Zweidimensionale Modelle der Berechenbarkeit: Theoretische und didaktische Überlegungen

Schriftliche Hausarbeit

im Rahmen der Ersten Staatsprüfung,
dem Landesprüfungsamt für Erste Staatsprüfungen
für Lehrämter an Schulen in Aachen
vorgelegt von

Katrin Moßbrucker

Aachen, den 2. Juli 2007

UNIV.-PROF. DR. WOLFGANG THOMAS
Lehrstuhl für Informatik 7

Inhaltsverzeichnis

1 Einführung in die Thematik **1**

1.1 Motivation und Gliederung der Arbeit 1

1.2 Berechnungsproblem und Algorithmus 3

1.2.1 Beispiele von Berechnungsproblemen und Algorithmen . . . 5

1.3 Berechnungsmodell 7

1.3.1 Das Berechnungsmodell „Bleistift und Papier“ 9

1.3.2 Exkurs: Geometrische Algorithmen 11

2 Theorie **17**

2.1 Zweidimensionale Berechnungsmodelle 17

2.1.1 Die zweidimensionale Turingmaschine 17

2.1.2 Paralleles zweidimensionales Berechnungsmodell 22

2.2 Zweidimensionale Berechenbarkeit 25

2.2.1 Definition und Beispiele 25

2.2.2 Reduktion auf das eindimensionale Modell 34

2.2.3 Offene und unlösbare Probleme 55

2.2.4 Idee der universellen Turingmaschine 66

3 Didaktische Überlegungen **73**

3.1 Zweidimensionaler Berechenbarkeitsansatz für die Schul-Informatik . 73

3.1.1 Einführung in die Thematik 75

3.2 TuringKara - Zweidimensionale Turingmaschine 80

3.2.1 Die TuringKara-Umgebung 80

3.2.2 Beispielaufgaben mit TuringKara 83

3.2.3 Offenes Problem in der TuringKara-Umgebung 88

3.2.4 Die eindimensionale Turingmaschine 90

3.2.5 Die universelle Turingmaschine bei [RNH04] 94

4 Zusammenfassung **99**

Literaturverzeichnis **100**

Die in der vorgestellten Examensarbeit benutzten Bezeichnungen beziehen sich gleichermaßen auf männliche und weibliche Personen. Allein aus Gründen der besseren Lesbarkeit wurde auf die zweifache Schreibweise verzichtet.

Abbildungsverzeichnis

1.1 Sieb des Eratosthenes für $S = 100$ (in Zehnerzeilen notiert) 6

1.2 Addition, Subtraktion, Multiplikation und Division nach Schulmethode mit Hilfe von Papier und Stift 9

1.3 Alan Turing 10

1.4 Konstruktion des Mittelpunkts einer gegebenen Strecke 13

1.5 Konstruktion eines Dreiecks mit drei gebenen Strecken 13

1.6 Konstruktion des Schwerpunkts eines gegebenen Dreiecks 14

1.7 Die Dreiteilung des Winkels nach Archimedes 15

2.1 Zeichensatz des Papieralphabets 18

2.2 Turingmaschine mit zweidimensionalem Arbeitspapier 20

2.3 Tabellarische Darstellung eines Turingprogramms auf einem Papier . 20

2.4 Berechnungsschritt einer zweidimensionalen Turingmaschine 21

2.5 Zellulärer Automat: Dreieckiges Gitter - Rechteckiges Gitter - Hexagonales Gitter 23

2.6 Nachbarschaften eines zweidimensionalen zellulären Automaten . . . 24

2.7 Programm einer zweidimensionalen Turingmaschine zur Berechnung der Addition zweier Binärzahlen 27

2.8 Beispielberechnung einer zweidimensionalen Turingmaschine zur Addition von $3+3=6$ in Binärkodierung 28

2.9 Initialisierung des Turingpapiers bei Addition und Subtraktion 29

2.10 Programm einer zweidimensionalen Turingmaschine zur Subtraktion zweier Binärzahlen 30

2.11 Beispielberechnung einer zweidimensionalen Turingmaschine zur Subtraktion von $6-3=3$ in Binärkodierung 31

2.12 Pixel-Bild und Zustandsüberführungsfunktion: Ist das Bild regelmäßig gemustert mit Stern und Kringel? 33

2.13 Die Geburt der Turingmaschine 34

2.14 Turingmaschine mit eindimensionalem Arbeitsband 35

2.15 Tabellarische Darstellung eines Turingprogramms einer eindimensionalen Turingmaschine 36

2.16 Berechnungsschritt einer eindimensionalen Turingmaschine 36

2.17 Initialisierung des Turingbandes bei Addition zweier Binärzahlen . . . 37

2.18 Programm einer eindimensionalen Turingmaschine zur Addition zweier Binärzahlen 39

2.19 Beispielberechnung einer eindimensionalen Turingmaschine zur Addition von $3+3=6$ in Binärkodierung 40

2.20 Abbildung eines Turingpapiers auf ein Turingband 41

2.21 Zustandsüberführungsfunktion zur Simulation einer Kopfbewegung O . . . 45

2.22 Beispiel zur Simulation einer Kopfbewegung O 48

2.23 Zustandsüberführungsfunktion zur Simulation einer Kopfbewegung R 51

2.24 Beispiel zur Simulation einer Kopfbewegung R 54

2.25 Electronic Numerical Integrator and Calculator - erster elektronischer digitaler Universalrechner 55

2.26 Beispielfolgen zum Collatz-Problem 56

2.27 Die ersten Schritte der Ameise 57

2.28 Das komplex chaotische Verhalten und die Autobahn der Ameise . . . 58

2.29 Übergangsregeln der Ameise 59

2.30 Die Dreiteilung des Winkels 61

2.31 Die Verdopplung des Würfels 62

2.32 Die Quadratur des Kreises 62

2.33 Übergangsregeln beim „Game of Life“ 63

2.34 Die ersten Generationen einer Anfangspopulation 65

2.35 Links: Oszillatoren, Rechts: Stabile Populationen 65

2.36 Gödelnummer einer Turingmaschine zur Invertierung eines Bitstrings 66

2.37 Die zweidimensionale universelle Turingmaschine U 68

2.38 Programm einer zweidimensionalen Turingmaschine zur Invertierung eines Bitstrings 71

2.39 Gödeltabelle einer Turingmaschine zur Invertierung eines Bitstrings . 71

2.40 Kodierung einer eindimensionalen Turingmaschine zur Invertierung eines Bitstrings über einem Alphabet 72

3.1 Modell der zweidimensionalen Turingmaschine 76

3.2 Die TuringKara-Umgebung (Welteditor und Programmeditor) 81

3.3 Invertieren eines Bitstrings mit Hilfe einer Kara-Turingmaschine . . . 84

3.4 Zustandsüberführungsfunktion einer Kara-Turingmaschine zur Invertierung eines Bitstrings 85

3.5 Addition zweier Binärzahlen mit TuringKara 85

3.6 Zustandsüberführungsfunktion einer Kara-Turingmaschine zur Addition zweier Binärzahlen 86

3.7 Traversierung eines Labyrinths mit TuringKara 87

3.8 Zustandsüberführungsfunktion einer Kara-Turingmaschine zur Traversierung eines Labyrinths 87

3.9 Zustandsüberführungsfunktion einer Kara-Turingmaschine zur Simulation des Verhaltens der Ameise 89

3.10 Langtons Ameise in der TuringKara-Umgebung 89

3.11 Verhalten der Ameise in der TuringKara-Umgebung 90

3.12 Die eindimensionale Turingmaschine in der TuringKara-Umgebung . 91

3.13 Zustandsüberführungsfunktion einer eindimensionalen Kara- Turingmaschine zur Untersuchung einer Zeichenkette bestehend aus n Nullen gefolt von n Einsen 92

3.14 Untersuchung einer Zeichenkette bestehen aus n Nullen gefolgt von n Einsen in der TuringKara-Umgebung 93

3.15 Eingabe der universellen Kara-Turingmaschine nach [RNH04] 96

3.16 Die universelle Turingmaschine in der TuringKara-Umgebung 97

Danksagung

Dank sagen möchte ich allen, die mir bei der Erstellung dieser Arbeit geholfen haben. Insbesondere Herrn Prof. Dr. Dr. h.c. Wolfgang Thomas für die Betreuung der Arbeit, Herrn Prof. Dr. Ulrik Schroeder für die Zweitbegutachtung der Arbeit, Herrn Dr. Christof Löding für die ausführliche Beratung in theoretischen Fragen, Herrn Dipl.-Ing. Architektur Lars Kratzheller und Herrn Georg Böhm für die Hilfe bei der Erstellung von Grafiken, Herrn StD Dieter Moßbrucker für die Unterstützung in didaktischen Fragen sowie Frau Dipl.-Math. Annette Moßbrucker und Frau Julia Bätz für das Korrekturlesen der Arbeit.

Kapitel 1

Einführung in die Thematik

1.1 Motivation und Gliederung der Arbeit

Das Ziel dieser Arbeit ist es, einen zweidimensionalen Ansatz für Begriffe der Berechenbarkeitstheorie einzuführen, motiviert durch das algorithmische Arbeiten mit Bleistift und Papier, d.h. auf einem zweidimensionalen Medium. Es soll deutlich gemacht werden, dass ein zweidimensionaler Zugang in der Schulinformatik sehr sinnvoll ist, um ausgewählte Themen rund um die Berechenbarkeitstheorie, anschaulich und verständlich zu vermitteln.

Im ersten Teil dieser Arbeit wird in die Thematik der zweidimensionalen Berechenbarekeitstheorie eingeführt, indem die Begriffe *Berechnungsproblem*, *Algorithmus* und *Berechnungsmodell* erläutert und an Hand der Beispiele „Addition“ und „Primzahlbestimmung“ verdeutlicht werden. Desweiteren wird die Grundidee zweidimensionalen Arbeitens durch das Berechnungsmodell „Papier und Stift“ erläutert. In einem Exkurs zu geometrischen Algorithmen wird ein weiteres Berechnungsmodell „Zirkel und Lineal“ vorgestellt.

Im zweiten Teil werden theoretische Grundlagen geschaffen. Es werden zweidimensionale Berechnungsmodelle wie die *Turingmaschine* und der *zelluläre Automat* vorgestellt. Anhand dieser Modelle wird der Begriff *berechenbar* diskutiert und Berechnungsprobleme aus der Arithmetik und Pixel-Bild-Geometrie vorgeführt, welche mit Hilfe dieser Modelle beschrieben und gelöst werden können. Anschließend wird die Reduktion auf das eindimensionale Modell beschrieben, um Vergleiche und Vorteile

zwischen dem eindimensionalen und zweidimensionalen Modell festhalten zu können. Außerdem wird ein Beweis geliefert, dass die zweidimensionale und eindimensionale Turingmaschine äquivalente Berechnungsmodelle sind. In einem weiteren Abschnitt werden die Grenzen in der Berechenbarkeitstheorie untersucht, indem die Begriffe *offenes Problem* und *unlösbar* erklärt und mit Hilfe von Beispielen veranschaulicht werden. Abschließend wird die Idee und vorallem der Vorteil einer zweidimensionalen universellen Turingmaschine vorgestellt.

Im dritten und letzten Teil werden didaktische Überlegungen angestellt, um einen zweidimensionalen Ansatz für die Schulinformatik zu schaffen. Hierfür werden Ideen zu Einführung in die Thematik beschrieben und ein spielerischer Zugang zur Berechenbarkeit mittels der Lernsoftware „TuringKara“ gewählt. Exemplarische Beispiele für den Unterricht werden vorgeführt und Ideen zur Ausführung einer Unterrichtseinheit zur Berechenbarkeit vorgestellt. Dabei werden die Grenzen der Berechenbarkeit, das eindimensionale Modell und die universelle Turingmaschine mit Hilfe von TuringKara thematisiert.

1.2 Berechnungsproblem und Algorithmus

Die zentrale Frage der Berechenbarkeitstheorie lautet „Welche Berechnungsprobleme sind lösbar“. Die Berechenbarkeitstheorie untersucht algorithmische Berechnungsprobleme bezüglich ihrer Lösbarkeit. Dies bedeutet, dass man daran interessiert ist, ob ein Algorithmus für die Lösung existiert und wenn, diesen anzugeben. Einen Algorithmus stuft man in einem weiteren Schritt nach seiner Qualität, das heißt nach seiner Effizienz, ab. Dieser Aspekt, welcher in der Komplexitätstheorie verfolgt wird, wird in der vorliegenden Arbeit nicht behandelt. Um sich mit der Lösbarkeit von algorithmischen Problemen beschäftigen zu können, muss geklärt werden, was ein *Berechnungsproblem* ist und was *lösbar* heißt. Dabei werden Berechnungsprobleme formal jeweils in einem entsprechenden sprachlichen System beschrieben (Umgangssprache, formale Logik), das hier nur grob festgelegt wird. Ein Berechnungsproblem und seine Lösung können in drei Stufen eingeführt werden:

1. Stufe:	abstraktes Problem
	$\downarrow$
2. Stufe:	formale Beschreibung
	$\downarrow$
3. Stufe:	Programm / Algorithmus

Gegeben ist die sprachliche Formulierung eines abstrakten Problems (1. Stufe), für welches eine formale mathematische Beschreibung (2. Stufe) gefordert ist. Die Berechenbarkeitsfrage besteht darin, die mathematische Beschreibung in ein Programm oder in einen Algorithmus (3. Stufe) zu übersetzen, so dass eine Berechnung Schritt für Schritt festgelegt ist, um zu einer eindeutigen Lösung in endlich vielen Schritten zu gelangen. Formal werden diese drei Stufen wie folgt definiert werden:

Definition 1.2.1 *Unter einem* Berechnungsproblem *einer abstrakten Formulierung eines gegebenen Problems versteht man die Beschreibung einer Funktion* $f : Input \to Output$*, die zu einer Eingabe* $x \in Input$ *aus einem Bereich* $Input$ *endlicher Objekte eine eindeutige Ausgabe* $y \in Output$ *aus einem Bereich* $Output$ *endlicher Objekte liefert. Die Elemente der Objektbereiche* $Input$ *und* $Output$ *heißen* Eingabe- *bzw.* Ausgabedaten.

Für die formale Beschreibung eines Berechnungsproblems müssen für die Ein- und Ausgabe Bereiche aus endlichen, handhabbaren Objekten charakterisiert sowie eine Funktion zwischen diesen beiden Bereichen angegeben werden. Als Objektbereiche können Zahlen, Wörter oder auch geometrische Figuren (siehe hierzu „Geometrische Algorithmen“ in Abschnitt 1.3.2) gewählt werden. Der Beschreibung der Vielzahl von Berechnungsproblemen sind hier keine Grenzen gesetzt. In der Informatik werden als Standardobjektbereiche Wörter als endliche Folgen von Zeichen (zum Beispiel Buchstaben oder Ziffern) verstanden. Dies hat historische Gründe, da Computer auf Bitwörtern arbeiten. Auch die Grundlagenforscher Alan M. Turing (1912-1954) und Emil L. Post (1897-1954) wählten für ihre logischen Betrachtungen von Berechnungsproblemen in den 30er Jahren als Grundbereich Wörter bestehend aus Zeichenketten.

Eine spezielle Variante von Berechnungsproblemen stellen *Entscheidungsprobleme* dar. Entscheidungprobleme sind solche, bei denen der Objektbereich Y nur aus zwei Objekten beispielsweise 0 und 1 oder *Nein* und *Ja* besteht. Während man bei einem Berechnungsproblem an der konkreten Lösung interessiert ist, möchte man beim Entscheidungsproblem wissen, ob eine Lösung existiert oder nicht. Hierfür stehen die Antworten *Ja* und *Nein*, welche durch 1 und 0 charakterisiert werden können.

Die Lösung eines Berechnungsproblems wird nun in der nächsten Stufe durch die Konstruktion eines Algorithmus beschrieben, der Schritt für Schritt die Lösung berechnet. [Tho99] definiert den Begriff des Algorithmus wie folgt:

Definition 1.2.2 *Unter einem* Algorithmus *versteht man intuitiv ein Verfahren, das*

- *Eingabewörter schrittweise verarbeitet,*
- *durch einen endlichen Text bis ins letzte Detail eindeutig festgelegt ist,*
- *bei Termination eine Ausgabe liefert, oder nicht terminiert.*

Unter Eingabe und Ausgabe werden hier endlich dargestellte Objekte aus einem Objektbereich verstanden. Ein Algorithmus beschreibt eine genau definierte Handlungsvorschrift über einem Objektbereich, mit der sich ein Berechnungsproblem in endlich vielen Schritten lösen lässt. In unserem Alltag können Bedienungsanleitungen, Kochrezepte und vieles mehr als Algorithmen angeführt werden.

1.2.1 Beispiele von Berechnungsproblemen und Algorithmen

Die folgenden Beispiele beschreiben die drei Stufen eines Berechnungsproblems. Als Objektbereich werden hier die natürlichen Zahlen gewählt.

Beispiel 1.2.3 (Addition, Subtraktion, Multipliaktion, Division)
Die vier Grundrechenarten Addition, Subtraktion, Multipliaktion und Division der Arithmetik stellen Berechnungsprobleme dar. Zu zwei gegebenen natürlichen Zahlen x und y sollen die Summe $x+y$, die Differenz $x-y$, das Produkt $x \cdot y$ und der Quotient $\frac{x}{y}$ berechnet werden. Die Symbole $+$ und $-$ für die Rechenoperation wurden 1498 von dem Mathematiker Johannes Widman (1460-1498) eingeführt.
Für die Addition zweier natürlicher Zahlen werden die drei Stufen dieses Berechnungsproblems und seiner Lösung vorgestellt:

1. <u>Abstraktes Problem</u>
   Gesucht ist die Summe $x+y$ zweier natürlicher Zahlen x und y.

2. <u>Formale Beschreibung</u>
   Das Berechnungsproblem *Addition zweier natürlicher Zahlen* ist gegeben durch die Funktion $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, welche zu zwei natürlichen Zahlen x und y sie Summe $f(x,y) = x+y$ liefert.

3. <u>Algorithmus</u>
   Zu gegebenem x und y tue folgendes:

   (a) Setze $n = 1$, $Uebertrag = 0$, $Summe = 0$, $Ergebnis = 0$.

   (b) Setze $Ergebnis = Uebertrag + n$-te Stelle von x + n-te Stelle von y.

   (c) Falls $Ergebnis < 9$ setze n-te Stelle von $Summe = Ergebnis$
   und $Uebertrag = 0$.
   Andernfalls setze n-te Stelle von $Summe = Ergebnis - 10$
   und $Uebertrag = 1$.

   (d) Erhöhe n um 1.

   (e) Falls x oder y mehr als n Stellen haben, springe zu (b).
   Ansonsten falls $Uebertrag = 0$ beende die Berechnung.
   Andernfalls setze $n+1$-te Stelle von $Summe = 1$ und beende die Berechnung.

Beispiel 1.2.4 (Primzahlenbestimmung)

Das Berechnungsproblem *Primzahlenbestimmung* wird duch den Algorithmus *Sieb des Erathostenes* gelöst, bei dem aus einer aufsteigenden Liste von natürlichen Zahlen alle Primzahlen bis zu einer vorgegebenen Endzahl, der sogenannte Schwellwert, ermittelt werden. Dieses Berechnungsproblem ist nach dem griechischen Mathematiker Eratosthenes von Kyrene (ca. 284 v.Chr.-202 v.Chr.) benannt.

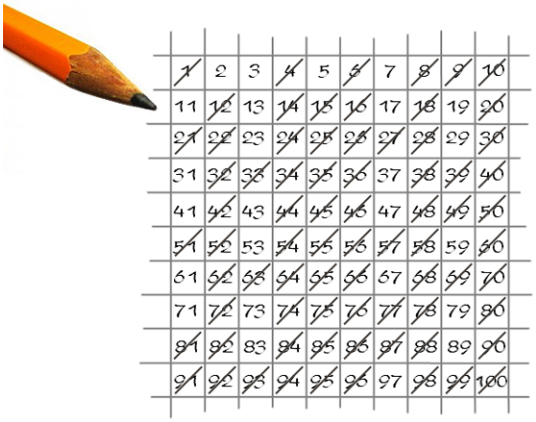

Abbildung 1.1: Sieb des Eratosthenes für $S = 100$ (in Zehnerzeilen notiert)

Zur Ermittlung der Primzahl bis zu einem freiwählbaren Schwellwert $S \in \mathbb{N}$ werden alle Zahlen von 1 bis S in einer Liste aufgeschrieben. Alle Zahlen sind zunächst unmarkiert und kommen damit als Primzahl in Frage. Die kleinste unmarkierte Zahl ist immer eine Primzahl, wobei die Zahl 1 gestrichen wird, da sie keine Primzahl ist. Die Berechnung startet bei der kleinsten unmarkieretn Zahl, der Zahl 2, und markiert diese als Primzahl. Nun werden alle Vielfache dieser Zahl gestrichen bis zum Schwellwert S. Dann beginnt die Berechnung wieder von vorne bei der nun kleinsten unmarkierten Zahl. Dieser Vorgang wird solange wiederholt, bis alle Vielfachen einer Zahl gestrichen sind. Es bleiben die Primzahlen übrig.

Auf `http://www.arndt-bruenner.de/mathe/scripts/eratosthenes.htm` kann das Sieb des Eratosthenes anschaulich getestet werden. Hier können der Schwellwert angegeben und die Vielfachen nach und nach gestrichen werden, so dass am Ende der Berechnung alle Primzahlen zwischen 1 und S übrigbleiben.

Die drei Stufen des Berechnungsproblems und seiner Lösung sehen für die Primzahlbestimmung folgendermaßen aus:

1. Abstraktes Problem
   Gesucht sind alle Primzahlen zwischen 1 und einem freiwählbaren Schwellwert $S \in \mathbb{N}$

2. Formale Beschreibung
   Das Berechnungsproblem „Sieb des Eratosthenes“ ist gegeben durch die Funktion $f : \mathbb{N} \to \mathbb{N}^\star$, welche zu einem gegebenen freiwählbarem Schwellwert $S \in \mathbb{N}$, die Menge $P \in \mathbb{N}^\star$ aller Primzahlen von 1 bis S liefert. $f(S)$ ist die Liste der Primzahlen im Intervall $[1, \ldots, S]$.

3. Algorithmus
   Zu gegebenem Schwellwert $S \in \mathbb{N}$ führe folgende Schritte durch:

   (a) Schreibe alle natürlichen Zahlen von 1 bis S auf.

   (b) Setze die Siebzahl p auf 2, das heißt, streiche bereits die 1.

   (c) Solange $p < \sqrt{S}$ tue folgendes:

      i. Siebe mit der Siebzahl p, das heißt, streiche jede $p-$te Zahl.

      ii. Setze p auf die nächste nichtgestrichene Zahl.

1.3 Berechnungsmodell

Um sich mit der Lösbarkeit von Berechnungsproblemen auseinandersetzen zu können, muss geklärt werden, was lösbar heißt. Da ein Berechnungsproblem durch eine Funktion, welche zu einer Eingabe eine eindeutige Ausgabe liefert, ausgedrückt wird, ist ein Berechnungsproblem lösbar, wenn die Funktion berechenbar ist. Durch die folgende intuitive Definition wird der Begriff *berechenbar* und damit auch der Begriff *lösbar* ausgedrückt:

Definition 1.3.1 *Man nennt eine Funktion $f : Input \to Output$* berechenbar*, wenn es einen Algorithmus gibt, der zu jeder Eingabe $w \in Input$ die Ausgabe $f(w) \in Output$ liefert.*

Lässt sich für ein Berechnungsproblem ein Algorithmus angeben, welcher das Berechnungsproblem löst, so ist das Berechnungsproblem lösbar bzw. die Funktion, welche die Lösung des Berechnungsproblems beschreibt, ist berechenbar. Um sich mit der Klasse der berechenbaren Funktionen beschäftigen zu können, muss ein *Berechnungsmodell* eingeführt werden. Ein Berechnungsmodell liefert Antworten auf die Möglichkeit und Unmöglichkeit einer Berechnung, welche die Lösung eines gegebenen Berechnungsproblems darstellt. Durch ein Berechnungsmodell wird mathematisch eindeutig festgelegt, auf welchen Daten, auf welchem Objektbereich, eine Berechnung mit welchen erlaubten Operationen durchgeführt werden soll. Diese Vorgaben sind dabei streng definiert, da kleinste Änderungen in den Vorgaben schon zu anderen Lösungen führen (vgl. hierzu in Abschnit 1.3.2 Beispiel 1.3.6 „Die Dreiteilung des Winkels nach Archimedes“). Der Begriff Berechnungsmodell wird demnach über den Begriff des Algorithmus verstanden, welcher ebenfalls festlegt, auf welchen Daten, mit was für erlaubten Operationen gearbeitet werden darf.

8 Um der strengen Definition des Berechnungsmodells gerecht zu werden, wurden Mitte des 20. Jahrhunderts durch die Logiker Alan Turing und Alonzo Church (1903-1995) formale Berechnungsmodelle definiert, die sich alle als äquivalent herausstellten und alle die gleichen berechenbaren Funktionen lösen können.

Diese historische Sicht verdeckt jedoch, dass uns seit je her spezielle Berechnungsmodelle, das heißt Berechnungsmodelle, welche für eine ganz bestimmte Berechnung eingesetzt werden, umgeben. Beispielsweise das Rechnen auf Papier mit Hilfe eines Stifts und Konstruktionsaufgaben mit Hilfe von Zirkel und Lineal stellen solche Berechnungsmodelle dar. Aus diesem Grund werden diese in den Abschnitten 1.3.1 „Bleistift und Papier“ und 1.3.2 „Geometrische Algorithmen“ vorgestellt. Hieraus ergibt sich die Motivation, zweidimensionale Berechnungsmodelle zu betrachten.

1.3.1 Das Berechnungsmodell „Bleistift und Papier“

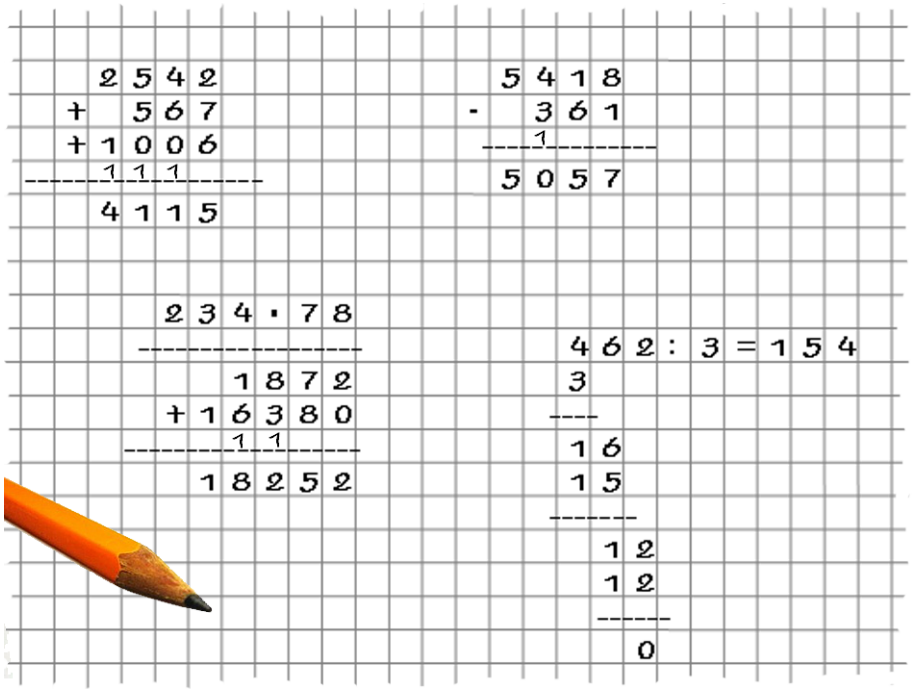

Abbildung 1.2: Addition, Subtraktion, Multiplikation und Division nach Schulmethode mit Hilfe von Papier und Stift

Die zentralen Hilfsmittel nicht nur in der Schule für Schüler und Lehrer, sondern in unserem alltäglichen Leben, Berechnungsprobleme gedanklich zu lösen, sind zunächst Papier und Stift. Sie stellen eine einfache Möglichkeit dar, sich gedanklich einer Problemstellung zu nähern, sie zu strukturieren, zu fixieren und zuletzt zu einem Ergebnis zu kommen. Sie sind Werkzeuge, welche wir von Klein auf trainieren. Ganz gleich, welche Richtung unser Leben einschlägt, kaum ein Berufsbild kommt ohne die Verwendung von Algorithmen mit Stift und Papier aus. Es gehört zur Allgemeinbildung, diese Werkzeuge bedienen zu können. Sie sind in unserem Denken und in unserem Leben fest verankert. Aus diesem Grund ist es sinnvoll, gerade für die Herangehensweise an eine Problemstellung, genau diese Werkzeuge zur Motivation zu Hilfe zu nehmen. Bereits in der Grundschule werden diese elementaren Gegenstände für Rechenmodelle eingesetzt. In der Mathematik kommen die Algorithmen für Addition, Subtraktion, Multiplikation und Division (vgl. Abbildung 1.2) mit dem Berechnungsmodell „Papier und Stift“ aus. Auch das Berechnunsproblem „Primzahlbestimmung“ wird durch den Algorithmus „Sieb des Erathostenes“ zweidimensional mit Papier und Stift gelöst (vgl. Abb. 1.1).

Diese Betrachtung legt doch nahe, ein Berechnungsmodell, welches sich an der Verwendung von Papier und Stift orientiert, zu Hilfe zu nehmen, um den Schülern den Einstieg in die Berechenbarkeitstheorie so auf gewohnte Weise zu erleichtern. Die Vorgaben des Berechnungsmodells „Papier und Stift“ sind wie folgt definiert:

Gegeben ist ein Objektbereich bestehend aus Symbolen. Der Stift des Modells darf

- alle Kästchen des Papiers mit den Symbolen des Objektbereichs beschriften,
- alle Symbole auf dem Papier durchstreichen und sie durch neue ersetzen.

Alan Turing hat bereits 1936 in einer knappen Formulierung versucht, auf diese Weise die Idee der Berechnbarkeitstheorie einzuführen. Dazu entwickelte er ein abstraktes Modell, welches die Ausführung eines Algorithmus durch einen Menschen, eben das Rechnen mit Papier und Stift, simuliert ([Tur36]):

Abbildung 1.3: Alan Turing

„Computing is normally done by writing certain symbols on paper. We may suppose this paper is divided into squares like a childs arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, i.e., on a tape divided into squares.“

Der Verzicht einer zweiten Dimension ist für den Leser nicht direkt nachvollziehbar. Und Turing bleibt ihm für die Formulierung „avoidable“ eine ausführliche Begründung schuldig. Gerade hier soll die vorliegende Arbeit ansetzen, indem auf die zweite Dimension nicht verzichtet und ein zweidimensionales Berechnungmodell betrachtet wird, an welchem sich theoretische, praktische und didaktische Überlegungen anknüpfen.

