TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Developing Mathematical Formalisms for Cellular Automata in Modelling and Simulation

Ausgeführt am Institut für

Analysis & Scientific Computing

der Technischen Universität Wien
unter der Anleitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker

durch

Günter Schneckenreither

Humboldtgasse 15/12, 1100 Wien

Wien, am 12. September 2014

Erklärung zur Verfassung der Arbeit

Günter Schneckenreither
Humboldtgasse 15/12, 1100 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift)

Danksagung

Zuerst möchte ich Prof. Felix Breitenecker danken, der sich stets für die wissenschaftliche und auch kulturelle Weiterbildung in der Forschungsgruppe Simulation / ARGESIM einsetzt und mir somit ein äußerst interessantes und lehrreiches Studium ermöglicht hat.

Außerdem gilt mein Dank Niki Popper und der Drahtwarenhandlung, die mich im Laufe der Jahre immer unterstützt und motiviert haben.

Schlussendlich möchte ich meiner gesamten Familie, Verwandten und Freunden und ganz besonders meiner Mutter für Unterstützung und Rückhalt während meines Studiums danken.

Kurzfassung

Zelluläre Automaten werden in verschiedensten wissenschaftlichen Gebieten als Methode der mathematischen Modellbildung und Simulation angewandt. Für gewöhnlich stellt ein Zellulärer Automat eine mikroskopische Beschreibung oder Abstraktion eines natürlichen Systems dar. Die Werkzeuge für die Herleitung und Validierung eines Modellbildungsansatzes mit Zellulären Automaten sind jedoch oft auf einen konkreten Anwendungsfall ausgelegt. Daher existiert weder ein etablierter mathematischer Formalismus für Zelluläre Automaten als Methode der Modellbildung und Simulation noch allgemeine Methoden für die Analyse und Validierung.

Diese Arbeit betrachtet Zelluläre Automaten als einen eigenständigen mathematischen Modellierungsansatz und beinhaltet eine rigorose und systematische Definition von verschiedenen Ausprägungen von Zellulären Automaten. Das Ziel ist es, Methoden und Ansätze für eine generelle Diskussion und einen Vergleich dieses Modellierungsansatzes bereitzustellen. Dabei liegt der Fokus weniger auf der Einführung einer Klassifikation von Zellulären Automaten, sondern auf der Bereitstellung und Vereinfachung von Methoden für Analyse, Validierung und Vergleich. Zusätzlich erleichtert eine systematische Beschreibung von Zellulären Automaten die Vermittlung des gewählten Modellierungsansatzes und legt den Grundstein für weitere und detailliertere Untersuchungen.

Die mathematische Beschreibung von Zellulären Automaten in dieser Arbeit basiert auf lokal definierten Aktualisierungsfunktionen, die in iterativer Weise auf einem Graphen oder regelmäßigen Gitter operieren. Ein Schwerpunkt liegt auf der charakteristischen topologischen Struktur, welche Zelluläre Automaten auch von anderen Konzepten wie Agenten-basierten Ansätzen oder allgemeinen Netzwerk Modellen abgrenzt.

Weiters wird unter der Bezeichnung Evolutions-Systeme eine Modifikation des Konzepts Zelluläre Automaten mit kontinuierlicher Zeit und Raum betrachtet. Dies liefert schlussendlich eine Verbindung zu stark stetigen Halbgruppen, abstrakten Cauchy Problemen und partiellen Differentialgleichungen.

Schließlich stellt ein stochastischer Formalismus diesen Modellierungsansatz als eine spezielle Variante eines Markov Prozesses dar und erlaubt aber gleichzeitig eine effiziente deterministische Methode für die Analyse und Simulation.

Abstract

Cellular Automata are being applied as a method for mathematical modelling and simulation in a variety of scientific areas. Usually a cellular automaton constitutes a microscopic description or abstraction of a natural system. The tools used for a mathematical derivation and validation of the cellular automaton modelling approach are however often optimised for specific natural systems. As a consequence there exists neither a dedicated mathematical formalism for cellular automata as a method for modelling and simulation nor general methods for model analysis and validation.

This thesis regards cellular automata as a distinct mathematical ansatz for modelling and contains a rigorous and systematic definition of various types of cellular automata. The aim is to develop methods and approaches for a general discussion and comparison of this modelling approach. Thereby the focus is less the introduction of a classification of cellular automata than the deduction and simplification of methods for analysis, validation and comparison. Additionally a systematic description of cellular automata can ease the communicability of the chosen modelling approach and provides a basis for further and more specific investigations.

The mathematical description of cellular automata in this thesis is based on locally defined update functions that operate in an iterative manner on graphs respectively regular lattices. A strong focus is put on the characteristic topological structure, which also delineates cellular automata from other concepts like agent-based modelling approaches and arbitrary network models.

Furthermore a modification of the concept of cellular automata with continuous time and space is introduced under the paradigm of evolution systems, which ultimately yields a connection to strongly continuous semigroups, abstract Cauchy problems and parabolic partial differential equations.

A stochastic formalism of cellular automata finally discloses this modelling approach as a very specific variant of Markov processes and allows efficient deterministic methods for analysis and simulation.

Preface

Sometimes mathematical concepts seem unnecessarily complex due to an extensive formalism. On the contrary, a mathematical description must be complete and systematic. In fact there is no mathematical benefit in a simplistic formalism which only allows an incomplete characterisation of an idea. This contradiction between complexity and completeness also applies to a mathematical formalisation of cellular automata. The basic principles of cellular automata are very simple and yet a mathematical definition of such systems is not.

Besides a purely descriptive point of view there may also exist an application oriented perspective, which imposes further requirements on the completeness of a mathematical formalism. If cellular automata shall not just exist but also be used to model natural systems, there must ultimately exist methods to validate such a modelling approach.

Fruitful investigation and successful application of a modelling approach also rely on the communicability of the underlying mathematical formalism. It is especially advantageous to have a common understanding of conceptions within a group of researchers. Additionally a written and established mathematical definition also provides a basis for further scientific investigations and teaching.

Sections 1.3 (Ontology and Classification) and 2.1 (Ordinary Cellular Automata) were written by myself in 2013 not exclusively for this thesis but within the scope of a combined attempt of the members of our research group 'Modelling and Simulation' under the direction of Felix Breitenecker to find a formal mathematical definition of ordinary cellular automata. Special thanks at this point for their work, constructive criticism and the allowance to use the results in my diploma thesis goes especially to Christoph Urach, Florian Miksch and Patrick Einzinger and in particular to Niki Popper for promoting the fundamental discussion of modelling approaches in general.

An intensive examination of the structural characteristics of a mathematical concept and cellular automata in particular inevitably exposes connections and relations to other mathematical concepts. Detailed formulation of such connections and their derivation can imply additional requirements on the mathematical formulation, which are not intuitive or obvious in the first point.

Consequently it was necessary to devote a certain effort to the reprocessing of mathematical theories like graph theory, strongly continuous semigroups, stochastic processes etc. in the context of cellular automata. Indeed an advanced investigation

on cellular automata with exclusively one specific mathematical theory in background would not be as interesting in the first point.

As usual, citations are indicated by quotation marks and a reference to literature. Longer quoted passages are left- and right-indented and offer a reference to literature at their end. Dots ... are used to indicate left out content and [square brackets] indicate modifications.

Definitions and propositions that were not originally made by myself are marked in their captions with corresponding references to literature. Definitions etc. that are used in a similar fashion in literature or are modified versions of statements from other sources are distinguished by the hint to *compare* with the respective source and a bibliographic reference. All other definitions, propositions and theorems were motivated by myself respectively in the relevant sections with the help of the members of our research group.

The bibliography is separated into literature that provides the required technical knowledge and literature that is – sometimes only briefly – referred to as application example.

This document was produced exclusively with open source software and in particular with LaTeX (including a variety of packages). All figures were created with *LibreOffice* and *GIMP* or generated with *GNU Octave*.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This diploma thesis investigates *cellular automata* mainly in the context of *modelling and simulation* but also – where necessary – as abstract *evolution systems* or stochastic processes.

The first aim is to provide a systematic mathematical approach to cellular automata, which allows us to use cellular automata as a mathematical technique for simulation (similar to differential equations or discrete network models etc.). Two important preconditions for allowing such a deployment are a common understanding of modelling and simulation and a consistent mathematical formalism of cellular automata.

The second main objective is the application of established mathematical methods and concepts in the context of cellular automata. This can also happen without any imminent or relevant outcome but may just serve as a basis for further investigations. This objective required that this thesis includes short introductions to – or rather basic definitions from – topics like graph theory (Section 3.2) or stochastic processes (Section 5.1) etc. at one point or another.

The most basic prerequirements of these principal tasks are treated within this chapter. Besides a very short introduction to modelling and simulation, this chapter also includes a survey on the history and present perceptions of cellular automata. Furthermore the basic concepts of cellular automata are discussed and examined under the paradigm of modelling and simulation in order to clear the way for a mathematical definition.

In Chapter 2 a mathematical formalisation of cellular automata as usually found in literature and a more general and abstract definition are presented. Chapter 3 is devoted to the characteristic topological structure of cellular automata. Chapter 4 presents continuous evolution systems as a super-class of cellular automata. An extended conception of cellular automata as stochastic processes is formalised and investigated in Chapter 5. The thesis is concluded with a summary and an outlook on further possible investigations.

1.1 Modelling and Simulation

Modelling and simulation is the discipline of applied mathematics, which deals with the abstraction of *natural systems*[1] and the application of any mathematical method in order to implement the resulting *conceptual model* [3] of a natural system in a mathematical fashion. The combination of an abstraction and a mathematical description is called *modelling approach*. The aim of simulating the resulting mathematical model is either prediction, that is the ability to predict system behaviour under simulated conditions, or validation of the abstraction and the modelling approach.

The applied mathematical methods are not bound to a certain area – like for example differential equations – but a modelling approach often incorporates methods from multiple mathematical areas at the same time. The combination, comparison and variation of methods and model assumptions in alternation with simulations and a following comparison with experimental data or measurements of the natural system allows us to improve the validity of a modelling approach. This process is sometimes simply called *modelling* or *comparative modelling and simulation* if multiple independent modelling approaches are compared. When a model describes the temporal evolution (not only a predefined temporal behaviour) of a system we talk of a *dynamic model*.

A very important subtlety of this discipline is the necessity of a strict differentiation between the natural system, the conceptual model and the mathematical implementation of the model. Starting from a given natural system, simplifications and assumptions are made in order to reduce the complexity of the system. This process yields an artificial system, which is an abstraction or approximation of the natural system and sometimes also referred to as a model of the natural system. In a second iteration this conceptual model is formulated in a mathematical fashion, which is either exact or allows the approximation of the model. The resulting *mathematical model* then is a mathematical implementation of the natural system and allows us to perform simulations.

There exists however a strong connection between the abstraction of a natural system and the mathematical technique to be applied. Of course certain mathematical approaches are suitable for certain types of problems. Furthermore the decision for a certain modelling technique is made concurrently to the abstraction of the system either due to the expertise of the operator or due to the lack of alternatives. Consequently the choice of a modelling technique always influences the abstraction of the system. Hence the result of these two steps in the process of modelling and simulation is called *modelling approach*.

Sometimes we have to deal with rather large or complex models and accordingly mathematical implementations that can not be treated or solved analytically. Often

[1] The term *natural system* was used by von Neumann [25]. Often simply *system* is used to denote the existing real or natural system which is to be modelled. Also the term *formulated problem* [3] can be used.

the mathematical implementation or formulation of a model either incorporates some kind of approximation like numerical solutions of equations or yields an iterative process per definition. The required solution or approximation procedure on a computer introduces a further iteration in the process of simulation and also a further decline in niceness.

The final result of a simulation is always compromised by multiple sources of errors. Starting from deliberate simplifications at the modelling stage and additional assumptions in the mathematical description, errors may also be introduced by an approximation method and numerical errors. Consequently, simulations with a mathematical model of a natural system can only be conducted after a validation of the modelling approach and an investigation on possible errors.

As this is not an essay on modelling and simulation, the reader is kindly referred to [3, 5] or similar literature for an extensive introduction.

1.2 History of Cellular Automata

The concept of cellular automata was formalised around 1950 "under the name of *cellular spaces*" [33] and is credited to Stanislaw Ulam and John von Neumann.

At that time Ulam was engaged with problems in pattern formation, biomathematics and chess [31]. He also worked on multiplicative processes, mathematical physics and especially on a numerical approach for solving nonlinear partial differential equations [10]. Von Neumann was developing a theory of self-reproduction [26]; alongside he published works on quantum mechanics, detonations, operator theory, artificial automata and other fields.

If one regards the fact that von Neumann and Ulam were deeply involved in the development of weapons of mass destruction at the same time, the idea of self-reproducing machines – on which von Neumann was later working in the context of artificial automata – gets a grotesque and antiquated apocalyptic taste.

However, partially based on a suggestion of Ulam, von Neumann formalised his ideas on self-reproduction as a formal automaton operating on discrete cells [26]. Accordingly a cellular automaton was originally a formal system which served as a testing environment for ideas on self-replication [33].

On the other hand, at this point von Neumann already tried to predict application scenarios for his *theory of logical automata* and also weakened the account of formal logic:

> In fact, there are numerous indications to make us believe that this new system of formal logic will move closer to another discipline which has been little linked in the past with logic. This is thermodynamics, primarily in the form it was received from Boltzmann, and is that part of theoretical physics which comes nearest in some of its aspects to manipulating and measuring information. Its techniques are indeed much more analytical than combinatorial, which again illustrates the point that I have

> been trying to make above. ... that a detailed, highly mathematical, and more specifically analytical, theory of automata and of information is needed. [25]

This is a very interesting statement especially in the context of this thesis. It shows that von Neumann recognised the implication of logical automata for simulating physical phenomena and noticed the accompanying necessity of a mathematical formalism.

He also made the following structural observation, which can be seen in connection with *microscopic* and *bottom-up* modelling approaches and comes rather close to what nowadays is understood as cellular automaton:

> The natural systems are of enormous complexity, and it is clearly necessary to subdivide the problem that they represent into several parts. One method of subdivision, which is particularly significant in the present context, is this: The organisms can be viewed as made up of parts which to a certain extent are independent, elementary units. We may, therefore, to this extent, view as the first part of the problem the structure and functioning of such elementary units individually. The second part of the problem consists of understanding how these elements are organized into a whole, and how the functioning of the whole is expressed in terms of these elements. [25]

Not only von Neumann was interested in simulating natural systems by dividing the system into interacting subsystems. A few other and even earlier examples are sketched below:

- The Ising model as presented in a 1925 paper (which despite its age and impact is not freely available).
- In 1946, Norbert Wiener and Arturo Rosenblueth published a work on the conduction of nervous impulses [47], which sometimes is explicitly not regarded as a cellular automaton and in other occasions interpreted as the first application of von Neumanns cellular automata [32] (despite possible chronological conflict).
- "A one-dimensional dynamical system of 64 particles with forces between neighbors containing nonlinear terms has been studied on the Los Alamos computer MANIAC I." [10] by Ulam in 1955.

The origins of cellular automata are not exclusively in the formal logic domain, but cellular automata and similar approaches were used to simulate natural systems even before the connotation of cellular automata was coined. It is even more surprising that in the present perception cellular automata are usually not discussed as a general approach for modelling and simulating natural systems. Even though

there exist very prominent cellular automaton approaches like the lattice Boltzmann method.

The ultimate "rise" of cellular automata in the 1960s and 1970s is clearly linked to the big advances in computer science of this time. Computers became available to a broad community of researchers and lots of experiments and implementations were conducted. Cellular automata inspired new theories (p.e. Konrad Zuse [51]) but also served as benchmarks in computer engineering (John Conway's game "Life" or "Game of Life"). In 2002 with "A New Kind of Science" [50] Wolfram delivered a complete classification of the sub-class of *elementary cellular automata*.

1.2.1 Present Major Consensus on Cellular Automata

From literature we can conclude that there exist neither a unique nor a formal mathematical definition of cellular automata, no clear assignment of cellular automata to a scientific domain or a classification of cellular automata as a scientific method. Also the history of cellular automata itself seems to be blurry and prone to varying interpretations.

> Even if the axioms are chosen within the common sense area, it is usually very difficult to achieve an agreement between two people who have done this independently. [26]

This section contains some short definitions of cellular automata from various sources and also serves as a first technical introduction to cellular automata. It is very clear that the basic concepts (see Section 1.3.1) usually coincide. Differences appear mostly at a rather high level of detail.

> Cellular automata can be viewed as a simple model of a spatially extended decentralized system made up of a number of individual components (cells). The communication between constituent cells is limited to local interaction. Each individual cell is in a specific state which changes over time depending on the states of its local neighbors. The overall structure can be viewed as a parallel processing device. However, this simple structure when iterated several times produces complex patterns displaying the potential to simulate different sophisticated natural phenomena. [12]

> A cellular automaton consists of a regular grid of cells, each in one of a finite number of states, such as on and off (in contrast to a coupled map lattice). The grid can be in any finite number of dimensions. For each cell, a set of cells called its neighborhood (usually including the cell itself) is defined relative to the specified cell. An initial state (time $t = 0$) is selected by assigning a state for each cell. A new generation is created (advancing t by 1), according to some fixed rule (generally,

> a mathematical function) that determines the new state of each cell in terms of the current state of the cell and the states of the cells in its neighborhood. Typically, the rule for updating the state of cells is the same for each cell and does not change over time, and is applied to the whole grid simultaneously, though exceptions are known, such as the probabilistic cellular automata and asynchronous cellular automaton. [48]

> Cellular automata are mathematical idealizations of physical systems in which space and time are discrete, and physical quantities take on a finite set of discrete values. A cellular automaton consists of a regular uniform lattice (or 'array'), usually infinite in extent, with a discrete variable at each site ('cell'). The state of a cellular automaton is completely specified by the values of the variables at each site. A cellular automaton evolves in discrete time steps, with the value of the variable at one site being affected by the values of variables at sites in its 'neighborhood' on the previous time step. The neighborhood of a site is typically taken to be the site itself and all immediately adjacent sites. The variables at each site are updated simultaneously ('synchronously'), based on the values of the variables in their neighborhood at the preceding time step, and according to a definite set of 'local rules'. [33]

> [Cellular automata] can be characterised as follows [compare the definitions of Wolfram and Hedrich]:
>
> - [Cellular automata] are regular arrangements of single cells of the same kind.
> - Each cell holds a finite number of discrete states.
> - The states are updated simultaneously ('synchronously') at discrete time levels.
> - The update rules are deterministic and uniform in space and time.
> - The rules for the evolution of a cell depend only on a local neighborhood of cells around it.
>
> Not all of these criteria are always fulfilled. The cells can be positioned, for example, at the nodes of a (quasiperiodic) Penrose lattice (Penrose, 1974, 1979) or at random (Markus and Hess, 1990). A random connection of cells was proposed by Richard Feynman (Hillis, 1989). The update rules of certain CA include probabilistic elements ... [32]

In Section 1.3 we will filter out a set of features which for us make up a *default* or *ordinary cellular automaton*.

1.2.2 Classification of Cellular Automata

None of the definitions in Section 1.2.1 excludes cellular automata from being a (bottom-up) modelling approach. On the other side there exists no mathematical framework – like we are used to from differential equations for example – which renders cellular automata a fully functional mathematical method for describing a natural system. Moreover cellular automata are often regarded as (natural) systems themselves. In some occasions these iterative systems are then interpreted as models of real existing phenomena or – even worse – they are for example entitled as being thermodynamic models when they feature concepts comparable to conservation of mass or momentum.

In my opinion – and especially in the context of modelling and simulation – this inverse approach of modelling is highly critical not only because an existing abstract system is considered first and a suitable natural system is seeked for "application" afterwards, but also because the natural bottom-up approach of cellular automata as a method for modelling is abandoned.

A possible way to avoid this clash of conceptions is to differentiate between cellular automata in formal logic and cellular automata in modelling and simulation. The first branch mainly deals with pattern formation (self-reproduction, reoccurency, etc.), Turing completeness, ergodicity, etc. where the second branch aims at modelling of natural systems in a bottom-up fashion.

Of course there exist scenarios in which results from the formal logic branch (e.g reversibility, ergodicity, etc.) are essential for simulating a system or where elementary cellular automata are perfectly suitable for simulating a dynamic system.

Further characteristics that can help to differentiate between these two branches:

- A cellular automaton which is merely a computer program and does not serve as a modelling approach is settled in the formal logic branch.

- Comparisons and classifications based on update rules and cell states happen in the formal logic branch. This concerns especially the classification of update rules for one dimensional elementary cellular automata as done by Wolfram [50].

- Comparison and identification of a cellular automaton with a differential equation happens in the modelling and simulation branch.

1.3 Ontology and Classification

As mentioned in the preface, this section originated prior to this thesis as an introduction to (ordinary) cellular automata in modelling and simulation and is included here in a slightly modified version. The discussion and schematic classification of cellular automata in Section 1.3.2 received a major overhaul.

1.3.1 Basic Concepts of Cellular Automata

The ontology and basic concepts of cellular automata may seem rather obvious. Nevertheless the conception of cellular automata is subject to experiences and approaches (compare Section 1.2.1). This section incorporates ideas from [12, 32, 33] and others and gives an overview of the basic concepts of cellular automata without sticking to one single or even clear mathematical definition but from a rather philosophic and conceptual point of view.

It should be mentioned that for a yet outstanding formal mathematical definition it is even more important to restrict the default case of cellular automata (*ordinary cellular automata*, as we will call them) to a well reasoned subset of cellular automata or entity-based dynamic systems. Deviations from this basic definition can then be discussed and formalised separately from the default definition.

Cell. The term *cell* comes from the Latin word *cella* which means chamber, small room, compartment [48]. In the context of cellular automata the term cell evolved from the idea of enclosed entities and is a pure historical notion. Other terms describing the same idea are:

site. Used by Wolfram for example [50, 33].

node, lattice node. Used especially in connection with lattice gas cellular automata or the lattice Boltzmann method [32].

discrete coordinate, raster element. In connection with rasterised geographic data [42].

pixel, dot. Anti-aliasing or image-processing in general.

element, field, entry. Matrix or array notion.

vertex. Compare Section 3.2.

individual. Used as adjective or in the context of sociological or demographic models [36].

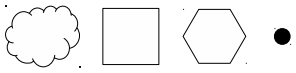

Figure 1.1: Different perceptions of a cell. A cell can represent an entity, a geometric part of a space like a square or a hexagon or a discrete coordinate.

A very low-level and subordinate term is *atom* or *entity*, which emphasise the consideration of a multitude of basically equal identifiable objects (also called homogeneity). A cell in the sense of cellular automata is a passive entity, whose (internal) properties are accumulated in a so-called *state*.

Cell-Space. We usually deal with entities that are arranged in a regular fashion (often spatial) resulting in a *grid* or *lattice* structure. Other terms are *cellular space*, *tessellation structure*, *homogeneous structure*, *cellular structure*, *iterative array* [32], *arrangement* or *alignment*.

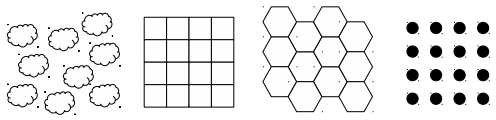

Figure 1.2: The alignment of cells among each other yields a so-called cell-space.

The idea of an *alignment* yields the introduction of a more or less coarse distance measure on the *domain* on which the cells are arranged. This motivates the term *space*. We usually deal with *regularly aligned* cells.

Neighbourhood. The concept of neighbourhood-relations between entities is one of the – if not the – fundamental feature of cellular automata.

We can distinguish between two main approaches. The first and obvious approach is a geometric or spatial one. The following terms fit into this concept: *adjacent*, *distance*, *neighbouring*, *proximity*. These concepts are based on a pre-existing space or domain which exhibits a distance measure.

A more general approach could be described by *peer-group*, *contact*, *related*, *connected*. In this case the idea of a cell-space or domain becomes obsolete and the neighbourhood is not necessarily of spatial character.

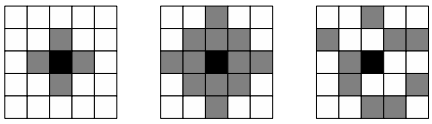

Figure 1.3: The neighbourhood of a cell can be based upon the distance between cells or defined through relative positions.

Anyhow, the placement of a cell and its neighbourhood is not part of the cell-state but an external feature, which is determined by its alignment among other cells. A cell itself is generally not aware of its alignment among the other cells.

State-Space. Other words for *state* are *condition* or *value*.

A *state-space* is a purely mathematical construction which facilitates the formal description or definition of update rules. The state of a cell is an element (e.g. integer, vector, real number, etc.) of the state-space. We can distinguish discrete and continuous states.

Since all cells are equal, they share the same state-space. Different *types* of cells can be distinguished for example by using a measure or an additional dimension in the state-space.

Update Method. The neighbourhood of a cell or more precisely the states of the cells in the neighbourhood influence the state of the cell itself. An update *rule*, *mechanism*, *algorithm*, *function*, *process* is applied on the states of the cells in the neighbourhood and delivers a new state for the cell.

Through update methods *interaction* between cells is defined.

Global State and Evolution. The collection of the states of all cells forms a *global state* or *configuration* of the cellular automaton. Accordingly we can describe a cellular automaton by its global evolution, which is composed of an iteration of global states.

The update rules can be applied in an *iterative* manner on all cells *simultaneously* and accordingly produce discrete "time" steps (*synchronicity*). It is however not necessary that the resulting *iterative process* describes a temporal *evolution*.

An important requirement is that this process is *memoryless* in the following sense: The state of a cell at the i-th iteration level only depends on the states of other cells in the $(i-1)$-th iteration level. This property allows the simultaneous application of the update method on all cells.

Border. An important topic is the treatment of cells, which compose the border or which are located at the border of a cell space. Since those cells exhibit an *altered* or *degraded neighbourhood*, there must exist mechanisms in the update rules which catch these instances and apply an appropriate special update rule. Such mechanisms are collected under the term *boundary conditions*.

A so-called border-cell is not a special type of cell. It is only distinguished by a degraded neighbourhood and the according update rule which depends on the structure of the neighbourhood. Nevertheless sometimes it may be useful to introduce a special cell state which indicates that a cell is a certain type of border-cell. For example lattice gas cellular automata and the lattice Boltzmann method use in- and outflow cells, whose state is altered in a different way than for ordinary cells. But these types of cells are not necessarily located at the border of a cell-space.

Common types of boundary conditions are:

periodic boundary conditions. A not occupied slot in the neighbourhood of a cell is filled with a cell from the opposite side of the lattice.

reflective boundary conditions. A not occupied slot in the neighbourhood of a cell is filled with a copy of the cell.

null boundary conditions. An empty slot leads to a different update mechanism.

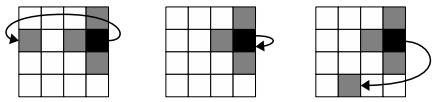

Figure 1.4: Different types of boundary conditions: Periodic, reflective and arbitrary.

Cellular Automaton. The term *automaton* originates from the Greek *autos* meaning *self*. Derived concepts include self moving, self willed, acting of one's own will, "in form of predefined reactions without interventions or external decisions", "a machine or robot designed to follow a precise sequence of instructions", a formal system [48].

In the context of history of arts automata are mechanic artwork built by goldsmiths and/or scientists. Their purpose was to simply just entertain and impress with their complex mechanics as well as to model phenomena from natural sciences[2] (clocks p.e.), which made them important but expensive instruments for daily life [38].

Automaton theory is a field of theoretic computer science which deals with abstract machines and automata. Some special cellular automata can be interpreted in this context and resemble to *finite state machines* [48]. We treat them as *cellular automata in formal logic*.

An automaton in the context of cellular automata is accordingly an automaton operating on cells.

1.3.2 Textual Definition of Ordinary Cellular Automata

The introduction of a mathematically correct formalism inevitably leads to a classification system – this is true not only for cellular automata. Once a formalism is created, such mathematical differentiations are usually very precise and clear.

A conceptual classification, which happens before the mathematical definition, on the other hand is not necessarily unique and exact. As mentioned before, subjective perceptions play an important role in defining cellular automata. One person may require that the cells are aligned on a lattice structure, another may prefer cells as abstract entities without a location.

The task of a formal definition approach is not only to deliver a basic definition for certain types of cellular automata, an elaborate formal definition also considers and embraces deviations from the basic definition.

Accordingly *ordinary cellular automata* are defined as a standard type of cellular automata which is compatible with (or maybe the average of) definitions found in literature. *Stochastic cellular automata* or *unaligned cellular automata* etc. can be formalised as extensions of this basic mathematical definition.

[2] Regard the analogy to the difference between artificial systems like the "Game of Life" and the use of cellular automata for modelling.

The first requirement for the definition of ordinary cellular automata was already mentioned:

Compatibility with Present Perceptions. In Section 1.2.1 an overview of textual definitions of cellular automata in literature was given. It is obviously not possible to find a basic definition that is fully compatible with every other definition.

Some features that are usually contained in definitions of cellular automata are

- a set of regularly aligned (lattice) cells as containers for a state,
- a finite state space, assignment of a state to each cell,
- locality of interaction, the concept of regular neighbourhoods,
- an update rule-set (implements boundary conditions if necessary) and
- iteration and synchronicity.

A Method for Modelling and Simulation. As we intend to use cellular automata as a simulation method or modelling approach it is important to achieve some kind of universality.

It is not a good idea to exclude continuous states from a default cellular automaton. If we look at lattice Boltzmann models we can see that this prototype of an advanced "applied" cellular automaton features continuous densities as the state of a cell. When we model a certain spatial system we should be able to use arbitrary boundary conditions. Furthermore we may want to observe systems in which some cells behave differently from others. We may also not exclude any specific type of update function from a default definition.

Furthermore a good technique for modelling features methods for model validation, parameter identification, error estimation (errors by applying the modelling technique as well as errors during simulation) and comparison with other modelling approaches. For this to be accomplished it is necessary that a formal definition is mathematically consistent and complete but also extendible at the same time.

Automatisation. We may never forget that cellular automata are automata operating on cells.

An automaton follows a predefined scheme of operations in an iterative manner. An automaton is also the aggregation of a complete and precisely defined set of rules [48]. To meet this rough definition, it is necessary that a cellular automaton

- performs the exact same update mechanisms for every cell and iteration and
- uses a neighbourhood of the same structure (coll. the same neighbourhood) for every cell and iteration.

- Furthermore the rule set must be compatible with any possible state of the cellular automaton (no dead locks).

Usually these features are constructed in a rather simple manner following the paradigm *simple individual behaviour leads to complex global dynamics* [12, 32, 33].

Nevertheless it can happen that degradations of these specifications are required. For example, a cellular automaton does usually not allow the modification of update rules during simulation. On the other hand, a more general entity-based iterative system could allow external modification of the rule set during simulation. Or it might be necessary that the update rules depend on the number of the current iteration or on the progression of a cells state.

Also the complete topology of a cellular automaton is usually constant for all iterations. Cells can neither change their position nor their neighbours. Theoretically it would however be possible to define a very complex rule set in order to simulate varying neighbourhoods by applying different update functions, which neglect certain neighbours under certain circumstances.

The only true variable of a cellular automaton should be the states of the cells respectively the state mapping. Accordingly only the state of a cell may change over iteration i.e. *invariance of update rules and topology.*

A Bottom-Up Modelling Approach. A cell can be regarded as an identifiable container for a state without any interior dynamics (lattice gas cellular automata not necessarily constitute an exception) or knowledge about its location among other cells.

Cellular automata are used to model dynamic systems in a bottom-up fashion. The global dynamics arise from simple individual changes of the cells' states. These changes are straight forward and individual, which means that there is no intended feedback from the global dynamics to the behaviour of a single entity.

Usually the typical individual behaviour of a single entity is known and the modelling approach of cellular automata permits to observe the arising global dynamic behaviour.

It is very important that a cell on its own is not aware of other cells or its location among other cells. Furthermore a cell is never an active agent, but rather just a container for a state.

The Concept of Sets is Fundamental. As in mathematics itself also for cellular automata a set theoretic point of view can be used as a starting point for setting up a mathematical description.

Rationale: A cellular automaton always deals with a set of cells, a set of states and sets of neighbouring cells regardless of the way these cells are arranged among each other or within a domain. Every approach for formulating the concepts of cellular automata incorporates this fundamental idea but introduces further, more sophisticated concepts like a lattice of cells or borders and boundary conditions.

A *set-based* point of view is also a good starting point for comparing different specifications of cellular automata with other modelling techniques like agent-based models or networks.

Alignment of Cells vs. Discretisation of Space (Topology I). From a set-based point of view the alignment of cells is a secondary feature, which is deployed on a previously existing set of cells (*a posteriori alignment*). This might not always correspond to the idea of a system which is to be modelled using cellular automata. A very common application scenario of cellular automata features the discretisation or partitioning of space into cells (*a priori alignment*). In this case cells can be regarded a result of discretisation and a spatial interpretation of the cellular automaton is self-evident.

Usually discretisation is performed in a manner that the resulting discrete entities are more or less equal. The advantages concerning formulation of functions and building algorithms (implementation) seems rather obvious. Note that for example the finite element method uses an irregular discretisation in a very successful manner. In the case of a regular discretisation the geometry of the individual cells is not very important. The perception of a discretised space can be replaced by a regular alignment of cells or entities.

Wherever an alignment originates from, it is a very important characteristic of the topology of a cellular automaton. It is the *first topological feature* of the cellular automaton and in the following also denoted as *topology I*. Also if the alignment of cells is not of spatial character, relative proximities between cells are introduced.

In a straight forward mathematical fashion a regular alignment can simply be described using indices from an arbitrary dimensional index set $I \subset \mathbb{Z}^d$. This approach will be called *index-based* or *lattice* approach.

Locality (Topology II). For a model or system it is often a crucial feature that information travels through space and time. The restriction of interaction (at discrete iterative steps) to a (local) subset of entities delays the spread of information and favours an inhomogeneous distribution of information. The set of cells that can be influenced by the information contained in a cell within a certain amount of time (usually one time unit) constitutes the concept of *neighbourhoods*.

From a different point of view, the neighbourhood of a certain cell consists of those cells for which there exists at least one condition (global state) under which an altered cell-state yields a different state for the actual cell in the next time step. Compare this description with the concept of differentiation.

This characteristic will be called the *second topological feature* of cellular automata or also simply *topology II*.

In the case of (regularly) aligned cells the neighbourhood of a cell is often symmetric and composed of adjacent cells. In other words, the neighbourhood of a cell can often be described using a metric on the cell-space i.e. based upon the first topological feature.

For regularly aligned cells and especially for an index-based alignment it is also possible to use the concept of translations to describe the neighbourhoods of the cells. In the index-based approach the result is a stencil method, which for a given index delivers through addition and subtraction other indices relative to the first index. The use of relative indices permits to define neighbourhoods of arbitrary structure like non-symmetric or even non-local neighbourhoods.

Usually the neighbourhoods of all cells are uniformly structured (number of neighbours, relative positioning of neighbours).

This second topological feature can also exist independently from the first topological feature. If the cells are not aligned, it is still possible to explicitly define "neighbours" for every cell.

Update. An update rule takes the states of the neighbours of a cell as input and delivers a new state for the cell.

This mapping is not necessarily reversible i.e. the states of the cells at time t can not be concluded from the states of the cells at time $t+1$. Consequently in addition to deterministic update rules also stochastic methods are possible.

An update rule may be a very complex composition of multiple functions and mappings or a simple linear function. However usually the ordering of the input arguments influences the result. This means that changing the ordering of the neighbours of a cell influences the state of the cell in the next iteration.

An update rule must be defined for all occurring sizes of neighbourhoods and all possible state-configurations of the cells in the neighbourhoods.

Borders and Boundary Conditions (Topology III). When modelling a spatial system or a physical body or continuum there usually exist boundaries of the observed domain. Accordingly a systematic definition approach for cellular automata must offer a direct method for formulating boundary conditions. Additionally the geometry of a cellular automaton must not be restricted to simple cuboid geometric shapes.

Especially in connection with cell-space and border we must never forget that a cellular automaton is – at least for us and our basic definition – an approach for modelling a dynamic system and not the implementation thereof. Mathematical models often introduce infinite domains but the implementation as a cellular automaton with a computer language must usually use a finite set off cells probably in connection with periodic boundary conditions. Of course a good mathematical model takes possible artefacts and limitations which arise at the implementation level into account.

It is possible to implement a cellular automaton with a dynamically growing cell-space. The size is then limited by the physically available memory.

Using the index-based approach, a border cell can be recognised by a degraded neighbourhood. That means that the relative indices representing the neighbourhood point to nonexistent cells. In this case there must either exist a method or rule

to treat this situation within the update rules or the missing neighbours have to be replaced with any other existing cell, which yields a distortion of the geometry of lattice of cells. If there exits such a distortion of a regular alignment, a *third topological feature* (also *topology III*) is introduced.

Modelling is not equal to Implementation. As mentioned before, cellular automata are regarded as an approach for modelling a (often spatial) dynamic system. It is however useful if the mathematical implementation of a modelling approach allows an as straight forward as possible implementation in a programming language. Since programming languages usually rely on the concept of arrays (indexed sets of entities) to store information, the index-based approach for describing a cellular automaton based model seems quite ideal in this context.

The definition approaches in this thesis are however not intended or especially suitable for describing the arithmetics of the implementation of a cellular automaton in a programming language. On the other hand this does not exclude the "Game of Life" for example from being described using the presented definition approaches.

Relation to Differential Equations. The reader may have noticed that cellular automata as a modelling approach were previously at several locations compared to differential equations. There are two aspects of this comparison.

In some publications [32, 33] cellular automata are treated or at least discussed as being an equivalent methodology as differential equations – at least for a certain type of problem. In order that this is possible in the first place, a systematic formalism of cellular automata is necessary.

Secondly the resulting algorithmic implementation of the discretisation of differential equations (finite differences p.e.) or even other more or less discrete approaches (finite element method) can be interpreted as cellular automata.

Relation to Agent-Based Systems. For an introduction to agent-based modelling see for example [39]. Also agent-based modelling has to be seen as a modelling approach rather than a modelling technique. The analysis, derivation or even technical implementation of an agent-based model can be based upon a multitude of different mathematical techniques like graphs, Markovian models [35], statistical models or differential equations and even cellular automata [36].

A very important difference between an agent-based modelling approach and a cellular automaton modelling approach are the clear specifications of cellular automata regarding the topological features (alignment and local interaction) and the automaton concept (Section 1.3.2), which includes passivity of the cells, memorylessness and uniform simple update rules.

Conclusion. Cellular automata in general are iterative entity-based dynamic systems. According to the previous discussion, the most important characteristics of

a default type of a cellular automaton in modelling and simulation are summarised below.

- cells are regarded as discrete, equal and passive entities
- characteristic topological features
  - topology I: regular alignment of cells (regardless of a priori or a posteriori), mathematically realised through an indexing
  - topology II: uniform and ordered neighbourhoods, implemented through index translations
  - topology III: possible distortions of the lattice
- an arbitrary set of states is allowed
- update rules
  - independent of the cell
  - calculate a new state given a tuple of states
  - memoryless
  - defined for all occurring configurations/states of neighbourhoods
- iteration scheme
  - synchronous alteration of states through locally applied update rules
  - invariance of topology and update rules
  - requirement of an initial condition

This choice of features and properties or ordinary cellular automata is the result of a discussion among members of our research group. A mathematical formulation implementing this concept of ordinary cellular automata can be found in Section 2.1. Of course different definitions of the prototype of cellular automata are possible.

Chapter 2

Basic Definitions of Cellular Automata

In this section mathematical definitions and formalisms for cellular automata are presented.

Due to fact that *ordinary cellular automata*, as described in Section 1.3.2, are regarded as the prototype of cellular automata, a straight forward and intuitive mathematical formalism for this class is presented first. The concept of an alignment on a lattice structure is implemented through an indexing of the cells. Despite the fact that the formalism of ordinary cellular automata is a self-contained mathematical definition, this index-based approach is rather complicated in the end.

In the second section of this chapter a class of cellular automata that lacks an alignment of the cells – namely *unaligned cellular automata* – is defined. It is clear that ordinary cellular automata are unaligned cellular automata with an additional special alignment – implemented by an indexing of the cells.

An alternative approach for a regular alignment is derived and discussed in Chapter 3 and finally recapitulated in Section 3.4.

2.1 Ordinary Cellular Automata

This mathematical formalism of ordinary cellular automata is based upon the definition derived in Section 1.3.2 and the results of an extensive reviewing by members of the research group 'Modelling and Simulation'.

This mathematical definition of the default type of a cellular automaton should be perspicuous without studying the derivation of the conceptual definition from Section 1.3.2.

2.1.1 Formalisation of Concepts

The best way for setting up a mathematical formalism is a bottom-up approach itself. In a first step the basic concepts of ordinary cellular automata are formalised

then the resulting definitions are used to describe ordinary cellular automata as a composition thereof.

2.1.1.1 Cells

The very basic requirement for defining cellular automata is a set of abstract entities called cells.

Definition 2.1.1 (Cell, Set of Cells)**.** Let M denote the *set of cells*. Consequently a *cell* is a – necessarily – unique element $m \in M$. The set of cells is not empty but countable and most often finite.

There is basically no limitation in the number of cells. However the observation of only one cell defeats the purpose of cellular automata and we then speak of a *trivial cellular automaton*. On the other hand in some occasions infinite or noncountable sets of cells can be problematic. The cardinality of the set of cells may be characterised through terms like *finite*, *countable*, *infinite*, etc.

Note that the implementation of infinite cellular automata on a computer system arises a lot of questions. There exists however the possibility to simulate an infinite domain by using periodic geometries as discussed in Section 2.1.1.4. The errors introduced by applying this technique must be taken into account when analysing the validity of the underlying model.

2.1.1.2 Alignment of Cells (Topology I)

In order to align cells in a regular – not necessarily geometric, despite there obviously exists a geometric interpretation – fashion, we index the set of cells using a set of indices, which is subject to certain requirements.

Definition 2.1.2 (Connectivity)**.** A subset I of $\mathbb{Z}^d$ is called *connected* if one of the following conditions is satisfied.

(i) For each two elements $a, b \in I$ there exists a series of elements $(z_\alpha)_{\alpha\in\mathbb{N}} \subset I$ for which $\|z_\alpha - z_{\alpha+1}\|_2 = 1\ \forall\alpha \in \mathbb{N}$ and for which $\exists\alpha_a, \alpha_b \in \mathbb{N}$ such that $a = z_{\alpha_a}$ and $b = z_{\alpha_b}$.

(ii) Analogously to (i) with the condition $\|z_\alpha - z_{\alpha+1}\|_2 \leq \sqrt{2}$ or $\|z_\alpha - z_{\alpha+1}\|_2 \leq \sqrt{d}$ etc.

(iii) Each cross-section

$$I_{(i_1,\ldots,i_{l-1},i_{l+1},\ldots,i_d)} = \{j \in \mathbb{Z} : i = (i_1,\ldots,i_{l-1},j,i_{l+1},\ldots,i_d) \in I\} \qquad (2.1.1)$$

where $i_\alpha \in \mathbb{Z}, \alpha \in \{1,\ldots,d\} \setminus l$ is the finite union of intervals of the whole numbers or the empty set.

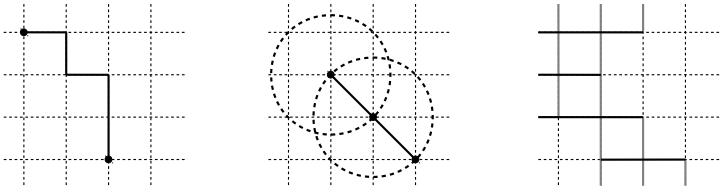

Figure 2.1: Different concepts for connectivity as defined in Definition 2.1.2 (i) to (iii) are sketched from left to right.

In the context of cellular automata another definition of connected may require that for an arbitrary cell each other cell can iteratively be reached through the neighbourhoods of the cells neighbours (see Definition 3.0.15).

Definition 2.1.3 (Index Set, Lattice)**.** We call a connected subset $I \subseteq \mathbb{Z}^d$, $d \in \mathbb{N}\backslash\{0\}$ an *index set*. We also call the index set of a cellular automaton a *lattice*. In this case cells correspond to lattice nodes.

In literature there exists a definition of *lattices*, which can be approached in an order-theoretic or an algebraic fashion using a join- and meet-operator. A lattice as defined in Definition 2.1.3 is a special case of this definition.

The idea of a connected index set is used to somehow avert trivial or eccentric cellular automata with multiple independent cellular automata on one lattice or cellular automata with a scattered or sparsely occupied lattice. All these situations can be excluded from ordinary cellular automata. In order to describe the character of the index set – respectively the geometry of a cellular automaton – in an even more detailed manner we can transfer concepts like *symmetric*, *convex*, *connected*, etc. from vector spaces respectively topological spaces to index sets. If cells represent the discretisation of a continuous space, topological concepts like *path-connectivity* or *simple connectivity* can be used directly. However since the index set describes the geometry of the cellular automaton it is defined at the stage of modelling. Accordingly the previous definition of connected may be obvious in some cases or to weak in other cases and is more the introduction of a concept than an exact definition.

Example 2.1.4. A typical square lattice is described by the index set $I = (1, \dots, n_1) \times \cdots \times (1, \dots, n_d)$. In this case the index set is of course connected.

Definition 2.1.5 (Index Mapping)**.** If M is a set of cells, we call M *indexed* or *regularly arranged* if there exists a bijective mapping $\mathcal{I} : M \to I : m \mapsto \mathcal{I}(m) =: i$ between M and an index set I. We call $\mathcal{I}$ an *index mapping* and also use $\mathcal{I}$ for mapping tuples of cells onto tuples of indices $\mathcal{I} : M^k \to I^k : (m_1, \dots, m_k) \mapsto$

$(\mathcal{I}(m_1), \ldots, \mathcal{I}(m_k)) =: (i_1, \ldots, i_k)$ where $k \in \mathbb{N} \setminus \{0\}$. If all cells are arranged or indexed using an index set $I \subseteq Z^d$, we call d the *dimension* of the cellular automaton.

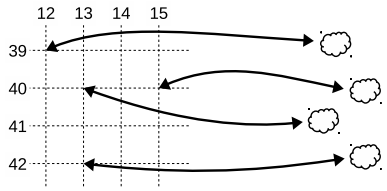

Figure 2.2: The index mapping is a bijective mapping between I and M.

The index mapping is the same for all iterations. Thus the arrangement of cells is constant and a cell does not change its location. The combination of an index set and an index mapping may be called *indexing*.

The reason for a differentiation between the actual cell $m \in M$ and its unique index $i \in I$ is in the first place purely conceptual. It however allows a more systematic definition and classification with respect to unaligned cellular automata (Section 2.2).

2.1.1.3 Neighbourhood (Topology II)

The concept of indices and (sorted) tuples can be used to define the neighbourhood of a cell. Beforehand some operations on indices have to be discussed. Throughout this section we let $k \in \mathbb{N} \setminus \{0\}$ be the (uniform) number of neighbours.

Definition 2.1.6 (Relative Index Tuple)**.** A tuple $J := (j_1, \ldots, j_k) \in (\mathbb{Z}^d)^k$ where $j_\alpha \neq j_\beta$ for $\alpha \neq \beta$ is called a *relative index tuple*.

For $i \in \mathbb{Z}^d$ the addition respectively subtraction $J \pm i := (j_1 \pm i, \ldots, j_k \pm i)$ is well-defined.

Definition 2.1.7 (Index Translation)**.** Given a relative index tuple J we define the *index translation* $\mathcal{T}_J$ of an index i from an index set I by $\mathcal{T}_J : I \to (\mathbb{Z}^d)^k : i \mapsto i + J$ and call the result an *absolute index tuple* or just *index tuple*.

Note that $i + J$ is not necessarily a subset of I.

The differentiation between absolute and relative index tuples depends on the application of the tuple of indices. A relative index tuple is used to specify the indices of other (neighbouring) cells relative to the index of one specific cell. An absolute index tuple on the other hand just contains the indices of a sorted set of cells.

Definition 2.1.8 (Neighbourhood)**.** For a cell m_i from an indexed set of cells with index set I and a relative index tuple J, we use the absolute index tuple $\mathcal{T}_J(i) =$

$(i_1, \ldots, i_k) \in (\mathbb{Z}^d)^k$ to define the *neighbourhood* of m_i as the tuple of cells $N_{m_i,J} := (n_1, \ldots, n_k) \in (M \cup \{\emptyset\})^k$ where

$$n_\alpha := \begin{cases} m_{i_\alpha} = \mathcal{I}^{-1}(i_\alpha) & i_\alpha \in I \\ \emptyset & i_\alpha \notin I \end{cases} \qquad \alpha \in \{1, \ldots, k\} \tag{2.1.2}$$

Furthermore we call k the *size of the neighbourhood*.

The mapping $\mathcal{I}^{-1} : (\mathbb{Z}^d)^k \to (M \cup \{\emptyset\})^k : (i_1, \ldots, i_k) \mapsto (n_1, \ldots, n_k)$ is the generalised inversion of $\mathcal{I}$. The *nonexistent cell* $\emptyset$ is required in order to maintain the original tuple structure of the neighbourhood and to be able to indicate that indices, which are outside the index set, do not refer to a cell.

Definition 2.1.9 (Characterisation of Neighbourhoods)**.** Neighbourhoods can be characterised in different ways, among them:

(i) A cell lies in its own neighbourhood if and only if $0 \in \mathbb{Z}^d$ is part of the relative index tuple. This kind of neighbourhood is called *reflexive*.

