\ p(P_i) \tag{4.2}$$

Clearly, this value does not only depend on the complexity of the spike train, but also on the chosen length of the spike pattern [Dim00] l, on the Δt and on the number of simultaneously observed droplets k. So for example, when reducing the Δt or when increasing the pattern length l the entropy of the observed patterns increases due to the higher number of possible patterns.

Spike Train Mutual Information

As mutual information is widely used as an indicator for correlation, it also qualifies as a natural candidate for finding dependencies between the spike trains of two droplets in a network. The common mutual information formulation

$$I(\mathcal{P}_a : \mathcal{P}_b) = \sum_{P_i \in \mathcal{P}_a} \sum_{P_j \in \mathcal{P}_b} p(P_i, P_j)\ \log_2\ (\frac{p(P_i, P_j)}{p(P_i)p(P_j)}) \tag{4.3}$$

or equivalently

$$I(\mathcal{P}_a : \mathcal{P}_b) = H(\mathcal{P}_a) + H(\mathcal{P}_b) - H(\mathcal{P}_a, \mathcal{P}_b) \tag{4.4}$$

can directly be applied to two different spike pattern distributions $\mathcal{P}_a$ and $\mathcal{P}_b$, where $a, b \subseteq D$ refer to different subsets of droplets in the network D. Here, the joint entropy $H(\mathcal{P}_a, \mathcal{P}_b)$ refers to the probability distribution of both patterns $P_i \in \mathcal{P}_a$ and $P_j \in \mathcal{P}_b$ occurring at the same time.

$$H(\mathcal{P}_a, \mathcal{P}_b) = -\sum_{P_i \in \mathcal{P}_a} \sum_{P_j \in \mathcal{P}_b} p(P_i, P_j) \log_2 p(P_i, P_j) \tag{4.5}$$

The symmetric value $I(\mathcal{P}_a : \mathcal{P}_b)$ gives the average number of bits that are known about the spike pattern from $\mathcal{P}_a$ by knowing the corresponding spike pattern from $\mathcal{P}_b$ at the same time.

Time-Delayed Mutual Information

Excitation waves require time to travel from one part of the droplet system to another, so no droplet can influence another one without a time delay. In principle, excitation waves can travel in either direction if the direction is not explicitly enforced [Szy11]. Here we assume that the signal propagation direction is mostly fixed for a specific stimulation situation, and thus also the time delay between the spike patterns in a pair of two droplets becomes constant. We calculate the mutual information between the spike patterns at droplet d_a for all times t and the spike patterns of another droplet d_b at times $t + \tau\Delta t$ for a limited range of τ values by the formula:

$$\begin{aligned} I^{\tau}_{TD}(\mathcal{P}_a : \mathcal{P}_b) =& H(\mathcal{P}_a) + H(\mathcal{P}_b) \\ & - H(\mathcal{P}_a(t), \mathcal{P}_b(t + \tau\Delta t)) \end{aligned} \tag{4.6}$$

and call it the time-delayed mutual information [Fra86, Per05, Jin10]. Here, $H(\mathcal{P}_a(t), \mathcal{P}_b(t+\tau\Delta t))$ is the entropy of the joint, but time shifted, spike trains.

Assuming that one droplet is maximally influencing another droplet at a specific time delay, we use the time-delayed mutual information to estimate the mutual information

between two droplets and their time delay at the same time. At a time delay τ', when the time-delayed mutual information is maximal, the interaction between the droplets should be strongest. Formally, we note this assumption as:

$$\begin{aligned} I'_{TD}(\mathcal{P}_a : \mathcal{P}_b) &= \max_\tau \quad I^\tau_{TD}(\mathcal{P}_a : \mathcal{P}_b) \\ \tau' &= \operatorname{argmax}_\tau \quad I^\tau_{TD}(\mathcal{P}_a : \mathcal{P}_b) \end{aligned} \tag{4.7}$$

Note that also time-delayed mutual information is a statistical value, such that a maximum in the mutual information at a particular time delay does not exclude information transmissions at other time delays, even in opposite directions.

Information Between Spike Trains and External Values

Instead of correlating the behavior of a specific droplet to another droplet's spike pattern, we can also compare it to external values such as the input class or the expected output class. We use the term input/output class here to distinguish the symbol from its representation as a specific spike pattern: In this case we calculate the mutual information of a spike pattern with the abstract input/output symbol only. For example, when considering a pattern recognition task where low, average and high chemical concentrations are sensed, there is no necessity of encoding a low chemical concentration in a low frequency spike signal. Instead, the low concentration could be encoded as high frequency or more complex spike patterns could be used, maybe even keeping the average spike frequency constant [Esc13].

A problem definition for a function with a finite number of input cases $i \in \{1, 2, .., p\}$, can be given in the form of a table, where the expected output values are listed for each input case i. An example is shown in Table 4.1 for the NOR function. Here we denote each input or output row j of the table as an i/o-channel. Not distinguishing between input and expected output symbols here, we might be interested in the mutual information between any combination of them and the spike patterns found in particular droplets. In the following, we use the symbol γ for a combination of rows from the function definition table. For the function with two inputs and one output

	i_1	i_2	o_1
case_1	0	0	1
case_2	0	1	0
case_3	1	0	0
case_4	1	1	0

Table 4.1: Inputs i_1 and i_2 vs. output o_1 symbols for the four input cases of the Boolean NOR function.

as defined in Table 4.1, γ is a subset of the inputs i_1,i_2 and the expected output o_1: $\gamma \subseteq \{i_1, i_2, o_1\}$. Not distinguishing between inputs and expected outputs is then useful because we can use the same formalism to analyze the mutual information of spike trains with different input channels, output channels or combinations thereof.

To calculate the mutual information with the combination of i/o channels γ, we first have to select an input distribution, i.e., which input cases are selected with which probability. A uniform distribution might be most informative here because it excludes correlation between different input channels that might be present in some data sets. Nonetheless, in the case of the machine learning data sets from Section 4.3.2, we used the original input distribution of the data for convenience. For any droplet d in the network, the mutual information with the combination of i/o channels γ is then calculated as:

$$I(\mathcal{P}_d : \gamma) = H(\mathcal{P}_d) + H(\gamma) - H(\mathcal{P}_d, \gamma) \tag{4.8}$$

This implies, that the droplet system is observed or simulated for all the input cases i. The joint entropy $H(\mathcal{P}_d, \gamma)$ between the input/output channel combination γ and a spike pattern $P \in \mathcal{P}_d$ can be generated by concatenating the appropriate input/output symbol γ to each spike pattern P. An exemplary analysis of the mutual information values of a noise-free Boolean NOR gate is shown in Figure 4.3.

Studying simple Boolean gates in Table 4.2, we can see that the mutual information between an input channel i_1 and the output of the computation $I(i_1 : o_1)$ can be equal to 0 for example in the case of the XOR function, or can be equal to $H(i_1)$ in the case of the NOT function. Furthermore, in the case of the OR or NOR

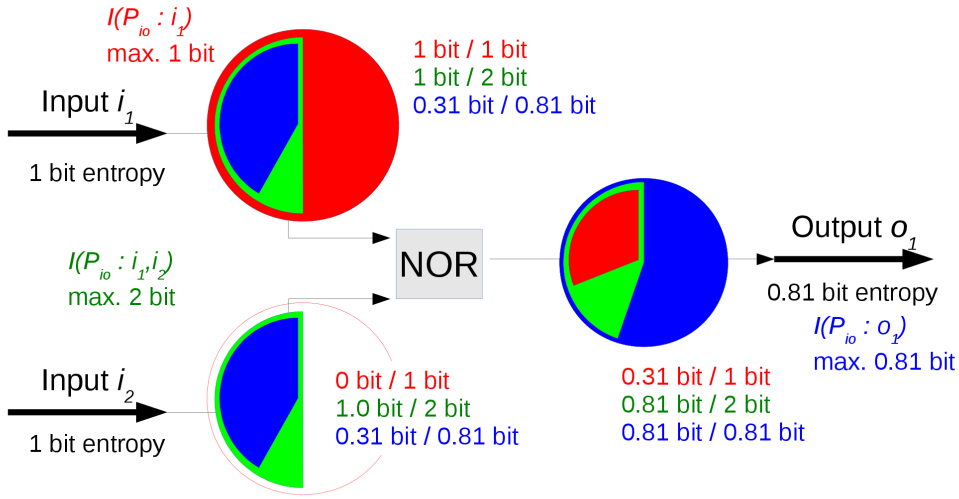


Figure 4.3: Exemplary mutual information diagram of a Boolean NOR gate like it will later be used to analyse larger droplet networks. The size of the circle sectors for each input and output channel of the gate indicates the mutual information of the i/o patterns P_{io} in the channels i_1, i_2 and o_1 with different reference channel combinations γ, divided by the maximum mutual information for this γ. Here, the i/o patterns of a mathematically perfect, noise-free NOR gate are used. Three different reference i/o channel combinations γ are shown: $\gamma_1 = \{i_1\}$ in red, $\gamma_2 = \{i_1, i_2\}$ in green and $\gamma_3 = \{o_1\}$. In other words, red circle sectors indicate the mutual information with the first input channel, green sectors indicate the mutual information with both input channels combined and blue sectors indicate the mutual information with the output symbol. Interestingly for $\gamma_3 = \{o_1\}$ (blue), the 0.31 bit of information that are present in each input channel about the output symbol do not add up to the final 0.81 bit of entropy in the output. Instead, only the combination of both inputs allows the exact prediction of the gate output. In a more extreme case, for the XOR gate, a single input value would not hold any information on the output value, without knowing the other input value. Furthermore, the mutual information for $\gamma_2 = \{i_1, i_2\}$ (green) illustrates the view of computation as information destruction, where only 0.81 bit remain of the initial 2 bits of input information.

	$H(i_1)$	$H(i_1, i_2)$	$H(o_1)$	$H(i_1, o_1)$	$I(i_1 : o_1)$
ID	1	-	1	1	1
NOT	1	-	1	1	1
AND	1	2	$\sim$0.81	1.5	$\sim$0.31
OR	1	2	$\sim$0.81	1.5	$\sim$0.31
NOR	1	2	$\sim$0.81	1.5	$\sim$0.31
XOR	1	2	1	2	0

Table 4.2: Entropies and mutual informations of input and output tuples for typical deterministic Boolean functions, measured in bits. ID here represents the identity function, having the same information characteristics as the NOT function. Some further properties are $H(i_1) = H(i_2)$ and $I(i_1 : i_2) = 0$ by definition for two input functions with equally distributed and independent Boolean input cases. $H(i_1, i_2) = H(i_1, i_2, o_1)$, because all the information about the output is already present in the input. This also implies $I(i_1, i_2 : o_1) = H(o_1)$, if the output is at all dependent on the input.

functions, even though there is information about the output present in each of the single inputs already, this information does not sum up to the entropy in the output. Hence, a synergy between the input channels is exploited for computing the output by combining both inputs. In the extreme example of the XOR function this becomes obvious: It is not possible to make any judgment about the output, if only a single inputs is known.

4.3 Results

4.3.1 Information Flow in an Experimental System

The experiment discussed below was conducted as described in Section 4.2.1. From the total recorded time evolution of five droplets of length 4505 s, we considered the data from the time interval [1585 s, 4505 s] because only after 1585 s the initiation procedure was completed.

The frames from the considered interval were extracted at one frame per second. This produced a video stream of 2920 video frames. To observe the time evolution

of oscillations in the droplets d_1 till d_5, arranged in a linear chain as shown in Figure 4.4, we cut the series of frames along the bright line and obtained the space-time plot presented in Figure 4.4(b).

Bright, periodic stripes correspond to the high level of the oxidized form of bathoferroin and they mark the moments at which the excitations occurred. We identified 28 excitations for the droplets d_1 and d_4, 27 excitations for d_2 and d_3 and 29 excitations for d_5. The minimum observed oscillation period was 63 seconds and it increased to 190 seconds when the medium was nearly depleted.

In the initial part of the experiment the droplets d_1, d_3 and d_5 oscillate spontaneously as marked schematically with the white, curved lines. Self-excitation chemical waves are initiated close to the geometrical center of a droplet and travel outwards. Initiation at the center is related to the high concentration of activator there. Activator diffuses to the organic phase through the lipid layer and its concentration near the boundary is lower. When the wave reaches the connection to a neighboring droplet that is in the excitable state then the activation occurs. In that case we observe a directional wave visible as inclined stripes on the space-time plot, with the initiation center at the connection point.

During the time-span of the experiment we illuminated droplets d_3, d_4 and d_5 with a low light intensity. As the result, the oscillation period in non-illuminated droplets d_1 and d_2 is shorter. After the time $t = 2800s$, waves originating from droplet d_1 spread out through the droplet chain, effectively controlling the oscillations of the remaining droplets. Then, at time $t = 2905s$, we turned on the illumination of droplet d_1, leading to slower oscillations in that droplet. This illumination was applied until time $t = 3506$ s and next it was switched off. It changed the dynamics in the investigated network: The droplet d_3 started to take control over the oscillations of the network till the end of the illumination but was superseded by droplet d_1 afterwards. We used this changing dynamics as a source of entropy in the droplet chain and follow the information flow from the illuminated droplets by means of time-delayed mutual information.

For each pair of adjacent droplets, we calculated the time-delayed mutual information,

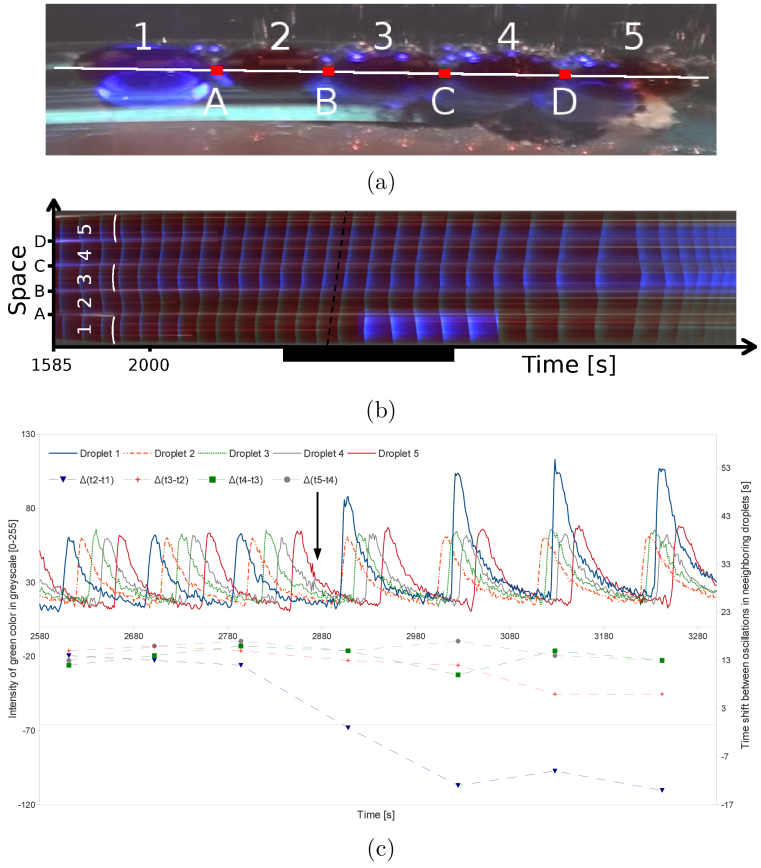


Figure 4.4: Dynamics of the experimental droplet system. (a) A snapshot of a coupled, five droplets chain observed in experiments. (b) A space-time plot obtained by cutting the series of frames along the bright line drawn in the first panel. The slightly curved lines, drawn above arrows, schematically represent the shape of the wave front that is characteristic for self-excitations. The dashed line, below the arrow, indicates a part of the experiment when the activation sequence of excitations 1→2→3→4→5 was observed. The black rectangle at the time axis marks the time interval for which the intensity of green color at the geometrical centers of the droplets as a function of time are plotted in Figure (c). The distance between the maxima in a selected droplet determines the period of oscillations, whereas maxima for two different droplets indicate the time shift between forced oscillations. The arrow (at Figure (c)) marks the moment (t=2862 s) at which illumination was applied to droplet d_1. The illumination was terminated at t=3484 s. Pictures and diagram by Konrad Gizynski (2014), Warsaw. [Gru14, Giz16]

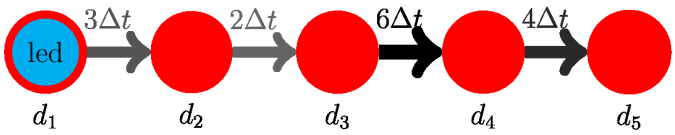


Figure 4.5: Schematic illustration of the five droplets from Figure 4.4(a) and the measurement of time-delayed mutual information for $l = 30$ and $\Delta t = 3s$. The droplets d_3, d_4 and d_5 are illuminated with low intensity light and their period of self-excitations is longer than the one for d_1 or d_2. Droplet d_1 is controlled by the LED. Droplet d_2 is probably also influenced by illumination of droplet d_1. Hence droplets d_1 and d_2 control the remaining part of the droplet system. The arrows indicate that we measured a time-delayed mutual information with the ideal delay in frames displayed above each arrow. We measured the strongest time-delayed mutual information between droplets d_3 and d_4 with a time delay of 6 frames, corresponding to 18 s.

resulting in the directions and time delays displayed in Figure 4.5. For the pattern length $l = 30$ frames and frame length $\Delta t = 3s$ we obtained entropy values of about 4.8 bits per droplet and maximal mutual information values I'_{TD} between 1.98 (d_2 to d_3) and 2.87 (d_3 to d_4) bits. These measures support the observation that information is transferred from the entropy source, the illuminated droplet d_1, to the droplets on the right.

Nonetheless, about 30 observed excitations do hardly suffice to build a probability distribution of spike patterns, in particular, when the oscillation times are constantly and continuously changing as the medium is aging in this example. Since the medium is changing in a similar way for all droplets in the system, a correlation between each droplet's behavior might be measured. But because this correlation should be similar for all delay values, the maximum of the information over the different time delay values should still be a useful indicator for the actual interaction.

4.3.2 Hand-Designed Linear Classifier Network

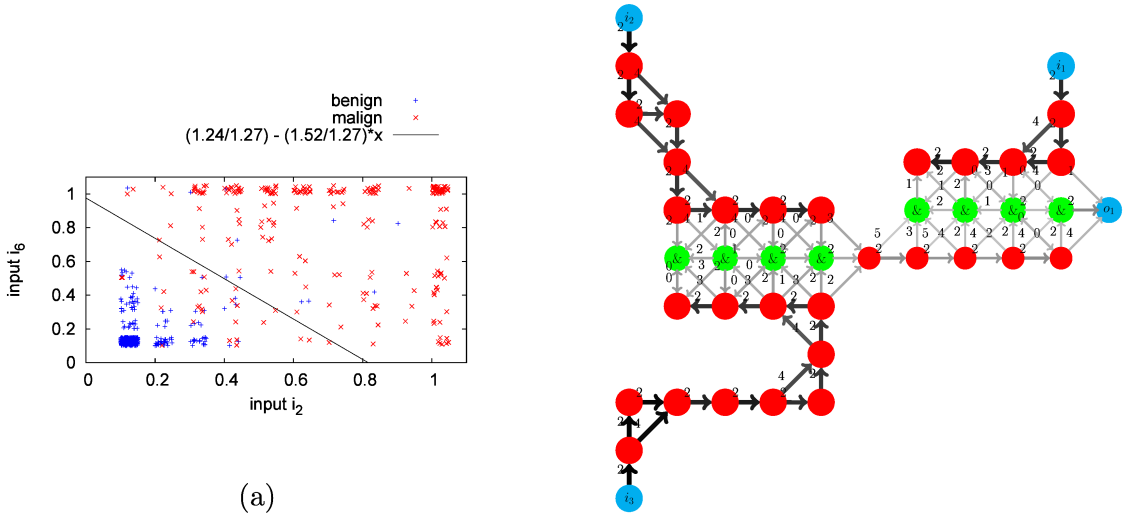
From the point of view of information theory, real droplet experiments cover a small number of oscillations. In the future, we plan to use microfluidic devices which should improve reproducibility of the experiments, though. Here, further studies are carried

out in simulation. First we will introduce and analyze a hand-designed linear classifier network design and next we discuss an evolved binary NOR gate. Then we will show how the information flow measures can support studying the function of network components.

A simple linear classifying network was hand-built in simulation for classifying samples from the *Proben1* [Pre94] data set. Each data record from the *cancer* subset is a 10 elements vector $\{i_1, i_2, ..., i_9, o\}$ where the first 9 values, $\{i_1, i_2, ..., i_9\}$ belong to the set $\{k \cdot 0.1; k = 0, ..., 10\}$ while the last value is the binary output class: either *benign* or *malign*. Of the 699 samples, 458 are from the *benign* class, while 241 are for the *malign* case. The output class entropy is ~ 0.93 bits. For the same data set, we already presented the evolution of a generative network description [Die12b]. But in contrast here, we hand-designed the network with the rationale of selecting three out of the nine input signals for which the mutual information to the expected output class has the maximum. We found that the inputs i_1, i_2 and i_6 combined, jointly show a mutual information to the output class of ~ 0.92 bits. First we combine the two inputs with the highest joint mutual information, i_2 and i_6, and then combine the result with the input i_1. These three inputs are fed into the droplet network as analogue, rate coded signals, where the value 0.1 corresponds to the lowest spike frequency, very close to self-excitation, while 0.9 corresponds to a very high spike frequency, very close to the highest possible spike rate.

According to Figure 4.6(a), most of the samples of class 1 are in one sector of the input values, so a linear classification symbolized by the black line allows a correct classification level of 94%. The droplet network design shown in Figure 4.6(b) performs this task when used with an ideal, external threshold function to a similar level of typically more than 90%, as show in Figure 4.7.

When applying random stimulation patterns at all three input nodes as entropy sources, we can follow precisely the propagation of time-delayed mutual information, droplet by droplet, from the inputs i_1, i_2 and i_6 to the central less excitable nodes and towards the output node o_1 in Figure 4.6(b). Analogously, when observing the



(a) (b)

Figure 4.6: Test cases and hand-designed droplet network for the classification of the *Proben1*, *cancer* data set. (a) Scatterplot of the input variables i_2 and i_6. To simplify the visualization, a uniform random number between 0 and 0.05 is added to the discrete input coordinates i_2 and i_6. Obviously a relatively accurate classification can already be achieved by a linear combination of i_2 and i_6 as indicated by the black line $i_6 = -\frac{1.52}{1.27} i_2 + \frac{1.24}{1.27}$. (b) Droplet network design for classification, showing the information flow when randomly stimulated at the input nodes i_1, i_2 and i_6. Spike train discretization with $\Delta t = 0.5s$ and a pattern length of $l = 45$ frames was used. The maximum entropy per droplet was 9.88 bits, the minimum 5.33 bits. The maximum time-delayed mutual information was between 1.49 and 8.78 bits. Differently colored circles represent different droplet types as described in Section 4.2.2.

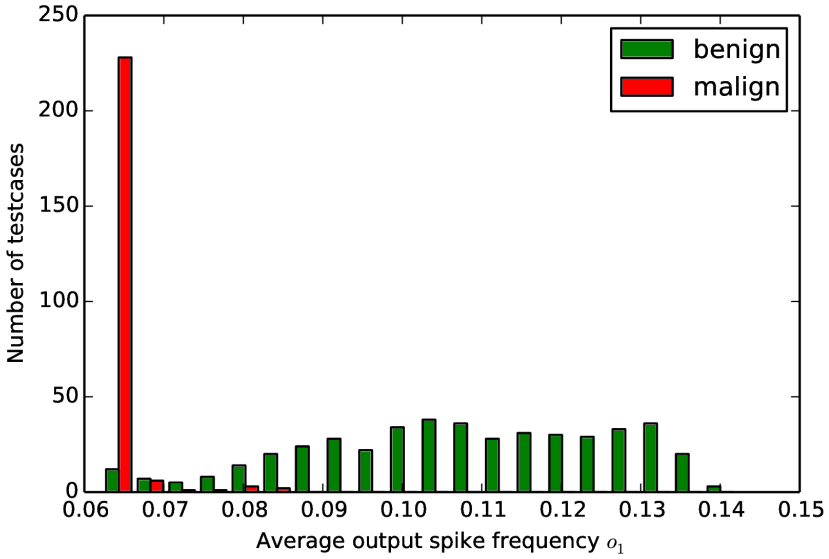
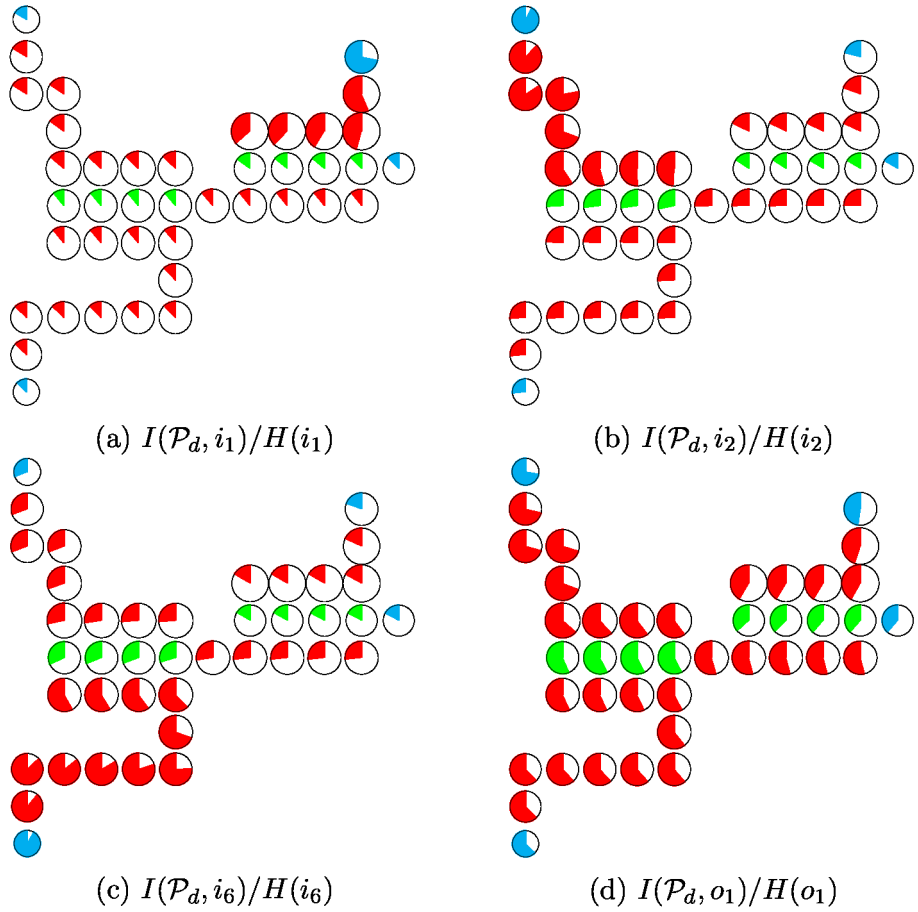


Figure 4.7: Output spike frequency of the droplet network. The 241 malign cases (red) typically result in a much lower spike output frequency than the 458 benign cases.

mutual information with the external input symbols, we can follow the input information of each channel individually in Figure 4.8. Also, we see that the information about the input symbols is decaying on its way to the central rows of less excitable droplets (green). Another aspect of these plots is, that, for example in Figure 4.8(a), the information about i_1 is not zero at the other input nodes i_2 and i_6. The reason here is, that there is already mutual information between the different input symbols in the data set.