1.3.2 Exkurs: Geometrische Algorithmen

Das zweidimensionale Arbeiten ist auch aus der Geometrie bekannt. Der Begriff des Berechnungsmodells wurde bereits in der antiken Mathematik studiert. In der Antike interessierten sich die griechischen Mathematiker für die Frage, welche geometrischen Konstruktionen sich mit Hilfe von Zirkel und Lineal, welche sie „die euklidischen Werkzeuge“ nannten, durchführen lassen. Die Konstruktion mit Zirkel und Lineal ist eine klassische Methode, um geometrische Figuren aus gegebenen Größen zu zeichnen. Dabei dürfen ausschließlich diese beiden Werkzeuge verwendet werden, wobei das Lineal keine Markierung besitzt. Das hier zugrundeliegende „special-purpose“ Berechnungsmodell soll also eine Antwort auf die Frage „Welche Aufgaben sind mit den Werkzeugen Zirkel und Lineal lösbar“ liefern. Auch hier müssen Daten und erlaubte Operationen des Berechnungsmodells „Zirkel und Lineal (und Papier)“ präzise definiert sein:

Gegeben ist ein Objektbereich bestehend aus geometrischen Figuren, wobei eine geometrische Figur im $\mathbb{R}^2$ aus der Vereinigung von bereits vorhandenen Punkten, Geraden und Kreisen besteht. Die geometrischen Objekte des Objektbereichs sind hier Punkte, Geraden und Kreise. Formal definiert [Lor96] eine geometrische Figur wie folgt:

Definition 1.3.2 *Seien $M \neq \emptyset$ eine Menge von Punkten, $\mathscr{G}(M) :=$ Menge von Geraden durch je zwei verschiedene Punkte von M und $\mathscr{K}(M) :=$ Menge der Kreise mit Mittelpunkt aus M und Radien, die durch den Abstand des Mittelpunkts zu einem anderen Punkt aus M gegeben sind. Dann ist eine* geometrische Figur *gegeben durch $\mathscr{F} := (M, G, K)$ mit $M \neq \emptyset$, $G \subseteq \mathscr{G}(M)$ und $K \subseteq \mathscr{K}(M)$.*

Durch folgende zugelassene Anwendungen der Werkzeuge Zirkel und Lineal können aus einer gegebenen Menge M von Punkten in diesem Berechnungsmodell neue Punkte konstruiert werden:

- Durch das Schneiden zweier Geraden aus $\mathscr{G}(M)$ erhält man die zwei Schnittpunkte als neue Punkte.
- Durch das Schneiden einer Geraden aus $\mathscr{G}(M)$ mit einem Kreis aus $\mathscr{K}(M)$ erhält man die beiden Schnittpunkte als neue Punkte.
- Durch das Schneiden zweier Kreise aus $\mathscr{K}(M)$ erhält man maximal zwei Schnittpunkte als neue Punkte.

Mit diesen erlaubten Operationen werden folgende algebraische Berechnungen möglich (vgl. hierzu [RNH04]):

- die Addition und Subtraktion zweier reeller Zahlen
- die Multiplikation zweier reeller Zahlen
- die Bestimmung der Inversen einer reellen Zahl und damit die Division zweier reeller Zahlen
- das Ziehen einer Quadratwurzel einer reellen Zahl

Im Folgenden werden die geometrischen Berechnungsprobleme „Bestimmung des Mittelpunkts einer gegebenen Strecke“, „Konstruktion eines Dreiecks mit drei gegebenen Strecken“ und „Bestimmung des Schwerpunkts eines gegebenen Dreiecks“ vorgestellt, indem sowohl die Problemsituation als auch die algorithmische Lösung sprachlich formuliert werden. Für jedes Berechnungsproblem ist eine Abbildung der Konstruktion angegeben. In den Abbildungen sind gegebene Objekte blau und gesuchte Objekte grün dargestellt.

Beispiel 1.3.3 (Konstruktion des Mittelpunkts einer gegebenen Strecke)
Gegeben ist eine Strecke mit den Endpunkten A und B. Nun soll der Mittelpunkt M dieser Strecke konstuiert werden (vgl. Abbildung 1.4). Hierbei wird wie folgt vorgegangen:

1. Zeichne jeweils um A und B einen Kreis mit demselben Radius, wobei der Radius so gewählt wird, dass sich die beiden Kreise schneiden.
2. Bezeichne die beiden Schnittpunkte der Kreise mit C und D.
3. Ziehe eine Gerade durch die beiden Punkte C und D.
4. Der Schnittpunkt der Gerade mit der gegebenen Strecke $\overline{AB}$ bildet den Mittelpunkt M.

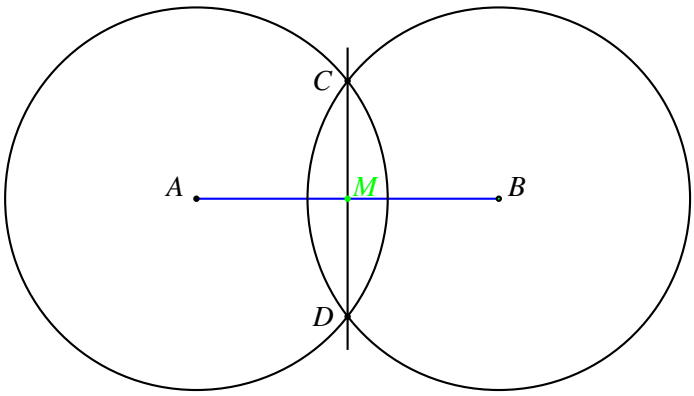

Abbildung 1.4: Konstruktion des Mittelpunkts einer gegebenen Strecke

Beispiel 1.3.4 (Konstruktion eines Dreiecks mit drei gegebenen Strecken)
Konstruiert werden soll ein Dreieck mit den Ecken A, B und C aus drei gegebenen Strecken a, b und c (vgl. Abbildung 1.5) mit Zirkel und Lineal. Für die Konstruktion geht man dabei in folgenden Schritten vor:

1. Zeichne die erste Strecke a mit den Endpunkten B und C.
2. Ziehe um B einen Kreis mit Radius c.
3. Ziehe um C einen Kreis mit Radius b.
4. Der Schnittpunkt der Kreisbögen um B und C bildet den Punkt A.
5. Verbinde A, B und C zu einem Dreieck.

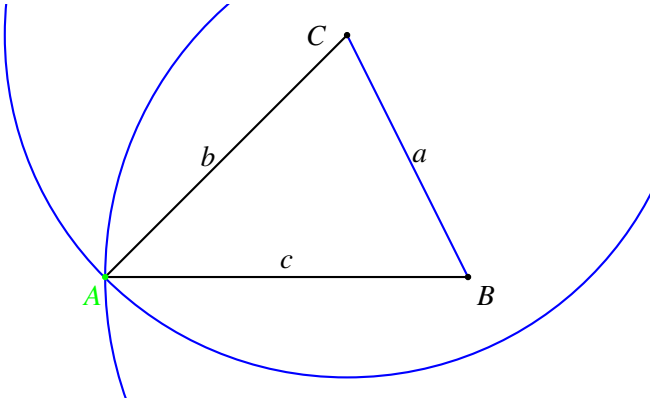

Abbildung 1.5: Konstruktion eines Dreiecks mit drei gebenen Strecken

Beispiel 1.3.5 (Konstruktion des Schwerpunkts eines gegebenen Dreiecks)
Gegeben ist ein Dreieck mit den Eckpunkten A, B und C. Der Schwerpunkt dieses Dreiecks wird in folgenden Schritten konstruiert (vgl. Abbildung 1.6):

1. Konstruiere die Mittelpunkte M_1, M_2 und M_3 der drei Seiten des gegebenen Dreiecks mit den Eckpunkten A, B und C (vgl. Beispiel 1.3.3 „Konstruktion des Mittelpunkts einer gegebenen Strecke“).

2. Verbinde die Punkte A und M_1, B und M_2 sowie C und M_3. Der Schnittpunkt dieser drei Strecken bildet den Schwerpunkt S.

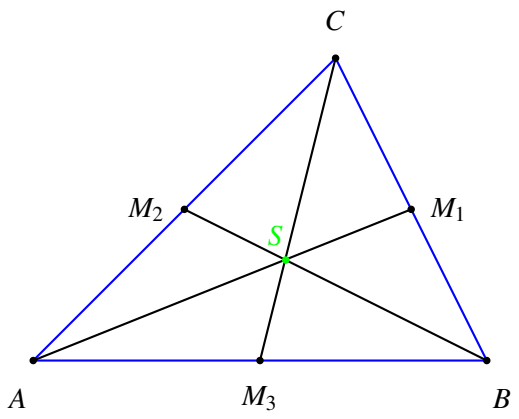

Abbildung 1.6: Konstruktion des Schwerpunkts eines gegebenen Dreiecks

Die zuvor beschriebenen Konstruktionsaufgaben sind mit den Werkzeugen Zirkel und Lineal lösbar. Eine Reihe von geometrischen Konstruktionen ist jedoch mit Hilfe der erlaubten Operationen mit Zirkel und Lineal nicht immer möglich, wie auch die „drei klassischen Probleme der Antike“.

Unter den drei klassischen Problemen der Antike versteht man folgende Probleme:

- Die Dreiteilung des Winkels
- Die Verdopplung des Würfels
- Die Quadratur des Kreises

Die Lösungen hiervon dürfen nur in endlich vielen Schritten, also in einer endlichen Anzahl von Anwendungen der beschriebenen erlaubten Operationen mit Hilfe der euklidischen Werkzeuge herbeigeführt werden. Mit Hilfe algebraischer Berechnungen, welche von Carl Friedrich Gauß (1777-1855) und Evariste Galois (1811-1832) erschaffen wurden, wurde erst im 19. Jahrhundert bewiesen, dass diese drei klassischen Probleme der antiken Mathematik allgemein nicht lösbar sind (vgl. Abschnitt 2.2.3 „Unlösbarkeit“). In Kapitel 2.2.3 „Offene Probleme und Unlösbarkeit“ werden weitere nichtlösbare Probleme vorgestellt.

Nun wird noch ein Konstruktionsmodell von Archimedes vorgestellt, welches durch eine Änderung der Vorgaben im Berechnunsmodell die Dreiteilung des Winkels mit Zirkel und Lineal möglich macht:

Beispiel 1.3.6 (Die Dreiteilung des Winkels nach Archimedes [RNH04])
Das Lineal bei der Konstruktion der Dreiteilung eines Winkels nach Archimedes von Syrakus (ca. 287 v.Chr.-212 v.Chr.) besitzt nun eine Maßeinteilung, so dass es möglich wird, einen Einheitsabstand abzutragen.
Die Konstruktion der Dreiteilung funktioniert dann wie folgt: Gegeben ist ein Winkel $\measuredangle AOB$ (vgl. Abbildung 1.7), so dass der Scheitel 0 des Winkels $\measuredangle AOB$ der Mittelpunkt eines Einheitskreises ist und die Punkte A und B auf dem Einheitskreis liegen. Nun wird das Lineal mit derselben Einheitsmarkierung zu Hilfe genommen. Es wird entlang der Geraden OB angelegt und so lange um den Drehpunkt B verschoben, bis der Abstand des Einheitskreises von der Geraden OA den Einheitsabstand beträgt. Dort wird ein Punkt C abgetragen. Der Winkel $\measuredangle OCB$ beträgt nun genau $1/3$ des Winkels $\measuredangle AOB$.

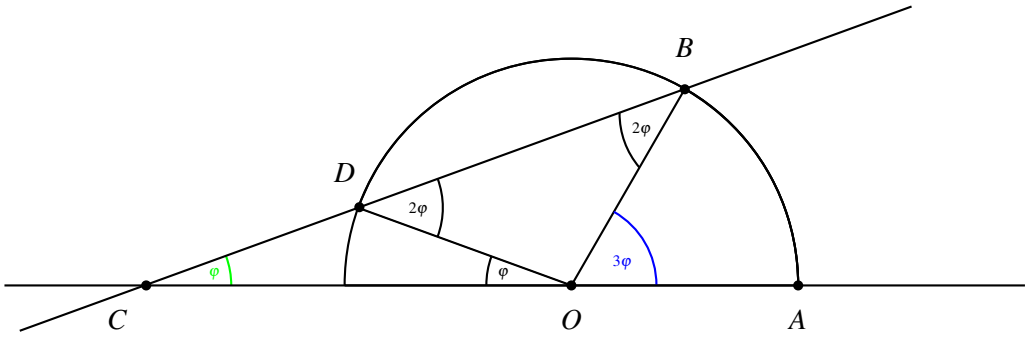

Abbildung 1.7: Die Dreiteilung des Winkels nach Archimedes

Eine anschauliche Demonstration der Dreiteilung des Winkels nach Archimedes ist unter http://agnes.dida.physik.uni-essen.de/~backhaus/Quanten/Werner/unmoeglich.ppt zu finden.

Durch eine Änderung im Berechnungsmodell wird nun die Dreiteilung eines Winkels möglich. Es kommt auf die präzise Definition des Erlaubten, also den Mitteln, welche benutzt werden dürfen, an. „Denn alles wäre berechenbar mit Hilfe eines Orakels“ (vgl. [RNH04]). Die hier behandelte Thematik zeigt, wie mächtig die Werkzeuge Zirkel und Lineal sind. Mit wenigen erlaubten Mitteln, können komplexe Probleme gelöst werden. Dies ist eine zentrale Beobachtung der Berechenbarkeitstheorie und führt zu dem Gedanken, möglichst einfache Berechnungsmodelle zu definieren, wie dies in Kapitel 2.1 „Zweidimensionale Berechnungsmodelle“ nach der Idee von Alan Turing durchgeführt wird. Im Folgenden werden diskrete Berechnungsmodelle betrachtet, welche Symbole als Berechnungsobjekte verwenden. Die Konzentration liegt hierbei auf Berechnungsbrobleme der Arithmetik und der Pixel-Bild-Geometrie.

Kapitel 2

Theorie

2.1 Zweidimensionale Berechnungsmodelle

Um ein Berechnungsproblem lösen bzw. es bezüglich seiner Lösbarkeit untersuchen zu können, benötigt man zur Durchführung von Algorithmen ein Medium. Dieses sollte eine möglichst einfache Form und die Eigenschaft besitzen, beschreib-, wiederbeschreib- und lesbar zu sein. Die Idee hierzu stammt von dem englischen Mathematiker Alan M. Turing (1912-1954) aus dem Jahr 1936. Er konzipierte eine rechnende Maschine, die *Turingmaschine*. Diese entspricht einem mentalen Rechenmodell, welches das menschliche Rechnen auf einem Blatt Papier mit Hilfe eines Bleistifts simuliert (vgl. Abschnitt 1.3.1 „Bleistift und Papier“). Bei der von Alan Turing konzipierten Turingmaschine handelt es sich um ein eindimensionales Rechenmodell, da er, wie bereits in 1.3.1 erwähnt, auf eine zweite Dimension verzichtet. Dieser Verzicht ist jedoch aus Sicht eines Schülers schwer nachvollziehbar. Aus diesem Grund und aus der Motivation heraus, ein zweidimensionales Rechenmodell zu finden, welches sich an die schriftliche Addition mit Papier und Stift anlehnt, soll ein Modell entwickelt werden, welches sich an Turings eindimensionaler Maschine orientiert.

2.1.1 Die zweidimensionale Turingmaschine

Betrachtet man die Turingmaschine als abstraktes Modell dessen, was auf einem Blatt Papier berechnet werden kann, ist die Betrachtung eines zweidimensionalen Modells naheliegend, da das Papier, auf welchem man rechnet, ebenfalls zweidimensional ist.

Gesucht ist eine Maschine, welche mit den Bausteinen *Papier*, *Stift* und „*menschliches Gehirn*“ auskommt. Dabei muss sie lesen, schreiben und überschreiben, was der Verwendung eines Radiergummies gleicht, sowie Befehle des Gehirns verarbeiten können.

Das karierte Papier wird durch das *Turingpapier*[1] dargestellt, welches in Felder eingeteilt und in alle vier Richtungen rechts, links, oben, unten) unbeschränkt ist.

Für den Stift wird ein *Lese-/Schreibkopf* eingesetzt, der sich auf dem Turingpapier in alle vier Richtungen (rechts, links, oben, unten) bewegen kann. Der Lese-/Schreibkopf der gesuchten Maschine steht also auf einem Kästchen des Turingpapiers und kann sich in eine beliebige Richtung (R, L, O, U) bewegen. Er kann erkennen welches Zeichen in dem Kästchen des Turingpapiers steht, dieses also lesen, und er kann dieses Zeichen durch ein anderes ersetzen, also das Zeichen überschreiben. Dabei steht ihm ein Zeichensatz zur Verfügung, um das Papier beschriften zu können.

Der Zeichensatz, welcher zur Beschriftung des Turingpapiers zur Verfügung steht, entstammt einem vorgegebenem Alphabet, sowie auch der Mensch einen Zeichenvorrat zur Verfügung hat, um sich mit Hilfe von Papier und Stift auszudrücken. Hierfür wird ein endliches *Eingabe-* und *Papieralphabet*[2] gewählt. Das Papieralphabet enthält Zeichen, die nicht im Eingabealphabet enthalten sind. Diese werden für die Durchführung der Berechnung benötigt. Bei der Addition zweier natürlicher Zahlen beispielsweise besteht das Eingabealphabet aus den natürlichen Zahlen. Bei der Berechnung der Addition kann ein Übertrag entstehen. Dies bedeutet, dass ein Zeichen durch ein Zeichen mit einem angehängten Übertrag überschrieben wird, welcher im nächsten Schritt mitverarbeitet wird. Die Zahl mit dem angehängten Übertrag (vgl. Abbildung 2.1) ist nicht im Eingabealphabet enthalten. Im Papieralphabet stehen die natürlichen Zahlen und alle natürlichen Zahlen mit einem angehängten Übertrag zur Verfügung.

$$\begin{array}{ccc} 3 & 9 & 5 \\ {}_1 & 2_1 & 7 \\ \hline 4 & 2 & 2 \end{array}$$

Abbildung 2.1: Zeichensatz des Papieralphabets

[1] Turing verwendet den Begriff *Turingband*, da sich der Kopf seiner eindimensionalen Turingmaschine auf einem Band bewegt. Hier wird der Begriff Turingpapier verwendet, um zu verdeutlichen, dass sich der Kopf einer zweidimensionalen Turingmaschine auf einem Papier befindet.

[2] Im eindimensionalen Modell wird der Begriff *Bandalphabet* verwendet.

Zusätzlich wird ein Zeichen eingeführt, welches ausdrückt, dass noch nichts in einem Kästchen steht, so wie auf einem Papier leere Kästchen zu finden sind. Dieses Zeichen wird mit *Blanksymbol* bezeichnet und ist nicht im Eingabealphabet enthalten. Es macht erkenntlich, welche Zeichen zur Eingabe und Ausgabe gehören.
Jetzt muss noch eine Beschreibung gefunden werden, um auszudrücken, woher die Maschine ihre Befehle für eine Berechnung bekommt, also eine Beschreibung für die Arbeit, die sonst das menschliche Gehirn übernimmt. Ein Befehl beinhaltet ein *Zeichen lesen*, eventuell ein neues *Zeichen schreiben* und eine *Kopfbewegung vollziehen*. Hiefür wird eine *Zustandsüberführungfunktion* eingeführt, das heißt die Maschine befindet sich zu jedem Zeitpunkt in einem von endlich vielen Zuständen. In Abhängigkeit vom aktuellen Zustand, in dem sich die Turingmachine befindet, und dem Zeichen, welches der Lese-/Schreibkopf gerade liest, beschriftet die Turingmaschine das Turingpapier neu. Anschließend vollzieht die Maschine eine Kopfbewegung, indem sie sich zum links, rechts, oben oder unten benachbarten Kästchen bewegt, oder stehen bleibt, und zuletzt in einen neuen Zustand wechselt (vgl. hierzu Abbildung 2.2). Zwei Zustände müssen als *Start-* und *Endzustand* ausgezeichnet werden, damit ein genau definierter Anfang und ein genau definiertes Ende einer *Berechnung* existiert.

All diese Ideen und Komponenten für die zweidimensionalen Turingmaschine werden formal durch die folgende Definition eines zweidimensionalen Rechenmodells ausgedrückt:

Definition 2.1.1 *Eine* zweidimensionale Turingmaschine *ist gegeben durch ein 7-Tupel* $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$ *mit*

- *einer endlichen Zustandsmenge* Q*,*
- *einem Eingabealphabet* Σ*,*
- *einem Papieralphabet* $\Gamma \supseteq \Sigma$*,*
- *einem ausgezeichneten Blanksymbol* $B \in \Gamma \setminus \Sigma$*,*
- *einem Anfangszustand* $q_{start} \in Q$*,*
- *einem Endzustand* $q_{stop} \in Q$*,*
- *und der Übergangsfunktion* $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, O, U, N\}$.

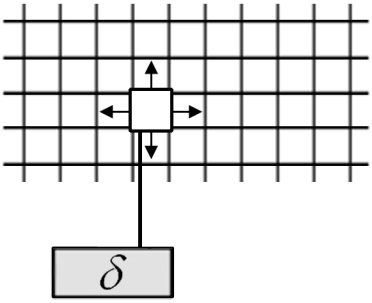

Abbildung 2.2: Turingmaschine mit zweidimensionalem Arbeitspapier

Die Zustandsüberführungsfunktion δ, das *Programm* der Turingmachine, ist wie folgt zu interpretieren:
Durch $\delta(q,a) = (q',a',L/R/O/U/N)$ wird beschrieben, dass in Zustand $q \in Q$ ein Zeichen $a \in \Gamma$ gelesen, ein Zeichen $a' \in \Gamma$ auf das Papier geschrieben und ein Feld nach links (L), rechts (R), oben (O), unten (U) gegangen oder auf dem aktuellen Feld stehen geblieben wird (N). Zuletzt wechselt die Maschine in den neuen Zustand $q' \in Q$.
In tabellarischer Form wird das Programm einer zweidimensionalen Turingmaschine wie in Abbildung 2.3 dargestellt. Die doppelten Striche dienen der besseren Lesbarkeit. Sie markieren die Teile des Programms, welche zusammen gehören.

δ		x_0			x_1			x_2		$\cdots$		x_n	
q_0	q_{00}	y_{00}	n_{00}	q_{01}	y_{01}	n_{01}	q_{02}	y_{02}	n_{02}	$\cdots$	q_{0n}	y_{0n}	n_{0n}
q_1	q_{10}	y_{10}	n_{10}	q_{11}	y_{11}	n_{11}	q_{12}	y_{12}	n_{12}	$\cdots$	q_{1n}	y_{1n}	n_{1n}
q_2	q_{20}	y_{20}	n_{20}	q_{21}	y_{21}	n_{21}	q_{22}	y_{22}	n_{22}	$\cdots$	q_{2n}	y_{2n}	n_{2n}
$\vdots$													
q_m	q_{m0}	y_{m0}	n_{m0}	q_{m1}	y_{m1}	n_{m1}	q_{m2}	y_{m2}	n_{m2}	$\cdots$	q_{mn}	y_{mn}	n_{mn}

mit

- $q_{start} = q_0, q_{stop} = q_m, q_{i,j} \in Q$ für alle i, j,
- $Q = \{q_0, \ldots, q_m\}$,
- $\Sigma \subseteq \Gamma = \{x_0, \ldots, x_n, y_{00}, \ldots, y_{mn}\}$ und
- $n_{00}, \ldots, n_{nm} \in \{L, R, N, O, U\}$.

Abbildung 2.3: Tabellarische Darstellung eines Turingprogramms auf einem Papier

Zu Beginn der Arbeit einer solchen Maschine befindet sich diese im Startzustand q_{start}. Das Turingpapier wird mit der Eingabe initialisiert, das heißt, die Kästchen des Papiers werden beschriftet. Alle Kästchen, welche nicht zur Eingabe gehören, werden standardmäßig mit einem Blank (B) beschriftet. Am Ende einer Berechnung steht das Ergebnis als Ausgabe der Turingmaschine auf dem Turnigpapier. War die Berechnung erfolgreich, das heißt die Turingmaschine akzeptiert die Eingabe, befindet sie sich im Endzustand q_{stop}.
Eine Berechnung einer zweidimensionalen Turingmaschine wird wie folgt veranschaulicht: Die aktuelle Beschriftung des Turingpapiers wird mit Hilfe eines karierten Blatt Papiers dargestellt. Die Position des Kopfs der Maschine wird deutlich gemacht, indem das Symbol, auf welchem sich der Kopf befindet, durch eine Box (□) erkenntlich gemacht wird. Der aktuelle Zustand, in dem sich die Maschine befindet, wird dem Turingpapier vorangestellt. Eine Berechnung wird durch die wiederholte Anwendung dieser Darstellung beschrieben. Abbildung 2.4 stellt einen Übergang $\delta(q,a) = (q',b,R)$, einen Berechnungsschritt einer zweidimensionalen Turingmaschine, dar.

$$
q \begin{array}{c|c|c|c|c|c}
 & B & B & B & B & \\ \hline
 & B & \boxed{a} & b & B & \\ \hline
 & B & c & d & B & \\ \hline
 & B & B & B & B &
\end{array}
\quad \rightarrow q' \quad
\begin{array}{c|c|c|c|c|c}
 & B & B & B & B & \\ \hline
 & B & a & \boxed{b} & B & \\ \hline
 & B & c & d & B & \\ \hline
 & B & B & B & B &
\end{array}
$$

Abbildung 2.4: Berechnungsschritt einer zweidimensionalen Turingmaschine

Zweidimensionale Turingmaschine arbeiteten mit einem zweidimensionalen Turingpapier, welches in vier Richtungen unbeschränkt ist. Das Konzept dieser Turingmaschine kann bis in beliebige Dimensionen erweitert werden. In diesem Fall spricht man von *mehrdimensionalen* Turingmaschinen. Allerdings ist der dann zur Verfügung stehende Befehlssatz nicht mehr übersichtlich.

Im Folgenden wird ein weiteres zweidimensionales Berechnungsmodell vorgestellt, welches von John von Neumann u.a. entwickelt wurde.

2.1.2 Paralleles zweidimensionales Berechnungsmodell

Bei zweidimensionalen Turingmaschinen ist der Lese-/Schreibkopf der Maschine, der sogenannte *Berechnungsagent*, jeweils nur an einer Stelle aktiv. Diese Einschränkung wird bei parallelen Modellen aufgehoben, das heißt, dass der Berechnungsagent zu jedem Zeitpunkt an jeder Stelle aktiv ist. Ein bekanntes paralleles zweidimensionales Berechnungsmodell ist ein *zellulärer Automat*. Zelluläre Automaten repräsentieren komplexe mathematische Systeme. Sie sind sehr nützlich, um das dynamische Verhalten realer Systeme darzustellen wie das Verhalten physikalischer Flüssigkeiten, neuronaler Netzwerke und molekularer Systeme [Ila01]. Zelluläre Automaten werden charakterisiert durch lokale Interaktionen, welche eine parallele Form der Entwicklung der Zellen beschreibt. Erste Versuche zur Einführung zellulärer Automaten [Cod68] wurden Ende der 50er Jahre von John von Neumann (1903-1957) und Stanislaw Ulam (1909-1984) vorgenommen. Sie wollten ein Computersystem entwickeln, welches Phänomene der Biologie modelliert und sich wie ein Lebewesen selbst reproduzieren kann.
Ein zellulärer Automat ist ein diskretes dynamisches System, das sich durch die wiederholte Anwendung einfacher deterministischer Regeln entwickelt. Er besteht aus einer regelmäßigen Anordnung von gleichartigen Zellen. Jede Zelle kann eine endliche Anzahl von Werten (=Zuständen) annehmen und hat eine begrenzte Anzahl von Nachbarzellen, die sie beeinflussen können. Das Muster des gesamten zellulären Automaten ändert sich in einzelnen Schritten mit Verstreichen einer diskreten Zeit. Dies bedeutet, dass sich in einem solchen Modell *parallel* für alle Zellen der Zustand ändert. Die einzelnen Schritte werden durch eine Reihe von Übergangsregeln bestimmt, die für alle Zellen gelten.

Formal lässt sich ein zellulärer Automat wie folgt angeben (vgl. [Vol79]):

Definition 2.1.2 *Ein* zellulärer Automat *ist gegeben durch ein 4-Tupel* $ZA = (R, N, Z, \delta)$ *mit*

- *einem Zellraum* R*,*
- *einer endlichen Nachbarschaft* $N_r \subseteq R$ *mit* $|N_r| = n$ *für jedes* $r \in R$*,*
- *einer endlichen, nichtleeren Zustandsmenge* Z
- *und der Zustandsübergangssfunktion* $\delta : Z \times Z^n \to Z$*.*

Die einzelnen Elemente eines zellulären Automaten werden im Folgenden beschrieben:

1. Der Zellraum
   Die Elemente des *Zellraums* R werden *Zellen* genannt. In der Regel werden eindimensionale (lineare Kette) oder zweidimensionale Räume betrachtet. Der Zellraum stellt eine diskrete Menge von Zellen dar. Seine Geometrie wird durch die *Gitterstruktur* des Zellraums festgelegt, welche dreieckig, rechteckig oder hexagonal sein kann (vgl. Abbildung 2.5). Da hier nur der zweidimensionale, rechteckige Fall betrachtet werden soll, entspricht der Zellraum einem Rechteck, welcher durch $R = \{(i,j)|i,j \in \mathbb{N}\}$ definiert ist. Eine Zelle des Zellraums wird durch das Tupel (i,j) angegeben.

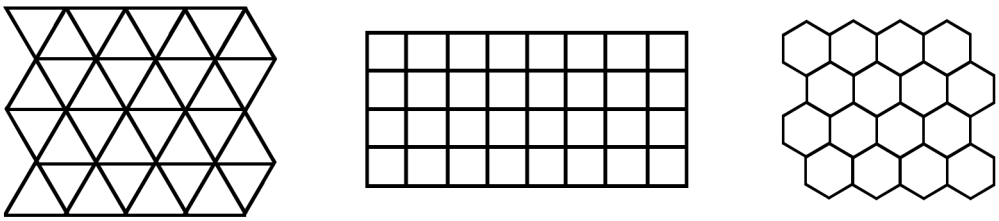

Abbildung 2.5: Zellulärer Automat: Dreieckiges Gitter - Rechteckiges Gitter - Hexagonales Gitter

2. Die Nachbarschaft der Zellen
   Die *Nachbarschaft* einer Zelle besteht aus einer endlichen Menge der Ordnung n von Zellen des Zellraums. Alle Zellen besitzen dieselbe Nachbarschaft. Man unterscheidet im zweidimensionalen rechteckigen Fall zwischen zwei Nachbarschaften (vgl. Abbildung 2.6):

   - Moore-Nachbarschaft:
     $N_{i,j} = \{(k,l) \in R|(k = i, l = j \pm 1) und (k = i \pm 1, l = j)\}$
     Dies beschreibt diejenigen Zellen (k,l) als Nachbarzellen einer Zelle (i,j), welche in Abbildung 2.6 links grau gefärbt sind.
   - Von-Neumann-Nachbarschaft:
     $N_{i,j} = \{(k,l) \in R||k-i| \leq 1 und |l-j| \leq 1\}$.
     Dies beschreibt diejenigen Zellen (k,l) als Nachbarzellen einer Zelle (i,j), welche in Abbildung 2.6 rechts grau gefärbt sind.