(ii) The neighbourhood relations are not necessarily *symmetric*, which means that if cell m is in the neighbourhood of cell n, cell n is not necessarily part of the neighbourhood of cell m. Especially for ordinary cellular automata symmetry is a geometric feature.

(iii) It is not unusual that neighbourhoods are of *local* character, which means that the neighbourhood relation is defined by the distance between cells. In this case, the index translation can be replaced by the corresponding function $\mathcal{T}_{\text{metric}}$.

Proposition 2.1.10 (Universality of the Relative Index Approach)**.** *Given a local neighbourhood structure defined through a metric on I by $\mathcal{T}_{\text{metric}} : I \to I^k$, it is possible to find an equivalent relative index tuple J for which the index translation $\mathcal{T}_J$ delivers the same neighbourhood structure.*

Definition 2.1.11 (Neighbourhood Mapping)**.** Accordingly for an indexed set of cells $(M,I,\mathcal{I},\mathcal{I}^{-1})$ and an index translation $\mathcal{T}$ the *neighbourhood mapping* is defined by $\mathcal{N} := \mathcal{I}^{-1} \circ \mathcal{T} \circ \mathcal{I} : M \to I \to (\mathbb{Z}^d)^k \to (M \cup \{\emptyset\})^k : m_i \mapsto i \mapsto (i_1, \ldots, i_k) \mapsto (n_1, \ldots, n_k)$.

The neighbourhood of a cell is the same for all iterations and theoretically it could be possible to investigate *infinite neighbourhoods*. A relative index tuple must then be replaced by a relative index family etc.

2.1.1.4 Border (Topology III)

The concept of a border is naturally induced by the (eventual) boundedness of the index set. Two types of special cells can be distinguished:

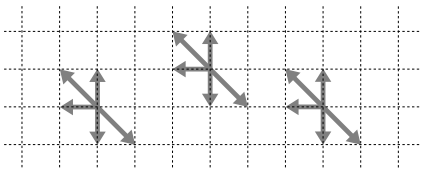

Figure 2.3: An index translation assigns an ordered set of neighbours to every cell.

Definition 2.1.12 (Border-Cell)**.** A *border-cell* is a cell, which is located at the boundary of the lattice. That is the case when the cell with index $i \in I$ misses at least one of its direct neighbours $\sharp\{\mathcal{I}^{-1}(i+j) : \mathcal{I}^{-1}(i+j) = \emptyset, j \in \{-1,0,1\}^d\} > 0$ or in other words when $\{i+j : j \in \{-1,0,1\}^d\} \not\subseteq I$.

The definition of a border-cell is actually neither utterly exact nor of primary interest for our definition of ordinary cellular automata. It is however necessary to dismiss the colloquial term "border-cell" in favour of a more precise definition suitable for describing the irregularities occurring at the boundary of a lattice in a mathematical fashion.

Definition 2.1.13 (Degraded Neighbourhood)**.** If the absolute index tuple of a cell m_i does not lie completely within the index set $\mathcal{T}_J(i) = (i+j)_{j\in J} \notin I^k$, we talk of (a cell with) a *degraded neighbourhood*.

Under certain conditions, like for some types of metric/local neighbourhoods, the set of the border-cells is a subset of the cells with degraded neighbourhoods. This differentiation becomes effective at the latest when the update rules are defined. We will actually see that an update rule must react on degraded neighbourhoods in order to implement boundary conditions.

Besides the distinction of cells with degraded neighbourhoods, we can also manipulate the geometry of a lattice by "gluing" specific cells from the lattice to the boundary. One possible reason for doing this is to avoid degraded neighbourhoods. We can describe such situations in a formal and exact manner using indices:

Definition 2.1.14 (Generalised Index Translation)**.** Given a relative index tuple J, a *generalised index translation* is defined by $\mathcal{T}_{\tau,J} : I \to I^k : i \mapsto (i_1, \ldots, i_k)$ where

$$i_\alpha := \begin{cases} i + j_\alpha & i + j_\alpha \in I \\ \tau(i, j_\alpha) & i + j_\alpha \notin I \end{cases} \qquad \alpha \in \{1, \ldots, k\} \tag{2.1.3}$$

and $\tau : I \times \mathbb{Z}^d \to I$ or sometimes also $\tau : \mathbb{Z}^d \setminus I \to I$ maps the absolute index i and the relative index j onto an absolute index in I.

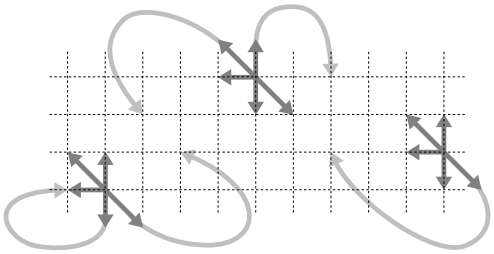

Figure 2.4: A generalised index translation usually replaces indices from $\mathbb{Z}^d \setminus I$ with indices from I.

Example 2.1.15. A toroid geometry for a two-dimensional ordinary cellular automaton (coll. "periodic boundary condition") can be achieved by applying the modulus function on nonexistent elements of the absolute index tuple. In this case there would no longer exist degraded neighbourhoods!

A possible consequence of a generalised index translation respectively of the mapping τ – as mentioned before – can be the elimination of degraded neighbourhoods, which facilitates the definition of update rules (see further below). The term "boundary conditions" obviously applies to the implementation of special update rules for cells with degraded neighbourhoods.

A generalised index translation leads to a distortion or manipulation of the natural geometry of the cellular automaton. In some cases it may be necessary to require that $\tau(i, j) \notin i + J$ in order to avoid neighbourhoods which contain one cell multiple times.

Another way to facilitate the definition of update rules or the analysis of the topology of a cellular automaton is to restrict the area of observation to cells respectively indices with nondegraded neighbourhoods.

Definition 2.1.16 (Cells with a Complete Neighbourhood)**.** The *nondegraded part* of an ordinary cellular automaton is composed of the cells with complete neighbourhood and mathematically defined as $M^\circ = \{m \in M : \mathcal{T} \circ \mathcal{I}(m) \subseteq I\}$ respectively $I^\circ = \{i \in I : \mathcal{T}(i) \subseteq I\}$.

Using this notation the restriction of the index translation $\mathcal{T}_J : I \to (\mathbb{Z}^d)^k$ can be written as $\mathcal{T}_J : I^\circ \to I^k$.

2.1.1.5 States

Another basic property of (ordinary) cellular automata is that every cell at any given time/iteration takes exactly one state from a set of possible states.

Definition 2.1.17 (Set of States)**.** The *set of possible states* is denoted as as $\mathbb{S}$.

It is neither required that the set of possible states if finite nor that there exists any special topological or algebraic structure on it. However, a non-trivial cellular automaton features more than one different element in $\mathbb{S}$. A *state mapping* assigns a (current or temporary) state to each cell.

Definition 2.1.18 (State Mapping)**.** A mapping $\mathcal{S} : M \to \mathbb{S}$ is called *state mapping*. Accordingly $\mathcal{S}(m) \in \mathbb{S}$ is the current or temporary state of cell m.

For mapping multiple cells, the notation $\left(\prod_{l=1}^{k} \mathcal{S}\right) : (M \cup \{\emptyset\})^k \to (\mathbb{S} \cup \{\emptyset\})^k : (m_1, \ldots, m_k) \mapsto (s_1, \ldots, s_k)$ where

$$s_\alpha := \begin{cases} \mathcal{S}(m_\alpha) & m_\alpha \in M \\ \emptyset & m_\alpha \notin M \iff m_\alpha = \emptyset \end{cases} \qquad \alpha \in \{1, \ldots, k\} \tag{2.1.4}$$

can be used. For simplicity we also use $\mathcal{S}$ as $\mathcal{S} : (M \cup \{\emptyset\})^k \to (\mathbb{S} \cup \{\emptyset\})^k$. The *nonexistent state* $\emptyset$ is required to maintain the tuple structure and to indicate a degraded neighbourhood.

The state mapping is the one and only variable of a cellular automaton. The state mapping evolves over iteration in the sense that the state mapping at a certain iterative step will be replaced by another state mapping in the next iterative step.

Definition 2.1.19 (Characterisations of States)**.** By introducing a partitioning on the set of all possible states, different *cell types* can be distinguished. The set of all states can be *finite* or *infinite*. Furthermore if $\mathbb{S}$ is a vector space, ring, etc. we can call $\mathbb{S}$ *state-space*.

Example 2.1.20. The set of all possible states can for example be a set of abstract states $\mathbb{S} = \{\text{red}, \text{green}, \text{blue}\}$ or the vector space $\mathbb{S} = \mathbb{R}^3$.

2.1.1.6 Update Rule

An update rule can be the explicit definition of a mapping but also a linear function.

Definition 2.1.21 (Update Rule)**.** An *update rule* (also *update rules*, *update rule set* or *update function*) is a mapping $\mathcal{F} : (\mathbb{S} \cup \{\emptyset\})^k \to \mathbb{S} : (s_1, \ldots, s_k) \mapsto s$

The update rules do not vary over iterations and stochastic update rules are explicitly excluded from a basic definition since the necessary introduction of probability spaces etc. requires additional formalisms and investigations (see Section 5.2).

A new state for a cell is obtained through $\mathcal{F} \circ \mathcal{S} \circ \mathcal{N}$:

$$\begin{cases} M & \to & (M \cup \{\emptyset\})^k & \to & (\mathbb{S} \cup \{\emptyset\})^k & \to & \mathbb{S} \\ m & \mapsto & (m_1, \ldots, m_k) & \mapsto & (s_1, \ldots, s_k) & \mapsto & s \end{cases} \tag{2.1.5}$$

or in a more detailed fashion by $\mathcal{F} \circ \mathcal{S} \circ \mathcal{I}^{-1} \circ \mathcal{T} \circ \mathcal{I}$:

$$\left\{ \begin{array}{ccccccccccc} M & \to & I & \to & (\mathbb{Z}^d)^k & \to & (M \cup \{\emptyset\})^k & \to & (\mathbb{S} \cup \{\emptyset\})^k & \to & \mathbb{S} \\ m & \mapsto & i & \mapsto & (i_1, \ldots, i_k) & \mapsto & (m_1, \ldots, m_k) & \mapsto & (s_1, \ldots, s_k) & \mapsto & s. \end{array} \right. \tag{2.1.6}$$

Since degraded neighbourhoods contain nonexistent cells respectively states ($\emptyset$), an update rule must react on a degraded neighbourhood and implement the desired "boundary conditions". An update rule never defines or modifies the geometry of a lattice!

Definition 2.1.22 (Boundary Condition)**.** A *boundary condition* describes the reaction of an update rule set on degraded neighbourhoods i.e. the treatment of degraded neighbourhoods by the update rules.

2.1.1.7 Excursus: Composite Update Rules

It is possible to define *boundary conditions* for an ordinary cellular automaton in a more systematic fashion by differentiating different types of degradations and applying different update functions depending on the type of degradation. Secondly it is often not necessary or unnecessarily complicated to define the update rule for a neighbourhood configuration that will never occur.

Example 2.1.23. For an ordinary two-dimensional square lattice of 10 by 10 cells and the classical von Neumann neighbourhood, a cell will never have more than two empty neighbours. This excludes all state-tuples containing more than two empty states. Furthermore a cell will never have two empty neighbours in opposing directions and so forth.

Generally in certain situations the update rules can be defined on a much smaller subset of possible neighbourhood configurations.

Example 2.1.24 (Composite Update Rule). The update rule set $\mathcal{F}$ can be interpreted or redefined as a mapping

$$\mathcal{F} : (\mathbb{S} \cup \{\emptyset\})^k \;\to\; \{0,1\}^k \times \mathfrak{T}(\mathbb{S}) \;\to\; \mathbb{S}^{\mathfrak{T}(\mathbb{S})} \times \mathfrak{T}(\mathbb{S}) \;\to\; \mathbb{S} \tag{2.1.7}$$

where $\mathfrak{T}(\mathbb{S})$ is the set of $\mathbb{S}$-tuples (with arbitrary number of elements) and $\mathbb{S}^{\mathfrak{T}(\mathbb{S})}$ is the set of mappings, which map $\mathbb{S}$-tuples onto elements of $\mathbb{S}$ and

$$\begin{aligned} \mathcal{F} : (s_1, \ldots, s_k) \mapsto \Big((o_1, \ldots, o_k), (s_{\alpha_1}, \ldots, s_{\alpha_l}) \Big) \mapsto \\ \mapsto \Big(f_{(o_1,\ldots,o_k)}, (s_{\alpha_1}, \ldots, s_{\alpha_l}) \Big) \mapsto f_{(o_1,\ldots,o_k)}\big((s_{\alpha_1}, \ldots, s_{\alpha_l}) \big) =: s \end{aligned} \tag{2.1.8}$$

where

1. $(o_1, \ldots, o_k)$ indicates occupied positions in the neighbourhood using $o_i = 1$ if and only if $s_i \neq \emptyset$ else $o_i = 0$ for $i = 1, \ldots, k$,
2. $l := \sharp\{s_i : s_i \neq \emptyset, i = 1, \ldots, k\} = \sum_{i=1}^{k} o_i \leq k$ is the number of occupied slots in the neighbourhood,
3. $(s_{\alpha_1}, \ldots, s_{\alpha_l}) \in \mathbb{S}^l$ is the sorted tuple $(s_1, \ldots, s_k)$ without empty states,
4. $f_{(o_1,\ldots,o_k)}$ are update functions from $\mathbb{S}^l$ to $\mathbb{S}$.

Accordingly $\mathcal{F}_{2,1} : \{0,1\}^k \to \mathbb{S}^{\mathfrak{T}(\mathbb{S})}$ assigns an update mapping $f : \mathbb{S}^l \to \mathbb{S}$ to a cell respectively neighbourhood depending on the degradation of the neighbourhood $(o_1, \ldots, o_k)$. It is sufficient when $\mathcal{F}_{2,1}$ is defined for all occurring neighbourhood degradations, which themselves are completely defined through the index set, the relative index tuple and eventually the generalised index translation.

2.1.1.8 Compatibility of Update Rules.

Since it is sufficient and even advantageous to define the update rules on a subset of $(\mathbb{S} \cup \{\emptyset\})^k$ as $\mathcal{F} : (\mathbb{S} \cup \{\emptyset\})^k \supset \operatorname{dom} \mathcal{F} \to \mathbb{S}$, compatibility of $\mathcal{F}$ with all possibly occurring state configurations of neighbourhoods has to be assured.

Definition 2.1.25 (Update Rule with Restricted Domain)**.** We redefine the update rules as a mapping $\mathcal{F} : (\mathbb{S} \cup \{\emptyset\})^k \supset \operatorname{dom} \mathcal{F} \to \mathbb{S}$.

Definition 2.1.26 (Compatibility of Update Rules)**.** If a state tuple for which $(s_1, \ldots, s_k) \notin \operatorname{dom} \mathcal{F}$ is encountered, we talk of *undefined behaviour*. Undefined behaviour can be avoided if $\mathcal{F}$ respectively $\operatorname{dom} \mathcal{F}$ satisfies the following conditions:

- An update rule must be *compatible with the topology of the cellular automaton* i.e. all occurring degradations or configurations of neighbourhoods must be taken into account by the update rules. We call an update rule *incompatible* with a state mapping $\mathcal{S}$ given $\mathcal{N}$ if

$$\exists m \in M : \quad \mathcal{S} \circ \mathcal{N}(m) \notin \operatorname{dom} \mathcal{F} \tag{2.1.9}$$

- An update rule must be *self-contained*. This means that all possible state configurations of the neighbourhoods, which – apart form the initial condition – arise from the update rule itself, must lie within the domain of the update rules. An update rule is *self-contained* if for the neighbourhood of each cell $m \in M$

$$\mathcal{S} \circ \mathcal{N}(n) \in \operatorname{dom} \mathcal{F} \quad \forall n \in \mathcal{N}(m) \quad \Longrightarrow \quad \tilde{\mathcal{S}} \circ \mathcal{N}(m) \in \operatorname{dom} \mathcal{F} \tag{2.1.10}$$

where $\tilde{\mathcal{S}} := \mathcal{F} \circ \mathcal{S} \circ \mathcal{N}$ is understood as an element-wise operator.

2.1.1.9 Global Description

The dynamics or behaviour of a cellular automaton can only be observed from a global perspective. A *global state* is the collection of the states of all cells.

Definition 2.1.27 (Global State, Phase Space)**.** Because the (current) states of all cells are accumulated in the state mapping $\mathcal{S}$, $\mathcal{S}$ can also be called *global state* and interpreted as an element of the *set of global states* or *phase space* $\mathbb{S}^M$.

$\mathcal{S}$ can be interpreted as state mapping (Definition 2.2.6) as well as global state (Definition 2.1.27).

2.1.1.10 Iterative Process and Evolution

The dynamics of a cellular automaton consists in the iteration of global states.

Definition 2.1.28 (Evolution Operator)**.** Given a neighbourhood mapping $\mathcal{N}$ and an update rule $\mathcal{F}$, the *evolution operator* is the mapping $\mathcal{E} : \mathbb{S}^M \to \mathbb{S}^M : \mathcal{S} \mapsto \tilde{\mathcal{S}} := \mathcal{F} \circ \mathcal{S} \circ \mathcal{N}$.

Definition 2.1.29 (Iteration of Global States)**.** For a given initial state $\mathcal{S}_0$, the iteration of global states is defined by $\mathcal{S}_{t+1} := \mathcal{E}(\mathcal{S}_t) = \mathcal{F} \circ \mathcal{S}_t \circ \mathcal{N}$ where $t \in \mathbb{N}$ or sometimes $t \in T$ for a connected subset $T \subset \mathbb{N}$. The iteration of global states yields a mapping $T \to \mathbb{S}^M : t \mapsto \mathcal{S}_t$. We also use the notation $\mathcal{S}(t)$ and $\mathcal{S}(t, m)$, which actually renders $\mathcal{S}$ a function $T \to \mathbb{S}^M$ respectively $T \times M \to \mathbb{S}$.

Usually the implementation of cellular automata only stores the current temporary global state so that this mapping is not necessarily available once the iteration of the cellular automaton is completed. This corresponds to the idea that a cellular automaton operates memoryless.

Again note, that only the states of the cells and accordingly the state mapping may change during iteration!

It is possible to use update rules that depend on the number of the current iteration. In this case we would have $\mathcal{F} : \mathbb{N} \times (\mathbb{S} \cup \{\emptyset\})^k \to \mathbb{S} : (t; s_1, \ldots, s_k) \mapsto s$. We will however exclude this special case from our default definition.

2.1.2 Final Definition of Ordinary Cellular Automata

Definition 2.1.30 (Ordinary Cellular Automaton)**.** An *ordinary cellular automaton* is completely defined by

(i) a set of cells M with indexing $(I, \mathcal{I})$,

(ii) a neighbourhood mapping $\mathcal{N}$, which is composed of a relative index tuple J and, if the geometry of the cellular automaton shall be manipulated, additionally a corresponding mapping τ,

(iii) a set of possible states $\mathbb{S}$ and

(iv) a self-contained update rule $\mathcal{F}$,

which in a general form can be collected in the 4-tuple $((M, I, \mathcal{I}), \mathcal{N}, \mathbb{S}, \mathcal{F})$ respectively in a more detailed fashion as $((M, I, \mathcal{I}), (J, \tau), \mathbb{S}, \mathcal{F})$.

Given a compatible initial global state $\mathcal{S}_0$ – such that the update rule is compatible with $\mathcal{S}_0$ given $\mathcal{N}$ –, an iteration of global states is defined. The resulting series of state mappings describes the evolution of the cellular automaton.

2.2 Unaligned Cellular Automata

An important key feature of ordinary cellular automata is the alignment of the cells on a regular grid structure, implemented in the previous section by an index mapping, which identifies cells with indices form an index set in $\mathbb{Z}^d$. Thereupon the translation of indices and the manipulation/distortion of lattices were defined. Furthermore the empty cell and the empty state had to be introduced. By omitting the alignment of cells it is possible/necessary to abandon this detour in favour of a more abstract characterisation of neighbourhoods.

Many definitions can be inherited from ordinary cellular automata. If a definition is an exact copy of a definition from Section 2.1, this is mentioned in the caption. Furthermore in this section some trivialities and basic conclusions which were already mentioned in Section 2.1 are omitted.

2.2.1 Generalisation and Reformulation of Concepts

The basic definition of a cell as an entity among many equal entities is a fundamental feature also for unaligned cellular automata.

Definition 2.2.1 (Cell, Set of Cells, compare Definition 2.1.1)**.** Let M denote the *set of cells*. Consequently a *cell* is a – necessarily – unique element $m \in M$. The set of cells is not empty but countable and most often finite.

2.2.1.1 Explicit Neighbourhoods

In the index-based approach the neighbourhood of a distinct cell i.e. the neighbouring cells respectively their indices were obtained from the index of the actual cell using a neighbourhood mapping which was composed of the index mapping, an index translation and the inverse of the index mapping

$$\mathcal{N} := \mathcal{I}^{-1} \circ \mathcal{T} \circ \mathcal{I} : \left\{ \begin{array}{ccccccc} M & \to & I & \to & (\mathbb{Z}^d)^k & \to & (M \cup \{\emptyset\})^k \\ m_i & \mapsto & i & \mapsto & (i_1, \ldots, i_k) & \mapsto & (n_1, \ldots, n_k) \end{array} \right. . \tag{2.2.1}$$

When the indexing of cells is abandoned there can not exists such an algorithmic method for assigning a neighbourhood to each cell. Therefore the neighbourhood of a cell must be defined explicitly as a mapping

$$\mathcal{N} : m \mapsto (n_1, \ldots, n_{k_m}) \tag{2.2.2}$$

where k_m might be different for every cell $m \in M$.

In order to achieve an universal notation we define $\mathfrak{T}(M)$ similar to the power set $\mathfrak{P}(M)$.

Definition 2.2.2 (Tuples of Cells)**.** The set of tuples of arbitrary length of elements in M is denoted as $\mathfrak{T}(M)$. If the ordering of cells is irrelevant the notation $\langle n_1, \ldots, n_k \rangle$ shall be used instead of $(n_1, \ldots, n_k)$. $\langle \cdot \rangle$ can be regarded as a set with possibly repeating elements.

Since M is countable we can interpret $\mathfrak{P}(M) \subseteq \mathfrak{T}(M)$ by simply ignoring any ordering. Accordingly a neighbourhood mapping can be defined as follows.

Definition 2.2.3 (Neighbourhood Mapping)**.** A *neighbourhood mapping* is a function

$$\mathcal{N} : M \to \mathfrak{T}(M) : m \mapsto (n_1, \ldots, n_{k_m}), \tag{2.2.3}$$

where $n_i \neq n_j$ for $i, j \in \{1, \ldots, k_m\}$ and $i \neq j$.

Definition 2.2.4 (Characterisation of Neighbourhoods and Neighbourhood Mappings)**.** In contrast to ordinary cellular automata the sizes of the neighbourhoods can now vary.

(i) The neighbourhood of a cell m is called *reflexive* if $m \in \mathcal{N}(m)$.

(ii) A neighbourhood mapping is called *symmetric* if for all $m, n \in M$, $n \in \mathcal{N}(m) \Leftrightarrow m \in \mathcal{N}(n)$.

(iii) The neighbourhoods are called *ordered* if $\mathcal{N} : m \mapsto (n_1, \ldots, n_{k_m})$ and *unordered* if $\mathcal{N} : m \mapsto \langle n_1, \ldots, n_{k_m} \rangle$.

(iv) Obviously $k_m \in \mathbb{N}$ denotes the *size of the neighbourhood* of m.

(v) The neighbourhoods are called *equally sized* if $k_m = k$ for all $m \in M$ and a $k \in \mathbb{N}$.

For certain special cases there exist different formulations. For example a neighbourhood mapping with unordered neighbourhoods could be written as

$$\mathcal{N} : M \to \mathfrak{P}(M) : m \mapsto N \tag{2.2.4}$$

(compare with the mathematical concept of *correspondences*) where a neighbourhood mapping with equally sized neighbourhoods could be written as

$$\mathcal{N} : M \to M^k : m \mapsto (n_1, \ldots, n_k). \tag{2.2.5}$$

The choice for a certain type of neighbourhood mapping is of course made during the process of modelling.

The requirement $n_i \neq n_j$ for $i, j \in \{1, \ldots, k_m\}$ and $i \neq j$ in Definition 2.2.3 corresponds to the idea that a cell can only be contained once within the same neighbourhood. In some situation this requirement cannot be fulfilled. We will however assume that almost all cells feature a neighbourhood without repeating neighbours and that repeating neighbours can only occur as a "boundary condition". Of course for unaligned cells the term "boundary" is not perfectly suitable but corresponds to the concept of distorted lattices for ordinary cellular automata. Furthermore degraded neighbourhoods can occur if almost all cells have equally sized neighbourhoods but only some cells have fewer or more neighbours.

2.2.1.2 State Mapping and Global State

The assignment of a (current or temporary) *state* to each cell happens through *state mappings*.

Definition 2.2.5 (Set of States, see also Definition 2.1.17)**.** The *set of possible states* is denoted as $\mathbb{S}$.

Definition 2.2.6 (State Mapping, see also Definition 2.1.18)**.** A mapping $\mathcal{S} : M \to \mathbb{S}$ is called *state mapping*. Accordingly $\mathcal{S}(m) \in \mathbb{S}$ is the current or temporary state of cell m.

Also the lazy notation

$$\mathcal{S} : \mathfrak{T}(M) \to \mathfrak{T}(\mathbb{S}) : \begin{cases} (n_1, \ldots, n_k) & \mapsto & (s_1, \ldots, s_k) \\ \langle n_1, \ldots, n_k \rangle & \mapsto & \langle s_1, \ldots, s_k \rangle \\ \{n_1, \ldots, n_k\} & \mapsto & \langle s_1, \ldots, s_k \rangle \end{cases} \tag{2.2.6}$$

where $\mathcal{S}(n_i) = s_i$ for all $i \in \{1, \ldots, k\}$ may be used. However $\mathcal{S} \circ \mathcal{N}(m) = \mathcal{S}(\mathcal{N}(m)) = (\mathcal{S}(n))_{n \in \mathcal{N}(m)}$ must be regarded as an element of $\mathfrak{T}(\mathbb{S})$ and not $\mathbb{S}^{\mathcal{N}(m)}$.

Definition 2.2.7 (Characterisations of States, see also Definition 2.1.19)**.** By introducing a partitioning on the set of all possible states, different *cell types* can be distinguished. The set of all states can be *finite* or *infinite*. Furthermore if $\mathbb{S}$ is a vector space, ring, etc. we can call $\mathbb{S}$ *state-space*.

Definition 2.2.8 (Global State, compare Definition 2.1.27)**.** As an element of the *phase space* $\mathfrak{S} := \mathbb{S}^M$ or $\mathfrak{S} \subseteq \mathbb{S}^M$, a state mapping can be interpreted as a *global state*.

2.2.1.3 Update Rules

An update rule delivers a state given a tuple of states. Usually the argument of an update function is the collection of states of cells within a certain neighbourhood.

Definition 2.2.9 (Update Rules)**.** An *update rule* or *update function* is a mapping $\mathcal{F} : \mathfrak{T}(\mathbb{S}) \to \mathbb{S}$.

As discussed for ordinary cellular automata, an update function must be compatible with all possibly occurring state-configurations of the neighbourhoods also for unaligned cellular automata. More exactly, an update rule must be *compatible with the topology* and *self-contained*.

Definition 2.2.10 (Compatibility)**.** An update function $\mathcal{F} : \mathfrak{T}(\mathbb{S}) \supset \operatorname{dom}\mathcal{F} \to \mathbb{S}$ is called

(i) *compatible* with the neighbourhood mapping $\mathcal{N} : M \to \mathfrak{T}(M)$ and a global state $\mathcal{S} : M \to \mathbb{S}$ if for all $m \in M$, $\mathcal{S} \circ \mathcal{N}(m) \in \operatorname{dom}\mathcal{F}$ and

(ii) *self-contained* if $\mathcal{S} \circ \mathcal{N}(m) \in \operatorname{dom}\mathcal{F}$ for all $m \in M$ leads to the compatibility of $\mathcal{F}$ with respect to the global state $\tilde{\mathcal{S}} := \mathcal{F} \circ \mathcal{S} \circ \mathcal{N}$ i.e. $\tilde{\mathcal{S}} \circ \mathcal{N}(m) \in \operatorname{dom}\mathcal{F}$ for all $m \in M$.

In certain special cases – also depending on the neighbourhood mappings – the update function can be written as a function $\mathcal{F} : \mathfrak{P}(\mathbb{S}) \supset \operatorname{dom}\mathcal{F} \to \mathbb{S}$ or $\mathcal{F} : \mathbb{S}^k \supset \operatorname{dom}\mathcal{F} \to \mathbb{S}$.

Usually the formulation of update rules for uniformly sized neighbourhoods is much simpler than for neighbourhoods with varying sizes. Especially for varyingly sized neighbourhoods it is necessary to restrict the domain of the update rule while simultaneously keeping the update function fully compatible with all occurring neighbourhood/state configurations. For example the set of all tuples with arbitrary number of elements $\mathfrak{T}(\,\cdot\,)$ is per se infinite.

2.2.1.4 Iteration and Evolution Operator

A new state for all cells can be obtained by $\mathcal{F} \circ \mathcal{S} \circ \mathcal{N} : M \to \mathfrak{T}(M) \to \mathfrak{T}(\mathbb{S}) \to \mathbb{S}$ in one of the following forms:

$$\begin{cases} m & \mapsto & (n_1, \ldots, n_{k_m}) & \mapsto & (s_1, \ldots, s_{k_m}) & \mapsto & s \\ m & \mapsto & \langle n_1, \ldots, n_{k_m} \rangle & \mapsto & \langle s_1, \ldots, s_{k_m} \rangle & \mapsto & s \\ m & \mapsto & \{n_1, \ldots, n_{k_m}\} & \mapsto & \langle s_1, \ldots, s_{k_m} \rangle & \mapsto & s \end{cases} . \tag{2.2.7}$$

Definition 2.2.11 (Evolution Operator, see also Definition 2.1.28)**.** Given a neighbourhood mapping $\mathcal{N}$ and an update rule $\mathcal{F}$, the *evolution operator* is the mapping $\mathcal{E} : \mathfrak{S} \to \mathfrak{S} : \mathcal{S} \mapsto \tilde{\mathcal{S}} := \mathcal{F} \circ \mathcal{S} \circ \mathcal{N}$.

Definition 2.2.12 (Iteration of Global States, see also Definition 2.1.29)**.** For a given initial state $\mathcal{S}_0 \in \mathfrak{S}$ that is compatible with $\mathcal{F}$ given $\mathcal{N}$, the iteration of global states is defined by $\mathcal{S}_{t+1} := \mathcal{E}(\mathcal{S}_t) = \mathcal{F} \circ \mathcal{S}_t \circ \mathcal{N}$ where $t \in \mathbb{N}$ or sometimes $t \in T$ for a connected subset $T \subset \mathbb{N}$. The iteration of global states yields a mapping $T \to \mathfrak{S} : t \mapsto \mathcal{S}_t$ where $\mathfrak{S} \subseteq \mathbb{S}^M$. We also use the notation $\mathcal{S}(t)$ and $\mathcal{S}(t, m)$, which actually renders $\mathcal{S}$ a function $T \to \mathfrak{S}$ respectively $T \times M \to \mathbb{S}$.

2.2.2 Final Definition of Unaligned Cellular Automata

Definition 2.2.13 (Unaligned Cellular Automaton)**.** An *unaligned cellular automaton* is completely defined by the 4-tuple $(M, \mathcal{N}, \mathbb{S}, \mathcal{F})$, which contains

(i) a set of cells M,

(ii) a neighbourhood mapping $\mathcal{N}$,

(iii) a set of possible states $\mathbb{S}$ and

(iv) a self-contained update rule $\mathcal{F}$.

Given a compatible initial global state $\mathcal{S}_0$ – such that the update rule is compatible with $\mathcal{S}_0$ given $\mathcal{N}$ –, an iteration of global states is defined. The resulting series of state mappings describes the evolution of the cellular automaton.

Compare the formal definition respectively tuple-representation of ordinary cellular automata $((M, I, \mathcal{I}), (J, \tau), \mathbb{S}, \mathcal{F})$ with $((M, \emptyset), \mathcal{N}, \mathbb{S}, \mathcal{F})$. The empty set $\emptyset$ indicates that there exists no topology I for unaligned cellular automata.

Chapter 3

Topology of Cellular Automata

As discussed in Section 1.3.1 and Section 1.3.2 the cells of a cellular automaton are usually embedded in some kind of environment. This is motivated on the level of modelling by the idea that either cells represent discrete regular compartments of space (a priori alignment) or that abstract entities are arranged as cells in a regular grid-like fashion (a posteriori alignment). This concept of arrangement introduces a more or less natural concept of proximity between cells and is the *first topological feature* of a cellular automaton and – in the context of this thesis – also summarised under the term *topology I*. For explicitly unaligned cellular automata there exists no natural structure on the set of all cells. Accordingly the first topological feature does not exist in this case. A *second topological feature* is introduced by the concept of neighbourhoods, denoted *topology II*. This feature is always available for cellular automata and constitutes one of the main principles of cellular automata since only neighbouring cells can influence each other.

However the notation *topology* is not intended to imply the existence of a topology as a mathematical structure in the context of open sets. Nevertheless we will see in Section 3.3 that both topological features always generate a topology in the mathematical sense on the set of cells.

The results of this section will be useful to discuss the connectivity between both structural features and even to combine them if necessary.

Characterisation of Neighbourhoods. In Chapter 2 neighbourhoods were defined in a mathematical fashion either as a composition of index operations (ordinary cellular automata, Section 2.1) or as explicitly defined mapping (unaligned cellular automata, Section 2.2).

Some basic characterisations of neighbourhoods respectively neighbourhood mappings like *symmetry*, *reflexivity*, *locality* and *size* were introduced. This paragraph provides additional characterisations and notations in connection with neighbourhoods.

Definition 3.0.14 (Higher Order Neighbourhood)**.** For a given neighbourhood mapping $\mathcal{N}$ the *neighbourhood of order k* of a cell m, denoted by $\mathcal{N}^k(m) \subseteq M$, is a subset

of M defined by: $n \in \mathcal{N}^k(m)$ if and only if one of the following (exclusive) situations applies.

$$\begin{array}{cc} n = m & \wedge \quad k = 0 \\ n \in \mathcal{N}(m) & \wedge \quad k = 1 \\ \exists\, n_1 : n \in \mathcal{N}(n_1), n_1 \in \mathcal{N}(m) & \wedge \quad k = 2 \\ \left\{ \begin{array}{c} \exists\, n_i,\ i \in \{1, \ldots, k-1\} : \\ n \in \mathcal{N}(n_1), n_1 \in \mathcal{N}(n_2), \ldots, n_{k-1} \in \mathcal{N}(m) \end{array} \right\} & \wedge \quad k \geq 3 \\ \text{else} & \wedge \quad k = \infty \end{array} \tag{3.0.1}$$

Accordingly also the notations *neighbour of order k* and *k-order neighbourhood mapping* etc. are justified.

Definition 3.0.15 (Connectivity of Cells)**.** If for any two cells $m \neq n$ there exists no number $k \in \mathbb{N}$ such that $n \in \mathcal{N}^k(m)$ or $m \in \mathcal{N}^k(n)$ we talk of a *disconnected* neighbourhood structure or *disconnected topology II*.

Example 3.0.16. For example $\mathcal{N} : m \mapsto \{m\}$ delivers a disconnected topology II.

If a neighbourhood mapping delivers a disconnected topology II, there are effectively multiple independent cellular automata in operation.

Definition 3.0.17 (Inverse Neighbourhood)**.** A cell n is called *inverse neighbour* of a cell m if m is a neighbour of n i.e. if $m \in \mathcal{N}(n)$. The set $\mathcal{N}^{-1}(m) := \{n \in M : m \in \mathcal{N}(n)\}$ is called *inverse neighbourhood* of m.

In some cases the inverse neighbourhood can be a tuple. This is at least possible for ordinary cellular automata. A symmetric neighbourhood structure is characterised by $\mathcal{N}(m) = \mathcal{N}^{-1}(m)$ (probably except for ordering).

The following definitions anticipate an important concept from graph theory (compare Section 3.2). For a connected topology II and two cells $m_a, m_b \in M$ there exists a $k \in \mathbb{N}$ and a $(k-2)$-tuple of cells $(m_2, \ldots, m_{k-1})$ such that

$$m_1 \in \mathcal{N}(m_2), m_2 \in \mathcal{N}(m_3), \cdots, m_{k-1} \in \mathcal{N}(m_k) \tag{3.0.2}$$

where either $m_a = m_1 \wedge m_b = m_k$ or $m_b = m_1 \wedge m_a = m_k$.

Definition 3.0.18 (Directed Path)**.** We call a tuple $(m_1, \ldots, m_k)$ that satisfies Equation 3.0.2 a *directed path* from m_1 and m_k with *length* $k-1$. We also call (m) a path with length 0 even when $m \notin \mathcal{N}(m)$.

Definition 3.0.19 (Undirected Path)**.** A tuple $(m_1, \ldots, m_k)$ is called *undirected path* between m_1 and m_k of *length* $k-1$ if

$$m_1 \in \mathcal{N}(m_2) \vee m_2 \in \mathcal{N}(m_1), \ldots, m_{k-1} \in \mathcal{N}(m_k) \vee m_k \in \mathcal{N}(m_{k-1}). \tag{3.0.3}$$

3.1 Vector Space Interpretation

The aim of this section is to equip the set of cells M with a vector space structure. It is however obvious to use $\mathbb{Z}$ as the scalar "field" (cells are discrete entities), which actually is a ring and thus renders the resulting structure – in a mathematically correct notation – a module. For readability modules over $\mathbb{Z}$ are called vector spaces.

If the set of cells M is a vector space respectively module, the addition of cells and the multiplication of cells with scalars as well as a neutral element and inverse cells are required. This is however not very intuitive in the context of our perception of cells.

Alternatively M can be interpreted as an affine space, which renders the cells points of the affine space. Assume an abstract vector space (module) V and define the *difference mapping*

$$\cdot \ - \ \cdot : M \times M \to V : (m, n) \mapsto m - n =: v. \tag{3.1.1}$$

$V := \operatorname{span}\{m - n : m, n \in M\} = \operatorname{span}(M - M)$ is called *difference space*. The vector space structure of V defines a relative position or alignment between each two cells. If for example $m_1 - n_1 = v_1 = v_2 = m_2 - n_2$, we would say that m_1 is aligned with respect to n_1 the same way as m_2 is aligned with respect to n_2.

Definition 3.1.1 (Alignment)**.** If for each two cells $m, n \in M$ there exists an abstract element v from a vector space or module V such that the difference $m - n$ can be represented by v, V is called *alignment* and the cells are *aligned* according to V.

Note that the vector $v := m - n$ points from the neighbour towards the cell. Again this way, the orientation of the vector describes the flow of information. In order to obtain the neighbour of a cell, the subtraction $n = m - v$ must be used.

Definition 3.1.2 (Basis of an Alignment)**.** Assume there exists a minimal set of (abstract) elements $B \subset V$ such that we can write for all $m, n \in M$

$$m - n = \sum_{b \in B} \beta_b b \qquad \text{where} \qquad \beta_b \in \mathbb{Z},\ b \in B. \tag{3.1.2}$$

We then call B a (normal) *basis* of the vector space or module V over $\mathbb{Z}$. The number of elements in B is called *dimension* of V, written $\dim V$.

Since V describes an alignment of the cells (first topological feature) it actually does not simply exist but must either be defined during the process of modelling (compare 1.3.1) or constructed in another way (compare Theorem 3.4.5).

Proposition 3.1.3 (Existence of an Alignment)**.** *For every cellular automaton with a finite set of cells $|M| < \infty$ there exists an alignment with dimension*

$$\dim V \leq \binom{|M|}{2} = \frac{|M|!}{2 \cdot (|M| - 2)!}. \tag{3.1.3}$$

Proof. Assume that all vectors $m - n$ are linearly independent. Then there exist $\binom{|M|}{2}$ basis vectors. □

Based on such an alignment of the cells, the neighbourhood structure can be represented in a vector space notation. This means that the second topological feature of a cellular automaton can be expressed using the first topological feature as it was done for ordinary cellular automata.

Proposition 3.1.4 (Vector Representation of Neighbourhoods)**.** *Let $V = M - M$ be an alignment and $\mathcal{N} : M \to \mathfrak{T}(M)$ a neighbourhood mapping. The neighbourhood mapping can be identified with an element from $M \times \mathfrak{T}(V)$ respectively with a mapping $M \to \mathfrak{T}(V)$.*

Proof. Since $m - (n_1, \ldots, n_{k_m}) = (m - n_1, \ldots, m - n_{k_m}) = (v_1, \ldots, v_{k_m}) \in \mathfrak{T}(V)$ for every $m \in M$, $\mathfrak{T}(M) \times M$ can be identified with $\mathfrak{T}(V) \times M$. □

3.1.1 Ordinary Cellular Automata

The concept of a difference space respectively of an alignment obviously supersedes a regular arrangement induced by an indexing as it was used in connection with ordinary cellular automata.

Proposition 3.1.5 (Universality of the Difference Vector Space Approach)**.** *Assume an indexed $(I \subset \mathbb{Z}^d, \mathcal{I})$ set of cells M. The thereof resulting first topological feature can be interpreted as an affine vector space with difference space $V = M - M \cong I - I$ with $\dim V = d$.*

Proof. M is isomorphic to I, I as a subset of $\mathbb{Z}^d$ can be equipped with a vector space structure. Accordingly the vector space structure of $I - I \subset \mathbb{Z}^d$ can be transferred to $V = M - M$ and there exist exactly d basis vectors. □

Let $\mathcal{N} : M \to M^k$ be a (ordered) neighbourhood mapping (for almost all cells) and let V be an alignment. By observing for all cells $m \in M$ the tuple of vectors $m - \mathcal{N}(m) = (m - n_1, \ldots, m - n_k) = \left(\sum_b \beta_{b,1} b, \ldots, \sum_b \beta_{b,k} b\right) \in V^k$, where $b \in B$ are basis vectors of V, the idea of uniform neighbourhoods can be generalised from indexed cellular automata to cellular automata with vector space structure.

Definition 3.1.6 (Uniform Neighbourhood Structure)**.** The neighbourhoods of a cellular automaton with alignment V are called *uniform* or *uniformly structured* if there exists a tuple $(v_1, \ldots, v_k) \in V^k$ without repeating elements such that $m - \mathcal{N}(m) = (v_1, \ldots, v_k)$ for all $m \in M$ or equivalently if $M \to V^k : m \mapsto m - \mathcal{N}(m)$ is a constant mapping. $(v_1, \ldots, v_k)$ then determines the structure of all neighbourhoods.

Note that for a finite set of cells with uniform alignment $(v_1, \ldots, v_k) \in V^k$ there inevitably exists a cell m and a $i \in \{1, \ldots, k\}$ such that $m - v_i \notin M$. Hence the conditions in Definition 3.1.6 can only be satisfied for cells with non-degraded neighbourhoods.

An alignment can be interpreted as an indexing or in other words a cellular automaton with vector space structure and uniform neighbourhoods can be interpreted as an ordinary cellular automaton. However the geometry of a cellular automaton with vector space structure can be richer than that of an indexed cellular automaton.

Lemma 3.1.7. *Let V be an alignment, and assume a uniform neighbourhood mapping $m - \mathcal{N}(m) = (v_1, \ldots, v_k)$ yielding a connected topology II (Definition 3.0.15). Then $(v_1, \ldots, v_k)$ spans V. In other words, for each two cells m and n there exist coefficients $\alpha_1, \ldots, \alpha_k \in \mathbb{Z}$ such that $m - n = \sum_{i=1}^{k} \alpha_i v_i$. Accordingly the dimension of V satisfies $\dim V \leq k$ and it is possible to construct a basis B from $(v_1, \ldots, v_k)$ such that $B \subseteq \{v_1, \ldots, v_k\}$.*

Proof. Uniformity: $m - \mathcal{N}(m) = (v_1, \ldots, v_k)$ independently of the choice of $m \in M$. Connectivity: for each two $m, n \in M$ there exists a directed path (Definition 3.0.18) between m and n. Consequently a representation $m - n = \sum_{i=1}^{k} \alpha_i v_i \in V$ is possible. □

Theorem 3.1.8 (Alternative Formulation of Ordinary Cellular Automata)**.** *Aligned cellular automata with uniform connected neighbourhood correspond to ordinary cellular automata and vice versa.*

Proof. It remains to show that for an existing alignment V and a connected uniform neighbourhood mapping with $m - \mathcal{N}(m) = (v_1, \ldots, v_k)$ for all $m \in M$, an equivalent indexing $(I, \mathcal{I})$ with $I \subset \mathbb{Z}^d$ where $d \leq k < \infty$ and a relative index tuple J such that $\mathcal{I}^{-1}(\mathcal{I}(m) + J) = (m - v_1, \ldots, m - v_k)$ can be found.

According to Lemma 3.1.7 we can find a basis B with $d := |B| = \dim V \leq k$. Choose a $m \in M$ and define $\mathcal{I}(m) := 0 \in \mathbb{Z}^d$. Identify $b_i \in B$ with the unit vector $e_i \in \mathbb{Z}^d$ for all $i = 1, \ldots, d$. Since every $n \in M$ can be represented as $n = m - \sum_{i=1}^{d} \beta_i b_i$, define $\mathcal{I}(n) := \sum_{i=1}^{d} \beta_i e_i$. Assume $v_l = \sum_{i=1}^{d} \alpha_i b_i$ for $l = 1, \ldots, k$, define the elements of the relative index tuple $J \in (\mathbb{Z}^d)^k$ as $j_l := \sum_{i=1}^{d} \alpha_i e_i$ where $l = 1, \ldots, k$. □

3.1.2 Velocity Models

If the state of a cell contains a tuple of values for which each is associated with one of the neighbours, we talk of a *velocity model*.

Example 3.1.9. Assume that every cell has $k = 4$ neighbours $\mathcal{N}(m) = (m_1, \ldots, m_4)$ and takes a state in $\mathbb{S} = \mathbb{R}^4$ such that $\mathcal{S}(m) = (s_1, \ldots, s_4)$. Every element of the state-vector is associated with a vector from $m - \mathcal{N}(m)$. For example s_1 describes the velocity (or a scaling thereof) in direction $m - m_1$.

Instead of regarding a k-state-tuple ($\mathbb{R}^4$ in Example 3.1.9), it is possible to talk of k *subcells* with scalar states (e.g. 4 subcells with state-space $\mathbb{R}$). Then each subcell is

associated with one of the k neighbours respectively with one of the k *lattice velocities* $v_i := m_i - m$ [32, p. 89]. We will however avoid formulations based on subcells and use state-tuples instead.

If there exists a norm on the difference space, a so-called *single-speed model* [32, p. 89] is characterised by $\|v_i\| = \|v_j\|$ for all $i, j \in \{1, \ldots, k\}$. Often the neighbourhood structure respectively the lattice velocities are used for classification. For example D2Q4, D3Q27, etc. (notation introduced by Quin [32]) describe the dimension of a regular lattice and the number of neighbours respectively lattice velocities.

Definition 3.1.10 (Lattice Tensor, compare [32, p. 89])**.** Let d be the dimension of the difference space $V = M - M$ and let v_{ij} denote the j-th coordinate of the i-th "neighbourhood" vector $v_i = m - m_i$ (of m) with respect to a certain basis. The *lattice tensor* (field) of rank n is defined as $L^{(m)} = (l^{(m)}_{\alpha_1 \ldots \alpha_n})_{\alpha_1, \ldots, \alpha_n \in \{1, \ldots, d\}}$ by

$$l^{(m)}_{\alpha_1 \ldots \alpha_n} = \sum_{i=1}^{k} v_{i\alpha_1} \cdots v_{i\alpha_n}. \tag{3.1.4}$$

Definition 3.1.11 (Isotropy, compare [32, p. 88])**.** A tensor is called *isotropic* if it is invariant with respect to arbitrary orthogonal transformations (rotations and reflections) i.e. if the components $l_{\alpha_1 \ldots \alpha_n}$ are the same for every orthogonal basis.

Isotropy of lattice tensors plays an important role in *multi-scale analysis*.

> [Usually regular lattices are used such that] [t]hese [lattice] tensors are [equal for all cells and] invariant with respect to elements of the associated finite symmetry group but in general not with respect to arbitrary orthogonal transformations (including continuous rotations). A sufficient condition for 'reasonable' macroscopic equations encloses the isotropy of lattice tensors ... of 2nd and 4th rank. The lattice tensors with odd rank vanish because of the symmetry of the lattices. [32, p. 88]

> Hasslacher (1987) has shown that models with several different non-vanishing lattice speeds may be equivalent to models with a single speed but larger symmetry group. [32, p. 125]

> [G]eneralized lattice tensors
>
> $$g_{\alpha_1 \ldots \alpha_n} = \sum_{i=1}^{k} w_i v_{i\alpha_1} \cdots v_{i\alpha_n}. \tag{3.1.5}$$
>
> occur naturally in the multi-scale analysis of multi-speed models. The weights correspond to different occupation numbers for the different speeds in the global equilibrium with vanishing macroscopic velocities. [32, pp. 93-94]

3.2 Interpretation as Graphs

This section interprets the topology of cellular automata and especially the second topological feature as graphs. A complete introduction to graph theory can be found in [22] for example or in any other introductory work on graphs. In literature there exist however different mathematical descriptions for graphs.

