**Simulation vernetzter Systeme mit Hilfe von zellulären   
probabilistischen Automaten**

„Vergleich verschiedener Systeme zur Simulation von Interaktionen in Zellsystemen.“

eingereicht von

Martin Krockert

Robert Erzgräber

Tag der Themenübergabe: 20.03.2016

Tag der Einreichung: 31.07.2016

Inhaltsverzeichnis

[Inhaltsverzeichnis 1](#_Toc454472334)

[Abbildungsverzeichnis 3](#_Toc454472335)

[Abkürzungsverzeichnis 4](#_Toc454472336)

[Tabellenverzeichnis 5](#_Toc454472337)

[1 Einleitung 6](#_Toc454472338)

[2 Grundlagen 7](#_Toc454472339)

[2.1 Modelle 7](#_Toc454472340)

[2.2 Das Geistige Modell 9](#_Toc454472341)

[2.3 Zelluläre Automaten 10](#_Toc454472342)

[2.4 Modell auf Basis von Graphen 13](#_Toc454472343)

[Problem der Netzerstellung 13](#_Toc454472344)

[3 Umsetzung in Morpheus 17](#_Toc454472345)

[3.1 Informationen zu Morpheus und dessen System 17](#_Toc454472346)

[3.2 Überlegungen zum Plug-In 19](#_Toc454472347)

[3.2.1 Habitatseigenschaften 19](#_Toc454472348)

[3.2.2 Zellsystem 20](#_Toc454472349)

[3.3 Erweiterungsmöglichkeiten 20](#_Toc454472350)

[4 Prototypische Umsetzung 21](#_Toc454472351)

[5 Auswertung 22](#_Toc454472352)

[5.1 Beobachtung für CA Model im symmetrischen Gitternetz 22](#_Toc454472353)

[5.2 Beobachtung für Graphen 22](#_Toc454472354)

[5.3 Beobachtungen bei Morpheus 23](#_Toc454472355)

[6 Ausblick 25](#_Toc454472356)

[Anlagenverzeichnis 26](#_Toc454472357)

[Anlagen 27](#_Toc454472358)

[Literaturverzeichnis 30](#_Toc454472359)

Abbildungsverzeichnis

[Abbildung 2.3‑1: Zellulärer Automat 11](#_Toc454472374)

[Abbildung 2.4‑1: Wachsender Graph in erster und zweiter Generation 14](#_Toc454472375)

[Abbildung 2.4‑2: Erdős-Rényi Zufallsgraph 15](#_Toc454472376)

[Abbildung 2.4‑3: Zufallsgraph mit Force Atlas 2 15](#_Toc454472377)

[Abbildung 2.4‑4: Umkreisbedingung 15](#_Toc454472378)

[Abbildung 2.4‑5: Beispiel eines Zufälligen Punktnetzes 16](#_Toc454472379)

[Abbildung 2.4‑6: Voronoi Zellen 16](#_Toc454472380)

[Abbildung 2.4‑7: Delaunay- und Voronoi Zellen 16](#_Toc454472381)

[Abbildung 5.1‑1: Torus 22](#_Toc454472382)

[Abbildung 5.2‑1: Sweep Line Fehler 23](#_Toc454472383)

Abkürzungsverzeichnis

CA = Cellular Automat

CGM = Cellular Graph Modell

CPM = Cellular-Potts-Model

GUI = Graphical User Interface

Tabellenverzeichnis

[Tabelle 2.1‑1: Modellbildung 8](#_Toc454472384)

# Einleitung

In der Biologie spielen Wachstumsvorgänge eine zentrale Rolle. Nahezu jedes Leben basiert auf dem einfachen Konzept der Zellteilung und Bewegung. Die Wissenschaft hat zur Modellierung und Simulation von Wachstum unterschiedliche Herangehensweisen hervorgebracht. In dieser Arbeit werden drei Modelle miteinander verglichen, die diese Wachstumsprozesse der Natur repräsentieren, um eine Aussage über die Qualität der Modelle zu erhalten. Bei diesem Vergleich wird von der Vorstellung ausgegangen, dass sich Zellen innerhalb eines Systems teilen und in eine Richtung bewegen können, sofern ihnen genügend Platz zur Verfügung steht. Auf dieser Basis wird zunächst ein gedankliches Modell des Wachstumsvorganges beschrieben. Darauf aufbauend wird das Modell in die mathematischen Modellkonzepte

* „*Cellular Potts Model*“[[1]](#footnote-1),
* „*Cellular Automata Model*“[[2]](#footnote-2) und
* „*Cellular Graph Model*“[[3]](#footnote-3)

überführt. Die mathematische Beschreibung der Modelle ermöglicht eine Umsetzung der Systeme in einer Softwarelösung zur Simulation des Zellverhaltens. Die Berechnungen für den Vergleich der Systeme wird unter Verwendung der Simulationssoftware „*Morpheus*“ durchgeführt. Morpheus in der aktuellen Version 1.9.1 setzt das CPM bereits vollständig um und wird daher in dieser Arbeit nicht näher betrachtet.