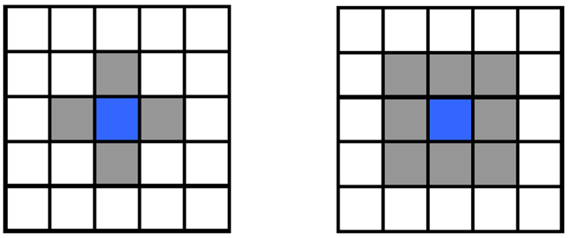

Abbildung 2.6: Nachbarschaften eines zweidimensionalen zellulären Automaten

3. Die Zustände
   Die Elemente der Zustandsmenge Z heißen Zustände. Diese können mit ganzen Zahlen durchnumeriert werden. Am häufigsten werden Automaten mit nur zwei Zuständen definiert, die man als 0 oder 1 bzw. „tot“ oder „lebendig“ interpretieren kann. Jede Zelle besitzt zu jedem Zeitpunkt genau einen Zustand $z \in Z$. Ihr Folgezustand ist abhängig vom eigenen Zustand und den Nachbarschaftszuständen.

4. Die Übergangsregeln
   Zustandsänderungen geschehen in diskreten Zeitschritten, wobei die Entwicklung einer Zelle nur von ihrem Zustand und von den Zuständen ihrer Nachbarn abhängt. Daraus ergibt sich auch die Form $\delta : Z \times Z^n \to Z$. Der Input eines Übergangs einer Zelle ist ihr eigener Zustand und die Zustände ihrer Nachbarn, ein n-Tupel von Zuständen. Der Output liefert den Folgezustand. Diese Übergangsregeln werden auf alle Zellen gleichzeitig angewendet. Bei p Zellen mit je q Zuständen gibt es p^q mögliche Zustände für den gesamten Automaten. Es lassen sich p^{p^q}-Übergänge als Spielregeln für einen Automaten mit p Zellen und q Zuständen definieren. Man sieht, dass sich eine große Anzahl möglicher zellulärer Automaten formulieren lässt, welche das Zusammenleben von Zellen beschreiben.

Ein berühmtes Beispiel für einen zellulären Automaten ist ein biologisches Modell namens „Game of Life“ welches in Kapitel 2.2.3 „Offene Probleme und Unlösbarkeit“ Beispiel 2.2.15 genauer beschrieben wird. Dieses gibt Einblicke in die Arbeitsweise eines zellulären Automaten.

2.2 Zweidimensionale Berechenbarkeit

Im Rahmen der Berechnbarkeitstheorie untersucht die Theoretische Informatik, welche Probleme lösbar sind, also mit Hilfe einer Maschine gelöst werden können. Die Churchsche These trifft Aussagen über die Fähigkeiten eines Rechenmodells. Sie lautet: *„Die Klasse der Turing-berechenbaren Funktionen ist genau die Klasse der intuitiv berechenbaren Funktionen“* [Tho99][Sip97]. Diese Aussage impliziert, dass alle Maschinen genau die gleiche Klasse von Problemen lösen können und dass es keine Maschine gibt, die intuitiv einem Algorithmus entspricht und berechnungsstärker ist. Dies ist formal nicht beweisbar, da der Begriff *intuitiv berechenbare Funktion* nicht exakt formalisiert werden kann. Man versteht darunter alle Funktionen, die prinzipiell auch von einem Menschen ausgerechnet werden könnten. Die These wird in der Informatik allgemein akzeptiert.

Im Folgenden wird eine Definition des Begriffs *Turing-berechenbar* gegeben und dieser anhand einiger Beispiele diskutiert. Die zugrundeliegenden Berechnungsmodelle sind die zweidimensionalen Modelle, welche in Kapitel 2.1 „Zweidimensionale Berechnungsmodelle“ vorgestellt wurden. Die folgenden Berechnungsmodelle verwenden als Objekte zweidimensionale Wörter bzw. Tabellen bestehend aus Symbolen.

2.2.1 Definition und Beispiele

Definition 2.2.1 *Eine Funktion $f : X \to Y$ heißt* Turing-berechenbar*, falls es eine Turingmaschine gibt, die f berechnet, d.h. die bei einer Eingabe $x \in X$ eines Objektbereichs X als Ausgabe $f(x) \in Y$ eines Objektbereichs Y liefert.*

Lösbare Probleme sind solche, welche durch die Angabe einer Turingmaschine gelöst werden können [Tho99]. Die Klasse der lösbaren Probleme ist sehr mächtig. Aus diesem Grund werden im Folgenden lösbare Probleme angegeben, deren Lösungen mit Hilfe einer zweidimenaionalen Turingmaschine angegeben werden können. Es werden für die einzelnen Beispiele die Turingmaschinen formal definiert, ihre Zustandsüberführungsfunktion sprachlich beschrieben und tabellarisch als Programm festgehalten. Zuletzt wird eine Beispielberechnung durchgeführt.

Arithmetik

Wie in Kapitel 1.3.1 „Bleistift und Papier“ vorgestellt sind gerade die Berechnungsmodelle Addition, Subtraktion, Multiplikation und Division für die Schule interessant. Die Berechnungsmodelle Addition und Subtraktion werden mit Hilfe einer zweidimensionalen Turingmaschine vorgestellt. Auf die Berechnungsmodelle Multiplikation und Division wird verzichtet.

Beispiel 2.2.2 (Addition mit einer zweidimensionalen Turingmaschine)
Die Berechnung der Summe zweier Binärzahlen wird auf zwei Summanden beschränkt. Um mehrere Summanden zu addieren, wird die Berechnung einfach mehrmals angewandt.
Gegeben ist eine zweidimensionale Turingmaschine $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$ mit $Q = \{q_{start} = u_0, u_1, o_0, o_1, o_2, s_0, s_1, s_2, s_3, q_{stop}\}$ und $\Sigma = \{0,1\}$, $\Gamma = \{0,1,B\}$. Die Zustandsüberführungsfunktion δ kann der Abbildung 2.7 entnommen werden.
Das Papier der Turingmaschine wird folgendermaßen initialisiert: Als Eingabe erhält T zwei natürliche Zahlen x und y in Binärkodierung, welche untereinander auf das Papier geschrieben werden, wobei die kürzere von beiden mit führenden Nullen aufgefüllt wird (vgl. hierzu Abb. 2.9). Der Kopf der Turingmaschine befindet sich auf dem rechten Zeichen der unteren Binärzahl. Die Maschine startet im Startzustand u_0 und endet im Endzustand q_{stop}. Am Ende der Berechnung steht das Ergebnis $x+y$ auf dem Papier von T.

Folgende Auflistung beschreibt die Vorgehensweise der Berechnung:

$u_{0,1}$ Diese beiden Zustände drücken aus, ob bereits eine 1 verarbeitet wurde oder nicht. Die Turingmaschine startet in u_0, das heißt bisher hat noch keine Addition stattgefunden. Liest T nun eine 0 in der <u>u</u>nteren Zeile, wechselt T von u_0 nach o_0 und vollzieht eine Kopfbewegung in die <u>o</u>bere Zeile. Liest T jedoch eine 1 in der <u>u</u>nteren Zeile, wechselt T von u_0 nach o_1 und vollzieht anschließend die Kopfbewegung in die <u>o</u>bere Zeile. Befindet sich die Turingmaschine bereits in u_1, im vorherigen Berechnungsschritt ist also ein Übertrag entstanden, wechselt sie beim Lesen einer 0 von u_1 nach o_1 und beim Lesen einer 1 von u_1 nach o_2. Bei 0 wechselt T von u_i nach o_i und bei 1 von u_i nach o_{i+1}. Anschließend wird die Zahl in der oberen Zeile untersucht. Bei allen Übergängen werden jeweils die gelesenen Zeichen durch ein Blank B überschrieben. Liest die Turingmaschine in diesen beiden Zuständen jedoch ein B, ist die Berechnung zu Ende. Es muss

nur noch ein eventuell entstandener Übertrag des letzten Berechnungsschrittes verarbeitet werden, die Turingmaschine befindet sich also in Zustand u_1. Sie schreibt noch eine 1 auf das Band und wechselt anschließend nach q_{stop}.

$o_{0,1,2}$ Hier wird die obere Zahl der entsprechenden Spalte der Berechnung verarbeitet. Befindet sich der Kopf der Turingmaschine auf einer 0, wechselt die Maschine von o_i nach s_{i+1}. Befindet er sich auf einer 1, wechselt die Maschine von o_i nach s_i und vollzieht jeweils eine Kopfbewegung nach unten, wobei das Zeichen der oberen Zeile durch ein Blank B überschrieben wird.

$s_{0,1,2,3}$ In dieser Zustandsgruppe schreibt die Turingmaschine das entsprechende Ergebnis jeder Spalte. Befindet sich die Turingmaschine in s_0, wird eine 0 geschrieben und nach u_0 gewechselt. In s_1 wird eine 1 geschrieben und nach u_0 gewechselt, in s_2 wird eine 0 geschrieben und nach u_1 gewechselt, und in s_3 wird eine 1 geschrieben und nach u_1 gewechselt, um dann mit der Berechnung in der nächsten Spalte fortzufahren.

Die genaue Durchführung einer Addition kann der Berechnung in Abbildung 2.8 entnommen werden.

δ		0			1			B	
u_0	o_0	B	O	o_1	B	O	q_{stop}	B	N
u_1	o_1	B	O	o_2	B	O	q_{stop}	1	N
o_0	s_0	B	U	s_1	B	U	-	-	-
o_1	s_1	B	U	s_2	B	U	-	-	-
o_2	s_2	B	U	s_3	B	U	-	-	-
s_0	-	-	-	-	-	-	u_0	0	L
s_1	-	-	-	-	-	-	u_0	1	L
s_2	-	-	-	-	-	-	u_1	0	L
s_3	-	-	-	-	-	-	u_1	1	L

Abbildung 2.7: Programm einer zweidimensionalen Turingmaschine zur Berechnung der Addition zweier Binärzahlen

u_0

B	B	B	B	B
B	B	1	1	B
B	B	1	$\boxed{1}$	B
B	B	B	B	B

$\rightarrow o_2$

B	B	B	B	B
B	B	$\boxed{1}$	B	B
B	B	B	0	B
B	B	B	B	B

$\rightarrow o_1$

B	B	B	B	B
B	B	1	$\boxed{1}$	B
B	B	1	B	B
B	B	B	B	B

$\rightarrow s_3$

B	B	B	B	B
B	B	B	B	B
B	B	$\boxed{\text{B}}$	0	B
B	B	B	B	B

$\rightarrow s_2$

B	B	B	B	B
B	B	1	B	B
B	B	1	$\boxed{\text{B}}$	B
B	B	B	B	B

$\rightarrow u_1$

B	B	B	B	B
B	B	B	B	B
B	$\boxed{\text{B}}$	1	0	B
B	B	B	B	B

$\rightarrow u_1$

B	B	B	B	B
B	B	1	B	B
B	B	$\boxed{1}$	0	B
B	B	B	B	B

$\rightarrow q_{stop}$

B	B	B	B	B
B	B	B	B	B
$\boxed{\text{B}}$	1	1	0	B
B	B	B	B	B

Abbildung 2.8: Beispielberechnung einer zweidimensionalen Turingmaschine zur Addition von $3+3=6$ in Binärkodierung

Beispiel 2.2.3 (Subtraktion mit einer zweidimensionalen Turingmaschine)

Bei der Subtraktion nach Schulmethode mit Hilfe der zweidimensionalen Turingmaschine wird nur die Subtraktion eines kleineren Minuenden von einem größeren Subtrahenden betrachtet. Bei mehreren Minuenden von einem Subtrahenden muss die Subtraktion mehrmals anwendet werden.

Gegeben ist eine zweidimensionale Turingmaschine $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$ mit $Q = \{q_{start} = u_0, u_1, o_0, o_1, o_2, s_0, s_1, s_2, s_3, q_{stop}\}$ und $\Sigma = \{0,1\}$, $\Gamma = \{0,1,B\}$. Die Zustandsüberführungsfunktion δ kann der Abbildung 2.10 entnommen werden.

Das Papier der Turingmaschine wird folgendermaßen initialisiert: Als Eingabe erhält die Turingmaschine T zwei natürliche Zahlen x und y in Binärkodierung, wobei die beiden Binärzahlen wieder untereinander auf das Papier geschrieben werden, indem die untere Zahl y, und zwar die kleiner Zahl von beiden, mit führenden Nullen aufgefüllt wird (vgl. hierzu Abb. 2.9). Der Kopf der Turingmaschine befindet sich auf

dem rechten Zeichen der unteren Binärzahl. Die Maschine startet im Startzustand u_0 und endet im Endzustand q_{stop}. Am Schluss der Berechnung steht das Ergebnis $x - y$ auf dem Band der Turingmaschine.

Folgende Auflistung beschreibt die Vorgehensweise der Berechnung:

$u_{0,1}$ Die Turingmaschine T startet im Zustand u_0, befindet sich also in der unteren Zeile und hat bisher kein Zeichen gelesen bzw. im vorherigen Berechnungsschritt ist kein Übertrag entstanden. Befindet sich die Maschine in u_1 ist im letzten Berechnungschritt ein Übertrag entstanden. Wird nun eine 0 gelesen, wechselt die Maschine von u_i nach o_i, bei einer 1 von u_i nach o_{i+1}, wobei das gelesene Zeichen durch ein Blank B überschrieben und in die obere Zeile gesprungen wird. Liest die Turingmaschine jedoch ein B, ist die Berechnung beendet und es wird nach q_{stop} gewechselt.

$o_{0,1,2}$ In dieser Zustandsgruppe wird die obere Zahl der Rechnung verarbeitet. Ist diese eine 0, wechselt die Maschine von o_i nach s_i. Ist sie eine 1, wird von o_i nach s_{i+1} gewechselt. Dabei wird das gelesene Zeichen durch ein B überschrieben und eine Kopfbewegung wieder in die untere Zeile vollzogen.

$s_{0,1,2,3}$ Hier wird die eigentliche Berechnung durchgeführt. Befindet sich die Turingmaschine in s_0, schreibt sie eine 0 auf das Papier und wechselt nach u_0. In s_1 wird eine 1 geschrieben und nach u_0 gewechselt, in s_2 wird eine 0 geschrieben und nach u_1 gewechselt, und in s_3 wird eine 1 geschrieben und nach u_1 gewechselt, um dann mit der Berechnung in der nächsten Spalte fortzufahren.

Die genaue Durchführung der Berechnung einer Subtraktion kann Abbildung 2.11 entnommen werden.

	B	B	B	B	B	B	
	B	1	0	1	0	B	
u_0	B	0	0	1	$\boxed{1}$	B	
	B	B	B	B	B	B	

Abbildung 2.9: Initialisierung des Turingpapiers bei Addition und Subtraktion

δ		0			1			B	
u_0	o_0	B	O	o_1	B	O	q_{stop}	B	N
u_1	o_1	B	O	o_2	B	O	-	-	-
o_0	s_0	B	U	s_1	B	U	-	-	-
o_1	s_2	B	U	s_0	B	U	-	-	-
o_2	s_3	B	U	s_2	B	U	-	-	-
s_0	-	-	-	-	-	-	u_0	0	L
s_1	-	-	-	-	-	-	u_0	1	L
s_2	-	-	-	-	-	-	u_1	1	L
s_3	-	-	-	-	-	-	u_1	0	L

Abbildung 2.10: Programm einer zweidimensionalen Turingmaschine zur Subtraktion zweier Binärzahlen

u_0

B	B	B	B	B
B	1	1	0	B
B	0	1	$\boxed{1}$	B
B	B	B	B	B

$\rightarrow o_1$

B	B	B	B	B
B	1	1	$\boxed{0}$	B
B	0	1	B	B
B	B	B	B	B

$\rightarrow s_2$

B	B	B	B	B
B	1	1	B	B
B	0	1	$\boxed{\text{B}}$	B
B	B	B	B	B

$\rightarrow u_1$

B	B	B	B	B
B	1	1	B	B
B	0	$\boxed{1}$	1	B
B	B	B	B	B

$\rightarrow o_2$

B	B	B	B	B
B	1	$\boxed{1}$	B	B
B	0	B	1	B
B	B	B	B	B

$\rightarrow s_2$

B	B	B	B	B
B	1	B	B	B
B	0	$\boxed{\text{B}}$	1	B
B	B	B	B	B

$\rightarrow u_1$

B	B	B	B	B
B	1	B	B	B
B	$\boxed{0}$	1	1	B
B	B	B	B	B

$\rightarrow o_1$

B	B	B	B	B
B	$\boxed{1}$	B	B	B
B	B	1	1	B
B	B	B	B	B

$\rightarrow s_0$

B	B	B	B	B
B	B	B	B	B
B	$\boxed{\text{B}}$	1	1	B
B	B	B	B	B

$\rightarrow u_0$

B	B	B	B	B
B	B	B	B	B
$\boxed{\text{B}}$	0	1	1	B
B	B	B	B	B

$\rightarrow q_{stop}$

B	B	B	B	B
B	B	B	B	B
$\boxed{\text{B}}$	0	1	1	B
B	B	B	B	B

Abbildung 2.11: Beispielberechnung einer zweidimensionalen Turingmaschine zur Subtraktion von $6-3=3$ in Binärkodierung

Eine Weitere Anwendung einer zweidimensionalen Turingmaschine wird vorgestellt, um einen Einblick in ihre vielseitige Verwendbarkeit und ihre zahlreichen Möglichkeiten zur Problemlösung zu bekommen.

Pixel-Bild-Analyse

Eine Anwendung der zweidimensionalen Turingmaschine neben arithmetischen Problemen stellen *Pixel-Bild-Analysen* dar. Das karierte Turingpapier kann als Pixel-Bild aufgefasst werden, welches auf seine Struktur hin analysiert werden soll. Man interessiert sich für das Muster oder eine abgebildete Figur eines Bildes. Es wird eine Pixelbildanalyse durch die Angabe einer zweidimensionalen Turingmaschine vorgestellt:

Beispiel 2.2.4 (Ist das quadratische Bild regelmäßig gemustert?)
Abbildung 2.12 zeigt links ein mit Stern ($\star$) und Kringel ($\circ$) gemustertes Bild dargestellt durch das Turingpapaier einer zweidimensionalen Turingmaschine. Diese soll untersuchen, ob das Bild regelmäßig gemustert ist, das heißt, ob sich Stern und Kringel abwechseln.
Gegeben ist eine zweidimensionale Turingmaschine $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$ mit $Q = \{q_{start}, k_r, \ldots, q_{kr}, q_{stop}\}$ (vgl. 2.12 rechts) und $\Sigma = \{\star, \circ\}$, $\Gamma = \{\star, \circ, B\}$. Die Zustandsüberführungsfunktion dieser Problemstellung ist in Abbildung 2.12 rechts angegeben. Das Turingpapier wird mit dem Pixelbild initialisiert. Es wird vorausgesetzt, dass sich der Kopf der Turingmaschine auf dem oberen linken Pixel des Bildes befindet. Die Maschine startet im Startzustand q_{start} und endet in dem akzeptierenden Endzustand q_{stop}, falls das Bild regelmäßig gemustert ist. Hierzu werden die Reihen von links nach rechts (s'_r, k'_r) und von rechts nach links (k'_l, s'_l) durchgelaufen, wobei sich Stern- und Kringelzustand abwechseln. Wird ein Blank (B) gelesen, kann in die nächste Zeile gesprungen und in einen neuen Zustand gewechselt werden ($q_{sl,kl,sr,kr}$). Um das Ende des Quadrates feststellen zu können, wird noch eine Auffangstufe in den Zuständen eingebaut ($k_{l,r}, s_{l,r}$). Hier wird zu Beginn jeder Zeile überprüft, ob das gelesene Zeichen ein B ist. Ist dies der Fall, befindet sich der Kopf unterhalb des Bildes. Die Überprüfung der Struktur ist beendet und die Maschine wechselt in den Endzustand q_{stop}.

B	B	B	B	B	B	B
B	$\circ$	$\star$	$\circ$	$\star$	$\circ$	B
B	$\star$	$\circ$	$\star$	$\circ$	$\star$	B
B	$\circ$	$\star$	$\circ$	$\star$	$\circ$	B
B	$\star$	$\circ$	$\star$	$\circ$	$\star$	B
B	$\circ$	$\star$	$\circ$	$\star$	$\circ$	B
B	B	B	B	B	B	B

δ	$\circ$	$\star$	B
q_{start}	$(s_r, \circ, R)$	$(k_r, \star, R)$	-
k_r	$(s'_r, \circ, R)$	-	(q_{stop}, B, N)
s_r	-	$(k'_r, \star, R)$	(q_{stop}, B, N)
k_l	$(s'_l, \circ, L)$	-	(q_{stop}, B, N)
s_l	-	$(k'_l, \star, L)$	(q_{stop}, B, N)
k'_r	$(s'_r, \circ, R)$	-	(q_{sl}, B, U)
s'_r	-	$(k'_r, \star, R)$	(q_{kl}, B, U)
k'_l	$(s'_l, \circ, L)$	-	(q_{sr}, B, U)
s'_l	-	$(k'_l, \star, L)$	(q_{kr}, B, U)
q_{sl}	-	-	(s_l, B, L)
q_{kl}	-	-	(k_l, B, L)
q_{sr}	-	-	(s_r, B, R)
q_{kr}	-	-	(s_r, B, R)

Abbildung 2.12: Pixel-Bild und Zustandsüberführungsfunktion: Ist das Bild regelmäßig gemustert mit Stern und Kringel?

2.2.2 Reduktion auf das eindimensionale Modell

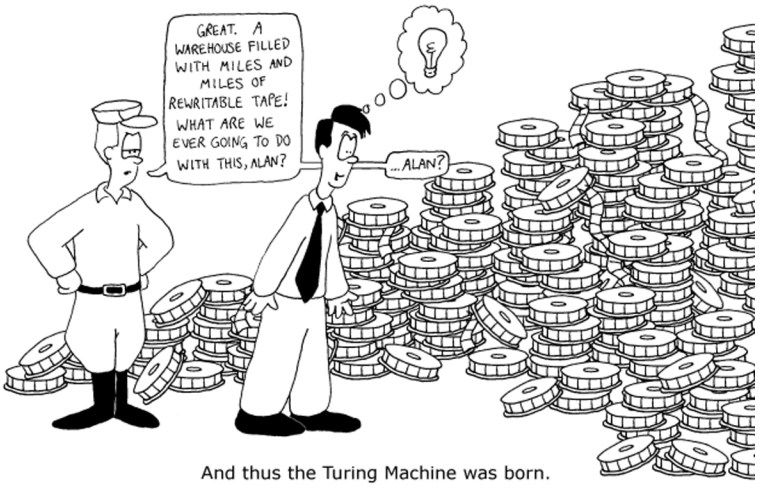

Abbildung 2.13: Die Geburt der Turingmaschine

Die von Alan Turing konzipierte Turingmaschine verzichtet, wie er in [Tur36] zitiert, auf eine zweite Dimension: „*I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, i.e., on a tape divided into squares*“ (vgl. Abbildung 2.13). Turing erwähnt ganz nebenbei, dass es ausreicht, ein eindimensionales Modell zu betrachten. Den Beweis hierfür übergeht er mit seiner Formulierung „it will be agreed“, da das eindimensionale Modell reicht, um das Zweidimensionale zu simulieren. Eigentlich wäre hier ein Beweis nötig. Dieser wird im Folgenden nachgeholt, um Turings Lücke zu schließen. Hiefür wird zunächst das eindimensionale Modell von Turing eingeführt, um auch Vergleiche zwischen dem eindimensionalen Modell und dem Zweidimensionalen ziehen zu können. Das eindimensionale Modell hat schließlich den Vorteil, dass Beweise über die Unentscheidbarkeit aller Turingmaschinen einfacher darstellbar sind, da der Befehlssatz der eindimensionalen Turingmaschine einfacher ist.

Turings eindimensionales Berechnungsmodell besitzt einen Lese-/Schreibkopf, der sich auf einem nach beiden Seiten unbegrenzten Arbeitsband, welches in Felder unterteilt ist. Dieser kann sich in zwei Richtungen (links, rechts) hin und her bewegen kann (vgl. hierzu Abbildung 2.14). Nachfolgend wird die eindimensionale Turingmaschine formal definiert (vgl. [Tho99]).

Definition 2.2.5 *Eine* eindimensionale Turingmaschine *ist gegeben durch ein 7-Tupel* $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$ *mit*

- *einer endlichen Zustandsmenge* Q*,*
- *einem Eingabealphabet* Σ*,*
- *einem Bandalphabet* $\Gamma \supseteq \Sigma$*,*
- *einem ausgezeichneten Blanksymbol* $B \in \Gamma \setminus \Sigma$*,*
- *einem Anfangszustand* $q_{start} \in Q$*,*
- *einem Endzustand* $q_{stop} \in Q$
- *und der Übergangsfunktion* $\delta : (Q \times \Gamma) \rightarrow (Q \times \Gamma \times \{L, R, N\})$.

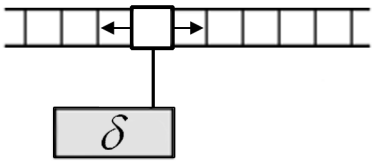

Abbildung 2.14: Turingmaschine mit eindimensionalem Arbeitsband

Die Zustandsüberführungsfunktion δ der eindimensionalen Turingmaschine ist analog zu der Zustandsüberführungsfunktion einer zweidimensionalen Turingmaschine zu interpretieren (vgl. 2.1.1 „Die zweidimensionale Turingmaschine“). Die einzige Reduktion, die vollzogen wird, ist, dass der Kopf nur noch zwei Möglichkeiten zur Bewegung erhält. Die tabellarische Darstellung der Zustandsüberführungsfunktion δ einer eindimensionalen Turingmaschine kann Abbildung 2.15 entnommen.

Zu Beginn der Arbeit einer eindimensionalen Turingmaschine befindet sich diese im Startzustand q_{start}. Das Turingband wird mit der Eingabe initialisiert, das heißt, die Felder des Turingbandes werden beschriftet. Alle Felder des Bandes, welche nicht zur Eingabe gehören, werden standardmäßig mit einem Blank (B) beschriftet. Am Ende einer Berechnung steht das Ergebnis als Ausgabe der Turingmaschine auf dem Turingband. War die Berechnung erfolgreich, das heißt die Turingmaschine akzeptiert die Eingabe, befindet sie sich im Endzustand q_{stop}.

δ	x_0	x_1	x_2	$\cdots$	x_n
q_0	(q_{00}, y_{00}, n_{00})	(q_{01}, y_{01}, n_{01})	(q_{02}, y_{02}, n_{02})	$\cdots$	(q_{0n}, y_{0n}, n_{0n})
q_1	(q_{10}, y_{10}, n_{10})	(q_{11}, y_{11}, n_{11})	(q_{12}, y_{12}, n_{12})	$\cdots$	(q_{1n}, y_{1n}, n_{1n})
q_2	(q_{20}, y_{20}, n_{20})	(q_{21}, y_{21}, n_{21})	(q_{22}, y_{22}, n_{22})	$\cdots$	(q_{2n}, y_{2n}, n_{2n})
$\vdots$					
q_m	(q_{m0}, y_{m0}, n_{m0})	(q_{m1}, y_{m1}, n_{m1})	(q_{m2}, y_{m2}, n_{m2})	$\cdots$	(q_{mn}, y_{mn}, n_{mn})

mit

- $q_{start} = q_0, q_{stop} = q_m, q_{i,j} \in Q$ für alle i, j,
- $Q = \{q_0, \ldots, q_m\}$,
- $\Sigma \subseteq \Gamma = \{x_0, \ldots, x_n, y_{00}, \ldots, y_{mn}\}$ und
- $n_{00}, \ldots, n_{nm} \in \{L, R, N\}$.

Abbildung 2.15: Tabellarische Darstellung eines Turingprogramms einer eindimensionalen Turingmaschine

Eine Berechnung einer eindimensionalen Turingmaschine wird folgendermaßen veranschaulicht: Der Kopf der Maschine wird durch einen Pfeil dargestellt, der sich unter dem Zeichen befindet, auf welchem der Kopf der Maschine steht. Der aktuelle Zustand, in welchem sich die Maschine befindet, wird dem Turingband rechts angehängt. Eine Berechnung wird durch wiederholte Anwendung dieser Darstellung beschrieben. Abbildung 2.16 stellt einen Übergang $\delta(q,a) = (q', a', L)$, einen Berechnungsschritt einer eindimensionalen Turingmaschine, dar.

$\cdots$	B	B	d	c	b	a	B	B	$\cdots$	q
						$\uparrow$				
$\cdots$	B	B	d	c	b	a'	B	B	$\cdots$	q'
					$\uparrow$					

Abbildung 2.16: Berechnungsschritt einer eindimensionalen Turingmaschine

Um die Arbeitsweise der eindimensionalen Turingmaschine mit der einer Zweidimensionalen vergleichen zu können, wird die Addition zweier Binärzahlen mit einer eindimensionalen Turingmaschine vorgestellt.

Beispiel 2.2.6 (Addition mit einer eindimensionalen Turingmaschine)
Für die eindimensionale Addition zweier natürlicher Zahlen x und y in Binärkodierung werden die beiden Binärzahlen auf das eindimensionale Turingband geschrieben, wobei sie durch ein Trennsymbol (#) voneinander getrennt sind. Außerdem werden die beiden Zahlen durch Auffüllen der kürzeren Binärzahl mit führenden Nullen auf die gleiche Länge gebracht (vgl. hierzu Abb. 2.17). Die Turingmaschine befindet sich zu Beginn der Rechnung im Zustand q_0 und ihr Kopf auf dem rechten Zeichen der rechten Binärzahl.

$\cdots$	B	B	1	1	1	#	0	1	1	B	B	$\cdots$	q_0
									$\uparrow$				

Abbildung 2.17: Initialisierung des Turingbandes bei Addition zweier Binärzahlen

Gegeben ist eine eindimensionale Turingmaschine $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$ mit $q_{start} = q_0$, $\Sigma = \{0, 1, \#\}$ und $\Gamma = \{0, 1, \hat{0}, \hat{1}, \#, B\}$. Die verwendeten Zustände und die Zustandsüberführungsfunktion δ können der Abbildung 2.18 entnommen werden. Die Maschine arbeitet mit 18 Zuständen. Sie verwendet eine Markierung ($\hat{}$), um die bisher gelesenen Zeichen zu erkennen. Hierfür läuft sie auf dem Turingband hin und her, markiert die Zeichen links und rechts des Trennsymbols (#), um sie anschließend entsprechend zu addieren.

Folgende Auflistung beschreibt die Arbeitsweise der Turingmaschine:

$q_{0/1}$ Diese beiden Zustände drücken aus, welche beiden Zeichen (0 oder 1) der Rechnung bereits gelesen wurden. Befindet sich die Turingmaschine in q_0 bedeutet dies, dass bisher noch keine 1 gelesen wurde, bzw. alle Überträge verarbeitet wurden. Die Rechnung beginnt von vorne. Das nun gelesene Zeichen wird markiert ($\hat{}$) und in den entsprechenden Zustand (l_0, falls eine 0, und l_1, falls eine 1 gelesen wurde) gewechselt. Befindet sich die Turingmaschine in q_1 bedeutet dies, dass noch eine 1 zur Verarbeitung der Rechnung offen ist. Das nun gelesene Zeichen wird ebenfalls markiert ($\hat{}$) und in den entsprechenden Zustand (l_1, falls eine 0, und l_2, falls eine 1 gelesen wurde) gewechselt. Befindet sich der Kopf der Maschine auf dem Zeichen #, ist die Rechnung fast zu Ende. Es wird noch der Übertrag verarbeitet, indem statt # in Zustand q_0 eine 0 und in q_1 eine 1 geschrieben und in den Zustand t gewechselt wird.