Surprisingly, when considering Figure 4.8(d), the mutual information between the spike patterns and the expected output declines when moving from either input along the droplet network towards the output droplet. That means, while a lot of information about the correct outcome of an experiment is present in the inputs initially, this information seems to be lost throughout the network. Nonetheless, the classification using a simple threshold function at the output works relatively well. The reason for this apparent loss of the information about the result towards the end of the computation lies in the considered time frame of the spike patterns: The threshold function



(a) $I(\mathcal{P}_d, i_1)/H(i_1)$ (b) $I(\mathcal{P}_d, i_2)/H(i_2)$

(c) $I(\mathcal{P}_d, i_6)/H(i_6)$ (d) $I(\mathcal{P}_d, o_1)/H(o_1)$

Figure 4.8: Mutual information between each droplet's spike train and the external input or expected output symbols, *i.e.* with input i_1 (a), input i_2 (b), input i_6 (c) and with the expected output o_1 (d). Spike train discretization with $\Delta t = 0.5s$ and a pattern length l of 45 frames. We measured spike pattern entropies per droplet between 5.4 and 9.7 bits. Entropy of the external inputs i_1, i_2, i_6 and the expected output o_1 were 3.0 bits, 2.3 bits, 2.0 bits and 0.92 bits, respectively. These external entropy values were used as reference for the per-droplet pie charts displayed. The colored fraction of the pie chart corresponds to the mutual information between this droplet's spike train and the external reference symbol, such that a full pie chart would indicate the maximal possible mutual information between the spike patterns and the external input/output. Different colors indicate different droplet types as explained in Figure 4.6(b).

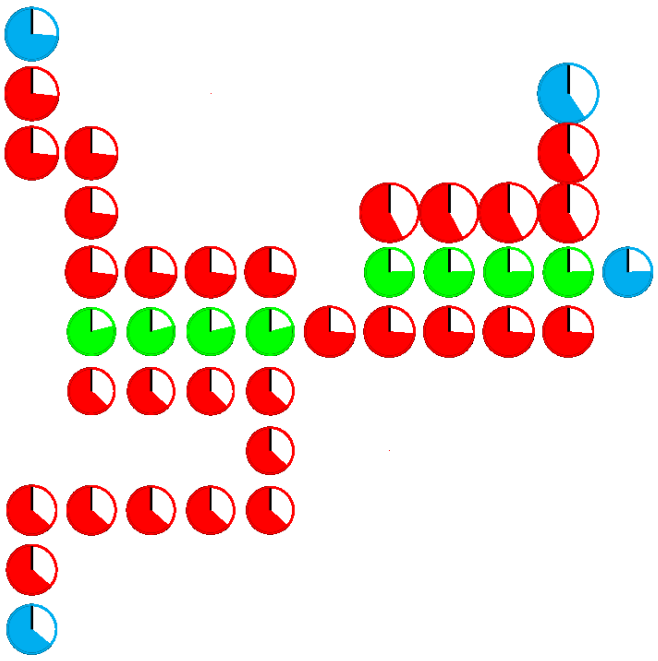


Figure 4.9: Using another spike train discretization, the mutual information with the expected output can be increased. Here, instead of using the distribution of all possible spike patterns, we only averaged the number of spikes in a window of 200 s and distinguished 10 different bins. Except for the discretization, this plot in analogy to Figure 4.8(d).

works on the averaged spike frequency over the whole experiment of 4000 s while our spike patterns only covered 22.5 seconds. But it is difficult to estimate the probability distribution of all possible spike patterns for patterns larger than 50 frames in this case. For estimating an average of the spike frequency over 200 seconds, 400 frames would be necessary at a frame length of $\Delta t = 0.5$ s. To measure the mutual information with the 200 seconds averaged spike frequency, we discretize the spike range of appearing spike frequencies into 10 different classes and build the probability distribution on the abundance of these classes instead. When using this average spike rate as discretization scheme instead for the mutual information measurement as shown in Figure 4.9, we observe an increase of the mutual information with the expected output class towards the output droplets.

4.3.3 Information Flow in an Evolved NOR Gate

Information flow analysis can be used to understand the function of a droplet network obtained, for example, as the result of an evolutionary algorithm. In this example, a NOR gate was evolved in simulation as a variation of our experiments described in Ref. [Esc13], forming the design shown in Figure 4.10. Using trivial rate coding, the NOR function implies that for a high frequency stimulation on either or both input channels, a low frequency should appear on the output droplet, while a low frequency on both input channels should lead to a high frequency output as shown in Figure 4.11. From the time-delayed mutual information diagram, shown in Figure 4.10(a), the network's mode of operation can hardly be deduced. When observing the information flow in dependence of the inputs or the expected output (cf. Figures 4.10(b-d)), it becomes obvious that a very good output of the computation is accumulated in the top left $\&_1$-droplet, very close to the input i_1. Furthermore, wherever the actual computation takes place, the result has to be transferred to the output droplet o_1. Along the way, some of the information may get lost again. When directly reading the signal from the top left droplet instead, the noise level in the rate coded result of the computation is actually reduced.

This leads to our assumption about the network function, that the actual computation, the modification of information both dependently on inputs i_1 and i_2 happens in droplet $\&_1$. The long trail of "full" droplets in Figure 4.10(c) on the other hand only transfers the information from input i_2 to the "computing center" $\&_1$, but also relays the result of the computation back to the output droplet o_1. In the next section, we will further investigate this assumption.

4.3.4 Effect of Manipulating the Information Flow

Mutual information analysis between spike patterns and inputs or expected outputs can also be used to better understand the function of particular droplets in the network by doing mutation or deletion experiments. Here we investigate the thesis that the computation of the NOR network is happening in the top left $\&_1$-droplet as

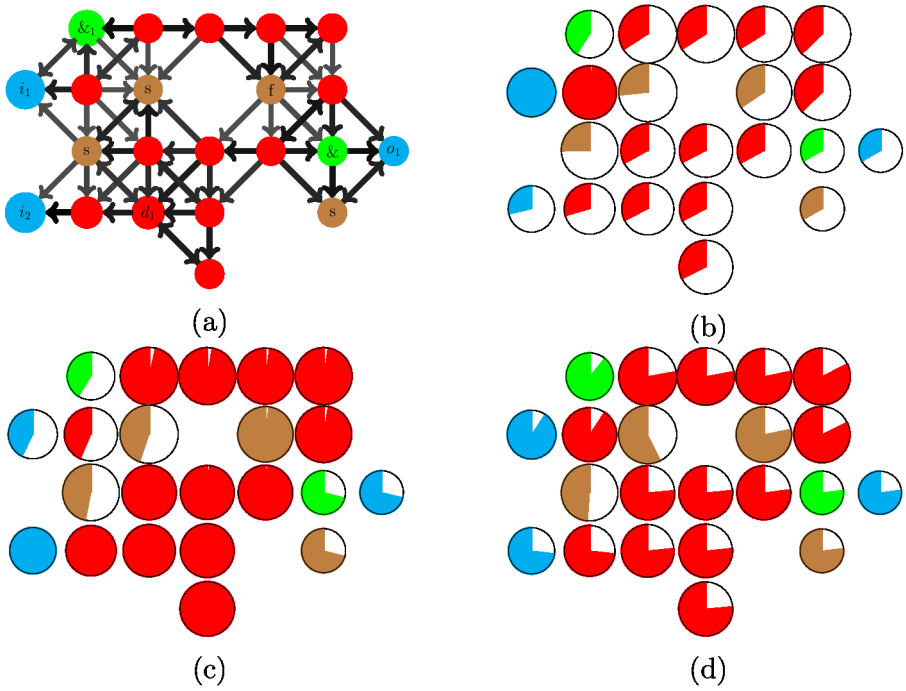


Figure 4.10: Droplet network designed by evolutionary algorithms to calculate the binary NOR function. Differently colored circles represent different droplet types as described in Section 4.2.2. The basic network setup including the information transmission using time-delayed mutual information is displayed in Panel (a). Arrows indicate the direction of the time-delayed mutual information between neighboring droplets. The pie charts depict the mutual information with the input i_1 (b), the input i_2 (c) and the expected output o_1 (d). Spike trains are discretized with a frame size $\Delta t = 1s$ and $l = 35$ frames. The displayed sizes of the droplets indicate the entropies of the spike trains which were between 5.4 and 9.5 bits. The entropies of the external inputs i_1 (b), i_2 (c) and the expected outputs o_1 (d) were 1 bit, 1 bit and 0.81 bits, respectively. In analogy to Figure 4.8, these Entropy values were used as reference for the per-droplet pie charts displayed, such that a full pie chart would indicate the maximal possible mutual information between the spike patterns and the external input/output.

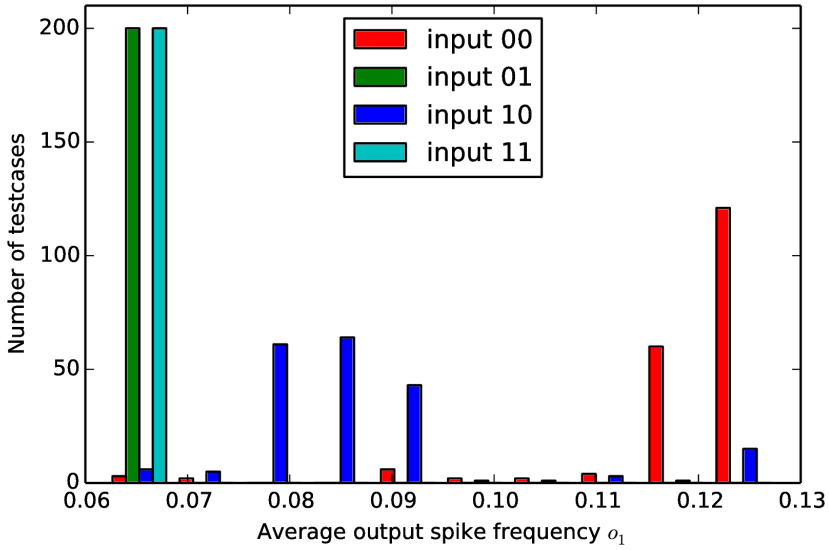


Figure 4.11: Output spike frequency of the NOR-gate droplet network. For each input case, 200 replications of the experiment were simulated. The input cases 00 that should be answered with symbols 1 can be distinguished by their higher spike frequencies from all the other spike symbols that result in lower spike frequencies. Selecting the threshold between low and high frequencies at 0.1, we obtained {185,200,180,200} out of 200 correct results for the input cases {00,01,10,11}, respectively. It means that the overall probability of a correct answer was ca. 95%, which is pretty high considering the stochastic character of the event simulator used.

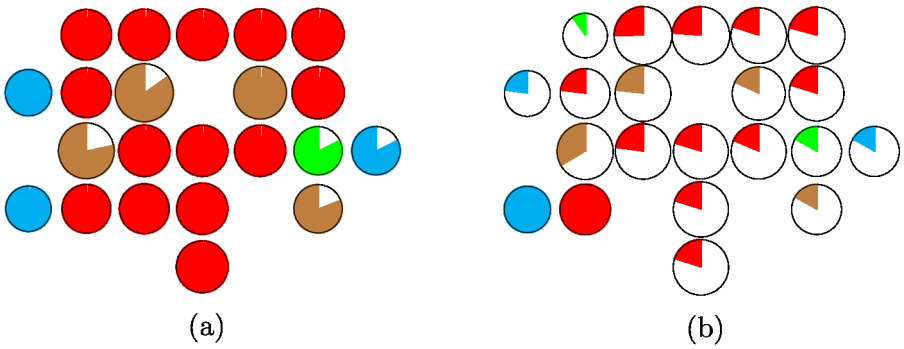


Figure 4.12: Effect of modifications on the information flow of the network design from Figure 4.10. (a) Mutual Information between spike trains and expected output symbol o_1 for modified droplet $\&_1$. (b) Mutual Information between spike trains and external input symbol i_2 for deleted droplet d_1.

implied by the high mutual information to the expected output, shown in Figure 4.10(d).

To investigate this, we first exchange droplet $\&_1$ by a normal droplet. This leads to a globally spread mutual information with the expected output, shown in Figure 4.12(a). But this also leads to an inversion of the spike frequency for each input case, *i.e.* the whole network acts as an OR gate instead of a NOR gate. Note that an OR gate can produce the complete 0.8 bits of mutual information with the expected NOR output, because the inversion of the signal encoding does not destroy information. Nonetheless, building an OR gate from droplets is far simpler than a NOR gate [Esc13]. Hence the $\&_1$ droplet is necessary for the correct network function.

In another experiment, we removed droplet d_1 from Figure 4.10(a), because it is distant from the possibly important center of computation $\&_1$. Then, the output mutual information is reduced in almost all nodes of the networks. In particular in the top left $\&_1$ droplet, the output mutual information is reduced to almost half its original value. The reason here is probably the perturbed information flow from input i_2 to the computing center $\&_1$ that becomes obvious when comparing Figures 4.10(c) and 4.12(b). Because the information from input i_2 does not arrive at the droplet $\&_1$ where it should be combined with input i_1, the synergy of the inputs cannot be exploited in the modified network.

4.4 Discussion

In this chapter, we presented a general approach that can be used to design and understand a structure of interacting neuron-like objects that perform a specific information processing task. Our investigation of information flows differs from conventional computer science methods, where the execution of a computer program is typically studied from the perspective of runtime and memory complexity. Instead, the measurement of the propagation of information between droplets is based on information theory and can be applied to different media and different types of coding. The principle advantage of this approach for understanding the droplet system's activity

is that the method is independent of a particular symbol encoding scheme [Str98]. This becomes more important when considering the possibility of improving the total droplet network performance by mixing different symbol encoding schemes to solve each subtask in an individual, optimal encoding scheme. Thus, methods based on information theory can be seen as tools helping to optimally exploit the capabilities of the computing medium. In this chapter we first theoretically described how to analyse such networks of computing components. Then we demonstrated the application of this analysis technique on a real experimental model system of five droplets as well as on simulations of larger classification and NOR-gate networks. Finally, for the NOR gate, we also showed how the information flow analysis helps understanding the network function in combination with targeted modifications and deletions.

Open Problems

By observing the distribution of spike patterns of length l, we are estimating the average reduction in uncertainty about either another droplet's spike pattern or about external values. But due to limited experiment length (real droplets are depleted after 20 - 100 excitations), due to limited experiment reproducibility and due to limited computational resources, it will not be possible for arbitrary long spike patterns to estimate their probability. When we enlarge the spike pattern length l by one, the number of possible patterns 2^l doubles even though not all spike patterns will actually appear because of comparatively long refractory times. In the presented examples, it was possible to reconstruct the information flow in the experiments and simulations albeit using naive sampling of the appearing patterns. But for example in Section 4.3.2, we observed a case where the practical pattern length was not sufficiently long to capture the mutual information with the computed output as seen in Figure 4.8(d). Only when we switched to another method of discretizing the spike trains, by taking the average spike frequency over a long time period, we observed the mutual information with the expected output in Figure 4.9. Other methods for calculating correction terms for the limited sampling and for alternatives to discretization schemes can be found in Refs. [Pan96, Str98, Rou99, Nem04, Kra04].

Another problem of using time-delayed mutual information but also of transfer entropy is the assumption that the cause and effect relationship, and thus also the time delay between two droplets, would be constant in a droplet network. This might not always be true, as we see for example in the case of an XOR network that the direction of signal propagation is changing with changing input patterns [Esc13]. Mutual information between spike trains and external symbols should not be affected by this effect so much, because it does not matter what generates the spike patterns in a droplet or in a set of droplets.

Furthermore, due to the energy consumption of the BZ medium and the non-equilibrium dynamics, its composition as well as its oscillation dynamics are constantly changing over the course of the experiment. On the one hand, this complicates the sampling of all possible spike patterns because it might be hard to repeat a particular situation, on the other hand it might produce the "symptoms" of correlations between all the droplets in the system, even though they are not really coupled by wave propagations. But a droplet's past is not sufficient to predict the varying next oscillation pattern, which might lead to an erroneously measured transfer entropy.

Generally, when calculating the mutual information between spike trains, it can happen that a correlation between physically disconnected droplets is observed. One reason for this effect is that in an experiment droplets have similar periods so once the initial conditions are fixed, one droplet's state can be predicted from the time evolution of another unconnected droplet. Another reason is, as seen in Figure 4.8, that already the supplied input cases might show some correlation. To determine causality in the spike patterns, we use time-delayed mutual information here as explained in Section 4.2.3. In further experiments it can be useful to combine the spike patterns of multiple experiments or to use transfer entropy [Sch00, Sta08, Wib11] instead. When calculating the time-delayed mutual information, instead of only plotting the time delay at the maximum mutual information as presented in Figures 4.5 or 4.6(b), it might be more useful to investigate the complete diagram of mutual information vs. time delay. Such a plot would show if an expressed peak in the mutual information actually exists at a particular time delay.

Combined Spike Patterns

Even though we considered only the spike trains of single droplets in the presented examples, the information of an input signal or of intermediate computations might be spread over multiple droplets. In that case, only the combined spike trains of many droplets would reproduce the complete information. Our framework for spike pattern entropy and mutual information is readily suited to capture this kind of information from aggregated droplets as mentioned in Section 4.2.3, even though it becomes harder to sample the distribution of spike patterns then.

Mutual Information Based Fitness Functions

Instead of only using the information flows to understand present droplet networks, information theory can also aid in the design of unconventional computing systems, e.g., as a fitness function in evolutionary computation. We demonstrated the feasibility of our method by evolving binary *in-silico* droplet classifiers for three machine learning data sets with different levels of difficulty in [Giz16]. Notably, our approach does not require to pre-specify how the output signal should be interpreted. This is achieved by a fitness function that measures the mutual information $I(\mathcal{P}_o, o)$ between the output droplet's spike patterns $\mathcal{P}_o$ and the desired output class o. In other words, this mutual information, and hence the fitness value, can be interpreted as the reduction of uncertainty about the output class when seeing a particular spike pattern or spike number.

When allowing very complex output patterns, the computing droplet system can become simpler and export more of the actual computation to the observer interpreting the output signal: In the extreme case, all inputs are just mapped into the output spike pattern without doing any information modification. By continuously changing the possible complexity of the spike patterns, *i.e.* the number of considered time frames, we might continuously increase the amount of computation that has to be done in the network itself. Furthermore, to reduce the variability in output patterns $\mathcal{P}_o$ for an expected output o, we can introduce a set of further fitness functions

$f_o = I(\mathcal{P}_o, i_o)$ to be minimized, where i_o is the set of all input cases that lead to the same output o: $i_o = \{i | f(x_i) = o\}$.

Conclusion

By following the information flow in unconventional computing systems which are for example built from BZ droplets, we gain a deeper understanding of the processes that are actually happening while abstracting away some physical peculiarities and properties of the system. So a very similar kind of analysis should be possible in other non-BZ systems, given a suitable discretization of the used signals. Considering the mutual information of spike trains with external inputs or with expected output values, we have shown that we can identify those droplets that fulfill important functions in the network. We hope that this kind of analysis will in future also allow a different concept for designing, specifying and synthesizing blueprints of unconventional computing systems, based on the desired information flows. For more complex tasks, we can identify possible intermediate results for the computation, positions where information needs to be joined or lost necessarily. Then, instead of using evolution with an unbiased mutation operator to design droplet systems, we can also use evolution-inspired systems [Zha14] that are biased on where to apply mutations, similar to a tinkerer that does not know the effect of his actions in detail but has an idea about where the information flow is impaired.

Chapter 5

Tautological Loops

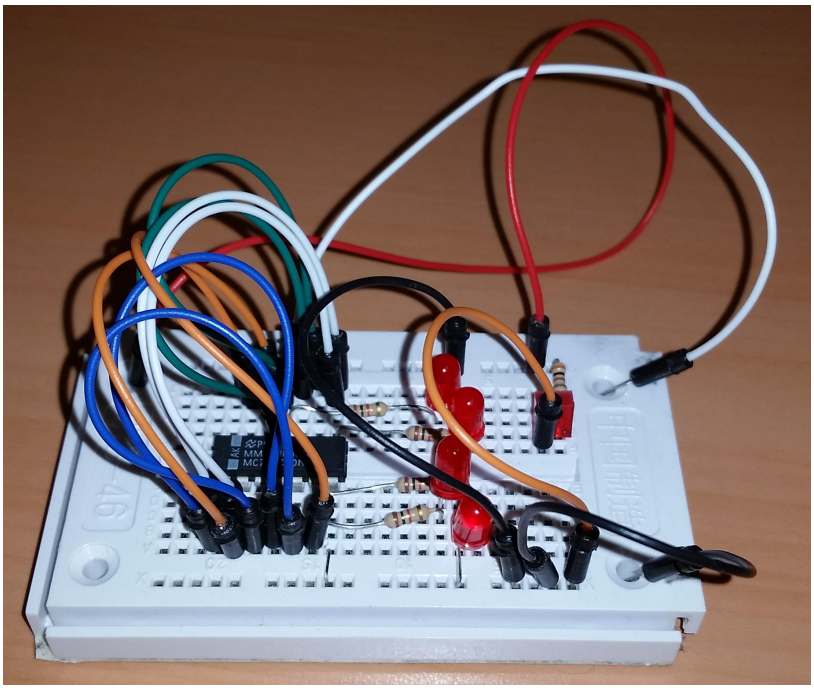


Figure 5.1: Real-world implementation of a tautological loop using an MM74HC00N IC that combines four NAND gates, assembled by Peter Dittrich.

As explained in the introduction, in Section 1.3, next to the pure information processing system that transforms input signals to output signals, we also need the

abstraction *ab* and concretization *co* functions that translate between the abstract input/output symbols $x \in X$ and the practical input output signals $s \in S$. We refer to this mapping between symbols and signals as the symbol representation. Throughout this thesis, it has always been a central issue, that particular symbol representations might be more or less useful for particular computing systems and for particular information transformation tasks. Hence the problem of finding information processing systems becomes even more complex, while at the same time a suitable symbol encodings for this system has to be found that allows to efficieciently use the computing resources. In Chapter 3 we used evolutionary algorithms to co-evolve symbol encodings with the computing system. Later, in Chapter 4, we used information theory to detach the symbol encoding from the actual information processing task.

As an alternative approach here, Egbert [Egb13] proposes to use *Tautological Loops* or *RERUN networks* (Re-entrant Recurrent networks of Repeated UNits) that make use of putative implementations' system dynamics to find abstraction- and concretization functions between the abstract symbols X and the signals S. Given that the inputs and outputs of a computation are using the same abstract set of symbols X, networks are constructed of multiple instances $p_f^{(i)}$ of gates that are connected in such a way, that all input signals for the gates are taken from the output signals of other instances in the same network. An electronic implementation of such a network is shown in Figure 5.1. Ideally, neither pre-defined inputs nor outputs to the whole systems need to be specified or interpreted. Starting from a randomly chosen initial state of the network, we expect the system dynamics to settle in an attractor where the signals between gate instances adopt useful values that might be used to represent the abstract values from X. In this Chapter, we will elaborate conditions for this to actually happen, investigate suitable tautological loop network designs and describe a proof of concept using the exemplary droplet computing systems introduced in Chapter 2.

symbol	meaning
f	abstract function that is to be computed
p_f	practical implementation of the abstract function f as a program, electronic circuit, gate or dynamical system
$x, y \in X$	abstract symbols
$s, o \in S$	signals that can be used in practical computations
$s_j^{(i)}$	j-th input signal to the i-th gate, $j \in \{1, .., k\}$
$o_j^{(i)}$	j-th output signal of the i-th gate, $j \in \{1, .., l\}$
k	number of inputs to each gate
l	number of outputs to each gate
m	number of gates in the Tautological Loop

Table 5.1: List of mathematical symbols that are used in this chapter.

5.1 Introduction to Tautological Loops

Given an abstract, deterministic function $f : X^k \to X^l, |X| \in N$ that maps k input elements to l output elements of a finite set X, we are searching for an implementation or gate $p_f : S^k \to S^l$, that maps k input signals to l output signals. But to actually implement the function, the abstract set of input / output symbols $x \in X$ needs to be concretized into a set of signals $s \in S$ which is not necessarily finite, as for example an electric voltage. This is done by an abstraction function $ab : S \to X$ and the related concretization function $co : X \to S$, as explained in Figure 5.2. As we showed in Section 3.1, finding this abstraction and the set of usable signals S is equally challenging. Instead of testing every symbol to signal mapping or using prior knowledge for the choice of signals, we use the systems own dynamics instead. By connection all the systems' outputs to its own inputs, we expect the system dynamics, under specific conditions, to settle in an attractor that exhibits useful signals s that can be used in the abstraction/concretization function. Because we will use many mathematical symbols in this chapter, a list of the used letters is given in Table 5.1.

5.1.1 Naive Approach for Finding Appropriate Signals

Following a naive approach, for each putative implementation p_f of the function f, we would have to test all mappings ab from every possible abstract value $x \in X$ to every possible signal $s_i \in S$, which becomes especially hard in the case of infinitely sized signal domains. Clearly, the result of the abstract computation should match the result of the implementation when transformed by the abstraction and concretization functions. But additionally, no two different abstract symbols should be mapped to the same signal. Nonetheless, two different signals might be used to encode the same abstract symbol, e.g., two very similar but different voltages might in digital electronics both be used for a logical zero.

$$\begin{aligned} \forall (x_1, ..., x_k) \in X^k &: f(x_1, ..., x_k) = ab(p_f(co_i(x_1, ..., x_k))) \\ \forall x, y \in X, x \neq y &: ab_i^{-1}(x) \neq ab_i^{-1}(y) \end{aligned}$$

Here, with the abstraction and concretization functions of a list of arguments, e.g. $ab(x_1, ..., x_k)$, we denote their component-wise application: $ab(x_1, ..., x_k) = (ab(x_1), ..., ab(x_k))$ and analogously for $co(s_1, ..., s_l)$.

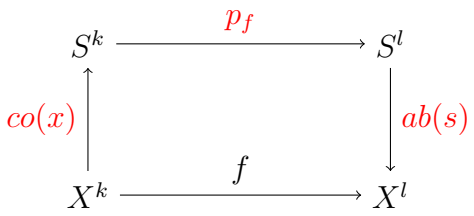


Figure 5.2: Computation diagram: To find an actual implementation p_f for an abstract function f, also the symbol encoding needs to be fixed, in the form of congretization co and abstraction ab functions that translate between the abstract symbols X and the signals S. The *tautological loop* approach allows the self-organization of the symbol encoding from the dynamics of the implementation p_f. Here we assume that the inputs and outputs to the abstract function f both use the same set of symbols X, albeit a different length of the inputs X^k and the outputs X^l is possible.