# Grundlagen

## Modelle

Modelle sind abgegrenzte, verständliche und analysierbare Abbilder von komplexen Zusammenhängen aus der Wirklichkeit. Je nach Beschreibung und Bestandteilen werden diese Modelle in drei Kategorien unterteilt. In geistige-, physikalische- und mathematische Modelle, siehe Tabelle 2.1‑1. Die drei Modell-Typen bauen jeweils aufeinander auf. Im Prozess der Modellerstellung wird eine Problemstellung zunächst verbal beschrieben. Dabei wird ein reales Problem anhand seiner Eigenschaften analysiert, das biologische und physikalische Verhalten beschrieben und zum Abschluss in ein mathematisches Modell überführt. Somit kann der gesamte Erstellungsprozess des Modells nachvollziehbar strukturiert werden, von der Idee zum geistigen Modell des Zellwachstums über die biologischen Beobachtungen, bis hin zur mathematischen Beschreibung der einzelnen Modelle (CPM, CA, CGM).

In dieser Arbeit wird auf Basis des geistigen Modells eine Beschreibung der mathematischen Modelle erstellt. Dabei werden auch die Unterschiede und Gemeinsamkeiten der Modelle untersucht. Die bei der Modellerstellung aufkommenden Fragen wie ein bestimmtes Verhalten beschrieben werden kann, machen das Modell immer detaillierter. Im letzten Schritt wird das Modell mit seinen Eigenschaften und seinem Verhalten, in ein mathematisches System überführt, welches wiederum mittels geeigneter Software abgebildet und Simuliert werden kann.

Tabelle ‑: Modellbildung

Quelle: TU BRAUNSCHWEIG (2002)

|  |  |  |  |
| --- | --- | --- | --- |
| Modelltyp | Geistige | Biologisch/Physikalisch | Mathematische |
| Modell | eine verbale Beschreibung des Systemverhaltens | eine Reproduktion eines real vorliegenden Systems | eine Verknüpfung von Funktionen, Regeln, Beziehungen |
| Besteht aus | Prämissen, Schlüssen (Syllogismen) | einem physikalischen, materiellen Gebilde. | mathematischen Funktionen, Variablen (Zustandsgrößen). |
| Plausibilität | die Schlussweise überprüft. | das Verhalten unter bekannten Zuständen analysiert. | das Modellverhalten mittels Stabilitäts-, Sensitivitätsanalyse untersucht. |
| Simulation | ein Gedankenexperiment. | eine physikalische Messung unter vorgegebenen Randbedingungen. | der Ablauf eines Programmes, eines Regelwerkes unter gegebenen Anfangsbedingungen. |

Anhand des in der Aufgabe beschriebenen Wachstumsvorganges wird im Folgenden erläutert, welche Besonderheiten jedes Modell mit sich bringt. Mit den Modellen soll die Zellbewegung und -teilung analysiert werden. In der verbalen Beschreibung des geistigen Modells wird zunächst die Frage geklärt, was dargestellt werden soll und welche Prozesse dafür benötigt werden. Auf Basis des geistigen Modells, lässt sich das biologische bzw. physikalische Modell, unter Einbeziehung der Gesetzmäßigkeiten der Realität, erstellen. Dabei werden vor allem die Eigenschaften überprüft, die durch das geistige Modell festgelegt wurden und hinsichtlich ihrer Machbarkeit analysiert. Das bedeutet, wenn das geistige Modell vorsieht, dass sich Zellen nach einem bestimmten Muster bewegen können, so muss mittels biologischer Prozesse nachgewiesen werden, dass das Bewegungsmuster korrekt ist. Meist werden dabei Spezifikationen des Modells angepasst, da die Analyse der Prozesse im Detail Änderungen nötig werden lässt. Somit entstehen häufig neue Komponenten beziehungsweise Eigenschaften, die bisher noch nicht berücksichtigt worden sind und mit einbezogen werden müssen.

Die spezifischen Informationen aus der darstellbaren Realität und die Eigenschaften des Gedankenspiels werden im mathematischen Modell zusammengeführt, das bedeutet die Zellen werden vereinheitlicht, weswegen sie nur noch eine Bewegung, eine Größe und die gleichen Eigenschaften haben. Des Weiteren werden die Grenzen des Systems, die Randbedingungen unter denen die Eigenschaften bestmöglich garantiert werden können, festgelegt. Die Randbedingungen in dem Versuch sind so definiert, dass ein Raster zur Ausbreitung, Bewegung und Zellaufteilung genutzt wird, welches jedoch unterschiedlich ausgelegt werden kann.

## Das Geistige Modell

Die grundlegenden Elemente für Wachstum und Bewegung können wie folgt beschrieben werden. Als Systemgrenze wird der zweidimensionale Raum definiert. Dieser ist mit einer Anzahl von Knoten gefüllt. Die Knoten besitzen in diesem Modell nur einen Zustand, „*belegt*“ oder „*nicht belegt*“. Im weiteren Verlauf der Arbeit sei der Zustand „*belegt*“ mit der Zahl 1 und der Zustand „*nicht belegt*“ mit 0 definiert. Der Zustand einer Zelle wird mit Spin bezeichnet. Jede dieser im System befindlichen Knoten besitzt eine Anzahl von Nachbarknoten. Der Spin und die Aktionen, die durch den aktuellen Knoten ausgeführt werden können, sind davon abhängig, welchen Spin die Nachbarknoten haben. So kann sich beispielsweise ein Knoten nur bewegen, wenn mindestens ein Nachbarknoten den Spin 0 innehat. Auf die Nachbarschaftsbeziehungen soll im weiteren Verlauf eingegangen werden. Neben diesen Nachbarschaftsbeziehungen hat jeder Knoten eigene Ambitionen sich mit einer gewissen Wahrscheinlichkeit zu bewegen oder zu teilen.