$l_{0/1/2}$ Diese Zustandsgruppe verhält sich folgendermaßen: Der Kopf der Turingmaschine befindet sich auf einem unmarkierten Zeichen der rechten Binärzahl auf dem Turingband. Die Turingmaschine läuft nun so lange nach links, bis sie auf das Trennsymbol (#) trifft und wechselt in den entsprechenden Zustand der Zustandsgruppe $l_{0,1,2}$.

$l'_{0,1,2}$ In diesen Zuständen läuft die Turingmaschine, deren Kopf sich nun auf einem Zeichen der linken Binärzahl befindet, solange nach links, bis sie auf ein unmarkiertes Zeichen trifft und markiert (ˆ) dieses. Anschließend wird in den entsprechenden Zustand $r_{0,1,2,3}$ gewechselt. Ist das unmarkiert gefundene Zeichen eine 0, wird von Zustand l'_i nach r_i gewechselt, ist es eine 1, wird von l'_i nach r_{i+1} gewechselt.

$r_{0,1,2,3}$ Hier läuft die Maschine über der linken Binärzahl auf dem Turingband solange nach rechts, bis sie das Trennsymbol (#) liest und wechselt in den entsprechenden Zustand r'_i.

$r'_{0,1,2,3}$ In diesen Zuständen wird die eigentliche Rechnung durchgeführt. Der Zustand, in der sich die Turingmaschine befindet, drückt das Ergebnis der Rechung aus: im einen Fall die Zahl, die geschrieben werden soll ($r'_0 \to 0$ und $r'_1 \to 1$ kein Übertrag, $r'_2 \to 0$ und $r'_0 \to 1$ Übertrag) im anderen Fall, ob bei der Rechnung ein Übertrag entstanden ist. Dementsprechend wird die Rechnung in q_0 (kein Übertrag) oder q_1 (Übetrag) fortgesetzt.

t Die Turingmaschine ist am Ende ihrer Rechnung angekommen. Sie überschreibt hier alle Zeichen mit Blanks (B), bis ebenfalls ein B gelesen wird. Dann wechselt die Turingmaschine in den Endzustand q_{stop}. Das Ergebnis der Rechnung steht auf dem Turingband.

δ	0	1	$\hat{0}$	$\hat{1}$	#	B
q_0	$(l_0,\hat{0},L)$	$(l_1,\hat{1},L)$	-	-	$(t,0,L)$	-
q_1	$(l_1,\hat{0},L)$	$(l_2,\hat{1},L)$	-	-	$(t,1,L)$	-
l_0	$(l_0,0,L)$	$(l_0,1,L)$	-	-	$(l'_0,\#,L)$	-
l_1	$(l_1,0,L)$	$(l_1,1,L)$	-	-	$(l'_1,\#,L)$	-
l_2	$(l_2,0,L)$	$(l_2,1,L)$	-	-	$(l'_2,\#,L)$	-
l'_0	$(r_0,\hat{0},R)$	$(r_1,\hat{1},R)$	$(l'_0,\hat{0},L)$	$(l'_0,\hat{1},L)$	-	-
l'_1	$(r_1,\hat{0},R)$	$(r_2,\hat{1},R)$	$(l'_1,\hat{0},L)$	$(l'_1,\hat{1},L)$	-	-
l'_2	$(r_2,\hat{0},R)$	$(r_3,\hat{1},R)$	$(l'_2,\hat{0},L)$	$(l'_2,\hat{1},L)$	-	-
r_0	-	-	$(r_0,\hat{0},R)$	$(r_0,\hat{1},R)$	$(r'_0,\#,R)$	-
r_1	-	-	$(r_1,\hat{0},R)$	$(r_1,\hat{1},R)$	$(r'_1,\#,R)$	-
r_2	-	-	$(r_2,\hat{0},R)$	$(r_2,\hat{1},R)$	$(r'_2,\#,R)$	-
r_3	-	-	$(r_3,\hat{0},R)$	$(r_3,\hat{1},R)$	$(r'_3,\#,R)$	-
r'_0	$(r'_0,0,R)$	$(r'_0,1,R)$	$(q_0,0,L)$	-	-	-
r'_1	$(r'_1,0,R)$	$(r'_1,1,R)$	$(q_0,1,L)$	$(q_0,1,L)$	-	-
r'_2	$(r'_2,0,R)$	$(r'_2,1,R)$	$(q_1,0,L)$	$(q_1,0,L)$	-	-
r'_3	$(r'_3,0,R)$	$(r'_3,1,R)$	-	$(q_1,1,L)$	-	-
t	-	-	(t,B,L)	(t,B,L)	-	(q_{stop},B,N)

Abbildung 2.18: Programm einer eindimensionalen Turingmaschine zur Addition zweier Binärzahlen

Folgende Beispielberechnung zur Addition zeigt die Arbeitsweise einer eindimensionalen Turingmaschine:

$\cdots$	B	B	1	1	#	1	1	B	B	$\cdots$	q_0
							$\uparrow$				
$\cdots$	B	B	1	1	#	1	$\hat{1}$	B	B	$\cdots$	l_1
						$\uparrow$					
$\cdots$	B	B	1	1	#	1	$\hat{1}$	B	B	$\cdots$	l_1
					$\uparrow$						
$\cdots$	B	B	1	1	#	1	$\hat{1}$	B	B	$\cdots$	l'_1
				$\uparrow$							
$\cdots$	B	B	1	$\hat{1}$	#	1	$\hat{1}$	B	B	$\cdots$	r_2
					$\uparrow$						

$\cdots$	B	B	1	$\hat{1}$	#	1	$\hat{1}$	B	B	$\cdots$	r_2'
						$\uparrow$					
$\cdots$	B	B	1	$\hat{1}$	#	1	$\hat{1}$	B	B	$\cdots$	r_2'
							$\uparrow$				
$\cdots$	B	B	1	$\hat{1}$	#	1	0	B	B	$\cdots$	q_1
						$\uparrow$					
$\cdots$	B	B	1	$\hat{1}$	#	$\hat{1}$	0	B	B	$\cdots$	l_2
					$\uparrow$						
$\cdots$	B	B	1	$\hat{1}$	#	$\hat{1}$	0	B	B	$\cdots$	l_2'
				$\uparrow$							
$\cdots$	B	B	1	$\hat{1}$	#	$\hat{1}$	0	B	B	$\cdots$	l_2'
			$\uparrow$								
$\cdots$	B	B	$\hat{1}$	$\hat{1}$	#	$\hat{1}$	0	B	B	$\cdots$	r_3
				$\uparrow$							
$\cdots$	B	B	$\hat{1}$	$\hat{1}$	#	$\hat{1}$	0	B	B	$\cdots$	r_3
					$\uparrow$						
$\cdots$	B	B	$\hat{1}$	$\hat{1}$	#	$\hat{1}$	0	B	B	$\cdots$	r_3'
						$\uparrow$					
$\cdots$	B	B	$\hat{1}$	$\hat{1}$	#	1	0	B	B	$\cdots$	q_1
					$\uparrow$						
$\cdots$	B	B	$\hat{1}$	$\hat{1}$	1	1	0	B	B	$\cdots$	t
				$\uparrow$							
$\cdots$	B	B	$\hat{1}$	B	1	1	0	B	B	$\cdots$	t
			$\uparrow$								
$\cdots$	B	B	B	B	1	1	0	B	B	$\cdots$	t
		$\uparrow$									
$\cdots$	B	B	B	B	1	1	0	B	B	$\cdots$	q_{stop}
		$\uparrow$									

Abbildung 2.19: Beispielberechnung einer eindimensionalen Turingmaschine zur Addition von $3+3=6$ in Binärkodierung

Abbildung 2.18 und die Beispielberechnung einer Addition in Abbildung 2.19 zeigen, dass die Arbeitsweise der Turingmaschine sehr aufwendig ist, da sie immer wieder auf dem Turingband hin- und herlaufen muss, um die entsprechenden Zeichen, welche addiert werden müssen zu suchen, zu markieren, um anschließend die Summe und den Übertrag berechnen zu können. Die Addition mit Hilfe einer zweidimensionalen Turingmaschine aus Kapitel 2.2 „Zweidimensionale Berechenbarkeit“ Beispiel 2.2.2 läuft nicht hin und her und benutzt keine Markierungsmethode. Die Berechnung wird an Ort und Stelle ausgeführt. Sie liefert einsichtige Bandbeschriftungen und effektive Algorithmen, da sie sich an unserem Handeln auf Papier orientiert. Auch der Zeitbedarf sinkt, da weniger Zuständen benötigt werden, wie der Vergleich von Beispiel 2.2.2 „Addition mit einer zweidimensionalen Turingmaschine“ und Beispiel 2.2.6 „Addition mit einer eindimensionalen Turingmaschine“ zeigt.

Nachstehender Satz zeigt, dass die zweidimensionale Turingmachine äquivalent zu allen anderen Maschinenmodellen ist. Da die zweidimensionale Turingmaschine eine Verallgemeinerung der eindimensionalen Turingmaschine darstellt, können diese beiden Maschinenmodelle die gleiche Klasse von lösbaren Problemen lösen.

Satz 2.2.7 *Eine zweidimensionale Turingmaschine kann durch eine eindimensionale Turingmaschine simuliert werden.*

Gegeben ist eine zweidimensionale Turingmaschine $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$, welche durch eine eindimensionale Turingmaschine $T' = (Q', \Sigma', \Gamma', B, q'_{start}, q'_{stop}, \delta')$ simuliert werden soll. Hierzu wird das zweidimensionale Blatt Papier auf ein eindimensionales Arbeitsband abgebildet, wie dies in Abbildung 2.20 zu sehen ist.

		B	B	a		
q_{start}		B	b	c		
		d	e	B		

$\Downarrow$

$\cdots$	\$	B	B	a	#	B	b	c	#	d	e	B	\$	$\cdots$	q'_{start}

Abbildung 2.20: Abbildung eines Turingpapiers auf ein Turingband

Für die Simulation ist es nötig, dass sich die Übertragung der Inhalte der Zeilen in der zweidimensionalen Turingmaschine auf ein eindimensionales Band an der längsten Zeile orientiert, das heißt, dass die entsprechenden Zeilen in der eindimensionalen Turingmaschine mit Blanks (B) aufgefüllt werden (vgl. hierzu Abbildung 2.20). Zur Veranschaulichung dient ein minimal umgebendes Rechteck (MUR), welches den Inhalt auf dem zweidimensionalen Turingpapier, der Zeichen enthält, einschließt. Desweiteren werden zwei zusätzliche Sonderzeichen benötigt: Das Sonderzeichen (#) trennt die Inhalte der einzelnen Zeilen der zweidimensionalen Turingmachine. Das andere Sonderzeichen ($) markiert das linke und rechte Ende der Bandinschrift der eindimensionalen Turingmaschine, damit erkenntlich wird, wann die Maschine sich aus dem MUR der Bandinschrift herausbewegt. Außerdem werden für die Simualtion zwei Markierungen ($\bar{}$) und ($\hat{}$) der Bandinschrift benötigt. Es gilt $\Sigma' = \Sigma$ und $\Gamma' = \Gamma \cup \{\bar{\sigma}, \hat{\sigma} | \forall \sigma \in \Sigma\} \cup \{\#, \$\}$.

Die Simulation der einzelnen Kopfbewegungen wird im Folgenden beschrieben:

Kopfbewegung O einer zweidimensionalen Turingmaschine

Für die Simulation einer Kopfbewegung einer zweidimensioanlen Turingmaschine nach oben, von einem *Ausgangszeichen* zu einem *Zielzeichen*, müssen 19 Zustände eingeführt werden, welche mit $\{q'_{O_start}, q'_1, \ldots, q'_{17}, q'_{O_stop}\} \in Q'$ bezeichnet werden. Zu Beginn eines Simulationsschrittes O befindet sich die Turingmaschine T im Zustand $q_{O_start} \in Q$ und wechselt für die Simulation durch die eindimensionale Turingmaschine T' in den Zustand $q'_{O_start} \in Q'$, um von dort die Simulation zu starten. Nachdem die Kopfbewegung auf T' vollzogen wurde, befindet diese sich im Zustand $q'_{O_stop} \in Q'$ und wechselt in den Zustand $q_{O_stop} \in Q$ der zweidimensionalen Turingmaschine T. Die Simulation ist vollzogen.

Nachfolgende Auflistung beschreibt die Simualtion durch die Funktion δ', welche in Abbildung 2.21 zu sehen ist. Mit oberer und unterer Zeile werden stets die beiden Zeilen auf dem eindimensionalen Arbeitsband bezeichnet, auf welchen die Kopfbewegung statt findet. Die obere und untere Zeile kann dabei durch ein # oder $ getrennt sein, je nach dem, um welche Zeilen es sich handelt. Die Besonderheit in der Simulation liegt darin, die gleiche Stelle der unteren Zeile in der oberen Zeile zu finden. Dies geschieht durch eine Markierungsmethode.

q'_{O_start} Die Turingmachine markiert (¯) als erstes das Ausgangszeichen, auf dem sie sich befindet, also von welchem sie aus eine Kopfbewegung nach oben vollziehen soll, und wechselt nach q'_1.

q'_1 Die Turingmachine läuft nach rechts, bis sie auf ein # oder $ trifft. Sie sucht den rechten Rand der unteren Zeile.

q'_2 Sie markiert (ˆ) die Zeichen rechts des Ausgangszeichens in der unteren Zeile (dies passiert in q'_2), wechselt dann nach q'_3, abwechselnd mit den Zeichen in der oberen Zeile (dies geschieht in q'_4), bis sie auf das Ausgangszeichen in der unteren Zeile (markiert durch ein (¯)) wieder trifft. Dann wechselt die Maschine in den Zustand q'_7.

q'_3 Hier läuft die Turingmaschine in der unteren Zeile bis zum # oder $, welche die beiden Zeilen, in denen die Kopfbewegung vollzogen wird, voneinander trennen, und wechselt anschließend in Zustand q'_4.

q'_4 Hier werden die Zeichen der oberen Zeile durch (ˆ) markiert, um erkenntlich zu machen, welche Zeichen bereits besucht wurden. Nach jeder Markierung wird in den Zustand q'_5 gewechselt.

q'_5 In der oberen Zeile läuft die Maschine nach rechts, bis sie ein # oder $ passiert, welche die beiden Zeilen voneinander trennen, und wechselt anschließend in den Zustand q'_6.

q'_6 In Zustand q'_6 wird in der unteren Zeile bis zum letzten markierten (ˆ) Zeichen gelaufen, um dann das nächste Zeichen links daneben zu markieren. Dann wird in Zustand q'_2 gewechselt, wo das nächste Zeichen markiert wird. Dieser Durchgang wird solange vollzogen, bis alle noch nicht markierten Zeichen rechts vom Ausgangszeichen in der unteren Zeile markiert sind, um das gesuchte Zeichen in der oberen Zeile zu finden.

q'_7 Auf dem markierten Ausgangszeichen wieder angekommen läuft die Maschine hier nach rechts bis zum # oder $ und wechselt nach q'_8.

q'_8 Hier wird bis zu einem unmarkierten Zeichen in der oberen Zeile gelaufen, welches das gesuchte Zielzeichen ist, und ebenfalls duch (¯) markiert. Dann wird nach q'_9 gewechselt.

q'_9 Im Zustand q'_9 testet die Turingmaschine nun, ob sie sich in der obersten Zeile des MUR's befindet, das Trennsymbol zwischen den beiden Zeilen, in denen die Kopfbewegung vollzogen wird, also ein $ ist. Ist dies der Fall, wird $ durch ein # überschrieben und die Maschine wechselt nach q'_{11}, um dort eine neue Zeile im MUR anzubauen. Falls nicht, wechselt sie nach q'_{10}.

q'_{10} Hier läuft die Maschine ganz nach rechts bis zum Ausgangszeichen, wechselt nach q'_{15}, um im letzten Abschnitt der Überführungsfunktion alle Zeichen wieder zu entmarkieren.

q'_{11} Es werden abwechselnd die Zeichen in oberer und unterer Zeile links des markierten Ausgangszeichens markiert (ˆ), um die Größe einer Zeile des MUR's herauszufinden. In Zustand q'_{11} werden dazu die noch nicht markierten Zeichen markiert (ˆ). Anschließend wird in Zustand q'_{12} gewechselt, bis alle Zeichen links neben dem Ausgangszeichen markiert sind. Dies geschieht in q'_{12}. Dann wird nach Zustand q'_{13} gewechselt. Sind alle Zeichen in beiden Zeilen links des Ausgangszeichens markiert, die obere Zeile enthält nur Blanks (B), wird in den Zustand q'_{14} gewechselt, um dort das obere Ende zu suchen. Dies ist das erste unmarkierte B.

q'_{12} Hier wird bis zum ersten unmarkierten Zeichen in der oberen Zeile nach links gelaufen. Dieses wird markiert durch (ˆ) und in Zustand q'_{13} gewechselt.

q'_{13} In q'_{13} läuft die Maschine bis zum #, welches die beiden Zeilen voneinander trennt, und wechselt dann nach q'_{11}, um dort wiederum die Zeichen in der unteren Zeile links vom Ausgangszeichen zu markieren.

q'_{14} In Zustand q'_{14} angekommen sind alle Zeichen links vom Ausgangszeichen sowohl in der unteren als auch in der oberen Zeile markiert. So wird bis zum ersten unmarkierten Zeichen in der oberen Zeile nach links gelaufen. Dieses ist der neue Rand des MUR und wird durch ein $ überschrieben. Anschließend wird nach q'_{15} gewechselt.

q'_{15} Nun beginnt der Teil der Zustandsüberführungsfunktion, in dem die Zeichen wieder entmarkiert werden, um die Turingmaschine auf einen nächsten Schritt vorzubereiten. Hierfür wird in q'_{15} bis zum Trennzeichen # der beiden Zeilen nach rechts gelaufen, wobei die Markierung (ˆ) für jedes Zeichen entfernt wird, und in den Zustand q'_{16} gewechselt.

q'_{16} Hier wird bis zum nächsten # oder \$, also dem Rand der unteren Zeile, gelaufen, wobei alle Zeichen entmarkiert werden.

q'_{17} Die Turingmaschine entmarkiert in q'_{17} alle Zeichen, indem sie nach links läuft, bis sie auf dem durch ($^{-}$) markierten Zeichen in der oberen Zeile - dies ist das Zielzeichen - zum Stehen kommt, also in den Endzustand q'_{O_Stop} wechselt und dabei das gesuchte Zeichen ebenfalls entmarkiert.

δ'	σ	$\bar{\sigma}$	$\hat{\sigma}$	#	\$
q'_{O_start}	$(q'_1, \bar{\sigma}, R)$	–	–	–	–
q'_1	(q'_1, σ, R)	–	–	$(q'_2, \#, L)$	$(q'_2, \$, L$
q'_2	$(q'_3, \hat{\sigma}, L)$	$(q'_7, \bar{\sigma}, L)$	–	–	–
q'_3	(q'_3, σ, L)	$(q'_3, \bar{\sigma}, L)$	–	$(q'_4, \#, L)$	$(q'_4, \$, L)$
q'_4	$(q'_5, \hat{\sigma}, R)$	–	$(q'_4, \hat{\sigma}, L)$	–	–
q'_5	–	–	$(q'_5, \hat{\sigma}, R)$	$(q'_6, \#, R)$	$(q'_6, \$, R)$
q'_6	(q'_6, σ, R)	$(q'_6, \bar{\sigma}, R)$	$(q'_2, \hat{\sigma}, L)$	–	–
q'_7	(q'_7, σ, L)	$(q'_7, \bar{\sigma}, L)$	$(q'_7, \hat{\sigma}, L)$	$(q'_8, \#, L)$	$(q'_8, \$, L)$
q'_8	$(q'_9, \bar{\sigma}, R)$	–	$(q'_8, \hat{\sigma}, L)$	–	–
q'_9	–	–	$(q'_9, \hat{\sigma}, R)$	$(q'_{10}, \#, L)$	$(q'_{11}, \#, R)$
q'_{10}	–	$(q'_{15}, \bar{\sigma}, R)$	$(q'_{10}, \hat{\sigma}, L)$	–	–
q'_{11}	$(q'_{12}, \hat{\sigma}, L)$	$(q'_{13}, \bar{\sigma}, L)$	$(q'_{14}, \hat{\sigma}, R)$	–	–
q'_{12}	$(q'_{13}, \hat{\sigma}, R)$	$(q'_{12}, \bar{\sigma}, L)$	$(q'_{12}, \hat{\sigma}, L)$	$(q'_{12}, \#, L)$	–
q'_{13}	–	$(q'_{13}, \bar{\sigma}, R)$	$(q'_{13}, \hat{\sigma}, R)$	$(q'_{11}, \#, R)$	–
q'_{14}	$(q'_{15}, \$, R)$	$(q'_{14}, \bar{\sigma}, L)$	$(q'_{14}, \hat{\sigma}, L)$	$(q'_{14}, \#, L)$	–
q'_{15}	–	$(q'_{15}, \bar{\sigma}, R)$	(q'_{15}, σ, R)	$(q'_{16}, \#, R)$	–
q'_{16}	(q'_{16}, σ, R)	(q'_{16}, σ, R)	(q'_{16}, σ, R)	$(q'_{17}, \#, L)$	$(q'_{17}, \$, L)$
q'_{17}	(q'_{17}, σ, L)	(q'_{O_stop}, σ, N)	–	$(q'_{17}, \#, L)$	–

Abbildung 2.21: Zustandsüberführungsfunktion zur Simulation einer Kopfbewegung O

In Abbildung 2.22 ist die Simulation einer Kopfbewegung O dargestellt. Die Turingmaschine T befindet sich zu Beginn der Simulation im Zustand q_{O_start} und ihr Kopf auf dem markierten Zeichen $\boxed{\text{a}}$ in der obersten Zeile des MUR's und soll sich in die darüber liegende Zeile bewegen. Das linke Trennsymbol nach der Übertragung der Inhalte des zweidimensionalen Turingpapiers auf das eindimensionale Arbeitsband von T' ist also ein \$ und die Turingmaschine T' startet im Zustand q'_{O_start}. Für diese Kopfbewegung muss eine Zeile im MUR aufgefüllt werden.

	B	B	B	B	
	B	B	$\boxed{\text{a}}$	B	
q_{O_start}	B	b	c	B	
	B	B	B	B	

$$\Downarrow$$

$\cdots$	B	B	B	\$	B	a	#	b	c	#	B	B	B	$\cdots$	q'_{O_start}
						$\uparrow$									
$\cdots$	B	B	B	\$	B	$\bar{a}$	#	b	c	#	B	B	B	$\cdots$	q'_1
							$\uparrow$								
$\cdots$	B	B	B	\$	B	$\bar{a}$	#	b	c	#	B	B	B	$\cdots$	q'_2
						$\uparrow$									
$\cdots$	B	B	B	\$	B	$\bar{a}$	#	b	c	#	B	B	B	$\cdots$	q'_7
					$\uparrow$										
$\cdots$	B	B	B	\$	B	$\bar{a}$	#	b	c	#	B	B	B	$\cdots$	q'_7
				$\uparrow$											
$\cdots$	B	B	B	\$	B	$\bar{a}$	#	b	c	#	B	B	B	$\cdots$	q'_8
			$\uparrow$												
$\cdots$	B	B	$\bar{B}$	\$	B	$\bar{a}$	#	b	c	#	B	B	B	$\cdots$	q'_9
				$\uparrow$											
$\cdots$	B	B	$\bar{B}$	#	B	$\bar{a}$	#	b	c	#	B	B	B	$\cdots$	q'_{11}
					$\uparrow$										
$\cdots$	B	B	$\bar{B}$	#	$\hat{B}$	$\bar{a}$	#	b	c	#	B	B	B	$\cdots$	q'_{12}
				$\uparrow$											

$\cdots$	B	B	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{12}
			$\uparrow$												
$\cdots$	B	B	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{12}
		$\uparrow$													
$\cdots$	B	$\hat{B}$	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{13}
			$\uparrow$												
$\cdots$	B	$\hat{B}$	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{13}
				$\uparrow$											
$\cdots$	B	$\hat{B}$	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{11}
					$\uparrow$										
$\cdots$	B	$\hat{B}$	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{14}
				$\uparrow$											
$\cdots$	B	$\hat{B}$	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{14}
			$\uparrow$												
$\cdots$	B	$\hat{B}$	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{14}
		$\uparrow$													
$\cdots$	B	$\hat{B}$	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{14}
	$\uparrow$														
$\cdots$	$\$$	$\hat{B}$	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{15}
		$\uparrow$													
$\cdots$	$\$$	B	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{15}
			$\uparrow$												
$\cdots$	$\$$	B	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{15}
				$\uparrow$											
$\cdots$	$\$$	B	$\bar{B}$	$\#$	$\hat{B}$	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{16}
					$\uparrow$										
$\cdots$	$\$$	B	$\bar{B}$	$\#$	B	$\bar{a}$	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{16}
						$\uparrow$									
$\cdots$	$\$$	B	$\bar{B}$	$\#$	B	a	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{16}
							$\uparrow$								
$\cdots$	$\$$	B	$\bar{B}$	$\#$	B	a	$\#$	b	c	$\#$	B	B	B	$\cdots$	q'_{17}
						$\uparrow$									

$\cdots$	\$	B	$\bar{B}$	#	B	a	#	b	c	#	B	B	B	$\cdots$	q'_{17}
					$\uparrow$										
$\cdots$	\$	B	$\bar{B}$	#	B	a	#	b	c	#	B	B	B	$\cdots$	q'_{17}
				$\uparrow$											
$\cdots$	\$	B	$\bar{B}$	#	B	a	#	b	c	#	B	B	B	$\cdots$	q'_{17}
			$\uparrow$												
$\cdots$	\$	B	B	#	B	a	#	b	c	#	B	B	B	$\cdots$	q'_{O_stop}
			$\uparrow$												

	B	B	$\boxed{\text{B}}$	B	
	B	B	a	B	
$\Rightarrow q_{O_stop}$	B	b	c	B	
	B	B	B	B	

Abbildung 2.22: Beispiel zur Simulation einer Kopfbewegung O

Kopfbewegung R einer zweidimensionalen Turingmaschine

Im Folgenden wird die Simulation einer Kopfbewegung einer zweidimensioanlen Turingmaschine nach rechts, von einem *Ausgangszeichen* zu einem *Zielzeichen*, vorgestellt. Es werden 19 Zustände benötigt, welche mit $\{q'_{R_start}, \ldots, q'_{R_stop}\} \in Q'$ bezeichnet werden (vgl. hierzu Abbildung 2.23). Zu Beginn eines Simualtionsschrittes R befindet sich die Turingmaschine T im Zustand $q_{R_start} \in Q$ und wechselt für die Simulation durch die eindimensionale Turingmaschine T' in den Zustand $q'_{R_start} \in Q'$, um von dort die Simulation zu starten. Nachdem die Kopfbewegung auf T' vollzogen wurde, befindet diese sich im Zustand $q'_{R_stop} \in Q'$ und wechselt in den Zustand $q_{R_stop} \in Q$ der zweidimensionalen Turingmaschine T. Die Simulation ist vollzogen.

Nachfolgende Auflistung beschreibt die Simualtion durch die Funktion δ', welche in Abbildung 2.23 zu sehen ist. Die Besonderheit in der Simulation liegt hierbei darin, dass, falls der Kopf sich aus dem MUR herrausbewegen soll, jede Zeile dessen durch ein Blank B aufgefüllt werden muss. Dies geschieht durch eine Kopiermethode, indem für jedes Zeichen σ des Eingabealphabets Σ ein Zustand $q_\sigma \in Q'$ eingeführt wird.

q'_{R_start} Die Turingmaschine führt zunächst die gewünschte Kopfbewegung nach rechts durch und wechselt in den Zustand $q_{\hat{B}}$.

$q_{\hat{B}}$ Befindet sich die Maschine auf einem Zeichen des Eingabealphabets, ist die Bewegung bereits beendet und es wird in den Endzustand q'_{R_stop} geschaltet. Liest die Turingmaschine jedoch ein # oder ein \$, wird dieses Zeichen durch ein markiertes Blank $\hat{B}$ überschrieben, um die Zeile aufzufüllen und das gesuchte Zielzeichen zu markieren. Dann wird in den entsprechenden Zustand $q_{\hat{\#}}$ bzw. $q_{\$}$ gewechselt.

$q_{\hat{\#}}$ In diesem Zustand wird das gelesene Zeichen durch ein $\hat{\#}$ ersetzt. Dies stellt die neue Trennung der Zeilen voneinander dar. Anschließend wird in den entsprechenden Zustand gewechselt, je nach dem, welches Zeichen gelesen wurde ($a \to q_a, b \to q_b, c \to q_c, \cdots, B \to q_B, \# \to q_{\#}$).

$q_{a,b,c,\ldots,B,\#}$ Hier wird das gelesene Zeichen durch das Zustandszeichen, also das Zeichen, welches eine Position zuvor gelesen wurde, überschrieben, und wiederum in den Zustand mit dem Zeichen gewechselt, welches nun gelesen wurde. Anschließend wird noch ein Schritt weiter nach rechts gegangen. So werden die Zeichen des Arbeitsbandes alle um eine Position nach rechts verschoben, bis die Maschine ein \$ liest, welches durch ein B ersetzt wird. Dann geht sie noch einen Schritt weiter nach rechts und wechselt in den Zustand $q_{\$}$.

$q_{\$}$ Die Maschine liest ein B, welches durch das neue Ende der Zeile mit einem \$ überschrieben wird. Die Turingmaschine wechselt anschließend nach q_l.

q_l Hier wandert der Kopf der Turingamschine solange nach links, bis er ein $\hat{\#}$ trifft, die rechte Seite ist noch nicht fertig mit Blanks aufgefüllt. $\hat{\#}$ wird durch ein # überschrieben und nach q_r gewechselt, um weitere Blanks aufzufüllen. Trifft der Kopf jedoch auf das markierte Zeichen $\hat{B}$, wird nach q' geschaltet. Die rechte Seite des markierten Zeichens ist mit Blanks aufgefüllt. Nun wird die linke Seite bearbeitet.

q_r Der Kopf wandert solange nach rechts, bis er das nächste Trennsymbol der Zeilen # liest. Dieses wird durch ein Blank überschrieben, hiermit ist die nächste Zeile aufgefüllt, und in den Zustand $q_{\hat{\#}}$ gewechselt, um hier mit der Verschiebung der Zeichen des Arbeitsbandes um eine Position nach rechts fortzufahren.

q' Nun ist die linke Seite des markierten Blanks mit der Auffüllung der Zeilen dran. Hierfür bewegt sich der Kopf bis zur nächsten Trennung nach links. Ist diese ein \$, ist die Bewegung zu Ende. Die Maschine wechselt nach q'_{R_stop}. Ist dies nicht der Fall, schaltet die Turingmaschine in den Zustand q''_B.

q''_B Das gelesene Zeichen wird durch ein B ersetzt, hierdurch wird die Zeile aufgefüllt, und nach q'_a, q'_b,..., q'_B gewechselt, je nach dem welches Zeichen gelesen wurde.