5.1.2 Definition of Tautological Loops

We define a tautological loop as a directed graph where each of the m nodes represents an equivalent implementation $p_f^{(i)}$ or gate and each directed edge is the connection of a gate's output with another gate's input. Each gate $p_f^{(i)}$ has k inputs $\{s_1^{(i)}, s_2^{(i)}, \ldots, s_k^{(i)}\}$ and l outputs $\{o_1^{(i)}, o_2^{(i)}, .., o_l^{(i)}\}$. Both the inputs s_i and the outputs o_j are elements from the same set S, such that the output signals of gates can be used as inputs as well. We can easily represent this kind of network by a matrix $N_{TL} = (r_{ij})^{m \times k}$, where r_{ij} is the output $o_v^{(u)}$ of another gate $p_f^{(u)}$ that is then used as the j-th input $s_j^{(i)}$ of gate $p_f^{(i)}(..., o_v^{(u)}, ...)$, such that $s_j^{(i)} = o_v^{(u)}$.

$$N_{TL} = \begin{array}{c} \\ p_f^{(1)} \\ p_f^{(2)} \\ \\ p_f^{(m)} \end{array} \begin{array}{c} \begin{array}{cccc} s_1 & s_2 & & s_k \end{array} \\ \begin{pmatrix} r_{11} & r_{12} & \ldots & r_{1k} \\ r_{21} & r_{22} & \ldots & r_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \ldots & r_{mk} \end{pmatrix} \end{array}$$

Hence, each row $p_f^{(i)}(r_{i1}, r_{i2}, \ldots, r_{ik})$ in the matrix represents the list of inputs supplied to gate $p_f^{(i)}$ that are taken from the other gates' outputs. Examples for this representation are given in Figure 5.3.

By using this matrix description of the network, each input of each gate $p_f^{(i)}$ is automatically connected to exactly one other gate's output. Additionally, at least one output(s) of each gate $p_f^{(i)}$ in the system should be used at least once in the tautological loop N_{TL}, thus:

$$\forall i \in \{1, .., m\} : \exists (a, b, c) : r_{ab} = o_c^{(i)}$$

Or, alternatively, if all outputs of each gate should be used, the condition would read:

$$\forall (i, j) \in \{1, .., m\} \times \{1, .., l\} : \exists (a, b) : r_{ab} = o_j^{(i)}$$

Potentially, tautological loops might lead to better results if they are (i) connected networks and if (ii) no output of a gate is used as its own input (self-loops). Further

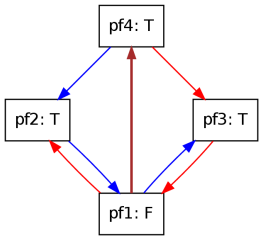
$$\begin{array}{c} \\ p_f^{(1)} \\ p_f^{(2)} \end{array} \begin{array}{c} s_1 \\ \begin{pmatrix} o_1^{(2)} \\ o_1^{(1)} \end{pmatrix} \end{array}$$

(a) Network N_{TL-NOT}

(b) Graph representation of N_{TL-NOT}

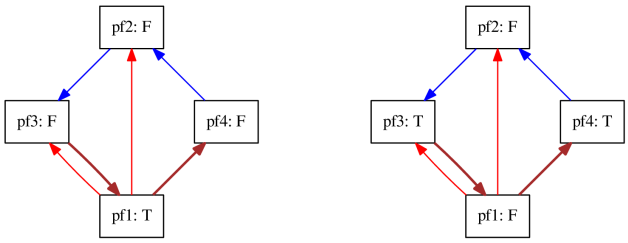
$$\begin{array}{c} \\ p_f^{(1)} \\ p_f^{(2)} \\ p_f^{(3)} \\ p_f^{(4)} \end{array} \begin{array}{c} s_1 \quad s_2 \\ \begin{pmatrix} o_1^{(2)} & o_1^{(3)} \\ o_1^{(4)} & o_1^{(1)} \\ o_1^{(1)} & o_1^{(4)} \\ o_1^{(1)} & o_1^{(1)} \end{pmatrix} \end{array}$$

(c) Network $N_{TL-NAND}$



(d) Graph representation of $N_{TL-NAND}$

Figure 5.3: Tautological loop networks in matrix (a,c) and graph (b,d) description that show stable states for the NOT (a,b) and for the NAND (c,d) function. The node names in panels (b,d) contain the Boolean value of the solving output state for each gate. The color of an arrow entering a gate indicates which of both inputs is connected to the source: red and blue arrows correspond to the inputs s_1 and s_2, while thick brown arrows indicate that both inputs are taken from that source.



(a) Network N_{1226}, favorable state

(b) Network N_{1226}, unfavorable state

Figure 5.4: The tautological loop N_{1226} for NAND and NOR gates has two different stable states, displayed in subfigures a) and b). Only the configuration in subfigure a) leads to a favorable state, where all input configurations are shown The configuration in panel b), on the other hand, is stable but not useful as a tautological loop because not all input configurations are exemplified. In this case, p_{f_4} but p_{f_3} are both presented with the same input pattern $\{T, T\}$.

studies will have to show the effect of these demands.

5.1.3 Estimating Tauological Loop Quality

Now that we have defined the networks, the major property of tautological loops is that their dynamics should be able to settle in an attractor, such that we can observe the final signals and use them in the abstraction and concretization functions. But before we can use the networks for this purpose, first we have to find suitable network topologies. As exemplified in Figure 5.4, not all network topologies are equally useful, for example because additional, unfavorable stable states can emerge in the network. Instead of the gate implementations $p_f^{(i)}$, we assemble the tautological loop from identical instances $f^{(i)}$ of the original function f here. When the network will be applied for finding signal encodings later, more complex attractors will probably be found, e.g. oscillations might be part of a valid voltage signal. But in the space of abstract symbols X, the abstract computation f is supposed to be deterministic, such that the result of a computation $y = f(x)$ should always be the same output y for a

given input vector x. So for the following theoretical considerations, we will stay with fixed point attractors to find valid tautological loop architectures. To this aim, we first choose the trivial gate implementations p_f to be exactly the abstract functions $p_f = f$. Consequently, the space of signals S becomes the space of abstract symbols X. Now we can search and investigate network topologies that support fixed point attractors and later use these topologies to design computing gate / signal pairs that work together nicely, even if the attractors in the tautological loop are more complex then.

Let $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_k^{(i)}) \in X^k$ be the list of the current symbols that are used as parameters to the function $y^{(i)} = f^{(i)}(x^{(i)})$. To define the state of the tautological loop, we denote the current output of the function $f^{(i)}$ as $y^{(i)} = (y_1^{(i)}, y_2^{(i)}, \ldots, x_l^{(i)}) \in X^l$. $x^{(i)}$ is assembled according to the tautological loop topology N_{TL} from the list of current output symbols $y^{(*)}$ in the network. Hence the state of the whole network is defined by the list of all functions' current outputs $y^{(*)} = (y^{(1)}, ..., y^{(m)}) \in X^{l \cdot m}$. Then the network state $\widetilde{y}^{(*)}$ is in a fixed point attractor iff $\forall i \in \{1, .., m\} : \widetilde{y}^{(i)} = f^{(i)}(\widetilde{x}^{(i)})$, such that no output symbol is changing any more.

Obviously, having attractors alone is not sufficient, as for example if f was a Boolean AND function, setting all signals to *false* would also constitute a fixed point attractor. But, if it was our task to design an electronic AND gate, this "all *false*" attractor would not be useful (i) because not all values from X are present in the attractor, thus we could not "harvest" the observed signals as a template for our abstraction/concretization function *ab* and (ii) because we could not be sure if the the designed gate would be working under all possible input conditions from X^k. Consequently we require the network template N_{TLT} to exhibit attractors that contain all possible input situations from X^k.

$$\bigcup_{i \in \{1,..,m\}} \{x^{(i)}\} = X^k \tag{5.1}$$

And additionally, when the tautological loop N_{TLT} is in a fixed point attractor $\widetilde{y}^{(*)}$,

we require its current output values $y_j^{(i)}$ to cover all possible symbols from X.

$$\bigcup_{(i,j)\in\{1,..,m\}\times\{1,..,l\}} \{y_j^{(i)}\} = X \tag{5.2}$$

Equation (5.1) already covers equation (5.2), because generating all possible input combinations already requires having all possible elements from X present in the system.

Furthermore, if there is an attractor $\widetilde{y}^{(*)}$ that covers all input situations, we expect tautological loop to work the better, the fewer other attractors $\widetilde{y}'^{(*)}$ there are that do not cover all input situations. The reasoning is, the more inappropriate attractors there are in the network template using the functions f, the more inappropriate attractors will we find in the network of actual implementations p_f.

Further studies might show if the size of the basins of attraction are of importance. Intuitively, a large basin of attraction for a suitable attractor might lead to faster convergence and more robust solutions when actual implementations p_f are used in the tautological loop.

5.2 Systematic Screening for Tautological Loops

In this section, we will consider only binary Boolean functions with exactly two inputs and one output to compose tautological loops of exactly four gates. For this kind of networks, we can enumerate all possible network structures and estimate how many of them are suitable as tautological loops in the sense that they (i) use all of the gates' outputs, that they (ii) have stable states and that they (iii) have at least one stable state in which each gate of the network is presented a unique input pattern, i.e. such that all input situations are exemplified.

As explained in Section 5.1.2, we represent potential tautological loop networks N_{TLi} as a matrix, which displays for each input of each gate the connected source, that is,

which other droplet's output is used as the source for this input:

$$N_{TLi} = \begin{array}{c} \\ p_f^{(0)} \\ p_f^{(1)} \\ p_f^{(2)} \\ p_f^{(3)} \end{array} \begin{array}{c} \begin{array}{cc} s_0 & s_1 \end{array} \\ \begin{pmatrix} o^{(c_0)} & o^{(c_1)} \\ o^{(c_2)} & o^{(c_3)} \\ o^{(c_4)} & o^{(c_5)} \\ o^{(c_6)} & o^{(c_7)} \end{pmatrix} \end{array} , \quad c_j \in \{0,1,2,3\}$$

As a short form, we only save the indices c_j as an 8-tuple

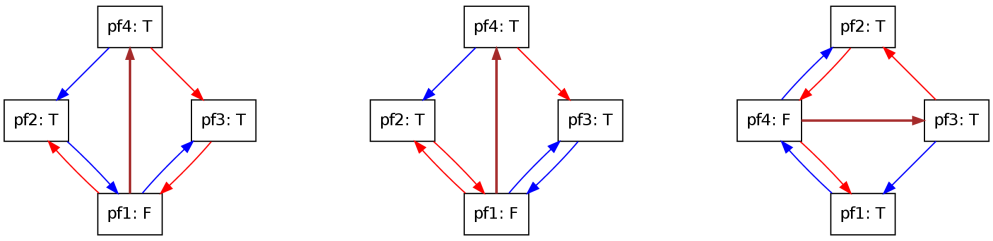
$$N_i = (c_0, c_1, ..., c_7),$$

where $c_j \in \{0,1,2,3\}$. A value c_j describes the (j mod 2)-th input source of node $\lfloor j/2 \rfloor$ to be connected to the output of node $p_f^{(c_j)}$. Thus, both inputs s_0 and s_1 to each gate are considered independently, because some gates g exist, for which $f_g(s_0, s_1) \neq f_g(s_1, s_0)$. This results in $4^8 = 65536$ different networks. To quickly discriminate networks, we will identify network $N_i = (c_0, c_1, ..., c_7)$ by its sequential number i, where i depends on the network description by the formula:

$$i = \sum_{j=0}^{7} c_j \cdot 4^j$$

Among these networks, there are many isomorphic networks that are identical except for exchanged indices of the nodes. When ignoring the numbering order of the nodes in the network, 3122 canonical networks remain.

In the following, we will independently consider those networks with- and without *self-loops*, i.e., gates that use the output of their computation as their own input. Of the 3122 canonical networks, 1980 with- and 337 without *self-loops* make use of all generated values, so that each calculated value is also fed back into the network. Then, dependent on the chosen Boolean function for the gates, the candidate networks are filtered for showing at least one stable state, for covering all input situations in those stable states and finally for having a minimal number of stable stats. Networks with



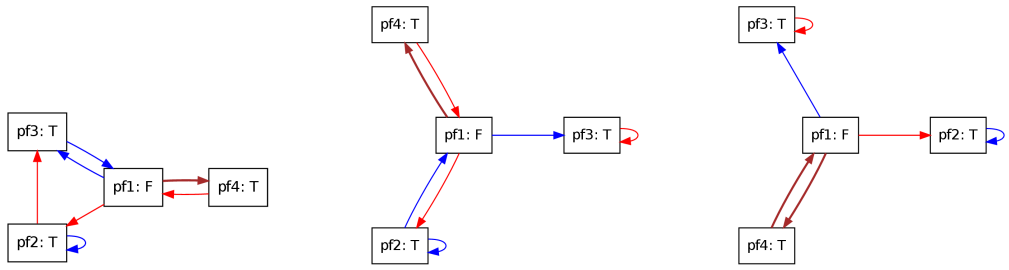
(a) Network N_{966} (b) Network N_{969} (c) Network N_{8171}

Figure 5.5: Tautological loops suitable for finding NAND gates that can be constructed without self-loops. All three networks are essentially the same, when allowing to exchange the inputs a and b to the Boolean function NAND(a, b). The node names contain the Boolean value of the solving output state for each gate. The color of an arrow entering a gate indicates which of both inputs is connected to the source: red and blue arrows correspond to the inputs a and b, while thick brown arrows indicate that both inputs are taken from that source.

a minimal number of stable states, appear more useful, because here the dynamics can easier settle into a particular envisaged stable state.

From the 'minStable' row of Table 5.2, we observe that only for the NAND and NOR gates, tautological loops can be found that show only a single stable state. When searching for tautological loops for other gate types, every network will have two or more stable states, such that the system's dynamic behavior can end up in a stable state that is not a solution because not all input situations are covered.

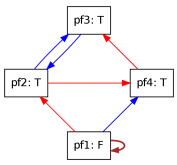
In Figure 5.5, we show all the network designs that can realize the NOR network topology without using *self-loops*. All three networks are very similar except for the assignment of inputs A and B. When allowing *self-loops*, the following network topologies appear as additional solution candidates for NOR gates, shown in Figure 5.6. An overview of exemplary tautological loops for all 14 non-trivial two-input Boolean gates is given in Figure 5.7.



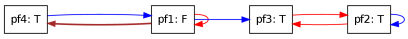
(a) Network N_{331} (b) Network N_{583} (c) Network N_{591}

(d) Network N_{709} (e) Network N_{710} (f) Network N_{711}

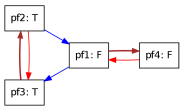
Figure 5.6: More tautological loops suitable for finding NAND gates, now allowing *self-loops* as well. The arrows are named and colored as in Figure 5.5. Networks that are equal when allowing to exchange the inputs a and b for some of the gates NAND(a, b) are omitted.



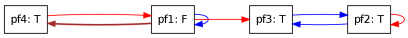
(a) G_1 0001_b, AND, N_{6016}



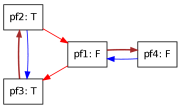
(b) G_2 0010_b, N_{364}



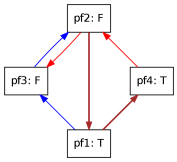
(c) G_3 0011_b, N_{423}



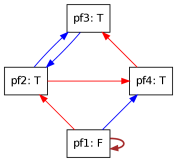
(d) G_4 0100_b, N_{1171}



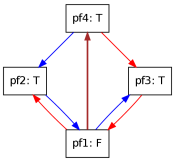
(e) G_5 0101_b, N_{1197}



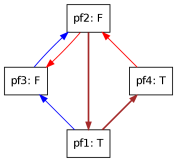
(f) G_6 0110_b, XOR, N_{437}



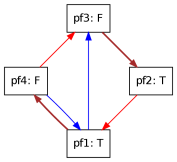
(g) G_7 0111_b, OR, N_{6016}



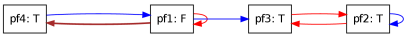
(h) G_8 1000_b, NOR, N_{966}



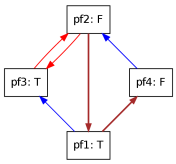
(i) G_9 1001_b, XNOR, N_{437}



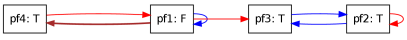
(j) G_{10} 1010_b, N_{941}



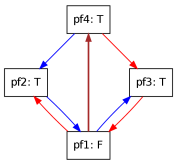
(k) G_{11} 1011_b, N_{364}



(l) G_{12} 1100_b, N_{485}



(m) G_{13} 1101_b, NAND, N_{1171}



(n) G_{14} 1110_b, N_{966}

Figure 5.7: Exemplary tautological loops for Boolean two-input gates G_i where i is the binary encoded output of the Boolean function for the inputs 00, 01, 10, and 11.

5.3 Implementation Fitness in the Tautological Loop

Once a valid tautological loop template is found using the original, abstract function f as described in the last section, the network can be observed or simulated with a putative implementation p_f and the signals at the appropriate network edges can be harvested to generate the concretization and abstraction functions *co* and *ab* as displayed in Figure 5.2.

But first it might be necessary to evaluate if (i) a gate implementation p_f is fulfilling its purpose and (ii) if the tautological loop network actually settled in an attractor such that the present signals bear any meaning. For simplicity, we assume here that each gate $p_f^{(i)}$ produces exactly one output signal $s^{(i)}$, such that $l = 1$ and we can omit the index of the gate's output. The general requirement for a set of signals $s^{(*)}$ in the tautological loop network is that two signals $s^{(i)}$ and $s^{(j)}$ are distinguishable iff the appropriate abstract symbols $x^{(i)}$ and $x^{(j)}$ in the tautological loop template are different.

$$\forall (i,j) \in \{1,...,m\}^2, i \neq j: \quad \begin{array}{lll} s^{(i)} = s^{(j)} & \text{if} & x^{(i)} = x^{(j)} \\ s^{(i)} \neq s^{(j)} & \text{if} & x^{(i)} \neq x^{(j)} \end{array}$$

Hence, there are two different fitness criteria, which are probably best treated using multi-criteria optimization techniques [Sch85, Zit04]. On the one hand, the gates in the network that are assigned different abstract symbols should be different. On the other hand, the gates that that are supposed to produce similar outputs should not be distinguishable.

5.3.1 Mutual Information Based Fitness

In order not to bias the selection of appropriate signal candidates by a particular spike-train similarity metric, we propose to use information theory as explained in Chapter 4. To this aim, we record the signal distributions $\mathcal{P}_{s^{(i)}}$ at the output of each gate $p_f^{(i)}$ from simulation or experiment. Distinguishing supposedly different symbols

is achieved by measuring the mutual information between signal distribution and symbol distribution:

$$q_{I_{pos}} = I(\mathcal{P}_{s^{(i)}}, \mathcal{P}_{x^{(i)}}),\ i \in \{1, .., m\} \tag{5.3}$$

For the other part of the fitness function, being unable to distinguish supposedly similar output symbols, we also suggest an information theoretic formulation: If, for any fixed abstract output symbol $\tilde{y} \in X$, there are multiple gates $\{p_f^{(a)} | f^{(a)}(x) = y^{(a)} = \tilde{y}, a \in \{1, \ldots, m\}\}$ in the network that produce this output symbol $\tilde{y}$ according to the tautological loop design N_{TL}, we measure the mutual information between the gate id a and the signal patterns found at these gates. In other words, if the gates' output signals contain information on the specific position a in the network, this would contradict the claim that the signals represent the same abstract symbol $\tilde{y}$. Hence we have at most $|X|$ penalizing terms, one for each $\tilde{y} \in X$:

$$q_{I_{neg-\tilde{y}}} = I(\mathcal{P}_{s^{(a)}}, \mathcal{P}_a),\ a \in \{i | y^{(i)} = \tilde{y}\} \tag{5.4}$$

Instead of using Pareto-optimization [Sch85, Zit04], a simple final fitness function can just combine the different fitness criteria q_{pos} and $q_{neg-\tilde{y}}$ in a linear combination to a single value q using a weighting factor ω:

$$q_I = \omega\ q_{I_{pos}} - \sum_{\tilde{y} \in X} q_{I_{neg-\tilde{y}}} \tag{5.5}$$

Assuming that the signals S in the network are a lot more noisy and complex than the abstract symbols X, we can also use the entropy $H(\mathcal{P}_{y^{(i)}})$ as upper bound for the mutual information and thus as a way of normalizing the fitness values to, e.g., a maximum of 1.

It might pose a problem for this approach, if multiple suitable fixed point attractors for the tautological loop N_{TL} are found. Sampling the tautological loop with its implementations might be irritating here, when different initial conditions lead to different attractors that can not trivially be distinguished in the formulation of

Equations (5.3) and (5.4). Then signals that should be similar might be different if taken from different attractors and vice versa, signals that should be different might be similar if taken from different attractors. Here it might be a solution to first classify the samples of the dynamics for its attractor and then to calculate the fitness function for each attractor individually.

5.3.2 Fitness based on Spike Frequency

As a simpler alternative, we can compare the spike frequencies $C^{(i)}$ that are measured at the different outputs of the gate instances p_{fi} in the tautological loop.

First, the differences in spike numbers between instances that are supposed to produce different signals are summed up:

$$q_{C_{pos}} = \sum_{(i,j)\in\{1,\ldots,m\}^2,\ i\neq j,\ x^{(i)}\neq x^{(j)}} |C^{(i)} - C^{(j)}| \tag{5.6}$$

Then the negative impact of the differences between instances that are supposed to produce imilar signals are summed up:

$$q_{C_{neg}} = \sum_{(i,j)\in\{1,\ldots,m\}^2,\ i\neq j,\ x^{(i)}=x^{(j)}} |C^{(i)} - C^{(j)}| \tag{5.7}$$

Finally, a linear combination of the positive and negative spike number differences with the factor ω can be used as the final fitness:

$$q_C = q_{C_{pos}} - \omega q_{C_{neg}} \tag{5.8}$$

5.4 Tautological Loops for Droplet Computers

To demonstrate that the tautological loop method can be used to design NAND-gates in more complex non-linear systems, we used the method to identify configurations of simulated droplet-computers (cf. Chapter 2) that can operate as NAND-gates. For

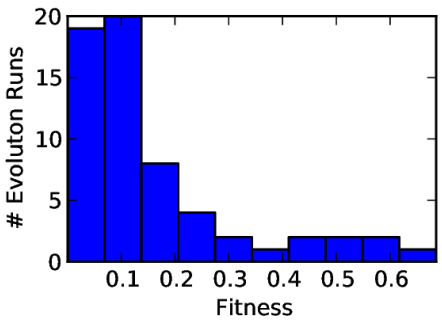
the simulation, we use the event-based simulation system 'DropSim', that abstracts complex chemical processes to only the three states *excited*, *refractory* and *responsive* and allows for very fast simulations of large droplet systems, as also explained in Chapter 2 and in [Gru13]. Nonetheless, evolution of particular Boolean gates, especially of NAND and NOR gates is not trivial in this system as discussed in Chapter 3.

For the evolution of a NAND gate from simulated BZ droplets, we used planar designs in a 5x5 grid where each horizontally and vertically adjacent droplet is connected. The normal droplets in simulation stayed excited for one second, were refractory for five seconds and then self-excited after 10 seconds if not externally triggered. The signal propagation delay from one droplet to the next was one second. For each of these values, a normal distributed noise term of the standard deviation 0.05 s is added for each event, such that the simulation becomes non-deterministic.

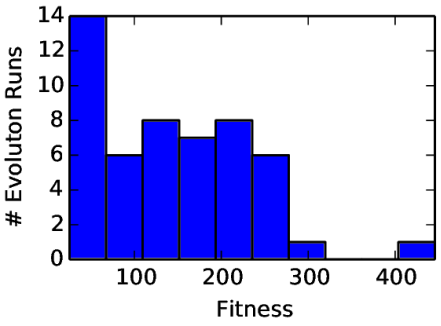
Similar to the evolution experiments in Chapter 3, the mutation function can exchange each of the 25 positions in the grid with four different droplet types, including no droplet. The remaining three droplet types were a normal droplet as described above, a slightly faster oscillating droplet with 0.8 times the original period and a less excitable droplet that requires two concurrent excitation at its neighbors to be triggered into an excitation. The probability for exchanging a droplet position in the grid was 0.05 per position.

Evolutionary studies were conducted in a 50-fold multi-start approach, where each evolution run consisted of 500 generations. For each run, a population of 10 parents was selected by truncation selection from 37 children and 3 random immigrants. Single point cross over was used for recombination.

To evaluate the fitness of a droplet network individual, we connected four instances of the individual in a tautological loop as proposed in our first initial tautological loop design from Figure 5.4 (a). As a fitness function, we used the mutual information based fitness formulation from Equation (5.5) of the last section. Because the resulting networks were not completely suitable for combination to a half-adder system, we also tried the fitness formulation from Equation (5.8). Because of the stochastic nature of



(a) Mutual information based fitness evaluation.



(b) Spike-frequency based fitness evaluation.

Figure 5.8: Distribution of final fitness values for 50 independent evolution runs using (a) the mutual information based fitness evaluation from Equation (5.5) and (b) the spike-frequency based fitness evaluation from Equation (5.8).

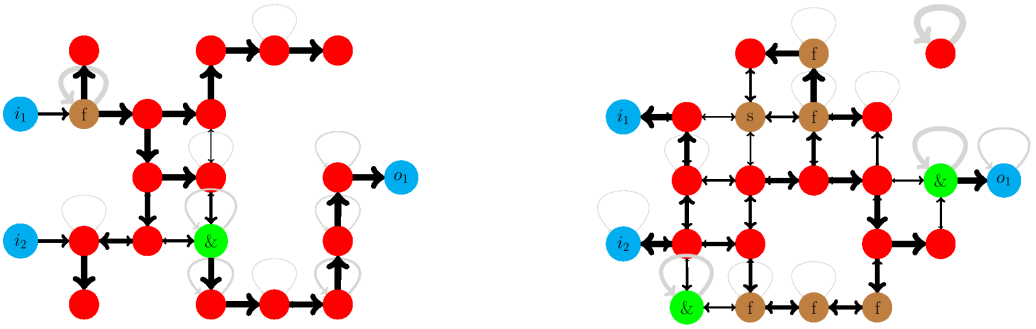
droplet simulations, we repeated 10 instances of the droplet simulation process, each of which returned one mutual information value q_i from Equations (5.5) or (5.8). Then, the final fitness q_f is the averaged fitness $\overline{q}$ over the ten repetitions minus $\frac{1}{10}$ of the standard deviation to penalize strong differences between repetitions of the experiment:

$$q_f = \overline{q} - \frac{1}{10}\sqrt{\frac{1}{10}\sum_{i=1}^{10}(\overline{q} - q_i)^2}$$

Additionally to the evolved part of the network, we added 'diode droplets' [Szy11] at the connections between the instances that allow the signal propagation to proceed only in the direction that is symbolized by the arrows in Figure 5.5.

Results for Mutual Information based Fitness Evaluation

The distribution of the final fitness values for the 50 independent evolution runs are shown in Figure 5.8 (a), exhibiting that only a small fraction of the evolution runs end up in the optimal fitness range. The best individual in the final generation of the best evolution run is shown in Figure 5.9 (a). The evolution dynamics of this run are shown in Figure 5.10 (a), where it becomes obvious that even with the averaging,



(a) Mutual information based fitness evaluation.