Der Raum aller Gitter definiert sich folgendermaßen:

Der Spin jeder Zelle x im Gitter S kann entweder 0 oder 1 annehmen.

* Eine Zustandsänderung von 0 zu 1 geschieht mit einer Wahrscheinlichkeit .
* Eine Zellbewegung geschieht mit einer Wahrscheinlichkeit .
* Eine Zellteilung geschieht mit einer Wahrscheinlichkeit .

Eine Zellbewegung bedeutet in diesem Fall, dass der Spin von Knoten A von 1 auf 0 und bei genau einem Nachbarknoten der Spin von 0 auf 1 gesetzt wird. Die Zellteilung hingegen besagt, dass der Spin von Knoten A unverändert bei 1 bleibt und bei genau einem Nachbarknoten der Spin von 0 auf 1 sich ändert. In beiden Fällen besteht für die Aktion die Voraussetzung, dass mindestens ein Nachbarknoten mit dem Wert 0 vorhanden ist.

Die Zustandsänderung eines Knotens wird mit Flip beschrieben, der Austausch zweier Knoten mit Exchange. Die Unterscheidung von Flip und Exchange wird vorgenommen, da zwei unterschiedliche Aktionen im System passieren. Beim Flip wird nur ein Knoten betrachtet und es werden keine Änderungen in ihrem Nachbarschaftsbereich vorgenommen. Beim Exchange hingegen hat die Aktion direkten Einfluss auf zwei Knoten und indirekt zusätzlichen Einfluss auf deren Nachbarschaftsbereiche.

Somit führt der Knoten, mit der Wahrscheinlichkeit von keine Aktion aus. Zu jedem Zeitpunkt wird für jeden Knoten im System eine zufällige Wahrscheinlichkeit ermittelt, die die Aktion des Knoten bestimmt. Dieser Wahrscheinlichkeitswurf ist für alle Knoten im System gültig und wird jede Zeiteinheit für alle Knoten wiederholt.

## Zelluläre Automaten

Zur Erstellung eines Modells zur Beschreibung von Zellsystemen, müssen diese verallgemeinert werden. Dabei muss das allgemeine Zellmodell für das vorliegende Problem angepasst werden. Der erste Ansatz ist es die beschriebene Situation mittels eines zellulären Automaten darzustellen. Dabei werden die Ortskoordinaten der Zellen auf ein Gitter projiziert. Vorlage hierfür ist dabei der von John Horton Conway beschriebene Automat – „Game of Life“. Im beschriebenen Automaten von John Horten Conway, kann jeder Knoten nur zwei Zustände besitzen und verschiedene Aktionen ausführen, die vordefinierten Regeln unterliegen. Diese Regeln sind dabei auf das zu beschreibende System dynamisch anpassbar. Ebenfalls ist es möglich einem Knoten weitere Eigenschaften zu geben, womit andere Zelltypen bzw. spezielle Aktionen realisiert werden können. Im geistigen Modell sind Knoten in der Lage sich zu teilen oder die Position mit einem Nachbarknoten zu tauschen. Eine weitere zu betrachtende Eigenschaft ist die Zellteilung. Diese lässt sich aus zwei unterschiedlichen Sichtweisen betrachten. Aus Sicht des ausführenden Knotens ist es eine einzelne Aktion, die unabhängig von Flip und Exchange wirkt, aus Sicht des freien Feldes kann es als Flip bezeichnet werden. Da der Spin des ausführenden Knotens nicht verändert wird und damit auch die Nachbarschaftsbeziehungen unangetastet bleiben fällt im weiteren Verlauf dieser Arbeit diese Aktion unter die Kategorie des Flips.

1.

2.

3.a

3.b

Das Zellsystem besteht aus einer Menge von Knoten und deren Beziehung untereinander, wobei der Zustand des gesamten Zellsystems zeitabhängig ist. Jeder Knoten besitzt dabei eine Koordinate, die durch i- und j-Werte repräsentiert werden, sowie ihrem Spin. Die Beziehungen untereinander können mittels einer Matrix beschrieben werden, die alle Knoten und deren Aktion im aktuellen Zeitintervall beinhaltet.

Das Modell besteht aus folgenden Bestandteilen:

* Randbedingungen
* Zellsystem
* Anfangszustand

Die Randbedingungen sind die Regeln, die aus den Gegebenheiten des Modells entstehen und beachtet werden müssen. Darunter fällt das zugrundeliegende Gitter, welches folgendermaßen beschrieben werden kann:

Damit kann der Zustand wie folgt dargestellt werden:

.