$q'_{a,b,c,\ldots,B}$ Hier werden analog zur anderen Seite nun alle Zeichen des Arbeitsbandes um eine Position nach links verschoben, bis die Maschine ein \$ liest. Es wird in den Zustand $q'_{\$}$ gewechselt.

$q'_{\$}$ Hier liest die Maschine ein B, welches durch ein \$ überschrieben wird, um das Ende der Zeile zu schreiben. Anschließend wird in den Zustand q'_r gewechselt.

q'_r Der Kopf der Maschine wandert solange nach rechts, bis er ein markiertes Trennsymbol $\hat{\#}$ trifft, die linke Seite noch nicht fertig mit Blanks aufgefüllt ist. $\hat{\#}$ wird durch ein # überschrieben und nach q'_l gewechselt, um weitere Blanks aufzufüllen. Trifft der Kopf jedoch auf das markierte Zeichen $\hat{B}$, ist die linke Seite des markierten Zeichens auch mit Blanks aufgefüllt. Die Bewegung ist vollbracht und die Maschine wechselt in den Endzustand q'_{R_stop}.

q'_l Der Kopf wandert solange nach links, bis er das nächste Trennsymbol der Zeilen # liest. Dieses wird markiert (ˆ) und anschließend nach q''_B gewechselt, um dort mit der Auffüllung der Zeilen fortzufahren.

q'_{R_stop} Der Kopf der Turingmaschine läuft solange nach rechts, bis er das markierte Zielzeichen $\hat{B}$ trifft, entmarkiert dieses und beendet die Kopfbewegung.

δ	a	b	c	$\cdots$	B	$\hat{B}$	$\#$	$\hat{\#}$	$\$$
q'_{R_start}	$(q_{\hat{B}},a,R)$	$(q_{\hat{B}},b,R)$	$(q_{\hat{B}},c,R)$	$\cdots$	$(q_{\hat{B}},B,R)$	-	-	-	-
$q_{\hat{B}}$	(q'_{R_stop},a,N)	(q'_{R_stop},b,N)	(q'_{R_stop},c,N)	$\cdots$	(q'_{R_stop},B,N)	-	$(q_{\hat{\#}},\hat{B},R)$	-	$(q_{\$},\hat{B},R)$
$q_{\hat{\#}}$	$(q_a,\hat{\#},R)$	$(q_b,\hat{\#},R)$	$(q_c,\hat{\#},R)$	$\cdots$	$(q_B,\hat{\#},R)$	-	-	-	-
q_a	(q_a,a,R)	(q_b,a,R)	(q_c,a,R)	$\cdots$	(q_B,a,R)	-	$(q_{\#},a,R)$	-	$(q_{\$},a,R)$
q_b	(q_a,b,R)	(q_b,b,R)	(q_c,b,R)	$\cdots$	(q_B,b,R)	-	$(q_{\#},b,R)$	-	$(q_{\$},b,R)$
$\vdots$									
q_B	(q_a,B,R)	(q_b,B,R)	(q_c,B,R)	$\cdots$	(q_B,B,R)	-	$(q_{\#},B,R)$	-	$(q_{\$},B,R)$
$q_{\#}$	$(q_a,\#,R)$	$(q_b,\#,R)$	$(q_c,\#,R)$	$\cdots$	$(q_B,\#,R)$	-	-	-	-
$q_{\$}$	-	-	-	$\cdots$	$(q_l,\$,L)$	-	-	-	-
q_l	(q_l,a,L)	(q_l,b,L)	(q_l,c,L)	$\cdots$	(q_l,B,L)	$(q',\hat{B},L)$	$(q_l,\#,L)$	$(q_r,\#,R)$	-
q_r	(q_r,a,R)	(q_r,b,R)	(q_r,c,R)	$\cdots$	(q_r,B,R)	-	$(q_{\hat{\#}},B,R)$	-	$(q_{\$},B,R)$
q'	(q',a,L)	(q',b,L)	(q',c,L)	$\cdots$	(q',B,L)	-	$(q''_B,\hat{\#},L)$	-	$(q'_{R_stop},\$,R)$
q''_B	(q'_a,B,L)	(q'_b,B,L)	(q'_c,B,L)	$\cdots$	(q'_B,B,L)	-	-	-	-
q'_a	(q'_a,a,L)	(q'_b,a,L)	(q'_c,a,L)	$\cdots$	(q'_B,a,L)	-	$(q'_{\#},a,L)$	-	$(q'_{\$},a,L)$
q'_b	(q'_a,b,L)	(q'_b,b,L)	(q'_c,b,L)	$\cdots$	(q'_B,b,L)	-	$(q'_{\#},b,L)$	-	$(q'_{\$},b,L)$
$\vdots$									
q'_B	(q'_a,B,L)	(q'_b,B,L)	(q'_c,B,L)	$\cdots$	(q'_B,B,L)	-	$(q'_{\#},B,L)$	-	$(q'_{\$},B,L)$
$q'_{\$}$	-	-	-	$\cdots$	$(q'_r,\$,R)$	-	-	-	-
q'_r	(q'_r,a,R)	(q'_r,b,R)	(q'_r,c,R)	$\cdots$	(q'_r,B,R)	$(q'_{R_stop},\hat{B},N)$	$(q'_r,\#,R)$	$(q'_l,\#,L)$	-
q'_l	(q'_l,a,L)	(q'_l,b,L)	(q'_l,c,L)	$\cdots$	(q'_l,B,L)	-	$(q''_B,\hat{\#},L)$	-	$(q'_r,\$,R)$
q'_{R_stop}	(q'_{R_stop},a,R)	(q'_{R_stop},b,R)	(q'_{R_stop},c,R)	$\cdots$	(q'_{R_stop},B,R)	(q'_{R_stop},B,N)	-	-	-

Abbildung 2.23: Zustandsüberführungsfunktion zur Simulation einer Kopfbewegung R

In Abbildung 2.24 ist die Simulation einer Kopfbewegung R dargestellt. Die Turingmaschine T befindet sich zu Beginn der Simulation im Zustand q_{R_start} und ihr Kopf auf dem markierten Zeichen $\boxed{c}$ in der unteren Zeile des MUR's und soll sich einen Schritt nach rechts bewegen, so dass für diese Kopfbewegung jede Zeile des MUR's um ein Blank (B) aufgefüllt werden muss.

	B	B	B	B	
	B	B	a	B	
q_{R_start}	B	b	$\boxed{c}$	B	
	B	B	B	B	

$$\Downarrow$$

$\cdots$	B	B	B	\$	B	a	#	b	c	\$	B	B	B	$\cdots$	q'_{R_start}
									$\uparrow$						
$\cdots$	B	B	B	\$	B	a	#	b	c	\$	B	B	B	$\cdots$	$q_{\hat{B}}$
										$\uparrow$					
$\cdots$	B	B	B	\$	B	a	#	b	c	$\hat{B}$	B	B	B	$\cdots$	$q_{\$}$
											$\uparrow$				
$\cdots$	B	B	B	\$	B	a	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q_l
										$\uparrow$					
$\cdots$	B	B	B	\$	B	a	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'
									$\uparrow$						
$\cdots$	B	B	B	\$	B	a	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'
								$\uparrow$							
$\cdots$	B	B	B	\$	B	a	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'
							$\uparrow$								
$\cdots$	B	B	B	\$	B	a	$\hat{\#}$	b	c	$\hat{B}$	\$	B	B	$\cdots$	q''_B
						$\uparrow$									
$\cdots$	B	B	B	\$	B	B	$\hat{\#}$	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_a
					$\uparrow$										

$\cdots$	B	B	B	\$	a	B	$\hat{\#}$	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_B
				$\uparrow$											
$\cdots$	B	B	B	B	a	B	$\hat{\#}$	b	c	$\hat{B}$	\$	B	B	$\cdots$	$q'_\$$
			$\uparrow$												
$\cdots$	B	B	\$	B	a	B	$\hat{\#}$	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
				$\uparrow$											
$\cdots$	B	B	\$	B	a	B	$\hat{\#}$	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
					$\uparrow$										
$\cdots$	B	B	\$	B	a	B	$\hat{\#}$	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
						$\uparrow$									
$\cdots$	B	B	\$	B	a	B	$\hat{\#}$	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
							$\uparrow$								
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_l
						$\uparrow$									
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_l
					$\uparrow$										
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_l
				$\uparrow$											
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_l
			$\uparrow$												
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
				$\uparrow$											
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
					$\uparrow$										
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
						$\uparrow$									
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
							$\uparrow$								
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
								$\uparrow$							
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
									$\uparrow$						

$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_r
										$\uparrow$					
$\cdots$	B	B	\$	B	a	B	#	b	c	$\hat{B}$	\$	B	B	$\cdots$	q'_{R_stop}
										$\uparrow$					
$\cdots$	B	B	\$	B	a	B	#	b	c	B	\$	B	B	$\cdots$	q'_{R_stop}
										$\uparrow$					

$\Rightarrow q_{R_stop}$

B	B	B	B	
B	B	a	B	
B	b	c	$\boxed{\mathrm{B}}$	
B	B	B	B	

Abbildung 2.24: Beispiel zur Simulation einer Kopfbewegung R

Die Simulationen der Kopfbewegungen L und U können analog zu den zuvor beschriebenen Methoden durchgeführt werden. Aus diesem Grund wird an dieser Stelle auf eine sprachliche Formulierung und eine exakte Definition verzichtet. Es ist festzuhalten, dass die Beschreibung der Simulation einer zweidimensionalen Turingmaschine durch eine eindimensionale Turingmaschine sehr aufwendig ist. Die zweidimensionale Turingmaschine erweist sich als eine sehr zweckmäßige Maschine, da sie durch Hinzunahme der Kopfbewegungen unten und oben anschaulich Berechnungsprobleme löst als die eindimensionale Turingmaschine.

Der zweidimensionalen Turingmaschine sind auch Grenzen gesetzt. Diese werden im folgenden Kapitel vorgestellt.

2.2.3 Offene und unlösbare Probleme

1946 wurde der Öffentlichkeit der erste elektronische digitale Universalrechner, der „Electronic Numerical Integrator and Calculator“ (ENIAC) (vgl. Abbildung 2.25) vorgestellt, welcher von J. Presper Eckert (1919-1995) und John W. Mauchly (1907-1980) entwickelt wurde. Angesichts der rasanten Entwicklung der Computertechnologie und die damit verbunden Möglichkeiten, glaubten Wissenschaft und Gesellschaft, dass der Computer über unbegrenzte Möglichkeiten verfügt. Doch der Welt des Computers sind Grenzen gesetzt.

Im Folgenden geht es um die Frage, „Was Computer prinzipiell nicht lösen können“. Es werden Probleme vorgestellt, welche sich mit Hilfe eines zweidimensionalen Berechnungsmodells *computergerecht spezifizieren* lassen, aber nicht mit einem Computer gelöst werden können. Hierbei unterscheidet man zwischen Berechnungsproblemen, die *Offene Probleme* genannt werden, und *unlösbaren* Berechnungsproblemen. Für ein offenes Problem weiß man nicht, ob eine Lösung existiert oder nicht, während man für unlösbare Probleme beweisen konnte, dass keine Lösung existiert, also keine Turingmaschine angegeben werden kann, welches dieses löst.

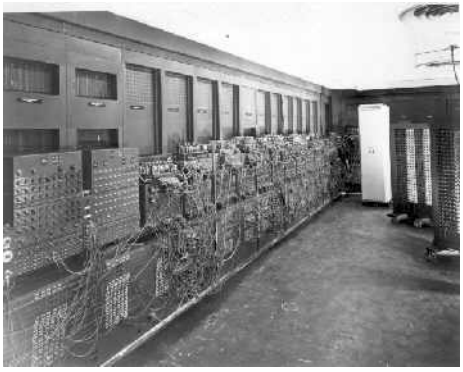

Abbildung 2.25: Electronic Numerical Integrator and Calculator - erster elektronischer digitaler Universalrechner

Offene Probleme

Offenes Problem bedeutet, dass nicht gesagt werden kann, ob das Problem lösbar oder unlösbar ist. Der der Ausgang des Problems ist *nicht-vorhersagbar*. Im Folgenden werden offene Probleme vorgestellt.

Beispiel 2.2.8 (Das Collatz-Problem)
Ein bis heute ungelöstes Problem aus der Arithmetik ist das *Collatz-Problem* des deutschen Mathematikers Lothar Collatz (1910-1990). Dieses Problem ist auch unter dem Namen $3n+1$-Problem bekannt. Es geht darum, ob ein rekursiver Algorithmus bei jeder beliebigen natürlichen Ausgangszahl eine Folge von natürlichen Zahlen berechnet, die immer mit einer 1 endet oder nicht. Viele Mathematiker haben sich seit der Veröffentlichung im Jahre 1937 damit beschäftigt. Für die Lösung oder einen Gegenbeweis sind Preisgelder festgelegt. Das Collatz-Problem kann durch folgenden Algorithmus beschrieben werden:

1. Wähle eine beliebige natürliche Zahl $a_0 \in \mathbb{N}\backslash\{0\}$.
2. Setze $n = 0$.
3. Solange $a_{n+1} \neq 1$ tue folgendes:
   (a) Falls a_n gerade, setze $a_{n+1} = \frac{1}{2}a_n$.
   (b) Falls a_n ungerade, setze $a_{n+1} = 3a_n + 1$.
4. Beende die Berechnung, sobald $a_{n+1} = 1$.

Es wird vermutet, dass die Folge für jede natürliche Zahl a_0 nach endlich vielen Schritten den Wert 1 erreicht. Diese Vermutung konnte bisher jedoch weder beweisen noch widerlegt werden. Computer haben alle natürlichen Zahlen bis zu einem enorm großen Wert durchprobiert. Immer endete die Folge mit einer 1, wodurch die Vermutung bestätigt, jedoch nicht bewiesen werden kann. Daher handelt es sich um ein offenes Problem. Abbildung 2.26 zeigt zwei Beispielfolgen zum Collatzproblem.

17	54	27	82	41	124	62	...
20	10	5	16	8	4	2	1

Abbildung 2.26: Beispielfolgen zum Collatz-Problem

Beispiel 2.2.9 (Langtons Ameise)

Eine bekannte Anwendung zweidimensionaler Turingmaschinen ist *Langtons Ameise*, welche ihren Namen dem Biologen Christopher Langton (*1949) verdankt. Sie ist ein einfacher Roboter, der sich auf einer unbegrenzten Ebene befindet, welche in quadratische Felder eingeteilt ist. Jedes Feld kann hierbei eine beliebige Farbe aus einer endlichen Palette haben; im einfachsten Falle handelt es sich um zwei Farben: die Farbe, die die Ebene zunächst besitzt (z.B. weiß), und die Farbe, mit der die Ameise Felder umfärbt (z.B. schwarz). Die Ameise sitzt auf einem Feld eines karierten Blatt Papiers und blickt nach Norden. Sie bewegt sich auf diesem Papier nach folgenden Regeln:

1. Sitzt sie auf einem weißen Feld, färbt sie es schwarz, dreht sich nach rechts und geht einen Schritt auf das nächste Feld.
2. Sitzt sie auf einem schwarzen Feld, färbt sie es weiß, dreht sich nach links und geht einen Schritt weiter auf das nächste Feld.

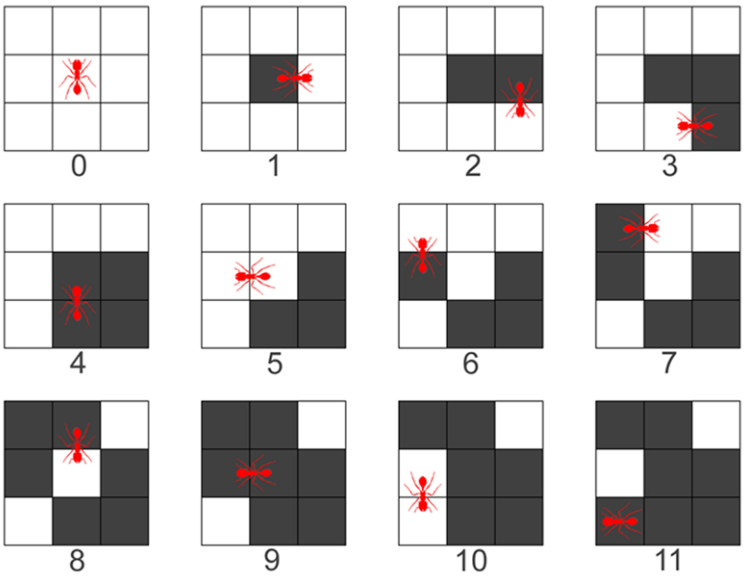

Abbildung 2.27: Die ersten Schritte der Ameise

In Abbildung 2.27 sind die ersten Schritte der Ameise dargestellt. Ihr Verhalten scheint zunächst chaotisch, indem sie ein komplexes chaotisches Muster bildet. Nach ca. 10.000 Schritten jedoch beginnt sie regelmäßige Muster (sogenannte *Autobahnen*) zu bauen (vgl. Abbildung 2.28)[3].

[3] http://de.wikipedia.org/wiki/Bild:AMEIHUND_Langtons_Ameise.png

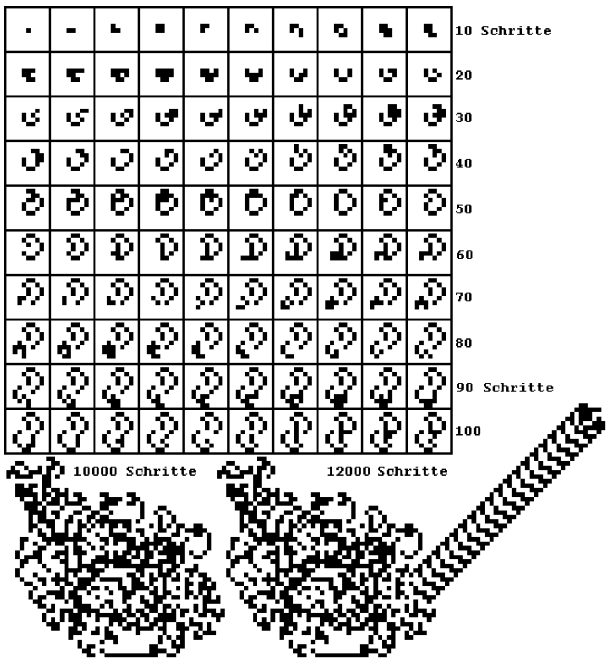

Abbildung 2.28: Das komplex chaotische Verhalten und die Autobahn der Ameise

Um das Verhalten einer Ameise zu verfolgen, kann man unter http://www.warrobs.com/langtonvslangton/startseite.php eine Simulation starten.

Im Folgenden wird Langtons Ameise formal definiert.

Definition 2.2.10 Langtons Ameise *ist eine zweidimensionale Turingmaschine* $T = (Q, \Sigma, \Gamma, \square, q_\uparrow, -, \delta)$ *mit*

- *der Zustandsmenge* $Q = \{q_\uparrow, q_\downarrow, q_\rightarrow, q_\leftarrow\}$,
- *dem Eingabealphabet* $\Sigma = \{\square\}$,
- *dem Papieralphabet* $\Gamma = \{\square, \blacksquare\}$,
- *dem Blanksymbol* $\square$,
- *dem Startzustand* $q_\uparrow$ *und*
- *der Zustandsüberführungsfunktion* $\delta : (Q \times \Gamma) \rightarrow (Q \times \Gamma \times \{R, L, O, U\})$ *(vgl. Abb. 2.29).*

δ	$\square$	$\blacksquare$
$q_\uparrow$	$(q_\rightarrow, \blacksquare, R)$	$(q_\leftarrow, \square, L)$
$q_\downarrow$	$(q_\leftarrow, \blacksquare, L)$	$(q_\rightarrow, \square, R)$
$q_\rightarrow$	$(q_\downarrow, \blacksquare, U)$	$(q_\uparrow, \square, O)$
$q_\leftarrow$	$(q_\uparrow, \blacksquare, O)$	$(q_\downarrow, \square, U)$

Abbildung 2.29: Übergangsregeln der Ameise

Die Zustände der Turingmaschine drücken jeweils aus, in welche Richtung die Ameise schaut. Je nach Farbe des Feldes, auf welchem sich die Ameise befindet, dreht sie sich nach rechts oder links, wechselt in den entsprechenden Zustand, der ihre neue Blickrichtung ausdrückt und vollzieht einen Schritt nach vorne.

Folgende Auflistung beschreibt die Umsetzung des Verhaltens der Ameise durch die angegebene Zustandsüberführungsfunktion:

$q_\uparrow$ Hier schaut die Ameise nach Norden. Ist das Feld, auf welchem sie steht, weiß, dreht sie sich nach rechts, wechselt in den Zustand $q_\rightarrow$, da sie nun nach Osten schaut. Der Kopf der Maschine bewegt sich ein Feld nach rechts. Ist das Feld, auf welchem sie steht, jedoch schwarz, dreht sie sich nach links, wechselt in den Zustand $q_\leftarrow$, da sie nun nach Westen schaut. Der Kopf der Maschine bewegt sich ein Feld nach links.

$q_\downarrow$ Hier schaut die Ameise nach Süden. Ist das Feld, auf welchem sie steht, weiß, dreht sie sich nach rechts, wechselt in den Zustand $q_\leftarrow$, da sie nun nach Westen schaut. Der Kopf der Maschine bewegt sich ein Feld nach links. Ist das Feld, auf welchem sie steht, jedoch schwarz, dreht sie sich nach links, wechselt in den Zustand $q_\rightarrow$, da sie nun nach Osten schaut. Der Kopf der Maschine bewegt sich ein Feld nach rechts.

$q_\rightarrow$ Hier schaut die Ameise nach Osten. Ist das Feld, auf welchem sie steht, weiß, dreht sie sich nach rechts, wechselt in den Zustand $q_\downarrow$, da sie nun nach Süden schaut. Der Kopf der Maschine bewegt sich ein Feld nach unten. Ist das Feld, auf welchem sie steht, jedoch schwarz, dreht sie sich nach links, wechselt in den Zustand $q_\uparrow$, da sie nun nach Norden schaut. Der Kopf der Maschine bewegt sich ein Feld nach oben.

$q_{\leftarrow}$ Hier schaut die Ameise nach Westen. Ist das Feld, auf welchem sie steht, weiß, dreht sie sich nach rechts, wechselt in den Zustand $q_{\uparrow}$, da sie nun nach Norden schaut. Der Kopf der Maschine bewegt sich ein Feld nach oben. Ist das Feld, auf welchem sie steht, jedoch schwarz, dreht sie sich nach links, wechselt in den Zustand $q_{\downarrow}$, da sie nun nach Sueden schaut. Der Kopf der Maschine bewegt sich ein Feld nach unten.

Langtons Ameise hat durch die geschickte Implementierung der Zustandsüberführungsfunktion einen Richtungssinn. Außerdem gibt es keinen Endzustand, da die Ameise bis in alle Ewigkeiten ihre Bewegung fortführt. Obwohl ihr Verhalten nichts mit Zufall zu tun hat, ja sogar bis in alle Zukunft vorherbestimmt ist, lässt sich aus der Kenntnis der Regeln allein nicht vorhersagen, was geschehen wird, also ob Langtons Ameise in einer Autobahn endet. Aus diesem Grund ist Langtons Ameise ein weiteres offenes Problem.

Unlösbare Probleme

Ein Problem heißt unlösbar, wenn es keinen Algorithmus gibt, der in endlich vielen Schritten die Lösung des Problems herbeiführen kann. Im Folgenden werden ein unlösbares Problem aus der Arithmetik, drei unlösbare Probleme aus der Geometrie, die drei klassischen Probleme der Antike, und zuletzt ein zweidimensionales unlösbares Berechnungsproblem (Game of Life) vorgestellt:

Beispiel 2.2.11 (Hilberts zehntes Problem)
Für *Hilberts zehntes Problem*, nach dem Mathematiker David Hilbert (1862-1943) benannt, gibt es allgemein keinen Algorithmus, der entscheidet, ob eine beliebige diophantische Gleichung lösbar ist.
Diophantische Gleichungen sind Gleichungen der Form $f(x_1, x_2, \ldots, x_n) = 0$, wobei f ein Polynom in mehreren Variablen ist. Als Lösung der Gleichung werden nur ganze Zahlen betrachtet. Viele große Mathematiker haben sich intensiv mit diesem Problem beschäftigt. Sie konnten immer wieder Spezialfälle lösen, doch eine allgemeine Lösung konnte im 19. Jahrhundert nicht gefunden werden. Erst 1970 gelang es Juri Matijassewitsch (*1947) zu beweisen, dass dieses Problem allgemein nicht lösbar ist.

Beispiel 2.2.12 (Die Dreiteilung des Winkels)

Unter der *Dreiteilung des Winkels* versteht man den Versuch, zu einem gegebenen Winkel φ mit Hilfe der euklidischen Werkzeuge Zirkel und Lineal den Winkel $\varphi/3$ zu konstruieren, also φ in drei gleichgroße Winkel zu teilen. Für bestimmte Winkelgrößen ist dies möglich, jedoch nicht im Allgemeinen. Die Dreiteilung für den Winkel $120°$ ist beispielsweise nicht möglich. (Zur genaueren Beweisidee hierzu siehe [Lor96]).

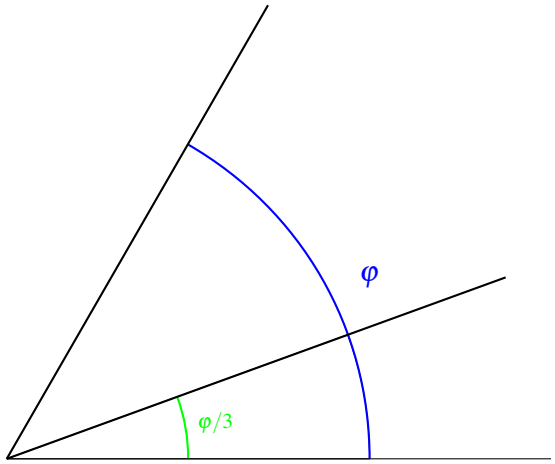

Abbildung 2.30: Die Dreiteilung des Winkels

Beispiel 2.2.13 (Die Verdopplung des Würfels)

Bei der *Verdopplung des Würfels* soll zu einem gegebenen Würfel ein Würfel doppelten Volumens konstruiert werden. Zur Umsetzung dieser Aufgabe betrachte man Abbildung 2.31. Der gegebene Würfel wird durch die Punkte P und Q festgelegt. Der Abstand der Punkte P und Q definiert die Kantenlänge des gegebenen Würfels und wird mit a bezeichnet. Der Würfel mit doppeltem Volumen ist durch die Punkte P und Q' festgelegt. Seine Kantenlänge ist a', wobei $(a')^3 = 2a^3$ also $a' = \sqrt[3]{2}a$ ist. Für die Konstruktion eines Würfels doppelten Volumens ist es demnach notwendig, den Punkt Q' zu konstruieren, der von P den Abstand $\sqrt[3]{2}a$ hat. Dies ist mit Hilfe algebraischer Mittel bewiesen, und zwar dadurch, dass der Grad der Körpererweiterung von $\mathbb{Q}$ adjungiert $\sqrt[3]{2}a$ in $\mathbb{Q}$ 3 beträgt. Für konstruierbare Körpererweiterungen muss der Grad jedoch eine 2-er Potenz sein (genaueres hierzu in [Lor96]).

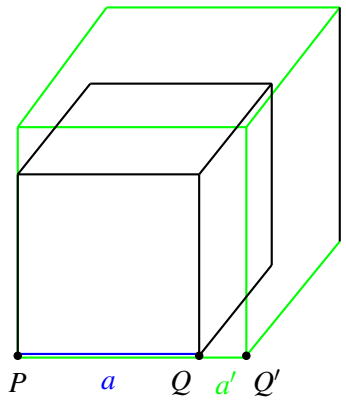

Abbildung 2.31: Die Verdopplung des Würfels

Beispiel 2.2.14 (Die Quadratur des Kreises)

Die *Quadratur des Kreises* beschreibt den Versuch, einen gegebenen Kreis in ein flächengleiche Quadrat zu überführen. Für dieses Problem ist es notwendig $\sqrt{\pi}$ zu konstruieren, wie in Abbildung 2.32 zu sehen ist. Aus $a^2 = r^2\pi$ folgt, dass $a = r\sqrt{\pi}$. Zum gegebenen Mittelpunkt O von Kreis und Quadrat muss der Punkt Q konstruiert werden, so dass der Abstand von O und Q $\frac{1}{2}r\sqrt{\pi}$ beträgt. Dieses Problem ist ebenfalls nicht konstruktiv lösbar, was mit Hilfe von algebraischen Mitteln gezeigt werden kann, welche darauf beruhen, dass die Zahl π transzendent ist (siehe hierzu ebenfalls [Lor96]).

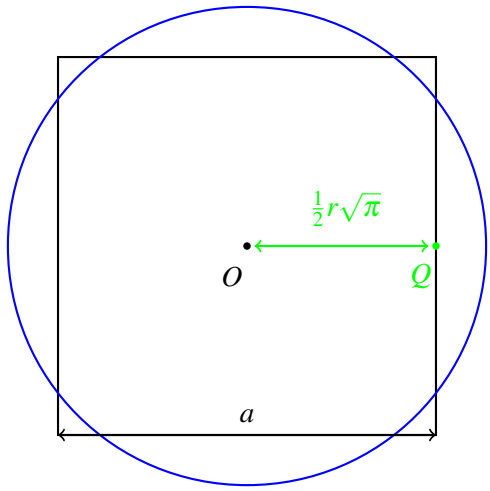

Abbildung 2.32: Die Quadratur des Kreises

Beispiel 2.2.15 (Das Lebensspiel)

Ein berühmtes Beispiel für einen zellulären Automaten ist das *Game of Life* (Lebensspiel) des englischen Mathematikers John H. Conway (*1937) (vgl. [Dew93] und [BCG85]). Gegeben ist ein unendliches Spielbrett bestehend aus Gitterquadraten,welche Zellen genannt werden. Jede Zelle besitzt zwei Zustände (lebend, tot). Die Umgebung einer Zelle besteht aus den acht angrenzenden Nachbarzellen, der von-Neumann-Nachbarschaft (vgl. Abschnitt 2.1.2 „Paralleles zweidimensionales Berechnungsmodell“). Die zu Beginn lebenden Zellen bilden die Anfangsgeneration. Das Spiel verläuft in Generationen, welche sich nach Verstreichen einer diskreten Zeit bilden. In jeder Generation können neue Zellen geboren werden, alte Zellen sterben oder einfach überleben. Eine Folgegeneration entsteht nach folgenden Regeln:

- Eine tote Zelle wird lebendig, wenn drei ihrer acht Nachbarn leben (Geburt).
- Eine Zelle stirbt, wenn mindestens vier Nachbarn leben (Übervölkerung).
- Eine Zelle stirbt, wenn sie keinen oder nur einen lebenden Nachbarn besitzt (Vereinsamung).