Definition 3.2.1 (Graph [22])**.** A *graph* is a triple $G = (M, E, p)$ where M is the set of vertices and E is the (abstract) set of edges. The mapping $p : E \to M \times M$ is called *incidence mapping* and indicates the (incident) connections between vertices of the graph.

To differentiate between directed and undirected edges the notations $p : e \mapsto (m_1, m_2)$ respectively $p : e \mapsto \{m_1, m_2\}$ could be used. We however regard an undirected graph as a special case of a directed graph.

Definition 3.2.2 (Simple Graph and Multigraph [22])**.** A *simple graph* is a graph whose incidence mapping is injective. A graph whose incidence mapping is not injective is called *multigraph*, because two vertices can be connected multiple times.

For a simple directed graph we can assume that $E \subseteq M \times M$ and $p = \mathrm{id}_E$, which is a much more intuitive representation and allows the notation $G = (M, E)$.

Definition 3.2.3 (Weighted Graph, compare [22])**.** A graph is called *weighted* if there exists a mapping $w : E \to \mathbb{N}$. This motivates the notation $G = (M, E, p, w)$.

For the investigations in this thesis weights in $\mathbb{N}$ are sufficient.

The fundamental substance for the investigations on cellular automata using graphs is framed in the following theorem.

Theorem 3.2.4 (Graph Representation of Topology II)**.** *The neighbourhood structure of a cellular automaton (second topological feature) can be interpreted as a directed graph.*

Proof (Unordered Neighbourhoods). Assume that the neighbourhood structure of a cellular automaton is defined through a neighbourhood mapping $\mathcal{N} : M \to \mathfrak{P}(M) : m \mapsto \mathcal{N}(m) = \{n_1, \ldots, n_{k_m}\}$. Define $E := \bigcup_{m \in M} \{(n, m) : n \in \mathcal{N}(m)\}$, which is a subset of $M \times M$ and thus renders $G := (M, E)$ a simple directed graph. □

Proof (Ordered Neighbourhoods). For ordered neighbourhoods on the other hand $\mathcal{N}(m) = (n_1, \ldots, n_{k_m}) \in \mathfrak{T}(M)$. Define $E := \bigcup_{m \in M} \{(n_i, m, i) : i = 1, \ldots, k_m\}$ and $w : e = (n, m, i) \mapsto i$. If a cell is contained multiple times within a neighbourhood we can not identify E with a subset of $M \times M$. In this case $p : e = (n, m, i) \mapsto (n, m)$ is not injective and we obtain a multigraph. □

In the case of ordered neighbourhoods the edges are weighted with the index of the corresponding neighbouring cell within the neighbourhood. For example if $\mathcal{N}(m) = (n_1, \ldots, n_k)$, the edge (n_i, m) has the weight i where $i \in \{1, \ldots, k\}$.

If it is required that a cell is contained multiple times within a neighbourhood, but we do not care for ordering, the mapping w can simply be ignored. An undirected graph corresponds to a cellular automaton with symmetric neighbourhoods.

Note that directed edges point from the neighbours towards the cell! The reason for this is that the direction represents the flow of information. We can conclude that the concepts of graphs and neighbourhood mappings are exchangeable.

3.2.1 Adjacency Mappings of Graphs

As mentioned before there exist different mathematical methods for describing the connections among vertices. This section deals with the concept of adjacency matrices respectively adjacency mappings and introduces *adjacency tensors*.

Let M be the set of all vertices. The basic idea of an adjacency matrix is the mapping

$$\mathcal{A} : M \times M \to \{0, 1\} : (m_1, m_2) \mapsto x, \tag{3.2.1}$$

indicating whether there is a connection from m_1 to m_2 ($x = 1$) or not ($x = 0$). We make the convention that the first argument is always the source of a connection and the second argument is the end point. Undirected graphs are characterised by symmetric adjacency mappings: $\mathcal{A}(m, n) = \mathcal{A}(n, m)$ for $m, n \in M$.

A multigraph can be represented by letting $\mathcal{A}(m_1, m_2)$ be the number of connections between m_1 and m_2. For graphs with weighted edges (for our purpose weights in $\mathbb{N}$ are sufficient) the adjacency mapping is also of type

$$\mathcal{A} : M \times M \to \mathbb{N} : (m_1, m_2) \mapsto x, \tag{3.2.2}$$

with the difference that x indicates the weight of the connection from m_1 to m_2. Consequently an adjacency mapping cannot fully represent a graph which is not simple and weighted at the same time.

Definition 3.2.5 (Adjacency Mapping)**.** We call $\mathcal{A}$ in all forms

$$\mathcal{A}(m_1, m_2) := \mathbb{I}_E\big((m_1, m_2)\big) \tag{3.2.3}$$

$$\mathcal{A}(m_1, m_2) := |\{e \in E : p(e) = (m_1, m_2)\}| \tag{3.2.4}$$

$$\mathcal{A}(m_1, m_2) := \begin{cases} w\big((m_1, m_2)\big) & (m_1, m_2) \in E \\ 0 & (m_1, m_2) \notin E \end{cases} \tag{3.2.5}$$

an *adjacency mapping*.

3.2.1.1 Matrix Representation

If the vertices are indexed – not to be confused with an indexing in the sense of ordinary cellular automata – with index set $I = \{1, \ldots, |M|\}$, we obtain an index

mapping $\mathcal{I} : M \to I$ analogously to ordinary cellular automata. Consequently an alternative version of the adjacency mapping is

$$\mathcal{A} \circ (\mathcal{I}^{-1}, \mathcal{I}^{-1}) : I \times I \to M \times M \to \mathbb{N} : (i,j) \mapsto (m_i, m_j) \mapsto a_{ij}. \tag{3.2.6}$$

Meaning that there is a directed connection from vertex m_i to vertex m_j if $a_{ij} > 0$.

A vertex with index i respectively the index i itself can be identified with the basis vector e_i of the $|M|$-dimensional vector space with scalar field $\{0,1\}$. This yields a matrix representation $A = (a_{ij})_{i,j \in I} \in \mathbb{N}^{|M| \times |M|} = \mathbb{N}^{I \times I}$ of the mapping $\mathcal{A}$.

$$\mathcal{A} : \begin{cases} M \times M & \to & \{0,1\}^{|M|} \times \{0,1\}^{|M|} & \to & \mathbb{N} \\ (m_i, m_j) & \mapsto & \left(\begin{pmatrix} \delta_{1,i} \\ \vdots \\ \delta_{|M|,i} \end{pmatrix}, \begin{pmatrix} \delta_{1,j} \\ \vdots \\ \delta_{|M|,j} \end{pmatrix} \right) & \mapsto & \begin{pmatrix} \delta_{1,i} \\ \vdots \\ \delta_{|M|,i} \end{pmatrix}^{\mathrm{T}} A \begin{pmatrix} \delta_{1,j} \\ \vdots \\ \delta_{|M|,j} \end{pmatrix} = a_{ij} \end{cases} \tag{3.2.7}$$

Definition 3.2.6 (Adjacency Matrix, compare [22])**.** A is called *adjacency matrix*.

According to our convention, a row (constant first index i) of the adjacency matrix A determines the outgoing connections to other vertices.

For an existing indexing $I = \{1, \ldots, |M|\}$ (compare basis) the adjacency mapping $\mathcal{A} \in \mathbb{N}^{M \times M}$ can be identified with the adjacency matrix $A \in \mathbb{N}^{I \times I}$.

3.2.1.2 Tensor Representation

Since for an ordinary cellular automaton the cells, which can be represented as vertices, feature a multidimensional indexing $I \subset \mathbb{Z}^d$, it is in our case practical to describe adjacency mappings using multidimensional arrays. Multidimensional arrays can be interpreted using the formalism of tensors. A tensor per se is independent of the basis. The representation of a tensor with respect to a certain basis is a multidimensional array. We will however refer to multidimensional arrays as *tensors with respect to a certain basis/indexing*.

Definition 3.2.7 (Adjacency Tensor)**.** Assume there exists a d-dimensional indexing with index set $I \subset \mathbb{Z}^d$ of the vertices $m \in M$ of a graph. The *tensor representation* $A = (a_{ij})_{i,j \in I} \in \mathbb{N}^{I \times I}$ of $\mathcal{A}$ with respect to the indexing I is defined through

$$\mathcal{A} : M \times M \to \mathbb{N} : (m_i, m_j) \mapsto a_{ij}. \tag{3.2.8}$$

Accordingly for a given indexing I the tensor representation $A \in \mathbb{N}^{I \times I}$ can be identified with the adjacency mapping $\mathcal{A} \in \mathbb{N}^{M \times M}$. We also call A *adjacency tensor* with respect to I and use the notation $\mathcal{A}(i,j)$.

3.2.1.3 Outlook

Usually matrices can be interpreted as linear mappings and tensors can be interpreted as multilinear mappings. Especially for weighted graphs this perception arises some questions: How is the underlying vector space defined (compare Section 3.1)? Could $\mathbb{Z}$ be a valid scalar field? What is the linear combination of two indices respectively vertices? In which sense can – if at all – $\mathcal{A}$ be interpreted as a bilinear mapping? This is also related to the perception of a tensor as either a multidimensional array or as a multilinear mapping.

If the introduction of dual spaces M^* respectively I^* can be motivated, an adjacency tensor can be interpreted as being of type $(1,1)$ and as an element of $M^* \otimes M$ respectively $I^* \otimes I$. The components of the tensor would then be written as $a^i_j = a^{i_1 \dots i_d}_{j_1 \dots j_d}$.

3.2.1.4 Characterisation of Graphs and Adjacency Mappings

Some of the following definitions are motivated by characteristics of matrices or tensors. If a property depends on the indexing of the vertices the concept of adjacency tensors together with the indexing must be used in the definition. For properties that are not influenced by the indexing of the vertices adjacency mappings can be used in the definition and we can talk of a graph-property, which is of course independent of the indexing.

Definition 3.2.8 (Composite Adjacency Mappings)**.** The *decomposition* and *addition of adjacency mappings* is defined as $\mathcal{A} = \mathcal{B} + \mathcal{C} + \mathcal{D} + \dots \in \mathbb{N}^{M \times M}$ with the requirement $\mathcal{B}(m,n) > 0 \Rightarrow \mathcal{C}(m,n) = 0, \mathcal{D}(m,n) = 0, \dots$.

Definition 3.2.9 (Symmetric Graph)**.** Let us call a graph or adjacency mapping *symmetric* if $\mathcal{A}(m,n) > 0 \;\Leftrightarrow\; \mathcal{A}(n,m) > 0$ for all $m, n \in M$.

Symmetric means that if n is a neighbour of m then m is a neighbour of n. For unordered neighbourhoods respectively nonweighted graphs the greater sign in Definition 3.2.9 can be replaced with an equal sign. Note that Definition 3.2.9 supersedes the use of *symmetric* as a geometric property of a neighbourhood in the context of indexed cellular automata (compare Definition 2.1.9).

Furthermore a directed graph with symmetric adjacency mapping can be interpreted as an undirected graph.

Definition 3.2.10 (Diagonal Structure of Adjacency Tensors)**.** Assume that the vertices of a graph are indexed with a d-dimensional index set $I \subset \mathbb{Z}^d$. An adjacency tensor is called *diagonally structured* if $\mathcal{A}(i,j) = \mathcal{A}(i+k, j+k)$ for all $k \in \mathbb{Z}^d$ such that $i + k \in I$ and $j + k \in I$ where $i, j \in I$.

Definition 3.2.11 (Regularity of Graphs, compare [22])**.** A weighted (directed) graph respectively the according adjacency mapping is called $\{1, \dots, k\}$-*regular*, if (at least) one of the following equal conditions is satisfied:

(i) Every vertex has an in- and outdegree of k and for every vertex $m \in M$ the sets $\{\mathcal{A}(m,n) : \mathcal{A}(m,n) > 0, n \in M\}$ as well as $\{\mathcal{A}(n,m) : \mathcal{A}(n,m) > 0, n \in M\}$ are permutations of respectively equal to the set $\{1,\ldots,k\}$.

(ii) There exists an indexing such that for every index $i \in I$ the tuple $(\mathcal{A}(i,j) : \mathcal{A}(i,j) > 0, j \in I)$ and for every index $j \in I$ the tuple $(\mathcal{A}(i,j) : \mathcal{A}(i,j) > 0, i \in I)$ are permutations of $(1,\ldots,k)$.

In the case of an adjacency matrix this structural feature can expressed as: Every column and every row must contain each of the values $\{1,\ldots,k\}$ exactly once. The remaining elements of the matrix are zero-valued.

Definition 3.2.12 (Not fully applying Properties of Graphs)**.** (i) For a property we use the prefix *almost* to indicate that it is not valid for all vertices M or indices I but only for a previously defined subset $M^\circ \subset M$ respectively $I^\circ \subset I$.

(ii) For a property we use the prefix *partially* to indicate that it is not valid for the adjacency mapping $\mathcal{A}$ but for a certain part $\mathcal{B}$ of the adjacency mapping $\mathcal{A} = \mathcal{B} + \mathcal{C} + \ldots$. The same applies to tensors representations $A + B + C + \ldots$

It is important to note that if a property of an adjacency mapping or tensor is valid partially then there exist additional connections that render the property unsatisfied. If those additional connections would not exists, the property would apply.

3.2.1.5 Distance Mappings

Another concept – similar to adjacency mappings – for describing the connections among the vertices of a graph are distance mappings.

Definition 3.2.13 (Distance Mapping [22])**.** The *distance mapping* $\mathcal{D} : M \times M \to \mathbb{N}$ describes the smallest number of edges that connects m to n. If it is not possible to reach n starting from m we define $\mathcal{D}(m,n) = \infty$.

For a cellular automaton with neighbourhood mapping $\mathcal{N}$ and distance mapping $\mathcal{D}$,

$$\mathcal{D}(m,n) = \min\{k \in \mathbb{N} \setminus \{0\} : n \in \mathcal{N}^k(m)\}. \tag{3.2.9}$$

If $n \in \mathcal{N}(m)$ then $\mathcal{D}(m,n) = 1$. $\mathcal{D}(m,n) = \infty \wedge \mathcal{D}(n,m) = \infty$ means that the prevailing neighbourhood structure is disconnected (Definition 3.0.15).

Like adjacency mappings also distance mappings feature a matrix or tensor representation given an existing indexing of the vertices. For weighted graphs the distance mapping is problematic since it gives no information about the weights of the edges.

3.2.2 Further Concepts of Graph Theory

This section mentions further concepts of graph theory which may be useful for characterising the topology (second topological feature) of cellular automata.

3.2.2.1 Neighbourhood Concept for Graphs

Since adjacency mappings and also distance mappings feature two input arguments, the idea of inverse neighbours (Definition 3.0.17) plays an important role for these techniques.

The term *neighbourhood* is used in graph theory with a slightly differing meaning.

Definition 3.2.14 (Neighbourhood Subgraph [22])**.** The neighbourhood of a vertex is the set of vertices connected to the vertex and the vertex itself. For each vertex the neighbourhood and the corresponding edges between the vertex and the neighbours and the edges among the neighbours form a subgraph, the *neighbourhood subgraph*.

The neighbourhood subgraph corresponds to the graph that is *induced* by the original graph on the neighbourhood [22].

It is very important to note that in contrast to a neighbourhood in the sense of cellular automata, a neighbourhood subgraph also contains information about the inverse neighbours of a vertex and the connections among the neighbours of the vertex.

Ordinary cellular automata feature a unique neighbourhood structure for (almost) all cells. This feature can be formalised for graphs by requiring that the neighbourhood subgraphs of (almost) all vertices are *isomorphic*.

3.2.2.2 Isomorphy

In graph theory the concept of isomorphy is defined as:

Definition 3.2.15 (Graph Isomorphy [22])**.** Two graphs are called *isomorphic* to each other if there exists a bijective mapping between the vertices of both graphs such that two vertices of the first graph are connected if and only if the corresponding vertices of the second graph are connected in the same way (weight and direction).

This property can also be formulated using adjacency mappings or tensors:

(i) Two graphs are isomorphic if and only if there exists a bijective mapping $f : M_1 \to M_2$ and the adjacency mappings $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $\mathcal{A}_1(m, n) = \mathcal{A}_2(f(m), f(n))$ for all $m, n \in M_1$.

(ii) Two graphs are isomorphic if and only if there exist index mappings with a common index set I and the corresponding adjacency tensor representations $A_1 \in \mathbb{N}^{I \times I}$ and $A_2 \in \mathbb{N}^{I \times I}$ are equal.

(iii) Two graphs with index mappings with common index set I are isomorphic if and only if the adjacency tensor representations ($A_1, A_2 \in \mathbb{N}^{I \times I}$) are permutation similar.

As mentioned before, the concept of isomorphy can also be applied to neighbourhood subgraphs. The following definition formalises the feature of isomorphic neighbourhood subgraphs.

Definition 3.2.16 (Local Isomorphy of Graphs [22])**.** A graph G is called *locally isomorphic* to a graph N if the neighbourhood subgraphs of all vertices of G are isomorphic to N.

Furthermore since the neighbourhood structure of a cellular automaton can be identified with a graph, the concept of a *uniform neighbourhood structure* can be interpreted as isomorphy of all neighbourhood subgraphs.

Let M_a and M_b be the sets of vertices respectively cells of two nonweighted graphs respectively cellular automata with unordered neighbourhoods denoted a and b. The cellular automata have the same second topological feature if and only if there exists a bijective mapping between M_a and M_b such that

$$\forall m, n \in M_a : \qquad n \in \mathcal{N}_a(m) \quad \Longleftrightarrow \quad f(n) \in \mathcal{N}_b(f(m)). \tag{3.2.10}$$

Especially translations can be regarded as isomorphisms.

3.2.2.3 Cycles and Cliques

Three or more vertices form an undirected cycle when there exists an undirected path among them such that vertex 1 is connected with vertex 2, vertex 2 is connected with vertex 3, and so forth and the last vertex is connected with the first vertex [22]. Compare this concept with *paths* as in Definition 3.0.19.

Definition 3.2.17 (Representation of Graph Cycles)**.** A *directed cycle* or sometimes just *cycle* is characterised by a tuple of vertices $(m_1, \ldots, m_k)$ together with a tuple of weights (> 0)

$$(\alpha_{12}, \alpha_{23}, \ldots, \alpha_{1k}) \tag{3.2.11}$$

where α_{ij} describes the weight of the connection from m_i to m_j. A *undirected cycle* can be characterised by a tuple of vertices $(m_1, \ldots, m_k)$ together with a tuple

$$\big((\alpha_{12}, \alpha_{21}), (\alpha_{23}, \alpha_{32}), \ldots, (\alpha_{k1}, \alpha_{1k})\big) \tag{3.2.12}$$

describing the weights of the connecting edges with respect to their direction. In this case at least one of α_{ij} and α_{ji} is always greater than 0.

According to this definition, undirected cycles are invariant under change of direction and starting point. We only observe *simple cycles*, which means that a vertex can only be contained once in a cycle.

Definition 3.2.18 (Acyclic Graph [22])**.** A graph that does not contain any (directed!) cycle is called *acyclic graph*.

Definition 3.2.19 (Complete Graph [22])**.** A graph or subgraph is *complete* if there exists a (in the case of a directed graph – bidirectional) direct connection between each two nodes.

Definition 3.2.20 (Clique [22])**.** For an undirected graph a maximal complete set of vertices forms a *clique*.

3.2.3 Graph Representation of Ordinary Cellular Automata

According to Theorem 3.2.4 a neighbourhood mapping is equivalent to an adjacency mapping. Because the neighbourhoods in indexed cellular automata – disregarding an eventual third topological feature – are described through a relative index tuple $J = (j_1, \dots, j_k)$ and the corresponding index translation $\mathcal{T}_J : I \to (\mathbb{Z}^d)^k : i \mapsto i+J = (i+j_1, \dots, i+j_k)$ the adjacency tensor $\mathcal{A} : I \times I \to \mathbb{N}$ (with respect to the indexing of the cellular automaton I) is defined by

$$\mathcal{A}(i, i+j) = \begin{cases} \alpha \quad \text{where} \quad \alpha \in \{1, \dots, k\} & \iff \quad j = j_\alpha \\ 0 & \iff \quad j \notin J \end{cases} \tag{3.2.13}$$

for all $i \in I$ and $j \in \mathbb{Z}^d$ such that $i + j \in I$.

The inverse neighbourhood of a cell is given by $i - J = (i - j_1, \dots, i - j_k)$. Consequently for $i \in I$ and $j \in \mathbb{Z}^d$ such that $i - j \in I$ the following must apply:

$$\mathcal{A}(i-j, i) = \begin{cases} \alpha \quad \text{where} \quad \alpha \in \{1, \dots, k\} & \iff \quad j = j_\alpha \\ 0 & \iff \quad j \notin J \end{cases} \tag{3.2.14}$$

Definition 3.2.21. For ordinary cellular automata, indices with nondegraded neighbourhoods are summarised as $I^\circ = \{i \in I : \ i + J \subseteq I\}$ (compare Definition 2.1.16). Indices with nondegraded inverse neighbourhood are summarised as ${}^\circ I = \{i \in I : \ i - J \subseteq I\}$.

The following proposition characterises the adjacency mapping of an indexed cellular automaton without distortion of the lattice.

Proposition 3.2.22 (Adjacency Mapping for Indexed Cells without Topology III)**.** *For an indexed cellular automaton with indexing I and $J = (j_1, \dots, j_k)$ without distortion of the lattice the adjacency mapping $\mathcal{A}$ satisfies the following conditions.*

(i) The tensor representation of $\mathcal{A}$ with respect to I is diagonally structured.

(ii) $\mathcal{A}$ is almost $\{1, \dots, k\}$-regular.

Proof. (i) Let $(i, j) \in I \times I$ and $h \in \mathbb{Z}^d$ such that $i + h \in I$ and $j + h \in I$. If $\mathcal{A}(i, j) = \alpha$ for an $\alpha \in \{1, \dots, k\}$, j can be written as $j = i + j_\alpha$. Accordingly $\mathcal{A}(i+h, j+h) = \mathcal{A}(i+h, i+h+j_\alpha)$ which is equal to α since $(i+h)+j_\alpha = j+h \in I$. If $\mathcal{A}(i, j) = 0$ then $j \notin i + J$ and accordingly $j + h \notin i + h + J$ which means $\mathcal{A}(i+h, j+h) = 0$.

(ii) The following proof depends on the indexing of the vertices/cells while the concept of regularity actually does not i.e. is permutation invariant. Let $i \in I^\circ$. Since $i + j_\alpha \in I$ we have $\alpha \in \{\mathcal{A}(i, j) : j \in I\}$ for all $\alpha \in \{1, \dots, k\}$. If $\mathcal{A}(i, j) \notin \{0, 1, \dots, k\}$ for a $j \in J$, $\mathcal{A}$ can not be a valid representation of the neighbourhood structure. If $\mathcal{A}(i, j_1) = \mathcal{A}(i, j_2) \neq 0$ for $j_1, j_2 \in I$ with $j_1 \neq j_2$ we would have $j_1 = i + j_\alpha$ and $j_2 = i + j_\beta$ with $j_\alpha \neq j_\beta$ but $\alpha = \beta$. The

equivalent statements can be made for $j \in {}^{\circ}I$. According to Definition 3.2.11 (ii) $\mathcal{A} : I \times I \to \mathbb{N}$ is almost $\{1, \ldots, k\}$-regular.

□

When the lattice is distorted through a generalised index translation $\mathcal{T}_{\tau,J} : I \to I^k$ with $\tau : I \times \mathbb{Z}^d \to I$, the adjacency mapping can be written as $\mathcal{A} + \mathcal{B}$ where $\mathcal{B} : M \times M \to \mathbb{N}$ represents the connections that are not defined by the index tuple J. We could say that the neighbourhood structure of the cellular automaton is partially represented by $\mathcal{A}$. In other words $\mathcal{B}$ additionally defines the connections of vertices/cells which would otherwise have degraded neighbourhoods. We can also say that $\mathcal{A}$ represents the second topological feature of the cellular automaton and $\mathcal{B}$ represents the third topological feature.

This notation can only be used under the condition that $\mathcal{A}(m, n) > 0 \Rightarrow \mathcal{B}(m, n) = 0$, which is somehow related to the requirement that a default ordered neighbourhood can contain a cell only once. This requirement however excludes some configurations with a combination of reflective boundary conditions and reflexive neighbourhoods. In this case a cell could be contained within a neighbourhood more than once.

Proposition 3.2.23 (Adjacency Mapping for Indexed Cells with Topology III)**.** *For an indexed cellular automaton with indexing I, index tuple $J = (j_1, \ldots, j_k)$ and suitable (see discussion above) $\tau : I \times \mathbb{Z}^d \to I$ the adjacency mapping can be decomposed as $\mathcal{A} + \mathcal{B}$ in a way that the following conditions are satisfied.*

(i) The tensor representation of $\mathcal{A}$ with respect to I is diagonally structured.

(ii) $\mathcal{A} + \mathcal{B}$ is $\{1, \ldots, k\}$-regular.

Proof. (i) See proof of Proposition 3.2.22.

(ii) Because τ is defined in an appropriate way, every cell has the same number of (equally indexed) neighbours and inverse neighbours.

□

In other words the adjacency mapping is $\{1, \ldots, k\}$-regular and the adjacency tensor is partially diagonally structured.

3.2.4 A Different Graph Representation

It is also possible to embed the iterative levels of a cellular automaton into a single graph by observing the nodes $T \times M$ and the edges

$$E := \bigcup_{t \in T, m \in M} \bigcup_{n \in \mathcal{N}(m)} \big\{\big((t-1, n), (t, m)\big)\big\}. \tag{3.2.15}$$

This graph representation is applied in Section 5.2. A graphical representation can be found in Figure 5.1. This section only serves the purpose of completeness.

3.3 Interpretation as Topological Spaces

For cellular automata with indexed cells there exists a natural concept of proximity (first topological feature). Also a neighbourhood mapping (second topological feature) introduces a proximity measure. Both concepts can imply a topological structure in the sense of open sets on the set of cells.

To avoid confusion we will not use the term *neighbourhood* as a concept from the field of topology. However a *neighbourhood filter* on M can be constructed with the neighbourhoods – induced by a neighbourhood mapping – as subbase. Define $\mathfrak{N}' := \{\mathcal{N}(m) \cup \{m\} : m \in M\}$.

Definition 3.3.1 (Directed Neighbourhood Topology)**.** For a neighbourhood mapping $\mathcal{N} : M \to \mathfrak{P}(M)$ the *directed neighbourhood topology* is defined as the smallest topology containing $\mathfrak{N}'$ as open sets.

Proposition 3.3.2 (Directed Neighbourhood Topology)**.** *The directed neighbourhood topology $\mathfrak{O}'$ is not necessarily the discrete topology on M.*

Proof. Assume m is not part of the neighbourhood of any other cell and that $\mathcal{N}(m)$ contains other cells apart from m. Accordingly m is only contained in $\mathcal{N}(m) \cup \{m\}$. m can not be obtained as a intersections of elements from $\mathfrak{N}'$ and therefore is not an open set. □

Define $\mathfrak{N} := \{\mathcal{N}(m) \cup \{m\} \cup \{n : m \in \mathcal{N}(n)\} : m \in M\}$.

Definition 3.3.3 (Neighbourhood Topology)**.** For a neighbourhood mapping $\mathcal{N} : M \to \mathfrak{P}(M)$ the *neighbourhood topology* $\mathfrak{O}$ is defined as the unique topology which is generated by $\mathfrak{N}$ as subbase.

Proposition 3.3.4 (Neighbourhood Topology)**.** *The topology $\mathfrak{O}$ is not necessarily the discrete topology.*

Proof. Assume m has only one neighbour n and is at most a neighbour of n. For all sets N in $\mathfrak{N}$ the following is true: $m \in N \Rightarrow n \in N$. m can not be obtained from intersections of elements from $\mathfrak{N}$. □

A measure for proximity based on the second topological feature is self-evident.

Definition 3.3.5 (Coarse Neighbourhood Metric)**.** For a given neighbourhood mapping $\mathcal{N} : M \to \mathfrak{P}(M)$ the *coarse neighbourhood metric* is defined as

$$d'_{\mathcal{N}} : (m, n) \mapsto \begin{cases} 0 & m = n \\ 1 & m \in \mathcal{N}(n) \vee n \in \mathcal{N}(m) \\ 2 & m \notin \mathcal{N}(n) \wedge n \notin \mathcal{N}(m) \end{cases} . \tag{3.3.1}$$

It is actually necessary to explicitly define $d'_{\mathcal{N}}(m,m) := 0$, $m \in M$, because for reflexive neighbourhoods $m \in \mathcal{N}(m)$ and for nonreflexive neighbourhood mappings $m \notin \mathcal{N}(m)$. It is however easy to prove that $d'_{\mathcal{N}}$ is a metric.

Another possible distance measure describes the number of neighbourhoods that has to be traversed in order to reach one cell starting from another.

Definition 3.3.6 (Neighbourhood Metric)**.** For a given neighbourhood mapping $\mathcal{N} : M \to \mathfrak{P}(M)$ the *neighbourhood metric* $d_{\mathcal{N}}(n,m)$ is defined as the minimum of the lengths of all undirected neighbourhood paths between m and n. If there exists no undirected path between m and n we define $d_{\mathcal{N}}(n,m) := \infty$.

Let $(n_1, \ldots, n_k)$ be the undirected path with minimum length between n and m with $n \neq m$ then

$$d_{\mathcal{N}}(n,m) = \sum_{i=1}^{k-1} d'_{\mathcal{N}}(n_i, n_{i+1}) = k-1. \tag{3.3.2}$$

Furthermore for arbitrary n and m in M

$$d_{\mathcal{N}}(n,m) \leq \min\ \left\{k \in \mathbb{N} : n \in \mathcal{N}^k(m) \vee m \in \mathcal{N}^k(n)\right\} \tag{3.3.3}$$

and if the neighbourhood structure is disconnected it can happen that $d_{\mathcal{N}}(n,m) = \infty$. In accordance to Definition 3.2.13, for each two cells $m, n \in M$

$$d_{\mathcal{N}}(m,n) \leq \mathcal{D}(m,n). \tag{3.3.4}$$

We have to show that $d_{\mathcal{N}}$ is a metric on M: Symmetry follows since undirected paths are observed. Identity is clear. The triangle inequality follows from the minimality of the lengths of all connecting paths.

Because for $0 < \varepsilon < 1$ the ε-ball with centre m is equal to $\{m\}$ for $d'_{\mathcal{N}}$ and $d_{\mathcal{N}}$, both metrics induce the discrete topology on M.

Another way to introduce a topology based on the second topological feature is provided by *topological graph theory*, which embeds the vertices and edges on a surface.

For indexed cells different metrics can be transferred from $I \subset \mathbb{Z}^d$ to M by interpreting $\mathcal{I} : M \to I$ as a continuous mapping.

$$d_{\mathcal{I},\max}(m_i, m_j) = \max_{\alpha \in \{1,\ldots,d\}} |i_\alpha - j_\alpha| \tag{3.3.5}$$

$$d_{\mathcal{I},\mathrm{sum}}(m_i, m_j) = \sum_{\alpha \in \{1,\ldots,d\}} |i_\alpha - j_\alpha| \tag{3.3.6}$$

$$d_{\mathcal{I},2}(m_i, m_j) = \Big(\sum_{\alpha \in \{1,\ldots,d\}} |i_\alpha - j_\alpha|^2 \Big)^{\frac{1}{2}} \tag{3.3.7}$$

All these metrics generate the discrete topology on M.

As discussed in Proposition 2.1.10 if the neighbourhood structure of an indexed cellular automaton (topology II) is described through a metric on I (topology I), there always exists an index translation that delivers the same neighbourhood structure. A neighbourhood defined through a metric is always reflexive and symmetric.

3.4 Implicit Alignment of Unaligned Cellular Automata

For aligned cellular automata with uniform (connected) neighbourhood structure – i.e. ordinary cellular automata (Theorem 3.1.8) – the neighbourhood mapping can be formulated using relative neighbourhood relations. In other words, for aligned cellular automata with uniform connected neighbourhood structure the second topological feature can be formulated based upon the first topological feature.

Conversely for a given neighbourhood mapping which satisfies some kind of uniformity it may be possible to find an alignment, for which the given neighbourhood structure is uniform and connected such that also in this case the second topological feature can be formulated using the first topological feature.

Example 3.4.1. Assume an unaligned cellular automaton with a configuration that makes the new state of a cell dependent on the state of only one other cell in a way such that every cell is the neighbour of another cell. We could align the cells on a one dimensional index set $I \subseteq \mathbb{Z}$ and use the relative index tuple $(-1) = -1 =: J$ to indicate that for a cell m_i the neighbouring cell has index $i - 1$.

This section aims at finding and characterising certain unaligned cellular automata which can be interpreted as ordinary cellular automata. For an unaligned cellular automaton to be interpreted as an indexed cellular automaton it is crucial that the neighbourhoods of all cells are somehow uniformly structured. Some cells with degraded neighbourhoods may be excepted from this requirement.

3.4.1 Graph Theoretic Approach

In Section 3.2 the necessary tools for interpreting the second topological feature of cellular automata in the context of graphs were presented. Section 3.2.3 provides especially a characterisation of ordinary cellular automata using adjacency mappings and tensors. Based thereon we can find conditions under which an unaligned cellular automaton can be interpreted as an ordinary cellular automaton.

Theorem 3.4.2 (Characterisation of Ordinary Cellular Automata without Topology III based on the Adjacency Mapping)**.** *If for an unaligned cellular automaton with ordered neighbourhoods there exists a d-dimensional indexing of the cells with index set I such that*

(i) the adjacency tensor with respect to I is diagonally structured and

(ii) the adjacency mapping is almost $\{1, \ldots, k\}$-regular

then the cellular automaton can be interpreted as an ordinary cellular automaton with index set I and relative index tuple $J = (j_1, \ldots, j_k)$ as defined in the proof.

Proof. Let $i \in I^\circ \subset I$ be the indices for which $(\mathcal{A}(j,i) : \mathcal{A}(j,i) > 0, j \in I)$ is a permutation of $\{1,\dots,k\}$. Almost every index satisfies this requirement. Because $\mathcal{A}$ is diagonally structured $\tilde{J} = \{j \in \mathbb{Z}^d : i + j \in I, \mathcal{A}(i+j,i) > 0\}$ is the same for all $i \in I^\circ$. Furthermore $\tilde{J}$ can be ordered as $J = (j_1,\dots,j_k)$ in a way such that $\mathcal{A}(i+j_\alpha,i) = \alpha$ for $\alpha \in \{1,\dots,k\}$ and for all $i \in I^\circ$. □

If the index mapping is denoted $\mathcal{I}$, the original neighbourhood mapping satisfies $\mathcal{N}(m_i) = \mathcal{I}^{-1} \circ \mathcal{T}_J \circ \mathcal{I}(m_i) = (m_{i+j_1},\dots,m_{i+j_k})$. Cells with $\mathcal{I}(m) \notin I^\circ$ have a degraded neighbourhood. For almost all cells the neighbourhood size is k.

Theorem 3.4.3 (Characterisation of Ordinary Cellular Automata with Topology III based on the Adjacency Mapping)**.** *If for an unaligned cellular automaton with ordered neighbourhoods there exists a d-dimensional indexing with index set I such that*

(i) the adjacency tensor is partially diagonally structured with k diagonals[1] *and*

(ii) the adjacency mapping is $\{1,\dots,k\}$-regular

then the cellular automaton can be interpreted as an indexed cellular automaton with index set I, relative index tuple $J = (j_1,\dots,j_k)$ as defined in the proof and lattice distortion $\tau : I \times \mathbb{Z}^d \to I$ also as defined in the proof.

Proof. Let $\mathcal{A}$ be the diagonally structured part of the adjacency tensor and $\mathcal{B}$ the remaining part. We can apply Theorem 3.4.2 on $\mathcal{A}$ and obtain a relative index tuple $J = (j_1,\dots,j_k)$. Let $i \in I$. For $i + j_\alpha \notin I$ where $j_\alpha \in J$ (respectively $\alpha \in \{1,\dots,k\}$) there exists $j \in I$ with $\mathcal{B}(j,i) = \alpha$ because $\mathcal{A} + \mathcal{B}$ is $\{1,\dots,k\}$-regular. Define $\tau(i,j_\alpha) = j$. □

The index mapping $\mathcal{I}$ is again defined through the given indexing I and the original neighbourhood mapping satisfies $\mathcal{N}(m_i) = \mathcal{I}^{-1} \circ \mathcal{T}_{\tau,J} \circ \mathcal{I}(m_i)$ where $\mathcal{T}_{\tau,J} : I \to I^k : i \mapsto (i_1,\dots,i_k)$ with

$$i_\alpha = \begin{cases} i + j_\alpha & i + j_\alpha \in I \\ \tau(i,j_\alpha) & i + j_\alpha \notin I \end{cases} \qquad \alpha \in \{1,\dots,k\}.$$

The existence of an appropriate indexing is a prerequisite in Theorem 3.4.2 and Theorem 3.4.3. In particular an index mapping of arbitrary dimension is needed such that the adjacency tensor with respect to this indexing is diagonally structured. Both theorems can be reformulated in a way that does not explicitly include a distinct index mapping as precondition. For simplicity only the version without distortion of the lattice i.e. with degraded neighbourhoods (Theorem 3.4.2) is reformulated here.

Corollary 3.4.4. *If for an unaligned cellular automaton with ordered neighbourhoods*

[1] This is necessary to ensure that the remainder of the adjacency mapping does not define an additional nonregular neighbour for every cell.

(i) the adjacency mapping has a tensor representation that is permutation similar to a diagonally structured tensor and

(ii) the adjacency mapping is almost $\{1, \ldots, k\}$-regular

then the cellular automaton is equivalent to an ordinary cellular automaton.

Proof. This corollary is actually equivalent to Theorem 3.4.2 with the only difference that the existence of an appropriate indexing is a consequence of the permutation similarity in (i). □

How can an appropriate indexing, which is determined by a dimension d, an index set $I \subset \mathbb{Z}^d$ and an (bijective) index mapping $\mathcal{I} : M \to I$ be found? Is there a method to test if an adjacency tensor is permutation similar to an arbitrary diagonally structured tensor?

3.4.1.1 Outlook: Technical Implementation

In the most common form, the adjacency mapping is formulated as an adjacency matrix, which corresponds to a one dimensional indexing with $I = \{1, \ldots, |M|\}$. As we know, regularity is invariant under permutation (index transformation). Accordingly the task is to search for an index transformation that yields a diagonally structured adjacency tensor.

Possible technical approaches:

- minimisation of the difference of the indices of all connected vertices
- graph canonisation, lexicographical ordering
- analysis of the elements of the adjacency matrix separately for every weight, identification of symmetric part

3.4.1.2 Outlook: Unordered Neighbourhoods

Assume an unaligned cellular automaton with update rules that do not depend on the ordering of cells within the neighbourhoods and a neighbourhood mapping given as a function $\mathcal{N} : M \to \mathfrak{P}(M)$ where the neighbourhood size is uniform for all cells. Accordingly the adjacency mapping is given as a function $\mathcal{A} : M \times M \to \{0, 1\}$, which contains much less information than the corresponding adjacency mapping $\mathcal{A} : M \times M \to \{1, \ldots, k\}$ from Section 3.4.1.

Nevertheless it should be sufficient to find a diagonally structured tensor representation of the k-regular adjacency mapping to show that the cells can be aligned on a index set and that there exists an index translation that delivers the original neighbourhood mapping.

Does there exist a simpler approach which is especially suitable for unordered neighbourhoods?

3.4.2 Other Graph-Theoretic Identification Approaches

In graph theory there exists further concepts that could be useful for identifying a regular alignment in unaligned cellular automata such that the neighbourhood structure is uniform. Instead of observing the adjacency mapping it may be possible to find requirements for the distance mapping. *Spectral graph theory* may provide an alternative approach since especially the spectrum and other features of matrices/tensors are index invariant. Furthermore also the concept of *cycle spaces* could be viable. The following paragraph outlines a straight forward approach based on cycles and isomorphy.

3.4.2.1 Cycles, Formulation as Graph Isomorphism Problem

In a previous section the feature of $\{1, \ldots, k\}$-regularity of the adjacency mapping in combination with the diagonal structure of the tensor representation were used to characterise the adjacency mappings of ordinary cellular automata. Observing the adjacency mapping gives somehow a global or macroscopic view of the second topological feature of a cellular automaton. Observing the neighbourhood subgraphs of individual cells on the other hand gives a microscopic view of the neighbourhood structure. On a microscopic scale the neighbourhoods of ordinary cellular automata can be characterised using the concept of graph isomorphy.

As mentioned earlier the neighbourhood subgraphs of an ordinary cellular automaton are (almost) all isomorphic ta a common graph. We call the vertex that represents the cell whose neighbourhood is described as the *central vertex* m and denote the neighbouring vertices as $m_1, m_2, \ldots$. The common neighbourhood subgraphs features the following properties:

(i) There exist k outgoing edges with weights $1, \ldots, k$ and there exist k incoming edges with weights $1, \ldots, k$ (compare $\{1, \ldots, k\}$-regularity for the central vertex).

(ii) Assume there exists a bidirectional connection, then there exists another (inverse) bidirectional connection. If a bidirectional connection has the incoming weight α_1 and the outgoing weight α_2 then there exists another bidirectional connection with incoming weight α_2 and outgoing weight α_1. This forbids the existence of bidirectional connections with the same weight for incoming and outgoing direction.

(iii) If there exists an (undirected) cycle (m, m_1, m_2) with weights

$$\big((\alpha_{01}, \alpha_{10}), (\alpha_{12}, \alpha_{21}), (\alpha_{20}, \alpha_{02})\big) \tag{3.4.1}$$

– i.e. a connection between two neighbours – then there exists another cycle (m, m_a, m_b) with weights

$$\big((\alpha_{12}, \alpha_{21}), (\alpha_{20}, \alpha_{02}), (\alpha_{01}, \alpha_{10})\big). \tag{3.4.2}$$

As a consequence there also exists a third cycle with weights

$$\big((\alpha_{20},\alpha_{02}),(\alpha_{01},\alpha_{10}),(\alpha_{12},\alpha_{21})\big). \tag{3.4.3}$$

(iv) The last property must also apply to cycles with four or more vertices.

3.4.3 Vector Space Approach

Assume a cellular automaton with connected ordered neighbourhood mapping $\mathcal{N} : M \to M^k$. For all cells $m \in M$ we assume that the relative neighbourhood relations can represented as $\mathcal{N}(m) - m = (b_{m,1},\ldots,b_{m,k})$ where $b_{m,i}, i = 1,\ldots,k$ are basis vectors from an abstract finite dimensional vector space – or more correctly module – with dimension $d \geq k$ and basis B.

For each two cells $m_x, m_y \in M$ there exists at least one directed path (Definition 3.0.18) – without loss of generality – from m_x to m_y, ($m_x = n_1, n_2, \ldots, n_{l_{xy}+1} = m_y$) with length l_{xy}. For every pair $n_{i+1} - n_i$ where $i \in \{1,\ldots,l_{xy}\}$ there exists a vector representation $n_{i+1} - n_i = \sum_{b\in B} \alpha_{xyib} b$. Together for every two cells $m_x, m_y \in M$ there exists a path defined by

$$m_y - m_x := \sum_{i=1}^{l_{xy}} \sum_{b\in B} \alpha_{xyib} b = \sum_{b\in B} \Big(\sum_{i=1}^{l_{xy}} \alpha_{xyib} \Big) b = \sum_{b\in B} \alpha_{xyb} b \tag{3.4.4}$$

where $\alpha_{xyb} = \sum_{i=1}^{l_{xy}} \alpha_{xyib}$.

Let us assume that there exist at least k different paths from m_x to m_y and use the notation

$$\overrightarrow{m_x m_y} = \left\{ \sum_{b\in B} \alpha_{xyb1} b, \sum_{b\in B} \alpha_{xyb2} b, \ldots \right\} = \left\{ \begin{pmatrix} \alpha_{xy11} \\ \vdots \\ \alpha_{xyd1} \end{pmatrix}, \begin{pmatrix} \alpha_{xy12} \\ \vdots \\ \alpha_{xyd2} \end{pmatrix}, \ldots \right\} \tag{3.4.5}$$

to accumulate these paths.

Consequently for every two cells we obtain a matrix representation of their connecting paths

$$\overrightarrow{mn} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1k} & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots \\ \alpha_{d1} & \alpha_{d2} & \cdots & \alpha_{dk} & \cdots \end{pmatrix}. \tag{3.4.6}$$

By eliminating linearly dependent rows, the set of basis vectors can be reduced. This means that for these two cells the connecting paths and the *local* vector space structure of the neighbourhoods are compatible. If for each two cells the same new reduced (actual) basis $\tilde{B} \subseteq B$ i.e. the same linear dependencies are obtained, the cellular automaton can be regarded as an aligned cellular automaton.

Theorem 3.4.5 (Construction of an Alignment)**.** *If for each two $m, n \in M$ (with nondegraded neighbourhoods) the same (reduced) basis $\tilde{B} \subseteq B$ is obtained then $\tilde{B}$ spans the difference space $M - M = V = \operatorname{span} \tilde{B}$.*

Proof. If the condition is satisfied, we can construct the same maximal linearly independent set of vectors $b_1, \ldots, b_{\tilde{d}}$ ($\tilde{d} \leq k$) from $v_1, \ldots, v_k$ independently of the choice of m and n. For arbitrary $m, n \in M$ we accordingly have a unique representation of $n - m = \sum_{i=1}^{k} \alpha_{1i} v_i = \sum_{i=1}^{k} \alpha_{2i} v_i = \ldots$, namely $n - m = \sum_{i=1}^{\tilde{d}} \beta_i b_i$. This renders $\{b_1, \ldots, b_{\tilde{d}}\}$ a basis of V. □

Corollary 3.4.6 (Characterisation of Ordinary Cellular Automata under the Condition of a Connected Neighbourhood Structure)**.** *Since for the resulting alignment the neighbourhood mapping is uniform by construction, there exists an indexing of the cells according to Theorem 3.1.8.*

3.5 Conclusions and Outlook

This chapter is concluded with some remarks on the previously discussed "topological" concepts.

A very interesting aspect is that corresponding to the differentiation between a priori and a posteriori alignment of cells, we can distinguish concepts that are more suitable for the former respectively the latter approach (Table 3.1).

	alignment in the context of modelling		interpretation of alignment	
	a priori	a posteriori	geometric spatial	abstract logical
index based approach	•		•	
vector space approach	•		•	
graph approach	•	•		•

Table 3.1: Classification of different concepts for describing the topological structure of cellular automata. This classification is however only valid in the most usual cases.

Besides the discussed methods other approaches for describing the topology of cellular automata are possible: Graph theoretic concepts like *hypergraphs* or *stochastic acyclic graphs* with *forbidden circular conditional probability* could probably be applied. A graph interpretation also induces a *complexity measure* on cellular automata and the *bandwidth* of an adjacency matrix/tensor could be used as a measure for "locality" of the neighbourhoods. Furthermore the spatial domain that is discretised into cells could have a representation as a manifold.

3.5.1 Generalised Definition of Cellular Automata

This short section features a generalised definition of cellular automata, which incorporates the different concepts of the topological structure of cellular automata which were discussed until now.

Definition 3.5.1 (Generalised Definition of Cellular Automaton)**.** A cellular automaton is completely defined by

(i) a set of cells M with optional alignment defined by either
   (a) an indexing $(I, \mathcal{I})$ as defined in Section 2.1,
   (b) a difference space V as defined in Section 3.1 or
   (c) a graphical structure G as defined in Section 3.2,

(ii) a neighbourhood mapping $\mathcal{N}$, which is either
   (a) composed of a relative index tuple J and, if the geometry of the cellular automaton shall be manipulated, additionally a corresponding mapping τ,
   (b) a tuple of vectors from V,
   (c) composed of the neighbouring vertices in G or
   (d) explicitly defined through a mapping $\mathcal{N}$,

(iii) a set of possible states $\mathbb{S}$ and

(iv) a self-contained update rule $\mathcal{F}$.

The tuple $((M, *), \mathcal{N}, \mathbb{S}, \mathcal{F})$ can serve as a short notation.

Chapter 4

Cellular Automata and Continuous Evolution Systems

Continuous systems suitable for applying cellular automaton modelling or approximation approaches are in the ideal case comparable to "cellular automata with continuous cell-space" and of course feature a continuous time. Hence in Section 4.2 *evolution systems* are defined using the paradigms of cellular automata.

In Section 4.2.3.1 we see that *linear evolution equations* and *parabolic differential equations* can be investigated in this context.

4.1 Alternative Definition of Cellular Automata

This section presents a top-down definition approach, which to a certain extent contradicts the basic principles of cellular automata but on the other hand proves to be useful in connection with special types of cellular automata (e.g. Section 4.2 or Section 5.2) and their analysis.