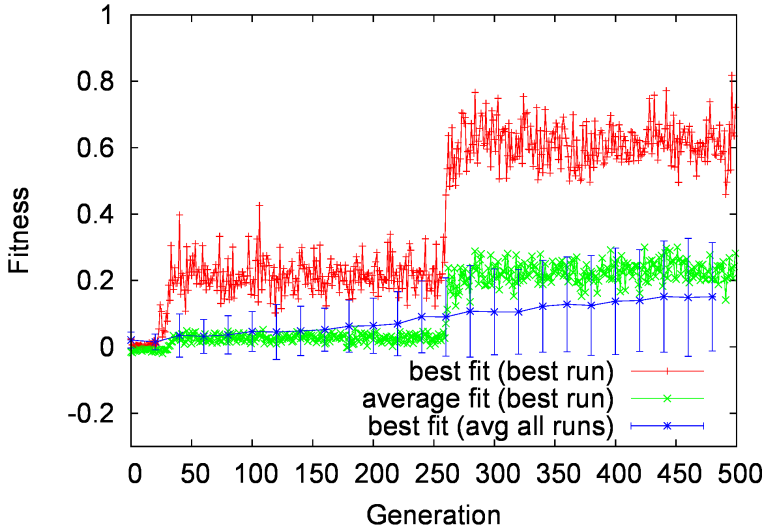
(b) Spike-frequency based fitness evaluation.

Figure 5.9: Evolved droplet network realizing the NAND function using (a) the mutual information based fitness evaluation from Equation (5.5) and (b) the spike-frequency based fitness evaluation from Equation (5.8). Blue droplets are the inputs (left) and outputs (right), red droplets are normal droplets, brown droplets are faster (f) and slower (s) oscillating droplets, green droplets (&) are less excitable.

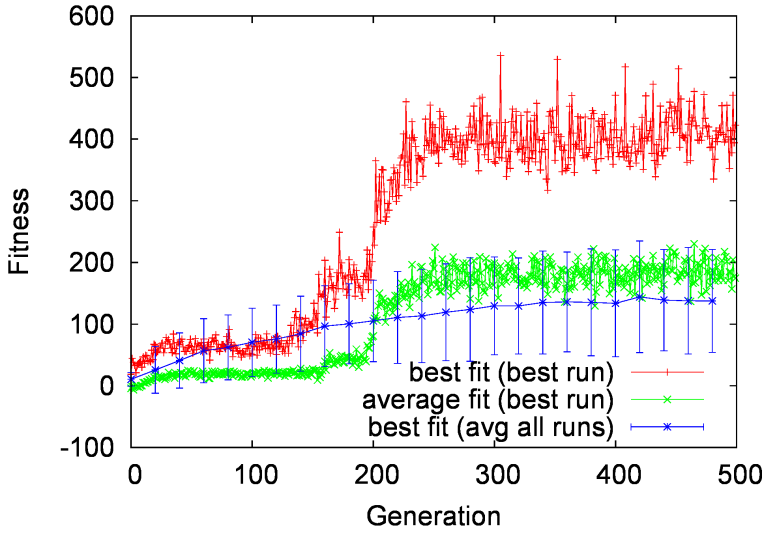
there is a strong stochastic influence in the fitness evaluation.

For the droplet network instance from Figure 5.9 (a) and for all switching processes between different input combinations, we plotted the network behavior in terms of spike frequency for an exemplary single simulation instance as well as averaged over 100 simulation runs in Figure 5.11. Looking at the individual spike patterns in Figure 5.12, very regular spike patterns are found such that we conclude that the system actually settles in a frequency coding regime. From the properties of the output droplets, a range of spike frequencies between 0.06 and 0.16 spikes per second would be possible. Within this range, when applying a threshold value of 0.08 spikes per second and when interpreting a low spike frequency (ca. 0.065) as a logical '0' and a high spike frequency (ca. 0.095) as a logical '1', the droplet network behaves as a NAND gate.

Then, to further evaluate the utility of the evolved droplet network, we connected multiple instances of the evolved NAND gate to a half-adder as illustrated in Figure 5.13. While a single gate instance seems to function adequately, the droplet network that was evolved using the mutual-information fitness function did not produce



(a) Mutual information based fitness evaluation.



(b) Spike-frequency based fitness evaluation.

Figure 5.10: Best (red) and average (green) fitness per generation from the best of the 50 indepentant evolution runs using (a) the mutual information based fitness evaluation from Equation (5.5) and (b) the spike-frequency based fitness evaluation from Equation (5.8). The average of the best individual of the final generation of each of the 50 runs is plotted in blue with errorbars indicating the relatively large standard deviation.

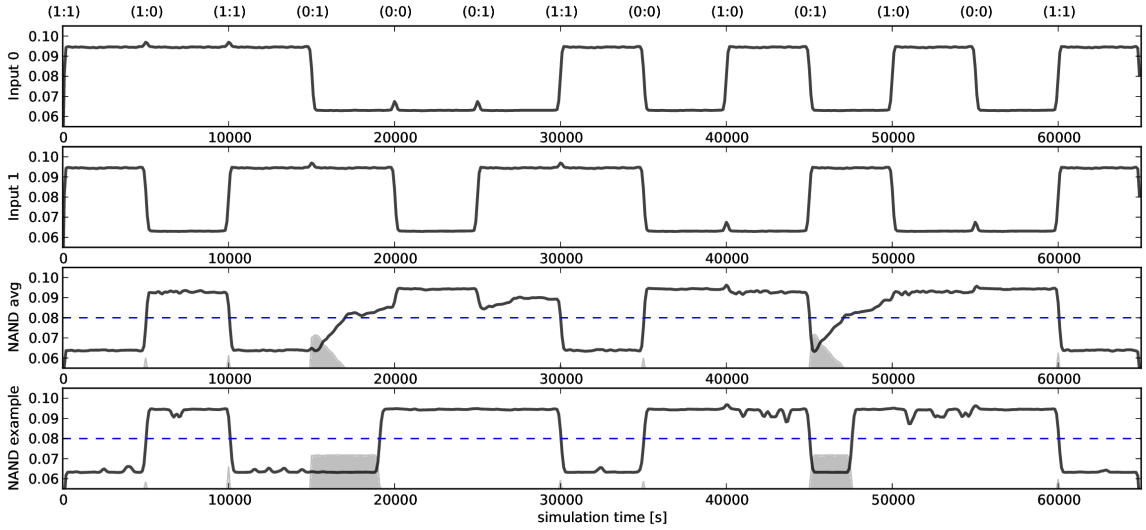
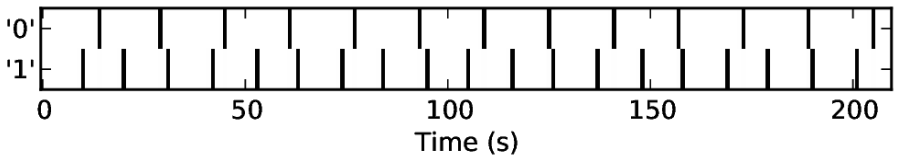
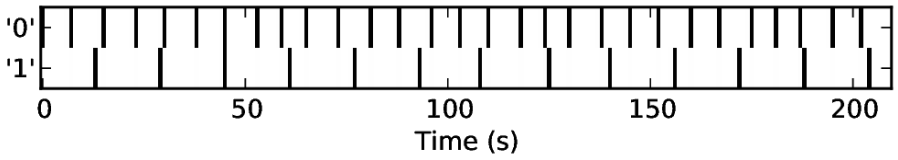


Figure 5.11: Input and output frequency of the evolved NAND network using the mutual information fitness function for all four input configurations, averaged over 100 runs. Each stimulation pattern (indicated in the top two plots of the figure) is applied for 5000 seconds. We interpret a high oscillation frequency (more than 0.08 spikes/second) as a logical one and a low oscillation frequency (less than 0.08 spikes/second) as a logical zero. The blue, dashed line in the NAND plot indicates an externally applied threshold function that can be used to distinguish a high from a low frequency. The gray peaks in the lower two plots indicate the error of the NAND output as the distance of the output frequency from the blue threshold line, if the signal is on the wrong side. While the third plot shows the average behavior of the 100 simulation runs, the lowest plot shows an arbitrary single trajectory, showing that especially the slow switching processes from state (11) and (10) to state (01) happen instantly when they happen but only after some time. This leads to an averaged signal that slowly climbs from a low to a high spike frequency in these cases.



(a) Mutual information based fitness evaluation.



(b) Spike-frequency based fitness evaluation.

Figure 5.12: Emergent symbol encodings that were produced in the tautological loop, evolved with (a) the mutual information based fitness evaluation from Equation (5.5) and (b) the spike-frequency based fitness evaluation from Equation (5.8). Only a small but representative time interval is shown here. A black vertical bar indicates an excited droplet at this time. The emergent spike patterns do not vary strongly over time, such that effectively rate coding symbols appeared. The rates in the unit 'spikes per second' for the symbols zero and one are approximately 0.065 and 0.095, respectively, for the mutual information fitness function (a) and 0.14 and 0.06, respectively, for the spike-frequency fitness function.

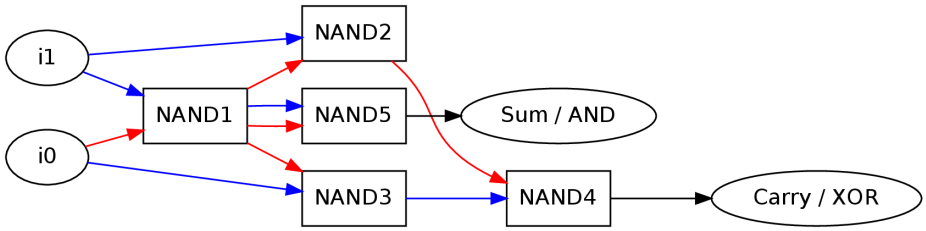


Figure 5.13: Half-adder assembled from the evolved NAND gates. Round nodes represent input and outputs to the network while rectangles represent independent instances of the evolved NAND gate. Similar to the earlier tautological loop networks, the inputs to each gates are symbolized by the colors red and blue.

observable results on all output channels for the half-adder: The Sum/And output channels stays at the output value of '1' for all input combinations. Nonetheless, the Carry/XOR output channel produces a useful output. The different switching processes between all input combinations are shown in Figure 5.14. Most probably, the poor switching behavior of the Sum/And channel is due to an amplification of stochastic variations in the output signals of each individual NAND gate, where especially switching from input combination (1:1) to (0:1) and from (1:0) to (0:1) seemed slow and error-prone as observed in Figure 5.11.

Results for Spike-Frequency based Fitness Evaluation

Because the composition of a hald-adder network from our evolved NAND gates, using the mutual information fitness function, did not produce satisfactory output signals, we additionally tested evolving a gate with the alternative, spike-frequency based fitness formulation from Equation (5.8). This resulted in the distribution of final fitness values displayed in Figure 5.8 (b). The droplet network shown in Figure 5.9 (b) is one of the final individuals of the evolution run with the highest maximum fitness. The evolution dynamics of this run are shown in Figure 5.10 (b), where it becomes obvious that similar to the mutual information fitness function, there is a strong stochastic influence in the fitness evaluation and that the fitness typically does

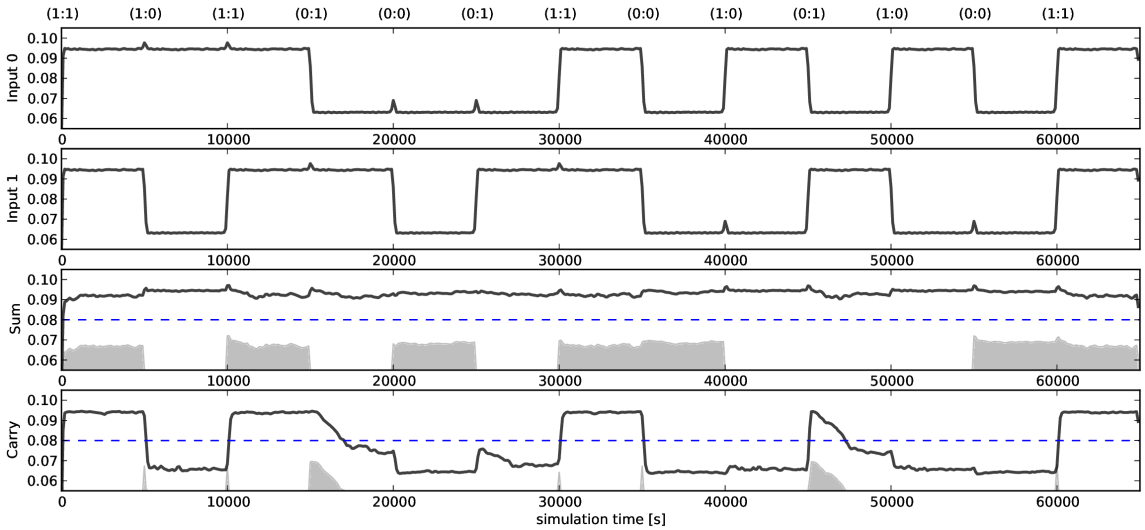


Figure 5.14: Input and output frequency of the half-adder network constructed from the mutual information fitness evolved NAND gate for all four input configurations, averaged over 100 runs. Each stimulation pattern is applied for 5000 seconds. We interpret a high oscillation frequency (more than 0.08 spikes/second) as a logical zero and a low oscillation frequency (less than 0.08 spikes/second) as a logical one. The gray peaks in the lower two plots indicate the error of the output: It indicates the distance of the output frequency from the blue threshold line, if the signal is on its wrong side. The third plot with the output of the Sum/And function exhibits large error values, not seeming to show anything else but a high output frequency, while the Carry/Xor function output appears almost as good as the output of a single NAND gate in Figure 5.11.

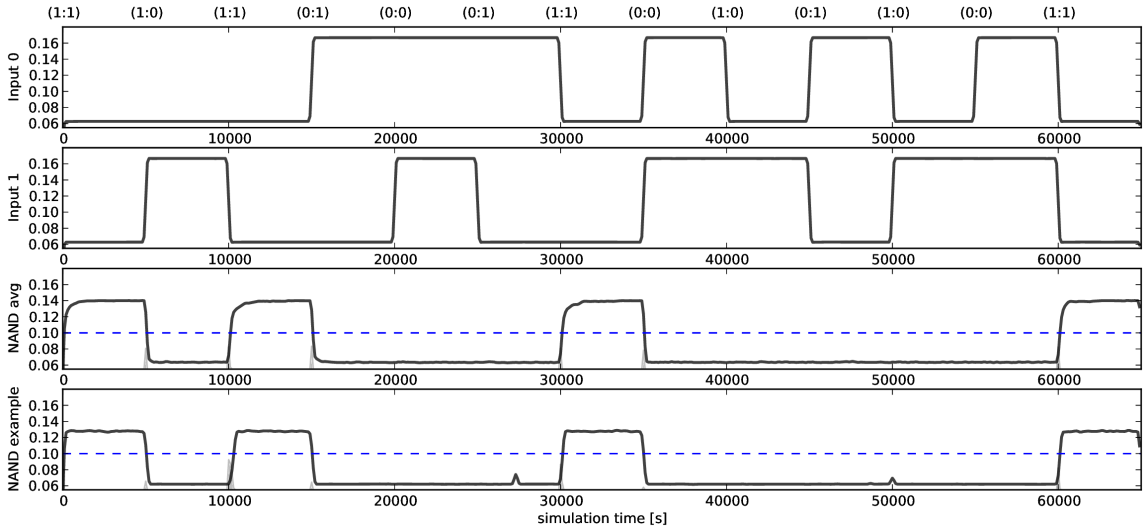


Figure 5.15: Output frequency of the evolved NAND network for all four input configurations, averaged over 100 runs. Each stimulation pattern (indicated on top of the figure) is applied for 5000 seconds. We interpret a high oscillation frequency (more than 0.10 spikes/second) as a logical zero and a low oscillation frequency (less than 0.10 spikes/second) as a logical one. The blue, dashed line in the NAND plots indicate an externally applied threshold function that can be used to distinguish a high from a low frequency. The gray peaks in the lower plots indicate the error of the NAND output, i.e., the distance of the output frequency from the blue threshold line if the signal is on the wrong side.

not increase continuously but takes almost discrete jumps to higher fitness values.

For all switching processes between different input combinations, we plotted the network behavior for an exemplary single simulation run as well as averaged over 100 simulation runs in Figure 5.15. When applying a threshold value of 0.10 spikes per second, which is higher than the 0.08 spikes per second in the mutual information fitness case, the network behaves like a correct NAND gate. Again, using the spike-frequency based fitness, we tried connecting multiple instances of the evolved NAND gate to a half-adder as illustrated in Figure 5.13. Here, both output channels for the Sum/And as well as for the Carry/Xor signal produce useful output frequencies as shown in Figure 5.16. Nonetheless, in the simulated 5000 s per input configuration, it took the half-adder network longer on average to move the output spike frequency

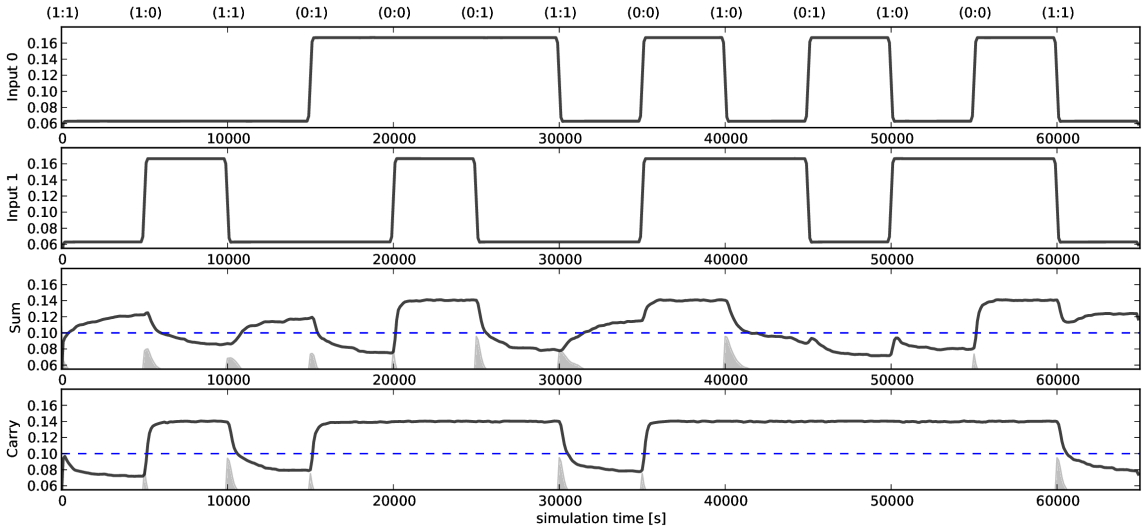


Figure 5.16: Output frequency of the half-adder network constructed from the evolved NAND gate for all four input configurations, averaged over 100 runs. Each stimulation pattern is applied for 5000 seconds. We interpret a high oscillation frequency (more than 0.10 spikes/second) as a logical zero and a low oscillation frequency (less than 0.10 spikes/second) as a logical one. The gray peaks in the lower two plots indicate the error of the output as the distance of the output frequency from the blue threshold line, if the signal is on its wrong side. On average, when measured at the end of each phase, e.g. 4900 seconds after each input change, all ouputs behave correctly like a half-adder.

towards the desired region. This implies that it will not trivially be possible to build arbitrarily large logical systems from these NAND gate.

5.5 Discussion

In this Chapter we presented the concept of tautological loops or RERUN networks and the idea was described both visually and formally. Furthermore, we considered conditions that lead to networks that can effectively be used for finding symbol encodings, albeit the practical use of these conditions still needs to be proven or examined. In order to evaluate potential gate implementations together with the self-organizing symbol encoding, we additionally showed how to extend the information-theoretic

analysis from Chapter 4 for tautological loops as a fitness function for evolutionary computation. As an example of the applicability of the tautological loop network concept, we ran evolution experiments with simulations of artificial droplet computers as modeled in Chapter 2. In this process, we evolved two different NAND gates that used different spike rate encodings for the symbols zero and one. We also verified the function of the NAND gates by coupling them to form a functional half-adder, which was successful only for one of the evolved NAND gates.

In the practical evolution experiments carried out in this chapter, even though the tautological loop was designed to fulfill the NAND-gate function, a NOR-gate and a NAND-gate are equivalently evolved, dependent on the interpretation of the observed signal patterns. When assuming that a low spike rate is interpreted as a logical zero and a high spike rate as a logical one, the gate design in Figure 5.9 (a) corresponds to a NAND gate here. But obviously, when inverting this assignment, a NAND gate becomes a NOR gate for the same operation on high and low signals, as explained in Table 5.3.

This interchangability of the symbol encodings might also be an obstacle for evolutionary optimization with tautological loops, because therefore, in our examples, there are always at least two stable states. When the system changes from one stable state to another, low fitness values are generated, because the signals in wires that are supposed to carry different symbols exchange. Consequently, it might help in the evolutionary process to gradually lengthen the simulation time, in such that early droplet network designs are not discarded because of occasional signal inversions.

Another interesting finding of the experiments carried out here is that even though the symbols that evolved in Section 5.4 were temporarily stable in the tautological loop during stochastic fitness evaluation, the resulting gates did not automatically exhibit error-correcting properties. Especially the droplet network in Figure 5.9 (a) was not suitable for composing a half-adder from individual gate instances. So apparently, the emerging symbols were temporarily stable in the tautological loop system but not necessarily outside, when run through the evolved gates by thesmselves. One reason might be timing effects, that work only because of the exact lengths of the

connecting segments between the gate instances in the loop. Here it could be tried to use arbitrary lengths for the connecting segments between the gate instances in the loop to reduce effects that require an exact timing. Additionally, it might prove effective to cycle through tautological loop designs with similar properties, based on our classifications in Section 5.1.3 in an evolutionary run. Also, similar to conventional electronics, switching processes in gates require time, such that only a limited number of gates can be combined in a clock cycle.

In following studies, it should be investigated whether the use of tautological loops bears actual advantages in the search for symbol encodings when compared to symbol co-evolution as shown in Chapter 3 or to naive enumerative approaches. For example, tautological networks might be useful for finding symbol encodings with less computational effort or to find encodings that have better error correction properties. Also cryptographic hash functions like the secure hashing algorithm (SHA, [Nat12]) might be a useful "substrate" for further studying the applicability of tautological loop topologies. A variable number of input and output bits could then be considered, such that the symbol complexity can be analyzed on different scales. Such cryptographic hash functions would be an extreme case, where it should be especially hard to find suitable symbol encodings by rational design decisions, because inputs and outputs are supposed to be not trivially related. An approach in this direction was taken in the bachelor's thesis [Ruh14].

In the tautological networks presented in this chapter, different instances of the same basis function f are connected in a loop to find suitable symbol encodings. But while NAND gates can be used to build all other binary Boolean gates, for other, maybe not Boolean information processing modules, it might be necessary to wire different basis functions together, probably resulting in more complex tautological loops.

		including self-loops			excluding self-loops		
Gate	Pattern	Stable	Covering	minStable	Stable	Covering	minStable
0 - FALSE	0000	1980	-	-	337	-	-
1 - AND	0001	1980	14	3: 8	337	-	-
2	0010	1980	52	2: 23	337	-	-
3	0011	1980	81	4: 52	337	5	4: 5
4	0100	1980	47	2: 23	337	-	-
5	0101	1980	49	4: 38	337	3	4: 3
6 - XOR	0110	1980	260	2: 182	337	28	2: 28
7 - OR	0111	1980	14	3: 8	337	-	-
8 - NOR	1000	708	57	1: 32	277	28	1: 3
9 - XNOR	1001	1980	260	2: 182	337	28	2: 28
10	1010	538	162	2: 149	199	47	2: 43
11	1011	1980	52	2: 23	337	-	-
12	1100	513	149	2: 135	214	52	2: 47
13	1101	1980	47	2: 23	337	-	-
14 - NAND	1110	708	57	1: 32	227	28	1: 3
15 - TRUE	1111	1980	-	-	337	-	-

Table 5.2: Number of possible tautological loops from four two-input Boolean gates with and without allowing self-loops.. The *Stable* row contains the number of networks that have at least one stable solution. Of those networks, the 'Covering' row contains the number of networks that have a stable solution that would generate every input situation. Of those networks, the 'minStable' rows contains the number of networks that show exactly k stable states, where k is the minimal number of stable states found.

i_0	i_1	output
A	A	B
A	B	B
B	A	B
B	B	A

(a) abstract

i_0	i_1	output
0	0	1
0	1	1
1	0	1
1	1	0

(b) A∼'0', B∼'1'

i_0	i_1	output
1	1	0
1	0	0
0	1	0
0	0	1

(c) A∼'1', B∼'0'

Table 5.3: Input/output signal table for NAND and NOR gates. In table a), A and B represent signals that encode Boolean values. When A represents a '0' and B a '1' as in table b), the table becomes a truth table for the NAND function. On the other hand, when A represents a '1' and B a '0' as in c), the table becomes the truth table for a NOR function. Because of this symmetry and the principle of not presetting the symbol encoding for the tautological loops, we can not distinguish a NAND gate from a NOR gate.

Chapter 6

Embodied Evolution

Evolutionary computation [Fog66, Rec71, Sch75, Hol75, Koz89, Fog94, Bey02, Wei02, Eib08] were used more excessively in the sense of genetic programming [Koz89] in Chapter 3 to find the algorithm or program p_f that computes some function f. In this section, instead, a step towards embodied evolution is taken such that the algorithm p_f itself is an evolutionary process that is embedded into the unconventional computing substrate. Although embodied evolution is defined as evolution happening in a real-world population of robots [Wat02], the agents here are molecular machines whose dynamics will only be explored in simulation.

In evolutionary computation systems, the optimality criteria, i.e., the fitness functions are often evaluated in the same digital computer that runs the evolutionary algorithm, for example in simulation or as ODE integration. When externalized however, the fitness evaluation can be realized in a technical or physical system as CCE (Computer Controlled Evolution) [Hig96, Dit98, Mil14] or even by human beings [Tak01]. Furthermore, also the selection operation can be externalized, e.g. by biochemistry methods like the SELEX (Systematic Evolution of Ligands by Exponential Enrichment) technique [Klu94, Mat06b]. Still, in contrast to the system proposed here, the populations of most common evolutionary computation systems are exclusively held as data structures maintained on digital computers and the generation scheme is mostly controlled externally.

In the following work that was done in collaboration with Gabi Escuela and Thomas Hinze in [Gru11b], the concept of molecular embodied evolution is elaborated in the context of solving the exact set cover problem. An evolutionary process is exemplified here that is, similar to a biological organism, embedded in a single physical world, which contains the fitness evaluation but also the population, the selection, the mutation operations as well as the control of an asynchronous generation scheme as proposed by Banzhaf [Ban90]. This single physical world of the work shown here is being simulated using the rule-based reaction system SRSim [Gru10] for structured molecules.