Eine lebende Zelle bleibt also genau dann am Leben, wenn sie zwei oder drei lebende Nachbarn besitzt. Sonst stirbt sie an Vereinsamung oder Überbevölkerung. Folgende Tabelle fasst die Übergangsregeln des „Game of Life“ zusammen:

Zustand	Nachbarzahl	Folgezustand
0	3	1
1	3	1
0	< 2 oder > 3	0
1	< 2 oder > 3	0
0	2	0
1	2	1

Abbildung 2.33: Übergangsregeln beim „Game of Life“

Das „Game of Life“ kann mathematisch formal wie folgt definiert werden:

Definition 2.2.16 *Das* Game of Life *ist eine zellulärer Automat* $ZA = \{R, N, Z, \delta\}$ *mit*

- *einem Lebensraum* $R = \{(i, j) | i, j \in \mathbb{N}\}$*, der zweidimensional und rechteckig ist,*
- *der von-Neumann-Nachbarschaft* $N_{i,j} = \{(k,l) \in R | |k-i| \leq 1 und |l-j| \leq 1\}$*,*
- *der Zustandsmenge* $Z = \{0, 1\}$
- *und der Zustandsübergangsfunktion*

$$\delta_{i,j}(z, \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix}) = \begin{cases} 1, & falls \sum_{i=1}^{n} z_i = 3; \\ 1, & falls \sum_{i=1}^{n} z_i = 2 \ und \ z = 1; \\ 0, & sonst. \end{cases}$$

Mit Hilfe der Übergangsregeln kann man zu jeder möglichen Anfangspopulation schrittweise die nachfolgende Generation ermitteln. Es genügen Bleistift, kariertes Papier und Radiergummi (vgl. Abbildung 2.34). Schneller ist es jedoch, die vielen einfachen Denkschritte einem Computer zu überlassen. Das Game of Life lässt sich auf dem Monitor eines Computers demonstrieren. Es gibt zahlreiche Programme hierfür im Internet zu finden. Ein Applet zum Ausprobieren dieses Spiels ist unter `http://www.bitstorm.org/gameoflife/` abrufbar.

In Abbildung 2.35 sind Anfangspopulationen dargestellt. Je nach Gestalt der Anfangspopulation ergeben sich ganz bestimmte Muster. Manche Figuren sind zu raschem Verschwinden verurteilt. Spezielles Interesse verdienen solche, die sich selbst reproduzieren und dabei durch das Gitter wandern, denn auf diese Weise ist so etwas wie die Ausbreitung einer Nachricht möglich [Fra96]. Unter stabilen Populationen versteht man Anfangspopulationen, welche nur Lebewesen mit 2 oder 3 Nachbarn enthalten. Sie ändern ihre Gestalt nicht mehr und heißen nach Conway Stillleben (still lifes) oder auch *stabile Populationen*. Mit *Oszillatoren* werden Anfangspopulationen bezeichnet, die ihre Gestalt innerhalb unterschiedlich langer Perioden wechseln. Sie kehren immer wieder zu ihrer Anfangsdarstellung zurück.

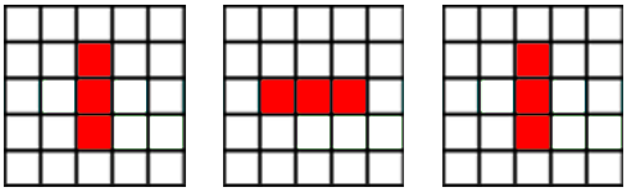

Abbildung 2.34: Die ersten Generationen einer Anfangspopulation

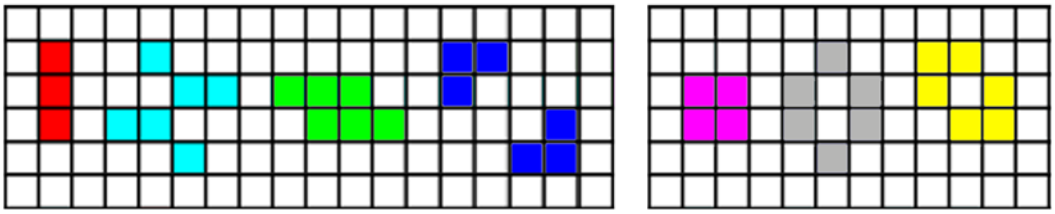

Abbildung 2.35: Links: Oszillatoren, Rechts: Stabile Populationen

Man ist natürlich an einem Algorithmus interessiert, der für eine beliebige gegebene Anfangspopulation des Lebensspiels entscheiden kann, ob diese unsterblich ist oder nicht. Der Algorithmus könnte wie in [Bau93] beschrieben vorgehen: Er versucht den Lebenslauf der Anfangspopulation Schritt für Schritt nachzuvollziehen. Stirbt die Population in endlich vielen Schritten aus, so kann der Algorithmus diese Tatsache feststellen. Stellt sich heraus, dass eine gegebene Anfangspopulation eine stabile Population oder ein Oszillator ist, so kann der Algorithmus feststellen, dass die Population nicht aussterben wird. Wenn dies jedoch nicht in endlich vielen Schritten geschieht, gelangt der Algorithmus zu keinem Ende und damit zu keiner Entscheidung. Die Entwicklung einer Anfangspopulation lässt sich in diesem Fall nicht vorhersagen. „Man muss abwarten und zusehen, was passiert“[Bau93]. Es läßt sich formal zeigen, dass die Eigenschaft, ob eine Population im Lebensspiel ausstirbt, nicht entscheidbar ist. Es kann also keine Turingmaschine angegeben werden, die für das Lebenspiel die Frage nach der Unsterblichkeit einer Population löst. Das Game of Life ist somit im Allgemeinen ein unlösbares Problem.

2.2.4 Idee der universellen Turingmaschine

Für die eindimensionale Turingmaschine wurde die Idee einer *universellen Turingmaschine* entwickelt (vgl. [HU90]), welche den zetralen Kern der Berechenbarkeitstheorie darstellt. Eine universelle Turingmaschine kann andere Turingmaschinen simulieren und stellt so eine programmierbare Variante der Turingmaschine dar. Sie erhält als Programm eine geeignete *Kodierung* einer beliebigen, aber speziellen Turingmaschine. Die Kodierung einer eindimensionalen Turingmaschine wird nach dem Mathetmatiker Kurt Gödel (1906-1978) *Gödelnummer* genannt und ist wie folgt definiert:

Definition 2.2.17 *Die* Gödelnummer $\langle T \rangle$ *einer beliebigen eindimensionalen Turingmaschine* $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$ *mit* $\Sigma \subseteq \Gamma = \{x_1, \ldots, x_n\}$, $x_1, \ldots, x_n \in \{0,1\}$, $Q = \{q_{start}, \ldots, q_{stop}\}$ *und Überführungsfunktion* δ *sieht wie folgt aus:*
Ein Übergang $\delta(q_i, x_j) = (q_k, x_l, D_m)$ *von* T*, wobei* $q_i, q_k \in Q$, $x_j, x_l \in \Gamma$ *und* $D_m \in \{L, R, N\}$*, wird kodiert durch*

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m.$$

Die Kodierung des j-ten Übergangs wird mit $code(j)$ *bezeichnet. Die Gödelnummer* $\langle T \rangle$ *bei s Übergängen der Zustandsüberführungsfunktion* δ *ist dann gegeben durch*

$$111\ code(1)\ 11\ code(2)\ 11 \cdots 11\ code(s)\ 111.$$

Die Zeichenfolge 11 *trennt die einzelnen Übergänge voneinander, die Zeichenfolge* 111 *markiert Ende und Anfang der Gödelnummer.*

Eine eindimensionale universelle Turingmaschine U erhält zur Simulation von T eine Eingabe der Form $\langle T \rangle w$, wobei w ein beliebiges Wort ist, auf welchem U das Verhalten von T simuliert. Abbildung 2.36 demonstriert die Kodierung einer eindimensionalen Turingmaschine zur Invertierung eines Bitstrings.

$\cdots$	111	0101010010	11	0100101010	11	0100010010001000	111	$\ldots$

Kodierung: $q_{invert} = 0$, $q_{stop} = 00$, $0 = 0$, $1 = 00$, $B = 000$, $R = 0$, $L = 00$, $N = 000$

Abbildung 2.36: Gödelnummer einer Turingmaschine zur Invertierung eines Bitstrings

Es wird nun auch eine zweidimensionale universelle Turingmaschine U konstruiert, die eine beliebige zweidimensionale Turingmaschine T simulieren kann.

Im Zweidimensionalen entspricht die Kodierung einer beliebigen zweidimensionalen Turingmaschine T einer Tabelle und wird im Folgenden auch mit $\langle T \rangle$ bezeichnet. Sie beinhaltet alle Informationen über T. Diese sind in der Zustandsüberführungsfunktion von T enthalten. Da die Zustandsüberführungsfunktion einer zweidimensionalen Turingmaschine bereits als Programm in tabellarischer Form vorliegt, kann diese als Kodierung bereits verwendet werden (vgl. hierzu die Programme zweidimensionaler Turingmaschinen in Kapitel 2.2 „Zweidimensionale Berechenbarkeit“ Abbildung 2.7 und 2.10). Es muss keine spezielle Kodierung gefunden werden. Dies ist der entscheidende Vorteil der zweidimensionalen Turingmaschine gegenüber der Eindimensionalen. Die Tabelle in nachfolgender Definition zeigt nocheinmal, wie das allgemeine Programm einer Turingmaschine T in tabellarischer Form aussehen könnte, damit es von einer zweidimensionalen universellen Turingmaschine U gelesen werden kann.

Definition 2.2.18 *Die* Kodierung $\langle T \rangle$ *einer beliebigen zweidimensionalen Turingmaschine* $T = (Q, \Sigma, \Gamma, B, q_{start}, q_{stop}, \delta)$ *ist gegeben durch die Tabelle*

δ		x_0			x_1			x_2		$\cdots$		x_n	
q_0	q_{00}	y_{00}	n_{00}	q_{01}	y_{01}	n_{01}	q_{02}	y_{02}	n_{02}	$\cdots$	q_{0n}	y_{0n}	n_{0n}
q_1	q_{10}	y_{10}	n_{10}	q_{11}	y_{11}	n_{11}	q_{12}	y_{12}	n_{12}	$\cdots$	q_{1n}	y_{1n}	n_{1n}
q_2	q_{20}	y_{20}	n_{20}	q_{21}	y_{21}	n_{21}	q_{22}	y_{22}	n_{22}	$\cdots$	q_{2n}	y_{2n}	n_{2n}
$\vdots$													
q_m	q_{m0}	y_{m0}	n_{m0}	q_{m1}	y_{m1}	n_{m1}	q_{m2}	y_{m2}	n_{m2}	$\cdots$	q_{mn}	y_{mn}	n_{mn}

mit

- $q_{start} = q_0$, $q_{stop} = q_m$, $q_{i,j} \in Q \; \forall i, j$,
- $Q = \{q_0, \ldots, q_m\}$,
- $\Sigma \subseteq \Gamma = \{x_0, \ldots, x_n, y_{00}, \ldots, y_{mn}\}$ *und*
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{R, L, N, O, U\}$, *wobei* $n_{00}, \ldots, n_{nm} \in \{L, R, N, O, U\}$ *ist.*

Als Eingabe erhält U eine erweiterte Tabelle der Form $\overset{\langle T\rangle}{t}$, wobei $\langle T\rangle$ die Kodierung der Turingmaschine T und t eine beliebige zweidimensionale Tabelle von Zeichen des Eingabealphabets ist. Die universelle Turingmaschine U simuliert nun das Verhalten der Turingmaschine T auf der Eingabe t, das heißt, sie wendet das Programm $\langle T\rangle$ auf t an. Bei inkorrekter Eingabe, die Eingabe beginnt nicht mit einer Tabelle, gibt U eine Fehlermeldung aus. Durch eine sehr aufwendige Implementation der universellen zweidimensionalen Turingmaschine U wird diese in der Kodierung $\langle T\rangle$ der beliebigen Turingmaschine T die entsprechenden Übergänge zu den gelesenen Zeichen suchen und diese eventuell überschreiben, indem sich der Kopf der universellen Turingmaschine U auf dem mit $\overset{\langle T\rangle}{t}$ beschrifteten Papier hin und her, und rauf und runter bewegt, so wie es das Programm von T fordert.
Ist $f : \Sigma \to \Sigma$ die Funktion, welche durch die beliebige Turingmaschine T berechnet wird, so liefert die Turingmaschine U als Ausgabe $f(t)$ nach der Simulation von T auf einer Eingabe t. Dies ist in Abbildung 2.37 grafisch dargestellt.

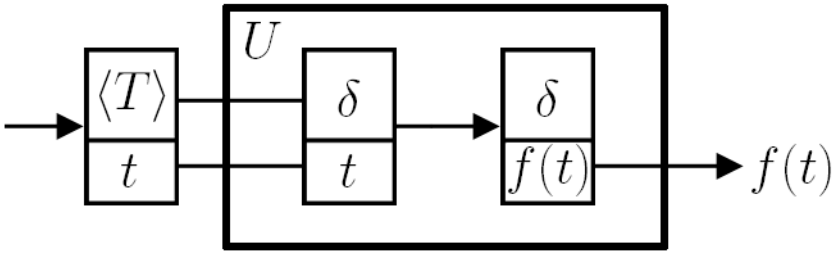

Abbildung 2.37: Die zweidimensionale universelle Turingmaschine U

Die *klassische Gödelisierung*, wie sie bei der eindimensionalen Turingmaschine angewendet wird, geht jedoch von einer Kodierung über dem Alphabet $\{0,1\}$ aus. Dies ist computerbauartbedingt. Aus diesem Grund wird auch überlegt, wie die klassische Kodierung, die „Gödeltabelle“ einer zweidimensionalen Turingmaschine über dem Alphabet $\{0,1\}$, aussehen könnte.

Klassische Gödelisierung einer Turingmaschine

Die Elemente der Zustandsüberführungsfunktion beinhalten alle Informationen der Turingmaschine T. Die Kodierung muss demnach als Kombination aus Zustand und Eingabezeichen mit den daraus folgenden Folgezuständen, Ausgabezeichen und Bandbewegungen angegeben werden. (vgl. [Mod86]). Für die Gödelisierung einer zweidimensionalen Turingmaschine müssen für die Elemente Q, Σ, Γ und δ der Turingmaschine geeignete Kodierungen über dem Alphabet $\{0,1\}$ gefunden werden. Dies könnte wie folgt aussehen:

Definition 2.2.19 *Sei $T = (Q,\Sigma,\Gamma,B,q_{start},q_{stop},\delta)$ eine zweidimensionale Turingmaschine mit $\Sigma \subseteq \Gamma = \{x_1,\ldots,x_n\}$, $x_1,\ldots,x_n \in \{0,1\}$, $Q = \{q_{start},\ldots,q_{stop}\}$ und Überführungsfunktion δ.*
Dann wird der Übergang $\delta(q_i,x_j) = (q_k,x_l,D_m)$, wobei $q_i,q_k \in Q$, $x_j,x_l \in \Gamma$ und $D_m \in \{L,R,O,U,N\}$, kodiert durch

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Die Kodierung des j-ten Übergangs wird mit $code(j)$ und die Kodierung des Startzustands mit $code(q_{start})$ bezeichnet. Die Gödeltabelle[4] *der zweidimensionalen Turingmaschine T ist dann bei s Übergängen gegeben durch nachstehende Tabelle, wobei die Zeichenfolge* 11 *die einzelnen Teile der Gödeltabelle trennt, nämlich die Kodierung der Übergänge von der Kodierung des Startzustands. Durch* 111 *wird der Anfang und das Ende der Gödeltabelle markiert wird.*

	1	1	1			
	$code(1)$					
	$code(2)$					
	$\vdots$					
	$code(s)$					
	1	1				
	$code(q_{start})$					
	1	1	1			

[4] Im Eindimensionalen wird der Begriff Gödelnummer verwendet, da die Kodierung einem Wort entspricht. Im Zweidimensionalen ist die Kodierung eine Tabelle.

Die Simulation eines Schrittes der Turingmaschine U bei einer Eingabe der Form $t^{\langle T \rangle}$ geschieht auf folgende Weise:

1. Initialisierung

   Zuerst überprüft U, ob die Eingabe mit einer korrekten Gödeltabelle beginnt. Dann schreibt U die Gödeltabelle $\langle T \rangle$ von T in tabellarischer Form wie zuvor beschrieben auf das zweidimensionale Arbeitsband. Anschließend wird die Tabelle t unter die Gödeltabelle geschrieben. Zwischen jeder Zeile der Tabelle t wird eine Leerzeile eingefügt, um die Kopfposition markieren zu können. Dies geschieht, indem eine 1 unter das Zeichen geschrieben wird, auf welchem sich der Kopf von T während der Simulation befindet. Es wird die Position markiert, auf welchem sich der Kopf zu Beginn der Simulation befindet. Dann startet die Simulation.

2. Simulation

   Für die Simulation wird an der Position der Gödeltabelle, an der sich der Startzustand befindet, immer die Kodierung des aktuellen Zustands gespeichert, in welchem sich die Maschine befindet. U sucht dann in jedem Simulationsschritt zu dem markierten Zeichen der Tabelle t, auf welchem sich der Kopf befindet bzw. auf welchem eine 1 steht, und zu dem aktuellen Zustand den entsprechenden Übergang. Wie in der Übergangsfunktion beschrieben, aktualisiert U den aktuellen Zustand in $\langle T \rangle$, überschreibt eventuell das gelesene Zeichen in t und führt die Kopfbewegung auf t aus, indem von der markierten Position aus die Kopfbewegung ausgeführt wird. Die alte Position wird entmarkiert und die neue Kopfposition wieder durch eine auf dem Zeichen stehende 1 markiert.

Beispiel 2.2.20 (Gödeltabelle einer Turingmaschine zur Invertierung eines Bitstrings)
Gegeben ist eine Turingmaschine $T = (Q, \Sigma, \Gamma, B, q_{invert}, q_{stop}, \delta)$ mit $Q = \{q_{invert}, q_{stop}\}$, $\Sigma = \{0,1\}$, $\Gamma = \{0,1,B\}$ und δ wie in Tabelle 2.38 angegeben.

δ		0			1			B	
q_{invert}	q_{invert}	1	R	q_{invert}	0	R	q_{stop}	B	N

Abbildung 2.38: Programm einer zweidimensionalen Turingmaschine zur Invertierung eines Bitstrings

Die einzelnen Elemente der Überführungsfuntkion δ werden folgendermaßen kodiert: $q_{invert} = 0$, $q_{stop} = 00$, $0 = 0$,$1 = 00$, $B = 000$, $L = 0$, $R = 00$, $O = 000$, $U = 0000$, $N = 00000$. Die Gödeltabelle dieser Turingmaschine T sieht dann wie in Abbildung 2.39 dargestellt aus.

1	1	1															
0	1	0	1	0	1	0	0	1	0	0							
0	1	0	0	1	0	1	0	1	0	0							
0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0
1	1																
0																	
1	1	1															

Abbildung 2.39: Gödeltabelle einer Turingmaschine zur Invertierung eines Bitstrings

Die Kodierung einer zweidimensionalen Turingmaschine geschieht auf einfachste Weise, da das Programm, welche jegliche Information einer Turingmaschine enthält, bereits wie in Abschnitt 2.1.1 „Die zweidimensionale Turingmaschine“ Abbildung 2.3 definiert, bereits in zweidimensioanler Form, und zwar ein Zeichen pro Kästchen, vorliegt. So ist die Kodierung der Maschine viel einsichtiger und sie lässt sofort erkennen, was die Turingmaschine berechnet.

Im eindimensionalen Fall sind die Übergänge in der Kodierung nicht sofort ersichtlich, da hier ein zweidimensionales Programm, welches leichter verständlich wäre, erst in eine eindimensionale Darstellung überführt werden muss. Auch bei einer Kodierung, die nicht über dem Alphabet $\{0,1\}$ vollzogen wird (vgl. Abb. **??**), sind die Übergänge nicht sofort ersichtlich, da der Anfang und das Ende eines Übergangs erst gesucht werden muss. Außerdem ist die Kodierung für Turingmachinen mit vielen Zustandsübergängen sehr lang und passt nicht mehr auf ein Blatt Papier. Das zugrundeliegende Eingabealphabet beinhaltet alle Zeichen, welche im Programm der zu simulierenden Turingmaschine enthalten sind. Ein Übergang wird durch die Aneinanderreihung seiner einzelnen Komponenten dargestellt (aktueller Zustand, gelesenes Zeichen, neuer Zustand, zu schreibende Zeichen, Kopfbewegung). Ein Zeichen # trennt die einzelnen Übergänge, zwei Zeichen ## markieren Anfang und Ende des Programms.

##	q_{invert}	0	q_{invert}	1	R	#	q_{invert}	1	q_{invert}	0	R	#	q_{invert} B q_{stop} B N	##

Abbildung 2.40: Kodierung einer eindimensionalen Turingmaschine zur Invertierung eines Bitstrings über einem Alphabet
kodierung1dim)

Es gibt zahlreiche Versuche, die eindimensionale universelle Turingmaschine zu implementieren. Da die Implementation der zweidimensionalen Turingmaschine aufwendig ist, wird an dieser Stelle auf diese verzichtet. Denn es geht nicht um die technische Ausführung einer universellen Turingmaschine, sondern um die Idee dahinter und vorallem den anschaulichen Vorteil der Kodierung einer beliebigen zweidimensionalen Turingmaschine und damit der Arbeitsweise einer zweidimensionalen universellen Turingmaschine.

In Kapitel 3 „Didaktische Überlegungen“ wird die bisher vorgestellte Thematik für die Schul-Informatik gefiltert und ein zweidimensionaler Ansatz der Berechenbarkeitstheorie vorgestellt.

Kapitel 3

Didaktische Überlegungen

3.1 Zweidimensionaler Berechenbarkeitsansatz für die Schul-Informatik

Die Turingmaschine bildet eine zentrale Idee in der Berechenbarkeitstheorie. Mit Hilfe dieser Maschine ist es anschaulich möglich, Berechnungsprobleme der Berechenbarkeitstheorie zu thematisieren und sie zu untersuchen. Aus diesem Grund ist es sinnvoll, ihre Arbeitsweise im Informatik-Unterricht zu behandeln. Mit Hilfe der Turingmaschine, deren Idee auch im Lehrplan NRW verankert ist (vgl. [MSW99]) können elementare Lösungsansätze und Denkweisen der Informatik verstanden werden. Außerdem bietet sie eine erste Form der Programmierung, an welche sich die Schüler auf einfache Weise mit Hilfe der Turingmaschine annähern können. Es stellt sich die Frage, wie ein zweidimensionaler Ansatz der Berechenbarkeitstheorie für die Schulinformatik aussehen könnnte.

Der Lehrplan des Landes Nordrhein-Westfalens ([MSW99]) setzt einen eindeutigen Schwerpunkt auf Themen der Berechenbarkeitstheorie. Er unterscheidet in den fachlichen Inhalten zwischen zwei Bereichen für die 11.-13. Klasse.

Der Bereich „Modellieren und Konstruieren“ fordert Folgendes:

- Anforderungen an ein Modell aufstellen
- ein reduziertes Modell für die Problemstellung definieren
- eine erste Lösungsskizze entwerfen
- Lösungskonzepte implementieren und testen

Im Bereich „Analysieren und Bewerten“ werden folgende Unterpunkte aufgeführt:

- den Begriff der Berechenbarkeit verstehen
- Grenzen von Verfahren und Methoden abschätzen

Die Sequenz „objektorientiert allgemein“ strebt schon in der Klassenstufe 11 an, „Grundlagen zu legen, die später zur Vorstellung eines allgemeinen Rechners (Turing-Maschine) und zum Begriff der Berechenbarkeit führen“. Im Bereich „Lernen im Kontext“ wird gefordert „Systematiken und Theorien zur Lösung von Anwendungssituationen“ zu schulen.

Im Folgenden wird ein zweidimensionaler Ansatz der Berechenbarkeitstheorie für die Schulinformatik in der Oberstufe nach den Richtlinien und Forderungen des Lehrplans NRW entwickelt.

Zweidimensionaler Lernzugang

Wie schon in Kapitel 1.3.1 „Bleistift und Papier“ thematisiert, ist der Zugang zur Berechnbarkeitstheorie über gewohnte Probleme und Arbeitsweisen, welche Schüler umgeben, von Vorteil. Das Denken und Arbeiten der Schüler geschieht meist über Blatt Papier und Stift. Da die Idee der zweidimensionalen Turingmaschine an dieses Denken und Arbeiten anknüpft, ist es sinnvoll, einen zweidimensionalen Ansatz zu wählen, um den Schülern den Einstieg in die Berechenbarkeitstheorie zu erleichtern. Im Folgenden wird darum ein zweidimensionaler Berechenbarkeitsansatz für die Schul-Informatik vorgestellt. Die Einführung in die Thematik wird gegeben und mögliche Anwendungsaufgaben für die Oberstufe werden vorgestellt. Auf eine konkrete didaktische Ausarbeitung wird hierbei verzichtet. Es sollen Denkanstöße gegeben werden, wie die Behandlung der Berechenbarkeitstheorie mit Hilfe eines zweidimensionalen Ansatzes aussehen könnte.

3.1.1 Einführung in die Thematik

Um die Schüler in die Thematik der Berechenbarkeitstheorie mit Hilfe der zweidimensionalen Turingmaschine einzuführen, wird der Zugang mittels Blatt Papier und Bleistift geschaffen. Die Einstiegs-Frage ist, mit welchen mathematischen Problemen ein Computer konfrontiert wird, und wie er zu deren Lösungen kommt. Mathematische Probleme können hierbei die Addition, Subtraktion, Multiplikation und Division zweier natürlicher Zahlen sein. Die Schüler sollen überlegen, wie sie sich selber mit diesen Problemen auseinandersetzen und sie selbstständig auf einem karierten Blatt Papier testen. Hieraus kann folgende Aufgabe resultieren:

Probleme eines Computers

Aufgabe:

Beschreibt, wie die schriftliche Addition zweier natürlicher Zahlen mit Hilfe von Stift und kariertem Papier vollzogen wird.

Die Schüler werden etwa Folgendes dazu vorbringen:

Verfahren zur Addition mit Papier und Stift

- Schreibe zwei Zahlen so in Spalten untereinander, dass die Einer, Zehner, Hunderter usw. immer untereinander stehen.
- Addiere die Einer und notiere ihre Summe in die Zeile darunter. Ist die Summe der Einer größer als die Zahl 10, so übertrage den Zehner in die Spalte davor und addiere die Zehner.
- Addiere die Zehner und notiere ihre Summe in die Zeile darunter. Ist die Summe der Zehner größer als die Zahl 10, so übertrage den Zehner in die Spalte davor und addiere die Hunderter.
- usw. mit Hunderter, Tausender,... bis alle Spalten addiert sind. Ist die Summe der letzten Spalte größer als die Zahl 10, so übertrage den Zehner in die Spalte davor und beende die Rechnung.

Anschließend können die drei wesentlichen Bausteine zur Durchführung einer Berechnung wie der Addition mit den Schülern wie in Abbildung 3.1 dargestellt an der Tafel festgehalten werden, um sich einem Berechnungsmodell anzunähern.

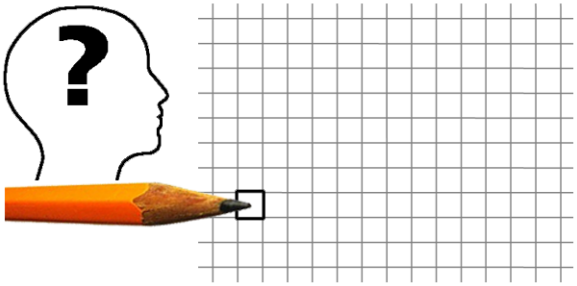

Abbildung 3.1: Modell der zweidimensionalen Turingmaschine

Zweidimensionales Berechnungsmodell

Durch die Auseinandersetzung mit ihrem eigenen Rechenvorgang angeregt soll mit den Schülern eine *Berechnungsmaschine* mit diesen Bausteinen (kariertes Papier, Stift und Gehirn) gebaut bzw. definiert werden. Dabei ist es zunächst wichtig, dass sich die Schüler überlegen, was eine solche Maschine braucht, was sie kann und vorallem, was sie nicht kann. In einer gemeinsamen Diskussion mit dem Lehrer werden die Schüler feststellen, dass die Maschine nicht selber denken kann und so einen Speicher benötigt, wo drin steht, wie sie in bestimmten Situationen reagieren soll. Anschließend muss überlegt werden, wie man diese verschiedenen Situationen, in welche die Maschine gelangen kann, beschreiben könnte. Dies geschieht durch die Einführung von Zuständen. Die Idee einer Zustandsmenge besprechen Lehrer und Schüler gemeinsam in einem offenen Lehrer-Schüler-Gespräch. Hiefür muss gemeinsam präzisiert werden, was *reagieren* für die Maschine heißt. Die Maschine muss, um reagieren zu können, einen Input bekommen. Hiefür braucht sie ein Auge, um den Input lesen zu können. Ihre Reaktion wird dadurch ausgedrückt, indem sie etwas tun soll, also einen Output liefert bzw. etwas aufschreibt. Durch einen Lese-/Schreibkopf, der den Stift bzw. das Auge der Maschine repräsentiert, kann dies zusammengefasst werden. Die Maschine liest und schreibt in Abhängigkeit von der Situation, einem Zustand, in der sie sich befindet. Um weitere Reaktionen ausdrücken zu können, kann die Maschine nicht immer in der gleichen Situation verharren. Wie kommt sie aber in eine andere Situation? Hierfür wird ein Situationswechsel benötigt, also ein Zustandswechsel. Dies ist der Knackpunkt bei der Findung und Einführung einer Maschine, der mit den Schülern diskutiert werden muss.

Im weiteren Verlauf kann folgende Aufgabe mit den Schülern präzisiert werden:

Aufgabe:

Ein kariertes Papier ist mit zwei Dezimalzahlen beschriftet. Die Einer, Zehner, Hunderter usw. der beiden Zahlen stehen untereinander. Schreibe Befehle für den Stift auf, welcher die Addition der zwei Zahlen ausführen soll.

Pseudocode der schriftlichen Addition

Es ist wichtig, mit den Schülern die Einführung einer *Zustandsmenge* zu thematisieren, damit die Durchführung einer Berechnung möglich wird. In diesem Fall werden die Zustände $q_0, \ldots, q_9$ eingeführt, welche ausdrücken, welche Zahlen bisher addiert wurden. Als Start einer Berechnung dient außerdem ein *Startzustand* und als Ende ein *Endzustand*. An dieser Stelle muss auch geklärt werden, wo sich der Stift der Maschine zu Beginn befindet. Außerdem sollte die kürzere der beiden Zahlen mit führenden Nullen verlängert werden. Dann können die Schüler anschließend etwa Folgendes festhalten:

- Setze $i = 0$.
- Solange das Kästchen nicht leer ist, tue Folgendes:
  - Der Stift befindet sich in q_i. Lese die Zahl j, wechsle nach q_{i+j}, bewege den Stift ein Kästchen nach oben.
  - Der Stift befindet sich in q_{i+j}. Lese die Zahl k, wechsle nach q_{i+j+k}, bewege den Stift zwei Kästchen runter.
    - Falls $i + j + k < 10$ schreibe $i + j + k$ in das Kästchen, bewege den Stift ein Kästchen nach links, wechsle nach q_0, das heißt setze $i = 0$.
    - Andernfalls schreibe $i + j + k - 10$ in das Kästchen, bewege den Stift ein Kästchen nach links, wechsle nach q_1, das heißt setze $i = 1$.
  - Bewege den Stift ein Kästchen nach oben.
- Der Stift befindet sich auf einem leeren Kästchen.
  - Falls er sich in q_1 befindet, schreibe eine 1 in das Kästchen.
  - Andernfalls beende die Berechnung.