Die jeweilige Position im System ist somit nur noch ein Wert, der entweder 0 – „*Knoten nicht belegt*“ – oder 1 – „*Knoten belegt*“ – annimmt.

Die Dynamik des Systems ist durch eine Funktion definiert, die den aktuellen Spin des Bezugsknotens und dem aktuellen Spin der benachbarten Knoten einem neuen Spin dem Bezugsknoten zuordnet. Bei einem deterministischen zellulären Automaten wird jedem möglichen neuen Spin eine Wahrscheinlichkeit zugeordnet, mit der dieser Spin gewählt wird.

|  |
| --- |
|  |

Abbildung ‑: Zellulärer Automat

Quelle: in Anlehnung an: TU BRAUNSCHWEIG (2002)

Das in Abbildung 2.3‑1 dargestellte System, beschreibt einen Automaten mit einer Neumann Nachbarschaft. Bei dieser werden für den Aktualisierungszyklus nur die direkten Nachbarn einbezogen. Nehmen alle acht Nachbarknoten Einfluss auf den Aktualisierungsvorgang, spricht man von einer Moore Nachbarschaft. Ein wichtiger Aspekt bei der Betrachtung, ist die Realisierung des Aktualisierungsvorgangs für das System.

Hierfür gibt es zwei Vorgehensweisen:

1. Globale Aktualisierung:
   * Vorberechnung des Spins für alle Knoten in
   * Gleichzeitige Aktualisierung aller im System befindlicher Knoten
2. Lokale Aktualisierung:
   1. Auswahl eines zufällig selektierten Knotens
   2. Bestimmung ihres Spins in
   3. Aktualisierung des Knotens

* Wiederholung der Schritte 1. – 3. (n + 1)\*(m + 1) mal

Beide Systematiken haben ihre Vor- und Nachteile. Während die globale Aktualisierung aktuelle Änderungen nicht mit einbezieht, entsteht für die lokale Aktualisierung ein immenser zusätzlicher Rechenaufwand, indem eine Änderung pro Knoten vorgenommen und bei jedem Schritt neu bestimmt werden muss. Die Methoden sind nur bedingt vergleichbar, da Änderungen der Nachbarknoten unterschiedlichen Einfluss auf das gesamte System haben. Im ersten Fall werden alle Knoten in der Matrix genau einmal aktualisiert. Im zweiten Fall kann es vorkommen, dass ein Knoten gar nicht bis mehrmals einen Aktualisierungsvorgang durchläuft. Die lokale Aktualisierung muss mindestens so oft durchgeführt werden, wie es Knoten im System gibt, daraus ergibt sich ein sogenannter „Monte-Carlo-Step“. Die lokale Aktualisierung verhindert ein örtliches Überschreiben der Knoten und schließt somit Fehler bei der Aktualisierung aus. Dies wird im weiteren Verlauf der Arbeit präferiert.

## Modell auf Basis von Graphen

Während der Gestaltung des Modells für den zellulären Automaten, erschien die Anordnung der Knoten in einem symmetrischen Netz sehr unnatürlich für ein biologisches Zellsystem. Daraufhin entstand die Idee, ein Modell auf der Basis von ungerichteten Graphen zu erstellen. Hierbei definiert sich der Graph aus einer endlichen Menge Knoten (Vertices), welche die Zellen repräsentieren. Die Menge , deren Elemente Kanten (Edges) heißen, stellen die Nachbarschaftsbeziehungen zwischen den Knoten dar.

### Problem der Netzerstellung

Für die Umsetzung eines solchen Modells muss zunächst die Frage beantwortet werden, wie der Graphen mit seinen Knoten und Kanten erstellt werden kann, so dass eine netzartige Struktur entsteht. Analog zu der Struktur eines biologischen Zellsystems.

In initialer Idee sollte der Graph mit zunehmender Ausbreitung der Knoten wachsen. Im Gedankenexperiment erscheint dies trivial, in der Umsetzung ist es jedoch sehr aufwändig. Die Abbildung 2.4‑1 zeigt eine erste Umsetzung für ein Ausbreitung mit immer gleicher Anzahl an Nachbarknoten. Mit einer zufällig gewählten Anzahl von neuen Knoten werden die Abstände mit jeder Generation kleiner (siehe Abbildung 2.4‑2). Kein unlösbares Problem jedoch für dieses Forschungsprojekt ungeeignet.

|  |  |
| --- | --- |
| 1. **Generation** | **3. Generation** |

Abbildung ‑: Wachsender Graph in erster und Dritter Generation. Wobei mit Berührung des Randbereiches ein neuer Rand entsteht. In den Grafiken durch die ‚+‘ gekennzeichnet.