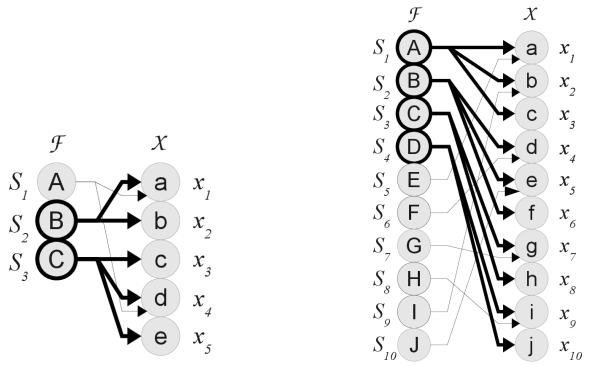
6.1 The Exact Set Cover Problem

We will first give a formal definition for the exact set cover problem, which is NP-complete [Kar72]: Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be a finite set of elements and $\mathcal{F} = \{S_1, S_2, \ldots, \mathcal{S}_k\}$ be a family of sets (also called collection), such that $\forall i \in \{1 \ldots k\} : (S_i \neq \emptyset) \wedge (S_i \subseteq \mathcal{X})$. Then, $\mathcal{C} \subseteq \mathcal{F}$ is an *exact cover* of $\mathcal{X}$ if and only if:

1. $\mathcal{X} = \bigcup_{S \in \mathcal{C}} S$, and
2. $\forall S, S' \in \mathcal{C}, S \neq S' : \; S \cap S' = \emptyset$.

The decision-problem here is defined as the question, if such an exact cover $\mathcal{C}$ exists, given the set of elements $\mathcal{X}$ and the set of subsets $\mathcal{F}$. To find all the solutions for this problem, Knuth [Knu00] proposed a so-called Algorithm X, a recursive, non-deterministic algorithm that combines depth-first and backtracking as programming strategies, and as data a matrix that represents the relation "contained in" between elements and subsets.

The exact set cover problem shares common properties with other NP-complete problems, including those that consider other kinds of cover. Set cover problems, in general, have their practical applications in conservation biology [Moo03], phylogenetics [Hal04] and protein identification [He11].



(a) Five elements and three subsets

(b) Ten elements and ten subsets

Figure 6.1: Two exemplary instances of the exact set cover problem: Circles on the left of each graph represent the different subsets $A, B, C, ...$ of the elements $a, b, c, ...$, which are shown on the right side of each graph. Arrows leaving a subset indicate all the elements that belong to this subset. Hence, these arrows are a representation of the externally given problem instance and cannot be modified by the algorithm. The thick circles on the left of each panel indicate the perfect set of subsets and thus a solution to the exact set cover task. This means that each element on the right is reached by exactly one arrow, when exactly the thick subsets on the left are chosen. In a simple instance of the problem $\mathcal{X} = \{a, b, c, d, e\}$ and $\mathcal{F} = \{A = \{a, d\}, B = \{a, b\}, C = \{c, d, e\}\}$, the subsets B and C lead to the perfect solution, while A only covers the elements a and d but also disallows the subsets B and C to cover more elements. (b) In the larger example with $\mathcal{X} = \{a, b, c, d, e, f, g, h, i, j\}$ and $\mathcal{F} = \{\{a, b, c\}, \{d, e, f\}, \{g, h\}, \{i, j\}, \{a\}, \{d\}, \{g\}, \{i\}, \{b\}, \{e\}\}$ there are ten elements as well as ten subsets of elements.

Two instances of the set cover problem are displayed in Figure 6.1. The search space is growing exponentially with the number of available *solution candidates* $C_i \in \mathcal{P}(\mathcal{F})$ from the power set of the available subsets. Each element C_i of the power set has to be considered as a potential solution. This means that the chance of randomly generating a correct solution is $\frac{1}{2^{|\mathcal{F}|}}$, where the notation $|\mathcal{F}|$ denotes the cardinality of the set $\mathcal{F}$. This implies a probability of $\frac{1}{8} = 12.5\%$ in the case displayed in Figure 6.1a and $\frac{1}{1024} \approx 0.1\%$ in the case of 10 subsets as in Figure 6.1b.

6.2 An Evolutionary Algorithm in Rule-Based Chemistry

We follow the design principles of an object centered view, where each of the molecules represents a possible solution candidate object. Each of the objects is following its own heuristic reaction path that can lead it to become a perfect solution to the exact set cover problem. This happens fully concurrently, i.e., there is no temporal dependency between different global phases of the algorithm that would have to be controlled from the outside.

We use molecular descriptions of molecules that evaluate, select and reproduce solution candidates, constituting a molecular implementation of a heuristic optimization algorithm that uses gradient information. In particular, we present an instance of an asynchronous evolutionary algorithm that is purely implemented in rule-based chemistry [Hla06, Gru10], similar to the evolutionary algorithm that was described by Banzhaf, 1990 [Ban90] but, in this case, used for the exact set cover problem. The population of individuals from the search space, the calculation of the fitness, the reproduction as well as the selection are realized by structured molecules in the reactor and the limited set of reaction rules between them.

Also different membrane computing frameworks, like the *Membrane Algorithms* that were presented by Nishida in 2005 [Nis06], can be used to implement evolutionary algorithms. But in contrast to these, our approach uses local rules instead of a global timing scheme or instead of a global spatial division into areas working under a different scheme. The computation happens according to local rules only.

6.2.1 Genotype and Phenotype

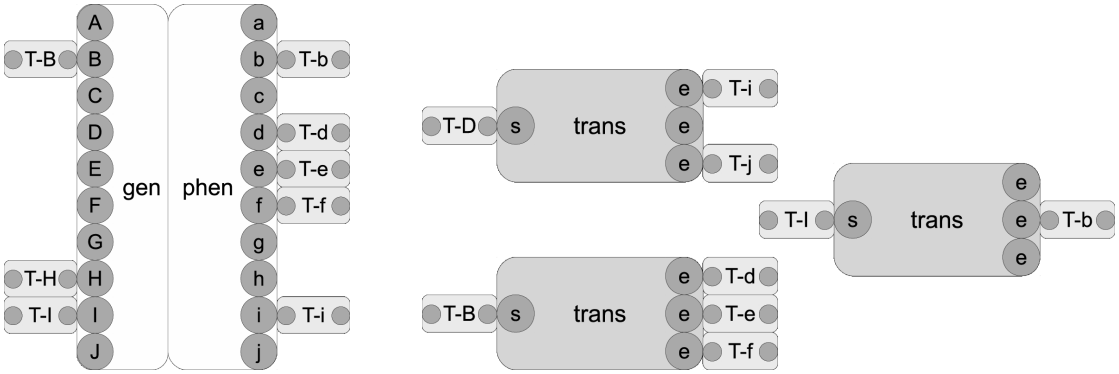
Each possible solution $C_i \in \mathcal{P}(\mathcal{F})$ to the exact set cover problem problem is represented by a *genotype* as well as its consequent *phenotype* in this approach. These two components are realized by a molecule dimer `gen - phen`. The genotype molecule `gen` has exactly one docking site for each possible subset $S_i \in \mathcal{F}$. "Token molecules" `T`

can attach to each binding site of `gen`, representing the inclusion of a certain subset S_i, when attached. The phenotype molecule `phen` on the other site of the dimer has exactly one docking site for each element $x_i \in \mathcal{X}$ of the set to be covered. A token `T` that is attached to a `phen` molecule represents a covered element x_i, as displayed in Figure 6.2a.

The genotype-phenotype mapping, i.e. the assignment stating which elements $\{x_1, ..., x_k\}$ belong to which subset S_i, is realized through *artificial transfer factors*. These transfer factors follow the example of biological transfer-RNAs (tRNAs) [Qui76], which facilitate the mapping between a triplet of DNA bases to a specific amino acid. In our approach, there is exactly one species of transfer molecules `trans` for each subset S_i. It is loaded with exactly one *subset-token* molecule that corresponds to the subset $S_i \in \mathcal{F}$ and a variable number of *element-token* molecules, each corresponding to one of the elements $x_i \in \mathcal{X}$. Some examples for transfer molecules are displayed in Figure 6.2b. This design has the advantage that the instance of the exact set cover problem is independent from the set of rules. We are even able to dynamically exchange the problem instance, while the "algorithm" is working in a reactor by feeding a different set of transfer molecules into the reaction vessel.

Each *subset-* and *element-token* molecule `T` is of a given sub-type, similar to the codon region of a transfer RNA. Hence we can allow a token to dock only to the binding sites that fit its sub-type. In total, we need a number of $|\mathcal{X}| + |\mathcal{F}|$ different types of tokens to allow for the specific binding of tokens to the distinct sites of the genotype and phenotype molecules.

A transfer molecule can dock to any fitting free site of the `gen` - `phen` dimer with its attached tokens, if they are of the correct type. If and only if a transfer molecule can dock all of its tokens into the docking sites of the `gen` - `phen` solution dimer, the transfer molecule can dissociate from its tokens, leaving them with the solution dimer. If there is already a token blocking one of the necessary sites of the `gen` - `phen` dimer, the complete transfer molecule with all of its tokens will dissociate after some time. Thus we ensure, that the tokens at the `gen` side corresponding to the selected subsets S_i are correctly mapped to the tokens on the `phen` side, corresponding to



(a) An arbitrary solution candidate represented by a `gen` - `phen` dimer with attached subset- and element-tokens.

(b) Exemplary transfer factors. The docking site `s` is loaded with a subset-token and the sites `e` are loaded with element-tokens.

Figure 6.2: Genotype - Phenotype mapping: molecules representing the genotype `gen` and the phenotype `phen` of an individual in the evolution. The mapping (a) between subsets and elements is realized by various transfer factors (b) which are specific to the problem instance displayed in Figure 6.1b. For this problem instance, a maximum of three element tokens is sufficient. Each token molecule is marked with the name of the binding sites that it will be able to bind to.

the elements x_i. In this way we cannot attach subsets to a solution that lead to a duplicate selection of an element x_i. On the other hand, when all the elements from $\mathcal{X}$ are present in a solution, we have also selected an appropriate set of subsets $\{S_a, ..., S_k\}$.

6.2.2 Evaluation and Inheritance

When implementing the system according to the design given in the last section, we obtain a molecular random search system that will automatically avoid choosing combinations of sets with overlapping elements. Nonetheless, when there is a large number of non-overlapping subset attachment pathways that lead to imperfect solutions, the chances for producing a correct solution to the problem are small. And even when one molecule in the reactor found the perfect solution, it would be hard to identify it among all the other wrong solutions. Consequently we add a selection

and reproduction operator here, which will help to cover a larger search space and to amplify good solutions.

Both functions, selection and reproduction, are realized by a single molecule `Copier` which goes through different internal phases. In the beginning, it attaches to one `gen - phen` dimer, then to an arbitrary other possible solution dimer as shown in Figure 6.3. When it has docked to two possible solution candidates, it "decides" which of them to discard and which to take as a template for reproduction.

This decision is made by a small number of $|\mathcal{X}|$ reaction rules: for each element $x_i \in \mathcal{X}$, a rule is defined that is applicable, when there is an element token (e.g. `T-e` in Figure 6.3) that is only present on one of the docked solution instances.

The reaction then marks the solution with the lacking element token as *dominated*, the other one as *template*. The more elements are present in one solution but missing in another, the higher the probability for the better solution to be marked as *template*. Still it is a stochastic process, so it is possible that a good solution A covering many elements can be marked as *dominated* in comparison with a competing solution B owning few elements.

Once one of the solution candidates is marked as *dominated*, it will reject all its tokens, effectively deleting this solution and remove the *dominated* marking. Now two alternative events can happen: Either the cleared solution can dissociate from the `Copier` to recruit new transfer molecules and thus generate a new solution candidate randomly. But until the dissociation happens, new transfer molecules can still dock to the cleared solution candidate. Here, the docked `Copier` molecule prevents transfer molecules to attach subset tokens to the cleared molecule, if its other *template* solutions does not show this token. The longer the cleared solution remains at the `Copier`, the more exact will the *template* solution be reproduced at the cleared solution. Consequently, by modifying the dissociation rate, we can alter the effective mutation rate in this evolutionary algorithm. Nonetheless, a mutation can only mean to leave out a subset-token in the overwritten molecule, not to induce a new subset-token. But after the new solution candidate dissociates from the `Copier`, it can indeed be attached to a different subset-token.

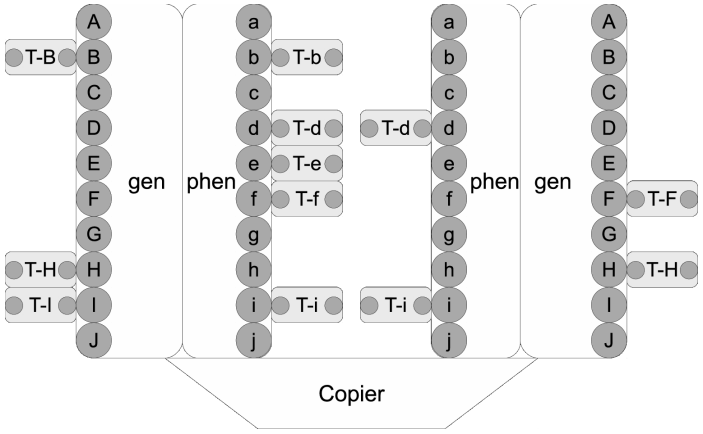
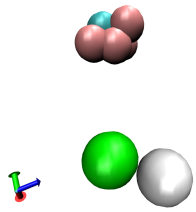


Figure 6.3: The `Copier` molecule realizes both, the selection and reproduction process. The left solution candidate is fitter than the right one, because it has more element-tokens attached to its phenotype side. As a result, the right candidate will be erased and overwritten by the `Copier` molecule.

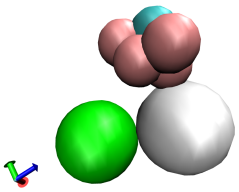
6.2.3 Differences to standard evolutionary algorithms

In comparison to traditional evolutionary algorithms as they are implemented on sequential computers, our approach is asynchronous to begin with, as described by Banzhaf, 1990 [Ban90] or in embodied evolution scenarios [Wat02]. All the components of the evolution system work continuously in the whole system, without separating the system into discrete generations as it is usually done. The evaluation of the fitness is also different, in that no global ranking is done and that no real valued score is calculated. Instead, the fitness of two solution candidates is compared locally and stochastically. Recombination is not implemented in our approach but might be realized by coupling multiple Copier molecules.

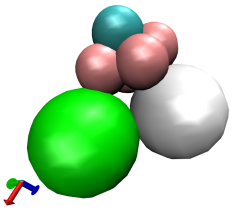
The most striking difference to traditional evolutionary algorithms is probably that we do not switch the "environment" between the execution of the evolutionary algorithm and the evaluation of the fitness function as it usually done, when the fitness evaluation appears as a black box in the algorithm. Here instead, all aspects of the evolutionary algorithm: the mutation operators, the reproduction and the selection are described in the rule-based chemistry language BNGL [Bli04, Hla06].



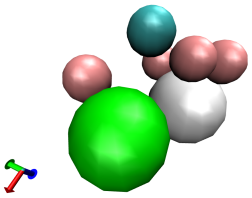
(a) An empty `gen - phen` dimer as solution candidate and a loaded transfer molecule.



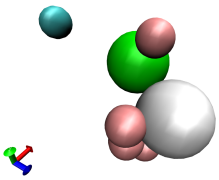
(b) The transfer factor has docked to the solution candidate with a single token.



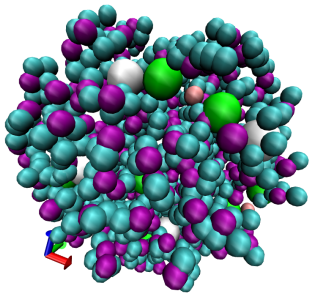
(c) All the transfer factor's tokens are bound to the solution candidate.



(d) The transfer factor jettisons its tokens...



(e) ...and diffuses away from the modified solution candidate.



(f) A snapshot of the whole reactor.

Figure 6.4: Snapshots of the transfer factors while delivering tokens to a solution candidate (a-e) and a visualization of the SRSim simulation of the entire reactor (f).

6.2.4 Simulation case study

To test the system with ten subsets, which was presented in Figure 6.1b, we implemented the asynchronous, local evolutionary algorithm in rule-based chemistry using the BioNetGen Language (BNGL) [Bli04, Hla06] and simulated it with SRSim, our simulation system for spatial and rule-based chemistries [Gru10]. Some snapshots of a docking transfer factor and the general simulation process are shown in Figure 6.4.

Though the size $|\mathcal{F}| = |\mathcal{X}| = 10$ of the problem instance might seem very small, nonetheless, a fully enumerative approach would still need to produce $2^{10} = 1024$ different tentative solutions and thus an even larger number of molecules would be necessary to stochastically cover a large fraction of them. Instead, the toy system we

investigated comprised 10 `gen` - `phen` dimers, three `Copier` molecules and 30 transfer factors for each subset $S_i \in \mathcal{F}$. To test different mutation rates, we varied the dissociation rate k_{off} of `gen` - `phen` dimers from the Copier molecules. We considered the three cases of $k_{\text{off}} = 10^1, 10^{-4}, 10^{-6}$ and the control case of no selection at all. In addition to the reaction system described before, we added rules to "recycle" used transfer factors. Alternatively, an inflow of loaded transfer factors and an outflow of used `trans` molecules might be used.

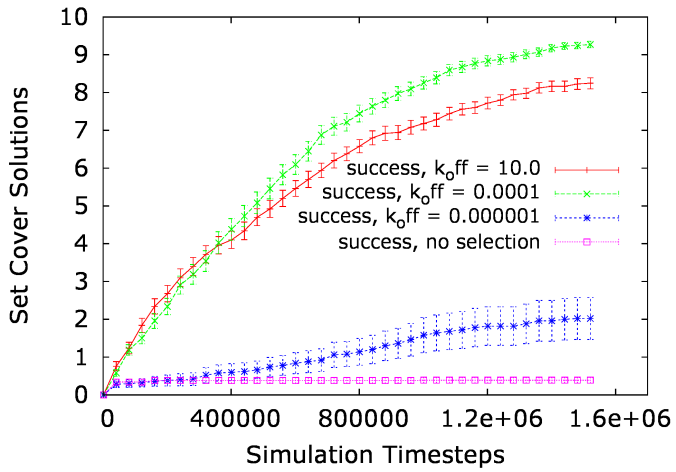


Figure 6.5: Development of the solution candidates in the optimizing molecular system over time. Averaged (n=50) number of generated perfect solutions. Error bars indicate the standard error of the mean. When omitting the copier molecules (box points, purple), we obtain a perfect solution in a fraction of the simulations close to the theoretical 2.56%, corresponding to a perfect solution in about every fourth simulation run. With the highest mutation and dissociation rate $k_{\text{off}} = 10^1$ instead, we obtain a high percentage of perfect solutions in the system quickly. This high mutation rate leads to the situation, that effectively no information is passed from a "parent" molecule to its offspring. A cleared solution candidate will dissociate very fast and generate a new potential solution from scratch (plus points, red). With a lower dissociation rate $k_{\text{off}} = 10^{-4}$, the cleared solution candidates will stay with the `Copier` molecule for some time, inheriting some or all of the template solutions features. This k_{off} rate leads to the fastest convergence in our simulations (x points, green). The lowest dissociation rate delays the whole process of copying a solution very long and leads to a slow convergence (star points, blue).

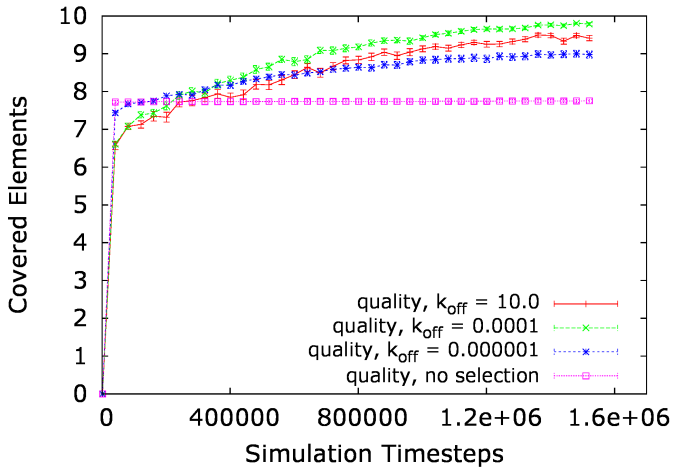


Figure 6.6: Development of the solution candidates in the optimizing molecular system over time. Averaged (n=50) quality of the solutions, in the form of the number of elements from X that are covered by the current solution. Error bars indicate the standard error of the mean. While Figure 6.5 showed the number of perfect solutions, the quality of the found solution candidates is plotted here. Although the number of perfect solutions is initially very low as seen in Figure 6.5, the average quality is quite high in all scenarios from the beginning at about 80%.

To average the results, we simulated 50 instances for each condition of the system for $1.5 \cdot 10^6$ time steps. Please note that one time step is not identical to one generation in the evolutionary algorithms sense in our simulation. Most simulation time steps will only update particle positions from diffusive movement, some will also incorporate chemical reactions. Also, since the algorithm works asynchronously, we cannot distinguish generations. Instead, we measured the number of dissociations of the `gen - phen` dimers from the `Copier` molecules over the whole simulation time to be about 140 for $k_{\text{off}} = 10^1$, 110 for $k_{\text{off}} = 10^{-4}$ and 10 for $k_{\text{off}} = 10^{-6}$ in average. Since we are not interested in the optimization performance per generation but per time, the plots in Figure 6.5 and 6.6 are not normalized in respect to the generations.

In Figure 6.5 we plot the number of perfect solutions present in a simulation averaged over 50 simulation runs with different dissociation rates, showing that a dissociation and mutation rate of $k_{\text{off}} = 10^{-4}$ led to the fastest increase of optimal solutions. As we described in Section 6.1, the problem instance with ten subsets from Figure 6.1(b) is still relatively easy to solve, meaning that about three percent of any randomly generated individual will be a perfect solution. Also, while evolution with an intermediate mutation rate (x points) produces the best results here, it is tightly followed by a system with an extremely high mutation rate and thus practically no inheritance (plus points). For more difficult instances of the problem, we expect the evolution with intermediate mutation rates to perform better in a more distinguished way. Considering the implicitly defined objective function as the number of covered elements, Figure 6.6 shows a very fast initial rise of the average population fitness, followed by a slower increase in average fitness.

6.3 Discussion

In this chapter, we introduced an evolutionary computation system that is implemented entirely as a rule-based reaction network. We demonstrated its heuristic problem-solving capacity with an exemplary instance of the exact set cover problem. Different ranges of dissociation parameters were analyzed for their effect on the

performance of the evolutionary algorithm.

So far, molecular computing methods were typically aiming at solving difficult computing problems by exploiting the vast amount of molecules in reaction vessels to enumerate all possible solutions in cases of high combinatorial complexity. Hence, the exponential effort of sequential algorithms in runtime is shifted to an exponential effort in material. While sequential algorithms could solve larger problem instances "in principle" in many years runtime, molecular computing setups could solve these problems "in principle" when using earth-sized amounts of DNA, protein or other molecular computing substrate [GN11].

In contrast to that, we follow the path of heuristics that helps to generate relatively good, though not necessarily perfect, results for a wide range of problems. Still, in our example example, the propagation of the best solution happens slower than the generation of many solutions of quite good fitness. Our focus here is not on solving NP-hard problems, but to generate relatively good solutions to non-trivial optimization tasks while staying in the chemical reaction medium. This provides the advantage of having control structures for bioreactors or other artificial chemical systems that could be implemented directly inside the reaction vessel instead of installing sensors, connecting silicon-based computers and feeding controlling actions back through actuators.

Nonetheless, it will probably be difficult to find or engineer molecules that behave in such a way as proposed here. But it might be an advantage to focus on the object-oriented view upon the molecules designed here to distinguish between different aspects of such dynamical chemical systems. Instead of searching for molecules that fulfill a hardly comprehensible large list of constraints, the different aspects and dependencies that are necessary for each molecule type are here explicitly mentioned in a rule-based description.

Chapter 7

Programmed Self-Assembly in Biology

Many of the established unconventional computing approaches like Membrane Computing [Pău06], Molecular- or DNA Computing [Adl94, Win98, Con98, Leh02, Rot04, Rot06, See06, Yin08, Had10, Rub14, Xia14] utilize self-assembly processes. Considering any self-assembled structure, there will typically be a range of final structures with these systems. But the variance of final molecular structures is small in comparison, when compared to a hypothetical random assembly of similar building blocks. Hence, the self-assembly of an artificial or biological structure [Zau05b] should be considered as controlled processes, similar to the execution of an algorithm, where the results of the computations are either the structures themselves or the functions they serve. Following the notation from Section 1.3, the human kinetochore complex, for example, fulfills among others, the function f_k of distributing the mechanical forces between the spindle apparatus and the chromosomes [Per11, Ibr13]. Hence, roughly speaking, the implementation p_{f_k} of this computation is described as "generate a structure s_k which realizes function f_k".

$$p_{f_k} : \emptyset \longrightarrow s_k$$

Similar to a computer program that could run on very different processor architectures, the actual task is abstracted from the molecular implementation of the kinetochore, i.e. the same task f_k could have been realized by a completely different set of biomolecules [Gör13].

Observing this molecular system as a computing process, where is the program? Similar to Zauner's anti-universal Turing Machine [Zau96], the systems is useful only for one particular computation and the program is encoded directly in the specification p_{f_k} of the "machine". Hence, the program can be encoded in the design of the building blocks themselves, in the quantity and timing of their injection into the reaction container, in the medium composition and temperature or in the spatial structure of the reaction vessel or in its fluid dynamics and stirring properties [Con98, Leh02, Rot04, Yin08, Had10, Rub14]. An example of an evolved self-assembly system for droplet computers (cf. Chapter 2) is discussed in Section 3.2, more details of which are presented in Refs. [Die12a, Die12b].

While the perspective of unconventional computing is obvious for artificial or engineered self-assembling systems, also for many existing biological systems it might be beneficial to understand the systems as computing processes, albeit designed by evolution. In return, this perspective might as well be helpful for learning new design principles for artificial systems. This is exemplified in the following summaries of the works on the human kinetochore multi-protein complex [Tsc13, Ibr13, Gör13] and on the human interphase chromatin organization [Geh12]. In both examples, the approach is shared to build a more accessible replica p'_f of the original biological system p_f in computer simulation as visualized in Figure 7.1.