Anschließend soll die Arbeitsweise der Maschine thematisiert und aufgeschlüsselt werden. Die Möglichkeiten des Stifts und seine Befehle werden besprochen. Außerdem muss man sich auf eine Ein- und Ausgabe des Maschinenpapiers einigen.

Arbeitsweise des Berechnungsmodells

An der Tafel wird Folgendes mit den Schülern festgehalten:

Der Stift einer Maschine kann in Abhängigkeit der Situation

- den Inhalt eines Kästchens lesen,
- etwas in ein Kästchen reinschreiben,
- den Inhlat eines Kästchens überschreiben (also radieren und schreiben),
- sich in ein anderes Kästchen bewegen und
- in eine neue Situation wechseln.

Woher bekommt die Maschine aber nun ihre Befehle? Ein Programm, welches das menschliche Gehirn ersetzt, soll alle Instruktionen für den Stift der Maschine enthalten. Dies geschieht durch die Definition einer Funktion, welche alle Regeln einer Maschine enthält. Die Schüler müssen sich hierzu überlegen, was dieser Funktion übergeben wird und was diese Funktion ausgeben soll. An der Tafel kann folgendes hierzu festgehalten werden:

Eine Befehlsfunktion einer Maschine beinhaltet

- die Situation, in der sich der Stift der Maschine befindet,
- welches Symbol der Stift liest,
- durch welches Zeichen das gelesene Symbol überschrieben wird,
- in welche neue Situation der Stift wechseln soll und
- zu welchem Kästchen der Stift sich bewegen soll.

Nun kann mit den Schülern besprochen werden, dass zur Eingabe einer Maschine alle beschrifteten Kästchen des Papiers vor der Durchführung einer Berechnung und zur Ausgabe alle beschrifteten Kästchen des Papiers nach der Durchführung einer Berechnung gehören. Im nächsten Schritt soll eine formale Definition der Berchnungsmaschine mit den Schülern gefunden werden. Hierzu kann an der Tafel nachfolgende Definition mit den Schülern umgangssprachlich festgehalten werden.

Schüler-Definiton einer Berechnungsmaschine

Definition 3.1.1 *Eine* Berechnungsmaschine *besteht aus einem karierten Papier, einem Stift und einem Programm, wobei*

- *ein Papieralphabet Σ beschreibt, mit welchen Symbolen die Kästchen des Papiers beschriftet werden können,*
- *ein Leerzeichen $\square$ eingeführt wird, mit dem die Kästchen des Papiers beschriftet werden, welche leer sind,*
- *Zustände einer Zustandsmenge Q beschreiben, in welcher Situation sich gerade die Maschine befindet,*
- *ein Anfangszustand $start \in Q$ wiedergibt, in welchem Zustand die Maschine startet*
- *ein Endzustand $stop \in Q$ wiedergibt, wann die Maschine ihre Berechnung abgeschlossen hat,*
- *und eine Befehlsfunktion $\delta : (Q \times \Sigma) \rightarrow (\Sigma \times \{R, L, N, O, U\}^{\star} \times Q)$ ausdrückt, was der Stift der Maschine zu tun hat, nämlich in welchem Zustand sich die Maschine befindet und welches Symbol sie dort liest, ob und mit welchem Symbol sie das gelesene Zeichen überschreibt und welche Bewegung der Stift ausführen soll. Hiefür stehen dem Stift die mehrfache Anwednung der Bewegungen rechts (R), links (L), nichts (N), oben (O) und unten (U) zur Verfügung. Man kann sich auch auf andere Möglichkeiten einigen.*

Formale Definition der 2D-Turingmaschine

Anschließend wird mit den Schülern die formale Definition der Turingmaschine aus Kapitel 2.1.1 „Die zweidimensionale Turingmaschine“ sowie die formale Notation eines Programms und einer Berechnung einer solchen Maschine besprochen werden. An einfachen Beispielen wie der Invertierung eines Bitstrings (siehe hierzu Abbildung 2.38) können sich die Schüler in der formalen Anwendung des Modells mit Hilfe von Papier und Stift trainieren.

Übergang zu TuringKara

Ist der theoretische Hintergedanke und die Arbeitsweise der Turingmaschine mit den Schülern besprochen, kann auf spielerische Weise nun die Lernsoftware TuringKara, welche im folgenden Kapitel 3.2 „Turing-Kara - Zweidimensionale Turingmaschine“ beschrieben wird, zum Einsatz kommen.

3.2 TuringKara - Zweidimensionale Turingmaschine

TuringKara ist ein Modul der Lernsoftware Kara, mit dem sich auf anschauliche Weise eine große Spannbreite von Aufgaben rund um Turingmaschinen wie Grundrechenarten oder Mustererkennung illustrieren lassen, indem ein kleiner Marienkäfer in einer einfachen grafischen Welt programmiert wird. Die Lernsoftware dient der Animation der Ausführung theoretischer Programme und erleichtert die Durchführung von Rechnungen, die per Hand eventuell zu umständlich sind. Dabei handelt es sich um ein Javaprogramm, das auf allen Betriebssystemen mit einer Java-Virtual-Maschine lauffähig ist. TuringKara ist eine zweidimensionale Turingmaschine und arbeitet, wie dies in den vorherigen Kapiteln beschrieben wurde, auf einem zweidimensionalen Blatt. Dies vereinfacht die Programme im Vergleich zu eindimensionalen Turing-Programmen. „Aus Sicht der theoretischen Berechenbarkeit bringt ein zweidimensionales Speichermedium zwar keine Vorteile. Es hat jedoch Vorteile in didaktischer Hinsicht“ [RNH04], da Lösungen von Aufgaben vereinfacht werden und die zu verarbeitenden Daten besser dargestellt werden können. Ebenfalls wird die Anzahl der Bewegungen des Lese-/Schreibkopfs stark reduziert. In erster Linie richtet sich TuringKara an Schüler und Studierende, die das Berechnungsmodell der Turing-Maschinen auf spielerische Weise näher kennenlernen und ausprobieren möchten.

3.2.1 Die TuringKara-Umgebung

Die TuringKara-Umgebung ist durch die Welt des Marienkäfers Kara visualisiert, wobei dieser hier als Lese-/Schreibkopf, einem roten Quadrat, statt als Marienkäfer dargestellt ist. Der Lese-/Schreibkopf befindet sich auf einem Rechteck, das in quadratische Felder eingeteilt ist, Auf diesem Rechteck kann der Kopf sich in alle vier Richtungen bewegen, dagestellt durch die Symbole $\Leftarrow, \Rightarrow, \Uparrow$ und $\Downarrow$. Er kann aber auch auf einem Feld stehen bleiben. Hiefür gibt es kein Symbol, es wird einfach kein Bewegungsbefehl definiert. Abbildung 3.2 links zeigt die TuringKara-Umgebung, welche als Welteditor bezeichnet wird und das Band der zweidimensionalen Turingmaschine darstellt. Hier kann die Größe des Rechtecks festgelegt sowie die Bandbelegung mit der Maus durch Klicken belegt werden. Die Felder der Welt können folgende Symbole aufnehmen: 0, 1, #, $\square$ sowie die vier Pfeilsymbole $\leftarrow$, $\rightarrow$, $\uparrow$, $\downarrow$. Die aktuelle Leseposition wird durch das rote Quadrat markiert. Ausgewählte Bandeinstellungen können

auch gespeichert und geladen werden. Ist ein Programm, welches im Programmeditor erstellt wird, geladen, so kann es hier gestartet werden. Neben einem Einzelschrittmodus kann die Ablaufgeschwindigkeit eingestellt werden, so dass es möglich ist, die Durchführung einer Berechnugn schritt für Schritt visuell nachvollziehen zu können. Über den Menüeintrag „Programmieren“ gelangt man in den Programmeditor (Abbildung 3.2 rechts). Hier werden die einzelnen Zustände definiert. Ein Zustand beinhaltet dabei, welches Zeichen gelesen wird, was Kara macht und den nächsten Zustand, also den Übergang zu einem neuen Zustand (vgl. Abbildung 3.2).

Ein wesentlicher Unterschied zu der bisher bekannten zweidimensionalen Turingmaschine ist, dass hier für den Übergang eine beliebige Anzahl von Schreib- und Bewegungsbefehlen definiert werden kann. Hierdurch können natürlich im Vergleich mit dem Standardmodell die Anzahl der Zustände reduziert und überschaubarere Maschinen gebaut werden. Zusätzlich wird automatisch der Zustandsautomat grafisch oberhalb der definierten Zustände abgebildet. Ein Zustand wird durch einen Kreis und Übergänge zwischen Zuständen durch Pfeile markiert. Ebenfalls werden der Startzustand und der Endzustand erkenntlich gemacht.

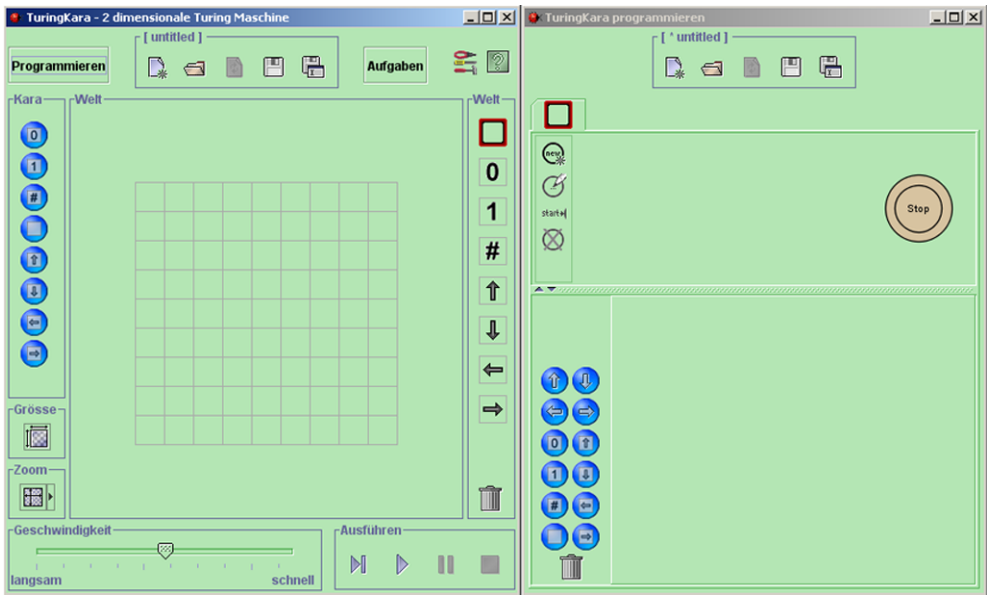

Abbildung 3.2: Die TuringKara-Umgebung (Welteditor und Programmeditor)

Im Folgenden wird eine formale Defintion einer Kara-Turingmaschine gegeben, um die Unterschiede zum Standardmodell zu verdeutlichen. Ob es sinnvoll ist, diese Definition im Unterricht mit den Schülern zu thematisiseren, bzw. aus Zeitgründen überhaupt möglich sein wird, bleibt jedem selbst überlassen. Um jedoch besonders in der Definition der Zustandsüberführungsfunktion auch formale Unterschiede zwischen Standardmodell und Kara-Modell zu erkennen, wird hier eine formale Definition gegeben:

Formale Definiton der Kara-Turingmaschine

Definition 3.2.1 *Eine* Kara-Turingmaschine *ist gegeben durch ein 7-Tupel* $T_{KARA} = (Q, \Sigma, \Gamma, \Box, start, Stop, \delta)$ *mit*

- *einer endlichen Zustandsmenge* Q*,*
- *dem Eingabealphabet* $\Sigma \subseteq \{0, 1, \#, \Box, \leftarrow, \rightarrow, \uparrow, \downarrow\}$*,*
- *dem Bandalphabet* $\Gamma \subseteq \{0, 1, \#, \Box, \leftarrow, \rightarrow, \uparrow, \downarrow\}$*,*
- *dem Blanksymbol* $\Box$
- *dem Anfangszustand* $start \in Q$*,*
- *dem Endzustand* $Stop \in Q$
- *und der Übergangsfunktion* $\delta : (Q \backslash \{Stop\} \times \Sigma) \rightarrow (Q \times (\Sigma \cup \{\Leftarrow, \Rightarrow, \Uparrow, \Downarrow\})^{\star})$.

Die Zustandsüberführungsfunktion δ ist wie folgt zu interpretieren:
Durch $\delta(q, a) = (q', b)$ wird beschrieben, dass in Zustand $q \in Q$ ein Zeichen $a \in \Gamma$ gelesen und anschließend eine beliebige Folge $b \in \Sigma \cup \{\Leftarrow, \Rightarrow, \Uparrow, \Downarrow\})^{\star}$ von neuen Zeichen schreiben und Kopfbewegungen vollziehen durch geführt und zuletzt in den Zustand q' gewechselt wird.

3.2.2 Beispielaufgaben mit TuringKara

Kennenlernen der Software

Die Schüler beginnen auf spielerische Weise und vielleicht unter Anleitung des Lehrers mit der TuringKara-Umgebung vertraut zu werden, indem sie das Programm durch „Rumklicken“ kennenlernen. Da sie sich bereits mit Hilfe von Papier und Stift mit der Arbeitsweise einer Turingmaschine beschäftigt haben, können die Schüler gezielt durch dieses Rumklicken die Arbeitsweise einer Turingmaschine in der TuringKAra-Umgebung kennenlernen.

Kara-Turingmaschine vs. Standardmodell

Hier werden ihnen gleich zentrale Merkmale und Funktionsweisen dieser Lernsoftware auffallen, so dass die Unterschiede der Kara-Turingmaschine zum definierten Standardmodell mit den Schülern thematisiert werden können. Es kann versucht werden, mit den Schülern eine formale Definition der Kara-Turingmaschine zu finden, wie diese in Abschnitt 3.2.1 „Die TuringKara-Umgebung“ zuvor gegeben ist. Es können aber auch einfach stichwortartig Unterschiede an der Tafel festgehalten werden.

Erste Beispiele in TuringKara

Ist dieser Schritt getan, kann mit einfachen Beispielaufgaben begonnen werden. Hierbei ist es sinnvoll, mögliche Lösungswege im Pseudocode an der Tafel mit den Schülern gemeinsam oder auch in Partnerarbeit vorerst besprechen zu lassen. So entwickeln sie ein Gespür für die Herangehensweise an die einzelnen Arbeitsschritte der Turingmaschine. In der TuringKara-Umgebung sind bereits einige Aufgaben mit Schwierigkeitsgrad „einfach“ bis „sehr schwierig“ integriert. Ein paar davon und andere werden im Folgenden vorgestellt.

Aufgabe, Ausgangslage, Schlussbedingung, Definition

Für die Aufgabenbearbeitung wird den Schülern zuerst immer der Arbeitsauftrag mitgeteilt. Im nächsten Schritt müssen sie sich überlegen, wie die Ausgangslage der Aufgabenstellung aussieht, das heißt, wie die Bandbeschriftung der Kara-Turingmaschine aussieht und wo sich der Kopf der Maschine zu Beginn der Berechnung befindet. Die Schüler müssen sich auch eine Schlussbedingung ihrer Maschine überlegen, das heißt, was das Ende der Berechnung der Maschine charakterisiert. Hierzu kann die formale Definition sowie die Regeln der Zustandsüberführungsfunktion zu jeder Kara-Turingmaschine von den Schülern angegeben werden. Für die Darstellung der Funktion wird die Darstellung aus Abbildung 2.15 gewählt.

Beispiel 3.2.2 (Invertieren eines Bitstrings)

Invertieren eines Bitstrings

Aufgabe:

Erstelle eine Turing-Maschine, die eine beliebige Zeichenkette aus Nullen und Einsen invertiert.

Ausgangslage:

Auf dem Turingband steht eine Zeichenkette, welche aus Nullen und Einsen besteht. Der Kopf der Maschine befindet sich auf dem ersten Zeichen links der Zeichenkette.

Schlussbedingung:

Alle Bits der Zeichenkette stehen invertiert auf dem Turingband.

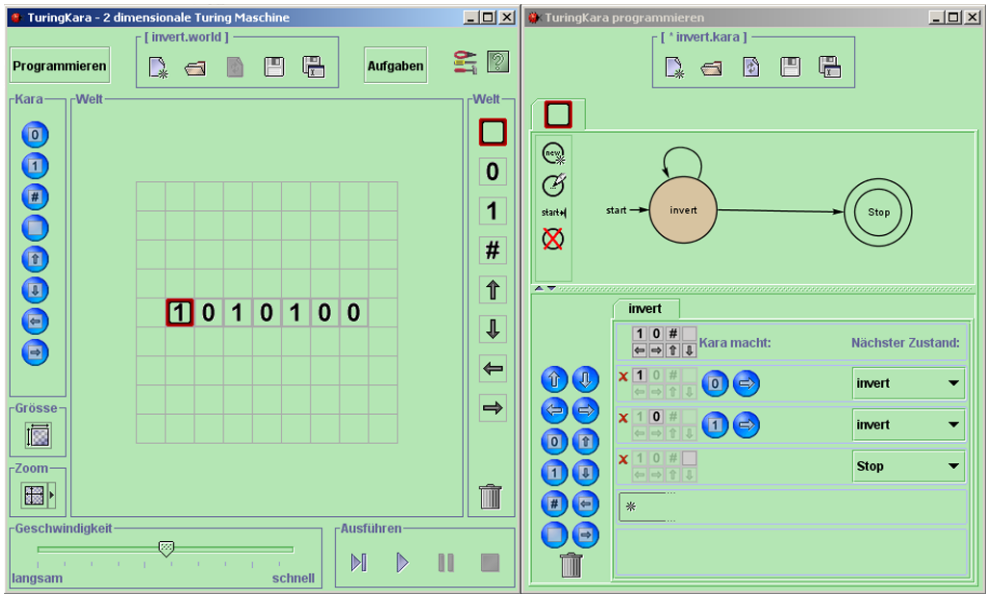

Abbildung 3.3: Invertieren eines Bitstrings mit Hilfe einer Kara-Turingmaschine

Abbildung 3.3 zeigt rechts die Zustandsüberführungsfunktion δ einer Turingmaschine T_{KARA} zur Invertierung eines Bitstrings mit Hilfe des Programmeditors der TuringKara-Umgebung.

Definiert wird eine Kara-Turingmaschine $T_{KARA} = (Q, \Sigma, \Gamma, \Box, invert, Stop, \delta)$ zur Invertierung eines Bitstrings mit $\Sigma = \{0,1\}$, $\Gamma = \{0,1,\Box\}$ und δ wie in Abbildung 3.4. Der Kopf der Turingmaschine läuft von links nach rechts über den String und invertiert in jedem Schritt das aktuelle Bit. Sobald der Kopf über einem Zeichen $\Box$ zu stehen kommt, ist die Invertierung zu Ende und die Turingmaschine wechselt in den Endzustand Stop.

δ	0	1	$\square$
invert	$(invert, (1 \Rightarrow))$	$(invert, (0 \Rightarrow))$	$(Stop, (\square))$

Abbildung 3.4: Zustandsüberführungsfunktion einer Kara-Turingmaschine zur Invertierung eines Bitstrings

Beispiel 3.2.3 (Binäre Addition)

Aufgabe:

Erstelle eine Turing-Maschine, die zwei binäre Zahlen addiert. **Binäre Addition**

Ausgangslage:

Die zwei Binärzahlen werden untereinander auf das Turingband geschrieben, wobei die kürzere von beiden mit führenden Nullen aufgefüllt wird, so dass beide Zalen die gleiche Länge besitzen.

Schlussbedingung:

Als Ergebnis steht die Summe der beiden Binärzahlen auf dem Turingband.

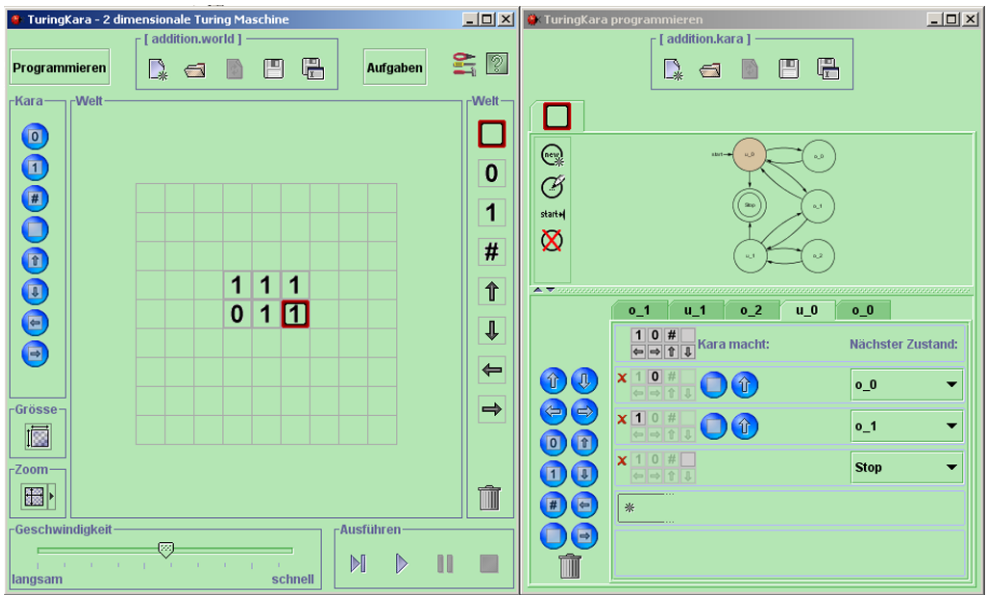

Abbildung 3.5: Addition zweier Binärzahlen mit TuringKara

Die Addition zweier Binärzahlen mit Hilfe von TuringKara kann noch einfacher als mit dem Standardmodell dargestellt werden, da man hier von der Fähigkeit, mehrere Symbole zu schreiben und mehrere Kopfbewegungen pro Übergang durchzuführen, Gebrauch machen kann. In Abbildung 3.5 sind die einzelnen Übergänge dargestellt.

Definiert wird eine Kara-Turingmaschine $T_{KARA} = (Q, \Sigma, \Gamma, \Box, u_o, Stop, \delta)$ zur Addition zweier Binärzahlen mit $\Sigma = \{0,1\}$, $\Gamma = \{0,1,\Box\}$ und δ wie in Abbildung 3.6 dargestellt, deren Arbeitsweise der in Beispiel 2.2.2 vorgestellten Turingmaschine entspricht. Es werden jedoch nur noch fünf Zustände benötigt, indem auf die Zustände $s_{0,1,2,3}$ verzichtet werden kann.

δ	0	1	$\Box$
u_0	$(o_0, (\Box \Uparrow))$	$(o_1, (\Box \Uparrow))$	$(Stop, (\Box))$
u_1	$(o_1, (\Box \Uparrow))$	$(o_2, (\Box \Uparrow))$	$(Stop, (1))$
o_0	$(u_0, (\Box \Downarrow 0 \Leftarrow))$	$(u_0, (\Box \Downarrow 1 \Leftarrow))$	-
o_1	$(u_0, (\Box \Downarrow 1 \Leftarrow))$	$(u_1, (\Box \Downarrow 0 \Leftarrow))$	-
o_2	$(u_1, (\Box \Downarrow 0 \Leftarrow))$	$(u_1, (\Box \Downarrow 1 \Leftarrow))$	-

Abbildung 3.6: Zustandsüberführungsfunktion einer Kara-Turingmaschine zur Addition zweier Binärzahlen

Beispiel 3.2.4 (Traversieren eines Labyrinths)

Traversieren eines Labyrinths

Aufgabe:

Erstelle eine Turing Maschine, die ein beliebiges Labyrinth mit Nullen füllt.

Ausgangslage:

Der Rand des Labyrinths ist durch #-Symbole markiert. Die Position des Lese-/Schreikopfs am Anfang ist beliebig innerhalb des Labyrinths.

Schlussbedingung:

Jedes Feld innerhalb des Labyrinths muss mit einer 0 belegt sein.

Unter der Traversierung eines Labyrinths versteht man das vollständige Besuchen der Wege in einem Labyrinth. Der Lösungsansatz hierbei besteht darin, für jedes Feld der Labyrinthwege festzuhalten, aus welcher Richtung es betreten wurde. Abbildung 3.7 zeigt ein Labyrinth, dessen Wege durch das Symbol # voneinander getrennt sind. Mit Hilfe der Pfeile, welche in TuringKara zur Verfügung stehen, kann das gesamte Labyrinth besucht und markiert werden, um es zuletzt mit Nullen zu füllen. Das Programm für diese Traversierung mit Hilfe von TuringKara ist in Abbildung 3.8 zu sehen (vgl. auch [RNH04]).

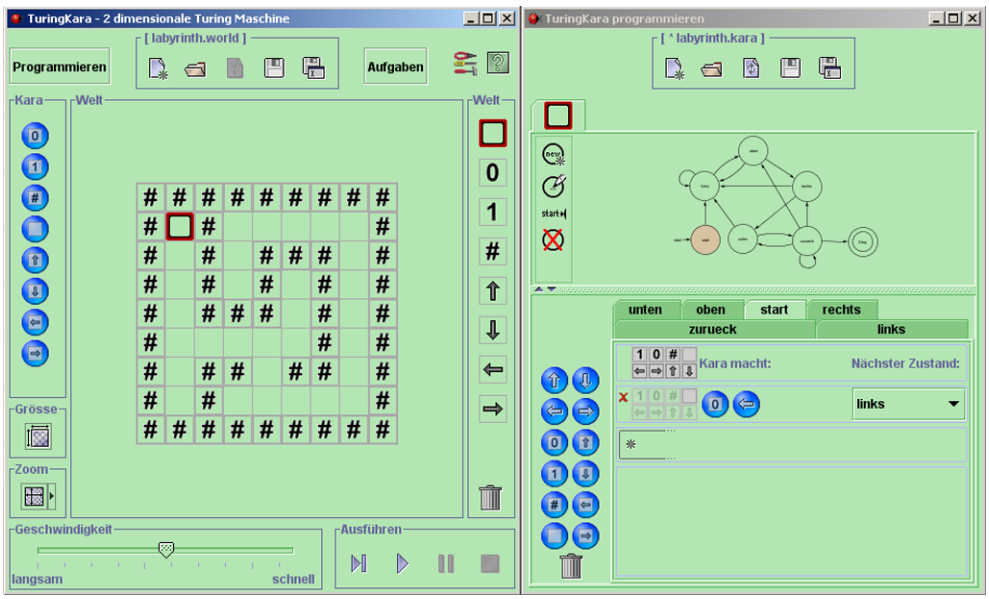

Abbildung 3.7: Traversierung eines Labyrinths mit TuringKara

Gegeben ist eine Kara-Turingmaschine $T_{KARA} = (Q, \Sigma, \Gamma, \Box, start, Stop, \delta)$ zur Traversierung eines Labyrinths durch Füllen aller Wege mit Nullen, wobei $\Sigma = \{0, \Box, \#\}$, $\Gamma = \{0, \Box, \#, \leftarrow, \rightarrow, \uparrow, \downarrow\}$, $Q = \{start, links, rechts, oben, unten, zurueck, Stop\}$ und δ wie in Abbildung 3.8 dargestellt.

δ	0	$\Box$	#	$\leftarrow$	$\rightarrow$	$\uparrow$	$\downarrow$
start	-	$(links, (0 \Leftarrow))$	-	-	-	-	-
links	$(oben, (\Rightarrow \Uparrow))$	$(links, (\rightarrow \Leftarrow))$	$(oben, (\Rightarrow \Uparrow))$	$(oben, (\Rightarrow \Uparrow))$	$(oben, (\Rightarrow \Uparrow))$	$(oben, (\Rightarrow \Uparrow))$	$(oben, (\Rightarrow \Uparrow))$
rechts	$(unten, \Leftarrow \Downarrow)$	$(links, \leftarrow \Leftarrow)$	$(unten, \Leftarrow \Downarrow)$	$(unten, \Leftarrow \Downarrow)$	$(unten, \Leftarrow \Downarrow)$	$(unten, \Leftarrow \Downarrow)$	$(unten, \Leftarrow \Downarrow)$
oben	$(rechts, \Downarrow \Rightarrow)$	$(links, \downarrow \Leftarrow)$	$(rechts, \Downarrow \Rightarrow)$	$(rechts, \Downarrow \Rightarrow)$	$(rechts, \Downarrow \Rightarrow)$	$(rechts, \Downarrow \Rightarrow)$	$(rechts, \Downarrow \Rightarrow)$
unten	(zurück, $(\Uparrow)$)	$(links, (\uparrow \Leftarrow))$	$(zurueck, (\Uparrow))$	$(zurueck, (\Uparrow))$	$(zurueck, (\Uparrow))$	$(zurueck, (\Uparrow))$	$(zurueck, (\Uparrow))$
zurueck	$(Stop, (0))$	-	-	$(oben, (0 \Rightarrow \Uparrow))$	$(rechts, (0 \Downarrow \Rightarrow))$	$(unten, (0 \Leftarrow \Downarrow))$	$(zurueck, (0 \Uparrow))$

Abbildung 3.8: Zustandsüberführungsfunktion einer Kara-Turingmaschine zur Traversierung eines Labyrinths

3.2.3 Offenes Problem in der TuringKara-Umgebung

Auch in der Schulinformatik lassen sich Offene Probleme der Berechenbarkeitstheorie thematisieren. Langtons Ameise lässt sich auf einfache Weise in der TuringKara-Umgebung implementieren. Sie bietet eine interessante Anwendung für Schüler, um sich mit der Thematik von offenen Porblemen auseinanderzusetzen. Der Kniff der Ameise ist, dass sich die Schüler überlegen, wie sie den Richtungssinn der Ameise umsetzen. Den Schülern kann folgende Aufgabe gestellt werden:

Langtons Ameise in der TuringKara-Umgebung

Aufgabe:

Erstelle eine Turingmaschine, die das Verhalten einer Ameise simuliert. Die Ameise sitzt auf einem karierten Blatt Papier, schaut in Richtung Norden und verhält sich nach folgenden Regeln:

- Ist das Kästchen, auf welchem sie sich befindet weiß, so färbt sie es schwarz, dreht sich nach rechts und geht ein Kästchen vorwärts.
- Ist das Kästchen, auf welchem sie sich befindet schwarz, so färbt sie es weiß, dreht sich nach links und geht ein Kästchen vorwärts.