Quelle: Eigene Darstellung

|  |
| --- |
|  |

Abbildung ‑: Wachsender Graph mit zufälliger Anzahl Nachbarn

Quelle: Eigene Darstellung

Die zweite Idee basiert auf dem von Erdős-Rényi definierten Zufallsgraphen[[4]](#footnote-4). Dessen Dichte d sich aus

ergibt. Wobei *n* die Anzahl der Vertices angibt und

die erwartete Anzahl der Kanten bezeichnet. Mit wird die Wahrscheinlichkeit bezeichnet, mit der der Knoten eine Kante zu einem zufällig gewählten Knoten bildet. Dieses Vorgehen realisiert einen vollständig zufälligen Graphen. Jedoch überlagern sich bei diesem Vorgehen die einzelnen Kanten gegenseitig, das hat zur Folge, dass sich die Anzahlen der direkten Nachbarn nicht durch ihre Lokalität beschränken lässt. Auch der Versuch durch die Minimierung der Kantenlänge mit dem Force Atlas 2 Algorithmus brachte daher keine Verbesserung hinsichtlich dieser Eigenschaft (vgl. Abbildung 2.4‑3 und Abbildung 2.4‑4).

|  |  |
| --- | --- |
|  |  |
| Abbildung ‑: Erdős-Rényi Zufallsgraph  Quelle: Eigene Darstellung. | Abbildung ‑: Zufallsgraph mit Force Atlas 2  Quelle: Eigene Darstellung |

Die geeignetste Lösung ist jedoch, Punkte zufällig in einem Koordinatensystem zu verteilen und die jeweiligen Nachbarn, unter Beachtung der Delaunay Bedingung miteinander zu verbinden. Die Delaunay Bedingung besagt, dass der Umkreis eines Dreiecks keine weiteren Punkte halten darf (siehe Abbildung 2.4‑5).

|  |
| --- |
| http://www.karlchenofhell.org/cppswp/bildchen2.png |

Abbildung ‑: Umkreisbedingung

Quelle: Eigene Darstellung

Ein auf diese Weise erstelltes Netz maximiert die Innenwinkel der im Netz erstellten Dreiecke. Abbildung 2.4‑6 veranschaulicht ein solches Netz. Diese Systematik wird vorwiegend in der Finiten-Elemente-Methode verwendet, welche zur Festigkeitsanalyse von Materialen dient. Ein sehr effizienter Algorithmus zur Berechnung eines solchen Netzes wurde von Steven Fortune entwickelt. Der sogenannte „Sweep Line“-Algorithmus hat einen Rechenaufwand von und einen Speicherplatzbedarf von . Eine inkrementelle Erstellung des Netzes führt bei jedem Durchlauf zu einer erneuten Überprüfung der bereits erstellten Bestandteile des Netzes. Damit geht einher, dass der Bedarf an Rechenzeit exponentiell ansteigt. Daher ist die Verwendung der inkrementellen Berechnung für eine Simulation ungeeignet.

|  |
| --- |
|  |

Abbildung ‑: Beispiel der Delaunay Verbindung einer zufälligen Punktwolke.

Quelle: Eigene Darstellung

Das Netz kann nach Erstellung eins zu eins als Graph für alle weiteren Berechnungen übernommen werden. Ein weiterer Vorteil dieser Methode ist die Möglichkeit der Visualisierung der Knoten als Voronoi-Diagramm. Denn die Delaunay Nachbarschaftsbeziehungen sind identisch zu denen eines Voronoi-Diagrammes. Sie werden daher auch als „Dualer Graph“ bezeichnet (vgl. Abbildung 2.4‑7 und Abbildung 2.4‑8).

|  |  |
| --- | --- |
|  |  |
| Abbildung ‑: Voronoi-Zellen um eine zufällige Punktwolke.  Quelle: Eigene Darstellung. | Abbildung ‑: Duale Darstellung der Delaunay- und Voronoi-Zellen um eine zufällige Punktwolke.  Quelle: Eigene Darstellung |

# Umsetzung in Morpheus

Morpheus ist ein System, das aktuell von Joern Starruß und Walter de Back an der TU Dresden entwickelt wird. Das Programm befindet sich in ständiger Weiterentwicklung und ist als Open Source erhältlich.

## Informationen zu Morpheus und dessen System

Morpheus stellt einen Metainterpreter für Zellsysteme dar und soll Wissenschaftlern, Studenten und Forschern die Möglichkeit geben, Zellsysteme zu simulieren und vorhersagen zum Verhalten dieser treffen zu können. Das System wird unter der Maßgabe entwickelt, die Verwendung mit möglichst minimalem informationstechnischen Hintergrund zu ermöglichen. Dazu wurde eine eigene Markupsprache entwickelt, die leicht verständlich ist und vom Interpreter eigenständig in die vorgesehenen Modelle „übersetzt“ werden kann. Das System ist durch Plug-Ins erweiterbar, womit eigenständig Modelle eingepflegt und Morpheus an die eigenen Anforderungen angepasst werden kann. Um diese Anforderungen zu realisieren wurde Morpheus in drei Ebenen aufgeteilt:

|  |
| --- |
|  |

Abbildung 3.1 Zusammenhang der einzelnen Morpheus Komponenten.