7.1 The Human Kinetochore Self-Assembly

As outlined in the introduction to Section 7, the kinetochore assembly can be observed from the perspective of a computation p_{f_k} generating a structure s_k that, in turn,

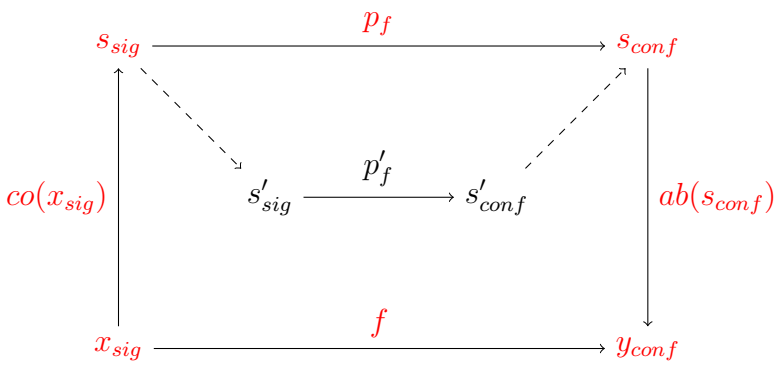


Figure 7.1: Interpretation of biological self-assembling conformations as a computing systems. Signals x_{sig} that reach the structure via signaling molecules s_{sig} somehow lead to the modification of the assembly process or the organization of the structure s_{conf}, eventually resulting in an altered behavior or function y_{conf} in the cell. While the actual biological purpose f of a system as well as the specifics of the real biological implementation of p_f are sometimes hardly accessible (red), a replica in simulation p'_f of the system allows for cheaper and more transparent experiments and hypothesis testing (black).

fulfills the many functions f_k of the kinetochore.

$$p_{f_k} : \emptyset \longrightarrow s_k$$

From the computational perspective, much is known about the abstract result of the computation, i.e. about the effect of the kinetochore which facilitates the proper segregation of the daughter chromatids during mitosis [San09, Cay10, Cay12]. These functions lead to the proper segregation of the chromatids into the daughter cells and include for example various logical checkpoints [Cay12], arresting the cell cycle in case of problems. Furthermore the kinetochore structure also allows for the distribution of physical force to the DNA strands so that they are not damaged when being dragged to the daughter cells [Mus07]. What is mostly unknown are the "implementation details", i.e. the exact set of molecules involved, their interactions and their particular roles in the process. Hence, the task here can be seen as a process of reverse engineering where the aim is to understand the syntax and semantics of the "molecular programming language" and the mechanisms that lead to the final

structure s_k and its functions. The computing substrate here is the cell physiology, the physics and chemistry of the cell. We might say that the primitives that the biological kinetochore is "programmed" with are the amino acid and RNA sequences of the involved proteins. Additionally, the intracellular regulation mechanisms generate a precise and complex timing scheme of the availability of particular biomolecules at particular times. This biological program p_{f_k} cannot simply be read out for interpretation; rather, complex laboratory experiments are required to characterize each aspect of the program step-by-step.

The approach that was taken in the papers [Tsc13, Ibr13, Gör13] was the effort to build an alternative implementation p'_{f_k} of the program on a different computing substrate, i.e., in the coarse grained molecular dynamics simulator SRSim [Gru10]. By using a similar set of "building blocks" that is constrained by the knowledge about the original system, a set of alternative implementations should be generated that show high similarities to the biological system and thus allow for a deeper understanding of the original system. A snapshot of a single simulated kinetochore structure is shown in Figure 7.2 while Figure 7.3 shows the similarity tree of an ensemble of computed kinetochore structures. In Figure 7.4, the interacting proteins are shown with their potential bonds colored in blue. Red lines of variable thickness correspond to the statistical dependency between different bonds. They indicate that found correlations between the realization of the H3/CenpW bond and the CenpB/CenpA bond. More details and further results of the simulations are given the manuscripts [Tsc13, Ibr13, Gör13].

7.2 Yeast Interphase Chromatin Conformation

In the previous section, the study on the peculiarities of the implementation of the molecular algorithm that leads to the partially understood effects of the kinetochore was outlined. Considering the interphase chromatin organization [Cre01, Ber08] of yeast cells, less is known about either the details of the implementation, the abstractions, or the input-output signals [HS06].

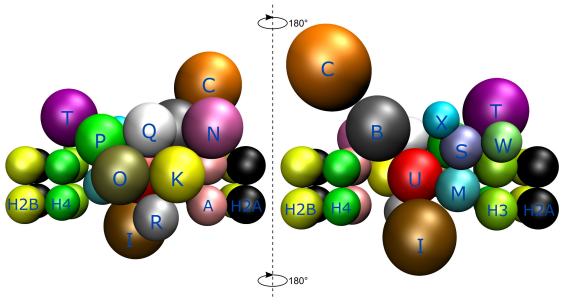


Figure 7.2: Snapshot of a simulation of the inner kinetochore model from a typical simulation run with all inner kinetochore proteins responsible for building the bridge between two nucleosomes. Kinetochore proteins are labeled in a short form (e.g. K for CenpK). The same structure is rotated 180° from left to right. Rendering by Sergej Tschernyschkow (2013), Jena. [Tsc13]

While chromosomes are highly condensed during mitosis, the genetic material is expanded throughout the nucleus during interphase, ready for transcription and regulation processes to access the information. In this state, the organization of the different genomic regions throughout the nucleus is of importance [Mis05, Spr05, Fra07, SB12]. The chromatin layout is controlled and not random; nonetheless, there is still a lot of variance in the positions of the individual pieces of DNA. Currently it is still hard to determine the exact distribution of positions for each piece of DNA in experiments. Also the exact function, the dynamics, and the precise effects of the chromatin layout are still being researched [VB12, dW12].

In the work outlined here, an alternative "molecular algorithm" was constructed in simulation that should generate an artificial output signal similar to the original biological one, i.e., the interphase chromatin organization of yeast *S. cerevisiae*. In this light, the nucleus and the interphase chromatin conformation can be seen as an information transforming system, i.e., as computation as well: input in the form of signaling molecules, mediated by proteins and DNA fiber dynamics, affects the chromatin organization and thus leads to an output in the form of different transcriptomes. This means that the nucleus implements a mapping from signaling molecules to different gene expression patterns. Comparing this potential computation of the

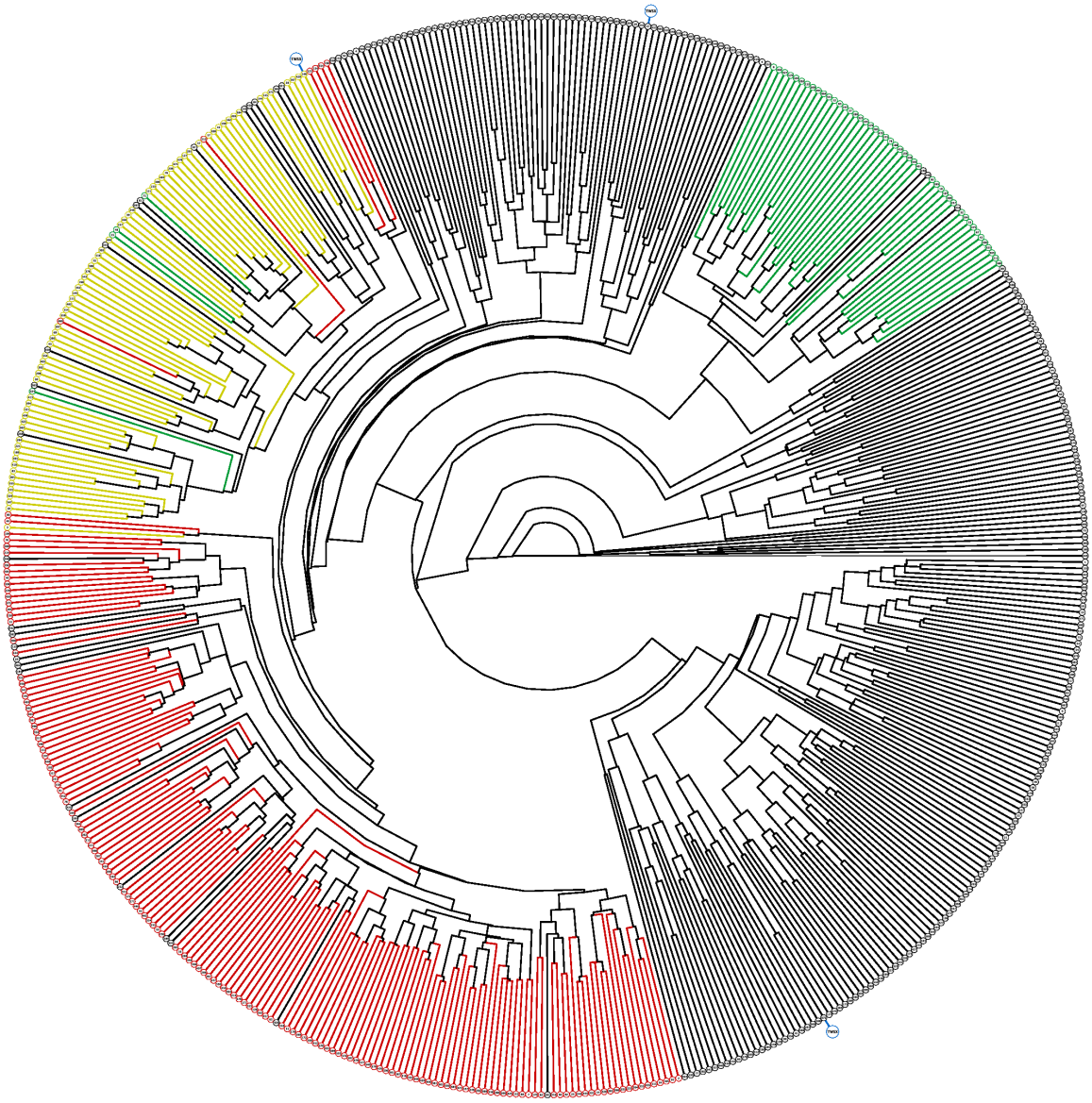


Figure 7.3: Structural comparison visualized as similarity tree. Each leaf represents one computed structure and the distance between two leaves indicates the dissimilarity between two computed structures. The distances between 537 structures are visualized in this tree. We highlighted three particular model variations, where a specific subset of the possible bonds is realized: 154 structures missing CenpA/CenpB in red, 71 missing CenpB/CenpW in yellow and 50 missing H3/CenpW in green. The tetramer CenpT/W/S/X, marked as external blue cycles, was found 3 times. The tight structure clusters indicate that a similar set of formed bonds also leads to a high similarity in the resulting spatial structures. Rendering by Sergej Tschernyschkow (2013), Jena. [Tsc13]

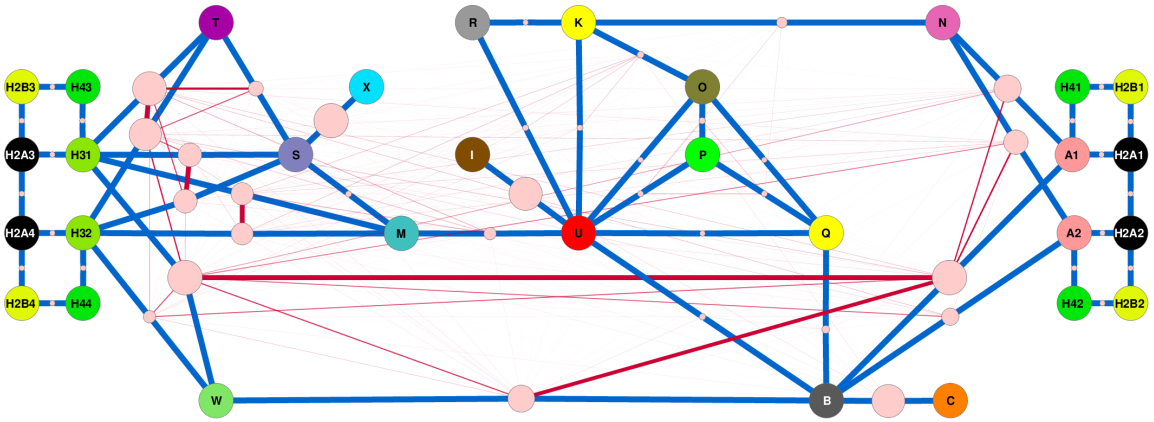


Figure 7.4: This graph displays the protein-protein bonds of the inner kinetochore as well as the mutual information between these bonds. Because a bond can or cannot be realized in an instance of the experiment, we can measure the entropy of a bond's presence as well as the mutual information between the presence of different bonds. Thick blue edges indicate protein-protein interactions, running over special nodes that represent each bond as separate pink node. Red colored edges are showing mutual information between two bonds with the thickness indicating the level of the mutual information. The thickest lines have mutual information values of about 0.67, while the cutoff for drawing lines was set to 0.05. The bond nodes are drawn larger for higher entropy. We found high mutual information between the CenpA/CenpB and the H3/CenpW bonds and moderate mutual information between CenpA/CenpB and CenpW/CenpB while lower mutual information between the CenpW/CenpB and the CenpW/H3 bonds.

nucleus to the diagram in Figure 1.4, a function f_{nuc} describes the finegrained transcriptional control in the nucleus. Some mostly unknown signals x_{nuc} can affect the chromatin structure and thus the final transcription pattern y_{nuc}. This computation is implemented by an unknown "program" $p_{f_{nuc}}$ constituted by the interactions between a large number of intracellular molecules and DNA strands. To understand, manipulate and utilize this regulation process, it would be helpful to understand $p_{f_{nuc}}$ itself, but also the final structure s_{nuc} as well as its meaning y_{nuc}.

Only very few aspects of the computing system $p_{f_{nuc}}$ are known: some of the participating molecules and some of their interactions [Got96, Bys04, Bys05] as well as some characteristics of the polymer dynamics of the various types of DNA fibers in different compaction states [Heu01, Bys04]. In particular, using genomic conformation capture (GCC) [Rod09], one can experimentally map some of the genomic regions that are found in close spatial proximity. What still needs to be elucidated are the abstraction functions, which external information x_{nuc} influences the conformational changes, how they alter the conformation and which mechanisms lead to the final conformation s_{nuc} and finally how this conformation is translated into a particular transcription pattern y_{nuc} by the "molecular program" p_f.

In [Geh12], we worked towards reconstructing the "output signal" s_{nuc}, i.e., the geometry of the DNA fibers in the biological nucleus, based on the GCC data as well as less specific information as for example the rough location of the centromeres and telomeres. In analogy to Figure 7.1, because of the hardly accessible real implementation p_f, we built an analogue system p'_f while relating the probabilistically sampled resulting chromatin conformations s'_{nuc} to the known facts about the real system. The complete yeast genome was simulated in a polymer simulation using the LAMMPS molecular dynamics simulation software [Pli95] with DNA segments comprising between 1300 and 3900 base pairs. A snapshot of the reconstructed output signal can be found in Figure 7.5, showing the generated probability distribution of the DNA segments that will hopefully lead to a better understanding of the processes that generate the spatial localization, i.e., the cellular algorithm p_f.

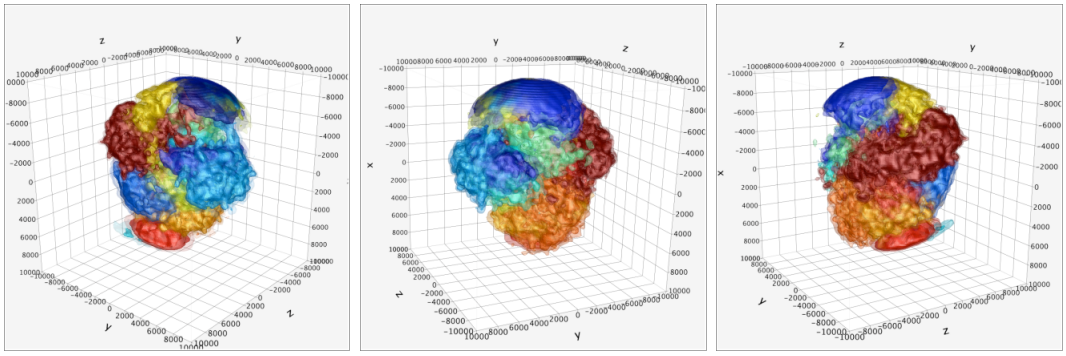


Figure 7.5: Color reconstruction of chromosome territories in exponentially growing respiro-fermenting yeast cells seen from different angles. Chromosomes are shown at a minimum 85% contour density. Rendering by Lutz Gehlen and Jörg Langowski (2012), Heidelberg. [Geh12]

Chapter 8

Conclusions

In this thesis, research towards unconventional programming methods, predominantly for unconventional computing systems, is documented. As a first step, a model system for unconventional computations is presented in combination with suitable simulation models in Chapter 2. Different formalisms for modeling the Belousov-Zhabotinsky reaction and Belousov-Zhabotinsky droplets are discussed. For the network scale of ten to 1000 droplets, the introduced event-based model offers fast simulation speeds, reproduces laboratory results and accounts for stochastic effects that can strongly influence practical computations *in vitro*.

Then programming approaches based on artificial evolution (cf. Chapter 3) are explored in the context of the model system. It is shown that evolution can be used as an indicator for task difficulty and that co-evolution of signals with computing systems allows finding suitable symbol encodings for computing networks. Here, the co-evolved symbols, as compared to naive rate-coding symbols, are shown to simplify the task of finding a droplet network structure for Boolean gates.

A novel method for understanding and analyzing the mechanisms of computation as *information flows* is introduced in Chapter 4 in the context of the model system. The investigation of information flows differs from conventional computer science methods, where the execution of a computer program is typically studied from the perspective of runtime and memory complexity. This understanding allows targeted

modifications of problematic positions where the information flow is impeded, which is also demonstrated. Furthermore, information theoretic measures can be used as fitness function for evolutionary computation.

Self-organization principles are presented in Chapter 5 to be suitable for finding symbol encodings by using the system's own dynamics, potentially leading to advantageous solutions as well as to an efficient fitness evaluation. The resulting self-organizing network structures, *tautological loops*, are formalized, partially enumerated and an information theoretic objective function for the observed signals is developed. Evolutionary computation using the self-organization of symbol encodings is demonstrated, resulting in two droplet NAND-gate implementations, one of which is shown to work in a half-adder setup.

Embodied evolution experiments are described in Chapter 6. They show that evolutionary processes can not only be implemented with classical RNA and DNA. Thus, evolutionary processes could also be applicable for continuously optimizing biochemical production processes *in vitro*. The domain-oriented view of rule-based modelling can here help designing molecules with appropriate properties. An *in-silico* implementation of an evolutionary computation system using rule-based chemistry only is tested using an instance of the exact set cover problem.

Additionally, an excursion is taken into the information processing perspective of biological, self-assembling systems in Chapter 7.

Further Research Options

In this thesis, it was a common theme to search for implementations p_f and modalities for the implementation of an already known function f. In contrast, especially in the unconventional computing [Ada01, Ada06, Mat07, Teu08, Ste12] community, it seems not uncommon to start from an information processing system with unknown properties and signal processing behavior that is then further characterized. Although this approach was neglected in this thesis, also for this direction of research the information flow analysis techniques from Chapter 4 might be helpful in narrowing down the range of potential target functions.

Also evolutionary algorithms were frequently used in this work to generate problem solutions of various forms. Typically this implies mimicking the process of biological evolution, including a population of solution candidates, random mutations, fitness based selection and recombination to generate the next generation. Nonetheless, in artificial evolution, it might be possible to increase the search speed by working around evolutionary bottlenecks, also in ways which would not be possible in biological evolution: For example, automatically defined functions [Koz96] can be used to modularize genetic programs, Differential Evolution (DE) [Das11], the Covariance Matrix Adapted Evolution Strategy (CMA-ES) [Han01] and many other schemes can be used to modify the mutation operators, potentially leading to a more efficient optimization process.

Another method that might allow bypassing evolutionary bottlenecks with specialized domain knowledge would be to make use of known intermediate results of the desired computation. Thus the optimization of a complex system could be simplified down to break the system into smaller and more independent parts. Similarly, also the information flow measures from Chapter 4, which were used as a fitness function in Chapter 5, might be helpful for improving the optimization process. In further works, the calculation of information flow measures might be revised for reducing discretization and sampling errors [Rou99, Kra04]. But these measures might not only be used as an analysis tool and fitness function, but also to modify the mutation operator in artificial evolution approaches: When the desired information flow is known an advance, mutations could be focused in a region where the information flow is obstructed or where the information recombination between different sources does not work properly.

While this approach would not resemble an evolutionary process with random mutations any more, the resulting process might still constitute a powerful optimization system worth exploring. But it should be noted that such a technique would be based on partially decoupling different parts of the evolved system, i.e., by splitting the information flows or by defining regions in which particular information is supposed to be mixed. Although this decoupling might lead to faster results, the “blindness” of

normal evolutionary algorithms might be a feature that sets them apart from design processes using common engineering principles. Hence the search space is narrowed down, potentially missing some good solution candidates. But it might also be possible to control how broadly the information of an input or an intermediate result spreads out through the network or how decentralized the information is represented. For example, we might force the symbols in the system to be represented using multiple nodes of the network, such that a single node does not bear the mutual information while a combination of some nodes would.

When following, as proposed in the last paragraphs, the approach of controlled information flows, it is still questionable how to derive information flow graphs that would be suitable for a given problem. One path toward a solution here might be offered by mutual information decomposition [Wil10a, Wil10b], which can show how much information about the output value of a computation is uniquely, redundantly and synergistically found in each combination of the input signals. Redundant information is equally present in all considered input channel combinations. Unique information can only be found in one particular input channel combination. Synergistic information only becomes available, when combining different input channels.

i_0	i_1	i_2	o
0	0	0	1
1	0	0	0
0	1	0	1
1	1	0	0
0	0	1	0
1	0	1	1
0	1	1	0
1	1	1	0

Table 8.1: Exemplary Boolean function f_{37} with three input values i_0, i_1, i_2 and one output value o for information decomposition.

The truth-table of an exemplary Boolean function f_{37} is displayed in Table 8.1. Assuming all input combinations are used with the same probability, the entropy of the output channel is $\sim$0.95 bits. In the corresponding mutual information decomposition analysis in Figure 8.1, it can be seen that each of the input channels shares ca. 0.049

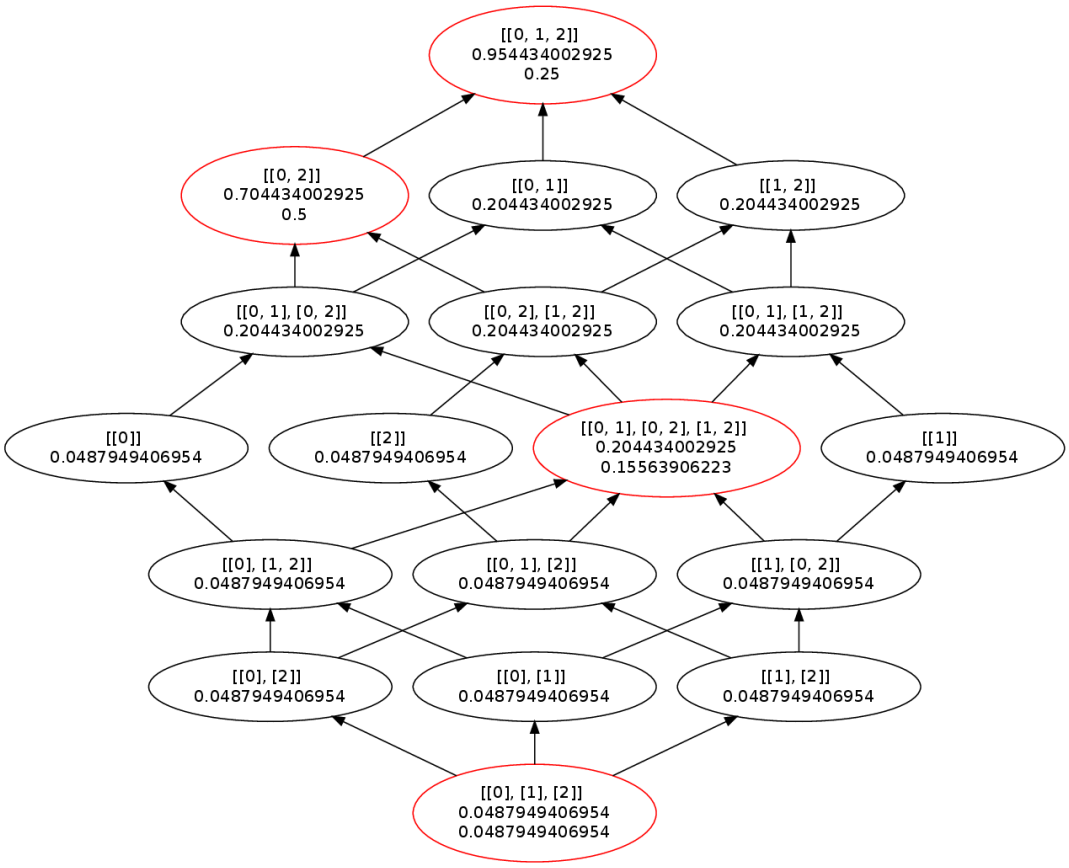


Figure 8.1: Mutual information decomposition [Wil10a, Wil10b] between three input values and one output value for the Boolean function f_{37} from Table 8.1. In this kind of diagram, combinations of two or more input signals for a computation are analyzed for their redundant, unique and synergistic contribution to the final result. Therefore, each node in the graph represents the redundant information between combinations of input signals: Exemplarily, $[[v, w], [x, y]]$ denotes the redundancy between the combined inputs $[v, w]$ and $[x, y]$. The redundant information structure is ordered in a lattice, such that the higher nodes that are pointed to by an arrow carry at least as much information as the nodes below. Nodes where a combination of input channels adds more information than the nodes below are colored in red and show a third line with a value indicating the additional redundant information of this node. In this case, from bottom to top, the graph can be interpreted followingly: All three input channels individually share $\sim$0.049 bits of information. These $\sim$0.049 bits are present in all input channels redundantly. Either combination of two of the channels adds additional $\sim$0.156 bit of information, but this information is only available when observing pairs of channels together - thus it is called synergistic information. The combination of inputs $[0, 2]$ has 0.5 bit of extra information next to the information that is already shared with the individual inputs and the combinations $[0, 1]$ and $[1, 2]$. When considering all combinations of two channels, the combination $[0, 2]$ has 0.5 bit of unique information, that can not be found in either $[0, 1]$ or $[1, 2]$. When combining all three channels, an additional 0.25 bit of information on the output of the computation are gained as synergistic information.

bits of mutual information about the output: Having a zero in either of the input channels makes an output of 1 slightly more likely. The red node in the central row indicates that either combination of two input channels adds another ∼0.16 bit of information. Then the combination of inputs i_0 and i_2 adds another 0.5 bits and finally the combination of all three inputs add the final 0.25 bits of information about the output value. This might indicate that the best way of computing the result from the three input channels is to first combine inputs i_0 and i_2 and to merge the result with input i_1. Because of the ∼0.16 bits that are shared by all input combinations, the results of the combinations (i_0,i_1) and (i_1,i_2) might be used for additional checking. Also, the decomposition reveals that the combination of the input channels i_0 and i_2 could maximally result in ∼0.7 bits of mutual information with the output and thus can be used as an intermediate goal to verify that the combination was successful, before it is attempted to include i_1 for the final result.

In conclusion, information theory, self-oranization and smart objective functions for optimization algorithms seem like helpful guidelines for understanding unconventional computing systems and programs in general. They can thus be understood as a particular perspective, as filters that allow a better understanding and design of the systems under particular angles. Hopefully, more unconventional programming paradigms, more filters and thus more perspectives will in future be established that allow transforming complex and promising systems that are not directly accessible for engineering principles into a more observable and more human-accessible form.

Bibliography

[Abe00] Abelson, H.; Allen, D.; Coore, D.; Hanson, C.; Homsy, G.; Knight Jr, T. F.; Nagpal, R.; Rauch, E.; Sussman, G. J.; Weiss, R. Amorphous computing. *Commun. Acm.*, 43(5):74–82, 2000.

[Ada01] Adamatzky, A. *Computing in nonlinear media and automata collectives.* IOP Publishing Ltd., Bristol, UK, 2001.

[Ada02a] Adamatzky, A. *Collision-based computing.* Springer Verlag, 2002.