Anschließend beginnen die Schüler sich mit dieser Aufgabe auseinanderzusetzen. Dies erfordert bereits etwas an Erfahrung im Umgang mit der Erstellung von Turingmaschinen. Sie werden Folgendes festhalten können:

Ausgangslage:

Der Kopf der Turingmaschine befindet sich auf einem leeren Blatt Papier.

Schlussbedingung:

Da die Ameise keinen Endzustand hat, gibt es keine Schlussbedingung.

Implementiert wird eine Kara-Turingmaschine $T_{KARA} = (Q, \Sigma, \Gamma, \Box, norden, Stop, \delta)$ mit $\Sigma = \Gamma = \{\Box, \#\}$, $Q = \{norden, sueden, westen, osten\}$ und δ wie in 3.9 zur Simulation des Ameisenverhaltens. Die zentrale Idee bei der Erstellung einer Turingmaschine, welche das Verhalten der Ameise simuliert, ist, vier Zustände zu wählen, welche den Richtungssinn der Ameise ausdrücken. Dies kann in analoger Weise zu Abschnitt 2.2.3 „Offene Probleme und Unlösbarkeit“ Beispiel 2.2.9 geschehen. Hier kann auch die Beschreibung der Übergangsregeln nach gelesen werden. Da in der TuringKara-Umgebung nicht das Symbol ■ zur Verfügung steht, wird hier das Symbol # zur Färbung der Kästchen gewählt.

δ	$\square$	$\#$
$norden$	$(osten, (\# \Rightarrow))$	$(westen, (\square \Leftarrow))$
$sueden$	$(westen, (\# \Leftarrow))$	$(osten, (\square \Rightarrow))$
$westen$	$(norden, (\# \Uparrow))$	$(sueden, (\square \Downarrow))$
$osten$	$(sueden, (\# \Downarrow))$	$(norden, (\square \Uparrow))$

Abbildung 3.9: Zustandsüberführungsfunktion einer Kara-Turingmaschine zur Simulation des Verhaltens der Ameise

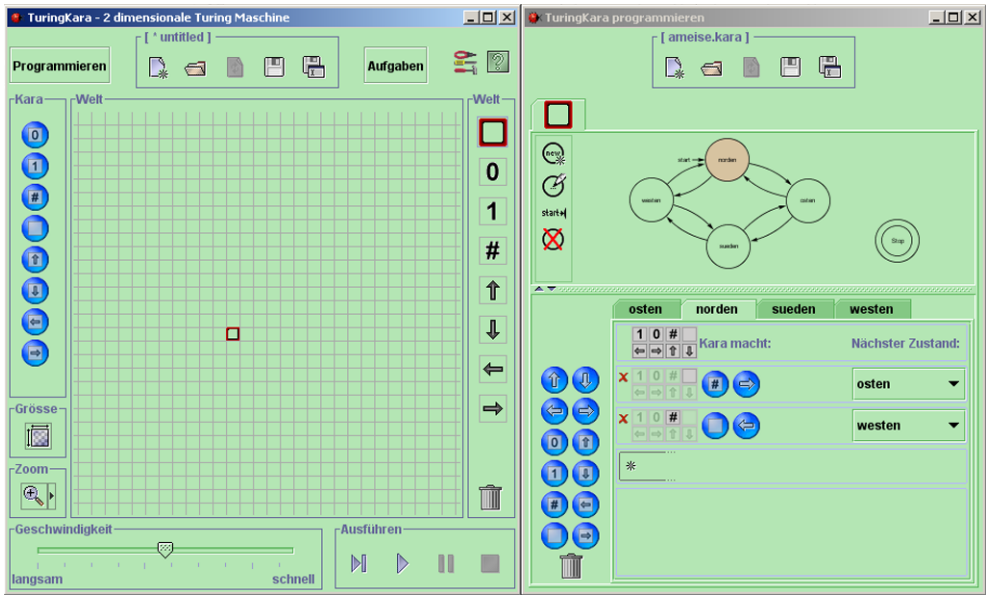

Abbildung 3.10: Langtons Ameise in der TuringKara-Umgebung

„Was macht die Ameise?“

Ist dieser Schritt getan, können die Schüler das Verhalten der Ameise einmal beobachten, indem sie ihr erstelltes Ameisen-Programm (vgl. Abbildung 3.10) einige Minuten laufen lassen und es beobachten. Das Papier muss möglichst groß gewählt und die Schrittgeschwindigkeit der Ameise hoch eingestellt werden. Der entscheidende Schritt hierbei ist, dass die Schüler die *Autobahnen* (vgl. Abb. 3.11) der Ameise erkennen.

Anschließend wird mit den Schülern in einem offenen Klassengspräch das Verhalten der Ameise analysiert, indem man gemeinsam überlegt, was man über das Verhalten der Ameise aussagen kann. Dabei werden Begriffe wie *nicht-vorhersehbar* und *offenes Problem* thematisiert.

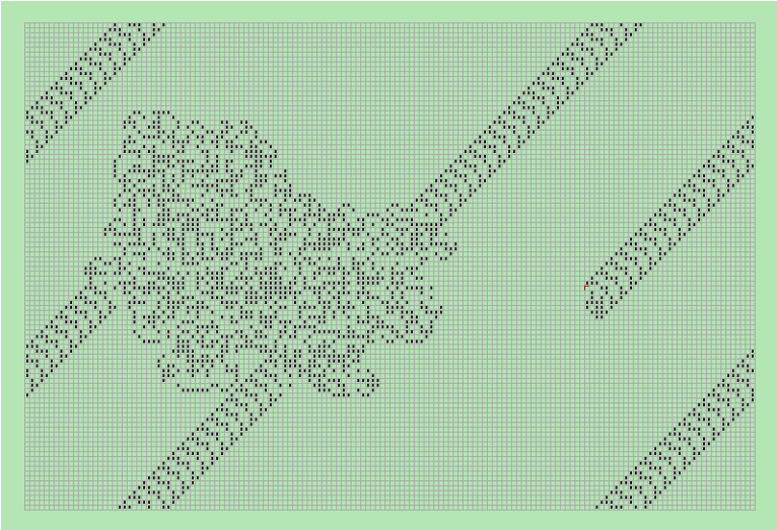

Abbildung 3.11: Verhalten der Ameise in der TuringKara-Umgebung

3.2.4 Die eindimensionale Turingmaschine

Um mit den Schülern im weiteren Verlauf die universelle Turingmaschine thematisieren zu können (vgl. Abschnitt 3.2.5 „Die universelle Turingmaschine in der TuringKara-Umgebung), ist es notwendig, die eindimensionale Turingmaschine zu besprechen. Dies kann in wenigen Schritten geschehen, indem die Schüler folgenden Arbeitsauftrag erhalten:

Definition der 1D-Turingmaschine

Aufgabe:

Finde eine Definition einer Turing-Maschine, die auf einem eindimensionalen Arbeitsband arbeitet. Orientiere Dich an der Definition der zweidimensionalen Turingmaschine, indem Du Dir überlegst, was sich bei der Reduktion auf ein eindimensionales Arbeitsband ändert.

Formale Definiton der eindimensionalen Kara-Turingmaschine

Definition 3.2.5 *Eine* eindimensionale Kara-Turingmaschine *ist gegeben durch ein 7-Tupel* $T_{KARA} = (Q, \Sigma, \Gamma, \Box, start, Stop, \delta)$ *mit*

- *einer endlichen Zustandsmenge* Q,
- *dem Eingabealphabet* $\Sigma \subseteq \{0, 1, \#, \Box, \leftarrow, \rightarrow, \uparrow, \downarrow\}$,
- *dem Bandalphabet* $\Gamma \subseteq \{0, 1, \#, \Box, \leftarrow, \rightarrow, \uparrow, \downarrow\}$,
- *dem Blanksymbol* $\Box$

- *dem Anfangszustand* $start \in Q$,
- *dem Endzustand* $Stop \in Q$
- *und der Übergangsfunktion* $\delta : (Q \backslash \{Stop\} \times \Sigma) \rightarrow (Q \times (\Sigma \cup \{\Leftarrow, \Rightarrow\})^{\star})$.

Hierbei ist es wichtig mit den Schülern die entsprechenden Änderungen vom eindimensionalen zum zweidimensionalen Modell festzuhalten und zu verstehen. Die einzige Reduktion, die hier vollzogen wird, ist, dass das Turingpapier nun ein Turingband ist und der Kopf der Maschine sich nur noch nach rechts und links auf dem Turinband hin- und herbewegen kann. Die Schüler können grafisch solch ein Turingband einmal darstellen, indem sie in der TuringKara-Umgebung die Größe des Bandes festlegen (vgl. Abb. 3.12).

1D vs. 2D-Modell

Grafik der 1D-Turingmaschine in TuringKara

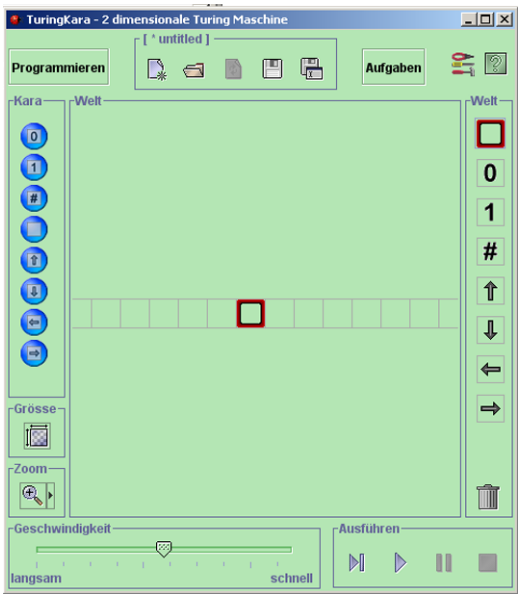

Abbildung 3.12: Die eindimensionale Turingmaschine in der TuringKara-Umgebung

Anschließend können die Schüler sich mit nachfolgender Aufgabe, welche auch in der TuringKara-Umgebung integriert ist, mit Hilfe einer eindimensionalen Turingmaschine auseinandersetzen.

Aufgabe zur 1D-Turingmaschine

Aufgabe:

Erstelle eine Turing Maschine, die für eine beliebige Zeichenkette feststellt, ob sie aus n Nullen gefolgt von n Einsen besteht. Überlege wieder Ausgangslage und Schlussbedingung dieser Aufgabe.

Ausgangslage:

Die Zeichenkette befindet sich auf den Feldern der Turingmaschine. Der Lesekopf steht am Anfang ganz links.

Schlussbedingung:

Falls es sich bei der Eingabe um eine korrekte Zeichenkette handelt, soll die Turing Maschine auf dem leeren Band terminieren. Ist die Eingabe nicht korrekt, darf das Band am Schluss nicht leer sein.

Es wird eine eindimensionale Kara-Turingmaschine $T_{KARA} = (Q, \Sigma, \Gamma, \Box, links, Stop, \delta)$ mit $\Sigma = \{0,1\}$, $\Gamma = \{0,1,\Box\}$, $Q = \{links, rechts, null, eins\}$ und δ wie in 3.13 implementiert, welche eine Zeichenkette bestehen aus n Nullen gefolgt von n Einsen akzeptiert. Sie erhält als Eingabe eine Zeichenkette bestehend aus Nullen und Einsen. Der Kopf der Turingmaschine startet auf dem linken Zeichen der Zeichenkette und wandert auf dieser immer abwechselnd vom linken zum rechten Ende der Zeichenkette, an denen eine 0 bzw. eine 1 gelöscht wird. Bleiben keine Zeichen übrig, so bestand die Zeichenkette aus n Nullen gefolgt von n Einsen. Die Turingmaschine akzeptiert die Eingabe.

δ	0	1	$\Box$
$links$	$(links, (\Leftarrow))$	$(links, (\Leftarrow))$	$(null, (\Box \Rightarrow))$
$null$	$(rechts, (\Rightarrow))$	-	$(Stop, (\Box))$
$eins$	-	$(links, (\Box \Leftarrow))$	-
$rechts$	$(rechts, (\Rightarrow))$	$(rechts, (\Rightarrow))$	$(eins, (\Box \Leftarrow))$

Abbildung 3.13: Zustandsüberführungsfunktion einer eindimensionalen Kara- Turingmaschine zur Untersuchung einer Zeichenkette bestehend aus n Nullen gefolt von n Einsen

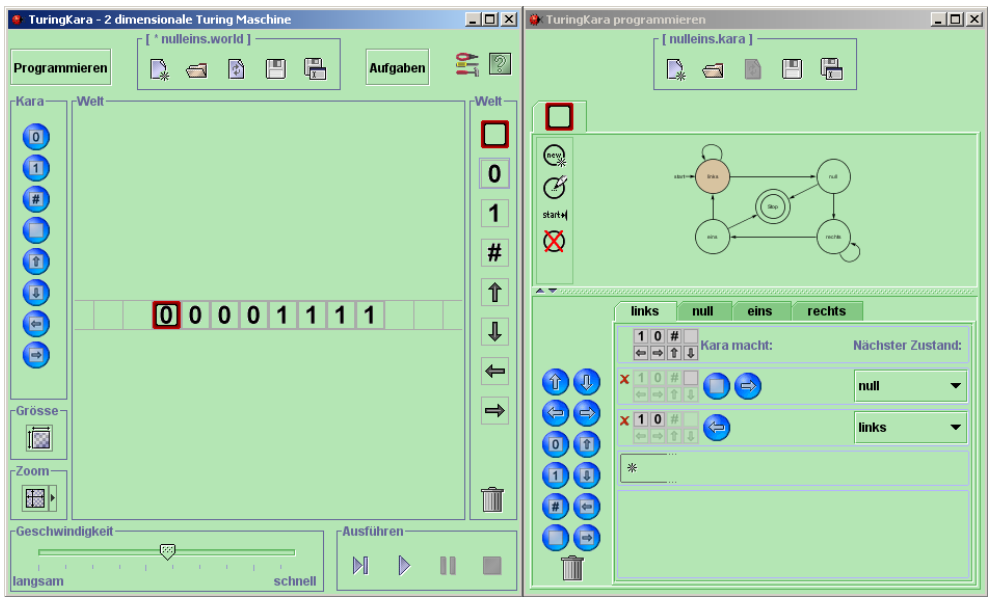

Abbildung 3.14: Untersuchung einer Zeichenkette bestehen aus n Nullen gefolgt von n Einsen in der TuringKara-Umgebung

Im nächsten Schritt kann man den Schülern folgende Aufgabe stellen, welche jedoch sehr zeitintensiv ist:

Aufgabe:

Überlege, wie die Addition zweier Binärzahlen mit Hilfe einer eindimensionalen Turingmaschine arbeitet. Erstelle anschließend eine eindimensionale Turingmaschine, welche zwei Binärzahlen addiert. Finde wieder eine geeignete Ausgangslage und Schlussbedingung der Berechnung. Vergleiche zuletzt Dein Ergebnis mit der Arbeitsweise der zweidimensionalen Turingmaschine.

Addition mit einer 1D-Turingmaschine

Aus Zeitgründen genügt es, auch hier nur die Arbeitsweise zu besprechen, um Vergleiche zwischen der Arbeit einer eindimensionalen und der einer zweidimensionalen Turingmaschine ziehen zu können. Die genaue Implementation wird analog zu der in Abschnitt 2.2.2 „Reduktion auf das eindimensionale Modell“ durchgeführt, welche wirklich aufwendig ist. Gerade dies sollen die Schüler erkennen, und damit die Vorteile einer zweidimensionalen Maschine gegenüber dem eindimensionalen Modell erfassen. Die Vorteile können stichwortartig gesammelt werden.

Vergleich von 1D- und 2D-Turingmaschine

In einerm offenen Klassengespräch wird anschließend mit den Schülern diskutiert, ob man durch die Reduktion von einem eindimensionalen Modell auf ein Zweidimensionales eingeschränkt wird. Die Schüler sollen sich Gedanken machen, ob alle Probleme, welche mit einer zweidimensionalen Maschine auch mit einer eindimensionalen Maschine gelöst werden können. Der Beweis der Äquivalenz dieser beiden Modelle kann mit den Schülern allerdings nicht thematisiert werden. Hier geht es darum, die Äquivalenz zu verstehen und nicht zu beweisen. In diesem Zusammenhang kann zuletzt die Churchsche These angeführt werden und mit den Schülern die Aussage, dass alle Maschinenmodelle die gleiche Klasse von Problemen lösen können, erörtert werden.

1D-Modell ⇔ 2D-Modell?

3.2.5 Die universelle Turingmaschine bei [RNH04]

Den Abschluss einer Unterrichtseinheit über ausgewählte Themen der Berechenbarkeitstheorie bildet die Entwicklung einer universellen Maschine in der TuringKara-Umgebung. Sie spielt eine wichtige Rolle in der Berechenbarkeitstheorie, da sie die Idee von programmierbaren Maschinen veranschaulicht. Mit Hilfe der universellen Turingmaschine wird den Schülern illustriert, wie eine Maschine arbeitet, die jede andere Maschine simulieren kann. Als Unterrichtsform kann hier die Methode eines gemeinsamen schulischen Projekts gewählt werden, da es sich hierbei um eine längere Aufgabe handelt.

[RNH04] haben das Programm einer universelle Kara-Turingmaschine in der TuringKara-Umgebung mit 41 Zuständen implementiert (vgl. Abbildung 3.16 links), welche eindimensionale Standardturingmaschinen simuliert. Es wird auf eine Beschreibung der Arbeitsweise dieser Turingmaschine an dieser Stelle verzichtet. Die Implementierung der universellen Kara-Turingmaschine ist in jeder TuringKara-Umgebung unter dem Menüeintrag „Aufgaben“ zu finden[1].

Kodierung einer 1D-Turingmaschine mit TuringKara

Für die Simulation einer beliebigen eindimensionalen Turingmaschine durch eine universellen Kara-Turingmaschine ist es zunächst nötig, eine geeignete *Kara-Kodierung* der beliebigen eindimensionalen Turingmaschine in der TuringKara-Umgebung zu finden, wie dies auch in Abschnitt 2.2.4 „Idee der universellen Turingmaschine“ durchgeführt wurde. Diese Kodierung wird mit den Schülern gemeinsam gefunden.

[1] Die Implementierung der universellen Kara-Turingmaschine ist in der TuringKara-Umgebung enthalten, sofern man die Variante mit den Lösungen runtergeladen hat. Alle Downloads der verschiedenen Kara-Varianten und -Versionen stehen unter http://swisseduc.ch/informatik/karatojava/download.html zur Verfügung

Im Folgenden wird die Kara-Kodierung einer eindimensionalen Turingmaschine in der TuirngKara-Umgebung definiert, wie dies in [RNH04] durch geführt wurde. Die zu simulierende Turingmaschine ist hier eine eindimensionale Standardmaschine. Das heißt, dass zum einen bei einem Übergang von einem Zustand in einen neuen Zustand nur ein Zeichen geschrieben und ein Bewegungsbefehl vollzogen wird und zum anderen nur die Kopfbewegungen rechts, links oder stehen bleiben zur Verfügung steht. Außerdem müssen die Symbole des Eingabe- und Bandalphabets aus der Menge $\{0,1,\#,\Box\}$ sein. Die Kara-Kodierung einer eindimensionalen Turinmaschine haben [RNH04] in der TuringKara-Umgebung wie folgt definiert:

Definition 3.2.6 *Sei $T = (Q,\Sigma,\Gamma,\Box,q_{start},q_{stop},\delta)$ eine eindimensionale Turingmaschine mit $\Sigma \subseteq \Gamma = \{0,1,\#,\Box\}$, $Q = \{q_{start},\ldots,q_{stop}\}$ und Überführungsfunktion δ. Dann wird der Zustandsübergang $\delta(q,x) = (q',x',m)$, wobei $q,q' \in Q$, $x,x' \in \Gamma$ und $m \in \{\leftarrow,\Box,\rightarrow\}$ wie folgt kodiert:*

$bin(q)$	#	x	$\rightarrow$	$bin(q')$	#	x'	#	m	$\rightarrow/\Box$

Mit $bin(q)$ wird die Binärkodierung des Zustands $q \in Q$ bezeichnet. Die Zustände werden binär durchnummeriert. Wird ein Übergang von einem Zustand zu einem neuen Zustand kodiert, der kein Endzustand ist, ist das letzte Symbol der Kodierung ein $\rightarrow$. Wird jedoch ein Übergang in einen Endzustand kodiert, ist das letzte Zeichen ein $\Box$.

Die Kodierung des j-ten Übergangs wird mit $code(j)$ bezeichnet. Die Kara-Kodierung *der Turingmaschine T ist bei s Übergängen gegeben durch nachstehende Tabelle (vgl. Abb. 3.15), wobei jeder Übergang durch eine Leezeile getrennt ist. Den Kodierungen der Übergänge wird die Kodierung des Startzustandes mit einem $\rightarrow$ und einem folgenden # vorangestellt. Eine Zeichenfolge bestehend aus dem Symbol # unter den Kodierungen der Übergänge trennt die Kara-Kodierung von dem eindimensionalen Eingabewort w. Das einzelne Symbol # in der sich darüberbefindenden Zeile dient der Markierung der aktuellen Kopfposition der zu simulierenden Turingmaschine.*

	$\rightarrow$	$bin(q_{start})$	#									
		$code(1)$										
		$code(2)$										
		$\vdots$										
		$code(s)$										
#												
#	#	#	#	#	#	#	#	#	#	#	#	#
w												

Abbildung 3.15: Eingabe der universellen Kara-Turingmaschine nach [RNH04]

In Abbildung 3.16 ist die Kara-Kodierung einer Turingmaschine zur Invertierung eines Bitstrings dargestellt.

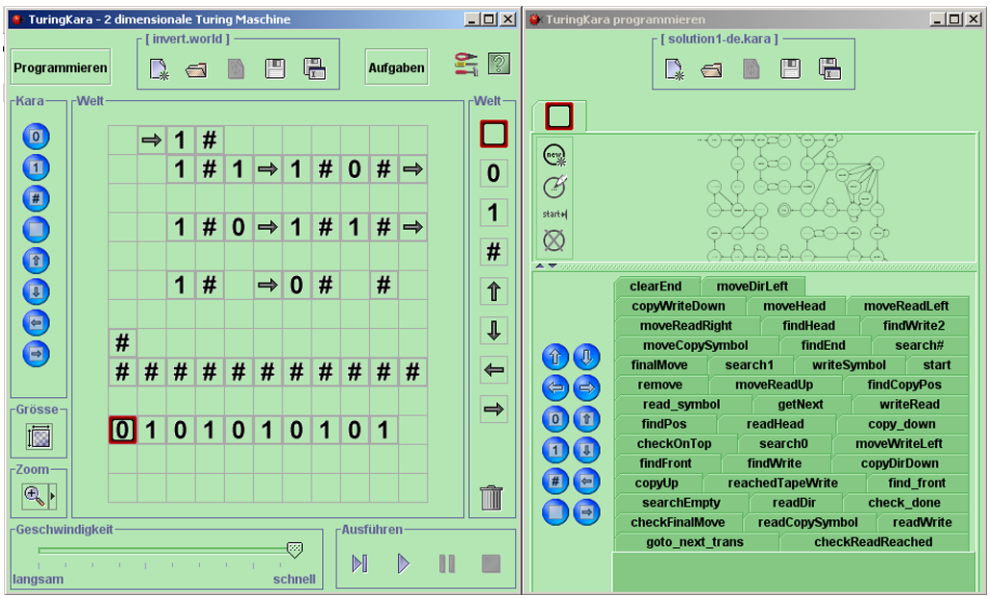

Abbildung 3.16: Die universelle Turingmaschine in der TuringKara-Umgebung

Kodieren von 1D-Turingmaschinen

Die Kara-Kodierung wird mit den Schülern besprochen, um anschließend einfache Turingmaschine selbst zu kodieren. Damit die Schüler die Arbeitsweise der universellen Turingmaschine verstehen, ist es hilfreich, das in der TuringKara-Umgebung integrierte Beispiel zur universellen Turingmaschine (Invertieren einer Zeichenkette, vgl. Abbildung 3.16) einmal durchlaufen zu lassen und die einzelnen Arbeitsschritte zu beobachten.

Testen der vorimplementierten universellen Kara-Turingmaschine an eigenen Beispielen

Die Implementierung der Simulation ist sehr aufwendig und nimmt einiges an Zeit mit den Schülern in Anspruch. Aus diesem Grund ist es zeitmäßig nicht in den Unterricht integrierbar. Auch die Sinnhaftigkeit muss jeder Lehrer selbst entscheiden. Der entscheidende Schritt bei der universellen Turingmaschine ist, eine geeignete Kodierung der zu simulierenden Turingmaschine zu finden. Dies geschieht mit den Schülern gemeinsam. Die Schüler kodieren eindimensionale Turingmaschinen, wie dies in Abbildung 3.16 links durchgeführt wird. Anschließend testen sie die in der TuringKara-Umgebung vorimplementierte universelle Turingmaschine an ihren Beispielen, um eine Simulation zu starten. So ist es ihnen möglich, einen anschaulichen Einblick in die Arbeitsweise einer universellen Turingmaschine zu bekommen, womit der Abschluss einer Unterrichtseinheit zur Berechenbarkeitstheorie erreicht ist.

Abschließend sei kritisch angemerkt, dass die universelle Turingmaschine in der TuringKara-

Umgebung gerade den zweidimensionalen Vorteil, den Turing- Kara bietet, nicht ausnutzt. TuringKara verpasst es, das Programm einer zweidimensionalen Turingmaschine für deren Kodierung zu wählen, da dieses bereits in Form einer zweidimensionalen Tabelle vorliegt. Hier sind TuringKara Grenzen gesetzt, da der Zeichenvorrat eingeschränkt ist und dies nicht hergibt. TuringKara fehlt die Eigenschaft, einen beliebigen Zeichenvorrat zu definieren. Um den Schülern diesen Vorteil demonstrieren zu können, muss die Lernsoftware verlassen werden. In einem abschließenden Klassengespräch kann dieser entscheidende Vorteil einer zweidimensionalen Turingmaschine diskutiert werden.

Kapitel 4

Zusammenfassung

Mit Hife zweidimensionaler Berechnungsmodelle lassen sich auf anschauliche Weise Eindrücke der Berechenbarkeitstheorie vermitteln. Sie stellen Themen und Begriffe rund um die Berechenbarkeitstheorie anschaulich dar. Die Kerninhalte der Theoretischen Informatik werden auf das Wesentliche reduziert. An einfachen und überzeugenden Beispielen lassen sich zentrale Themen vorstellen, die es ermöglichen, ein Verständnis der abstrakten Theorie zu erlangen. Wie der Vergleich von eindimensionalem und zweidimensionalem Modell zeigt, lassen sich anschaulich und schnell Beispiele aus der Berechenbarkeit darstellen. Ein zentraler Vorteil des zweidimensionalen Ansatzes, besonders fuer den Einsatz in der Schule, liegt darin, dass den Schülern das abstrakte Modell genau auf ihrem Erfahrungshintergrund des algorithmischen Rechnens mit Papier und Stift nahegebracht wird. Weitere Argumente auch auf theoretischer Ebene zur Einführung universeller Maschinen und zur Verbindung mit zellulären Automaten wurden in dieser Arbeit ebenfalls vorgestellt. Zum Beispiel bietet die zweidimensionale universelle Turingmaschine den Vorteil, dass für die Eingabe von Turingmaschinen keine spezielle Kodierung gefunden werden muss. An den theoretischen Teil schließen sich didaktische Überlegungen an, welche einen zweidimensionalen Berechenbarkeitsansatz für die Schul-Informatik einführen. Hier kommt die Möglichkeit hinzu, mit TuringKara interessante und praktische Übungen durchzuführen. Der zweidimensionale Ansatz stellt eine attraktive und vielleicht vorzuziehende Alternative zum eindimensionalen Ansatz dar, wie er bisher in den Lehrbüchern vorherrscht.

Literaturverzeichnis

[Bau93] BAUMANN, Rüdeger: *Informatik für die Sekundarstufe II - Band 2 - Höhere Datentypen, Automaten, Sprachen*. Klett, 1993. – ISBN 3–12–717742–9

[BCG85] BERLEKAMP, Elwyn R. ; CONWAY, John H. ; GUY, Richard K.: *Gewinnen - Strategien für mathematische Spiele - Band 4 - Solitairspiele*. Addison-Wesley, 1985. – ISBN 3–528–08534–7

[Cod68] CODD, E.F.: *Cellular Automata*. Academic Press, 1968

[Dew93] DEWDNEY, A.K.: *Der Turing Omnibus*. Springer-Verlag, 1993. – ISBN 3–540–57780–0

[Fra96] FRANKE, Herbert W.: *Das Lebensspiel und andere Gitterautomaten*. 1996. – http://www.heise.de/tp/r4/artikel/6/6040/1.html

[Hro04] HROMKOVIČ, Juraj: *Theoretische Informatik*. Wiesbaden : Teubner, 2004. – ISBN 3–519–10332–X

[HU90] HOPCROFT, John E. ; ULLMAN, Jeffrey D.: *Einführung in die Automatnetheorie, Formale Sprachen und Komplexitätstheorie*. Addison-Wesley, 1990. – ISBN 3–89319–181–X

[Ila01] ILACHINSKI, Andrew: *Cellular Automata - A Discrete Universe*. World Scientific, 2001. – ISBN 981–02–4623–4

[Lor96] LORENZ, Falko: *Einführung in die Algebra I, §1 Konstruierbarkeit mit Zirkel und Lineal*. Spektrum Akademischer Verlag, 1996. – ISBN 3–86025–397–2

[Mod86] MODROW, Eckart: *Automaten Schaltwerke Sprachen*. Ferd. Dümmlers Verlag, 1986. – ISBN 3–427–42911–3

[MSW99] MSWWF, Land Nordrhein-Westfalen (Hrsg.): *Richtlinien und Lehrpläne für die Sekundarstufe II - Gymnasium und Gesamtschule in Nordrhein-Westfalen. Informatik.* Düsseldorf : Ritterbach, 1999. – ISBN 3–89314–612–1

[RNH04] REICHERT, Raimond ; NIEVERGELT, Jürg ; HARTMANN, Werner: *Programmieren mit Kara - Ein spielerischer Zugang zur Informatik.* Springer-Verlag, 2004. – ISBN 3–540–40362–0

[Sip97] SIPSER, Michael: *Introduction to the Theory of Computation.* PWS Publishing Company, 1997. – ISBN 0–534–94728–X

[Tho99] THOMAS, Wolfgang: *Berechenbarkeit und Komplexität - eine Mitschrift zur Vorlesung im WS 98/99.* 1999. – http://www.s-inf.de/Skripte/BuK.1998-WS-Thomas.(ME).SKRIPT.pdf

[Tur36] TURING, Alan M.: On computable numbers, with an application to the entscheidungsproblem. In: *Proceedings of the London Mathematical Society* Series 2(42) (1936), S. 230–265

[Vol79] VOLLMAR, Roland: *Algorithmen in Zellularautomaten.* Teubner Studienbücher, 1979. – ISBN 3–519–02350–4

Erklärung

Ich versichere, dass ich die schriftliche Hausarbeit - einschließlich beigefügter Zeichnungen, Kartenskizzen und Darstellungen - selbständig verfasst und keine als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle deutlich als Entlehnung kenntlich gemacht.

Aachen, den 2.07.2007

Katrin Moßbrucker