Since $\mathcal{S}$ is not only a mapping but can also be interpreted as an element of $\mathbb{S}^M$, we can introduce the notation

$$\mathcal{S} = \big(\mathcal{S}(m)\big)_{m\in M} = \prod_{m\in M} \mathcal{S}(m) \tag{4.1.1}$$

and accordingly

$$\mathcal{S}\circ\mathcal{N}(m) = \big(\mathcal{S}(n)\big)_{n\in\mathcal{N}(m)} = \prod_{n\in\mathcal{N}(m)} \mathcal{S}(n). \tag{4.1.2}$$

$\mathcal{S}\circ\mathcal{N}(m)$ is a more compact notation but – as mentioned earlier – not completely correct ($\mathcal{S} : \mathfrak{T}(M) \to \mathfrak{T}(\mathbb{S})$).

Definition 4.1.1 (Projection Operator)**.** For $\mathcal{S} \in \mathbb{S}^M$ and $N \subseteq M$ define the *projection operator* $\mathrm{proj}_N : \mathbb{S}^M \to \mathbb{S}^N$ as

$$\mathrm{proj}_N \mathcal{S} = \prod_{n\in N} \mathcal{S}(n). \tag{4.1.3}$$

If $\mathcal{S}$ is a mapping then also $\mathrm{proj}_N \mathcal{S}$ can be used as a mapping from N to $\mathbb{S}$. In this case also the notation $\mathcal{S}|_N$ can be used.

4.1.1 Scalar Evolution Operators

Let $\mathcal{E} : \mathfrak{S} \to \mathfrak{S}$ (where $\mathfrak{S} \subseteq \mathbb{S}^M$) be the global evolution operator of a cellular automaton $(M, \mathcal{N}, \mathbb{S}, \mathcal{F})$.

Definition 4.1.2 (Scalar Evolution Operator)**.** Because $\mathcal{E}$ is defined through an update function $\mathcal{F}$ and a neighbourhood mapping $\mathcal{N}$ such that $\mathcal{E}\mathcal{S} = \mathcal{F} \circ \mathcal{S} \circ \mathcal{N}$, we can define *scalar evolution operators* $\mathcal{L}_m : \mathfrak{S} \to \mathbb{S}^{\{m\}}$ such that

$$\mathcal{L}_m \mathcal{S} = \mathcal{F} \circ \mathcal{S} \circ \mathcal{N}(m) \tag{4.1.4}$$

for all $m \in M$ and $\mathcal{S} \in \mathfrak{S}$ respectively, using the projection operator,

$$\mathcal{L}_m := \mathcal{F} \circ \mathrm{proj}_{\mathcal{N}(m)} . \tag{4.1.5}$$

The scalar evolution operators incorporate the neighbourhood mapping as well as the update rule and exist independently of an alignment or indexing of the cells.

We can also use the notation

$$\mathcal{E}\mathcal{S} = \prod_{m \in M} \mathcal{L}_m \mathcal{S} = \prod_{m \in M} \mathcal{F} \circ \mathrm{proj}_{\mathcal{N}(m)} \mathcal{S}. \tag{4.1.6}$$

which indicates that $\mathcal{E}$ can be decomposed into its scalar evolution operators and interpreted as their Cartesian product

$$\mathcal{E} = \prod_{m \in M} \mathcal{L}_m. \tag{4.1.7}$$

Under which conditions can we construct $\mathcal{N}$ and $\mathcal{F}$ from a given collection of scalar evolution operators respectively from a given global evolution operator?

Definition 4.1.3 (Local Characterisation)**.** Let $\mathcal{L} : \mathbb{S}^M \to \mathbb{S}$. $\mathcal{L}$ is called *locally characterised* if there exists a real subset $N \subset M$ and

$$n \in N \iff \exists\, \mathcal{S}_1, \mathcal{S}_2 \in \mathbb{S}^M : \Big(\big(\mathcal{S}_1(l) \neq \mathcal{S}_2(l) \iff l = n \big) \implies \mathcal{L}\mathcal{S}_1 \neq \mathcal{L}\mathcal{S}_2, \Big) \tag{4.1.8}$$

or in other words, N is the set of elements n of M for which there exist $\mathcal{S}_1, \mathcal{S}_2 \in \mathfrak{S}$ with

$$\mathcal{S}_1(l) \neq \mathcal{S}_2(l) \iff l = n, \tag{4.1.9}$$

$$\mathcal{L}\mathcal{S}_1 \neq \mathcal{L}\mathcal{S}_2, \tag{4.1.10}$$

or, N is the set of cells that influence the state of the cell in the next iterative step.

We obviously want to use a local characterisation of $\mathcal{L}_m$ to define the neighbourhood mapping $\mathcal{N}(m) := N_m$ where N_m is the set from the previous definition.

For aligned cells it is necessary that the neighbourhoods can be constructed from the first topological feature. Hence in this case the sets N_m must be uniformly structured (Definition 3.1.6).

Furthermore there must exist a mapping $\mathcal{F} : \mathfrak{T}(\mathbb{S}) \to \mathbb{S}$ such that

$$\mathcal{L}_m = \mathcal{F} \circ \mathrm{proj}_{\mathcal{N}(m)} \tag{4.1.11}$$

independently of m. Of course $\mathcal{F}$ then is compatible with $\mathcal{N}$ and self-contained (Definition 2.1.26).

Proposition 4.1.4 (Scalar Evolution Operators)**.** *A collection of scalar evolution operators $\mathcal{L}_m$ can be identified with a neighbourhood mapping $\mathcal{N}$ and an update rule $\mathcal{F}$ under the condition that*

(i) the $\mathcal{L}_m$ are locally characterised,

(ii) if M is aligned, the neighbourhood mapping respectively the sets $\mathcal{N}(m)$ – which arise from the local characterisation – are uniformly structured,

(iii) there exists a mapping $\mathcal{F} : \mathfrak{T}(\mathbb{S}) \to \mathbb{S}$ such that $\mathcal{L}_m = \mathcal{F} \circ \mathrm{proj}_{\mathcal{N}(m)}$ for all $m \in M$.

If $\mathcal{L}_m$ is locally characterised with the set $\mathcal{N}(m)$, then there obviously exists a mapping $\mathcal{F}_m : \mathbb{S}^{\mathcal{N}(m)} \to \mathbb{S}$ such that for all $\mathcal{S} \in \mathbb{S}^M$, $\mathcal{L}_m\mathcal{S} = \mathcal{F}_m \circ \mathrm{proj}_{\mathcal{N}(m)}\mathcal{S}$. Requirement (iii) of the proposition states that this mapping is equal for all $m \in M$.

4.1.2 Alternative Definition of Cellular Automata

Definition 4.1.5 (Iteration of Global States, compare Definition 2.2.12)**.** For a given initial state $\mathcal{S}_0 \in \mathfrak{S}$, the iteration of global states is defined by $\mathcal{S}_{t+1} := \mathcal{E}(\mathcal{S}_t) = \prod_{m\in M} \mathcal{L}_m\mathcal{S}_t$ where $t \in \mathbb{N}$ or sometimes $t \in T$ for a connected subset $T \subset \mathbb{N}$. The iteration of global states yields a mapping $T \to \mathfrak{S} : t \mapsto \mathcal{S}_t$ where $\mathfrak{S} \subseteq \mathbb{S}^M$. We also use the notation $\mathcal{S}(t)$ and $\mathcal{S}(t, m)$, which actually renders $\mathcal{S}$ a function $T \to \mathfrak{S}$ respectively $T \times M \to \mathbb{S}$.

Definition 4.1.6 (Alternative Definition of Cellular Automaton)**.** A *cellular automaton* is completely defined by the 3-tuple $\big(M, \mathbb{S}, (\mathcal{L}_m)_{m\in M}\big)$, which contains

(i) a set of cells M with an optional alignment,

(ii) a set of possible states $\mathbb{S}$ and

(iii) a collection of scalar evolution operators whose Cartesian product defines a global evolution operator $\mathcal{E} := \prod_{m\in M} \mathcal{L}_m : \mathfrak{S} \to \mathfrak{S}$ and which satisfy the conditions of Proposition 4.1.4.

Given an initial global state $\mathcal{S}_0 \in \mathfrak{S}$, an iteration of global states is defined. The resulting series of state mappings describes the evolution of the cellular automaton.

4.2 Evolution Systems

This section introduces a definition of *evolution systems* that will prove to be compatible with what is known as *evolution equations* or *abstract Cauchy problems*.

First some properties of operators and collections of operators have to be discussed. Throughout this section we use a similar notation as for cellular automata but do not require any of the typical properties for cellular automata, like countability of the set M for example.

Definition 4.2.1 (Features of Operators [29, p. 21])**.** Let $\mathfrak{S}$ be a Banach space. For $\mathcal{E} : \mathfrak{S} \to \mathfrak{S}$ the *operator norm* is defined as $\|\mathcal{E}\| := \sup_{\|\mathcal{S}\|\leq 1} \|\mathcal{E}\mathcal{S}\|$. A $L(\mathfrak{S}, \mathfrak{S})$ valued function $T \to L(\mathfrak{S}, \mathfrak{S}) : t \mapsto \mathcal{E}(t)$ is called *strongly continuous* at t_0 if $\lim_{t\to t_0} \|\mathcal{E}(t) - \mathcal{E}(t_0)\| = 0$.

Definition 4.2.2 (Contraction Semigroup [29, pp. 23-24])**.** Let $\mathfrak{S}$ be a Banach space and $T = \mathbb{R}_+$, a one-parameter collection $(\mathcal{E}_t)_{t\in T}$ of bounded linear operators in $L(\mathfrak{S}, \mathfrak{S})$ is called a *contraction semigroup* if

(i) $\mathcal{E}_{t+s} = \mathcal{E}_t\mathcal{E}_s$ for all $s, t \in T$ (*semigroup property*),

(ii) $\lim_{t\to 0} \|\mathcal{E}_t\mathcal{S} - \mathcal{S}\| = 0$ for all $\mathcal{S} \in \mathfrak{S}$ (*strong continuity*) and

(iii) $\|\mathcal{E}_t\| \leq 1$ for all $t \in T$ (*contraction*).

Definition 4.2.3 (Infinitesimal Generator [29, p. 24])**.** Define

$$D := \Big\{\mathcal{S} \in \mathfrak{S} : \lim_{h\to 0+} \frac{(\mathcal{E}_h - I)\mathcal{S}}{h} \text{ exists in } \mathfrak{S}\Big\}. \tag{4.2.1}$$

The operator A defined on $\operatorname{dom} A = D$ by

$$A\mathcal{S} := \lim_{h\to 0+} \frac{(\mathcal{E}_h - I)\mathcal{S}}{h} \tag{4.2.2}$$

is called *infinitesimal generator* of $(\mathcal{E}_t)_{t\in T}$.

The theory of strongly continuous semigroups provides a lot of interesting results for generators and their corresponding semigroups. For example, if $(\mathcal{E}_t)_{t\in T}$ is a uniformly continuous semigroup there exists an infinitesimal generator $A \in L(\mathfrak{S}, \mathfrak{S})$ and

$$\mathcal{E}_t = \mathrm{e}^{tA} = \sum_{n=0}^{\infty} \frac{t^n}{n!} A^n. \tag{4.2.3}$$

Conversely the Theorem of Hille-Yosida guarantees the existence of a strongly continuous semigroup for a given infinitesimal generator.

Theorem 4.2.4 (Hille-Yosida [29, pp. 29-34])**.** *Let $A \in L(\mathfrak{S}, \mathfrak{S})$. A is the infinitesimal generator of some contraction semigroup if and only if A is closed*[1] *and its domain* $\operatorname{dom} A$ *is dense in* $\mathfrak{S}$.

[1] If for every sequence $\operatorname{dom} A \supset x_n \to x \in \mathfrak{S}$ with $Ax_n \to y$, it follows that $x \in \operatorname{dom} A$ and $Ax = y$, then A is called *closed*.

4.2.1 Definition of Evolution Systems

Evolution systems shall basically consist of a collection of operators with $\mathcal{E}_t \in L(\mathfrak{S},\mathfrak{S})$ and may feature finite ($|M| < \infty$) or infinite ($|M| = \infty$) dimensional vector spaces $\mathfrak{S} \subseteq \mathbb{S}^M$. The set M itself is equipped with a vectorspace structure (topology I) and the parameter set T may be discrete or continuous. The following requirements are used in Definition 4.2.6 to define evolution systems.

(M) Let M be a domain[2] in a discrete or continuous topological vector space.

($\mathbb{S}$) Assume a set of possible states $\mathbb{S}$ such that the current (t) state of the system can be described by a *state function*[3] $\mathcal{S}(t,\,\cdot\,) \in \mathfrak{S} \subseteq \mathbb{S}^M$ such that $\mathfrak{S}$ is a Banach space.

($\mathcal{E}$) Let $\mathcal{E}_{dt} \in L(\mathfrak{S},\mathfrak{S})$ be bounded linear *evolution operators* that evolve the state of the system by $dt \in T - T$ time units where for continuous time, $T = \mathbb{R}_+$, and for discrete iterations, $T = \mathbb{N}$.

$(\mathcal{E}_{dt})_{dt\in T-T}$ shall be a contraction semigroup (Definition 4.2.2).

(i) The evolution operators satisfy the so-called *semigroup property* $\mathcal{E}_{ds}\mathcal{E}_{dt} = \mathcal{E}_{ds+dt}$.

(ii) If $T \subseteq \mathbb{R}$ then $\mathcal{E}_{dt}$ is *strongly continuous* (at $dt = 0$) and $\mathcal{E}_0 = I$.

(iii) $\mathcal{E}_{dt}$ is a *contraction* ($\|\mathcal{E}_{dt}\| \leq 1$) for all $dt \in T - T$.

As in Section 4.1 we need a scalar decomposition of the evolution operators. Let the collection $(\mathcal{L}_{dt,m})_{(dt,m)\in(T-T)\times M}$ be defined by $\mathcal{L}_{dt,m} := \mathrm{proj}_{\{m\}} \circ \mathcal{E}_{dt}$ as in Definition 4.1.2.

(iv) The evolution operators can be represented as the Cartesian product of their *scalar decomposition* $\mathcal{L}_{dt,m} : \mathbb{S}^M \supseteq \mathfrak{S} \to \mathbb{S}^{\{m\}}$,

$$\mathcal{E}_{dt} = \prod_{m\in M} \mathcal{L}_{dt,m}, \tag{4.2.4}$$

which allows the notation $(\mathcal{E}_{dt}\mathcal{S})(m) = (\mathcal{L}_{dt,m}\mathcal{S})$.

The second topological feature of cellular automata implies that the functions $\mathcal{L}_{dt,m}$ only depend on a local configuration of $\mathcal{S} \in \mathbb{S}^M$. In addition to Definition 4.1.3 we define for differentiable mappings:

Definition 4.2.5 (Local Characterisation)**.** Let $\mathcal{L} : \mathbb{S}^M \to \mathbb{S}$ be differentiable. $\mathcal{L}$ is called *locally characterised* if there exists a real subset $N \subset M$ and

$$n \notin N \Longrightarrow \frac{\partial \mathcal{L}}{\partial s_n}(s_M) = 0, \qquad s_M = (s_m)_{m\in M} \in \mathbb{S}^M \tag{4.2.5}$$

[2] Compare the definition of *connected* for index sets $I \subseteq \mathbb{Z}^d$ in Definition 2.1.2.

[3] The notation *function* seems more appropriate than *mapping* for continuous domains.

N is again the set of cells that influence the state of the cell in the next iterative step. Both formulations state that the outcome of applying $\mathcal{L}$ on a state function in $\mathbb{S}^M$ is independent of the values of the state functions at $n \notin N$.

($\mathcal{L}$) The scalar operators $\mathcal{L}_{dt,m}$ additionally satisfy the following requirements.

(i) The $\mathcal{L}_{dt,m}$ are locally characterised with local sets $\mathcal{N}_{dt}(m)$.

Accordingly they can be decomposed as

$$\mathcal{L}_{dt,m} = \mathcal{F}_{dt,m} \circ \operatorname{proj}_{\mathcal{N}_{dt}(m)} \tag{4.2.6}$$

and the *neighbourhood functions* $\mathcal{N}_{dt} : M \to \mathfrak{P}(M)$ determine the "coordinates" that have influence on the result of $\mathcal{L}_{dt,m}$.

(ii) Since M is a vector space, we require uniform neighbourhoods respectively that the neighbourhood functions are linear mappings $\mathcal{N}_{dt} : M \to \mathfrak{P}(M) : m \mapsto m + N_{dt}$ where $N_{dt} \subset M - M$ is a bounded domain for all $dt < \infty$.

Actually $\mathcal{N}_{dt}(m) := (m + N_{dt}) \cap M$, which delivers degraded neighbourhoods for certain points $m \in M$.

(iii) The *(update) functions* $\mathcal{F}_{dt,m} : \mathbb{S}^{m+N_{dt}} \to \mathbb{S}^{\{m\}}$ shall be independent of m. Accordingly $\mathcal{F}_{dt} : \mathbb{S}^{N_{dt}} \to \mathbb{S}$.

$\mathcal{F}_{dt,m}$ are obviously linear maps.

Definition 4.2.6 (Evolution System)**.** $M, \mathbb{S}, T$ and $\mathcal{E}_{dt}$ as defined above form an *evolution system.* If additionally ($\mathcal{L}$) is satisfied, the system is called a *locally characterised evolution system.* Based on the set M a differentiation of *continuous-space* and *discrete-space* evolution systems can be made. The same applies to *continuous-time* and *discrete-time* evolution systems.

Evolution systems are semigroups with specially structured spaces $\mathfrak{S}$. Locally characterised evolution systems additionally satisfy the properties defined in ($\mathcal{L}$).

The definition of a global evolution operator contradicts somehow to the bottom-up concept of cellular automata. This shows especially for locally characterised evolution systems in the fact that, if a global evolution operator is defined, the neighbourhood mapping must be regarded as a result of this definition, which for cellular automata would not be a comprehensive approach. Actually the neighbourhood mappings $\mathcal{N}_{dt}$ are a consequence of the functions $\mathcal{F}_{dt}$ which map functions $M \supset \mathcal{N}_{dt}(\,\cdot\,) \to \mathbb{S}$ onto $\mathbb{S}$. Compare the concept of local characterisations with the alternative definition of unaligned cellular automata in Section 4.1 or with the local kernel of stochastic cellular automata in Section 5.2.

The differentiation between evolution systems and locally characterised evolution systems is motivated by the second topological feature of cellular automata. The

evolution (iteration) of the state of an individual location (cell) only depends on a bounded subset of neighbouring elements in M, the neighbourhood. For not locally characterised evolution systems on the other hand the change of the state of a location in M may depend on the global configuration of the system.

In order to discuss regular alignments or regular neighbourhoods it is necessary that M features a vector space structure. The topological structure of M represents the first topological feature.

The semigroup property is a logical constraint. If it is not satisfied, the system would not be well-defined. The strong continuity at $dt = 0$ states that during a small time period only small changes happen and that no changes happen if no time passes.

In the following sections we will see that this generic approach to evolution systems with arbitrary Banach spaces $\mathfrak{S}$ can be problematic in some situations. Section 4.2.2 provides a special type of *integral evolution systems* which will prove to be much easier to handle.

It is also important to note that evolution systems with linear evolution operators are actually not very useful in the context of cellular automata. We usually want cellular automata to behave nonlinearly on the microscopic scale at least. On the other hand linear evolution operators allow a broader range of tools for analysis (semigroup theory, infinitesimal generators etc.) than nonlinear evolution operators. A common approach for this kind of problem is to regard nonlinear systems as distortions of linear systems respectively to approximate nonlinear systems with linear evolution systems. In Section 4.3 *quasilinear evolution systems* are defined in a way such that basic results from semigroup theory can be applied nevertheless.

	discrete time	discrete space	local characterisation	linear evolution operators
cellular automata	•	•	•	
LES			•	•
ES				•

Table 4.1: Comparison of cellular automata, locally characterised evolution systems (LES) and evolution systems (ES).

4.2.1.1 Some Immediate Results and Conclusions

Proposition 4.2.7. *For any evolution system it is true that* $\lim_{dt\to 0} \mathcal{L}_{dt,m} = \mathrm{proj}_{\{m\}}$.

Proof. From the definition of the scalar evolution operator (Definition 4.1.2) it follows that

$$\lim_{dt\to 0} \mathcal{L}_{dt,m} = \lim_{dt\to 0} \mathrm{proj}_{\{m\}} \circ \mathcal{E}_{dt} = \mathrm{proj}_{\{m\}} \circ I_{\mathbb{S}^M}. \tag{4.2.7}$$

□

Accordingly

$$\lim_{dt\to 0} \mathcal{L}_{dt,m}\mathcal{S} = \mathcal{S}(m) \qquad \mathcal{S} \in \mathfrak{S} \tag{4.2.8}$$

and with the notation from Equation 4.2.5 we can write

$$\lim_{dt\to 0} \frac{\partial \mathcal{L}_{dt,m}}{\partial s_n}(s_M) = 0 \qquad \forall n \neq m. \tag{4.2.9}$$

Corollary 4.2.8. *If the scalar evolution operators $\mathcal{L}_{dt,m}$ are locally characterised with neighbourhoods $\mathcal{N}_{dt}(m)$ and a common update function $\mathcal{F}_{dt}$ such that $\mathcal{L}_{dt,m} = \mathcal{F}_{dt} \circ \mathrm{proj}_{\mathcal{N}_{dt}(m)}$, then*

$$\mathcal{F}_{dt} \longrightarrow I_{\mathbb{S}} \qquad \text{and} \qquad \mathcal{N}_{dt}(m) \longrightarrow \{m\} \tag{4.2.10}$$

for $dt \longrightarrow 0$.

Proof. Convergence of a set to a point can for example be formalised by the point wise limit of the indicator function. $\mathcal{F}_{dt} \longrightarrow I_{\mathbb{S}}$ actually means that the dimension of the domain of $\mathcal{F}_{dt}$, which is $\mathbb{S}^{\mathcal{N}_{dt}(m)}$, converges towards 1. This is of course implied by the convergence of the neighbourhoods.

Because of Proposition 4.2.7 we know that $\lim_{dt\to 0} \mathcal{F}_{dt} \circ \mathrm{proj}_{\mathcal{N}_{dt}(m)} = \mathrm{proj}_{\{m\}}$. And because the neighbourhood sets $\mathcal{N}_{dt}(m)$ are defined to be the minimum set of elements of M that influence the outcome of $\mathcal{L}_{dt,m}$, it is necessary that $\lim_{dt\to 0} \mathcal{N}_{dt}(m) = \{m\}$. □

Proposition 4.2.9. *Let for a locally characterised evolution system $\mathcal{N}_{dt}(m) = m + N_{dt}$, then $N_{ds+dt} = N_{ds} + N_{dt}$.*

Proof. The semigroup property yields

$$\mathcal{E}_{ds+dt} = \mathcal{E}_{ds} \circ \mathcal{E}_{dt} \tag{4.2.11}$$

$$\mathcal{L}_{ds+dt,m} = \mathcal{L}_{ds,m} \circ \prod_{n\in m+N_{ds}} \mathcal{L}_{dt,n} \tag{4.2.12}$$

$$\mathcal{F}_{ds+dt} \circ \mathrm{proj}_{m+N_{ds+dt}} = \mathcal{F}_{ds} \circ \mathrm{proj}_{m+N_{ds}} \circ \prod_{n\in m+N_{ds}} \mathcal{F}_{dt} \circ \mathrm{proj}_{n+N_{dt}} \tag{4.2.13}$$

$$= \mathcal{F}_{ds} \circ \prod_{n\in m+N_{ds}} \mathcal{F}_{dt} \circ \mathrm{proj}_{n+N_{dt}}\,. \tag{4.2.14}$$

Again from the minimality of the neighbourhoods we can conclude that $N_{ds+dt} = N_{ds} + N_{dt}$. □

Corollary 4.2.10. *If the neighbourhoods are star-shaped, then $N_{dt} \subset N_{ds}$ for $dt < ds$. Without proof.*

Example 4.2.11. A prototype for the neighbourhoods is for example $N_{dt} := dt \cdot N_1$ where N_1 is a star-shaped or convex set containing 0. In this situation also the limit $\lim_{dt\to 0} \mathcal{N}_{dt}(m) = \{m\}$, which corresponds to $\lim_{dt\to 0} N_{dt} = \{0\}$, can be motivated more easily.

The properties $N_{ds+dt} = N_{ds} + N_{dt}$ and $N_0 = \{0\}$ render the collection of all neighbourhoods a commutative semigroup or monoid.

4.2.1.2 Some Considerations on Generators

Since $I_{\mathbb{S}^M} = \prod_{m\in M} \mathrm{proj}_{\{m\}}$, the infinitesimal generator of the semigroup $\mathcal{E}_{dt}$ can be decomposed as

$$A = \lim_{dt\to 0} \frac{\mathcal{E}_{dt} - I_{\mathbb{S}^M}}{dt} = \lim_{dt\to 0} \frac{\prod_{m\in M} \mathcal{L}_{dt,m} - \prod_{m\in M} \mathrm{proj}_{\{m\}}}{dt} \tag{4.2.15}$$

$$= \prod_{m\in M} \underbrace{\lim_{dt\to 0} \frac{\mathcal{L}_{dt,m} - \mathrm{proj}_{\{m\}}}{dt}}_{=:A_m} \tag{4.2.16}$$

where $A_m : D \to \mathbb{S}^{\{m\}}$ and D is the domain of A.

For locally characterised evolution systems we can write

$$\mathcal{E}_{dt} = \prod_{m\in M} \mathcal{L}_{dt,m} \qquad = \prod_{m\in M} \mathcal{F}_{dt} \circ \mathrm{proj}_{m+N_{dt}} \tag{4.2.17}$$

$$\mathcal{E}_0 = I_{\mathbb{S}^M} = \prod_{m\in M} \mathrm{proj}_{\{m\}} \qquad = \prod_{m\in M} I_{\mathbb{S}^{\{m\}}} \circ \mathrm{proj}_{\{m\}} \tag{4.2.18}$$

And the infinitesimal generator of the semigroup $\mathcal{E}_{dt}$ can at least formally be written as

$$\lim_{dt\to 0} \frac{\mathcal{E}_{dt} - I_{\mathbb{S}^M}}{dt} \tag{4.2.19}$$

$$= \lim_{dt\to 0} \frac{1}{dt} \left(\prod_{m\in M} \mathcal{F}_{dt} \circ \mathrm{proj}_{m+N_{dt}} - \prod_{m\in M} I_{\mathbb{S}^{\{m\}}} \circ \mathrm{proj}_{\{m\}} \right) \tag{4.2.20}$$

$$= \lim_{dt\to 0} \frac{1}{dt} \prod_{m\in M} \left(\mathcal{F}_{dt} \circ \mathrm{proj}_{m+N_{dt}} - \mathrm{proj}_{\{m\}} \right) \tag{4.2.21}$$

$$= \prod_{m\in M} \underbrace{\lim_{dt\to 0} \frac{\left(\mathcal{F}_{dt} - \mathrm{proj}_{\{0\}}\right) \circ \mathrm{proj}_{m+N_{dt}}}{dt}}_{=A_m}. \tag{4.2.22}$$

4.2.1.3 Outlook

For a not locally characterised evolution system we concluded in Proposition 4.2.7 that $\lim_{dt\to 0} \mathcal{L}_{dt,m} = \mathrm{proj}_{\{m\}} \circ I_{\mathbb{S}^M}$. Without changing the result we can prepend a projection operator such that

$$\lim_{dt\to 0} \mathcal{L}_{dt,m} = \mathrm{proj}_{\{m\}} \circ I_{\mathbb{S}^{\{m\}}} \circ \mathrm{proj}_{\{m\}}, \tag{4.2.23}$$

which, using the concept of neighbourhoods, can be expressed as $\mathcal{N}_0(m) = \{m\}$ or for smooth mappings as

$$\lim_{dt\to 0} \frac{\partial \mathcal{L}_{dt,m}}{\partial s_n}(s_M) = 0 \qquad \forall n \neq m. \tag{4.2.24}$$

Under the assumption that $dt \mapsto \mathcal{L}_{dt,m}$ is somehow a smooth mapping the question arises, whether for small $dt > 0$ there exists a strict bounded "neighbourhood" subset $N_{dt} \subset M$ with

$$\frac{\partial \mathcal{L}_{dt,m}}{\partial s_n}(s_M) = 0 \qquad \forall n \notin N_{dt} \tag{4.2.25}$$

or at least

$$\frac{\partial \mathcal{L}_{dt,m}}{\partial s_n}(s_M) < \varepsilon \qquad \forall n \notin N_{dt} \tag{4.2.26}$$

for arbitrary small $\varepsilon > 0$.

This would mean that at least for small times, an evolution system can be approximated by a locally characterised evolution system and hence probably also by a cellular automaton.

4.2.2 Integral Form of Evolution Systems

This section provides a discussion of a measure theoretic or integral approach for evolution systems. The characteristics of evolution systems respectively locally characterised evolution systems can be reformulated without hurting the validity of Definition 4.2.6. In contrast to generic evolution systems this special type allows easier analysis.

(*M*) Let $(M, \mathfrak{M}, \mu)$ be a finite ($\mu(M) < \infty$) measure space where $\mathfrak{M}$ is the Borel σ-algebra[4].

For simplicity let $\mathbb{S} = \mathbb{R}$ throughout this section. It should however also be possible to use arbitrary Banach spaces.

($\mathbb{S}$) $\mathcal{S} : M \to \mathbb{S}$ shall (always) be an integrable function such that the integral

$$\int_N \mathcal{S}(n)\, d\mu(n) = \int \mathbb{I}_N(n)\, \mathcal{S}(n)\, d\mu(n), \tag{4.2.27}$$

[4]The (unique) smallest σ-algebra containing the open sets is called *Borel σ-algebra*. [6, p. 18]

as a limit of the integrals of step-functions [6, p. 129], exists for all $N \in \mathfrak{M}$ as integration domains. The set of integrable functions on M is denoted by $\mathfrak{S} := L^1(M)$ and is a Banach space (Lebesgue space respectively Lebesgue-Bochner space if $\mathbb{S}$ shall be an arbitrary Banach space). If M is a discrete space, all integral expressions are sums for which convergence has to be asserted instead of integrability.

($\mathcal{E}$) $\mathcal{E}_{dt} : \mathfrak{S} \to \mathfrak{S}$ shall define the evolution of the system by

$$\mathcal{E}_{dt}\mathcal{S}(m) = \mathcal{L}_{dt,m}\mathcal{S} := \int \mathcal{K}\big(dt, \mathcal{S}(n), n, m\big)\, d\mu(n) \tag{4.2.28}$$

where $\mathcal{K}$ is a suitable mapping $T \times \mathbb{S} \times M \times M \to \mathbb{S}$.

In order to be able to draw relevant conclusions for this type of evolution systems, the mapping $\mathcal{K}$ must be assumed to admit a simpler (generic or linear) form[5]. Assume that for a given location the function $\mathcal{K}$ determines weights for the influence of the states of the "neighbouring" locations. That is, $\mathcal{K}$ is a function $T \times M \times M \to \mathbb{K}$ where $\mathbb{K}$ is the scalar space of $\mathbb{S}$ and

$$\mathcal{E}_{dt}\mathcal{S}(m) = \mathcal{L}_{dt,m}\mathcal{S} = \int \mathcal{K}(dt, n, m)\, \mathcal{S}(n)\, d\mu(n). \tag{4.2.29}$$

($\mathcal{L}$) There exists a collection of mappings $\mathcal{N}_{dt} : M \to \mathfrak{M}$ such that $\mathcal{K} : T \times M \times M \to \mathbb{K}$ is an integrable function with support $\mathcal{N}_{dt}(m)$ in its second argument for all $dt \in T$. Accordingly,

$$\mathcal{E}_{dt}\mathcal{S}(m) = \mathcal{L}_{dt,m}\mathcal{S} = \int \mathcal{K}(dt, n, m)\, \mathcal{S}(n)\, d\mu(n) \tag{4.2.30}$$

$$= \int_{\mathcal{N}_{dt}(m)} \mathcal{K}(dt, n, m)\, \mathcal{S}(n)\, d\mu(n). \tag{4.2.31}$$

If the system shall not be locally characterised, define $\mathcal{N}_{dt}(m) := M$ for all $m \in M$ or neglect the conditions in ($\mathcal{L}$).

4.2.2.1 Some Measure Theoretic Considerations

This section features a measure theoretic interpretation of evolution systems and shows under which conditions $\mathcal{E}_{dt}$ is actually a bounded linear operator.

For convenience also the notations $\mathcal{K}_{dt,m}(n) := \mathcal{K}(dt, n, m)$ and $\mathcal{K}_{dt}(n, m) := \mathcal{K}(dt, n, m)$ will be used.

Let $\mathbb{S} = [0, \infty]$. Since $\mathcal{S} : M \to \mathbb{S}$ is a (non-negative) measurable function, $\mathcal{S}$ defines a measure

$$\mu_{\mathcal{S}}(N) := \int_N \mathcal{S}(n)\, d\mu(n) \qquad N \in \mathfrak{M} \tag{4.2.32}$$

[5] An operator defined by a kernel as in Equation 4.2.28 is also called *nonlinear integral operator*.

on $(M, \mathfrak{M})$ [6, p. 127].

Conversely the Theorem of Radon-Nikodým (Theorem 5.3.2) states that for two measures $\mu_{\mathcal{S}}$ and μ, where μ is σ-finite and $\mu_{\mathcal{S}}$ is absolutely continuous[6] with respect to μ, there exists a (quasi-)[7] integrable function $\mathcal{S} : M \to \mathbb{S}$ – called *density with respect to* μ – such that Equation 4.2.32 holds for all $N \in \mathfrak{M}$.

If $\mathcal{K} : T \times M \times M \to [0, \infty]$ is a (non-negative) measurable function in its second argument (n) with support $\mathcal{N}_{dt}(m)$, then we can define

$$\begin{aligned}\nu_{\mathcal{S},dt,m}(N) := \int_{N \cap \mathcal{N}_{dt}(m)} \mathcal{K}_{dt,m}(n)\, \mathcal{S}(n)\, d\mu(n) &= \int_N \mathcal{K}(dt, n, m)\, \mathcal{S}(n)\, d\mu(n) \\ = \int_{N \cap \mathcal{N}_{dt}(m)} \mathcal{K}_{dt,m}(n)\, d\mu_{\mathcal{S}}(n) &= \int_N \mathcal{K}(dt, n, m)\, d\mu_{\mathcal{S}}(n) \quad (4.2.33)\end{aligned}$$

which is a measure on $(M, \mathfrak{M})$ for every $m \in M$ with density $\mathcal{K}(dt, \cdot, m)\mathcal{S}(\cdot)$ with respect to μ and density $\mathcal{K}(dt, \cdot, m)$ with respect to $\mu_{\mathcal{S}}$.

According to Fubinis Theorem the function $\tilde{\mathcal{S}} : M \to \mathbb{R} : m \mapsto \nu_{\mathcal{S},dt,m}(M) = \nu_{\mathcal{S},dt,m}(\mathcal{N}_{dt}(m))$ is non-negative and measurable if $\mathcal{K}_{dt} : M \times M \to [0, \infty]$ is (product-) measurable [6, p. 175].

Lemma 4.2.12 (Integral Evolution operators)**.** *Assume a measurable space* $(M, \mathfrak{M})$ *with finite measure* μ *and the Borel* σ*-algebra. Set* $\mathbb{S} = \mathbb{R}_+$ *and let* $\mathfrak{S}$ *be the set of integrable respectively measurable (since* $\mathbb{S} = \mathbb{R}_+$*) functions. Operators* $\mathcal{E}_{dt}$ *on* $\mathfrak{S}$ *defined through a product-measurable function* $\mathcal{K} : T \times M \times M \to [0, \infty]$ *(also called* kernel*) as in Equation 4.2.29 are linear integral operators* $\mathfrak{S} \to \mathfrak{S}$*.*

Proof. By definition. □

Proposition 4.2.13 (Integral Evolution operators)**.** *With the same preconditions as in Lemma 4.2.12,* $\mathcal{E}_{dt}$ *are* bounded *linear operators on* $\mathfrak{S}$ *if the kernels* $\mathcal{K}_{dt}$ *are bounded on* $M \times M$*.*

Proof. The Hölder inequality yields[8]

$$\mathcal{E}_{dt}\mathcal{S}(m) = \int \mathcal{S}(n)\mathcal{K}_{dt,m}(n)\, d\mu(n) = \|\mathcal{S}\mathcal{K}_{dt,m}\|_{L^1} \leq \quad (4.2.34)$$

$$\leq \|\mathcal{K}_m\|_{L^\infty}\, \|\mathcal{S}\|_{L^1} = \sup_{n \in M} \mathcal{K}_m(n)\, \|\mathcal{S}\|_{L^1}. \quad (4.2.35)$$

Consequently

$$\|\mathcal{E}_{dt}\mathcal{S}\|_{L^1} = \int_M \mathcal{E}_{dt}\mathcal{S}(m)\, d\mu(m) \leq \int_M \sup_{n \in M} \mathcal{K}_{dt}(m, n)\, d\mu(m)\, \|\mathcal{S}\|_{L^1} \quad (4.2.36)$$

$$\leq \mu(M) \sup_{n,m \in M} \mathcal{K}_{dt}(m, n)\, \|\mathcal{S}\|_{L^1} \quad (4.2.37)$$

$$= \mu(M)\, \|\mathcal{K}_{dt}(m, n)\|_{L^\infty(M \times M)}\, \|\mathcal{S}\|_{L^1(M)}. \quad (4.2.38)$$

[6] $\nu \ll \mu$ if and only if $\mu(N) = 0 \Rightarrow \nu(N) = 0$ for all $N \in \mathfrak{M}$.

[7] This prefix is not necessary since $\mathbb{S} = [0, \infty]$.

[8] One may replace sup with esssup.

□

In [29, pp. 75-76, Theorem 4.2] Taira uses weaker requirements to ensure that $\|\mathcal{E}_{dt}S\|_{L^p} \leq C\|\mathcal{S}\|_{L^p}$, namely $\mathcal{K}_{dt} : M \times M \to \mathbb{R}$ is a product measurable function with

$$\sup_{m\in M} \int_M |\mathcal{K}_{dt}(m,n)|\, d\mu(n) \leq C, \tag{4.2.39}$$

$$\sup_{n\in M} \int_M |\mathcal{K}_{dt}(m,n)|\, d\mu(m) \leq C. \tag{4.2.40}$$

Since M has a vector space structure (V), we can require that $\mathcal{K}(dt,n,m) = \mathcal{K}(dt,n-m)$, which means that $\mathcal{K}_{dt}$ is independent of the location m respectively *uniform*.

Accordingly $\operatorname{supp}\mathcal{K}_{dt} = \mathcal{N}_{dt}(m) - m = N_{dt}$ and the evolution operator can be written as

$$\mathcal{E}_{dt}\mathcal{S}(m) = \mathcal{L}_{dt,m}\mathcal{S} = \int_{m+N_{dt}} \mathcal{S}(n)\,\mathcal{K}(dt,n-m)\, d\mu(n) \tag{4.2.41}$$

$$= \int_{N_{dt}} \mathcal{S}(m+v)\,\mathcal{K}(dt,v)\, d\mu(v) \tag{4.2.42}$$

$$= \int \mathcal{S}(m+v)\,\mathcal{K}(dt,v)\, d\mu(v). \tag{4.2.43}$$

It is rather obvious that this formulation using density kernels features a connection to Markov processes and satisfies a special Chapman-Kolmogorov-type of equation (see Chapter 5).

Corollary 4.2.14 (Boundedness of Integral Evolution Operators)**.** *Again with the preconditions from Lemma 4.2.12 and uniform kernel,* $\|\mathcal{E}_{dt}\| \leq \|\mathcal{K}_{dt}\|_{L^1(V)}$.

Proof. Since $\mathcal{K}_{dt}(v)$ does not depend on the location within M, we can set $C := \|\mathcal{K}_{dt}\|_{L^1(V)}$ in Tairas theorem. □

4.2.2.2 Approximation of Integral Evolution Systems with Locally Characterised Integral Evolution Systems

The Dirac delta distribution δ_0 is often defined in the weak sense by the limit of functions $\mathcal{K}_{dt}$ under the condition that

$$\lim_{dt\to 0} \int_V \mathcal{K}_{dt}(v)\mathcal{S}(v)\, dv = \mathcal{S}(0). \tag{4.2.44}$$

These functions can for example be probability density functions like the Gauss function but also functions with compact support. A typical condition for such a sequence of functions is that

$$\lim_{dt\to 0} \int_{B_\varepsilon(0)} \mathcal{K}_{dt}(v)\, dv = 1 \qquad \forall \varepsilon > 0. \tag{4.2.45}$$

As a consequence we can find a constant $C_{dt,\varepsilon}$ such that

$$\int_V \mathcal{K}_{dt}(v)\ dv - \int_{B_\varepsilon(0)} \mathcal{K}_{dt}(v)\ dv \leq C_{dt,\varepsilon}, \tag{4.2.46}$$

or if we define $\tilde{\mathcal{K}}(v) := \mathbb{I}_{B_\varepsilon(0)}(v)\mathcal{K}(v)$,

$$\|\mathcal{K}_{dt} - \tilde{\mathcal{K}}_{dt}\|_{L^1(V)} \leq C_{dt,\varepsilon}. \tag{4.2.47}$$

There may however exist better approximations $\tilde{\mathcal{K}}_{dt}$ with (bounded) support in $B_\varepsilon(0)$ of $\mathcal{K}_{dt}$ than the version of $\mathcal{K}_{dt}$ which is cut off and continued by 0 outside of $B_\varepsilon(0)$ (compare Section 4.3.3.4).

Proposition 4.2.15 (Approximation of Integral Evolution Systems)**.** *For small $dt > 0$ an integral evolution operator $\mathcal{E}_{dt}$ can be approximated by a locally characterised integral evolution operator $\tilde{\mathcal{E}}_{dt}$.*

Proof. The previous considerations (respectively Equation 4.2.47) together with Corollary 4.2.14 deliver the desired inequality. □

This result makes it even more interesting whether it is possible to approximate evolution operators with locally characterised evolution operators even if there exists no integral representation (compare Section 4.2.1.3).

4.2.2.3 Further Conclusions for Integral Evolution Systems

The identity operator can be formulated as ($V := M - M$)

$$I_{\mathbb{S}^M}\mathcal{S}(m) = \int_V \mathcal{S}(m+v)\delta_0(v)\ dv. \tag{4.2.48}$$

And accordingly the infinitesimal generator satisfies

$$A\mathcal{S}(m) = \lim_{dt\to 0}\frac{1}{dt}\left(\int_V \mathcal{S}(m+v)\mathcal{K}_{dt}(v)\ dv - \int_V \mathcal{S}(m+v)\delta_0(v)\ dv\right) \tag{4.2.49}$$

$$= \lim_{dt\to 0}\frac{1}{dt}\int_V \mathcal{S}(m+v)\big(\mathcal{K}_{dt}(v) - \delta_0(v)\big)\ dv \tag{4.2.50}$$

$$= \int_V \mathcal{S}(m+v)\underbrace{\lim_{dt\to 0}\frac{\mathcal{K}_{dt}(v) - \delta_0(v)}{dt}}_{=:a_{dt}(v)}\ dv. \tag{4.2.51}$$

a_{dt} is the kernel of the infinitesimal generator.

Proposition 4.2.16. *The kernels of integral evolution systems satisfy $\mathcal{K}_{ds+dt} = \mathcal{K}_{dt} * \mathcal{K}_{ds}$. If the evolution system is locally characterised, then additionally $N_{ds+dt} = N_{ds} + N_{dt}$.*

Proof. By observing the semigroup property $\mathcal{E}_{ds+dt} = \mathcal{E}_{ds}\mathcal{E}_{dt}$ and using a standard integral transformation ($z := x + y$),

$$\mathcal{E}_{ds+dt}\mathcal{S}(m) = \int_{N_{ds+dt}} \mathcal{S}(m+z)\mathcal{K}_{ds+dt}(z)\, dz \tag{4.2.52}$$

$$\mathcal{E}_{ds}\mathcal{E}_{dt}\mathcal{S}(m) = \int_{N_{ds}}\int_{N_{dt}} \mathcal{S}(m+x+y)\mathcal{K}_{dt}(x)\mathcal{K}_{ds}(y)\, dx\, dy \tag{4.2.53}$$

$$= \int_{N_{dt}+N_{ds}} \mathcal{S}(m+z) \underbrace{\int_{N_{ds}} \mathcal{K}_{dt}(z-y)\mathcal{K}_{ds}(y)\, dy}_{=\mathcal{K}_{dt}*\mathcal{K}_{ds}(z)}\, dz, \tag{4.2.54}$$

we can conclude that $\mathcal{K}_{ds+dt} = \mathcal{K}_{dt} * \mathcal{K}_{ds}$ and $N_{ds+dt} = N_{ds} + N_{dt}$. □

4.2.3 Evolution Equations

This section presents a short discussion of *evolution equations* as defined for example in [1, 7, 13, 14, 18, 30].

Evolution equations can also be seen as an implicit description of (not necessarily locally characterised) continuous evolution systems. Or conversely, Definition 4.2.6 can be regarded as a "bottom-up" description of evolution equations.

Let $\mathcal{E}_{dt}$ be the evolution operator of an evolution system with $\mathcal{S}(t+dt, m) = (\mathcal{E}_{dt}\mathcal{S})(t, m)$. The time-derivative of the state function is defined as

$$\partial_t\mathcal{S}(t,m) := \lim_{dt\to 0} \frac{\mathcal{S}(t+dt,m) - \mathcal{S}(t,m)}{dt} = \lim_{dt\to 0} \Big(\frac{\mathcal{E}-I}{dt}\,\mathcal{S}\Big)(t,m). \tag{4.2.55}$$

If A is the infinitesimal generator of the semigroup $\mathcal{E}_{dt}$, we obtain a so-called *evolution equation*,

$$\partial_t\mathcal{S}(t,m) = (A\mathcal{S})(t,m). \tag{4.2.56}$$

Vice versa – under the conditions of the Hille-Yosida theorem (Theorem 4.2.4) for example –, a formal evolution operator can be derived from an evolution equation by

$$\mathcal{S}(t+dt,m) = \mathcal{S}(t,m) + \int_t^{t+dt} (A\mathcal{S})(\tau,m)\, d\tau \approx \mathcal{S}(t,m) + dt\cdot(A\mathcal{S})(t,m) \tag{4.2.57}$$

or from an exponential representation.

Accordingly under certain conditions an evolution system can be identified with an evolution equation.

4.2.3.1 Abstract Cauchy Problems

In literature [1, 7, 13, 14, 18, 30] equations as in Equation 4.2.56 are called *evolution equations*, *abstract parabolic equations* or *abstract Cauchy problems* and usually

discussed in the more general form

$$\begin{aligned} \partial_t \mathcal{S} &= A(t)\mathcal{S} + G(t) \qquad t \in [0,T] \\ \mathcal{S}(0) &= \mathcal{S}_0 \end{aligned} \tag{4.2.58}$$

where $\mathfrak{S}$ is a Banach or Hilbert space and $\mathcal{S}$ as well as G are functions $[0,T] \to \mathfrak{S}$, $A : [0,T] \to L(\mathfrak{S},\mathfrak{S})$ and $\mathcal{S}_0 \in \mathfrak{S}$.

The initial value problem in Equation 4.2.58 can be solved in different senses. Usually it is required that

$$\begin{aligned} \mathcal{S} &\in C\big([0,T],\mathfrak{S}\big) \\ \mathcal{S}(t) &\in \operatorname{dom} A(t)\ \forall t \in [0,T] \\ t \mapsto A(t)\mathcal{S}(t) &\in C\big([0,T],\mathfrak{S}\big) \end{aligned} \tag{4.2.59}$$

or that there exists an approximation $(\mathcal{S}_n)_{n\in\mathbb{N}}$ of $\mathcal{S}$ such that $\mathcal{S}_n$ solves the equation for approximations G_n of G [1] or if $\mathfrak{S}$ is a Hilbert space, *weak solutions* can be seeked etc.

If a "solution ... can be expressed in the form"

$$\mathcal{S}(t) = U(0,t)\mathcal{S}(0) + \int_0^t U(s,t)G(s)\,ds \tag{4.2.60}$$

then $U(s,t)$, $0 \le s \le t \le T$, is called "*evolution operator* or *fundamental solution*" [18, p. 98]. A typical precondition is that $A(t)$ is the infinitesimal generator of an analytic semigroup for all $t \in [0,T]$ [1, 18, 30].

In the context of cellular automata it is sufficient to discuss constant (time-independent) A and G because we required that the update rules do not change over time ($U(s,t) = U(t-s)$). This allows to avoid the integral term in Equation 4.2.60 (by including G in A) and to weaken the preconditions for the existence and uniqueness of a solution.

4.2.3.2 Parabolic Partial Differential Equations

If $M \cong V \subseteq \mathbb{R}^d$, $\mathbb{S} = \mathbb{R}^d$ and if A is a second order elliptic differential operator, the evolution equation is called *parabolic partial differential equation* [8, p. 350]. Of course we only observe linear parabolic equations with constant coefficients (in nondivergence form),

$$\frac{\partial \mathcal{S}}{\partial t}(t,v) = \sum_{i,j=1}^{d} a_{ij} \frac{\partial^2 \mathcal{S}}{\partial v_i \partial v_j}(t,v) - \sum_{i=1}^{d} b_i \frac{\partial \mathcal{S}}{\partial v_i}(t,v) - c\mathcal{S}(t,v) \tag{4.2.61}$$

where the *ellipticity condition*

$$\sum_{i,j=1}^{d} a_{ij} v_i v_j > \|v\| \tag{4.2.62}$$

is satisfied.