[Ada02b] Adamatzky, A.; Costello, B. Experimental logical gates in a reaction-diffusion medium: The xor gate and beyond. *Physical Review E*, 66(4):046112, 2002.

[Ada04] Adamatzky, A. Collision-based computing in belousov-zhabotinsky medium. *Chaos, Solitons & Fractals*, 21(5):1259–1264, 2004.

[Ada05] Adamatzky, A.; Costello, B. D. L.; Asai, T. *Reaction-diffusion computers.* Elsevier Science Limited, 2005.

[Ada06] Adamatzky, A.; Teuscher, C., editors. *From Utopian to Genuine Unconventional Computers.* Luniver Press, 2006.

[Ada09] Adamatzky, A.; Bull, L. Are complex systems hard to evolve? *Complexity*, 14(6):15–20, 2009.

[Ada10] Adamatzky, A. On excitable beta-skeletons. *Journal of Computational Science*, 1(3):175 – 186, 2010.

[Ada11a] Adamatzky, A.; de Lacy Costello, B.; Bull, L. On polymorphic logical gates in sub-excitable chemical medium. *International Journal of Bifurcation and Chaos*, 21(7):1977–1986, 2011.

[Ada11b] Adamatzky, A.; De Lacy Costello, B.; Bull, L.; Holley, J. Towards arithmetic circuits in sub-excitable chemical media. *Isr. J. Chem.*, 51(1):56–66, 2011.

[Ada11c] Adamatzky, A.; de Lacy Costello, B.; Holley, J.; Gorecki, J.; Bull, L. Vesicle computers: Approximating a voronoi diagram using voronoi automata. *Chaos Solitons and Fractals*, 44:480–489, 2011.

[Ada11d] Adamatzky, A.; Holley, J.; Bull, L.; De Lacy Costello, B. On computing in fine-grained compartmentalised belousov-zhabotinsky medium. *Chaos Solitons and Fractals*, 44:779–790, 2011.

[Adl94] Adleman, L. M. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024, 1994.

[Agh08] Aghdaei, S.; Sandison, M.; Zagnoni, M.; Green, N.; Morgan, H. Formation of artificial lipid bilayers using droplet dielectrophoresis. *Lab Chip*, 8(10):1617–1620, 2008.

[Agl96] Agladze, K.; Aliev, R.; Yamaguchi, T.; Yoshikawa, K. Chemical diode. *J. Phys. Chem.*, 100(33):13895–13897, 1996.

[Ali97] Ali, F.; Menzinger, M. Stirring effects and phase-dependent inhomogeneity in chemical oscillations: The belousov-zhabotinsky reaction in a CSTR. *The Journal of Physical Chemistry A*, 101(12):2304–2309, 1997.

[Alu94] Alur, R.; Dill, D. L. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[Bab89] Babbage, C. *Science and reform: selected works of Charles Babbage*. Cambridge University Press, 1989.

[Ban90] Banzhaf, W. The "molecular" traveling salesman. *Biological Cybernetics*, 64(1):7–14, 1990.

[Ban05] Banâtre, J.-P.; Fradet, P.; Giavitto, J.-L.; Michel, O., editors. *Unconventional Programming Paradigms, International Workshop UPP 2004, Le Mont Saint Michel, France, September 15-17, 2004, Revised Selected and Invited Papers*, volume 3566 of *Lect. Notes Comput. Sc.* Springer-Verlag Berlin Heidelberg, 2005.

[Ban13] Banda, P.; Teuscher, C.; Lakin, M. R. Online learning in a chemical perceptron. *Artificial Life*, 19(2):195–219, 2013.

[Bar91] Barkley, D. A model for fast computer simulation of waves in excitable media. *Physica D: Nonlinear Phenomena*, 49(1-2):61–70, 1991.

[Bar08] Barbieri, M. Biosemiotics: a new understanding of life. *Naturwissenschaften*, 95(7):577–599, 2008.

[Bea05] Beal, J. Programming an amorphous computational medium. In Banâtre, J.-P.; Fradet, P.; Giavitto, J.-L.; Michel, O., editors, *Unconventional Programming Paradigms*, Lect. Notes Comput. Sc., pages 121–136. Springer-Verlag Berlin Heidelberg, 2005.

[Bea11] Beal, J.; Michel, O.; Schultz, U. P. Spatial computing: Distributed systems that take advantage of our geometric world. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 6(2):11, 2011.

[Bee92] Beer, R. D.; Gallagher, J. C. Evolving dynamical neural networks for adaptive behavior. *Adaptive behavior*, 1(1):91–122, 1992.

[Ben82] Bennett, C. H. The thermodynamics of computation—a review. *Int. J. Theor. Phys.*, 21(12):905–940, 1982.

[Ber08] Berger, A. B.; Cabal, G. G.; Fabre, E.; Duong, T.; Buc, H.; Nehrbass, U.; Olivo-Marin, J.-C.; Gadal, O.; Zimmer, C. High-resolution statistical mapping reveals gene territories in live yeast. *Nature methods*, 5(12):1031–1037, 2008.

[Bes10] Bessey, A.; Block, K.; Chelf, B.; Chou, A.; Fulton, B.; Hallem, S.; Henri-Gros, C.; Kamsky, A.; McPeak, S.; Engler, D. A few billion lines of code later: using static analysis to find bugs in the real world. *Commun. Acm.*, 53(2):66–75, 2010.

[Bey02] Beyer, H.-G.; Schwefel, H.-P. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[BL01] Berners-Lee, T.; Hendler, J.; Lassila, O.; others, . The semantic web. *Sci. Am.*, 284(5):28–37, 2001.

[Bli04] Blinov, M. L.; Faeder, J. R.; Goldstein, B.; Hlavacek, W. S. Bionetgen: software for rule-based modeling of signal transduction based on the interactions of molecular domains. *Bioinformatics*, 20(17):3289–3291, 2004.

[Bod00] Bodenschatz, E.; Pesch, W.; Ahlers, G. Recent developments in rayleigh-bénard convection. *Annual review of fluid mechanics*, 32(1):709–778, 2000.

[Bor99] Borst, A.; Theunissen, F. E. Information theory and neural coding. *Nat. Neurosci.*, 2(11):947–957, 1999.

[Bou02] Bourret, R. B.; Stock, A. M. Molecular information processing: lessons from bacterial chemotaxis. *J. Biol. Chem.*, 277(12):9625–9628, 2002.

[Bro04] Brown, E. N.; Kass, R. E.; Mitra, P. P. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nat. Neurosci.*, 7(5):456–461, May 2004.

[Bul13] Bull, L.; Holley, J.; Costello, B. D. L.; Adamatzky, A. Toward Turing's a-type unorganised machines in an unconventional substrate: a dynamic representation in compartmentalised excitable chemical media. In Dodig-Crnkovic, G.; Giovagnoli, R., editors, *Computing Nature*. Springer, 2013.

[Bys04] Bystricky, K.; Heun, P.; Gehlen, L.; Langowski, J.; Gasser, S. M. Long-range compaction and flexibility of interphase chromatin in budding yeast analyzed by high-resolution imaging techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 101(47):16495–16500, 2004.

[Bys05] Bystricky, K.; Laroche, T.; van Houwe, G.; Blaszczyk, M.; Gasser, S. M. Chromosome looping in yeast telomere pairing and coordinated movement reflect anchoring efficiency and territorial organization. *The Journal of cell biology*, 168(3):375–387, 2005.

[Car14] Carman, C. C.; Evans, J. On the epoch of the antikythera mechanism and its eclipse predictor. *Archive for History of Exact Sciences*, pages 1–82, 2014.

[Cas10] Caschera, F.; Gazzola, G.; Bedau, M. A.; Moreno, C. B.; Buchanan, A.; Cawse, J.; Packard, N.; Hanczyc, M. M. Automated discovery of novel drug formulations using predictive iterated high throughput experimentation. *PLoS One*, 5(1):e8546, 2010.

[Caw11] Cawse, J. N.; Gazzola, G.; Packard, N. Efficient discovery and optimization of complex high-throughput experiments. *Catal. Today*, 159(1):55–63, 2011.

[Cay10] Caydasi, A. K.; Ibrahim, B.; Pereira, G. Monitoring spindle orientation: Spindle position checkpoint in charge. *Cell Div*, 5:28, 2010.

[Cay12] Caydasi, A. K.; Lohel, M.; Grünert, G.; Dittrich, P.; Pereira, G.; Ibrahim, B. A dynamical model of the spindle position checkpoint. *Molecular Systems Biology*, 8(1), 2012.

[Che02] Cheon, Y.; Leavens, G. T. A simple and practical approach to unit testing: The JML and JUnit way. In Magnusson, B., editor, *ECOOP 2002 – Object-Oriented Programming*, volume 2374 of *Lect. Notes Comput. Sc.*, pages 231–255. Springer-Verlag Berlin Heidelberg, 2002.

[Cli93] Cliff, D.; Harvey, I.; Husbands, P. Incremental evolution of neural network architectures for adaptive behaviour. In Verleysen, M., editor, *European Symposium on Artificial Neural Networks (ESANN'93)*, pages 39–44, Brussels, 1993. D Facto.

[Con89] Conrad, M. The brain-machine disanalogy. *BioSystems*, 22(3):197–213, 1989.

[Con95a] Conrad, M. The price of programmability. In *The Universal Turing Machine A Half-Century Survey*, pages 261–281. Springer, 1995.

[Con95b] Conrad, M. Scaling of efficiency in programmable and non-programmable systems. *BioSystems*, 35(2):161–166, 1995.

[Con98] Conrad, M.; Zauner, K.-P. DNA as a vehicle for the self-assembly model of computing. *BioSystems*, 45(1):59 – 66, 1998.

[Cop08] Copeland, B. J. The church-turing thesis. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Stanford University, fall 2008 edition, 2008.

[Cou77] Cousot, P.; Cousot, R. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, POPL '77, pages 238–252, New York, NY, USA, 1977. ACM.

[Cre01] Cremer, T.; Cremer, C. Chromosome territories, nuclear architecture and gene regulation in mammalian cells. *Nature reviews genetics*, 2(4):292–301, 2001.

[Cyp93] Cypher, A.; Halbert, D. C. *Watch what I do: programming by demonstration*. MIT press, 1993.

[Das11] Das, S.; Suganthan, P. N. Differential evolution: A survey of the state-of-the-art. *Evolutionary Computation, IEEE Transactions on*, 15(1):4–31, 2011.

[Dau09] Dauwels, J.; Vialatte, F.; Weber, T.; Cichocki, A. On similarity measures for spike trains. In Köppen, M.; Kasabov, N.; Coghill, G., editors, *Advances in Neuro-Information Processing*, volume 5506 of *Lect. Notes Comput. Sc.*, pages 177–185. Springer-Verlag Berlin Heidelberg, 2009.

[dC07] de Castro, L. N. Fundamentals of natural computing: an overview. *Physics of Life Reviews*, 4(1):1 – 36, 2007.

[DC11] Dodig-Crnkovic, G. Significance of models of computation, from turing model to natural computation. *Minds and Machines*, 21(2):301–322, 2011.

[Die12a] Diem, A. Design principles for droplet-based computing for the classification of environmental situations. diploma thesis, Friedrich-Schiller-Universität Jena, 2012.

[Die12b] Diem, A.; Gruenert, G.; Dittrich, P. Evolution and growth of molecular networks for disease classification. In *Book of Abstracts, European Conference on Complex Systems*, page 44, Brussels, September 2012.

[Dim00] Dimitrov, A. G.; Miller, J. P. Natural time scales for neural encoding. *Neurocomputing*, 32–33:1027 – 1034, 2000.

[Dit98] Dittrich, P.; Buergel, A.; Banzhaf, W. Learning to move a robot with random morphology. In Husbands, P.; Meyer, J.-A., editors, *Evolutionary Robotics*, volume 1468 of *Lect. Notes Comput. Sc.*, pages 165–178. Springer Berlin Heidelberg, 1998.

[Dit01] Dittrich, P.; Ziegler, J.; Banzhaf, W. Artificial chemistries–a review. *Artif Life*, 7(3):225–275, 2001.

[Dit05] Dittrich, P. Chemical computing. In Banâtre, J.-P.; Fradet, P.; Giavitto, J.-L.; Michel, O., editors, *Unconventional Programming Paradigms*, volume 3566 of *Lect. Notes Comput. Sc.*, pages 19–32. Springer Berlin Heidelberg, 2005.

[Dit07] Dittrich, P.; di Fenizio, P. Chemical organisation theory. *Bulletin of Mathematical Biology*, 69(4):1199–1231, May 2007.

[DL06] De Luca, M.; Beckmann, C.; De Stefano, N.; Matthews, P.; Smith, S. M. fmri resting state networks define distinct modes of long-distance interactions in the human brain. *Neuroimage*, 29(4):1359–1367, 2006.

[DLC11] De Lacy Costello, B.; Adamatzky, A.; Jahan, I.; Zhang, L. Towards constructing one-bit binary adder in excitable chemical medium. *Chem. Phys.*, 381:88–99, 2011.

[dW12] de Wit, E.; de Laat, W. A decade of 3c technologies: insights into nuclear organization. *Genes & development*, 26(1):11–24, 2012.

[Egb13] Egbert, M.; Grünert, G.; Escuela, G.; Dittrich, P. Synthetic signalling protocell networks as models of neural computation. In Liò, P.; Miglino, O.; Nicosia, G.; Nolfi, S.; Pavone, M., editors, *Advances in Artificial Life, ECAL 2013*, volume 12, pages 248–249. MIT Press, 2013.

[Egb15] Egbert, M.; Grünert, G.; Dittrich, P. Using feedback to find natural boolean representations in uncoventional computational media. in preparation, 2015.

[Eib08] Eiben, A.; Smith, J. *Introduction to evolutionary computing*. Springer, 2008.

[Esc13] Escuela, G.; Gruenert, G.; Dittrich, P. Symbol representations and signal dynamics in evolving droplet computers. *Natural Computing*, 13(2):247–256, 2013.

[Ese14] Eser, J.; Zheng, P.; Triesch, J. Nonlinear dynamics analysis of a self-organizing recurrent neural network: Chaos waning. *PloS one*, 9(1):e86962, 2014.

[Fer10] Ferrucci, D.; Brown, E.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A. A.; Lally, A.; Murdock, J. W.; Nyberg, E.; Prager, J.; Schlaefer, N.; Welty, C. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79, 2010.

[Fie72] Field, R.; Körös, E.; Noyes, R. Oscillations in chemical systems. II. Thorough analysis of temporal oscillation in the bromate-cerium-malonic acid system. *J. Am. Chem. Soc.*, 94(25):8649–8664, 1972.

[Fis01] Fishman, G. *Discrete-Event Simulation: Modeling, Programming, and Analysis*. Springer Verlag, 2001.

[Fis04] Fischer, G.; Giaccardi, E.; Ye, Y.; Sutcliffe, A. G.; Mehandjiev, N. Meta-design: a manifesto for end-user development. *Commun. Acm.*, 47(9):33–37, 2004.

[Fog66] Fogel, L.; Owens, A.; Walsh, M. *Artificial intelligence through simulated evolution.* John Wiley, New York, 1966.

[Fog90] Fogel, D.; Fogel, L.; Porto, V. Evolving neural networks. *Biological Cybernetics*, 63(6):487–493, 1990.

[Fog94] Fogel, D. An introduction to simulated evolutionary optimization. *IEEE T. Neural Networks*, 5(1):3–14, 1994.

[Fow10] Fowler, M. *Domain-specific languages.* Pearson Education, 2010.

[Fra86] Fraser, A. M.; Swinney, H. L. Independent coordinates for strange attractors from mutual information. *Phys. Rev. A*, 33:1134–1140, Feb 1986.

[Fra07] Fraser, P.; Bickmore, W. Nuclear organization of the genome and the potential for gene regulation. *Nature*, 447(7143):413–417, 2007.

[Fri14] Friedrich, J.; Urbanczik, R.; Senn, W. Code-specific learning rules improve action selection by populations of spiking neurons. *Int. J. Neural Syst.*, 24(05):1450002, 2014.

[Geh12] Gehlen, L.; Gruenert, G.; Jones, M.; Rodley, C.; Langowski, J.; O'Sullivan, J. Chromosome positioning and the clustering of functionally related loci in yeast is driven by chromosomal interactions. *Nucleus (Austin, Tex.)*, 3(4):370, 2012.

[Gen12] Gentili, P. L.; Horvath, V.; Vanag, V. K.; Epstein, I. R. Belousov-zhabotinsky "chemical neuron" as a binary and fuzzy logic processor. *International Journal of Unconventional Computing*, 8(2):177–192, 2012.

[Ger90] Gerhardt, M.; Schuster, H.; Tyson, J. A cellular automation model of excitable media including curvature and dispersion. *Science*, 247(4950):1563, 1990.

[Gia05] Giavitto, J.-L.; Michel, O.; Cohen, J.; Spicher, A. Computations in space and space in computations. In Banâtre, J.-P.; Fradet, P.; Giavitto, J.-L.; Michel, O., editors, *Unconventional Programming Paradigms*, pages 137–152. Springer-Verlag Berlin Heidelberg, 2005.

[Gil77] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[Giz16] Gizynski, K.; Gruenert, G.; Dittrich, P.; Gorecki, J. Evolutionary design of classifiers made of droplets containing a nonlinear chemical medium. *Evolutionary Computation, published online*, 2016.

[GN11] Gutiérrez-Naranjo, M.; Pérez-Jiménez, M. Depth-first search with p systems. In Gheorghe, M.; Hinze, T.; Păun, G.; Rozenberg, G.; Salomaa, A., editors, *Membrane Computing*, volume 6501 of *Lect. Notes Comput. Sc.*, pages 257–264. Springer Berlin Heidelberg, 2011.

[Gor03] Gorecki, J.; Yoshikawa, K.; Igarashi, Y. On chemical reactors that can count. *J. Phys. Chem. A*, 107(10):1664–1669, 2003.

[Gor05] Gorecki, J.; Gorecka, J. N.; Yoshikawa, K.; Igarashi, Y.; Nagahara, H. Sensing the distance to a source of periodic oscillations in a nonlinear chemical medium with the output information coded in frequency of excitation pulses. *Phys. Rev. E*, 72:046201, Oct 2005.

[Gor09] Gorecki, J.; Gorecka, J. N. *Computing in Geometrical Constrained Excitable Chemical Systems*, pages 1352–1376. Springer-Verlag, 2009.

[Gor11a] Gorecki, J.; Szymanski, J.; Gorecka, J. N. Realistic parameters for simple models of the belousov–zhabotinsky reaction. *J. Phys. Chem. A*, 115(32):8855–8859, 2011.

[Gör11b] Görlich, D.; Artmann, S.; Dittrich, P. Cells as semantic systems. *Biochimica et Biophysica Acta (BBA)-General Subjects*, 1810(10):914–923, 2011.

[Gor12] Gorecka, J. N.; Gorecki, J.; Szymanski, J.; Gizynski, K. A simple model of interactions between belousov-zhabotinsky droplets. *not yet published*, 2012.

[Gör13] Görlich, D.; Escuela, G.; Gruenert, G.; Dittrich, P.; Ibrahim, B. Molecular codes through complex formation in a model of the human inner kinetochore. *Biosemiotics*, 7(2):223–247, 2013.

[Gor14] Gorecki, J.; Gorecka, J. N.; Adamatzky, A. Information coding with frequency of oscillations in belousov-zhabotinsky encapsulated disks. *Phys. Rev. E*, 89:042910, Apr 2014.

[Got96] Gotta, M.; Laroche, T.; Formenton, A.; Maillet, L.; Scherthan, H.; Gasser, S. M. The clustering of telomeres and colocalization with rap1, sir3, and sir4 proteins in wild-type saccharomyces cerevisiae. *The Journal of cell biology*, 134(6):1349–1363, 1996.

[Gre78] Greenberg, J. M.; Hastings, S. P. Spatial patterns for discrete models of diffusion in excitable media. *SIAM Journal on Applied Mathematics*, 34(3):pp. 515–523, 1978.

[Gre96] Green, T. R. G.; Petre, M. Usability analysis of visual programming environments: a 'cognitive dimensions' framework. *Journal of Visual Languages & Computing*, 7(2):131–174, 1996.

[Gru10] Gruenert, G.; Ibrahim, B.; Lenser, T.; Lohel, M.; Hinze, T.; Dittrich, P. Rule-based spatial modeling with diffusing, geometrically constrained molecules. *BMC Bioinformatics*, 11(1):307, 2010.

[Gru11a] Gruenert, G.; Dittrich, P.; Zauner, K.-P. Artificial wet neuronal networks from compartmentalised excitable chemical media. *ERCIM News*, (85):30–32, 2011.

[Gru11b] Gruenert, G.; Escuela, G.; Dittrich, P.; Hinze, T. Morphological algorithms: Membrane receptor-ligand interactions and rule-based molecule graph evolution for exact set cover problem. In Gheorghe, M.; Pâun, G.; Verlan, S., editors, *Proceedings of the Twelfth Conference on Membrane Computing (CMC12)*, pages 169 – 190, Fontainebleau/Paris, France, July 2011.

[Gru12] Gruenert, G.; Escuela, G.; Dittrich, P. Symbol representations in evolving droplet computers. In Durand-Lose, J.; Jonoska, N., editors, *Unconventional Computation and Natural Computation - 11th International Conference, UCNC 2012, Orléan, France, September 3-7, 2012. Proceedings*, volume 7445 of *Lect. Notes Comput. Sc.*, pages 130–140. Springer-Verlag Berlin Heidelberg, 2012.

[Gru13] Gruenert, G.; Szymanski, J.; Holley, J.; Escuela, G.; Diem, A.; Ibrahim, B.; Adamatzky, A.; Gorecki, J.; Dittrich, P. Multi-scale modelling of computers made from excitable chemical droplets. *Int. J. Unconv. Comput.*, 9:237–266, 2013.

[Gru14] Gruenert, G.; Gizynski, K.; Escuela, G.; Ibrahim, B.; Gorecki, J.; Dittrich, P. Understanding networks of computing chemical droplet neurons based on information flow. *Int. J. Neur. Syst.*, page 1450032, 2014.

[Gyo90] Gyorgyi, L.; Turànyi, T.; Field, R. Mechanistic details of the oscillatory belousov-zhabotinskii reaction. *J. Phys. Chem.*, 94(18):7162–7170, 1990.

[Had10] Hadorn, M.; Hotz, P. Dna-mediated self-assembly of artificial vesicles. *PLoS One*, 5(3):e9886, 2010.

[Hal04] Halperin, E.; Karp, R. M. Perfect phylogeny and haplotype assignment. In *Proceedings of the eighth annual international conference on Research in computational molecular biology*, pages 10–19. ACM, 2004.

[Han01] Hansen, N.; Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, 9(2):159–195, 2001.

[Har96] Harrison, R.; Smaraweera, L.; Dobie, M.; Lewis, P. Comparing programming paradigms: an evaluation of functional and object-oriented programs. *Software Engineering Journal*, 11(4):247–254, 1996.

[He11] He, Z.; Yang, C.; Yu, W. A partial set covering model for protein mixture identification using mass spectrometry data. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(2):368–380, 2011.

[Heu01] Heun, P.; Laroche, T.; Shimada, K.; Furrer, P.; Gasser, S. M. Chromosome dynamics in the yeast interphase nucleus. *Science*, 294(5549):2181–2186, 2001.

[Hig96] Higuchi, T.; Iwata, M.; Kajitani, I.; Yamada, H.; Manderick, B.; Hirao, Y.; Murakawa, M.; Yoshizawa, S.; Furuya, T. Evolvable hardware with genetic learning. In *Circuits and Systems, 1996. ISCAS'96., Connecting the World., 1996 IEEE International Symposium on*, volume 4, pages 29–32. IEEE, 1996.

[Hje91] Hjelmfelt, A.; Weinberger, E. D.; Ross, J. Chemical implementation of neural networks and turing machines. *Proceedings of the National Academy of Sciences*, 88(24):10983–10987, 1991.

[Hla06] Hlavacek, W. S.; Faeder, J. R.; Blinov, M. L.; Posner, R. G.; Hucka, M.; Fontana, W. Rules for modeling signal-transduction systems. *Sci STKE*, 2006(344):re6, 2006.

[Hoh02] Hohmuth, M.; Tews, H.; Stephens, S. G. Applying source-code verification to a microkernel: the vfiasco project. In *Proceedings of the 10th workshop on ACM SIGOPS European workshop*, EW 10, pages 165–169, New York, NY, USA, 2002. ACM.

[Hol75] Holland, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* The University of Michigan Press, Ann Arbor, 1975.

[Hol11a] Holley, J.; Adamatzky, A.; Bull, L.; De Lacy Costello, B.; Jahan, I. Computational modalities of belousov-zhabotinsky encapsulated vesicles. *Nano Communication Networks*, 2:50–61, 2011.

[Hol11b] Holley, J.; Jahan, I.; Costello, B.; Bull, L.; Adamatzky, A. Logical and arithmetic circuits in belousov zhabotinsky encapsulated discs. *Physical Review E*, 84(5):056110, 2011.

[Hor01] Hornby, G. S.; Pollack, J. B. The advantages of generative grammatical encodings for physical design. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 600–607. IEEE, 2001.

[How03] Howard, M.; LeBlanc, D. *Writing Secure Code.* Best Practices Series. Microsoft Press, 2003.

[HS06] Horowitz-Scherer, R. A.; Woodcock, C. L. Organization of interphase chromatin. *Chromosoma*, 115(1):1–14, 2006.

[Hüt12] Hütt, M.-T.; Jain, M. K.; Hilgetag, C. C.; Lesne, A. Stochastic resonance in discrete excitable dynamics on graphs. *Chaos, Solitons & Fractals*, (0):–, 2012.

[Ibr08] Ibrahim, B.; Diekmann, S.; Schmitt, E.; Dittrich, P. In-silico modeling of the mitotic spindle assembly checkpoint. *PLoS One*, 3(2):e1555, 2008.

[Ibr13] Ibrahim, B.; Henze, R.; Gruenert, G.; Egbert, M.; Huwald, J.; Dittrich, P. Spatial rule-based modeling: A method and its application to the human mitotic kinetochore. *Cells*, 2(3):506–544, 2013.

[Iga11] Igarashi, Y.; Gorecki, J. Chemical diodes built with controlled excitable media. *Int. J. Unconv. Comput.*, 7(3):141–158, 2011.

[Jae01] Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148:34, 2001.

[Jaf87] Jaffar, J.; Lassez, J.-L. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '87, pages 111–119, New York, NY, USA, 1987. ACM.

[Jan10] Januszewski, M.; Kostur, M. Accelerating numerical solution of stochastic differential equations with cuda. *Comput. Phys. Commun.*, 181(1):183 – 188, 2010.

[Jan12] Janowicz, K.; Hitzler, P. The digital earth as knowledge engine. *Semantic Web*, 3(3):213–221, 2012.

[Jen02] Jenkins, T. On the difficulty of learning to program. In *Proceedings of the 3rd Annual Conference of the Learning and Teaching Support Network (LTSN) Centre for Information and Computer Sciences*, volume 4, pages 53–58, 2002.