Quelle: Eigene Darstellung

Die Core-Dateien beinhalteten alle grundlegenden Funktionalitäten des Systems. Das umfasst die Verarbeitung der Eingaben, das Steuern der Plug-Ins und deren Kommunikation sowie die gesamte Programmsteuerung. Diese Dateien werden in der Regel nicht angepasst. Die zweite Ebene sind die entsprechenden Plug-Ins, die entweder geladen werden oder selbst entwickelt werden können. Zur eigenständigen Entwicklung von Plug-Ins werden bereits vorgefertigte Dateien bereitgestellt, die grundlegende Funktionen beinhalten. Zu diesen gehören eine C-Datei und ein XML-Schema. Diese müssen durch den Programmierer dann an die Anforderungen des Systems angepasst werden. Über das XML-Schema wird der Datenaustausch zwischen den einzelnen Instanzen geregelt, sodass über das Plug-In auf Objekte von Morpheus zugegriffen werden kann. Die letzte Ebene enthält die GUI[[5]](#footnote-5) für den Benutzer. Über diesen ist es möglich das zu bearbeitende Programm einzugeben. Dabei wird die entwickelte Sprache genutzt, um dem Nutzer die Eingabe zu erleichtern. Des Weiteren können Plug-Ins eingebunden werden, die dann im Code des Users an entsprechenden Stellen eingefügt werden können.

Morpheus verwendet das CPM, um seine Systeme zu beschreiben. In diesem wird das Verhalten von Zellverbünden modelliert, denen ein Raster zugrunde liegt. Eine Zelle ist dabei ein Verbund aus Pixeln, die die gleiche Zell-Id[[6]](#footnote-6) bekommen. Pro Zeiteinheit wird ein Pixel aktualisiert und das Verhalten berechnet. Das Verhalten wird dabei durch vorgegebene Wahrscheinlichkeit bestimmt, die vom User festzulegen sind. Das Update der Zellen erfolgt in einem globalen Schritt.

## Überlegungen zum Plug-In

Wie im Kapitel 2.1 beschrieben, besteht das geplante System aus:

* Randbedingungen
* Zellsystem
* Anfangszustand

Im Folgenden wird auf die einzelnen Bestandteile eingegangen und wie diese mittels Morpheus umgesetzt werden können. Start- und Endzustände werden dabei nicht gesondert betrachtet. Der Startzustand ist eine vorgegebene Anzahl an Zellen in einem System, die per Hand oder per Zufall eingetragen werden können. Der Endzustand ist ebenfalls händisch durch eine Regel – bestimmte Anzahl an Zyklen, Zellen, etc. – anzugeben.

### Randbedingungen

Die zu erzeugenden Randbedingungen sind die folgenden:

* Gitter ist an den Rändern beschränkt
* Zellen bestehend aus einem einzigen Knoten

Morpheus verwendet in seinen Eigenschaften ein CPM, womit ein wie in Abbildung 1.2-1 vorgestelltes System zugrunde liegt. Im CPM werden Zellen verwendet, die in der Implementierung, durch mehrere Knoten mit demselben Zustand repräsentiert werden. Diese Knoten befinden sich alle in einer Umgebung zueinander und werden damit als ein Verbund an Knoten angesehen. Die Interaktion mit anderen Zellen geschieht über den Rand des Verbunds von Knoten. Diese Knoten können den Zustand ändern und damit die Zugehörigkeit zu einer bestimmten Zelle.

Übersetzung dieses Verhaltens kann folgendermaßen geschehen. Das System wird beschränkt auf die Knoten im Randbereich des Verbundes, wobei die Zellen im Inneren beim Update aller Knoten übergangen werden. Lediglich die Knoten im Rand des Verbundes können einen Exchange, Spin oder Flip vornehmen. Daher muss das System eingeschränkt werden, dass ein Verbund an Zellen nur aus einer Zelle besteht, sodass sein Verhalten simuliert werden kann.

### Zellsystem

Die Modellierung der einzelnen Zelle ist bereits in Kapitel 3.2.1 beschrieben, weswegen in diesem Abschnitt nur auf die Funktionalitäten eingegangen wird. Wie bereits in Kapitel 3.1 beschrieben, kann eine Funktionalität über ein Plug-In eingebaut werden. Die Funktionalitäten des Systems wurden auf folgende Bestandteile festgelegt:

* Exchange
* Flip
* Divide

Diese Funktionalitäten sollen jeweils über ein eigenes Plug-In realisiert werden, sodass dieses Plug-In an verschiedenen Stellen des Codes aufgerufen kann. Alle Plug-Ins sollen folgenden Aufbau besitzen. Sie bestehen aus zwei Komponenten, zum einen der Übergabe des Zellsystems und zum anderen der Wahrscheinlichkeit des Eintretens dieser Aktion. Im Zellsystem wird eine spezifische Zelle ausgewählt und die Wahrscheinlichkeit auf sie angewandt.

## Erweiterungsmöglichkeiten

Da Morpheus aktuell nur das CPM als eine Umgebung unterstützt, ist es nur über eine eigens definierten Umgebung möglich den zellulären Automaten als Funktionalität von Morpheus einzubauen. Diese Umgebung könnte über einen Button in der GUI ausgewählt werden. Dafür sind Umbauten am System notwendig. Dazu muss im Core-Bereich von Morpheus eine neue Funktionalität eingebaut werden, die eine andere Interpretierung des eingegebenen Code vornimmt. Zudem müssen die bestehenden Strukturen an die neue Umgebung angepasst werden, sodass der gleiche Code verwendet werden kann, jedoch in einem anderen Kontext.