Classical solutions are characterised by additional requirements on the initial condition or state ($T = [0, \infty)$), a *Sobolev* state space $\mathfrak{S} \subseteq H_0^1(V)$ (see for example [8]) and

$$\mathcal{S}(\,\cdot\,,\,\cdot\,) \in C^0\big([0,\infty), L^2(V)\big) \cap C^1\big((0,\infty), L^2(V)\big). \tag{4.2.63}$$

Weak solutions of the partial differential equation can be found with $\mathfrak{S} \subseteq H_0^1(V)$ and

$$\mathcal{S}(\,\cdot\,,\,\cdot\,) \in L^2\big(0, T_{\text{end}}; H_0^1(V)\big) \tag{4.2.64}$$

$$\partial_t \mathcal{S}(\,\cdot\,,\,\cdot\,) \in L^2\big(0, T_{\text{end}}; H^{-1}(V)\big) \tag{4.2.65}$$

where $L^2\big(0, T_{\text{end}}; H_0^1(V)\big)$ imposes additional requirements on measurable functions $(0, T_{\text{end}}) \to H_0^1(V)$ [8].

4.2.3.3 Integral Equations

Let $(M, \mathfrak{M}, \mu)$ be a measure space and $\mathfrak{S} := L^1(M)$ the set of integrable functions. Let A be an integral operator such that the evolution equation has the form

$$\partial_t \mathcal{S}(t, m) = \int_M \mathcal{S}(t, n)\, f(n, m)\, d\mu(n). \tag{4.2.66}$$

The kernel f is necessarily a (non-negative) function that is measurable in its first argument.

Assume that f is the derivative of a function $\mathcal{K} : T \times M \times M \to \mathbb{S}$ that satisfies $\mathcal{K}(t, n, m) \longrightarrow \delta(n, m)$ for $t \longrightarrow 0$. Then

$$f(n, m) = \partial_t \mathcal{K}(t = 0; n, m) = \lim_{dt \to 0} \frac{\mathcal{K}(dt, n, m) - \delta(n, m)}{dt}. \tag{4.2.67}$$

And the evolution system with the evolution operator $\mathcal{E}_{dt}$ defined by the function $\mathcal{K}$,

$$\mathcal{S}(t + dt, m) = (\mathcal{E}_{dt}\mathcal{S})(t, m) = \int_M \mathcal{S}(t, n)\, \mathcal{K}(dt, n, m)\, d\mu(n) \tag{4.2.68}$$

is the "antiderivative" of the original evolution equation because

$$\underbrace{\frac{\mathcal{S}(t + dt, m) - \mathcal{S}(t, m)}{dt}}_{\longrightarrow\, \partial_t \mathcal{S}(t,m)} = \int_M \underbrace{\frac{\mathcal{K}(dt, n, m) - \delta(n, m)}{dt}}_{\longrightarrow\, \partial_t \mathcal{K}(t=0,n,m) = f(n,m)} \mathcal{S}(t, n)\, d\mu(n). \tag{4.2.69}$$

Since the kernels $\mathcal{K}_{dt}$ form a semigroup $(\mathcal{E}_{dt})$, f can be interpreted as the "infinitesimal kernel".

Example 4.2.17. The function $\mathcal{S}$ – if smooth enough – can be approximated using the Taylor polynomial

$$\mathcal{S}(t, m+v) \approx \mathcal{S}(t,m) + v \cdot \nabla_m \mathcal{S}(t,m) + \tag{4.2.70}$$

$$+ \frac{1}{2} \sum_{j=1}^{d} \sum_{k=1}^{d} \partial_{m_j, m_k} \mathcal{S}(t,m) v_j v_k + \dots \tag{4.2.71}$$

This means that Equation 4.2.66 can be approximated for example (first order polynomial) by

$$\partial_t \mathcal{S}(t,m) = \int_V \Big(\mathcal{S}(t,m) + v \cdot \nabla_m \mathcal{S}(t,m) \Big) f(v)\, dv \tag{4.2.72}$$

$$= \mathcal{S}(t,m) \int_V f(v)\, dv + \nabla_m \mathcal{S}(t,m) \int_V v f(v)\, dv \tag{4.2.73}$$

if f does not depend on the location. Accordingly an integral equation can under certain conditions be approximated by a partial differential equation.

4.3 Application Scenarios and Outlook

The last section in this chapter provides some application scenarios and underlines the main idea behind evolution systems, namely to provide an analytical formalism for the gap between continuous evolution systems and in particular implicit evolution equations on the one side and cellular automata on the other.

4.3.1 A Simple Classification of Evolution Systems

Depending on the structure of the topological vector space M there exist different ways to define a Banach space $\mathfrak{S} \subseteq \mathbb{S}^M$ which is a necessary precondition for evolution systems and semigroup theory. The following list categorises different mathematical approaches as evolution systems based on the structure of the underlying space.

- Trivial case: $\mathfrak{S}$ is a Banach space.
  - abstract Cauchy problems (Section 4.2.3.1)
  - one-parameter semigroups in general
- If M is not discrete, then M must be a domain in a compact Hausdorff space. Let $\mathbb{S} = \mathbb{R}$ and set $\mathfrak{S} := C(M)$ which together with $\|\mathcal{S}\|_\infty := \sup_{m \in M} |\mathcal{S}(m)|$ is a Banach space.
  - classical solutions of parabolic partial differential equations (Section 4.2.3.2)

- Also if $\mathbb{S} \neq \mathbb{R}$, $C(M;\mathbb{S})$ might be a Banach space (compare [8, p. 285]).
- Let $\mathfrak{S}$ be the Lebesgue space $L^p(M)$ where M is a suitable measure space.
  - integral equations (Section 4.2.3.3)
  - weak solutions of parabolic partial differential equations (Section 4.2.3.2)
- Assume that $(\mathbb{S}, \|\cdot\|_{\mathbb{S}})$ is an arbitrary Banach space. If $(M, \mathfrak{M}, \mu)$ is a measure space define $\mathfrak{S}$ as the Bochner-Lebesgue[9] space $L^p(M;\mathbb{S})$ with $\|\mathcal{S}\|_p := \int_M \|\mathcal{S}(m)\|_{\mathbb{S}}^p \, d\mu(m)$ which is also a Banach space.
- If M is a discrete finite space[10] (e.g. $\subset \mathbb{Z}^d$), let $\mathfrak{S}$ be the Banach space $\mathbb{S}^{|M|}$.
  - (ordinary) cellular automata (Section 2.1)
  - method of lines, finite difference method
- Let M be discrete finite set equipped with an arbitrary graphical structure.
  - unaligned cellular automata (Section 2.2)
  - finite element method

4.3.2 Quasilinear Evolution Systems

Cellular automata are often nonlinear systems or approximate nonlinear equations. But such an application scenario conflicts with the definition of evolution systems, which relies on the basic theory of strongly continuous semigroups and hence linear evolution operators on $\mathfrak{S}$.

However under certain conditions we can try to maintain the basic characteristics of evolution systems even if the evolution operators are not strictly linear maps. We will therefore require a specific kind of nonlinearity in order to ease formalisation, which actually might not even be a necessity for nonlinear evolution systems. Such evolution systems will be called *quasilinear* or *spatially linear* evolution systems.

Definition 4.3.1 (Quasilinear Evolution System)**.** We call an evolution system *quasilinear* if the scalar evolution operators can be written as

$$\mathcal{L}_{dt,m}\mathcal{S} = \Big(\underbrace{\mathcal{F}_{dt}\big(\mathcal{S}(m)\big) \circ \mathrm{proj}_{\mathcal{N}_{dt}(m)}}_{=:\mathcal{L}^H_{dt,m}(\mathcal{S}(m))} \Big) \mathcal{S} + \mathcal{G}_{dt}\big(\mathcal{S}(m)\big) \tag{4.3.1}$$

where $\mathcal{F}_{dt}(s) : \mathbb{S}^{N_{dt}} \to \mathbb{S}$ is a linear map for all $s \in \mathbb{S}$ and $\mathcal{G}_{dt} : \mathbb{S} \to \mathbb{S}$. The *homogeneous part* $\mathcal{L}^H_{dt,m}(s)$ is a linear mapping $\mathfrak{S} \to \mathbb{S}$ for all $s \in \mathbb{S}$. In this case a future state at location m, $\mathcal{S}_{t+dt}(m)$ can depend on $\mathcal{S}_t(m)$ in a nonlinear fashion.

[9] Bochner-Lebesgue spaces are a simple extension of Lebesgue spaces for functions with values in arbitrary Banach spaces other than $\mathbb{R}$.

[10] A discrete space is always Hausdorff. A discrete space is compact if and only if it is finite.

The evolution operators of such a system can also be written as

$$\mathcal{E}_{dt}\mathcal{S} = \prod_{m\in M} \mathcal{L}_{dt,m}\mathcal{S} = \underbrace{\prod_{m\in M} \mathcal{L}^H_{dt,m}(\mathcal{S}(m))}_{=:\mathcal{E}^H_{dt}(\mathcal{S})}\ \mathcal{S} + \underbrace{\prod_{m\in M} \mathcal{G}_{dt}\big(\mathcal{S}(m)\big)}_{=:\mathcal{E}^I_{dt}(\mathcal{S})} \tag{4.3.2}$$

where $\mathcal{E}^H_{dt}(s_M) \in L(\mathfrak{S},\mathfrak{S})$ for all $s_m \in \mathbb{S}^M$ and $\mathcal{E}^I_{dt} : \mathfrak{S} \to \mathfrak{S}$.

Since obviously $\mathcal{S}_{t+dt}(m)$ depends on $\mathcal{S}_t(m)$, it is necessary that $m \in \mathcal{N}(m)$ in order to comply with the definition of neighbourhoods. Additionally if $m \notin \mathcal{N}(m)$, then it can happen that the $\mathcal{E}_{dt}$ are linear.

A quasilinear integral evolution system can be formalised as

$$\mathcal{E}_{dt}\mathcal{S}(m) = \mathcal{L}_{dt,m}\mathcal{S} = \int_{\mathcal{N}(m)} \mathcal{K}_{dt}\big(\mathcal{S}(m), n, m\big)\mathcal{S}(n)\ dn + \mathcal{G}_{dt}\big(\mathcal{S}(m)\big). \tag{4.3.3}$$

The main drawback in contrast to fully linear evolution systems is that standard semigroup theory is not applicable because the evolution operators $\mathcal{E}_{dt}$ are not linear. If $\mathcal{G}_{dt} = 0$, the linear operators $\mathcal{E}^H_{dt}(s)$ still vary over time, which also does not comply with the standard concept of semigroups. It is also not clear if and how infinitesimal generators can be formalised and in which way an infinitesimal generator would depend on the local state.

Note that this formalisation of quasilinear evolution systems corresponds to the classification of (parabolic) partial differential equations. Instead of differential operators, we deal with locally characterised evolution operators. Again this leads to the question whether basic concepts form parabolic partial differential equations (p.e. weak formulation, Galerkin approximation, maximum principles, etc.) can be adopted for evolution systems and cellular automata.

4.3.3 Discretisation of Evolution Systems

Throughout Section 4.3.3 continuous state functions (e.g. $M \subset \mathbb{R}^d$) are written using standard capital letters $S : M \to \mathbb{S}$ and discrete state mappings (e.g. $M \subset \mathbb{Z}^d$) using the calligraphic typeface $\mathcal{S} : M \to \mathbb{S}$. The same is true for all kind of operators.

The straight forward discretisation for evolution systems is time-discretisation. Under the circumstances that the evolution operators E_{dt} form a contraction semigroup it is sufficient to choose a $dt \in T - T$, without loss of generality $dt := 1$, and to define the iterative system

$$S_{t_{n+1}}(m) = E_1 S_{t_n}(m). \tag{4.3.4}$$

The resulting states satisfy $S_{t_n}(m) = E_{t_n} S_0(m)$ and accordingly form an exact time-discretisation of the original evolution system.

This situation is however only theoretical. Usually evolution systems are

- either given in an implicit form as evolution equations and do not allow an explicit formulation in terms of evolution operators (analytic solution),

- or the semigroup property $E_{s+t} = E_s E_t$ is by design (model) not an exact equation but only an acceptable approximation for small s and t.

In order to obtain a cellular automaton from an evolution system it is also necessary to have a discrete domain M. Independently of the discretisation approach, a locally characterised iterative discrete system must be obtained. Local characterisation is

- either given by design (model),
- obtained by approximating a not locally characterised evolution system with a locally characterised one
- or by choosing a discretisation method that yields spatially restricted neighbourhoods.

The latter two cases can be found in the following linear example.

Example 4.3.2. A very prominent example for a not locally characterised evolution system is the Gaussian diffusion semigroup ($M = \mathbb{R}^d$, $\mathbb{S} = \mathbb{R}$, $\mathfrak{S} = L^1(M)$)

$$(E_{dt}S)(t,m) = \int_{\mathbb{R}^d} (4\pi\, dt)^{-\frac{d}{2}} e^{-\frac{\|m-n\|^2}{4\,dt}} S(t,n)\; dn. \tag{4.3.5}$$

The most obvious discretisation ($dt = 1$) of this system has the form

$$\mathcal{S}(t_{i+1},m) = \sum_{n\in\mathbb{Z}^d} \alpha(n,m)\mathcal{S}(t_i,n) \tag{4.3.6}$$

where $\alpha(n,m)$ describes suitable coefficients (compare Section 4.3.3.4). A locally characterised approximation can be written as

$$\mathcal{S}(t_{i+1},m) = \sum_{n\in\mathcal{N}(m)} \beta(n,m)\mathcal{S}(t_i,n). \tag{4.3.7}$$

Using Fourier transformation it can be shown [8] that a solution of the diffusion or heat equation

$$\partial_t S(t,m) = \Delta S(t,m) \tag{4.3.8}$$

with initial condition $S(0,m)$ satisfies

$$S(t,m) = \int_{\mathbb{R}^d} (4\pi\, t)^{-\frac{d}{2}} e^{-\frac{\|m-n\|^2}{4\,t}} S(0,n)\; dn. \tag{4.3.9}$$

Obviously Equation 4.3.5 and Equation 4.3.8 describe the same evolution system.

Also for the heat equation there exists a straight forward discretisation ($\mathbb{Z}^d \subset \mathbb{R}^d$) method: the method of finite differences

$$\mathcal{S}(t_{i+1},m) = \sum_{n=1}^{d} \Big(\mathcal{S}(t_i,m+e_n) - 2\mathcal{S}(t_i,m) + \mathcal{S}(t_i,m-e_n)\Big) \tag{4.3.10}$$

which by definition is a locally characterised discrete evolution system.

Even for a spatially interpreted domain in $\mathbb{R}^d$ there exist different methods to discretise functions $S : \mathbb{R}^d \to \mathbb{S}$:

- interpolation approach: $\mathcal{S}(z) := S(z)$ for $z \in \mathbb{Z}^d$
- $\mathcal{S}(z) := \big(S(z), S^{(1)}(z), S^{(2)}(z)\big)$ for $z \in \mathbb{Z}^d$
- volumetric: $\mathcal{S}(z) := \int_{[z_1-\frac{1}{2},z_1+\frac{1}{2})\times\cdots\times[z_d-\frac{1}{2},z_d+\frac{1}{2})} S(x)\, dx$
- ...

In general a discretisation of $M \subset \mathbb{R}^d$ can be interpreted as the approximation of state functions S by a parametrised set of functions $\mathcal{S}_p$ on $[z_1, z_1+1)\times\cdots\times[z_d, z_d+1)$ such that $\|S - \mathcal{S}_p\|$ is minimal for a certain norm.

The choice for a specific discretisation method is of course influenced during the process of modelling. Also the discretisation of the domain M itself must not necessarily yield a regular lattice (compare the finite element method) despite this is the most obvious approach.

As indicated before in an ideal situation, the error $\|S(t,x) - \mathcal{S}(t,z)\|$ should be minimal independently of $t \in T$.

4.3.3.1 Finite Difference Discretisation of Partial Differential Equations

For parabolic differential equations the method of finite differences is the standard discretisation method. For example [14] provides advanced finite difference discretisation methods for abstract parabolic differential equations with nonconstant coefficients and inhomogeneities.

Regard the equation

$$\partial_t S = A(t,S)S + G(t,S) \tag{4.3.11}$$

with differential operator A. Usually the time derivative is replaced by a Euler forward discretisation scheme, which also implies a linearisation of the right hand side.

$$S(t+dt,x) = S(t,x) + dt\, A\big(t,S(t,x)\big)S(t,x) + dt\, G\big(t,S(t,x)\big) \tag{4.3.12}$$

Differential operators in the right hand side are discretised using finite differences which finally yields a difference equation.

4.3.3.2 Discretisation of Linear Evolution Operators

Usually a spatial discretisation yields a finite number of cells. If $\mathbb{S}$ is one-dimensional, then $\mathbb{S}^M$ is a finite-dimensional vector space over $\mathbb{S}$ which means that a linear map $\mathcal{E} \in L(\mathfrak{S},\mathfrak{S})$ has a matrix representation and the iteration of global states can be written as

$$\mathcal{S}_{t+1} = \mathcal{E}\cdot\mathcal{S}_t \tag{4.3.13}$$

respectively

$$\mathcal{S}_{t+1}(m) = \mathcal{L}_m \cdot \mathcal{S}_t \tag{4.3.14}$$

which means that the scalar evolution operators $\mathcal{L}_m$ are the row vectors of the *evolution matrix* $\mathcal{E}$. According to the matrix representation there exists an indexing or ordering of the cells m_i where $i \in \{1, \ldots, |M|\}$ such that every cell m_i corresponds to a unit vector e_i.

If $\mathbb{S}$ is multi-dimensional, then $\mathcal{E}$ is a tensor respectively has a tensor representation but still every cell is associated with a unique index in $\mathbb{N}$.

Let us now for simplicity assume that $\mathbb{S}$ is one-dimensional – otherwise we have to deal with tensors instead of matrices. We can write the vector-vector multiplication from Equation 4.3.14 as

$$\mathcal{S}_{t+1}(m_i) = \mathcal{L}_{m_i} \cdot \mathcal{S}_t = \sum_{j=1}^{|M|} l_{i,j}\, \mathcal{S}_t(m_j) = \sum_{m_j \in \mathcal{N}(m_i)} l_{i,j}\, \mathcal{S}_t(m_j) \tag{4.3.15}$$

since the scalar components $l_{i,j}$ of the evolution matrix are non-zero only if $m_j \in \mathcal{N}(m_i)$. Accordingly an evolution matrix is usually sparsely occupied.

Corollary 4.3.3 (Evolution Matrix)**.** *The evolution matrix of a cellular automaton (discretised evolution system) with – necessarily – linear evolution operator has the same shape as the adjacency matrix. For regularly arranged cellular automata (regular discretisation), "matrix" can be replaced with "tensor".*

Proof. Follows from the definitions in Section 3.2. □

Note that for a regular discretisation the resulting evolution matrix features very specific characteristics (Section 3.2.3).

4.3.3.3 Discretisation of Quasilinear Evolution Operators

At this point at the latest we can recognise that linear evolution systems are not the main application area for cellular automata. Because usually a cellular automaton is not a linear system.

For quasilinear systems a discretisation has the form

$$\mathcal{S}_{t+1} = \mathcal{E}^H(\mathcal{S}_t) \cdot \mathcal{S}_t + \mathcal{E}^I(\mathcal{S}_t) \tag{4.3.16}$$

with a matrix $\mathcal{E}^H(s_M) \in \mathbb{S}^{|M|\times|M|}$ for all $s_M \in \mathbb{S}^{|M|}$ with the same characteristics as the evolution matrix in the previous section (e.g. Corollary 4.3.3) and a component wise mapping $\mathcal{E}^I : \mathbb{S}^{|M|} \to \mathbb{S}^{|M|}$.

In scalar form,

$$\mathcal{S}_{t+1}(m) = \mathcal{L}^H_m\big(\mathcal{S}(m)\big) \cdot \mathcal{S}_t + \mathcal{G}\big(\mathcal{S}(m)\big) \tag{4.3.17}$$

$$= \mathcal{F}\big(\mathcal{S}(m)\big) \cdot \big(\operatorname{proj}_{\mathcal{N}(m)} \mathcal{S}_t\big) + \mathcal{G}\big(\mathcal{S}(m)\big) \tag{4.3.18}$$

where $\mathcal{F}(s) \in \mathbb{S}^{1\times k}$ (for equally sized neighbourhoods), $\mathcal{L}^H_m(s) \in \mathbb{S}^{1\times|M|}$ and $\mathcal{G} : \mathbb{S} \to \mathbb{S}$.

4.3.3.4 Discretisation of Integral Evolution Operators

Regard the following integral evolution system with $M \subset \mathbb{R}^d$,

$$E_{dt}S(m) = L_{dt,m}S = \int_{\mathbb{R}^d} \kappa_{dt}(v)\, S(m+v)\, dv. \tag{4.3.19}$$

A discretisation for example minimises $\|S(x) - \mathcal{S}(z)\|_{L^1}$ on cubes $x \in [z_1 - \frac{1}{2}, z_1 + \frac{1}{2}) \times \cdots \times [z_d - \frac{1}{2}, z_d + \frac{1}{2})$ with volume 1. A possible discretisation of the evolution operator respectively the integral kernel is given by

$$\alpha_{dt}(w) := \int_{w+[-\frac{1}{2},\frac{1}{2})^d} \kappa_{dt}(v)\, dv \tag{4.3.20}$$

and

$$\mathcal{E}_{dt}\mathcal{S}(m) = \mathcal{L}_{dt,m}\mathcal{S} = \sum_{v\in\mathbb{Z}^d} \alpha_{dt}(v)\, \mathcal{S}(m+v) \tag{4.3.21}$$

for discrete $m \in M \cap \mathbb{Z}^d$.

For simplicity assume that κ is a probability density on V, $\|\kappa\|_{L^1(V)} = 1$. In order to maintain this normalisation for a local characterisation of κ we can define

$$\alpha_{dt}(w) := \begin{cases} C \cdot \int_{w+[-\frac{1}{2},\frac{1}{2})^d} \kappa_{dt}(v)\, dv & \|w\| \le \rho \\ 0 & \|w\| > \rho \end{cases} \tag{4.3.22}$$

where C is the inverse of the integral of κ over the corresponding set of cubes. C should ideally be as close to 1 as possible. This can for example be achieved by a good (large) choice of ρ which may depend on dt.

In Example 4.2.17 a polynomial approximation of S was used to obtain a differential equation from an integral equation. Applying the method of finite differences on the differential equation yields a locally characterised discrete evolution system also if the kernel of the original integral formulation is not locally characterised.

If κ is a probability density function, then the coefficients $\alpha(v)$ are composed of the moments of the corresponding probability distribution. In [44] and Section 4.3.4 it is shown that a symmetric distribution greatly simplifies the resulting discrete evolution system respectively the coefficients.

4.3.3.5 Outlook: Stochastic Approximation of Integral Evolution Operators

A further method for discretising an integral evolution system with a probability density function as kernel relies on a stochastic interpretation of the evolution process. Let $N_{dt} : (\Omega, \mathcal{P}) \to V$ be random variables with densities $\kappa_{dt} : V \to \mathbb{R}_+$, then

$$E_{dt}S(m) = L_{dt,m}S = \int_V \kappa_{dt}(v)\, S(m+v)\, dv = \mathbb{E}\big[S(m+N_{dt})\big]. \tag{4.3.23}$$

We can say that N_{dt} randomly selects an element of V or that N_{dt} describes a *stochastic neighbourhood*.

Let $v_1, \dots, v_k$ be realisations of N_{dt}, then for $k \longrightarrow \infty$

$$\frac{1}{k}\sum_{i=1}^{k} S(m+v_i) \longrightarrow \mathbb{E}\big[S(m+N_{dt})\big] \tag{4.3.24}$$

according to the law of large numbers.

The same results also apply for discrete random variables $N_{dt} : (\Omega, \mathcal{P}) \to V \subset \mathbb{Z}^d$ and accordingly also for discretised evolution systems. Furthermore for a finite set of realisations, the support of the density must not necessarily be bounded in order to obtain a bounded (stochastic) neighbourhood.

This approach was successfully used in [44].

If not a probability density but a function $V \to [0,1]$, κ_{dt} can be interpreted as a membership function [45] of a fuzzy neighbourhood set. However in both cases further investigations are out of scope for this thesis but could provide interesting results.

4.3.4 Application Example: Reaction Diffusion System

Regard the following quasilinear parabolic partial differential equation for $S : T \times M \to \mathbb{R}^2$ where M is a domain in $\mathbb{R}^d$.

$$\partial_t S_1(t,x) = \big(D_1 \cdot S(t,x)\big)\Delta S_1(t,x) + G_1\big(S(t,x)\big) \tag{4.3.25}$$

$$\partial_t S_2(t,x) = \big(D_2 \cdot S(t,x)\big)\Delta S_2(t,x) + G_2\big(S(t,x)\big). \tag{4.3.26}$$

The time discretisation of this system ($i = 1, 2$) is given by

$$S_i(t+dt,x) = S_i(t,x) + dt\,\big(D_i \cdot S(t,x)\big)\Delta S_i(t,x) + dt\,G_i\big(S(t,x)\big). \tag{4.3.27}$$

Assume that there exists an integral representation in the form

$$S_i(t+dt,x) = \int \kappa_{i,dt}(v) S_i(t,x+v)\,dv + dt\,G_i\big(S(t,x)\big). \tag{4.3.28}$$

Because we deal with diffusion, we can assume that $\kappa_{i,dt}$ (compare Example 4.3.2) are Gaussian densities (variance $\sigma_i^2 = 2\tilde{D}_i dt$)

$$\kappa_{i,dt}(v) = (4\tilde{D}_i dt\,\pi)^{-\frac{d}{2}} e^{-\frac{\|v\|^2}{4\tilde{D}_i dt}}. \tag{4.3.29}$$

The second order Taylor expansion of S_i, $i = 1, 2$ is

$$S_i(t,x+v) \approx\ S_i(t,x) + \sum_{j=1}^{d} v_j\,\partial_{x_j} S_i(t,x) + \tag{4.3.30}$$

$$+\frac{1}{2}\sum_{j=1}^{d} v_j^2\,\partial_{x_j x_j} S_i(t,x) + \frac{1}{2}\sum_{\substack{j,l\in\{1,\dots,d\}\\ j\neq l}} v_j v_l\,\partial_{x_j x_l} S_i(t,x) \tag{4.3.31}$$

and substituting in Equation 4.3.28 yields

$$S_i(t+dt,x) = S_i(t,x)\underbrace{\int \kappa_{i,dt}(v)\,dv}_{①} + \sum_{j=1}^{d} \partial_{x_j} S_i(t,x) \underbrace{\int v_j\,\kappa_{i,dt}(v)\,dv}_{②} + \tag{4.3.32}$$

$$+\frac{1}{2}\sum_{j=1}^{d} \partial_{x_j x_j} S_i(t,x) \underbrace{\int v_j^2\,\kappa_{i,dt}(v)\,dv}_{③} + \tag{4.3.33}$$

$$+\frac{1}{2}\sum_{\substack{j,l\in\{1,\dots,d\}\\ j\neq l}} \partial_{x_j x_l} S_i(t,x) \underbrace{\int v_j v_l\,\kappa_{i,dt}(v)\,dv}_{④} + dt\,G_i\big(S(t,x)\big). \tag{4.3.34}$$

Because $\kappa_{i,dt}$ usually are radial symmetric probability densities we have

$$① = 1 \tag{4.3.35}$$

$$② = ④ = 0 \tag{4.3.36}$$

$$③ = \sigma_i^2 = 2\tilde{D}_i dt \tag{4.3.37}$$

and accordingly Equation 4.3.28 can be approximated by

$$S_i(t+dt,x) = S_i(t,x) + dt\,\tilde{D}_i \Delta S_i(t,x) + dt\,G_i\big(S(t,x)\big). \tag{4.3.38}$$

If we finally set $\tilde{D}_i := D_i \cdot S(t,x)$, the original partial differential equation approximates the integral evolution system. The kernels must then be written as

$$\kappa_{i,dt}\big(S(t,x),v\big) = \big(4dt\pi D_i \cdot S(t,x)\big)^{-\frac{d}{2}} e^{-\frac{\|v\|^2}{4dtD_i\cdot S(t,x)}} \tag{4.3.39}$$

which also means that the variance of the diffusion distribution $\sigma_i^2 = 2dtD_i \cdot S(t,x)$ depends on the density S at the actual location. This however suits the definition of quasilinear evolution systems in Definition 4.3.1.

4.3.4.1 Discretisation Approaches – Cellular Automata

The finite difference discretisation of the differential equation yields the following difference equation:

$$S_i(t+dt,x) = S_i(t,x) + dt\,G_i\big(S(t,x)\big) + \tag{4.3.40}$$

$$+\,dt\,D_i \cdot S(t,x) \sum_{j=1}^{d} \frac{1}{(dx)^2}\Big(S_i(t,x+dx\,e_j) + S_i(t,x-dx\,e_j) - 2S_i(t,x)\Big) \tag{4.3.41}$$

The integral evolution system can be approximated using the following simple cubic ($N_{dt} = [-\rho,\rho]^d$) stencil method – neglecting normalisation –

$$S_i(t+dt,x) = dt\,G_i\big(S(t,x)\big) + \tag{4.3.42}$$

$$+ \sum_{l_d=-\rho}^{\rho} \cdots \sum_{l_1=-\rho}^{\rho} \kappa_{i,dt}\big(S(t,x), \sum_{j=1}^{d} l_j dx\,e_j\big)\; S_i(t, x + \sum_{j=1}^{d} l_j dx\,e_j). \tag{4.3.43}$$

A stochastic approximation of the integral evolution system can be obtained using random variables $N_{i,dt} \sim \kappa_{i,dt}\big(S(t,x),\,\cdot\,\big)$ respectively discrete realisations $v_1, \ldots, v_k$ by

$$S_i(t+dt,x) = \frac{1}{k}\sum_{l=1}^{k} S_i(t, x+v_i) + dt\, G_i\big(S(t,x)\big) \longrightarrow \tag{4.3.44}$$

$$\longrightarrow \mathbb{E}\big[S_i(t, x+N_{i,dt})\big] + dt\, G_i\big(S(t,x)\big). \tag{4.3.45}$$

4.3.4.2 Outlook: A Further Application Example

This approach was also used in [44] on a model for spatial epidemic spread, which simulates spatially distributed infections on a three compartment population (susceptible, infected, recovered – also known as SIR-type epidemic models).

$$S(t+dt,x) = S(t,x) \quad - S(t,x)\int \alpha_{dt}(v) I(t,x+v), dv \tag{4.3.46}$$

$$I(t+dt,x) = I(t,x) \quad + S(t,x)\int \alpha_{dt}(v) I(t,x+v), dv \quad -\beta_{dt} I(t,x) \tag{4.3.47}$$

$$R(t+dt,x) = R(t,x) \quad +\beta_{dt} I(t,x) \tag{4.3.48}$$

Chapter 5

Stochastic Cellular Automata

Because in this chapter stochastic cellular automata will be defined as stochastic processes, a short compendium on probability theory and stochastic processes precedes the definition of stochastic cellular automata.

5.1 Formalisms of Probability and Stochastic Processes

This section recapitulates or introduces basic concepts of probability theory and stochastic processes in a general form.

5.1.1 Basic Definitions

The following definitions serve as an introduction and should clarify notational conventions. Especially the formulation of conditional probability can be rather complicated.

Definition 5.1.1 (Probability Distribution [27, 28, 29])**.** Given a probability space $(\Omega, \mathfrak{A}, \mathcal{P})$ and a measurable space $(\Upsilon, \mathfrak{B})$, a measurable function $X : (\Omega, \mathfrak{A}) \to (\Upsilon, \mathfrak{B})$ is called *random variable*. A *probability distribution* on $(\Upsilon, \mathfrak{B})$ is defined through the push-forward measure $P_X := \mathcal{P} \circ X^{-1}$.

Accordingly a probability distribution is always associated with a probability measure, which can be either discrete $P : \Upsilon \to [0, 1]$ or continuous $P : \mathfrak{B} \to [0, 1]$.

Definition 5.1.2 (Expectation Value, compare [27])**.** The *expectation value* of a random variable X is defined as

$$\mathbb{E}[X] := \int_\Omega X(\omega)\, d\mathcal{P}(\omega) = \int_\Omega X(\omega)\, \mathcal{P}(d\omega) = \int_\Omega X\, d\mathcal{P}. \tag{5.1.1}$$

Definition 5.1.3 (Stochastic Independence [27])**.** A countable collection of sub-σ-algebras $\mathfrak{A}_T := (\mathfrak{A}_t)_{t\in T}$ is *independent* if for each finite subset $T' \subseteq T$

$$\mathcal{P}\Big(\bigcap_{t\in T'} A_t\Big) = \prod_{t\in T'} \mathcal{P}(A_t) \qquad \forall\, (A_t)_{t\in T'} \in \prod_{t\in T'} \mathfrak{A}_t. \tag{5.1.2}$$

A countable collection of random variables is independent if the induced collection of σ-algebras is independent. A countable collection of events $(A_t)_{t\in T}$ is independent if Equation 5.1.2 holds for all finite subcollections. For two independent sub-σ-algebras, random variables respectively events the notation $A \perp\!\!\!\perp B$ is used.

5.1.1.1 Conditional Probability

In a direct approach the *conditional probability* of an event A given another event B is defined as

$$\mathcal{P}(A|B) = \frac{\mathcal{P}(A \cap B)}{\mathcal{P}(B)}. \tag{5.1.3}$$

A more abstract approach to conditional probability is based on *conditional expectation.*

Definition 5.1.4 (Conditional Expectation [27, 29])**.** Let $X : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\Upsilon, \mathfrak{B})$ be a random variable. The *conditional expectation of X given* $\mathfrak{A}' \subseteq \mathfrak{A}$ is a $\mathfrak{A}'$-measurable random variable denoted $\mathbb{E}[X|\mathfrak{A}'] : (\Omega, \mathfrak{A}', \mathcal{P}) \to (\Upsilon, \mathfrak{B})$ for which equally either

(i) $\mathbb{E}\big[\mathbb{I}_{A'}\mathbb{E}[X|\mathfrak{A}']\big] = \mathbb{E}\big[\mathbb{I}_{A'}X\big]$ for all $A' \in \mathfrak{A}'$ or

(ii) $\int_{A'} \mathbb{E}[X|\mathfrak{A}']\, d\mathcal{P} = \int_{A'} X\, d\mathcal{P}$ for all $A' \in \mathfrak{A}'$.

The random variable $\mathbb{E}[X|\mathfrak{A}']$ is only almost surely[1] unique.

If Y is another random variable $(\Omega, \mathfrak{A}, \mathcal{P}) \to (\Upsilon, \mathfrak{B})$, we define $\mathbb{E}[X|Y] : (\Upsilon, \mathfrak{B}) \to (\Upsilon, \mathfrak{B})$ such that

$$\int_{Y^{-1}(B)} X(\omega)\, d\mathcal{P}(\omega) = \int_B \mathbb{E}[X|Y](\upsilon)\, dP_Y(\upsilon) \qquad B \in \mathfrak{B}. \tag{5.1.4}$$

It is possible to show that $\mathbb{E}[X|Y] \circ Y$ is equal to the conditional expectation $\mathbb{E}[X|\sigma(Y)]$ by regarding the transformation

$$\int_{A'} \big(\mathbb{E}[X|Y] \circ Y\big)(\omega)\, d\mathcal{P}(\omega) = \int_{Y(A')} \mathbb{E}[X|Y](\upsilon)\, d\mathcal{P} \circ Y^{-1}(\upsilon) \tag{5.1.5}$$

where $A' \in \sigma(Y)$ and $\sigma(Y) \subseteq \mathfrak{A}$ is the σ-algebra induced by Y on Ω with $Y^{-1}(\mathfrak{B}) \subseteq \sigma(Y) \subseteq \mathfrak{A}$. Also the notations $\mathbb{E}[X|Y = Y(\omega)] = \mathbb{E}[X|Y] \circ Y(\omega)$ and $\mathbb{E}[X|Y = \upsilon] = \mathbb{E}[X|Y](\upsilon)$ shall be used and $\mathbb{E}[X|Y]$ can be called *conditional expectation of X given* Y.

Motivated by $\mathbb{E}[\mathbb{I}_A] = \mathcal{P}(A)$, *conditional probability* can also be formalised in the following way (compared to Equation 5.1.3).

Definition 5.1.5 (Conditional Probability [27, 29])**.** The *conditional probability with respect to* $\mathfrak{A}' \subseteq \mathfrak{A}$ is a function $\mathcal{P}(\,\cdot\,|\mathfrak{A}')(\,\cdot\,) : \mathfrak{A} \times \Omega \to [0, 1]$ and equally either defined as

[1] An expression holds almost surely (a.s.) or almost everywhere if there exists a zero-measure set N such that the expression is valid for all elements in the complement N^c [6, p. 140].

(i) $\mathcal{P}(A|\mathfrak{A}') := \mathbb{E}[\mathbb{I}_A|\mathfrak{A}']$ for all $A \in \mathfrak{A}$ or such that

(ii) $\int_{A'} \mathcal{P}(A|\mathfrak{A}')(\omega)\, d\mathcal{P}(\omega) = \mathcal{P}(A \cap A')$ for all $A \in \mathfrak{A}$ and $A' \in \mathfrak{A}'$.

According to Definition 5.1.4 the conditional probability of A given $\mathfrak{A}'$ is a random variable $\mathcal{P}(A|\mathfrak{A}') : (\Omega, \mathfrak{A}') \to ([0,1], \mathfrak{B}_{[0,1]})$ where $\mathfrak{B}_{[0,1]}$ is the corresponding Borel σ-algebra and $\mathbb{E}[\mathcal{P}(A|\mathfrak{A}')] = \mathbb{E}[\mathbb{I}_A] = \mathcal{P}(A)$.

For a random variable $Y : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\Upsilon, \mathfrak{B})$ we can introduce the notations $\mathcal{P}(A|Y) \circ Y = \mathcal{P}\big(A\big|\sigma(Y)\big)$ where $\sigma(Y) = \sigma\big(Y^{-1}(\mathfrak{B})\big) \subseteq \mathfrak{A}$ as well as $\mathcal{P}(A|Y = \upsilon) = \mathcal{P}(A|Y)(\upsilon)$ analogously to the expectation value. In this case $\mathcal{P}\big(A\big|Y = Y(\omega)\big) = \mathcal{P}(A|Y) \circ Y(\omega)$,

$$\mathcal{P}\big(A \cap Y^{-1}(B)\big) = \int_{Y^{-1}(B)} \mathcal{P}\big(A\big|\sigma(Y)\big)(\omega)\, d\mathcal{P}(\omega) = \int_B \mathcal{P}(A|Y)(\upsilon)\, dP_Y(\upsilon) \quad (5.1.6)$$

for $B \in \mathfrak{B}$ and

$$\mathcal{P}(A|Y = \upsilon) = \mathcal{P}\big(A\big|\sigma(Y)\big)(\omega) \quad \omega \in Y^{-1}(\{\upsilon\}). \quad (5.1.7)$$

Definition 5.1.6 (Regular Conditional Probability, compare [23])**.** If for a conditional probability with respect to $\mathfrak{A}' \subset \mathfrak{A}$ the function $\mathcal{P}(\,\cdot\,|\mathfrak{A}')(\omega) : \mathfrak{A} \to [0,1]$ is a probability measure for all $\omega \in \Omega$ we talk of a *regular conditional probability* (from $(\Omega, \mathfrak{A}')$ to $(\Omega, \mathfrak{A})$) with respect to $\mathfrak{A}'$.

Given a random variable $Y : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\Upsilon, \mathfrak{B})$ a regular conditional probability with respect to Y "from $(\Upsilon, \mathfrak{B})$ to $(\Omega, \mathfrak{A})$" [23] is a function $\mathcal{P}(\,\cdot\,|Y = \,\cdot\,) : \mathfrak{A} \times \Upsilon \to [0,1]$ which

(i) is a measurable function $(\Upsilon, \mathfrak{B}) \to ([0,1], \mathfrak{B}_{[0,1]})$ for all $A \in \mathfrak{A}$,

(ii) is a probability measure on $(\Omega, \mathfrak{A})$ for all $\upsilon \in \Upsilon$ (compare Definition 5.1.6) and

(iii) which satisfies

$$\mathcal{P}\big(A \cap Y^{-1}(B)\big) = \int_B \mathcal{P}(A|Y = \upsilon)\, dP_Y(\upsilon) \quad (5.1.8)$$

for all $A \in \mathfrak{A}$ and $B \in \mathfrak{B}$.

If $Y : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\Omega, \mathfrak{A})$, then $P(\,\cdot\,|Y = \,\cdot\,) : \mathfrak{A} \times \Omega \to [0,1]$ with (i), (ii) and

$$\mathcal{P}\big(A \cap Y^{-1}(A')\big) = \int_{A'} \mathcal{P}(A|Y = \omega)\, dP_Y(\omega) \quad (5.1.9)$$

for all $A \in \mathfrak{A}$ and $A' \in \sigma(Y) \subset \mathfrak{A}$, is a regular conditional probability from $(\Omega, \sigma(Y))$ to $(\Omega, \mathfrak{A})$.

If ultimately $Y = I : \mathfrak{A} \to \mathfrak{A}$, then the equation

$$\mathcal{P}\big(A \cap A'\big) = \int_{A'} \mathcal{P}(A|\omega)\, d\mathcal{P}(\omega) \quad (5.1.10)$$

can be compared to Equation 5.1.3.

In contrast to conditional probability, regular conditional probability allows to formalise $\mathcal{P}(A|\omega)$ even if $\mathcal{P}(\omega) = 0$ respectively $\mathcal{P}(A|X = \upsilon)$ even if $\mathcal{P}(X = \upsilon) = 0$.

Definition 5.1.7 (Conditional Independence [20, 27])**.** Let $\mathfrak{A}_1, \ldots, \mathfrak{A}_{k+1}$ be sub-σ-algebras of $\mathfrak{A}$. $\mathfrak{A}_1, \ldots, \mathfrak{A}_k$ are said to be *conditionally independent given* $\mathfrak{A}_{k+1}$ if

$$\mathbb{E}\Big[\prod_{i=1}^{k} X_i \;\Big|\; \mathfrak{A}_{k+1}\Big] = \prod_{i=1}^{k} \mathbb{E}\left[X_i|\mathfrak{A}_{k+1}\right] \quad \text{a.s.} \tag{5.1.11}$$

for any collection of bounded random variables $X_1, \ldots, X_k$ where $X_i : (\Omega, \mathfrak{A}_i) \to (\Upsilon, \mathfrak{B})$. For two conditionally independent sub-σ-algebras we use the notation $\mathfrak{A}_1 \perp\!\!\!\perp \mathfrak{A}_2 \mid \mathfrak{A}_3$.

Also conditional independence can be formulated for random variables: $X \perp\!\!\!\perp Y \mid Z$ means X and Y are independent given Z; in signs either

$$\mathbb{E}[\mathbb{I}_A \mathbb{I}_B|Z] = \mathbb{E}[\mathbb{I}_A|Z] \cdot \mathbb{E}[\mathbb{I}_B|Z] \quad \text{a.s.} \tag{5.1.12}$$

$$\mathcal{P}(A \cap B|Z) = \mathcal{P}(A|Z) \cdot \mathcal{P}(B|Z) \quad \text{a.s.} \tag{5.1.13}$$

for all $A \in \sigma(X), B \in \sigma(Y)$.

5.1.1.2 Disintegration

This section is mainly based on [9, 23], which provide a complete and compact overview on regular conditional probability and disintegration.

Definition 5.1.8 (Radon Space [6, p. 313])**.** A *Radon space* is a separable[2] metric space, for which every probability measure is *inner regular*, that is, every set can be approximated by compact subsets from within with respect to this probability measure.

Theorem 5.1.9 (Radon Space [23, p. 21])**.** *A separable Hausdorff measurable space* $(\Omega, \mathfrak{A})$ *is Radon (a Radon space) if and only if for every measurable function* Y *from* $(\Omega, \mathfrak{A})$ *into another measurable space* $(\Upsilon, \mathfrak{B})$ *and any probability measure* $\mathcal{P}$ *on* $(\Omega, \mathfrak{A})$ *there exists a regular conditional probability with respect to* Y*,* $\mathcal{P}(\,\cdot\,|Y = \,\cdot\,) : \mathfrak{A} \times \Upsilon \to [0, 1]$.

Proof. See [23]. □

In the following quote the mathematical denotations were adapted to the nomenclature of this thesis. Furthermore regular conditional probability is abbreviated by RCP.

[2] "A topological space is called *separable* if there exists a countable dense subset." [6, p. 242]

> Let $(\Omega, \mathfrak{A})$ be a Radon space with $Y : (\Omega, \mathfrak{A}) \to (\Upsilon, \mathfrak{B})$ a measurable function, where $(\Upsilon, \mathfrak{B})$ is a separable Hausdorff measurable space. If $\mathcal{P}$ is a probability on $(\Omega, \mathfrak{A})$, then it follows from [Theorem 5.1.9] that there exists a RCP $\mathcal{P}(\,\cdot\,|Y = \cdot\,)$ from $(\Upsilon, \mathfrak{B})$ to $(\Omega, \mathfrak{A})$. The added difficulty in disintegration problem is that of delimiting the support of the probability $\mathcal{P}(\,\cdot\,|Y = \upsilon)$. The space Ω is partitioned into the fibers $\{Y^{-1}(\{\upsilon\}) : \upsilon \in \Upsilon\}$. Hence the natural question which appears is if the probability [measure] $\mathcal{P}(\,\cdot\,|Y = \upsilon)$ is concentrated on $Y^{-1}(\{\upsilon\})$ for all $\upsilon \in Y(\Omega)$. In sequence, we shall establish necessary and sufficient conditions for the existence of such RCP. [23, p. 22]

From a less technical but specialised point of view:

> [Regular conditional probability] rigorously defines the idea of a non-trivial 'restriction' of a measure to a measure zero subset of the measure space in question. ... In a sense, 'disintegration' is the opposite process to the construction of a product measure. [48]

$\mathfrak{B}^\star$ shall indicate a σ-algebra on Υ that is complete with respect to all measures on Υ. A function that is measurable with respect to $\mathfrak{B}^\star$ is called *universally measurable* [23]. We further assume that $(\Omega, \mathfrak{A})$ is a Radon space and that Υ is separable and Hausdorff.

Definition 5.1.10 (Universally Measurable Disintegration [23, p. 24])**.** Given a random variable $Y : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\Upsilon, \mathfrak{B})$ a *universally measurable disintegration* with respect to Y from $(\Upsilon, \mathfrak{B})$ to $(\Omega, \mathfrak{A})$ is a function $\mathcal{P}(\,\cdot\,|Y = \cdot\,) : \mathfrak{A} \times \Upsilon \to [0, 1]$ which

(i) is a measurable function $(\Upsilon, \mathfrak{B}^\star) \to ([0, 1], \mathfrak{B}_{[0,1]})$ for all $A \in \mathfrak{A}$,

(ii) is a probability measure on $(\Omega, \mathfrak{A})$ for all $\upsilon \in \Upsilon$ and

(iii) which satisfies

$$\mathcal{P}\big(A \cap Y^{-1}(B)\big) = \int_B \mathcal{P}(A|Y = \upsilon)\, dP_Y(\upsilon) \tag{5.1.14}$$

for all $A \in \mathfrak{A}$ and $B \in \mathfrak{B}$ as well as

(iv) $\mathcal{P}\big(Y^{-1}(\{\upsilon\})\big|Y = \upsilon\big) = 1$ for $\upsilon \in Y(\Omega) \subset \Upsilon$.

A function $\Upsilon \to \Omega$ that assigns an arbitrary element from $\{\omega \in \Omega : Y(\omega) = \upsilon\} = Y^{-1}(\{\upsilon\})$ to each $\upsilon \in Y(\Omega) \subset \Upsilon$ is called *selection function*.

Theorem 5.1.11 (Existence of Universally Measurable Disintegrations [23, p. 25])**.** *Using the preconditions of the previous definition, if there exists a universally measurable selection function $Y(\Omega) \to \Omega$, then there exists a universally measurable disintegration for every probability measure on $(\Omega, \mathfrak{A})$.*

Proof. See [23]. □

It is possible to show that such a measurable selection functions always exists if $(\Omega, \mathfrak{A})$ is a Suslin space[3] [23].

5.1.1.3 Markov Kernels

Definition 5.1.12 (Markov or Probability Kernel [23, 28], compare also [4])**.** Let $(\Omega, \mathfrak{A})$ and $(\Upsilon, \mathfrak{B})$ be measurable spaces. A *Markov kernel from* $(\Omega, \mathfrak{A})$ *to* $(\Upsilon, \mathfrak{B})$ is a function $\mathcal{K} : \Omega \times \mathfrak{B} \to [0, 1]$ which satisfies

(i) $\mathcal{K}(\omega, \cdot\,)$ is a probability measure on $(\Upsilon, \mathfrak{B})$ for every $\omega \in \Omega$ and

(ii) $\mathcal{K}(\cdot\,, B)$ is a measurable function on $(\Omega, \mathfrak{A})$ for every $B \in \mathfrak{B}$.