[Jet89] Jetschke, G. *Mathematik der Selbstorganisation: qualitative Theorie nichtlinearer dynamischer Systeme und gleichgewichtsferner Strukturen in Physik, Chemie und Biologie*. Dt. Verl. d. Wiss., 1989.

[Jin10] Jin, S.-H.; Lin, P.; Hallett, M. Linear and nonlinear information flow based on time-delayed mutual information method and its application to corticomuscular interaction. *Clin. Neurophysiol.*, 121(3):392 – 401, 2010.

[Jol02] Jolliffe, I. T. *Principal Component Analysis*. Springer, New York, 2nd edition, 2002.

[Kae96] Kaelbling, L. P.; Littman, M. L.; Moore, A. W. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Kar72] Karp, R. M. *Reducibility among combinatorial problems*. Springer, 1972.

[Kee86] Keener, J. P.; Tyson, J. J. Spiral waves in the Belousov-Zhabotinskii reaction. *Physica D: Nonlinear Phenomena*, 21(2-3):307–324, September 1986.

[Kic97] Kiczales, G.; Lamping, J.; Mendhekar, A.; Maeda, C.; Lopes, C.; Loingtier, J.-M.; Irwin, J. *Aspect-oriented programming*. Springer, 1997.

[Klu94] Klug, S. J.; Famulok, M. All you wanted to know about selex. *Mol. Biol. Rep.*, 20(2):97–107, 1994.

[Knu69] Knuth, D. E. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*. Addison-Wesley, 1969.

[Knu73] Knuth, D. E. *The Art of Computer Programming, Volume I: Fundamental Algorithms, 2nd Edition*. Addison-Wesley, Reading (Mass.) Menlo Park (Calif.) London etc, 1973.

[Knu00] Knuth, D. E. Dancing links. *arXiv preprint cs/0011047*, 2000.

[Kot07] Kotsiantis, S. B. Supervised machine learning: a review of classification techniques. *Informatica*, 31(3):249–268, 2007.

[Koz89] Koza, J. R. Hierarchical genetic algorithms operating on populations of computer programs. In Sridharan, N. S., editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, volume 1, pages 768–774, Detroit, MI, USA, 20-25 August 1989. Morgan Kaufmann.

[Koz96] Koza, J. R.; Andre, D.; Bennett III, F. H.; Keane, M. A. Use of automatically defined functions and architecture-altering operations in automated circuit synthesis with genetic programming. In *Proceedings of the First Annual Conference on Genetic Programming*, pages 132–140. MIT Press, 1996.

[Kra04] Kraskov, A.; Stögbauer, H.; Grassberger, P. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004.

[Kuh86] Kuhnert, L. A new optical photochemical memory device in a light-sensitive chemical active medium. *Nature*, 319:393, 1986.

[Lan61] Landauer, R. Irreversibility and heat generation in the computing process. *IBM journal of research and development*, 5(3):183–191, 1961.

[Lat09] Latham, P. E.; Roudi, Y. Mutual information. *Scholarpedia*, 4(1):1658, 2009. revision #122173.

[Laz09] Lazar, A.; Pipa, G.; Triesch, J. Sorn: a self-organizing recurrent neural network. *Frontiers in computational neuroscience*, 3, 2009.

[Leh02] Lehn, J.-M. Toward self-organization and complex matter. *Science*, 295(5564):2400–2403, 2002.

[Liz08] Lizier, J. T.; Prokopenko, M.; Zomaya, A. Y. Local information transfer as a spatiotemporal filter for complex systems. *Phys. Rev. E*, 77:026110, Feb 2008.

[Liz10] Lizier, J.; Prokopenko, M.; Zomaya, A. Information modification and particle collisions in distributed computation. *Chaos*, 20(3):037109, 2010.

[Luk09] Lukoševičius, M.; Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149, 2009.

[Maa02] Maass, W.; Natschläger, T.; Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Comput.*, 14(11):2531–2560, November 2002.

[Maa04] Maass, W.; Markram, H. On the computational power of circuits of spiking neurons. *J. Comput. Syst. Sci.*, 69(4):593 – 616, 2004.

[Mac04] MacLennan, B. J. Natural computation and non-turing models of computation. *Theoretical Computer Science*, 317(1–3):115 – 145, 2004.

[Mag09] Maglia, G.; Heron, A.; Hwang, W.; Holden, M.; Mikhailova, E.; Li, Q.; Cheley, S.; Bayley, H. Droplet networks with incorporated protein diodes show collective properties. *Nat. Nanotechnol.*, 4(7):437–440, 2009.

[Mar06] Markram, H. The blue brain project. *Nat. Rev. Neurosci.*, 7(2):153–160, 2006.

[Mar10] Markstrum, S. Staking claims: a history of programming language design claims and evidence: a positional work in progress. In *Evaluation and Usability of Programming Languages and Tools*, page 7. ACM, 2010.

[Mat06a] Matsumaru, N.; Dittrich, P. Organization-oriented chemical programming for the organic design of distributed computing systems. In *Proceedings of the 1st international conference on Bio inspired models of network, information and computing systems*, page 14, New York, NY, USA, 2006. ACM.

[Mat06b] Matsuura, T.; Yomo, T. In vitro evolution of proteins. *J. Biosci. Bioeng.*, 101(6):449–456, 2006.

[Mat07] Matsumaru, N.; Centler, F.; di Fenizio, P. S.; Dittrich, P. Chemical organization theory as a theoretical base for chemical computing. *Int. J. Unconv. Comput.*, 3(4), 2007.

[Mil00a] Miller, J.; Job, D.; Vassilev, V. Principles in the evolutionary design of digital circuits—part i. *Genetic programming and evolvable machines*, 1(1):7–35, 2000.

[Mil00b] Miller, J. F.; Thomson, P. Cartesian genetic programming. In *Genetic Programming*, pages 121–132. Springer, 2000.

[Mil14] Miller, J. F.; Harding, S. L.; Tufte, G. Evolution-in-materio: evolving computation in materials. *Evolutionary Intelligence*, 7(1):49–67, 2014.

[Min69] Minsky, M.; Papert, S. *Perceptron: an introduction to computational geometry*. 1969.

[Mis05] Misteli, T. Concepts in nuclear architecture. *BioEssays*, 27(5):477–487, 2005.

[Moo03] Moore, J. L.; Folkmann, M.; Balmford, A.; Brooks, T.; Burgess, N.; Rahbek, C.; Williams, P. H.; Krarup, J. Heuristic and optimal solutions for set-covering problems in conservation biology. *Ecography*, 26(5):595–601, 2003.

[Mot99] Motoike, I.; Yoshikawa, K. Information operations with an excitable field. *Physical Review E*, 59(5):5354, 1999.

[Mot01] Motoike, I.; Yoshikawa, K.; Iguchi, Y.; Nakata, S. Real-time memory on an excitable field. *Physical Review E*, 63(3):036220, 2001.

[Mus07] Musacchio, A.; Salmon, E. D. The spindle-assembly checkpoint in space and time. *Nature reviews Molecular cell biology*, 8(5):379–393, 2007.

[Nat12] National Institute of Standards and Technology, . *FIPS PUB 180-4: Secure Hash Standard*. 2012. Supersedes FIPS 180-3.

[Nem04] Nemenman, I.; Bialek, W.; de Ruyter van Steveninck, R. Entropy and information in neural spike trains: Progress on the sampling problem. *Phys. Rev. E*, 69:056111, May 2004.

[Nic12] Nichele, S.; Tufte, G. Genome parameters as information to forecast emergent developmental behaviors. In *Unconventional Computation and Natural Computation*, pages 186–197. Springer, 2012.

[Nis06] Nishida, T. Y. Membrane algorithms. In *Membrane Computing*, pages 55–66. Springer, 2006.

[Noy72] Noyes, R.; Field, R.; Koros, E. Oscillations in chemical systems. i. detailed mechanism in a system showing temporal oscillations. *J. Am. Chem. Soc.*, 94(4):1394–1395, 1972.

[Ong03] Ong, Y. S.; Nair, P. B.; Keane, A. J. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA journal*, 41(4):687–696, 2003.

[Pah08] Pahle, J.; Green, A.; Dixon, C. J.; Kummer, U. Information transfer in signaling pathways: A study using coupled simulated and experimental data. *BMC Bioinformatics*, 9(1):139, 2008.

[Pan96] Panzeri, S.; Treves, A. Analytical estimates of limited sampling biases in different information measures. *Network-Comp. Neural*, 7:87–107, 1996.

[Pău06] Păun, G. *Applications of Membrane Computing*, chapter Introduction to Membrane Computing, pages 1–42. Springer Berlin, 2006.

[Per05] Pereda, E.; Quiroga, R. Q.; Bhattacharya, J. Nonlinear multivariate analysis of neurophysiological signals. *Prog. Neurobiol.*, 77(1–2):1 – 37, 2005.

[Per11] Perpelescu, M.; Fukagawa, T. The abcs of cenps. *Chromosoma*, 120(5):425–446, 2011.

[Pfa01] Pfaffmann, J. O.; Zauner, K.-P. Scouting context-sensitive components. In *Evolvable Hardware, 2001. Proceedings. The Third NASA/DoD Workshop on*, pages 14–20. IEEE, 2001.

[Pli95] Plimpton, S. J. Fast parallel algorithms for short-range molecular dynamics. *J Comp Phys*, 117:1–19, 1995.

[Pre92] Press, W.; Flannery, B.; Teukolsky, S.; Vetterling, W. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, England, 2nd edition, 1992.

[Pre94] Prechelt, L. Proben1: A set of neural network benchmark problems and benchmarking rules. Technical report, 1994.

[Pri68] Prigogine, I.; Lefever, R. Symmetry breaking instabilities in dissipative systems. ii. *J. Chem. Phys.*, 48(4):1695–1700, 1968.

[Qui76] Quigley, G. J.; Rich, A. Structural domains of transfer rna molecules. *Science*, 194(4267):796–806, 1976.

[Qui09] Quiroga, R. Q.; Panzeri, S. Extracting information from neuronal populations: information theory and decoding approaches. *Nat. Rev. Neurosci.*, 10(3):173–185, 2009.

[Rec71] Rechenberg, I. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. PhD thesis, Technical University of Berlin, Department of Process Engineering, 1971.

[Rep06] Repenning, A.; Ioannidou, A. What makes end-user development tick? 13 design guidelines. In Lieberman, H.; Paternò, F.; Wulf, V., editors, *End User Development*, Human-Computer Interaction Series, pages 51–85. Springer Netherlands, 2006.

[Rob03] Robins, A.; Rountree, J.; Rountree, N. Learning and teaching programming: A review and discussion. *Computer Science Education*, 13(2):137–172, 2003.

[Rod09] Rodley, C.; Bertels, F.; Jones, B.; O'sullivan, J. Global identification of yeast chromosome interactions using genome conformation capture. *Fungal Genet. Biol.*, 46(11):879–886, 2009.

[Roe09] Roegel, D. Anecdotes: Prototype fragments from babbage's first difference engine. *Annals of the History of Computing, IEEE*, 31(2):70–75, April 2009.

[Rog87] Rogers, H. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, Mass, 1987.

[Ros14] Rosselló, J. L.; Canals, V.; Oliver, A.; Morro, A. Studying the role of synchronized and chaotic spiking neural ensembles in neural information processing. *Int. J. Neural Syst.*, 24(05):1430003, 2014.

[Rot04] Rothemund, P.; Papadakis, N.; Winfree, E. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biol.*, 2(12):e424, 2004.

[Rot06] Rothemund, P. W. K. Folding dna to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, Mar 2006.

[Rou99] Roulston, M. S. Estimating the errors on measured entropy and mutual information. *Physica D: Nonlinear Phenomena*, 125(3–4):285–294, 1999.

[Rub14] Rubenstein, M.; Cornejo, A.; Nagpal, R. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.

[Ruh14] Ruhland, F. Evolution tautologischer netzwerke. Bachelor's thesis, Friedrich-Schiller-University Jena, Germany, 2014.

[San09] Santaguida, S.; Musacchio, A. The life and miracles of kinetochores. *The EMBO journal*, 28(17):2511–2531, 2009.

[SB12] Schuster-Böckler, B.; Lehner, B. Chromatin organization is a major influence on regional mutation rates in human cancer cells. *Nature*, 2012.

[Sch75] Schwefel, H.-P. *Evolutionsstrategie und numerische Optimierung.* PhD thesis, Technische Universität Berlin, 1975.

[Sch85] Schaffer, J. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st international conference on genetic algorithms*, pages 93–100. L. Erlbaum Associates Inc., 1985.

[Sch00] Schreiber, T. Measuring information transfer. *Phys. Rev. Lett.*, 85:461–464, Jul 2000.

[See06] Seelig, G.; Soloveichik, D.; Zhang, D. Y.; Winfree, E. Enzyme-free nucleic acid logic circuits. *Science*, 314(5805):1585–1588, 2006.

[Sha48] Shannon, C. E. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

[Sha14] Shapero, S.; Zhu, M.; Hasler, J.; Rozell, C. Optimal sparse approximation with integrate and fire neurons. *Int. J. Neural Syst.*, 24(05):1440001, 2014.

[Shi03] Shimizu, T. S.; Aksenov, S. V.; Bray, D. A spatially extended stochastic model of the bacterial chemotaxis signalling pathway. *J. Mol. Biol.*, 329(2):291–309, May 2003.

[Sim04] Simpson, T. W.; Booker, A. J.; Ghosh, D.; Giunta, A. A.; Koch, P. N.; Yang, R.-J. Approximation methods in multidisciplinary analysis and optimization: a panel discussion. *Structural and multidisciplinary optimization*, 27(5):302–313, 2004.

[Smi94] Smith, D. C.; Cypher, A.; Spohrer, J. Kidsim: programming agents without a programming language. *Commun. Acm.*, 37(7):54–67, 1994.

[Sol04] Solé, R. V.; Munteanu, A. The large-scale organization of chemical reaction networks in astrophysics. *Europhys. Lett.*, 68(2):170, 2004.

[Spr05] Sproul, D.; Gilbert, N.; Bickmore, W. A. The role of chromatin structure in regulating the expression of clustered genes. *Nat. Rev. Genet.*, 6(10):775–781, 2005.

[Sta08] Staniek, M.; Lehnertz, K. Symbolic transfer entropy. *Phys. Rev. Lett.*, 100:158101, Apr 2008.

[Ste96] Steinbock, O.; Kettunen, P.; Showalter, K. Chemical wave logic gates. *J. Phys. Chem.*, 100(49):18970–18975, 1996.

[Ste98] Steinbock, O.; Müller, S. Radius-dependent inhibition and activation of chemical oscillations in small droplets. *J. Phys. Chem. A*, 102(32):6485–6490, 1998.

[Ste12] Stepney, S. Programming unconventional computers: Dynamics, development, self-reference. *Entropy*, 14(10):1939–1952, 2012.

[Sto08] Stone, C.; Toth, R.; de Lacy Costello, B.; Bull, L.; Adamatzky, A. Coevolving cellular automata with memory for chemical computing: Boolean logic gates in the bz reaction. In *PPSN*, pages 579–588, 2008.

[Str98] Strong, S. P.; Koberle, R.; de Ruyter van Steveninck, R. R.; Bialek, W. Entropy and information in neural spike trains. *Phys. Rev. Lett.*, 80:197–200, Jan 1998.

[Sug12] Sugihara, G.; May, R.; Ye, H.; Hsieh, C.-h.; Deyle, E.; Fogarty, M.; Munch, S. Detecting causality in complex ecosystems. *Science*, 338(6106):496–500, 2012.

[Suz13] Suzuki, Y. Harness the nature for computation. In *Natural Computing and Beyond*, pages 49–70. Springer, 2013.

[Szy10] Szymanski, J.; Gorecki, J. Chemical pulses propagating inside a narrowing channel and their possible computational applications. *Int. J. Unconv. Comput.*, 6(6):461–471, 2010.

[Szy11] Szymanski, J.; Gorecka, J. N.; Igarashi, Y.; Gizynski, K.; Gorecki, J.; Zauner, K.-P.; Planque, M. D. Droplets with information processing ability. *Int. J. Unconv. Comput.*, 7(3):185–200, 2011.

[Tak01] Takagi, H. Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. *Proc. IEEE*, 89(9):1275–1296, 2001.

[Teu08] Teuscher, C.; Nemenman, I.; Alexander, F. J. Novel computing paradigms: Quo vadis? *Physica D: Nonlinear Phenomena*, 237(9):v–viii, 2008.

[Thu13] Thutupalli, S.; Herminghaus, S. Tuning active emulsion dynamics via surfactants and topology. *Eur. Phys. J. E*, 36(8):1–10, 2013.

[Tót95] Tóth, Á.; Showalter, K. Logic gates in excitable media. *J. Chem. Phys.*, 103:2058, 1995.

[Tsc13] Tschernyschkow, S.; Herda, S.; Gruenert, G.; Döring, V.; Görlich, D.; Hofmeister, A.; Hoischen, C.; Dittrich, P.; Diekmann, S.; Ibrahim, B. Rule-based modeling and simulations of the inner kinetochore structure. *Progress in Biophysics and Molecular Biology*, 113(1):33 – 45, 2013.

[Tur36] Turing, A. M. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58:345–363, 1936.

[Vap00] Vapnik, V. *The nature of statistical learning theory.* springer, 2000.

[VB12] Van Bortle, K.; Corces, V. G. Nuclear organization and genome function. *Annu. Rev. Cell Dev. Biol.*, 28:163, 2012.

[Vid11] Vidybida, A. Testing of information condensation in a model reverberating spiking neural network. *Int. J. Neural Syst.*, 21(03):187–198, 2011.

[vN66] von Neumann, J. *Theory of Self-Reproducing Automata.* University of Illinois Press, Champaign, IL, USA, 1966.

[Wat02] Watson, R. A.; Ficici, S. G.; Pollack, J. B. Embodied evolution: Distributing an evolutionary algorithm in a population of robots. *Robotics and Autonomous Systems*, 39(1):1–18, 2002.

[Wei02] Weicker, K. *Evolutionäre Algorithmen.* Vieweg+Teubner, 2002.

[Wib11] Wibral, M.; Rahm, B.; Rieder, M.; Lindner, M.; Vicente, R.; Kaiser, J. Transfer entropy in magnetoencephalographic data: Quantifying information flow in cortical and cerebellar networks. *Prog. Biophys. Mol. Biol.*, 105:80 – 97, 2011.

[Wil10a] Williams, P. L.; Beer, R. D. Information dynamics of evolved agents. In Doncieux, S.; Girard, B.; Guillot, A.; Hallam, J.; Meyer, J.-A.; Mouret, J.-B., editors, *From Animals to Animats 11*, volume 6226 of *Lect. Notes Comput. Sc.*, pages 38–49. Springer Berlin Heidelberg, 2010.

[Wil10b] Williams, P. L.; Beer, R. D. Nonnegative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*, 2010.

[Win72] Winfree, A. T. Spiral waves of chemical activity. *Science*, 175(4022):634–636, 1972.

[Win98] Winfree, E.; Liu, F.; Wenzler, L.; Seeman, N. Design and self-assembly of two-dimensional DNA crystals. *Nature*, 394(6693):539–544, 1998.

[Win06] Wing, J. M. Computational thinking. *Commun. Acm.*, 49(3):33–35, 2006.

[Wol83] Wolfram, S. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, July 1983.

[Wol97] Wolpert, D. H.; Macready, W. G. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[Xia14] Xiang, L.; Qiangbin, W. DNA-programmed self-assembly of photonic nanoarchitectures. *NPG Asia Mater*, 6:e97, apr 2014.

[Yam07] Yamazaki, T.; Tanaka, S. The cerebellum as a liquid state machine. *Neural Networks*, 20(3):290–297, 2007.

[Yao97] Yao, X.; Liu, Y. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, 1997.

[Yin08] Yin, P.; Choi, H. M. T.; Calvert, C. R.; Pierce, N. A. Programming biomolecular self-assembly pathways. *Nature*, 451(7176):318–322, January 2008.

[Zai70] Zaikin, A. N.; Zhabotinsky, A. M. Concentration wave propagation in two-dimensional liquid-phase self-oscillating system. *Nature*, 225(5232):535–537, February 1970.

[Zau96] Zauner, K.-P.; Conrad, M. Parallel computing with dna: Toward the anti-universal machine. In Voigt, H.-M.; Ebeling, W.; Rechenberg, I.; Schwefel, H.-P., editors, *Parallel Problem Solving from Nature — PPSN IV*, volume 1141 of *Lect. Notes Comput. Sc.*, pages 696–705. Springer Berlin Heidelberg, 1996.

[Zau05a] Zauner, K.-P. From prescriptive programming of solid-state devices to orchestrated self-organisation of informed matter. In *Unconventional Programming Paradigms*, pages 47–55. Springer, 2005.

[Zau05b] Zauner, K. Molecular information technology. *Crit. Rev. Solid State*, 30(1):33–69, 2005.

[Zha73] Zhabotinsky, A. M.; Zaikin, A. N. Autowave processes in a distributed chemical system. *J. Theor. Biol.*, 40(1):45–61, 1973.

[Zha14] Zhang, G.; Rong, H.; Neri, F.; Pérez-Jiménez, M. J. An optimization spiking neural p system for approximately solving combinatorial optimization problems. *Int. J. Neural Syst.*, 24(05):1440006, 2014.

[Zit04] Zitzler, E.; Laumanns, M.; Bleuler, S. A tutorial on evolutionary multiobjective optimization. *Metaheuristics for Multiobjective Optimisation*, pages 3–37, 2004.

Appendix A

List of Incorporated Publications

In the course of this thesis, 14 manuscripts for conferences and journals were (co-) authored by GG, which partially contributed to this thesis. These articles are listed in Table A.1 together with each manuscript's contribution by the (co-) author GG.

Sections	Paper	Role of GG
Chapter 2 [Gru13, Gru11a]	Gruenert, G.; Szymanski, J.; Holley, J.; Escuela, G.; Diem, A.; Ibrahim, B.; Adamatzky, A.; Gorecki, J.; Dittrich, P. Multi-scale modelling of computers made from excitable chemical droplets. *Int. J. Unconv. Comput.*, 9:237–266, 2013 Gruenert, G.; Dittrich, P.; Zauner, K.-P. Artificial wet neuronal networks from compartmentalised excitable chemical media. *ERCIM News*, (85):30–32, 2011	Main work.
Chapter 3 [Gru12, Esc13]	Gruenert, G.; Escuela, G.; Dittrich, P. Symbol representations in evolving droplet computers. In Durand-Lose, J.; Jonoska, N., editors, *Unconventional Computation and Natural Computation - 11th International Conference, UCNC 2012, Orléan, France, September 3-7, 2012. Proceedings*, volume 7445 of *Lect. Notes Comput. Sc.*, pages 130–140. Springer-Verlag Berlin Heidelberg, 2012, Escuela, G.; Gruenert, G.; Dittrich, P. Symbol representations and signal dynamics in evolving droplet computers. *Natural Computing*, 13(2):247–256, 2013	Main work.
Section 3.2 [Die12b]	Diem, A.; Gruenert, G.; Dittrich, P. Evolution and growth of molecular networks for disease classification. In *Book of Abstracts, European Conference on Complex Systems*, page 44, Brussels, September 2012,	Fitness function and simulator.
Chapter 4 [Gru14]	Gruenert, G.; Gizynski, K.; Escuela, G.; Ibrahim, B.; Gorecki, J.; Dittrich, P. Understanding networks of computing chemical droplet neurons based on information flow. *Int. J. Neur. Syst.*, page 1450032, 2014	Main work.
Chapter 4 [Giz16]	Gizynski, K.; Gruenert, G.; Dittrich, P.; Gorecki, J. Evolutionary design of classifiers made of droplets containing a nonlinear chemical medium. *Evolutionary Computation, published online*, 2016	Fitness function and simulator.
Chapter 5 [Egb13], [Egb15]	Egbert, M.; Grünert, G.; Escuela, G.; Dittrich, P. Synthetic signalling protocell networks as models of neural computation. In Liò, P.; Miglino, O.; Nicosia, G.; Nolfi, S.; Pavone, M., editors, *Advances in Artificial Life, ECAL 2013*, volume 12, pages 248–249. MIT Press, 2013 Egbert, M.; Grünert, G.; Dittrich, P. Using feedback to find natural boolean representations in uncoventional computational media. in preparation, 2015	Formalization, loop enumeration, evolution of tautological loops and droplet simulation.
Chapter 6 [Gru11b]	Gruenert, G.; Escuela, G.; Dittrich, P.; Hinze, T. Morphological algorithms: Membrane receptor-ligand interactions and rule-based molecule graph evolution for exact set cover problem. In Gheorghe, M.; Pâun, G.; Verlan, S., editors, *Proceedings of the Twelfth Conference on Membrane Computing (CMC12)*, pages 169 – 190, Fontainebleau/Paris, France, July 2011	Embodied evolution experiment.
Section 7.1 [Tsc13, Gör13, Ibr13]	Tschernyschkow, S.; Herda, S.; Gruenert, G.; Döring, V.; Görlich, D.; Hofmeister, A.; Hoischen, C.; Dittrich, P.; Diekmann, S.; Ibrahim, B. Rule-based modeling and simulations of the inner kinetochore structure. *Progress in Biophysics and Molecular Biology*, 113(1):33 – 45, 2013 Görlich, D.; Escuela, G.; Gruenert, G.; Dittrich, P.; Ibrahim, B. Molecular codes through complex formation in a model of the human inner kinetochore. *Biosemiotics*, 7(2):223–247, 2013 Ibrahim, B.; Henze, R.; Gruenert, G.; Egbert, M.; Huwald, J.; Dittrich, P. Spatial rule-based modeling: A method and its application to the human mitotic kinetochore. *Cells*, 2(3):506–544, 2013	Help with experiment design, analysis and simulation software.
Section 7.2 [Geh12]	Gehlen, L.; Gruenert, G.; Jones, M.; Rodley, C.; Langowski, J.; O'Sullivan, J. Chromosome positioning and the clustering of functionally related loci in yeast is driven by chromosomal interactions. *Nucleus (Austin, Tex.)*, 3(4):370, 2012	Simulation system and data analysis.

Table A.1: List of journal and conference papers associated with this thesis.

Appendix B

Ehrenwörtliche Erklärung

Hiermit erkläre ich,

- dass mir die Promotionsordnung der Fakultät für Mathematik und Informatik bekannt ist,
- dass ich die Dissertation selbst angefertigt habe, keine Textabschnitte oder Ergebnisse eines Dritten oder eigenen Prüfungsarbeiten ohne Kennzeichnung übernommen und alle von mir benutzten Hilfsmittel, persönliche Mitteilungen und Quellen in meiner Arbeit angegeben habe,
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe,
- dass ich die gleiche, eine in wesentlichen Teilen ähnliche bzw. eine andere Abhandlung nicht bereits als Dissertation eingereicht habe.

Bei der Auswahl des Materials sowie bei der Herstellung des Manuskripts haben mich Prof. Peter Dittrich und Prof. Jerzy Gorecki unterstützt. Weitere Kooperationspartner sind in den einzelnen Kapiteln erwähnt und in Tabelle A.1 zusammengefasst.

Frau Stephanie Luther wurde für das Korrekturlesen in Bezug auf orthographische und grammatikalische Korrektheit des Einleitungskapitels bezahlt. Darüber hinaus haben Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Jena, den ....

Unterschrift