Des Weiteren könnte den Knoten unterschiedliche Eigenschaften zugewiesen werden, womit verschiedene Zelltypen im System realisiert werden können. Diese Zellen haben unterschiedliche Verhaltensweisen und damit auch angepasste Wahrscheinlichkeiten für ihre Aktionen. Somit wäre es möglich das Verhalten von Krebszellen innerhalb von gesunden Zellen zu simulieren. Diese Möglichkeiten sind im Verlauf der Arbeit noch in Betracht gezogen, jedoch aufgrund des Zeitaufwandes verworfen worden.

# Prototypische Umsetzung

Die Umsetzung des Graphen basierten Systems, nutzt das JavaScript Framework „*AngularJS*“ von Google. Die Plattformunabhängigkeit, Trennung von Datenstruktur und Oberfläche sowie die einfache Einbindung von neuen Bibliotheken, machen das JavaScript Framework zu einem hervorragenden Werkzeug zur Realisierung eines Prototyps.

In der ersten Programmversion wurde auf die JavaScript Bibliothek von „*SigmaJS*“ zurückgegriffen. Diese auf Graphen spezialisierte Bibliothek bringt ihr eigenes Datenmodell für die Vertices und Kantenbeziehungen mit. Dieses erwies sich jedoch, für die Simulation von Graphen mit mehr als 500 Knoten mit knapp 2 Minuten pro Zyklus, als zu langsam. Eine eigens geschriebene Implementation der Datenstruktur und Visualisierung mit Hilfe von Canvas, verringerte den Rechenaufwand um knapp 90% auf 10 Sekunden im Durchschnitt.

In der aktuellen Version setzt der Prototyp auf die 3D.js Bibliothek zur Berechnung des dualen Graphen und dessen Visualisierung. Durch die Verwendung von 3D.js konnte die Berechnungszeit zur Erstellung des Graphen auf <1 Sekunde reduziert werden. Während der Simulation wird über AngularJS die Berechnung der Nachbarschaften und die Aktualisierung des Spins der Knoten im Modell durchgeführt. Der Prototyp ist ebenfalls in der Lage eine Simulation auf Grundlage des zellulären Automaten durchzuführen. Für beide Varianten können die folgenden Parameter spezifiziert werden:

* die Anzahl der Zellen,
* die Teilungsrate der Zellen,
* die Bewegungsrate der Zellen sowie die
* initiale Menge der erkrankten Zellen.

Das Projekt kann unter der URL <http://krockema.github.io/> von jedem Interessierten eingesehen werden. Zum Ausführen der Simulation wird Firefox ab Version 45 oder Chrome ab Version 42 benötigt. Alternativ jeder andere Browser mit „*ECMAScript 6*“ Unterstützung verwendet werden.

# Auswertung

## Beobachtung für CA Model im symmetrischen Gitternetz

Erste Versuche mit dem CA Modell scheinen vorwiegend einzelne Haufen in Randbereichen auszubilden, bevor das System vollständig belegt ist. Für dieses Phänomen gibt es drei mögliche Lösungsstrategien. Zum einen kann der Rand als solcher belassen werden, da es ja auch innerhalb eines Organs Grenzen gibt. Denkbar ist auch einen reflektierenden Rand zu schaffen der bei der Aktualisierung wie ein Spiegel wirkt. Dabei wird die Bewegungsrichtung des Knotens, wenn er sich über den Rand hinausbewegen will, um 180° gedreht, sofern es ihm erlaubt ist, sich in diese Richtung zu bewegen. Die dritte Möglichkeit ist die Verknüpfung der Ränder miteinander. So würde ein Torus entstehen der nunmehr keine Randbereiche mehr in seiner Fläche aufweist (siehe Abbildung 5.1‑1).

|  |
| --- |
| \\DEDRSFIL001\krockema$\Eigene Bilder\310px-Simple_Torus.svg.png |

Abbildung ‑: Veranschaulichung der Verbindung der Randbereiche des Gitters zu einem Torus.

Quelle: Eigene Darstellung

## Beobachtung für Graphen

Bei den Tests für die Erstellung des Graphen zeigte sich, dass die Kanten im Randbereich nicht immer korrekt erzeugt werden. In der Abbildung 5.2‑1 ist der Effekt rot eingerahmt und gut zu erkennen. Es scheint bei zu weit auseinanderliegenden Punkten keine Kanten einzuzeichnen. Da die Randbereiche jedoch einen Spezialfall des Systems darstellen und die Fehlerquote bei einer hohen Punktdichte minimal ist, sollte dies jedoch nur geringe Auswirkungen haben.

|  |
| --- |
|  |

Abbildung ‑: Sweep Line Algorithmus – Fehlende Verbindungen bei der Erstellung des Graphen.

Quelle: Eigene Darstellung

## Beobachtungen bei Morpheus

Morpheus ist ein sehr umfangreiches Programm, das zum aktuellen Stand über 43 Klassen und etliche Plug-Ins verfügt. Jede dieser Klassen beinhaltet eine große Anzahl an Code, der durchgearbeitet und verstanden werden muss. Dies ist notwendig, da die Objekte zur Darstellung des Systems in den eigenen Plug-Ins verwendet werden müssen. Somit erschwert die Komplexität von Morpheus die Erstellung einer neuen Umgebung für den CA.