Accordingly a kernel from $(\Omega, \mathfrak{A}')$ to $(\Omega, \mathfrak{A})$ is a function $\mathcal{K} : \Omega \times \mathfrak{A} \to [0, 1]$ for which $\mathcal{K}(\omega, \cdot\,) : \mathfrak{A} \to [0, 1]$ is a probability measure and $\mathcal{K}(\cdot\,, A) : \Omega \to [0, 1]$ is measurable. Compare the requirements with the definition of disintegration (Definition 5.1.10) and regular conditional probability (Definition 5.1.6).

Proposition 5.1.13 (Regular Conditional Probability Implied by a Markov Kernel, compare [23])**.** *Given a sub-σ-algebra $\mathfrak{A}' \subseteq \mathfrak{A}$, a Markov kernel from $(\Omega, \mathfrak{A}')$ to $(\Omega, \mathfrak{A}, \mathcal{P})$ implies a regular conditional probability with respect to $\mathfrak{A}'$ if for all $A \in \mathfrak{A}$ and $A' \in \mathfrak{A}'$*

$$\mathcal{P}(A \cap A') = \int_{A'} \mathcal{K}(\omega, A)\, d\mathcal{P}(\omega). \tag{5.1.15}$$

Proof. Define $\mathcal{P}(A|\mathfrak{A}')(\omega) := \mathcal{K}(\omega, A)$. Condition (ii) of Definition 5.1.5 is satisfied according to Equation 5.1.15 which renders $\mathcal{P}(\cdot\,|\mathfrak{A}')(\cdot\,)$ a conditional probability. Together with Definition 5.1.12 (i) we have a regular conditional probability. □

Definition 5.1.14 (Composition of Probability Kernels [28])**.** Let $\mathcal{K}_1$ be a probability kernel from $(\Omega, \mathfrak{A})$ to $(\Upsilon, \mathfrak{B})$ and $\mathcal{K}_2$ a probability kernel from $(\Omega \times \Upsilon, \mathfrak{A} \otimes \mathfrak{B})$ to $(\Xi, \mathfrak{C})$ then $\mathcal{K}_1 \otimes \mathcal{K}_2$ defined by

$$(\mathcal{K}_1 \otimes \mathcal{K}_2)(\omega, B) = \int_\Upsilon \mathcal{K}_1(\omega, d\upsilon) \int_\Xi \mathcal{K}_2(\omega, \upsilon, d\xi) \mathbb{I}_B(\upsilon, \xi) = \int_B \mathcal{K}_1(\omega, d\upsilon) \mathcal{K}_2(\omega, \upsilon, d\xi) \tag{5.1.16}$$

for $B \subseteq \Upsilon \times \Xi$ is a probability kernel from $(\Omega, \mathfrak{A})$ to $(\Upsilon \times \Xi, \mathfrak{B} \otimes \mathfrak{C})$.

Definition 5.1.15 (Product of Probability Kernels [28])**.** Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be two probability kernels from $(\Omega, \mathfrak{A})$ to $(\Omega, \mathfrak{A})$. The *product* $\mathcal{K}_1\mathcal{K}_2$ defined by

$$(\mathcal{K}_1\mathcal{K}_2)(\omega, A) = \int \mathcal{K}_1(\omega, d\upsilon) \mathcal{K}_2(\upsilon, A) \tag{5.1.17}$$

is also a probability kernel from $(\Omega, \mathfrak{A})$ to $(\Omega, \mathfrak{A})$. In a not fully exact notion we can also write $(\mathcal{K}_1\mathcal{K}_2)(\omega, A) = (\mathcal{K}_1 \otimes \mathcal{K}_2)(\omega, \Omega \times A)$.

[3] "A Hausdorff space X is called *Suslin space* if there exists a Polish space Y and a continuous surjective mapping $Y \to X$." [6, p. 322]

Proposition 5.1.16 (Cartesian Product of Probability Kernels)**.** *Let $\mathcal{K}_i$ be probability kernels from $(\Omega_i, \mathfrak{A}_i)$ to $(\Upsilon_i, \mathfrak{B}_i)$ for $i = 1, 2$. Then the function $\mathcal{K} : (\Omega_1 \times \Omega_2) \times \sigma(\mathfrak{B}_1 \times \mathfrak{B}_2) \to [0, 1]$ defined as the product measure on $\sigma(\mathfrak{B}_1 \times \mathfrak{B}_2)$ for fixed first argument $(\omega_1, \omega_2) \in \Omega_1 \times \Omega_2$ is a probability kernel from $\big(\Omega_1 \times \Omega_2, \sigma(\mathfrak{A}_1 \times \mathfrak{A}_2)\big)$ to $\big(\Upsilon_1 \times \Upsilon_2, \sigma(\mathfrak{B}_1 \times \mathfrak{B}_2)\big)$.*

Proof. $\mathcal{K}\big((\omega_1, \omega_2), \cdot\,\big)$ is a probability measure for all $(\omega_1, \omega_2) \in \Omega_1 \times \Omega_2$ per definition. It remains to show that $\mathcal{K}(\,\cdot\,, B)$ is a $\sigma(\mathfrak{A}_1 \times \mathfrak{A}_2)$-measurable function for all $B \in \sigma(\mathfrak{B}_1 \times \mathfrak{B}_2)$. So far, $\mathcal{K}(\,\cdot\,, B_1 \times B_2)$ is at least measurable for all $B_1 \times B_2 \in \mathfrak{B}_1 \times \mathfrak{B}_2$ because the product of measurable functions $(\mathcal{K}(\,\cdot\,, B_1)\mathcal{K}(\,\cdot\,, B_2))$ is measurable. A monotone class argument (compare [20, pp. 403-404] and [6, p. 23]) delegates measurability for sets in $\sigma(\mathfrak{B}_1 \times \mathfrak{B}_2)$ to measurability for sets in $\mathfrak{B}_1 \times \mathfrak{B}_2$: The sets $B \in \sigma(\mathfrak{B}_1 \times \mathfrak{B}_2)$ for which $\mathcal{K}(\,\cdot\,, B)$ is measurable is a monotone class and the sets $B_1 \times B_2$ are contained in this class. Hence $\sigma(\mathfrak{B}_1 \times \mathfrak{B}_2)$ is a subset of this class. □

5.1.2 Stochastic Processes and Random Fields

A *stochastic process* describes a collection of random variables and stochastic dependencies among them. Major interests of this theory and also important points of attack include the discussion of joint distributions, local characterisations by conditional probabilities, limits, prediction, nonlocal influence etc.

For the investigation of stochastic processes in the context of cellular automata it seemingly suffices to observe *discrete-parameter stochastic processes* such that the parameters describe the (discrete) cells and the (discrete) iterations. But on the one hand because the theory is often discussed either in a discrete-parameter and discrete-state or continuous-parameter and continuous-state fashion, and secondly because a cellular automaton may be used to discretise a continuous process/system, the following sections also deal with *continuous-parameter processes*.

However the nomenclature in this area is occasionally rather confusing – maybe due to the diverse historical background. Basically a stochastic process is a collection of random variables.

Definition 5.1.17 (Random or Stochastic Process [20, 28, 29])**.** Let $(T, <)$ be a totally ordered set. A *stochastic process* $X := X_T := (X_t)_{t \in T}$ is a collection of random variables $X_t : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\Upsilon, \mathfrak{B})$ with indexing T.

If T is a totally ordered countable set (usually $\mathbb{N}$ or $\mathbb{Z}$), X is called *discrete-parameter stochastic process* or sometimes also simply *chain* (compare Markov chains, Section 5.1.3). If $T = \mathbb{R}_+$ we talk of a *continuous-parameter stochastic process*.

Random fields are understood as a generalisation of random processes with respect to the indexing respectively parameter. The following quotations and Definition 5.1.18 give a very good impression of the purpose of introducing *random fields* in the context of an investigation on cellular automata.

> ... for us a *random field* is simply a stochastic process ... defined over a parameter space of dimensionality at least 1. Although we shall be

> rather loose about exchanging the terms 'field' and 'process', in general we shall use 'field' when the geometric structure of the parameter space is important to us ... [2, p. 7]

Definition 5.1.18 (Random Field [2])**.** Given $(\Omega, \mathfrak{A}, \mathcal{P})$, $(\Upsilon, \mathfrak{B})$ and a topological space $(T, \mathfrak{O})$, a measurable mapping $X : \Omega \to \Upsilon^T$ is called *random field*.

> ... T ... probably stands for 'time'. While the whole point of this book is to get away form the totally ordered structure of $\mathbb{R}$, the notation is too deeply entombed in the collective psyche of probabilists to change it now. Later on, however, when we move to manifolds as parameter spaces ... [2, p. 7]

Compare Definition 5.1.18 also with the concept of random functions (Definition 5.1.19). The exact approach for defining random fields and also stochastic processes is a matter of taste.

Definition 5.1.19 (Random Function [28])**.** Let X_T be a stochastic process from $(\Omega, \mathfrak{A}, \mathcal{P})$ to $(\Upsilon, \mathfrak{B})$. For fixed ω the mapping $t \mapsto X_t(\omega)$ is called *random function*.

Random function corresponds to trajectory through the phase space.

5.1.3 Markov Processes

This section shall provide a short introduction to the formalisms of Markov processes. Accordingly $(T, <)$ is a totally ordered set. Some of the following definitions can either be found as slight modifications or one-to-one copies – except for notation – of definitions in [4, 20, 28], which provide a detailed discussion of stochastic processes.

Definition 5.1.20 (Filtration [20, 28, 34])**.** Given a σ-algebra $\mathfrak{A}$, a *filtration* is a collection of sub-σ-algebras $\mathfrak{A}_T := (\mathfrak{A}_t)_{t \in T}$ satisfying $\mathfrak{A}_s \subseteq \mathfrak{A}_t$ if $s \leq t$.

Definition 5.1.21 (Adapted Stochastic Process [20, 28, 34])**.** A stochastic process X_T from $(\Omega, \mathfrak{A}, \mathcal{P})$ to $(\Upsilon, \mathfrak{B})$ is called *adapted* to a filtration $\mathfrak{A}_T \subseteq \mathfrak{A}$ if X_t is $\mathfrak{A}_t$-measurable for all $t \in T$. A stochastic process is always adapted to its *natural filtration* $\big(\sigma(X_s : s \leq t)\big)_{t \in T}$.

Definition 5.1.22 (Markov Process [4, 20, 28])**.** A stochastic process $X_T : (\Omega, \mathfrak{A}) \to (\Upsilon, \mathfrak{B})$ is called *Markov process*, if X_T is adapted to its natural filtration $\mathfrak{A}_T \subseteq \mathfrak{A}$ and the *Markov property*, which for all $s > t$ is equally either

(i) $X_s \perp\!\!\!\perp \mathfrak{A}_t \mid X_t$,

(ii) $\mathcal{P}(X_s \in B \mid \mathfrak{A}_t) = \mathcal{P}(X_s \in B \mid X_t)$ a.s. for all $B \in \mathfrak{B}$ or

(iii) $\mathbb{E}[f(X_s)|\mathfrak{A}_t] = \mathbb{E}[f(X_s)|X_t]$ a.s. for all bounded measurable $f : \Upsilon \to \mathbb{R}$,

is satisfied.

When $T = \mathbb{N}$ or $T = \mathbb{Z}$ we talk of a *Markov chain*. In literature Markov chains are often assumed to have also a discrete state-space [4, 20]. A good approach for constructing such a *memoryless* (compare Section 1.3.1) stochastic chain is by specifying transition probabilities in the form of *Markov kernels*.

Since we observe random processes from $(\Omega, \mathfrak{A}, \mathcal{P})$ to $(\Upsilon, \mathfrak{B})$ we will use Markov kernels $\mathcal{K} : \Upsilon \times \mathfrak{B} \to [0, 1]$ in a way such that

$$\mathcal{K}_{s,t}(\upsilon, B) = \mathcal{P}(X_t \in B \mid X_s = \upsilon) \tag{5.1.18}$$

respectively

$$\mathcal{K}_{s,t}(X_s, B) = \mathcal{P}(X_t \in B \mid \mathcal{A}_s) \quad \text{a.s.} \tag{5.1.19}$$

5.1.3.1 Transition Functions

This section introduces the concept of *transition functions* in order to facilitate the construction of a connection between Markov processes and collections of Markov kernels.

For the following investigations it is necessary that $(\Upsilon, \mathfrak{B})$ is either a Polish space[4] with the σ-algebra of Borel sets [4] or (rather similar) a locally compact separable[5] metric space with the Borel σ-algebra [20].

> Neben den lokal-kompakten Hausdorff-Räumen sind die polnischen Räume von besonerer Wichtigkeit. Ein wesentlicher Grund dafür ist, daß wichtigen Konvergenzsätzen für stochastische Prozesse Maße auf polnischen (aber nicht lokal-kompakten) Räumen zugrundeligen ... [6, p. 320]
>
> Besides locally compact Hausdorff-spaces, Polish spaces are of particular importance. A significant reason is that important convergence theorems for stochastic processes require Polish (but not locally compact) spaces ... [6, p. 320]

Definition 5.1.23 (Transition Function [4, 20], compare also [28, 29])**.** Given an index set T, if for every $(s, t) \in T \times T$ with $s \leq t$ a probability kernel $\mathcal{K}_{s,t}$ from $(\Upsilon, \mathfrak{B})$ to $(\Upsilon, \mathfrak{B})$ is defined such that

(i) $\mathcal{K}_{t,t}(\upsilon, \cdot) = \delta_\upsilon$ for all $t \in T$ and

(ii) $\mathcal{K}_{s,u}\mathcal{K}_{u,t} = \mathcal{K}_{s,t}$ for all $s \leq u \leq t \in T$ (*Chapman-Kolmogorov equation*, product defined in Definition 5.1.15),

we call the collection of kernels *transition function* and also write $\mathcal{K}(s, \upsilon, t, B) : T \times \Upsilon \times T \times \mathfrak{B} \to [0, 1]$.

[4] A topological space X is called *Polish* if there exists a generating metric d, for which (X, d) is separable and complete [6, p. 320].

[5] "A topological space is called *separable* if there exists a countable dense subset." [6, p. 242], see also p. 90.

Accordingly a transition function is a collection of Markov kernels satisfying a certain semigroup property.

Proposition 5.1.24 (Transition Function of a Markov Process [4, 28])**.** *Given a Markov process* $X_T : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\Upsilon, \mathfrak{B})$*, the function*

$$\mathcal{K} : T \times \Upsilon \times T \times \mathfrak{B} \to [0,1] : (s, \upsilon, t, B) \mapsto \mathcal{P}(X_t \in B \mid X_s = \upsilon) \tag{5.1.20}$$

is a transition function i.e.

(i) $\mathcal{K}(s, \cdot, t, \cdot) : \Upsilon \times \mathfrak{B} \to [0,1]$ *is a Markov kernel (Definition 5.1.12) for fixed* s, t *with* $s \leq t$*,*

(ii) $\mathcal{K}(s, \upsilon, s, B) = \mathbb{I}_B(\upsilon)$ *and*

(iii) $\mathcal{K}$ *satisfies the* Chapman-Kolmogorov equation

$$\mathcal{K}(s, \upsilon, t, B) = \int_\Upsilon \mathcal{K}(s, \upsilon, r, dx)\, \mathcal{K}(r, x, t, B) \tag{5.1.21}$$

for $s \leq r \leq t$ *and all* $(\upsilon, B) \in \Upsilon \times \mathfrak{B}$*.*

Proof. $\mathcal{K}(s, \upsilon, t, B)$ is a probability kernel for every $s < t$ because $\mathcal{P}(X_t \in B | X_s = \upsilon)$ is a regular conditional probability. Requirement (ii) respectively Definition 5.1.23 (i) is clear. The proof that the Chapman-Kolmogorov equation – Definition 5.1.23 (ii) respectively (iii) in this proposition – is satisfied is more complicated. Complete proofs can be found in [4, 28]. □

Definition 5.1.25 (Initial Measure [4, 20])**.** If X_T is a Markov process $(\Omega, \mathfrak{A}, \mathcal{P}) \to (\Upsilon, \mathfrak{B})$ and $t_0 := \min T$, then the probability measure P_0 induced by X_0 on $(\Upsilon, \mathfrak{B})$ is called *initial measure*.

Theorem 5.1.26 (Markov Process Induced by Transition Function [4, 28])**.** *For a given probability measure* P_0 *on* $(\Upsilon, \mathfrak{B})$ *and a transition function* $\mathcal{K} : T \times \Upsilon \times T \times \mathfrak{B} \to [0,1]$ *there exists a unique Markov process.*

Proof. This theorem is very prominent in literature on Markov processes [20, 28]. □

The transition probabilities of the resulting Markov process satisfy [4, 20]

$$\mathcal{P}(X_t \in B \mid X_s = \upsilon) = \mathcal{K}(s, \upsilon, t, B). \tag{5.1.22}$$

The results on Markov processes and their transition functions can be summarised as follows.

Corollary 5.1.27 (Identification of Markov Processes and Transition Functions)**.** *A Markov process is always uniquely associated with a transition function.*

Theorem 5.1.28 (Chapman-Kolmogorov Equation [4], compare [20])**.** *A collection of random variables X_T is a Markov process with initial measure P_0 if and only if for every integer k and any family $f_0, \dots, f_k$ of non-negative measurable real-valued bounded functions and all $0 = t_0 < \cdots < t_k$,*

$$\begin{aligned}\mathbb{E}\Big[\prod_{i=0}^{k} f_i(X_{t_i})\Big] &= \int_\Upsilon P_0(d\upsilon_0) f_0(\upsilon_0) \int_\Upsilon \mathcal{K}(0,\upsilon_0,t_1,d\upsilon_1) f_1(\upsilon_1) \cdots \\ &\quad \cdots \int_\Upsilon \mathcal{K}(t_{k-1},\upsilon_{k-1},t_k,d\upsilon_k) f_k(\upsilon_k) \qquad (5.1.23)\\ &= \int_\Upsilon P_0(d\upsilon_0) f_0(\upsilon_0) \prod_{i=1}^{k} \int_\Upsilon \mathcal{K}(t_{i-1},\upsilon_{i-1},t_i,d\upsilon_i) f_i(\upsilon_i) \qquad (5.1.24)\\ &= \mathbb{E}[f_0(X_0)] \prod_{i=1}^{k} \mathbb{E}[f_{t_i}(X_{t_i}) | X_{t_{i-1}} = \upsilon_{i-1}] \qquad (5.1.25)\end{aligned}$$

where $\mathcal{K}(s,\upsilon,t,B) := \mathcal{P}(X_t \in B | X_s = \upsilon)$.

Proof. This theorem including proofs can be found in most works on Markov processes [4, 20]. □

Note that $\mathbb{E}[f_i(X_{t_i})|X_{t_{i-1}} = \cdot\,] = \mathbb{E}[f_i(X_{t_i})|X_{t_{i-1}}](\cdot)$ is a measurable random variable $(\Upsilon, \mathfrak{B}, P_0) \to ([0,1], \mathfrak{B}_{[0,1]})$ (compare Equation 5.1.5).

5.1.3.2 Markov Semigroups

This section formalises a commonly used functional analytic approach to transition functions for continuous-parameter Markov processes, so-called *Markov semigroups*. We use the notation $L^\infty(\Upsilon)$ for all bounded measurable functions $\Upsilon \to \mathbb{R}$ [20]. Then $C_0(\Upsilon) \subset L^\infty(\Upsilon)$.

Definition 5.1.29 (Transition Operator [4, 20])**.** Given a Markov process X_T the *transition operators* $(\mathcal{E}_{s,t})_{s,t\in T}$ are defined as bounded linear operators $L^\infty(\Upsilon) \to L^\infty(\Upsilon)$ such that for any bounded measurable function $f : \Upsilon \to \mathbb{R}$ and $\upsilon \in \Upsilon$

$$(\mathcal{E}_{s,t} f)(\upsilon) = \mathbb{E}[f(X_t)|X_s = \upsilon] \qquad s \le t. \qquad (5.1.26)$$

Sometimes [20, 28, 34] the following notation is used for Equation 5.1.26

$$(\mathcal{E}_{s,t} f)(X_s) = \mathbb{E}[f(X_t)|\mathfrak{A}_s] \quad \text{a.s.} \qquad (5.1.27)$$

This equation may be interpreted in the following context

$$\begin{aligned}\mathcal{E}_{s,t} f(\upsilon) &= \mathbb{E}[\mathcal{E}_{s,t} f(X_s)|X_s = \upsilon] \qquad (5.1.28)\\ &= \mathbb{E}\big[\mathbb{E}[f(X_t)|\mathfrak{A}_s]\big|X_s = \upsilon\big] \qquad (5.1.29)\\ &= \mathbb{E}[f(X_t)|X_s = \upsilon]. \qquad (5.1.30)\end{aligned}$$

Proposition 5.1.30 (Markov Semigroup [4, 20])**.** *Given a Markov process respectively its transition function, the collection of transition operators $(\mathcal{E}_{s,t})_{s\leq t}$ is a (contraction) semigroup of linear operators on $L^\infty(\Upsilon)$, which means ($r \leq s \leq t$)*

(i) $\mathcal{E}_{s,t}$ *is a linear operator on* $L^\infty(\Upsilon)$*,*

(ii) $\mathcal{E}_{t,t}$ *is the identity operator,*

(iii) $\|\mathcal{E}_{s,t}\| \leq 1$ *(*contraction*),*

(iv) $\mathcal{E}_{r,t} = \mathcal{E}_{r,s}\mathcal{E}_{s,t}$ *(*semigroup property*),*

(v) $f \geq 0$ *implies* $\mathcal{E}_{s,t}f \geq 0$ *(*non-negativity preserving*).*

The collection $(\mathcal{E}_{s,t})_{s\leq t}$ is called Markov semigroup*. Properties (ii-iv) are the classical properties of a contraction semigroup (Definition 4.2.2).*

Proof. A formal proof can be found in literature [4, 20, 29]. □

If $\mathcal{K}(s, \upsilon, t, B) := \mathcal{P}(X_t \in B | X_s = \upsilon)$ is the transition function of the Markov process X_T, we can write

$$(\mathcal{E}_{s,t}f)(\upsilon) = \mathbb{E}[f(X_t)|X_s = \upsilon] = \int_\Upsilon f(\tilde{\upsilon})\mathcal{K}(s, \upsilon, t, d\tilde{\upsilon}). \tag{5.1.31}$$

The transition operators can be identified with this transition function by [4, 20]

$$\mathcal{K}(s, \upsilon, t, B) = (\mathcal{E}_{s,t}\mathbb{I}_B)(\upsilon) \qquad \upsilon \in \Upsilon, B \in \mathfrak{B}. \tag{5.1.32}$$

Sometimes also the transition function is regarded as a semigroup (of Markov kernels) and then called *transition semigroup* [4, 28].

Theorem 5.1.31 (Markov Process Induced by Semigroup [20, 29])**.** *Given a Markov semigroup (properties (i-v) in Proposition 5.1.30) and an initial measure it is possible to construct a Markov process.*

Proof. See [20, 29] for a formal proof. □

The following corollary recapitulates the results from this section (compare Corollary 5.1.27.

Corollary 5.1.32 (Identification of Continuous-Parameter Markov Processes and Markov Semigroups)**.** *A continuous-parameter Markov process is uniquely associated with a Markov semigroup.*

5.1.3.3 Further Characterisations of Markov Processes

Definition 5.1.33 (Homogeneous Markov Process, compare [20, 28])**.** A Markov process is called *homogeneous* if the transition function is invariant under translations $(s,t) \to (s+u, t+u)$. In this case also the corresponding semigroup(s) is (are) called *homogeneous* and it is sufficient to define $\mathcal{E}_T := (\mathcal{E}_t)_{t\in T} := (\mathcal{E}_{0,t})_{t\in T}$ respectively $\mathcal{K}_T := (\mathcal{K}_t(\upsilon, B))_{t\in T} := (\mathcal{K}(0, \upsilon, t, B))_{t\in T}$.

Equation 5.1.26 can then be written as

$$(\mathcal{E}_{dt} f)(\upsilon) = \mathbb{E}[f(X_{s+dt})|X_s = \upsilon] \qquad \upsilon \in \Upsilon. \tag{5.1.33}$$

For continuous-parameter Markov processes and especially for $T = \mathbb{R}_+$ a lot of additional concepts can be introduced. Among them the *Feller property*.

Definition 5.1.34 (Feller Semigroup, Feller Process [20, 29])**.** A homogeneous Markov semigroup $\mathcal{E}_T$ is a *Feller semigroup* if

(i) $\mathcal{E}_t$ is a mapping $C_0(\Upsilon) \to C_0(\Upsilon)$ for all $t \in T$ and

(ii) $\lim_{t\to t_0+} \|\mathcal{E}_t f - f\|_\infty = 0$ for all $f \in C_0(\Upsilon)$ (*Feller property*).

The associated Markov process is called *Feller process*.

If Υ is compact, *resolvents* and *generators* of Feller processes can be introduced [20, 29].

5.1.4 Multiparameter Markov Processes

Multiparameter stochastic processes can be seen as an extension of one-parameter processes. Per definition, a multiparameter process is a random field (see Definition 5.1.18) – either with discrete or continuous parameter. However, in literature especially *Markovian* multiparameter processes are often associated with a parameter set in $\mathbb{R}^d$, which is also the case for [20], which served as a basis for this section.

> We can informally interpret ... a Markov process as a (one-parameter) process whose 'future' values depend on the past only through its current value. While this is perfectly intuitively clear (due to the well-ordering of the 'time axis'), it is far less clear what a multiparameter Markov process should be. ... We will also see how this multiparameter theory can be used to study intersections of ordinary one-parameter processes. [20, p. 391]

Let Υ be a compact separable metric space[6] equipped with the Borel σ-algebra and let $(T, \preccurlyeq)$ be a half-ordered multidimensional set, usually $T \subseteq \mathbb{R}^d_+$ together with $s \preccurlyeq t$ if and only if $s_i \leq t_i$ for all $i = 1, \ldots, d$. Let further $s \curlywedge t$ denote the element wise minimum.

[6] or the one-point compactification of a locally compact separable metric space [20]

Definition 5.1.35 (d-parameter Filtration [19, 20])**.** A d-parameter collection of σ-algebras $(\mathfrak{A}_t)_{t\in T}$ is called *d-parameter filtration* if $s \preccurlyeq t$ implies $\mathfrak{A}_s \subseteq \mathfrak{A}_t$.

Definition 5.1.36 (Commutation [19, 20])**.** A d-parameter filtration $\mathfrak{A}_T$ is called *commuting*, if for all $s, t \in T$

$$\mathfrak{A}_s \perp\!\!\!\perp \mathfrak{A}_t \mid \mathfrak{A}_{s \curlywedge t} \tag{5.1.34}$$

or equivalently if for all bounded $\mathfrak{A}_t$-measurable random variables X,

$$\mathbb{E}[X|\mathfrak{A}_s] = \mathbb{E}[X|\mathfrak{A}_{s\curlywedge t}] \quad \text{a.s.} \tag{5.1.35}$$

Definition 5.1.37 (Multiparameter Markov Process [20, p. 392])**.** A stochastic process X_T is called *multiparameter Markov process* if there exists a d-parameter filtration $\mathfrak{A}_T$ and a family of operators $\mathcal{E}_T$ such that for all $\upsilon \in \Upsilon$, there exists a probability measure P_υ on $(\Upsilon, \mathfrak{B})$ and

(i) X_T is adapted to $\mathfrak{A}_T$,

(ii) $t \mapsto X_t$ is right-continuous[7] P_υ-a.s. for all $\upsilon \in \Upsilon$,

(iii) $\mathfrak{A}_T$ is a commutating filtration and $\mathfrak{A}_t$ is complete[8] with respect to P_υ for all $\upsilon \in \Upsilon$ and all $t \in T$,

(iv) for all $s, t \in T$, $f \in C_0(\Upsilon)$ and $\upsilon \in \Upsilon$,

$$\mathcal{E}_t f(X_s) = \mathbb{E}_{P_\upsilon}[f(X_{t+s})|\mathfrak{A}_s] \qquad P_\upsilon\text{-a.s.} \tag{5.1.36}$$

and

(v) $P_\upsilon(X_0 = \upsilon) = 1$ for all $\upsilon \in \Upsilon$.

A multiparameter process is sometimes also called *set-indexed process*. Also for multiparameter Markov processes, $\mathcal{E}_T$ is a (multiparameter) semigroup[9] [20].

Furthermore Equation 5.1.36 is equivalent to

$$\begin{aligned}
\mathcal{E}_t f(\upsilon) &= \mathbb{E}[\mathcal{E}_t f(X_s)|X_s = \upsilon] && (5.1.37)\\
&= \mathbb{E}\big[\mathbb{E}[f(X_{t+s})|\mathfrak{A}_s]\big|X_s = \upsilon\big] && (5.1.38)\\
&= \mathbb{E}[f(X_{t+s})|X_s = \upsilon]. && (5.1.39)
\end{aligned}$$

Define the parameter helper σ as [20]

$$\begin{aligned}
\sigma_1(r) &= (r, 0, 0, \ldots, 0) && (5.1.40)\\
\sigma_2(r) &= (0, r, 0, \ldots, 0) && (5.1.41)\\
&\vdots && (5.1.42)\\
\sigma_d(r) &= (0, 0, \ldots, 0, r). && (5.1.43)
\end{aligned}$$

[7] "A function $f : \mathbb{R}^d_+ \to \mathbb{R}$ is *right-continuous* (with respect to the partial order $\preccurlyeq$) if for all $t \in \mathbb{R}^d_+$, $\lim_{s\succcurlyeq t, s\to t} f(s) = f(t)$." [20, p. 236]

[8] i.e. contains all zero-measure/null sets [6, p. 63]

[9] For an exact formal definition see for example [17, p. 28].

Definition 5.1.38 (Marginal Semigroups [20, p. 396])**.** The *marginal semigroups* $\mathcal{E}^j$, $j = 1, \ldots, d$ of a d-parameter Markov process $(\mathcal{E}_t)_{t\in T}$ are defined such that

$$\mathcal{E}^j_r f(\upsilon) = \mathcal{E}_{\sigma_j(r)} f(\upsilon) = \mathcal{E}_{(\ldots,r,\ldots)} f(\upsilon) \tag{5.1.44}$$

for all $f \in L^\infty(\Upsilon)$ and $\upsilon \in \Upsilon$.

As a direct consequence,

$$\mathcal{E}_t = \mathcal{E}^1_{t_1} \cdots \mathcal{E}^d_{t_d} \tag{5.1.45}$$

if $t = (t_1, \ldots, t_d)$.

5.1.5 Markov Random Fields

If the topological space $(T, \mathfrak{O})$ of a random field is – or, more correctly and simpler, is replaced by – a graph $G = (M, E, p)$ and the random field satisfies a certain local stochastic feature, it is called *Markov random field* [21]. Surprisingly the notation *Markov random field* is only used in connection with graphs as parameter set, which are discrete parameter sets.

We assume that the neighbourhoods are not ordered respectively that the graph is not weighted and simple. It is furthermore necessary that the underlying graph is undirected (respectively symmetric) since stochastic dependence is a symmetric relation.

For a collection $(x_n)_{n\in M}$ the notation $x_N := (x_n)_{n\in N}$ where $N \subseteq M$ shall describe the corresponding subcollection.

Definition 5.1.39 (Markov Random Field, compare [15, 21])**.** Given a simple undirected graph $G = (M, E)$ with/respectively a symmetric neighbourhood mapping $\mathcal{N} : M \to \mathfrak{P}(M)$ and a collection of random variables $X_M := (X_m)_{m\in M}$ i.e. a random field on M, we call X a *Markov random field* if the *local Markov property*

$$X_m \perp\!\!\!\perp X_{M\setminus(\mathcal{N}(m)\cup\{m\})} \mid X_{\mathcal{N}(m)\setminus\{m\}} \qquad m \in M \tag{5.1.46}$$

is satisfied.

This definition is valid if $m \in \mathcal{N}(m)$ and if $m \notin \mathcal{N}(m)$. In case of a reflexive neighbourhood structure i.e. if $m \in \mathcal{N}(m)$ the local Markov property simplifies to

$$X_m \perp\!\!\!\perp X_{M\setminus\mathcal{N}(m)} \mid X_{\mathcal{N}(m)\setminus\{m\}} \qquad m \in M. \tag{5.1.47}$$

We also define two further Markov properties for Markov random fields.

Definition 5.1.40 (Pairwise and Global Markov Property [15, 48])**.** The *pairwise Markov property* for $m \neq n$ with $m \notin \mathcal{N}(n)$ – and accordingly $n \notin \mathcal{N}(m)$ – is defined as

$$X_m \perp\!\!\!\perp X_n \mid X_{M\setminus\{m,n\}}. \tag{5.1.48}$$

The *global Markov property* for $A, B, C \subseteq M$ such that every path between a vertex in A and a vertex in B contains a vertex in C is defined as

$$X_A \perp\!\!\!\perp X_B \mid X_C. \tag{5.1.49}$$

It is easy to show that the global Markov property implies the local Markov property and the local Markov property implies the pairwise Markov property [48].

While the preceding definitions are also valid for continuous state spaces Υ, the following statements as originally proposed in [21] and [16] are not. The reason for this may be the historical connection to the Ising model [21].

Definition 5.1.41 (Local Characteristic [21])**.** The *local characteristic* or also *local conditional* of a Markov random field is defined as a function $\mathcal{L}_m : \Upsilon^M \to [0,1]$ for every $m \in M$,

$$\begin{aligned}\mathcal{L}_m(x_M) &:= \mathcal{P}\big(X_m = x_m \mid X_{\mathcal{N}(m)\setminus\{m\}} = x_{\mathcal{N}(m)\setminus\{m\}}\big)\\ &= \mathcal{P}\big(X_m = x_m \mid X_{M\setminus\{m\}} = x_{M\setminus\{m\}}\big).\end{aligned} \tag{5.1.50}$$

The collection $\mathcal{E} := (\mathcal{L}_m)_{m\in M}$ is called *local specification.*

The local specification $\mathcal{E}$ may be interpreted as a mapping $\Upsilon^M \to [0,1]^M$.

When constructing a Markov random field these conditional probabilities should be consistent in the sense that they yield a joint probability distribution on Υ^M, which is a mapping $\Upsilon^M \to [0,1]$. The Hammersley-Clifford theorem provides a condition for probability distributions to describe a Markov random field. This theorem is the main result of basic Markov random field theory.

Definition 5.1.42 (Gibbs Distribution, compare [15, 16])**.** For a given graph denote the set of all cliques[10] as $\mathfrak{C}$. A probability mass function $P : \Upsilon^M \to [0,1]$ satisfying

$$P(x_M) = \prod_{C\in\mathfrak{C}} f_C(x_M) \qquad x_M \in \Upsilon^M \tag{5.1.51}$$

where f_C only depends on the components of x_M with index in C is called *Gibbs distribution.*

In the original form an exponential notation of Equation 5.1.51 is used.

Theorem 5.1.43 (Hammersley-Clifford [16], compare [15])**.** *Assume X_M as a random field on an undirected finite graph has a positive joint probability distribution on Υ^M. X_M is a Markov random field if and only if the distribution is a Gibbs distribution.*

Proof. This is a prominent theorem. For one of the earliest proofs see [16]. □

The Gibbs distribution defines the Markov random field as a whole, $P(x_M) = \mathcal{P}(X_M = x_M)$.

[10] The nodes within a clique are connected to each other. In other words a clique is a complete subgraph. See also Definition 3.2.19 and Definition 3.2.20.

Application and History of Markov Random Fields. Especially in the context of this work, [21] provides an interesting application of Markov random fields on cell growth models by Eden [37] (not related to the Garden of Eden cellular automaton), Williams and Bjerknes [49] (simulation of tumour growth using *biased voter model*) and Richardson [41]. These works are the basis of many interesting publications and must also be seen in the context of pattern formation [40] and image processing.

5.1.6 Bayesian Networks

> Bayesian networks are used to represent joint probability distributions over sets of random variables. A Bayesian network is made up of two components: a directed acyclic graph, and a set of conditional probability tables. Each node in the graph represents a random variable and for each node there is a probability table specifying the conditional distribution of the variable given (each possible combination of) the values of its parents in the graph. [11]

We will use the notation of neighbourhood mappings to indicate the "neighbouring" vertices of a vertex (Section 3.2). Note that acyclic implies nonreflexive ($m \notin \mathcal{N}(m)$) and that $\mathcal{N}(m)$ is the set of vertices that feature a directed connection towards m.

Definition 5.1.44 (Bayesian Network, compare [11, 15, 24])**.** A directed acyclic simple graph $G = (M, E)$ together with a collection of discrete random variables $X_M = (X_m)_{m\in M}$ with states in Υ and conditional probabilities for all $m \in M$

$$\mathcal{P}(X_m = x_m \mid X_{n_1} = x_1, \ldots, X_{n_{k_m}} = x_{k_m}) \qquad \{n_1, \ldots, n_{k_m}\} = \mathcal{N}(m) \tag{5.1.52}$$

where $x_* \in \Upsilon$, is called *Bayesian network*.

In order to allow this short notation, – as in Section 5.1.5 – $x_{\mathcal{N}(m)}$ is understood as the subcollection of $x_M = (x_m)_{m\in M}$ with indices in $\mathcal{N}(m)$.

From the acyclic structure of the graph it is possible to derive the *Markov Assumption* which is a *local Markov property* for Bayesian networks:

Proposition 5.1.45 (Markov Assumption [11, 24])**.** *Each variable is independent of its nondescendants, given its parents.*

$$X_m \perp\!\!\!\perp X_{M\setminus\mathcal{N}^{-1}(m)} \mid X_{\mathcal{N}(m)}, \tag{5.1.53}$$

which also can be written as

$$\mathcal{P}\big(X_m = x_m \mid X_{M\setminus\mathcal{N}^{-1}(m)} = x_{M\setminus\mathcal{N}^{-1}(m)}\big) = \tag{5.1.54}$$

$$= \mathcal{P}\big(X_m = x_m \mid X_{\mathcal{N}(m)} = x_{\mathcal{N}(m)}\big). \tag{5.1.55}$$

Proposition 5.1.46 (Joint Distribution [11, 24])**.** *The joint probability distribution of a Bayesian network satisfies*

$$\mathcal{P}(X_M = x_M) = \prod_{m \in M} \mathcal{P}\big(X_m = x_m \mid X_{\mathcal{N}(m)} = x_{\mathcal{N}(m)}\big). \tag{5.1.56}$$

Proof. Can be found in literature. □

For continuous states the preceding notations can not be used. The next section will introduce a class of stochastic process with continuous states suitable for use with cellular automata. The following quote describes a method for "simulating" continuous distributions.

> Unlike the case of discrete variables, when the variable X_m and its parents ... are real valued, there is no representation that can represent all possible densities. A natural choice for multivariate continuous distributions is the use of Gaussian distributions. These can be represented in a Bayesian network by using linear Gaussian conditional densities. In this representation, the conditional density of X_m given its parents is given by
>
> $$\mathcal{P}(X_m = v | X_{\mathcal{N}(m)} = \vec{v}) \sim \text{Norm}(a_0 + \sum_i a_i, \sigma^2) \tag{5.1.57}$$
>
> That is, X_m is normally distributed around a mean that depends linearly on the values of its parents. The variance of this normal distribution is independent of the parents' values. If all the variables in a network have linear Gaussian conditional distributions, then the joint distribution is a multivariate Gaussian (Lauritzen and Wermuth, 1989). [24]

Corollary 5.1.47 (A Bayesian Network is a Markov Random Field)**.** *Every Bayesian network is a Markov random field.*

Proof. Since an acyclic graph only has one-vertex cliques, Proposition 5.1.46 delivers a Gibbs representation of the joint probability distribution. □

5.2 Formalism of Stochastic Cellular Automata

This section introduces a class of stochastic processes that is an extension of Bayesian networks (Section 5.1.6) and at the same time a special case of multiparameter Markov processes (Section 5.1.4) as well as Markov random fields (Section 5.1.5). Such a stochastic process will be referred to as *stochastic cellular automaton.*

5.2.1 Basic Concepts

Regard a discrete set of nodes M called *cells* of a graph G_2 with neighbourhood mapping $\mathcal{N}$ such that the neighbourhoods are equally sized with k neighbours each. The neighbourhoods may be ordered $\mathcal{N}(m) = (n_1, \ldots, n_k)$ or unordered $\mathcal{N}(m) = \langle n_1, \ldots, n_k \rangle$ (Definition 2.2.2).

This graph then is k-regular, simple as well as directed and represents the *second topological feature* of a stochastic cellular automaton. In case of ordered neighbourhoods there exists an additional mapping $w : M \times M \to \mathbb{N}$ (Theorem 3.2.4). This however does not hinder the following discussions.

A second (trivial) graph G_1 is given by a totally ordered discrete parameter set $(T, \leq)$ like for example $(\mathbb{N}, \leq)$ such that there exists a directed edge between two consecutive elements of T. The elements respectively nodes in T are referred to as *iterative steps*.

Definition 5.2.1 (Product Graph)**.** Given two graphs $G_1 = (M, \mathcal{N})$ and $G_2 = (T, \leq)$ as discussed before, define the directed acyclic simple k-regular graph $G = (T \times M, E)$ as the graph tensor product $G := G_1 \times G_2$ such that each node is an element from $\{(t, m) : t \in T, m \in M\}$ and

$$E = \bigcup_{t \in T} \bigcup_{m \in M} \bigcup_{n \in \mathcal{N}(m)} \big\{\big((t, n), (t+1, m)\big)\big\}. \tag{5.2.1}$$

We call this graph the *product graph* or simple the *graph* of a stochastic cellular automaton.

Every node (t, m) can be interpreted as a cell $m \in M$ at a specific iterative step $t \in T$. Figure 5.1 shows an illustration of such a product graph. Compare this graphical structure also with the concept of *nets*.

On every directed acyclic graph there exists a canonical partial order $\preccurlyeq$ with $a \preccurlyeq b$ if and only if there is a directed path from node a to node b [48]. For the (product-) graph G this partial order can be decomposed as

$$(s, n) \preccurlyeq (t, m) \quad \Longleftrightarrow \quad s \leq t \wedge n \in \mathcal{N}^{t-s}(m) \tag{5.2.2}$$

which also explicitly causes

$$(t, n) \not\preccurlyeq (t, m) \qquad \forall\, t \in T \quad m, n \in M \quad m \neq n. \tag{5.2.3}$$

Define for all $(s, n), (t, m) \in T \times M$, $(s, n) \curlywedge (t, m)$ as

$$\max\{r \in T : r \leq s, r \leq t, \exists l \in M : l \in \mathcal{N}^{s-r}(n) \cap \mathcal{N}^{t-r}(m)\}. \tag{5.2.4}$$

For a stochastic cellular automaton it is necessary to define an abstract probability space with certain filtrations.

Definition 5.2.2 (Probability Space and Filtrations)**.** Define the *probability space* respectively the *filtrations* of a stochastic cellular automaton as follows: Let $(\Omega, \mathfrak{A})$ be a Radon space (Definition 5.1.8), $\mathcal{P}$ a probability measure on $(\Omega, \mathfrak{A})$ and $(\mathfrak{A}_t)_{t\in T}$ a filtration such that $\mathfrak{A}_s \subseteq \mathfrak{A}_t$ if $s \leq t$. Let $(\mathfrak{A}_{t,m})_{(t,m)\in T\times M}$ be a second filtration with $\mathfrak{A}_{t,m} \subseteq \mathfrak{A}_t$ for all $t \in T$ and $m \in M$ and $\mathfrak{A}_{s,n} \subseteq \mathfrak{A}_{t,m}$ if $(s, n) \preccurlyeq (t, m)$.

See Figure 5.2 for an illustration of the inclusion relations among the various σ-algebras.

Each cell shall take states form the set of states $\mathbb{S}$ according to some probability distribution. Or in other words for every cell there shall exist a random variable from the common underlying probability space into the state space.

Definition 5.2.3 (Random States)**.** Given a product graph (as in Definition 5.2.1) as well as a probability space and filtrations (as in Definition 5.2.2), let $(\mathbb{S}, \mathfrak{B})$ be a separable Hausdorff measurable space.

Associate each node (t, m) of the product graph with a random variable $S_{t,m} : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\mathbb{S}, \mathfrak{B})$ and let the stochastic process $(S_{t,m})_{(t,m)\in T\times M}$ be adapted to the filtration $(\mathfrak{A}_{t,m})_{t\in T,m\in M}$ in the sense that $S_{t,m}$ is $\mathfrak{A}_{t,m}$-measurable for all $t \in T$ and $m \in M$.

The random variables $(S_{t,m})_{(t,m)\in T\times M}$ are then called *random states* and the multivariate random variables $\mathcal{S}_t := (S_{t,m})_{m\in M}$ are called *global random states*.

As a direct consequence of this definition, the stochastic process $(\mathcal{S}_t)_{t\in T}$ is adapted to the filtration $(\mathfrak{A}_t)_{t\in T}$.

Definition 5.2.4 (Markov Property for Stochastic Cellular Automata)**.** It is required that the *Markov property*

$$S_{t+1,m} \perp\!\!\!\perp \mathfrak{A}_t \mid S_{t,\mathcal{N}(m)} \tag{5.2.5}$$

is satisfied for all $t \in T$ and $m \in M$.

Since $(\Omega, \mathfrak{A})$ is a Radon space, there exist for all t, m regular conditional probabilities from $\left(\mathbb{S}^k, \sigma(\mathfrak{B}^k)\right)$ to $(\Omega, \mathfrak{A}_{t+1,m})$ with respect to $S_{t,\mathcal{N}(m)}$ (Theorem 5.1.9). These regular conditional probabilities $\mathcal{P}(\,\cdot\,|S_{t,\mathcal{N}(m)} = \,\cdot\,)$ will also be called *stochastic update rules*.

As an immediate consequence, $S_{t+1,m} \perp\!\!\!\perp \mathcal{S}_t \mid S_{t,\mathcal{N}(m)}$ as well as $S_{t+1,m} \perp\!\!\!\perp S_{t,M\setminus\mathcal{N}(m)} \mid S_{t,\mathcal{N}(m)}$ and

$$\mathcal{P}(S_{t+1,m} \in B|\mathcal{S}_t = s_M) = \mathcal{P}(S_{t+1,m} \in B|S_{t,\mathcal{N}(m)} = s_{\mathcal{N}(m)}) \tag{5.2.6}$$

for any $s_M \in \mathbb{S}^M$.

For cellular automata in general we declared that the update mechanism is equal for all cells and iterative steps. In the context of a stochastic (Markov) process this feature is called homogeneity.

Definition 5.2.5 (Homogeneity of Transition Probabilities)**.** Let

$$\mathcal{P}(S_{t+1,m} \in B|S_{t,\mathcal{N}(m)} = \vec{s}) = \mathcal{P}(S_{s+1,n} \in B|S_{s,\mathcal{N}(n)} = \vec{s}) \tag{5.2.7}$$

for all $t, s \in T$ and $m, n \in M$ where $B \in \mathfrak{B}$ and $\vec{s} \in \mathbb{S}^k$.

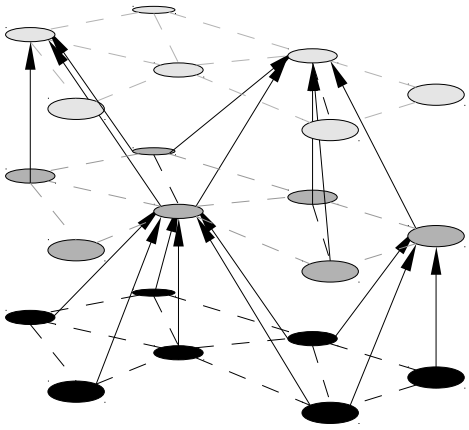

Figure 5.1: Horizontal layers represent a (directed) graph G_2 with nodes M and edges indicated by dashed lines (compare this graph with the graph representation of the second topological feature of a cellular automaton Section 3.2). The whole graph is the tensor product of a trivial graph G_1 with nodes T (sketched vertically) and G_2. Arrows represent a selection of directed edges of the product graph. The product graph can be regarded as a graph representation of a cellular automaton that depicts cells separately for every iterative step.

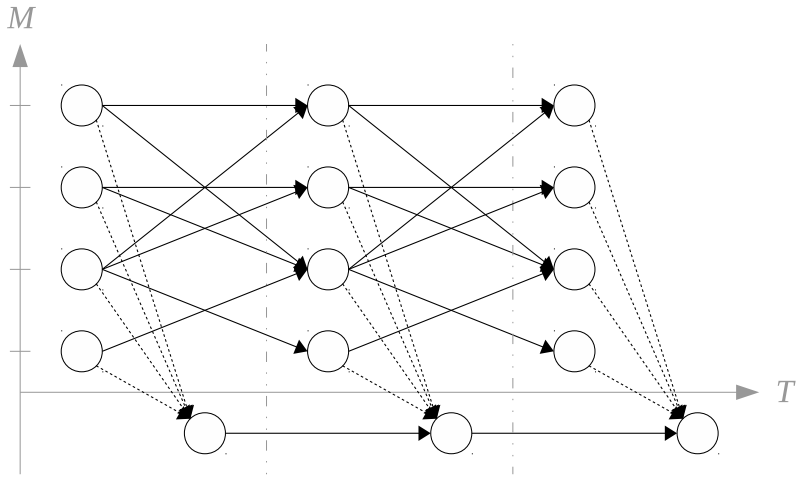

Figure 5.2: Filtrations for stochastic cellular automata. Circles indicate σ-algebras and arrows represent the inclusion relation. The horizontal axis concatenates iterative steps (T). The positive vertical axis represents elements of M such that circles above the horizontal axis represent σ-algebras $\mathfrak{A}_{t,m}$ and circles below the horizontal axis represent σ-algebras $\mathfrak{A}_t$.