Mit diesem Problem geht die Tatsache einher, dass die Software Open Source ist und sich damit in einer ständigen Veränderung befindet und Dokumentation nur unzureichend vorhanden ist. Das erschwert die Suche nach der richtigen Wahl der Objekte erheblich. Zusammenfassend ist zu sagen, dass keine einfache Anpassung der bestehenden Strukturen an die CA-Struktur möglich ist.

Zur Hilfestellung können bereits existierende Plug-Ins genutzt werden. Jedoch verwenden diese zur Realisierung das CPM. Dabei entstehen Schwierigkeiten in der Übersetzung der Informationen vom CPM in das CA-Modell. Bezüglich des Modells existiert eine weitere Hürde in der Verwendung der unterschiedlichen Zeitschritte. Während Morpheus das globale Aktualisierung umgesetzt hat, soll im vorliegendem System ein lokales Update angewandt werden. Dies ließe sich umsetzen, indem das globale Update durch das Update einer einzigen Zelle ersetzt wird. Unter der Annahme, dass das globale Update auf einen einzigen Knoten beschränkt ist, muss um ein Update des gesamten Modells durchzuführen, der Vorgang mehrfach aufgerufen werden. Die auswertenden Systeme gehen jedoch davon aus, dass in einem einzigen Schritt alle Knoten ein Update erfahren haben. Dies führt zu einer Diskrepanz, die bei der Auswertung der Daten mit einbezogen werden muss.

In einem Gespräch mit dem Entwickler von Morpheus, wurde auch eine Umsetzung des zellulären Automaten als eigenständige Umgebung diskutiert. Diese Idee wurde jedoch durch die dafür notwendigen, umfangreichen Anpassungen in der GUI, als auch in den Core-Dateien von Morpheus verworfen.

# Ausblick

Alle drei Modelle sind mit ihren Eigenschaften und Verhalten modelliert. Ebenso steht für alle drei Modelle eine Softwareumgebung zur Verfügung in denen das Modellverhalten simuliert werden kann. Aufgrund der Komplikationen im Zusammenhang mit Morpheus, wird die weitere Betrachtung von und Entwicklung der Modelle in Morpheus eingestellt. Der Fokus in den nächsten Schritten liegt auf der Auswertung der Daten. So sollen geeignete Parameter gewählt und die Modelle anhand dieser verglichen werden. Dabei sollen Statistiken mittels der Programmiersprache R erstellt werden, die den Vergleich der verschiedenen Systeme veranschaulichen kann. Dazu müssen aus den erzeugten bzw. vorhandenen Systemen Daten zur Verfügung gestellt und bestenfalls in ein einheitliches Format überführt werden.

Ebenso ist eine Weiterentwicklung des CGM und CA Simulators denkbar. So könnte versucht werden die Zellverteilung in einem vorgegebenen Polygon vorzunehmen und so einen unsymmetrischen Randbereich zu erzeugen. Zudem wäre es denkbar im CGM eine „echte“ Zellbewegung zu simulieren. Das kann realisiert werden, indem man die Punkte als Zellkern interpretiert und diesen eine Richtung und Geschwindigkeit gibt. Eine weitere interessante Erweiterung ist die Zellteilung nicht mehr anhand der Frage zu stellen, ob ein Nachbarplatz frei ist, sondern anhand der Zelldichte im umliegenden Areal und ob diese eine weitere Zelle aufnehmen kann.

Anlagenverzeichnis

[Anlage 1: Bewertungsgrundlage 27](#_Toc454360168)

Anlagen

Auf den folgenden Seiten befinden sich die Anlagen.

Bewertungsgrundlage Anlage 1  
 Blatt 1

Anlage

Anlage : Bewertungsgrundlage

Quelle: eigene Darstellung

Bewertungsgrundlage Anlage 1  
 Blatt 2

Anlagebeispiel

Literaturverzeichnis

**Kort, S.** (2012), skanderkort.com. An Erdos-Renyi Random Graph Generator and Analyzer. in: https://skanderkort.com/erdos\_renyi\_graph\_generator\_analyzer. Abgerufen am: 17. Juni 2016.

**TU BRAUNSCHWEIG.** (2002), Institut für Geologie. GIS & Simulationsmodelle. in: http://rzv037.rz.tu-bs.de/gis/gis/drucken/ft12.htm. Abgerufen am: 05. Mai 2016.

Eidesstattliche Erklärung

Wir erklären an Eides statt, dass wir die vorliegende Arbeit selbständig und ohne unerlaubte fremde Hilfe angefertigt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt haben. Die aus fremden Quellen direkt oder indirekt übernommenen Stellen sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ort, Abgabetermin Unterschrift der Verfasser

1. Cellular Potts Modell – CPM [↑](#footnote-ref-1)
2. Cellular Automata – CA [↑](#footnote-ref-2)
3. Cellular Graph Model – CGM [↑](#footnote-ref-3)
4. Kort, S. (2012) [↑](#footnote-ref-4)
5. GUI – Grafical User Interface [↑](#footnote-ref-5)
6. Id – Identifikator [↑](#footnote-ref-6)