5.2.2 Definition of Stochastic Cellular Automata

The definitions of the previous section can now be used to define stochastic cellular automata.

Definition 5.2.6 (Stochastic Cellular Automaton)**.** A *stochastic cellular automaton* consists of

(i) a set of cells M,

(ii) a k-regular simple directed graphical structure on the set of cells ($\mathcal{N}$) which together with a totally ordered set of iterative steps (T) forms a product graph,

(iii) a separable Hausdorff measurable state space $(\mathbb{S}, \mathfrak{B})$, a Radon probability space $(\Omega, \mathfrak{A}, \mathcal{P})$ with filtrations $(\mathfrak{A}_t)_{t\in T}$ and $(\mathfrak{A}_{t,m})_{(t,m)\in T\times M}$ and

(iv) a collection of random variables $(S_{t,m})_{(t,m)\in T\times M}$ from Ω to $\mathbb{S}$ which is adapted to $(\mathfrak{A}_{t,m})_{(t,m)\in T\times M}$ and satisfies the homogeneity property as well as the Markov property for stochastic cellular automata.

Note that in contrast to deterministic cellular automata, which are iterative systems, a stochastic cellular automaton is a stochastic process. For a given *initial global random state* $\mathcal{S}_0$, the push-forward measure $\mathcal{P} \circ \mathcal{S}_0^{-1}$ is called *initial distribution*.

In analogy to the theory of stochastic processes we also call the collection $(S_{t,m})_{(t,m)\in T\times M}$ a stochastic cellular automaton.

5.2.3 Immediate Results and Conclusions

This section elaborates some characteristics and mathematical properties of stochastic cellular automata.

5.2.3.1 Markov Kernels

For this and the following sections a stochastic cellular automaton shall be given, let all variables be defined as in Definition 5.2.6.

Definition 5.2.7 (Local Kernel)**.** We call a Markov kernel from $\left(\mathbb{S}^k, \sigma(\mathfrak{B}^k)\right)$ to $(\mathbb{S}, \mathfrak{B})$ written $\mathcal{K} : \mathbb{S}^k \times \mathfrak{B} \to [0, 1]$ which satisfies

$$\mathcal{K}(\vec{s}, B) = \mathcal{P}(S_{t+1,m} \in B | S_{t,\mathcal{N}(m)} = \vec{s}) \qquad \forall\, t \in T, m \in M \tag{5.2.8}$$

a *local kernel*.

The existence of a local kernel implies homogeneity of the transition probabilities and vice versa. From the local kernel further probability kernels can be constructed.

Definition 5.2.8 (Local Characteristic)**.** For every $m \in M$ we define the *local characteristic* as the Markov kernel $\mathcal{L}_m : \mathbb{S}^M \times \mathfrak{B} \to [0, 1]$ with

$$\mathcal{L}_m(s_M, B) := \mathcal{K}(\mathrm{proj}_{\mathcal{N}(m)}\, s_M, B) = \mathcal{K}(s_{\mathcal{N}(m)}, B). \tag{5.2.9}$$

$\mathcal{L}_m$ actually is a probability kernel because $\mathrm{proj}_{\mathcal{N}(m)}$ is a measurable function an therefore $\mathcal{L}_m$ is measurable in its first argument for all $B \in \mathfrak{B}$. That $\mathcal{L}_m$ is a probability measure in its second argument is clear.

Because $S_{t+1,m} \perp\!\!\!\perp S_{t,M\setminus\mathcal{N}(m)} \mid S_{t,\mathcal{N}(m)}$ (Definition 5.2.4), we can write

$$\mathcal{L}_m(s_M, B) = \mathcal{P}\big(S_{t+1,m} \in B \big| \mathcal{S}_t = s_M\big) = \tag{5.2.10}$$

$$= \mathcal{P}\big(S_{t+1,m} \in B \big| S_{t,\mathcal{N}(m)} = s_{\mathcal{N}(m)}\big) = \mathcal{K}(s_{\mathcal{N}(m)}, B) \tag{5.2.11}$$

Definition 5.2.9 (Global Kernel)**.** The *global kernel* $\mathcal{E} : \mathbb{S}^M \times \sigma(\mathfrak{B}^M) \to [0,1]$ of a stochastic cellular automaton is defined as the Cartesian ("semi"-) product of the local characterisations $\mathcal{L}_m$,

$$\mathcal{E}(s_M, B_M) := \Big(\prod_{m\in M} \mathcal{L}_m(s_M, \cdot\,)\Big)(B_M) \tag{5.2.12}$$

where $B_M \in \sigma(\mathfrak{B}^M)$ and $s_M \in \mathbb{S}^M$.

It is clear that $\mathcal{E}$ is a probability measure in its second argument. Also because of Proposition 5.1.16 the (full) Cartesian product of the probability kernels $\mathcal{L}_m$ is a probability kernel from $\big(\mathbb{S}^M, \sigma(\mathfrak{B}^M)\big)^M$ to $\big(\mathbb{S}^M, \sigma(\mathfrak{B}^M)\big)$, it remains to acknowledge that the projection $\big(\mathbb{S}^M\big)^M \to \mathbb{S}^M$ is a measurable mapping in order to see that $\mathcal{E}$ is a probability kernel.

For every $s_M \in \mathbb{S}^M$, the local characteristics $\mathcal{L}(s_M, \cdot\,)$ are obviously the marginal measures of the global kernel $\mathcal{E}(s_M, \cdot\,)$.

5.2.3.2 Chapman-Kolmogorov Equation

Theorem 5.2.10 (Chapman-Kolmogorov Equation)**.** *A stochastic cellular automaton satisfies the following Chapman-Kolmogorov equations.*

$$\mathcal{P}(S_{t+2,m} \in B | \mathcal{S}_t = s_M) \tag{5.2.13}$$

$$= \int_{\mathbb{S}^M} \mathcal{E}(s_M, dq)\ \mathcal{L}_m(q, B) = \int_{\mathbb{S}^M} \prod_{n\in M} \mathcal{L}_n(s_M, dq_n)\ \mathcal{L}_m(q, B) \tag{5.2.14}$$

$$\stackrel{?}{=} \int_{\mathbb{S}^{\mathcal{N}(m)}} \prod_{n\in\mathcal{N}(m)} \mathcal{K}(s_{\mathcal{N}(n)}, dp_n)\ \mathcal{K}(p, B) \tag{5.2.15}$$

$$= \mathcal{P}(S_{t+2,m} \in B | S_{t,\mathcal{N}^2(m)} = s_{\mathcal{N}^2(m)}) \tag{5.2.16}$$

Proof. We only prove ($\stackrel{?}{=}$). Replacing $\mathcal{L}$ with $\mathcal{K}$ in Equation 5.2.14 yields

$$\int_{\mathbb{S}^M} \prod_{n\in M} \mathcal{K}(s_{\mathcal{N}(n)}, dq_n)\, \mathcal{K}(q_{\mathcal{N}(m)}, B) \tag{5.2.17}$$

$$= \int_{\mathbb{S}^{\mathcal{N}(m)}\times\mathbb{S}^{M\setminus\mathcal{N}(m)}} \prod_{n\in\mathcal{N}(m)} \mathcal{K}(s_{\mathcal{N}(n)}, dq_n) \prod_{n\notin\mathcal{N}(m)} \mathcal{K}(s_{\mathcal{N}(n)}, dq_n)\, \mathcal{K}(q_{\mathcal{N}(m)}, B) \tag{5.2.18}$$

$$= \int_{\mathbb{S}^{\mathcal{N}(m)}} \prod_{n\in\mathcal{N}(m)} \mathcal{K}(s_{\mathcal{N}(n)}, dp_n)\, \mathcal{K}(p_{\mathcal{N}(m)}, B) \int_{\mathbb{S}^{M\setminus\mathcal{N}(m)}} \prod_{n\notin\mathcal{N}(m)} \mathcal{K}(s_{\mathcal{N}(n)}, dq_n). \tag{5.2.19}$$

The second integral in Equation 5.2.19 can be written as $\prod_{n\notin\mathcal{N}(m)} \mathcal{K}(s_{\mathcal{N}(n)}, \mathbb{S})$ and is equal to 1. □

Lemma 5.2.11. *For $n \notin \mathcal{N}(m)$, $m \notin \mathcal{N}(n)$ and $\mathcal{N}(m) \cap \mathcal{N}(m) \neq \emptyset$,*

$$S_{t,m} \perp\!\!\!\perp S_{t,n} \mid S_{t-1,\mathcal{N}(m)\cap\mathcal{N}(n)}. \tag{5.2.20}$$

Proof. Set $X := \mathbb{S}^{\mathcal{N}(m)\cup\mathcal{N}(n)\setminus\mathcal{N}(m)\cap\mathcal{N}(n)}$ and $N := \mathcal{N}(m) \cap \mathcal{N}(n)$. Regarding the equation for conditional independence proves the lemma.

$$\mathcal{P}(S_{t,m} \in A, S_{t,n} \in B | S_{t-1,N} = s_N) \tag{5.2.21}$$

$$= \int_X \mathcal{K}(p_{\mathcal{N}(m)\setminus\mathcal{N}(n)} \times s_N, A)\, \mathcal{K}(p_{\mathcal{N}(n)\setminus\mathcal{N}(m)} \times s_N, B)\, dp \tag{5.2.22}$$

$$= \int_{\mathbb{S}^{\mathcal{N}(m)\setminus\mathcal{N}(n)}} \mathcal{K}(p \times s_N, A)\, dp \int_{\mathbb{S}^{\mathcal{N}(n)\setminus\mathcal{N}(m)}} \mathcal{K}(p \times s_N, B)\, dp \tag{5.2.23}$$

$$= \mathcal{P}(S_{t,m} \in A | S_{t-1,N} = s_N)\, \mathcal{P}(S_{t,n} \in B | S_{t-1,N} = s_N) \tag{5.2.24}$$

□

As a consequence $S_{t,m} \perp\!\!\!\perp S_{t,n} \mid \mathfrak{A}_{t-1,\mathcal{N}(m)\cap\mathcal{N}(n)}$. An we can conclude (without proof) the following:

Corollary 5.2.12 (Local Markov Property)**.** *The* local Markov property

$$r \geq (s,n) \curlywedge (t,m) \Longrightarrow S_{t,m} \perp\!\!\!\perp S_{s,n} \mid \mathfrak{A}_r \tag{5.2.25}$$

is satisfied.

5.2.3.3 Global Markov Process

Furthermore the global evolution of a stochastic cellular automaton is a Markov process.

Proposition 5.2.13 (Global Markov Process)**.** *The global states $(\mathcal{S}_t)_{t\in T}$ form a homogeneous Markov process.*

Proof. $(\mathcal{S}_t)_{t\in T}$ is of course adapted to the filtration $(\mathfrak{A}_t)_{t\in T}$. The classical Markov property $\mathcal{S}_{t+1} \perp\!\!\!\perp \mathfrak{A}_t \mid \mathcal{S}_t$ is satisfied because the Markov property from Definition 5.2.4 implies that $S_{t+1,m} \perp\!\!\!\perp \mathfrak{A}_t \mid \mathcal{S}_t$ for every $m \in M$. Homogeneity of $(\mathcal{S}_t)_{t\in T}$ follows from Definition 5.2.5 together with Equation 5.2.6. □

Proposition 5.2.14 (Global Kernel and Global Markov Process)**.** *The global kernel $\mathcal{E}$ of a stochastic cellular automaton is the transition kernel of the global Markov process $(\mathcal{S}_t)_{t\in T}$.*

Proof. $\mathcal{E}$ is associated with the stochastic process $(\mathcal{S}_t)_{t\in T}$ because

$$\mathcal{E}\Big(s_M, \prod_{m\in M} B_m\Big) = \prod_{m\in M} \mathcal{L}_m(s_M, B_m) \tag{5.2.26}$$

$$= \prod_{m\in M} \mathcal{P}\big(S_{t+1,m}^{-1}(B_m)\big|\mathcal{S}_t = s_M\big) \tag{5.2.27}$$

$$\stackrel{?}{=} \mathcal{P}\Big(\bigcap_{m\in M} S_{t+1,m}^{-1}(B_m)\Big|\mathcal{S}_t = s_M\Big) \tag{5.2.28}$$

$$= \mathcal{P}\Big(\mathcal{S}_{t+1} \in \prod_{m\in M} B_m\Big|\mathcal{S}_t = s_M\Big). \tag{5.2.29}$$

$(\stackrel{?}{=})$ holds because $S_{t+1,m} \perp\!\!\!\perp S_{t+1,n} \mid \mathcal{S}_t$ for $m \neq n$ (Corollary 5.2.12). Since the assignment of a probability kernel (transition function) to a Markov process is unique it follows that

$$\mathcal{E}(s_M, B_M) = \mathcal{P}(\mathcal{S}_{t+1} \in B_M|\mathcal{S}_t = s_M) \tag{5.2.30}$$

for all $s_M \in \mathbb{S}^M$ and $B_M \in \sigma(\mathfrak{B}^M)$. It may be necessary to use a monotone class argument as in Proposition 5.1.16. □

$\mathcal{E}$ also satisfies the classical Chapman-Kolmogorov equation

$$\mathcal{P}(\mathcal{S}_{t+2} \in B_M|\mathcal{S}_t = s_M) = \int_{\mathbb{S}^M} \mathcal{E}(s_M, dq)\, \mathcal{E}(q, B). \tag{5.2.31}$$

The previous considerations raise the question, under which conditions a Markov process $(\mathcal{S}_t)_{t\in T}$ with states in $\big(\mathbb{S}^M, \sigma(\mathfrak{B}^M)\big)$ is also a stochastic cellular automaton?

Lemma 5.2.15 (Local Kernel Induced by Global Kernel)**.** *Let $(\mathcal{S}_t)_{t\in T}$ with $\mathcal{S}_t = (S_{t,m})_{m\in M}$ be a Markov process with states in $\big(\mathbb{S}^M, \sigma(\mathfrak{B}^M)\big)$, filtration $(\mathfrak{A}_t)_{t\in T}$ and kernel $\mathcal{E} : \mathbb{S}^M \times \sigma(\mathfrak{B}^M) \to [0,1]$. $(S_{t,m})_{(t,m)\in T\times M}$ is a stochastic cellular automaton with global kernel $\mathcal{E}$ if there exists a $k \in \mathbb{N}$, a kernel $\mathcal{K} : \mathbb{S}^k \times \mathfrak{B} \to [0,1]$ and a neighbourhood mapping $\mathcal{N} : M \to M^k$ such that*

$$\mathcal{E}(s_M, B \times \mathbb{S}^{M\setminus\{m\}}) = \mathcal{K}(s_{\mathcal{N}(m)}, B) \tag{5.2.32}$$

for every $m \in M$ and $B \in \mathfrak{B}$.

Proof. (i) We need to prove that there exists a product graph (as defined in Definition 5.2.1) on $T \times M$: We can assume that T is totally ordered. Together with the mapping $\mathcal{N}$ this requirement is satisfied.

(ii) There exists an additional filtration $(\mathfrak{A}_{t,m})_{(t,m)\in T\times M}$ as defined in Definition 5.2.2 and the random variables $S_{t,m}$ are adapted to this filtration: Define $(\mathfrak{A}_{t,m})_{(t,m)\in T\times M}$ as the natural filtration of the stochastic process $(S_{t,m})_{(t,m)\in T\times M}$, $\mathfrak{A}_{t,m} := \sigma(S_{t,m})$. Then $\mathfrak{A}_{t,m} \subseteq \mathfrak{A}_t$ for all $t \in T$ and $m \in M$ because

$$\prod_{m\in M} S_{t,m}^{-1}(B_m) = \mathcal{S}_t^{-1}\Big(\prod_{m\in M} B_m\Big) \tag{5.2.33}$$

for arbitrary $B_m \in \mathfrak{B}$ where $m \in M$.

(iii) We have to show that the Markov property (Definition 5.2.4) is satisfied,

$$S_{t+1,m} \perp\!\!\!\perp \mathfrak{A}_t \mid S_{t,\mathcal{N}(m)}. \tag{5.2.34}$$

This can be concluded from

$$\mathcal{P}(S_{t+1,m} \in B|\mathcal{S}_t = s_M) = \mathcal{P}(\mathcal{S}_{t+1} \in B \times \mathbb{S}^{M\setminus\{m\}}|\mathcal{S}_t = s_M) = \tag{5.2.35}$$

$$= \mathcal{E}(s_M, B \times \mathbb{S}^{M\setminus\{m\}}) = \mathcal{K}(s_{\mathcal{N}(m)}, B). \tag{5.2.36}$$

(iv) $\mathcal{K}$ is the local kernel and $\mathcal{E}$ is the global kernel of this stochastic cellular automaton: Obviously

$$\mathcal{K}(s_{\mathcal{N}(m)}, B) = \mathcal{P}(S_{t+1,m} \in B|S_{t,\mathcal{N}(m)} = s_{\mathcal{N}(m)}) \tag{5.2.37}$$

and

$$\mathcal{E}\Big(s_M, \prod_{m\in M} B_m\Big) = \mathcal{E}\Big(s_M, \bigcap_{m\in M} \big(B_m \times \mathbb{S}^{M\setminus\{m\}}\big)\Big) \tag{5.2.38}$$

$$= \prod_{m\in M} \mathcal{E}\big(s_M, B_m \times \mathbb{S}^{M\setminus\{m\}}\big) \tag{5.2.39}$$

$$= \prod_{m\in M} \mathcal{K}\big(s_{\mathcal{N}(m)}, B_m\big) \tag{5.2.40}$$

where $B_m \in \mathfrak{B}$ for all $m \in M$ shows that $\mathcal{E}$ is the global kernel. □

There may however exist different conditions – and more elaborate proofs – under which a (global) Markov process can be identified as a stochastic cellular automaton. The previous lemma however gives a general idea of how such a situation would look like. If it is not explicitly known that a global random state $\mathcal{S}_t$ is the product of random states $\prod_{m\in M} S_{t,m}$, it might be necessary to require that the transition probabilities of the global Markov process are disintegrations (Definition 5.1.10) rather than regular conditional probabilities. A detailed investigation is out of the scope of this thesis.

A primary conclusion from this section is that stochastic cellular automata also allow an alternative definition (compared to Definition 5.2.6) which only implicitly features the graphical structure defined in Definition 5.2.1.

5.3 Application Scenarios and Outlook

The aim of this section is to demonstrate the application of statistical methods on stochastic cellular automata and to provide some approaches for reducing the computational effort by finding deterministic representations of stochastic cellular automata.

Since stochastic cellular automata are settled between Bayesian networks and multiparameter Markov processes, obviously results from both areas can be adopted.

Finally an artificial application scenario is discussed in detail.

5.3.1 Introduction

As an introduction this section contains a short discussion of statistical and probabilistic information in stochastic cellular automata and some considerations on the Monte Carlo method.

5.3.1.1 Statistical Information in Stochastic Cellular Automata

The (stochastic) update rule for a stochastic cellular automaton is defined as a probability kernel $\mathcal{K} : \mathbb{S}^k \times \mathfrak{B} \to [0,1]$. This function can be thought of as an implicit representation of an (explicit) function $\mathcal{F}$ that maps (a posteriori) information about the states of neighbouring cells onto (a priori) information about the state of a cell in the next iterative (time) step.

We characterise different types of information:

- possible (a posteriori) information about states of neighbouring cells:
  - actual states of the neighbouring cells
  - distribution of the random states of the neighbouring cells
- cases for (a priori) information about the state of a cell:
  - actual state of the cell
  - distribution of the state of the cell
  - weaker information about the distribution of the state of the cell

And we can also distinguish two different ways of how a stochastic update rule defined through a probability kernel maps information about neighbouring random states onto information about a new state:

(K1) actual states (realisations of random states) of neighbouring cells $\mapsto$ distribution of a new random state

$$\mathbb{S}^k \to (\mathfrak{B} \to [0,1]) : (s_1, \ldots, s_k) \mapsto P(\,\cdot\,) := \mathcal{K}\big((s_1, \ldots, s_k), \cdot\,\big) \tag{5.3.1}$$

(K2) distributions of random states (possibly a posteriori) $\mapsto$ a priori information about the distribution of a new random state (in fact, a conditional probability distribution)

$$(\mathfrak{B} \to [0,1])^k \to (\mathfrak{B} \to [0,1]) : (P_1, \ldots, P_k) \mapsto P(\,\cdot\,|P_1, \ldots, P_k) := \tag{5.3.2}$$

$$:= \int_{\mathbb{S}} \cdots \int_{\mathbb{S}} \mathcal{K}\big((s_1, \ldots, s_k), \,\cdot\,\big)\, dP_1(s_1) \cdots dP_k(s_k) \tag{5.3.3}$$

Since integral operators are linear, the abstract mapping $\mathcal{F}$ in form of (K2) is multilinear with the limitation that only convex linear combinations of the form $\delta P_{j1} + (1-\delta)P_{j2}$ are permitted:

$$(P_1, \ldots, \delta P_{j1} + (1-\delta)P_{j2}, \ldots, P_k) \mapsto \tag{5.3.4}$$

$$\mapsto \delta P(\,\cdot\,|P_1, \ldots, P_{j1}, \ldots, P_k) + (1-\delta)P(\,\cdot\,|P_1, \ldots, P_{j2}, \ldots, P_k) \tag{5.3.5}$$

Definition 5.3.1 (Multiconvex mapping)**.** We may call $\mathcal{F}$ in form of (K2) a *multiconvex* mapping.

This feature is due to the fact that the set of probability measures is closed under convex combination.

5.3.1.2 Interpretation of Stochastic Cellular Automaton Models

Although stochastic cellular automata are defined as an iteration (or collection) of random variables through Markov kernels, especially the observation of single trajectories (realisations of random variables) allows different and more abstract interpretations of the iteration process (see Figure 5.3).

(U1) Iteration of stochastic states, random variables or their distributions using a probability kernel. This situation corresponds to the definition of stochastic cellular automata and is a straight forward application of Markov kernels (K2).

(U2) Iteration of probability distributions, which are chosen randomly depending on the distributions of the neighbouring cells. Such an iteration represents a single random trajectory of a stochastic cellular automaton. For calculating a new stochastic state (distribution), a realisation of the neighbouring random states is mapped (K1) onto a probability distribution for the new stochastic state of the actual cell.

(U3) Iteration of states, which are chosen randomly depending on the states of the neighbouring cells. The corresponding update mechanism consists of applying the kernel on a deterministic state in order to obtain a probability distribution (K1) and then choosing a realisation of this distribution as the new state of a cell.

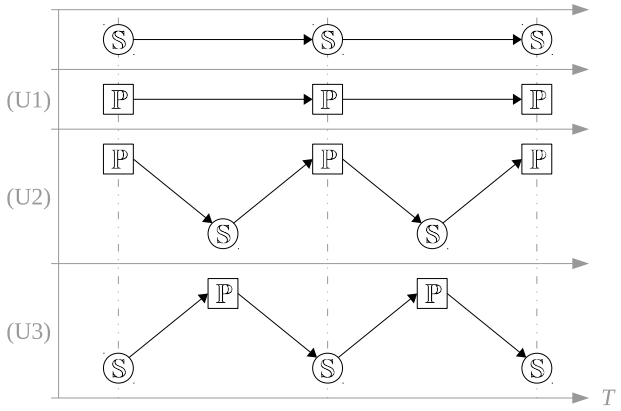

Figure 5.3: Different interpretations of the iterative process of stochastic cellular automata. Iterative steps are indicated by vertical lines. For comparison, at the top we can see an iteration of deterministic states in $\mathbb{S}$ (deterministic cellular automaton). Below (U1) the iteration of random states (random variables) $\mathbb{P}$ is sketched. The third row (U2) represents a trajectory of a stochastic cellular automaton and the last row (U3) is a different interpretation of a trajectory. Of course a cellular automaton features multiple input arguments for an update rule (neighbourhood), for simplicity multiple states (stochastic or deterministic) are indicated by $\mathbb{P}$ respectively $\mathbb{S}$. Arrows with a positive vertical component indicate an increase in entropy, arrows with a negative vertical component indicate decrease of entropy.

5.3.1.3 Simulation of Stochastic Cellular Automaton Models

For a cellular automaton with random states, (a posteriori) information about the state of a cell – if not available directly – can be gathered from realisations of the random state (p.e. through parameter estimators). In fact the empirical measure deduced from realisations of a random variable is equal (converges uniformly almost surely) to its distribution at least for real valued random variables (Glivenko-Cantelli Theorem, compare [27, p. 353]).

Hence we can say that a very large number of realisations of a random state contains almost the same information as its probability distribution. The realisation of random variables destroys the entropy but the observation of a large number of realisation (at least partially) restores the entropy.

This is of course also the basis for the application of the Monte Carlo method on which simulations of stochastic cellular automaton models usually rely when the Markov kernel does not allow an explicit calculation or is not given explicitly at all.

Basically a large number of simulations with one of the methods (U2) or (U3) is observed and the different outcomes are combined to a probability distribution for the final state of the stochastic cellular automaton. Two different Monte Carlo approaches can be distinguished:

- Multiple runs with the method described in (U3) and using the final empirical distribution.
- For every step a large number of realisations are mapped onto new states which are then combined to an empiric distribution. Compare also with (U2).

The application of the Monte Carlo method can be compared to the situation of a deterministic (evolution) system when no analytic solutions exist and iterative numerical approximations must be used. Generally it would be favourable if there exists an "analytic" method to calculate the final global state (i.e. its distribution) from the initial global state. We will however see that greater "analyticity" often also means reduced entropy.

Results from the field of *Markov chain Monte Carlo methods* (compare [46]) could be used also for stochastic cellular automata to analyse the computational effort and accuracy of such simulation approaches.

Furthermore stochastic cellular automata can also be used to describe implicit systems like for example the Ising model (compare *mean field approach*).

5.3.2 Classification of Random States

Instead of observing abstract random variables, different representations of probability distributions can be used to formulate the iteration of random states in a more accessible way.

5.3.2.1 Absolutely Continuous Random States

Theorem 5.3.2 (Radon-Nikodým, compare [6, p. 281])**.** *Let $P_{t,m}$ be the measures $\mathfrak{B} \to [0,1]$ associated with the random states $S_{t,m}$. Assume that there exits a σ-finite measure μ on $(\mathbb{S}, \mathfrak{B})$ such that*[11] *$P_{t,m} \ll \mu$ for all $(t,m) \in T \times M$. Then there exist probability densities $\rho_{t,m}(\,\cdot\,) : \mathbb{S} \to \mathbb{R}_+$ such that for all $B \in \mathfrak{B}$*

$$\mathcal{P}(S_{t,m} \in B) = P_{t,m}(B) = \int_B \rho_{t,m}(s)\, \mathrm{d}\mu(s). \tag{5.3.6}$$

Proof. Literature. □

Let all random states $S_{t,m}$ be absolutely continuous with densities $\rho_{t,m}$. The local kernel implies a mapping (K1)

$$\check{\mathcal{F}} : \mathbb{S}^k \to (\mathbb{S} \to \mathbb{R}_+) : (s_1, \ldots, s_k) \mapsto \rho(s|s_1, \ldots, s_k) := \mathcal{K}\big((s_1, \ldots, s_k), s\big) \tag{5.3.7}$$

and a multiconvex mapping (K2) $\hat{\mathcal{F}}$ from $(\mathbb{S} \to \mathbb{R}_+)^k$ to $(\mathbb{S} \to \mathbb{R}_+)$

$$(\rho_1, \ldots, \rho_k) \mapsto \rho(s) := \int_{\mathbb{S}} \cdots \int_{\mathbb{S}} \rho(s|s_1, \ldots, s_k)\rho_1(s_1) \cdots \rho_k(s_k)\, d\mu(s_1) \cdots d\mu(s_k). \tag{5.3.8}$$

$\rho(\,\cdot\,)$ is a marginal density of the joint density of S and S_i for all $i = 1, \ldots, k$.

[11] ν is absolutely continuous with respect to μ, $\nu \ll \mu$, if $\mu(N) = 0 \Longrightarrow \nu(N) = 0$ [6, p. 279].

5.3.2.2 Parametrised Random States

Definition 5.3.3 (Parametrised Random States)**.** The random states of a stochastic cellular automaton are called *parametrised random states* if there exists a distribution or family of distributions Dist_θ with parameter space Θ and

$$S_{t,m} \sim \mathrm{Dist}_{\theta_{t,m}} \tag{5.3.9}$$

for all $(t, m) \in T \times M$ and some $\theta_{t,m} \in \Theta$.

The family of distributions $(\mathrm{Dist}_\theta)_{\theta\in\Theta}$ is called a *stochastic model* for the states of a stochastic cellular automaton. The restriction to parametrised states is always associated with the introduction of a stochastic model for the random states and also presents a very strong restriction, comparable to reducing the state space in a deterministic cellular automaton.

Since a probability kernel maps the distributions of the neighbouring cells onto a new distribution (K2), there exists a mapping

$$\hat{\mathcal{F}} : \Theta^k \to \Theta : (\theta_1, \ldots, \theta_k) \mapsto \theta, \tag{5.3.10}$$

which satisfies

$$S_{t,n_i} \sim \mathrm{Dist}_{\theta_i} \quad \forall n_i \in \mathcal{N}(m) \qquad \Longrightarrow \qquad S_{t+1,m} \sim \mathrm{Dist}_{\hat{\mathcal{F}}(\theta_1,\ldots,\theta_k)}\,. \tag{5.3.11}$$

Compared to the general form of (K2) we can not assume that $\hat{\mathcal{F}}$ is multiconvex or multilinear.

Because the kernel can also be used to map actual states onto a new distribution (K1), there also exists a mapping

$$\check{\mathcal{F}} : \mathbb{S}^k \to \Theta : (s_1, \ldots, s_k) \mapsto \theta \tag{5.3.12}$$

with

$$S_{t,n_i} = s_i \quad \forall n_i \in \mathcal{N}(m) \qquad \Longrightarrow \qquad S_{t+1,m} \sim \mathrm{Dist}_{\check{\mathcal{F}}(s_1,\ldots,s_k)}\,. \tag{5.3.13}$$

Proposition 5.3.4. *Let for all $i = 1, \ldots, k$, P_i be the probability measure on $(\mathbb{S}, \mathfrak{B})$ of the distribution Dist_{θ_i} associated with the parameter θ_i. Then*

$$\int_{\mathbb{S}} \cdots \int_{\mathbb{S}} \check{\mathcal{F}}(s_1, \ldots, s_k)\, dP_1(s_1) \cdots dP_k(s_k) = \hat{\mathcal{F}}(\theta_1, \ldots, \theta_k). \tag{5.3.14}$$

Proof. Without proof. □

For the probability measure $P : \mathfrak{B} \to [0, 1]$ associated with the random variable $S \sim \mathrm{Dist}_\theta$ we also use the notation $P(\,\cdot\,|\theta)$ in order to avoid having to explicitly mention the distribution of the associated random variable.

5.3.2.3 Discrete Random States and Discretisation of Random States

Besides parametrised stochastic states, a discretisation of the state space $\mathbb{S}$ can be used to obtain a deterministic representation of random states even if there exists no common family of distributions (i.e. stochastic model).

Definition 5.3.5 (Discretised Random States)**.** Let $\mathbb{S}$ be a (bounded) domain, which can be discretised into r compartments $s_j \cong \mathbb{S}_j$, $j = 1, \ldots, r$. The probability measure of a random state $(\Omega, \mathfrak{A}, \mathcal{P}) \to (\mathbb{S}, \mathfrak{B})$ can then be approximated by a probability function $\overline{\mathbb{S}} := \{s_1, \ldots, s_r\} \cong \bigcup_{j=1}^{r} \mathbb{S}_j \to [0,1]$ respectively $\{1, \ldots, r\} \to [0,1]$ and consequently also by a vector in $[0,1]^r$. For such a vector $\vec{p} \in [0,1]^r$, obviously $\sum_{j=1}^{r} p_j = 1$.

Analogously to the previous section, (K1) can be represented as a mapping

$$\check{\mathcal{F}} : \overline{\mathbb{S}}^k \to [0,1]^r : (s_1, \ldots, s_k) \mapsto \vec{p} \tag{5.3.15}$$

and (K2) yields

$$\hat{\mathcal{F}} : \left([0,1]^r\right)^k \to [0,1]^r : (\vec{p}_1, \ldots, \vec{p}_k) \mapsto \vec{p}. \tag{5.3.16}$$

Proposition 5.3.6. *Let $\vec{p}_i = (p_{i,1}, \ldots, p_{i,r})$ be discrete distributions for $i = 1, \ldots, k$ and assume that $\overline{\mathbb{S}} = \{s_1, \ldots, s_r\}$. Then*

$$\sum_{j_1=1}^{r} \cdots \sum_{j_k=1}^{r} \check{\mathcal{F}}(s_{j_1}, \ldots, s_{j_k})\, p_{1,j_1} \cdots p_{k,j_k} = \hat{\mathcal{F}}(\vec{p}_1, \ldots, \vec{p}_k) \tag{5.3.17}$$

Proof. Without proof. □

The configuration of a neighbourhood is determined by the product distribution of k random variables (probability vectors with dimension r)

$$p_{\mathcal{N}(m)} := \vec{p}_1 \otimes \cdots \otimes \vec{p}_k \in \underbrace{[0,1]^r \otimes \cdots \otimes [0,1]^r}_{k} \tag{5.3.18}$$

which is a tensor of type $(k, 0)$.

Since an integral transform is a linear operation, it is clear that also for discrete random states the local kernel (K2) respectively $\hat{\mathcal{F}}$ defines a (convex) linear mapping

$$\hat{F} : [0,1]^r \otimes \cdots \otimes [0,1]^r \to [0,1]^r : \vec{p}_1 \otimes \cdots \otimes \vec{p}_k \mapsto \vec{p} \tag{5.3.19}$$

of $(k, 0)$-tensors onto $(1, 0)$-tensors. Accordingly there exists a multilinear – or more accurately a multiconvex – mapping $[0,1]^r \times \cdots \times [0,1]^r \to [0,1]^r$ which obviously corresponds to $\hat{\mathcal{F}}$ (see Proposition 5.3.6) such that

$$\hat{F}(\vec{p}_1 \otimes \cdots \otimes \vec{p}_k) = \hat{\mathcal{F}}(\vec{p}_1, \cdots, \vec{p}_k). \tag{5.3.20}$$

Definition 5.3.7 (Transition Tensor)**.** In correspondence with Markov chains we call $\hat{F}$ and also $\hat{\mathcal{F}}$ *transition tensor*.

A transition matrix for ordinary Markov chains is obviously a (multi-) convex mapping.

Let $(s_{j_1}, \dots, s_{j_k})$ be a realisation of random states of a neighbourhood. This (deterministic) state configuration corresponds to the distribution vectors $(e_{j_1}, \dots, e_{j_k})$ and

$$\hat{F}(e_{j_1} \otimes \cdots \otimes e_{j_k}) = \check{F}(s_{j_1}, \dots, s_{j_k}). \tag{5.3.21}$$

We can say that the vectors of the form $\check{F}(s_{j_1}, \dots, s_{j_k})$ present a basis of $[0,1]^r$.

5.3.3 Pseudo-Stochastic Cellular Automata

In this section some special cases for which there exist deterministic representations of stochastic cellular automata are presented. These types of stochastic cellular automata will henceforth be called *pseudo-stochastic cellular automata*.

5.3.3.1 Deterministic Iteration of Random States

Given a stochastic cellular automaton with random states $S_{t,m} : (\Omega, \mathfrak{A}, \mathcal{P}) \to (\mathbb{S}, \mathfrak{B})$, assume that there exists a measurable function $\mathcal{F} : \big(\mathbb{S}^k, \sigma(\mathfrak{B}^k)\big) \to (\mathbb{S}, \mathfrak{B})$ such that

$$S_{t+1,m} = \mathcal{F}(S_{t,n_1}, \dots, S_{t,n_k}) \qquad \forall\, t \in T, m \in M \tag{5.3.22}$$

where $\mathcal{N}(m) = (n_1, \dots, n_k)$.

In this case the corresponding local kernel satisfies

$$\mathcal{K}(\vec{s}, B) = \mathcal{P}(S_{t+1,m} \in B | S_{t,\mathcal{N}(m)} = \vec{s}) \tag{5.3.23}$$

$$= \mathcal{P}(S_{t,\mathcal{N}(m)} \in \mathcal{F}^{-1}B | S_{t,\mathcal{N}(m)} = \vec{s}) = \begin{cases} 1 & \vec{s} \in \mathcal{F}^{-1}B \\ 0 & \text{else} \end{cases} \tag{5.3.24}$$

and accordingly can be written as $\mathcal{K}(\vec{s}, B) = \mathbb{I}_{\mathcal{F}^{-1}B}(\vec{s})$. For the iteration of distributions (K2) this means

$$P(B|P_1, \dots, P_k) = \int_{\mathbb{S}} \cdots \int_{\mathbb{S}} \mathbb{I}_{\mathcal{F}^{-1}B}\big((s_1, \dots, s_k)\big)\, dP_1(s_1) \dots dP_k(s_k). \tag{5.3.25}$$

In contrast to the formalisms with deterministic states in $\mathbb{S}$, the update function $\mathcal{F}$ is required to be measurable (with respect to the product-σ-algebra). Since $\mathcal{F}$ is measurable, the distributions of all individual random states $S_{t,m}$ for higher times can be obtained from the given initial condition as push-forward measures.

Example 5.3.8. If $\mathbb{S}$ is finite, the associated (pseudo-) stochastic cellular automaton is a Bayesian network,

$$\mathcal{P}\big(\mathcal{S}_{t+1} = s_{t+1,M} \,\big|\, \mathcal{S}_t = s_{t,M}\big) = \prod_{m \in M} \mathcal{P}\big(S_{t+1,m} = s_{t+1,m} \,\big|\, S_{t,\mathcal{N}(m)} = s_{t,\mathcal{N}(m)}\big) \tag{5.3.26}$$

and the joint distribution is a product of local kernels (Proposition 5.1.46)

$$\mathcal{P}\big(\mathcal{S}_T = s_{T,M}\big) = \prod_{t\in T} \mathcal{P}\big(\mathcal{S}_{t+1} = s_{t+1,M} \,\big|\, \mathcal{S}_t = s_{t,M}\big) \cdot \mathcal{P}\big(\mathcal{S}_0 = s_{0,M}\big) \tag{5.3.27}$$

$$= \prod_{t\in T} \prod_{m\in M} \mathbb{I}_{\mathcal{F}^{-1}\{s_{t+1,m}\}}(s_{t,\mathcal{N}(m)}) \cdot \mathcal{P}\big(\mathcal{S}_0 = s_{0,M}\big). \tag{5.3.28}$$

As a consequence, $\mathcal{P}(\mathcal{S}_T = s_{T,M}) = \mathcal{P}(\mathcal{S}_0 = s_{0,M})$ if $s_{t+1,m} = \mathcal{F}(s_{t,n_1}, \ldots, s_{t,n_k})$ for all $t \in T$ and $m \in M$ where $\mathcal{N}(m) = (n_1, \ldots, n_k)$. $\mathcal{S}_T$ itself is a random variable $(\Omega, \mathfrak{A}, \mathcal{P}) \to (\mathbb{S}^{T\times M}, \mathfrak{B}_{T\times M})$.

In the general case, for not necessarily finite state space $\mathbb{S}$, there exists a *global (pseudo-stochastic) evolution operator* that maps the distribution of a global random state onto the distribution of the global random state in the next iterative step. This global evolution operator is a measurable mapping

$$\mathcal{E} = \prod_{m\in M} \mathcal{F} \circ \mathrm{proj}_{\mathcal{N}(m)} : (\mathbb{S}^M, \mathfrak{B}_M) \to (\mathbb{S}^M, \mathfrak{B}_M). \tag{5.3.29}$$

Compare the definition of local and global evolution operators in Section 4.1.

5.3.3.2 Deterministic Iteration of Absolutely Continuous Random States

Let $\rho_{t,m}$ be the densities of the random states $S_{t,m}$. Assume that $\mathcal{F}$ is a measurable differentiable mapping $\mathbb{S}^k \to \mathbb{S}$. The random state $S_{t+1,m} = \mathcal{F}(S_{t,n_1}, \ldots, S_{t,n_k})$ has the density function

$$\rho_{t+1,m}(s) = \big[\rho_{t,n_1}(\,\cdot\,) \cdots \rho_{t,n_k}(\,\cdot\,)\big]\big(\mathcal{F}^{-1}(s)\big) \cdot \left|\frac{d\mathcal{F}^{-1}(s)}{ds}\right|. \tag{5.3.30}$$

Bringing this equation in the form

$$\rho_{t+1,m}(s) = \int_{\mathbb{S}} \cdots \int_{\mathbb{S}} \left|\frac{d\mathcal{F}^{-1}(s)}{ds}\right| \mathbb{I}_{\{\mathcal{F}^{-1}(s)\}}(s_1, \ldots, s_k)\rho_1(s_1) \cdots \rho_k(s_k)\, d\mu(s_1) \cdots d\mu(s_k) \tag{5.3.31}$$

and comparing with Equation 5.3.8 shows that

$$\check{\mathcal{F}}(s_1, \ldots, s_k)(s) = \mathcal{K}\big((s_1, \ldots, s_k), s\big) = \tag{5.3.32}$$

$$= \rho(s|s_1, \ldots, s_k) = \left|\frac{d\mathcal{F}^{-1}(s)}{ds}\right| \mathbb{I}_{\{\mathcal{F}^{-1}(s)\}}(s_1, \ldots, s_k). \tag{5.3.33}$$

5.3.3.3 Deterministic Iteration of Parametrised Random States

Assume that in addition to the measurable mapping

$$\mathcal{F} : \big(\mathbb{S}^k, \sigma(\mathfrak{B}^k)\big) \to (\mathbb{S}, \mathfrak{B}) \tag{5.3.34}$$

which also implies a mapping $(S_{t,n_1}, \ldots, S_{t,n_k}) \mapsto S_{t+1,m}$, there exists a stochastic model Dist_θ, $\theta \in \Theta$ for the random states and a mapping

$$\hat{\mathcal{F}} : \Theta^k \to \Theta : (\theta_1, \ldots, \theta_k) \mapsto \theta \tag{5.3.35}$$

such that $S_{t,n_i} \sim \mathrm{Dist}_{\theta_i}$ for all $n_i \in \mathcal{N}(m)$ leads to $S_{t+1,m} \sim \mathrm{Dist}_\theta$.

Accordingly

$$S_i \sim \mathrm{Dist}_{\theta_i} \quad i = 1, \ldots, k \quad \Longrightarrow \quad \mathcal{F}(S_1, \ldots, S_k) \sim \mathrm{Dist}_{\hat{\mathcal{F}}(\theta_1, \ldots, \theta_k)} \tag{5.3.36}$$

and

$$P\big(B \big| \hat{\mathcal{F}}(\theta_1, \ldots, \theta_k)\big) = \int_{\mathcal{F}^{-1}B} \prod_{i=1}^{k} P(ds_i|\theta_i). \tag{5.3.37}$$

Definition 5.3.9 (Conjugate Update Rules)**.** We call $\mathcal{F}$ and $\hat{\mathcal{F}}$ *conjugate* update functions.

Again we cannot make any general statements about the (multi-) linearity of $\hat{\mathcal{F}}$. The following example (Example 5.3.10) shows a stochastic model (parametrisation) for which there exists an almost linear conjugate update function, namely the family of normal distributions.

Example 5.3.10. A good but rather theoretic example for conjugate update rules is where the random states are normally distributed ($\mathrm{Dist}_\theta = \mathrm{Norm}_\theta$) on $\mathbb{S} = \mathbb{R}$ and the update function $\mathcal{F}$ is a linear map $\mathbb{R}^k \to \mathbb{R}$. In this case $\hat{\mathcal{F}}$ can directly be obtained from $\mathcal{F}$ or vice versa. Compare the second quote in Section 5.1.6.

$$S_i \sim \mathrm{Norm}_{\mu_i, \sigma_i^2} \quad \Longrightarrow \quad \sum_{i=1}^{k} \alpha_i S_i \sim \mathrm{Norm}_{\sum_{i=1}^k \alpha_i \mu_i, \sum_{i=1}^k (\alpha_i \sigma_i)^2} \tag{5.3.38}$$

There exist two different approaches for simulating such a pseudo-stochastic cellular automaton model.

- The direct approach is to generate a realisation of the initial global random state and then iterate the resulting deterministic states using the function $\mathcal{F}$. Doing this repeatedly for many different realisations of the initial global states delivers an empiric distribution of the final global random state. See Figure 5.4 left-hand side for the estimated expectation value.
- Deterministic iteration of meta states. Given a global initial random state (distribution) with independently distributed random states including parameters $(\theta_{t,m})_{m \in M}$, it is easy to iteratively calculate the parameters of the distribution of the global random state at an arbitrary time using the function $\hat{\mathcal{F}}$. This scheme itself is also a deterministic cellular automaton.

From realisations of the obtained final distribution we could again estimate the expectation value (Figure 5.4 right-hand side) or any other parameter. For the normal distribution this is obviously not necessary since the expectation value is included in the parameter.

Listing A.1 shows the source code of an implementation in Octave. This implementation was also used to generate the images in Figure 5.4. The distortions in the right image can be smoothed out by averaging over a larger number of realisations of the final global state. Even then the computational effort of this parameter-iteration approach is much lower.

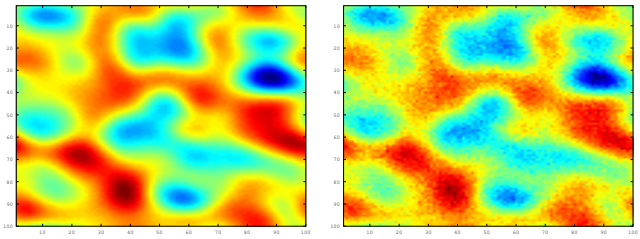

Figure 5.4: Application example of conjugate update rules (Example 5.3.10). Left: Iteration of randomly generated initial conditions. Right: iteration of parameters and estimated expectation value. Estimation of the expectation value is actually not necessary, since the expectation value is contained in the parameter. The graphical representation of the actual expectation value is equal to the left image. See Listing A.1 for the Octave source code.

5.4 Example: "Game of Life" – Stochastic Formulation

In this section an example for the application of the stochastic formalism on a deterministic cellular automaton is presented. This particular scenario is however purely theoretical and rather abstract since the underlying deterministic "model" is the "Game of Life" cellular automaton. This application example was also published in [43] whilst this thesis was being written. Several phrases and formulations from [43] are translated from German without quoting due to their generic character.

5.4.1 Application of the Cellular Automaton Formalism

The "Game of Life" cellular automaton is defined by the following parameters:

(M) The cells are regularly arranged on a rectangular two-dimensional lattice and we use periodic boundary conditions.

($\mathcal{N}$) The neighbourhood is a Moore-neighbourhood which also contains the actual cell itself and accordingly consists of 9 cells altogether.

($\mathbb{S}$) The state of a cell is either "dead" or "alive", we use the representation $\mathbb{S} = \{0, 1\}$.

($\mathcal{F}$) The local update rule $\mathcal{F}$ is given as

$$\big(S(t,n)\big)_{n\in\mathcal{N}(m)} \mapsto S(t,m) f_a(\Sigma) + \big(1 - S(t,m)\big) f_d(\Sigma) \tag{5.4.1}$$

where $f_a = \mathbb{I}_{\{\frac{2}{8},\frac{3}{8}\}}$ and $f_d = \mathbb{I}_{\{\frac{3}{8}\}}$ and the aggregate Σ is defined as

$$\Sigma(t,m) = \frac{1}{8} \sum_{n\in\mathcal{N}(m)\setminus\{m\}} S(t,n) \tag{5.4.2}$$

which is a sum over the states of the (eight) neighbours excluding the actual cell.

The following cellular automaton is a continuous version of the "Game of Life". Because of our theoretical approach we can say that the previous original formulation is a discretisation of this virtual model.

($\mathbb{S}$) Let the state space $\mathbb{S}$ be the interval $[0, 1]$.

($\mathcal{F}$) The local update rule $\mathcal{F}$ must be altered with $f_a = \mathbb{I}_{[\frac{2}{9},\frac{4}{9})}$ and $f_d = \mathbb{I}_{[\frac{3}{9},\frac{4}{9})}$ or similar. See Table 5.1 for an explanation of the values. A smoothed version of the functions f_a and f_d can be achieved for example by using functions

$$f_\lambda : \mathbb{R} \to [0,1] : x \mapsto \frac{1}{2}\left(\tanh\frac{x - x_1}{\lambda} - \tanh\frac{x - x_2}{\lambda}\right). \tag{5.4.3}$$

See Figure 5.5.

Due to the special choice of the functions $f_{\lambda,a}$ and $f_{\lambda,d}$ we can see in the top centre of Figure 5.6 that the continuous-state cellular automaton with very small λ respectively $f_a = \mathbb{I}_{[\frac{2}{9},\frac{4}{9})}$ delivers the same typical behaviour as the original discrete-sate "Game of Life" cellular automaton.

Because $f_{\lambda,a} \longrightarrow \mathbb{I}_{[\frac{2}{9},\frac{4}{9})}$ and $f_{\lambda,d} \longrightarrow \mathbb{I}_{[\frac{3}{9},\frac{4}{9})}$ for $\lambda \longrightarrow 0$, this transition can be used for identifying the smoothed update method with the original formulation for which $f_a = \mathbb{I}_{[\frac{2}{9},\frac{4}{9})}$ etc. See also Figure 5.6.

For larger λ we can expect different behaviour since the rule set deviates from the "Game of Life" configuration.

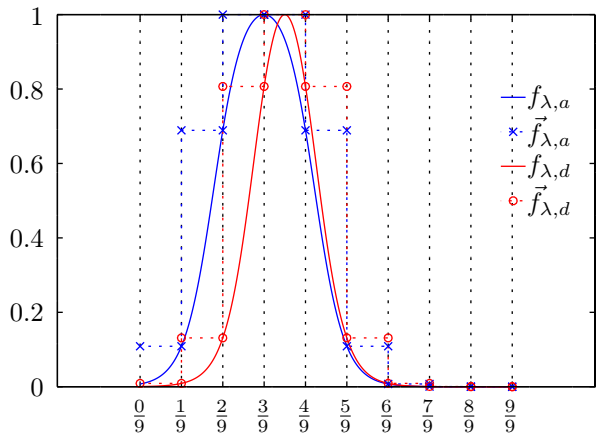

Figure 5.5: Smoothed update functions for the "Game of Life" cellular automaton. The figure shows the functions $f_{\lambda,a}$ (blue/crosses) and $f_{\lambda,d}$ (red/circles) and their discretised versions for $r = 9$ (see Table 5.1). There also exist different ways for defining the values of a discretisation other than by maximum.

5.4.2 Pseudo-Stochastic Formulation

($\mathbb{S}$) Let $(S_{t,m})_{(t,m)\in T\times M}$ be stochastic variables $(\Omega, \mathfrak{A}, \mathcal{P}) \to (\mathbb{S}, \mathfrak{B})$ where $\mathbb{S} = [0,1]$.

($\mathcal{K}$) Define the (pseudo-stochastic) local kernel through a measurable function (compare Section 5.3.3.1)

$$\mathcal{F} : (s_n)_{n\in\mathcal{N}(m)} \mapsto \Big(s_m, \underbrace{\frac{1}{8}\sum_{n\in\mathcal{N}(m)\setminus\{m\}} s_n}_{=:\Sigma_{t,m}}\Big) \mapsto \big[\underbrace{s_m f_a + (1-s_m) f_d}_{=:\mathcal{G}(s_m)}\big](\Sigma_{t,m}). \tag{5.4.4}$$

Accordingly $S_{t+1,m} = \mathcal{F}\big((S_{t,n})_{n\in\mathcal{N}(m)}\big)$ and

$$\mathcal{P}\Big(S_{t+1,m} \in B \Big| (S_{t,n})_{n\in\mathcal{N}(m)} = \vec{s}\Big) = 1 \quad \Longleftrightarrow \quad \vec{s} \in \mathcal{F}^{-1}B. \tag{5.4.5}$$

If we assume that for every random state $S_{t,m}$ there exists a probability density $\rho_{t,m} : \mathbb{S} \to \mathbb{R}_+$, the *transformation* of random variables defined in Equation 5.4.4 can be formulated as a transformation of probability densities.

In this case

$$\sigma_{t,m}(s) = 8\Big(\bigoplus_{n\in\mathcal{N}(m)\setminus\{m\}} \rho_{t,n}\Big)(8s) := 8\cdot \rho_{t,n_1} * \cdots * \rho_{t,n_8}(8s) \tag{5.4.6}$$

is the density of $\Sigma_{t,m}$. Since $f_{a,d}$ are piecewise monotonic, the densities $\gamma_{a,d}$ of $f_{a,d}(\Sigma_{t,m})$ can also be obtained using the transformation theorem. The density of

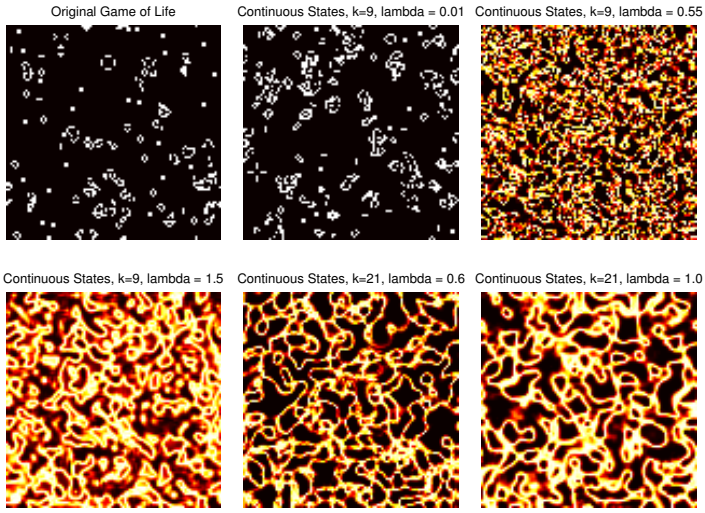

Figure 5.6: This figure shows the lattice of different versions of the "Game of Life" as defined in Section 5.4.1 after 100 iterations. The initial conditions were generated randomly with continuous states and rounded to discrete states for the original version of the "Game of Life" (top-left). For small λ the behaviour of the continuous models approaches the behaviour of the discrete original system. For an increased size of the neighbourhood $k = 21$ (compared to the default Moore neighbourhood with $k = 9$) the patterns are less tight. For values of λ greater than 2 the individual continuous states begin to jump between very high and very low in every iteration.

the random state in the next time-step $\rho_{t+1,m}$ finally (multiplication and convolution) yields

$$\rho_{t+1,m}(p) = \int_{\mathbb{S}}\int_{\mathbb{S}} \gamma_a(s)\gamma_d(t) \int_{\mathbb{S}} \rho_{t,m}\left(\frac{c}{s}\right) \rho_{t,m}\left(1 - \frac{p-c}{t}\right)\, dc\, dt\, ds. \tag{5.4.7}$$

We obviously achieved a deterministic formulation of the pseudo-stochastic cellular automaton defined in the beginning of this section. Without doubt this formulation is not practical for implementation respectively simulation on a computer system.

A further approach for obtaining the density $\rho_{t+1,m}$ consists in observing a different decomposition of $\mathcal{F}$:

The function $\mathcal{G}$ defined in Equation 5.4.4 maps values in $\mathbb{S}$ onto linear (convex) combinations $X := \{\delta f_a + (1-\delta)f_d, \delta \in [0,1]\}$ in the function-space $\mathbb{S} \to \mathbb{S}$,

$$\mathcal{G} : \mathbb{S} \to X : \delta \mapsto \delta f_a + (1-\delta)f_d =: f_\delta. \tag{5.4.8}$$

The inverse of $\mathcal{G}$ can be defined as

$$\mathcal{G}^{-1} : X \to \mathbb{S} : f_\delta(\,\cdot\,) \mapsto \frac{f_\delta(\,\cdot\,) - f_d(\,\cdot\,)}{f_a(\,\cdot\,) - f_d(\,\cdot\,)} \equiv \delta. \tag{5.4.9}$$

Accordingly we define the density $\gamma_{t,m}$ of $\mathcal{G}(S_{t,m})$ as the formal transformation

$$\gamma_{t,m}(f_\delta) = \rho_{t,m}\big(\mathcal{G}^{-1}(f_\delta)\big)\Big|\frac{d\mathcal{G}^{-1}(f_\delta)}{df_\delta}\Big| = \rho_{t,m}(\delta). \tag{5.4.10}$$

Using the functional derivative formalism for $\frac{d\mathcal{G}^{-1}(f_\delta)}{df_\delta}$ it might be possible to obtain more accurate results.

Altogether a future state $S_{t+1,m}$ takes a value of the form $f_\delta(s)$ with probability $\rho_{t,m}(\delta)\sigma_{t,m}(s)$. The expectation value of $S_{t+1,m}$ can be calculated explicitly as

$$\mathbb{E}[S_{t+1,m}] = \int_{\mathbb{S}}\int_{\mathbb{S}} f_\delta(s)\rho_{t,m}(\delta)\sigma_{t,m}(s)\,d\delta\,ds. \tag{5.4.11}$$

A simplified pseudo-stochastic formulation, which finally allows an explicit formulation of the update rule, can be obtained by combining the functions f_a and f_d.

($\mathcal{K}$) As we see in Figure 5.5, the difference $\|f_a - f_d\|$ is usually rather small. In this sense we can substitute the random variable respectively function $\mathcal{G}(S_{t,m}) = S_{t,m}\cdot(f_a - f_d) + f_d$ by $\mathbb{E}[\mathcal{G}(S_{t,m})] = \mathcal{G}(\mathbb{E}[S_{t,m}])$ which means that

$$S_{t+1,m} = \mathcal{F}\big((S_{t,n})_{n\in\mathcal{N}(m)}\big) = f_{\mathbb{E}[S_{t,m}]}(\Sigma_{t,m}) \tag{5.4.12}$$

5.4.3 Discretised Stochastic Formulation

($\mathbb{S}$) Let the continuous state space $\mathbb{S} = [0,1]$ be discretised into r compartments (Table 5.1) such that the probability densities of random states can be represented as (stochastic[12]) column vectors $\vec{\rho}$, $\vec{\sigma}$, ...

1	2	3	4	...	r
$\frac{0}{r-1}$	$\frac{1}{r-1}$	$\frac{2}{r-1}$	$\frac{3}{r-1}$	...	$\frac{r-1}{r-1}$
$[\frac{0}{r},\frac{1}{r})$	$[\frac{1}{r},\frac{2}{r})$	$[\frac{2}{r},\frac{3}{r})$	$[\frac{3}{r},\frac{4}{r})$	...	$[\frac{r-1}{r},\frac{r}{r})$

Table 5.1: Discretisation of the state-space of the continuous formulation of the "Game of Life" into r compartments. Since the original formulation of the "Game of Life" can be understood as a discretisation with $r = 8+1$ compartments, the distinct values $\frac{2}{8}$ respectively $\frac{3}{8}$ can be used to find corresponding intervals $[\frac{j}{r}.\frac{j+1}{r})$ for the continuous formulation.

In contrast to the previous section, where $\Sigma = \frac{1}{8}\sum_{j=1}^{8} S_{n_j}$, we now define $\Sigma = \sum_{j=1}^{8} S_{n_j}$. Accordingly the discretised density (probability function) $\vec{\sigma}$ is the (discrete) convolution of 8 vectors from $[0,1]^r$ and hence a vector in $[0,1]^{r+7(r-1)}$.

[12]The term *stochastic* is sometimes used to indicate that the sum of elements of a vector equals one.

The evaluation of $f_{a,d}(\Sigma)$ respectively the corresponding transformation can be formulated using the scalar product $\vec{f}_{a,d}^{\top} \cdot \vec{\sigma}$ which requires that also $\vec{f}_{a,d} \in [0,1]^{r+7(r-1)}$. The evaluation $\vec{f}_{a,d}(\frac{j}{r+7(r-1)-1})$ then corresponds to the scalar product $\vec{f}_{a,d}^{\top} \cdot \vec{e}_j$.

With the simplifications from Equation 5.4.12,

$$\vec{f} := \mathbb{E}[\mathcal{G}(S_m)] \qquad = \sum_{j=0}^{r-1} \mathcal{G}\big(\tfrac{j}{r-1}\big)\rho_j \tag{5.4.13}$$

$$= \Big(\big(\tfrac{0}{r-1}, \ldots, \tfrac{r-1}{r-1}\big) \cdot \vec{\rho}\Big)\vec{f}_a + \Big(\big(\tfrac{r-1}{r-1}, \ldots, \tfrac{0}{r-1}\big) \cdot \vec{\rho}\Big)\vec{f}_d \tag{5.4.14}$$

$$= \mathcal{G}(\mathbb{E}[S_m]) \qquad =: G\vec{\rho} \tag{5.4.15}$$

for a matrix $G \in [0,1]^{(r+7(r-1))\times r}$.

Since f is a mixture of f_a and f_d, we can give the following interpretation of the evaluation $\vec{f}(\vec{\sigma}) = \vec{f} \cdot \vec{\sigma}$: If the mass of $\vec{\sigma}$ is concentrated in the compartments around $\frac{2}{8}$ and $\frac{3}{8}$ then the result of $\vec{f} \cdot \vec{\sigma}$ is near 1. If on the other hand the probability in this region is small, then $\vec{f} \cdot \vec{\sigma}$ should be near 0.

Following this observation we define the following ("column-stochastic") matrix in $[0,1]^{r\times(r+7(r-1))}$

$$F := B + \vec{b} \cdot \vec{f}^{\top} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix} + \vec{b} \cdot \vec{f}^{\top} = \begin{pmatrix} (1,\ldots,1) - \vec{f}^{\top} \\ \vdots \\ \vec{f}^{\top} \end{pmatrix}. \tag{5.4.16}$$

This matrix actually delivers the desired behaviour: $\vec{\sigma} \approx \|\vec{f}\|_1^{-1}\vec{f}$ leads to a result $\vec{\rho} := F\vec{\sigma}$ with its mass concentrated in the compartments with higher indices. If $\vec{\sigma}$ is orthogonal to $\vec{f}$, the mass of the resulting vector is concentrated in the compartments with lower indices.

In other words, the orthogonality of $\vec{\sigma}$ and $\vec{f}$ determines if the cell is rather "alive" or "dead" in the next step. The vertical transition of the rows in the matrix F must not necessarily be linear. For example if z is the row-index, then $z^{\lambda^{-1}}(1 - \vec{f}^{\top}) + (1-z)^{\lambda^{-1}}\vec{f}^{\top}$ is also a valid transition. However the matrix F has to be "column-stochastic". In the bottom-left corner of Figure 5.8 a visualisation of a typical mass distribution of F is shown.

This somehow heuristic approach for the matrix F can be proved – or at least motivated – using a more general transformation theorem for piecewise monotonic functions (also known as *Frobenius-Perron operator*) on $S_{t+1,m} = f(\Sigma_{t,m})$.

$$\rho_{t+1,m}(y) = \sum_{i=1}^{2} \sigma_{t,m}\big(f_i^{-1}(y)\big) \left| \frac{df_i^{-1}(y)}{dy} \right| \tag{5.4.17}$$

where f_i are the monotonic components of f (see Figure 5.5 and Figure 5.7).

A straight forward discretisation of the functions used in Equation 5.4.17 is given by $f_{i,j} := f_i(\frac{j}{r-1})$ yielding a vector representation $\vec{f}_i$. $\vec{f}_i^{-1}$ and $|(\vec{f}_i^{-1})'|$ can be

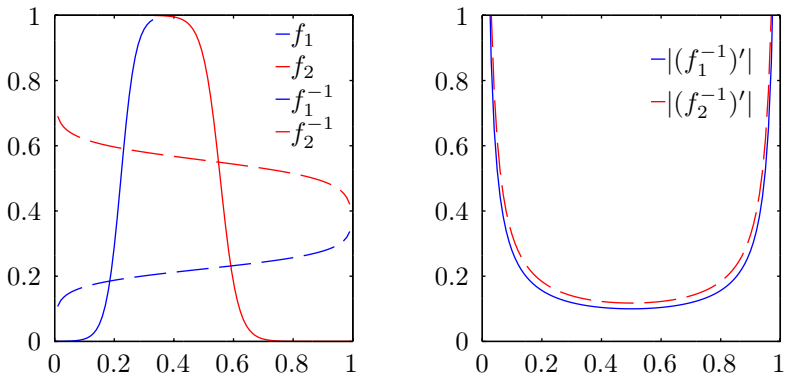

Figure 5.7: Transformation of a random variable by a piecewise monotonic function. The left figure shows a function f of the same type as in Equation 5.4.3 and the inverse functions $f_{1,2}^{-1}$ of its monotonic components $f_{1,2}$. The figure on the right depicts the derivatives $(f_{1,2}^{-1})'$ of the inverse functions.

obtained in the same way. Let furthermore the components of $\vec{f}_i^{-1}$ be rounded to natural numbers.

The evaluation $\sigma\big(f_i^{-1}(y)\big)$ and the multiplication with $|(f_i^{-1}(y))'|$ can be represented as a single matrix-vector multiplication $A_i \cdot \vec{\sigma}$. The x-th component of $\vec{\rho}_{t+1,m}$ is the sum of two products $\vec{a}_{ix} \cdot \vec{\sigma} = \sum_{j=1}^{r} a_{ixj}\sigma_j$ where $a_{ixj} \neq 0 \iff f_{ij}^{-1} = x$. Accordingly a_{1x} has the general form $(*, \ldots, *, 0, 0, 0, \ldots, 0)$ where elements with lower index are greater than zero for small x and elements with higher indices (actually still lower than the half of the index range) are greater than zero if x is large. a_{2x} has the structure $(0, \ldots, 0, *, *, *, \ldots, *)$ vice versa. The multiplication with the x-th component of $|(\vec{f}_i^{-1})'|$ implies that rows a_{ix} are more weighted for very small and very large x. Together the matrix $F = A_1 + A_2$ has the same structure as in Equation 5.4.16.

In either case the matrix F can be thought of as a (local) transition matrix of the stochastic cellular automaton. By multiplication with a diffusion matrix (from left and right), a smother version of F can be obtained.

We can avoid the use of a transition tensor because Σ aggregates the random states of the neighbourhood (respectively their multivariate distribution) into a one-dimensional distribution.

($\mathcal{K}$) The deterministic update rule $\hat{\mathcal{F}} : \big([0,1]^r\big)^9 \to [0,1]^r$ is given by

$$(\vec{\rho}_{t,n})_{n\in\mathcal{N}(m)} \mapsto F(G \cdot \vec{\rho}_{t,m}) \cdot \Big(\bigoplus_{n\in\mathcal{N}(m)\setminus\{m\}} \vec{\rho}_{t,n} \Big) \tag{5.4.18}$$

During simulations (see Listing A.2) with this model/system, we can observe the same patterns as for the continuous-state version. A dynamic visualisation (compare

Figure 5.8) shows that the variation of the matrix $F(G \cdot \vec{\rho}_{t,m})$ and also its eigenvectors is marginal. This corresponds to the conclusions already made in context of Equation 5.4.12.

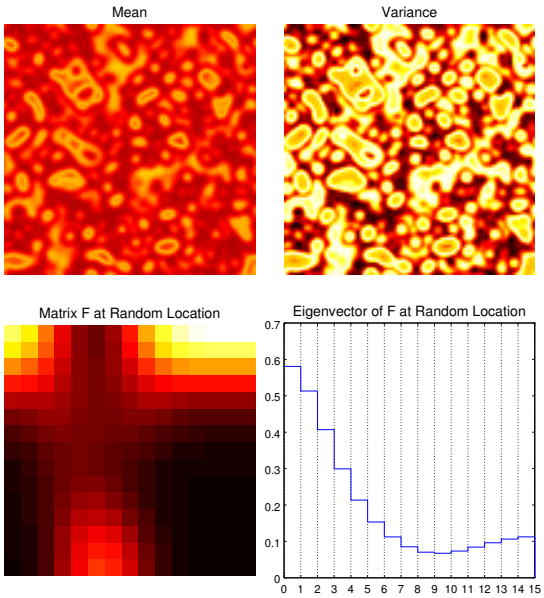

Figure 5.8: Iteration of discrete distributions as of Equation 5.4.18. The images on the top show the mean and variance of the random states on a lattice of 100 by 100 cells after 50 iterations. The discrete distributions of the random states are represented as vectors in $[0,1]^{15}$, which means that the state-space $[0,1]$ has been discretised into 15 compartments. The bottom-left depicts a visualisation of (a square version) of the transition matrix F at a random location on the lattice (brightness indicates greater weight). The bottom-right figure shows the eigenvector ($\in [0,1]^{15}$) corresponding to the eigenvalue 1 of the stochastic "transition" matrix F.

Furthermore (compare Figure 5.9) we can observe that the distribution of Σ is – corresponding to the central limit theorem – bell-shaped and that its only evident variation consists in different locations of its peak, which is yet always concentrated around the middle.

For the distribution of random states there exist two characteristic shapes. The left plot in Figure 5.9 shows the distributions of a large number of randomly chosen cells – the first type of distribution concentrates its mass in a certain region of lower indices and the second concentrates is mass in the very high indices ("alive").

From a comparison of the matrix F and its main eigenvector depicted in Figure 5.8 and the distribution of Σ in Figure 5.9 we can see that a lower expectation value $\mathbb{E}[\Sigma]$ yields a higher chance for the cell for "being alive" in the next step etc.

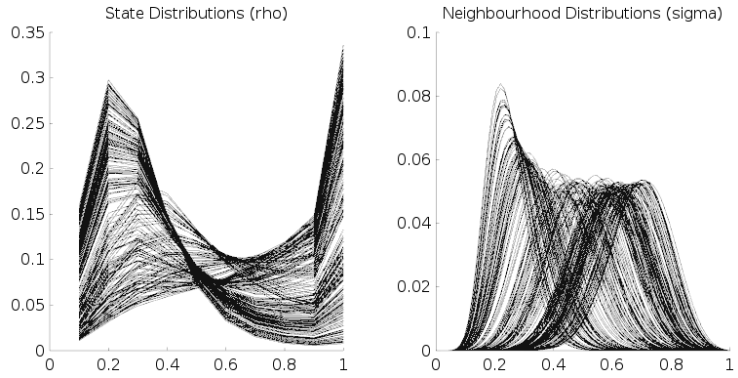

Figure 5.9: The left plot shows the state-distribution vectors $\vec{\rho}$ of a large number of randomly chosen cells at different iterations. The right figure shows a large number of distribution vectors $\vec{\sigma}$ of Σ.

5.4.3.1 Parametrised Stochastic Formulation

This final section concerns a stochastic formulation with random states in $\mathbb{S} = \{0, 1\}$. Accordingly a random state is always Bernoulli-distributed with a parameter $\theta_{t,m} \in [0, 1]$. Corresponding to the previous section the distribution of a random state can also be described by a vector $\vec{\rho}_{t,m} = (1 - \theta_{t,m}, \theta_{t,m})^\top \in [0, 1]^2$.

Since the distribution of Σ is a convolution of k probability vectors from $[0, 1]^2$, $\vec{\sigma}$ is a vector in $[0, 1]^{N+1}$.

Analogously to Equation 5.4.16 there exists a "column-stochastic" matrix $F \in [0, 1]^{2\times(N+1)}$ such that

$$\vec{\rho}_{t+1,m} = \begin{pmatrix} 1-\theta_{t+1,m} \\ \theta_{t+1,m} \end{pmatrix} = \begin{pmatrix} (1,\ldots,1)-\vec{f}_t^\top \\ \vec{f}_t^\top \end{pmatrix} \vec{\sigma}_{t,m}. \tag{5.4.19}$$

The distribution of Σ is a generalised binomial distribution (also known as Poisson binomial distribution) with

$$\sigma_{t,m,i} = \sum_{\substack{|A|=(i-1) \\ A\subseteq\mathcal{N}(m)}} \left[\prod_{n\in A} \theta_{t,n} \prod_{l\in A^c} \left(1 - \theta_{t,l}\right) \right] \tag{5.4.20}$$

where A are subsets of $\mathcal{N}(m)$ containing $(i - 1)$ elements.

Together with $\vec{f} = \theta_{t,m}\vec{f}_a + (1 - \theta_{t,m})\vec{f}_d$ we have

$$\theta_{t+1,m} = \left(\theta_{t,m}(\vec{f}_a - \vec{f}_d)^\top + \vec{f}_d^\top\right)\vec{\sigma}_{t,m}. \tag{5.4.21}$$

Obviously Equation 5.4.21 can be interpreted as an iteration of deterministic states. This update function is conjugate to the pseudo-stochastic formulation in Equation 5.4.4.

Chapter 6

Summary, Conclusions, Outlook

To conclude this thesis, this final chapter provides a summary of the presented and developed concepts. Additionally a discussion on possible future investigations covers some ideas that did not make it into the final version or would be very interesting to work on in the first place.

6.1 Overview of Cellular Automaton Concepts

The basic idea and general perception of cellular automata was analysed in the introduction. A textual description was set up to summarise the most characteristic features of a prototype of a cellular automaton.

T	topology I	topology II	$\mathbb{S}$	$\mathfrak{S} \subseteq \mathbb{S}^M$
iteration	regular arrangement	based on alignment		

Table 6.1: Textual description of the prototype of cellular automata.

The update rules map the states of a neighbourhood onto a new state independently of the location of the neighbourhood and without any restrictions on the type of mapping. Also the set of states is not restricted in any way. It was solely discussed that the local update rules must be consistent and defined in a way such that there cannot arise state configurations for which the update rules are not defined.

6.1.1 Basic Definitions of Cellular Automata

From this starting point the most intuitive (but also complete) definition approach was developed in a rather straight forward fashion. The most significant characteristic of the approach used in Section 2.1 to define ordinary cellular automata is clearly the alignment of cells on a regular lattice in $\mathbb{Z}^d$ and the use of this alignment in order to implicitly define a regular neighbourhood structure. This index-based

approach is also the only case for which boundary conditions of cellular automata were introduced.

The second more abstract definition approach of unaligned cellular automata in Section 2.2, which from a logical point of view should have come before the index-based approach, neglects the alignment of cells and features an explicitly defined neighbourhood mapping.

Based on a clear differentiation between the alignment of cells (first topological feature) and the construction of neighbourhoods (second topological feature) the third chapter provides different mathematical concepts for the description of the topology of cellular automata.

T	topology I	topology II	$\mathbb{S}$	$\mathfrak{S} \subseteq \mathbb{S}^M$
$\mathbb{N}$	$\mathbb{Z}^d$	J		
$\mathbb{N}$	V	N		
$\mathbb{N}$		$\mathcal{N}$		
$\mathbb{N}$		G		

Table 6.2: The two basic concepts for defining cellular automata are distinguished by the presence of an alignment of cells. The first two rows represent ordinary cellular automata in the index-based approach with a regular arrangement on $\mathbb{Z}^d$ and an implicit definition of neighbourhoods through a relative index tuple J as well as the vectorspace approach with an abstract alignment $V = M - M$ and vectorised neighbourhoods $m + N$ where $N \subset V$. The third and fourth row shows definition approaches for unaligned cellular automata by explicitly defined neighbourhood mappings $\mathcal{N}$ respectively a graphical structure $G = (M, E)$.

The update rules can – without any restrictions except compatibility – be arbitrary mappings from either a set or a tuple of states onto a new state.

6.1.2 Locally Characterised Evolution Systems

Since cellular automata are often used to model continuous-space and continuous-time systems, a big question always concerns the formalisation of space discretisations and the construction of update rules, which approximate the underlying continuous model in a sufficient manner. A conclusion from Chapter 4 is that cellular automata are closely related to strongly continuous semigroups on Banach spaces and that the local characterisation of update rules is a very special characteristic of cellular automata. Furthermore evolution equations in differential or integral form can often easily be translated into a cellular automaton formulation.

From a top-down perspective the global evolution operator of cellular automata $\mathcal{E} : \mathfrak{S} \to \mathfrak{S}$ can be decomposed into scalar components $\mathcal{E} = \prod_{m \in M} \mathcal{L}_m$. The second topological feature then implies that the mappings $\mathcal{L}_m$ actually only depend on

certain dimensions of their domain $\mathfrak{S} \subseteq \mathbb{S}^M$ namely $\mathbb{S}^{\mathcal{N}(m)}$. This approach is used for the definition of locally characterised evolution systems.

T	topology I	topology II	$\mathbb{S}$	$\mathfrak{S} \subseteq \mathbb{S}^M$
$\mathbb{R}_+$	topological vectorspace	$\mathcal{N}_{dt}$		Banach space

Table 6.3: Locally characterised evolution systems are basically semigroups on a Banach space $\mathfrak{S}$ with a certain local characterisation ($\mathcal{N}_{dt}$) of the evolution operators $\mathcal{E}_{dt}$.

The evolution operators $\mathcal{E}_{dt}$ form a semigroup and are constructed as a product of scalar evolution operators with a spatially independent component $\mathcal{F}$ in the form $\mathcal{L}_m = \mathcal{F} \circ \mathrm{proj}_{\mathcal{N}(m)}$. Since this formalism is rather abstract (compare Section 4.3.1), a special case of integral evolution operators is investigated in a more detailed fashion.

T	topology I	topology II	$\mathbb{S}$	$\mathfrak{S} \subseteq \mathbb{S}^M$
$\mathbb{R}_+$	$(M, \mathfrak{A}, \mu)$	$\mathcal{K}_{dt}$	Banach space	$L^1(M)$

Table 6.4: Locally characterised integral evolution systems evolve measurable functions from a measure space $(M, \mathfrak{A}, \mu)$ into a Banach space $\mathbb{S}$ using integral operators with kernels $\mathcal{K}_{dt}$.

The scalar evolution operators can explicitly be written as

$$\mathcal{L}_{dt,m}\mathcal{S} = \int_{\mathcal{N}_{dt}(m)} \mathcal{K}_{dt}(m,n)\mathcal{S}(n)\, d\mu(n). \tag{6.1.1}$$

In order for semigroup theory to be applicable and evolution equations to be a valid alternative formulation it is necessary that the global evolution operator is linear $\mathcal{E} \in L(\mathfrak{S}, \mathfrak{S})$, which is a very strong requirement but can be neglected for discrete time respectively be bypassed by observing quasi-linear systems.

Finally Chapter 4 also discusses discretisation approaches and outlines application scenarios for reaction-diffusion systems.

6.1.3 Definition of Stochastic Cellular Automata

Stochastic cellular automata can be formalised as stochastic processes. A formal approach regards the states of all cells in all iterative steps as random variables which together form a multiparameter stochastic process.

T	topology I	topology II	$\mathbb{S}$	$\mathfrak{S} \subseteq \mathbb{S}^M$
$G_1 \cong \mathbb{N}$		$G_2 \cong \mathcal{N}$	$(\mathbb{S}, \mathfrak{B})$ separable Hausdorff	$(\Omega, \mathfrak{A}, \mathcal{P}) \to$ $\to (\mathbb{S}, \mathfrak{B})$

Table 6.5: The definition of stochastic cellular automata as multiparameter stochastic processes does not explicitly define an update method but rather relies on the description of random states as random variables on a product-graph $G_1 \times G_2$.

The graphical structure $G = G_1 \times G_2$ is used to define stochastic dependencies through a filtration of σ-algebras in the underlying probability space $(\Omega, \mathfrak{A}, \mathcal{P})$ and through a specialised Markov property.

An equivalent iterative definition approach describes the update mechanism for random states through probability kernels.

T	topology I	topology II	$\mathbb{S}$	$\mathfrak{S} \subseteq \mathbb{S}^M$
$\mathbb{N}$		$\mathcal{N}$	$(\mathbb{S}, \mathfrak{B})$ separable Hausdorff	$(\Omega, \mathfrak{A}, \mathcal{P}) \to$ $\to (\mathbb{S}, \mathfrak{B})$

Table 6.6: At every iterative step each cell is associated with a random state.

A local "Markov" kernel $\mathcal{K}$ defines transition probabilities from the random states of a neighbourhood $\mathcal{N}(m)$ to a new random state for the actual cell m. With the help of local characterisations a global transition kernel can be defined as a product comparable to evolution systems. The resulting global stochastic process is a Markov process itself.

A complete mathematical formalisation is in both cases much more complicated than for deterministic cellular automata. Methods for reducing the complexity of the mathematical description are necessary. A rather theoretical application example concludes the section on stochastic cellular automata.

6.2 Outlook

Among the many aspects, that seem worth discussing but are out of the scope of this thesis, some shall be briefly mentioned in form of keywords:

- cellular automata with fuzzy states
- eigenvectors of global evolution operators and equilibrium states for stochastic cellular automata

Further topics that are not included in a sufficient manner in this thesis, should be discussed in more detail:

Nonlinear Evolution Systems. In Chapter 4 evolution systems with linear evolution operators were discussed. For a more general application scenario quasilinear evolution systems were introduced in Section 4.3.2. However the basic theory of semigroups is only applicable for linear evolution operators.

Nonlinear semigroups can for example be targeted using distortions of infinitesimal generators or fixpoint theorems for nonlinear maps. The theory of nonlinear differential equations also relies on these techniques and also leads back to abstract Cauchy problems.

Also for locally characterised evolution systems it should be possible to investigate nonlinear evolution operators. Especially for integral evolution systems a nonlinear evolution operator has the form (compare Equation 4.2.28)

$$\mathcal{E}_{dt}\mathcal{S}(m) = \mathcal{L}_{dt,m}\mathcal{S} := \int \mathcal{K}\big(dt, \mathcal{S}(n), n, m\big)\, d\mu(n). \tag{6.2.1}$$

Stochastic Neighbourhoods. A basic concept for stochastic neighbourhoods was presented in Section 4.3.3.5 as a discretisation method for integral evolution systems with L^1-normed kernels.

However, a stochastic neighbourhood may also be intrinsic to a model or system. In this case a random neighbour also causes a random input state for an update rule such that we ultimately deal with random states respectively stochastic cellular automata. Hence the formalisation of stochastic neighbourhoods should be compatible to the formalism of stochastic cellular automata.

Topology. In the context of Markov processes (and especially Markov chains) the iteration of distributions is a linear process. The local kernel of a discrete-state stochastic cellular automaton can be interpreted as a multilinear map and as a tensor.

Since also the second topological feature can be characterised by adjacency tensors, it would be interesting to find a common tensor representation of the neighbourhood structure and the update rule.

Lattice Boltzmann Method. The Boltzmann equation describes a distribution of particles on the space $T \times M \times V$. Correspondingly, every pair (t, m) is associated with a density and a probability distribution on V and a stochastic cellular automaton formulation of the Boltzmann equation is obvious.

A discretisation of M and V (and also T) yields a velocity model as described in Section 3.1.2. During the diffusion process every discrete velocity is associated with one of the neighbouring cells. The collision process is often modelled (Bhatnagar–Gross–Krook) as a linear approaching of a so-called equilibrium distribution.

A detailed formulation of the lattice Boltzmann method using the stochastic cellular automaton formalism would be a very interesting challenge.

Appendix A

Code Listings

Listing A.1: Deterministic Iteration of Distributions. Note that the computational effort in the second approach (iteration of parameters) is significantly lower since the loops are not nested.

```
global SIZE = 100;
global ITERATIONS = 100;
global MONTECARLO = 4000;

global p1 = rand;
global p2 = rand;
global p3 = 2*rand;
global p4 = 2*rand;
global p5 = rand;
global p6 = p1 + p2 + p3 + p4 + p5;

% generate normally distributed random values from parameters
function S = Norm(theta)
    global SIZE;
    S = randn(SIZE).*theta(:,:,2) + theta(:,:,1);
end

% (global) update function
function S = F(S)
    global SIZE p1 p2 p3 p4 p5 p6;
    S = ( p1*S(:,:) + p2*S([SIZE,1:SIZE-1],:) ...
       + p3*S([2:SIZE,1],:) + p4*S(:,[SIZE,1:SIZE-1]) ...
       + p5*S(:,[2:SIZE,1]) )/p6;
end

% (global) conjugate update function (parameter space)
function theta = Fhat(theta)
    global SIZE p1 p2 p3 p4 p5 p6;
    theta(:,:,1) = ( p1*theta(:,:,1) + p2*theta([SIZE,1:SIZE-1],:,1) ...
       + p3*theta([2:SIZE,1],:,1) + p4*theta(:,[SIZE,1:SIZE-1],1) ...
       + p5*theta(:,[2:SIZE,1],1) )/p6;
    theta(:,:,2) = ( p1*theta(:,:,2) + p2*theta([SIZE,1:SIZE-1],:,2) ...
       + p3*theta([2:SIZE,1],:,2) + p4*theta(:,[SIZE,1:SIZE-1],2) ...
       + p5*theta(:,[2:SIZE,1],2) )/p6;
end

% initial parameters
```

```
theta0 = cat(3,rand(SIZE)*0.1,rand(SIZE)*0.1);

A = zeros(SIZE);
B = zeros(SIZE);

% iterate multiple times with random initial conditions
for i = 1:MONTECARLO
    S = Norm(theta0);
    for t = 1:ITERATIONS
        S = F(S);
    end
    A = A + S;
end

% iterate the parameters
% and then generate a random global state multiple times
theta = theta0;
for t = 1:ITERATIONS
    theta = Fhat(theta);
end
for i = 1:MONTECARLO*10
    B = B + Norm(theta);
end

A = A/MONTECARLO;
B = B/MONTECARLO/10;
```

Listing A.2: Some excerpts from the discrete random state formulation of the "Game of Life". The code makes use of the *parallel* package for Octave.

```
% k is the number of neighbours excluding the local cell

alpha_stencil_8 = ...
    [   [   1   1   1   ];
        [   1   0   1   ];
        [   1   1   1   ]   ];
alpha_stencil_8 = alpha_stencil_8 / sum(sum(alpha_stencil_8));
[xx,yy,weight] = find(alpha_stencil_8);
alpha_stencil_8 = [xx,yy,weight,2*ones(size(xx,1),1)];

function Sigma = acc_neighbourhood (S, stencil)
    % for every cell accumulate (gather) the states
    % of neighbouring cells as vector
    % the result is a 4d matrix and the input must be a
    % 2d matrix for collection of states
    % or 3d matrix for collection of distributions
    % the stencil weights are ignored
    % the third dimension is used to accumulate distributions.
    % size of 3rd dimension then is r
    % size of 4th dimension is number of neighbours k
    Sigma = zeros(size(S,1),size(S,2),size(S,3),size(stencil,1));
    for i = 1:size(stencil,1)
        Sigma(:,:,1:size(S,3),i) = shift(...
            shift(S,stencil(i,1)-stencil(i,4),1),stencil(i,2)-stencil(i,4),2);
    end
end

function out = convolute (neighbourhood)
```

```
    % helper function to calculate convolution
    % first dimension of neighbourhood should have size r
    % output size of 3rd dimension is r + k*(r-1)
    out = vec(neighbourhood(:,:,:,1));
    for i = 2:size(neighbourhood,4)
        out = conv(out,vec(neighbourhood(:,:,:,i)));
    end
    out = permute(out,[3,2,1]);
end

function sigma = convolution_of_neighbouring_distributions (rho, stencil)
    % for every cell convolute the discrete distributions of
    % all neighbouring cells. the result is a r+(r-1)*(k-1) vector
    % where k is the number of neighbours.
    % 3rd dimension of result of acc_neighbourhood has size r
    % 4th dimension of result of acc_neighbourhood has size k
    sigma = acc_neighbourhood (rho, stencil);
    cells = mat2cell(sigma,ones(1,size(sigma,1)),...
                ones(1,size(sigma,2)),size(sigma,3),size(sigma,4));
    cells = parcellfun(4,@convolute,cells, ...
                "UniformOutput",false,'Vectorized',false,...
                'ChunksPerProc',size(sigma,1),'VerboseLevel',0);
    sigma = cell2mat(cells);
end

function B = transition_matrix_stochastic_linear ...
            (f, B1, b2, lambda, r, k = 0, r2 = r + (r-1)*(k-1))
    f = f/max(f);
    B = (B1 + b2*f').^(4/sqrt(lambda));
    B = B*diag(1./sum(B,1));
end

function rho_local = discrete_distribution_to_discrete_distribution ...
            (rho_local, sigma_local, transition_matrix, B1, b2, G, ...
            lambda, kappa1, kappa2, r, k = 0, r2 = r + (r-1)*(k-1))
    sigma_local = permute(sigma_local,[3,2,1]);
    rho_local = permute(rho_local,[3,2,1]);
    f = G*rho_local;
    F = diffusion_matrix(r,-1)^kappa1 * transition_matrix(f,...
            B1,b2,lambda,r,k,r2) * diffusion_matrix(r2,1)^kappa2;
    F = F*diag(1./sum(F,1));
    rho_local = F * sigma_local;
    rho_local = permute(rho_local,[3,2,1])/sum(rho_local);
end

function S = F (S, sigma, transition_type, transition_matrix, ...
                    lambda = 1.0, kappa1 = 0, kappa2 = 0, ...
                    r = 0, k = 0, r2 = r + (r-1)*(k-1))
    G = matrix_f_dead_alive(lambda,r,k,r2);
    [transition_matrix, B1, b2] = transition_matrix_prepare ...
                    (transition_matrix,lambda,r,k,r2);
    cells_sigma = mat2cell(sigma,ones(1,size(sigma,1)), ...
                    ones(1,size(sigma,2)),size(sigma,3));
    cells_S = mat2cell(S,ones(1,size(S,1)),ones(1,size(S,2)),size(S,3));
    cells_S = parcellfun(4,transition_type, ...
            cells_S,cells_sigma,{transition_matrix},{B1},{b2},{G},...
            {lambda},{kappa1},{kappa2},{r},{k},{r2}, ...
            "UniformOutput",false,'Vectorized',false,...
            'ChunksPerProc',size(S,1),'VerboseLevel',0);
    S = cell2mat(cells_S);
end
```

Bibliography

References

[1] P. Acquistapace and B. Terreni. "Some existence and regularity results for abstract non-autonomous parabolic equations". In: *Journal of Mathematical Analysis and Applications* 99.1 (1984).

[2] R. J. Adler and J. E. Taylor. *Random Fields and Geometry*. Springer Monographs in Mathematics. New York: Springer, 2007.

[3] O. Balci. "A life cycle for modeling and simulation". In: *Simulation: Transactions of the Society for Modeling and Simulation International* 88.7 (2012), pp. 870–883.

[4] V. Capasso and D. Bakstein. *An Introduction to Continuous-Time Stochastic Processes*. Modeling and Simulation in Science, Engineering and Technology. Boston: Birkhäuser, 2012.

[5] F. E. Cellier. *Continuous System Modeling*. Springer, 1991.

[6] J. Elstrodt. *Maß- und Integrationstheorie*. Vierte, korrigierte Auflage. Berlin: Springer-Verlag, 2005.

[7] K. J. Engel and R. Nagel. *One-Parameter Semigroups for Linear Evolution Equations*. Graduate Texts in Mathematics. New York: Springer-Verlag, 2000.

[8] L. C. Evans. *Partial Differential Equations*. Vol. 19. Graduate Studies in Mathematics. American Mathematical Society, 1998.

[9] A. M. Faden. "The Existence of Regular Conditional Probabilities: Necessary and Sufficient Conditions". In: *The Annals of Probability* 13.1 (1985), pp. 288–298.

[10] E. Fermi, M. Tsingou, J. Pasta, and S. Ulam. "Studies of nonlinear problems. Part I". In: *Los Alamos Scientific Laboratory report* LA-1940 (1955).

[11] A. Freno. "Statistical Machine Learning and the Logic of Scientific Discovery". In: *Iris: European Journal of Philosophy and Public Debate* 1.2 (2009), pp. 375–388.

[12] N. Ganguly, B. K. Sikdar, A. Deutsch, G. Canright, and P. P. Chaudhuri. *A Survey on Cellular Automata*. Tech. rep. 2003.

[13] J. A. Goldstein. *Semigroups of Linear Operators and Applications*. Oxford Mathematical Monographs. New York: Oxford University Press, 1985.

[14] F. F. Gonçalves, M. do Rosário Grossinho, and E. Morais. "Discretisation of abstract linear evolution equations of parabolic type". In: *Advances in Difference Equations* 14 (2012).

[15] G. Grimmett. *Robability on Graphs: Random Processes on Graphs and Lattices*. IMS Textbooks. Cambridge: Cambridge University Press, 2010.

[16] J. M. Hammersley and P. Clifford. *Markov Fields on Finite Graphs and Lattices*. Tech. rep. 1971.

[17] N. Jacob and M. Schicks. "Multiparameter Markov Processes: Generators and Associated Martingales". In: *Revue Roumaine de Mathematiques Pures et Appliquees* 55.1 (2010), pp. 27–34.

[18] T. Kato. "Abstract evolution equations of parabolic type in Banach and Hilbert spaces". In: *Nagoya Mathematical Journal* 19 (1961), pp. 93–125.

[19] D. Khoshnevisan. "Lecture Notes on Multiparameter Processes: Ecole Polytechnique Fédérale de Lausanne, Switzerland". 2001.

[20] D. Khoshnevisan. *Multiparameter Processes: An Introduction to Random Fields*. Springer Monographs in Mathematics. New York: Springer-Verlag, 2002.

[21] R. Kindermann and J. L. Snell. *Markov Random Fields and Their Applications*. Vol. 1. Contemporary Mathematics. Providence: American Mathematical Society, 1980.

[22] U. Knauer. *Algebraic Graph Theory: Morphisms, Monoids and Matrices*. Studies in Mathematics. Berlin: De Gruyter, 2011.

[23] D. Leao Jr., M. Fragoso, and P. Ruffino. "Regular Conditional Probability, Disintegration of Probability and Radon Spaces". In: *Proyecciones Journal of Mathematics* 23.1 (2004), pp. 15–29.

[24] I. N. N. Friedman M. Linial and D. Pe'Ee. "Using Bayesian Networks to Analyze Expression Data". In: *Journal Of Computational Biology* 7.3/4 (2000), pp. 601–620.

[25] J. von Neumann. "John von Neumann: Collected Works". In: ed. by A. H. Taub. Vol. V: Design of Computers, Theory of Automata and Numerical Analysis. New York: Pergamon Press, 1961. Chap. The general and logical theory of automata (1951).

[26] J. von Neumann. *Theory of Self-Reproducing Automata*. Ed. by A. W. Burks. Essays on Cellular Automata. University of Illinois Press, Urbana and London, 1966.

[27] K. D. Schmidt. *Maß und Wahrscheinlichkeit*. Berlin Heidelberg: Springer-Verlag, 2009.

[28] C. R. Shalizi and A. Kontorovich. "Almost None of the Theory of Stochastic Processes". Lecture Notes. 2010.

[29] K. Taira. *Semigroups, Boundary Value Problems and Markov Processes*. Springer Monographs in Mathematics. Berlin: Springer-Verlag, 2004.

[30] H. Tanabe. "A class of the equations of evolution in a Banach space". In: *Osaka Mathematical Journal* 11 (1959), pp. 121–145.

[31] "The Publications of Stanislaw M. Ulam". In: *Los Alamos Science* 15 (1987), pp. 313–317.

[32] D. A. Wolf-Gladrow. *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction*. Lecture Notes in Mathematics 1725. Berlin: Springer, 2000.

[33] S. Wolfram. "Statistical Mechanics of Cellular Automata". In: *Reviews of Modern Physics* 55.3 (1983), pp. 601–644.

[34] H. van Zanten. "An Introduction to Stochastic Processes in Continuous Time". Lecture Notes. 2004.

Application Examples

[35] M. Bicher. "Agentenbasierte Modellbildung und Simulation auf Basis der Fokker-Planck-Gleichung". MA thesis. Vienna University of Technology, 2013.

[36] M. Bruckner, S. Tauböck, and N. Popper. "Orientation inside of a pedestrian flow simulation". In: *ABSTRACTS ASIM TCSE 2012 Vienna*. Ed. by F. Breitenecker and S. Tauböck. Vol. 37. ARGESIM Report. Vienna, Austria: ARGESIM / ASIM, 2012.

[37] M. Eden. "A two-dimensional growth process". In: *Berkeley Symposium on Mathematical Statistics and Probability*. University of California Press, Berkeley, 1961, pp. 223–239.

[38] *Kunstkammer, Kunsthistorisches Museum Wien*.

[39] M. J. North and C. M. Macal. *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*. New York: Oxford University Press, 2007.

[40] D. O'Sullivan and G. L. W. Perry. *Spatial Simulation: Exploring Pattern and Process*. Wiley, 2013.

[41] D. Richardson. "Random growth in a tessellation". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 74.03 (1973), pp. 515–528.

[42] G. Romstorfer, S. Parragh, G. Schneckenreither, M. Landsiedl, P. Einzinger, and M. Scheuringer. "Integration of GIS Data in Health Care Utilization". Englisch. In: *Tagungsband Abstracts und Fullpapers*. Ed. by R. Bödi and W. Maurer. ZHAW, Winterthur, Schweiz: Pabst Science Publishers, 2011.

[43] G. Schneckenreither, N. Popper, and F. Breitenecker. "Modellierung von Evolutions Systemen mit Stochastischen Zellulären Automaten". In: *Tagungsband ASIM 2014, 22. Symposium Simulationstechnik*. Ed. by J. Wittmann and C. Deatcu. Vol. 43. ARGESIM Reports. Wien: ARGESIM/ASIM, 2014.

[44] G. Schneckenreither, N. Popper, and F. Breitenecker. "Modelling SIR-type epidemics by ODEs, PDEs, difference equations and cellular automata – A comparative study". In: *Simulation Modelling Practice and Theory* 16.8 (2008), pp. 1014–1023.

[45] R. Viertl. *Statistical Methods for Fuzzy Data*. Wiley, 2011.

[46] B. Walsh. "Markov Chain Monte Carlo and Gibbs Sampling". Lecture Notes. 2004.

[47] N. Wiener and A. Rosenblueth. "The mathematical formulation of the problem of conduction of impulses in a network of connected excitable elements, specifically in cardiac muscle". In: *Arch. Inst. Cardiol. Mexico* 16 (1946), pp. 205–265.

[48] *Wikipedia, the Free Encyclopedia*. Online Encyclopedia: `www.wikipedia.org`.

[49] T. Williams and R. Bjerknes. "Stochastic model for abnormal clone spread through epithelial basal layer". In: *Nature* 236 (1972), pp. 19–21.

[50] S. Wolfram. *A New Kind of Science*. 2002.

[51] K. Zuse. *Rechnender Raum*. Friedrich Vieweg und Sohn, 1969.