Tagung zur Schulinformatik

GI-FIBBB 2007

7. GI-Tagung der Fachgruppe

Informatik-Bildung in Berlin und Brandenburg

am 8. März 2007 (Freie Universität Berlin)

Workshop „Turingmaschinen”

Walter Gussmann (wagu1@web.de)
Mathias Müller (math.mueller@claranet.de)

Inhaltsverzeichnis

1 Turingmaschinen

Alan Turing veröffentlichte 1937 seinen berühmten Artikel „On Computable Numbers” (berechenbare Zahlen), in dem er eine virtuelle Maschine entwarf, mit der er bestimmte mathematische Operationen algorithmisch lösen konnte. Seine Arbeit war ein Meilenstein in der Entwicklung der theroetischen Grundlagen der Informatik und bestätigte die Theorie Gödels, dass nicht alle mathematischen Probleme beweisbar sind.

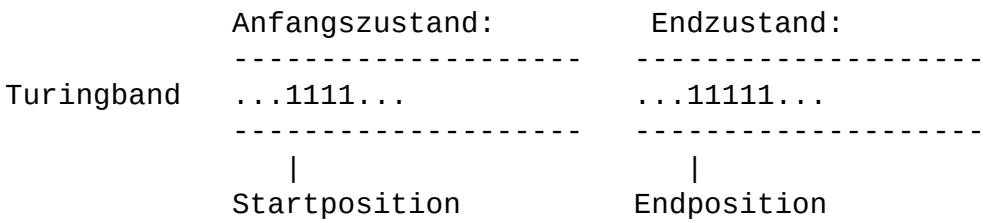
Turings Maschine wird durch einen (im Prinzip unendlich) langen Papierstreifen gefüttert (Eingabeband), auf dem die zu verarbeitenden Informationen stehen. Die Maschine besteht aus einem Lese-/Schreibkopf, der sich über dem Band bewegen lässt und jeweils genau ein Zeichen des Bandes lesen bzw. überschreiben kann.

Die Maschine befindet sich zu jeder Zeit in einem genau definierten Zustand und weiß, wie sie auf die möglichen Zeichen reagieren muss. Die Steuerung des Kopfes entspricht einem endlichen, deterministischen Automaten. Damit hatte Turing, lange vor der Entwicklung des ersten Computers, ein sehr einfaches Modell mit unvorstellbaren Möglichkeiten entwickelt.

Für den Schulunterricht ist entscheidend, motivierende Beispiele zu finden, die mit einem handlungsorientierten Zugang einfach umsetzbar sind. So steigt die Möglichkeit, dass auch theoretische Schlussfolgerungen (etwa zur Berechenbarkeit bzw. zum Halteproblem) Eingang in den Schulalltag finden können.

1.1 Meine erste Turingmaschine

Aufgabe: Es soll eine Maschine entworfen werden, die die Anzahl der Striche (Zeichen 1) auf einem Band um 1 erhöht. Zu Beginn befindet sich der Lesekopf über dem 1. Strich des Eingabebandes (Das Band kann aber auch leer sein). Am Ende soll die Maschine wieder auf dem 1. Strich stehen.



Verbale Problemlösung

- Zuerst bewegt man den Kopf solange nach rechts, bis die erste freie Stelle des Bandes erreicht ist. Dieser Schritt entfällt, wenn die aktuelle Position bereits leer ist.
- Jetzt überschreibt man das Band mit einer 1.
- Zum Schluss bewegt man den Kopf solange nach links, bis man auf einen freien Platz trifft. Von dort geht man dann noch einen Schritt nach rechts. Die Maschine beendet hier ihre Arbeit.

Pseudocode:

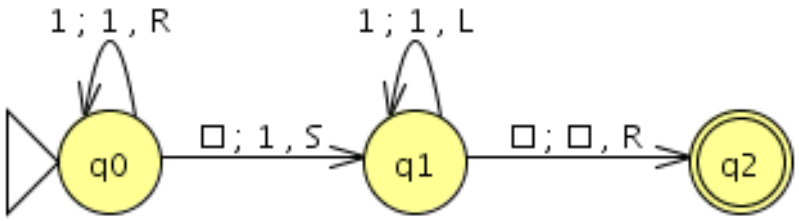
```
solange Band nicht leer,
   bewege Kopf nach rechts
Schreibe "1" auf Band (ohne Kopfbewegung)
Solange Band nicht leer,
   bewege Kopf nach links
Bewege Kopf nach rechts
Stopp
```

Zustandstabelle:

Anfangs-Zustand	altes Zeichen	End-Zustand	neues Zeichen	Kopf-Bewegung
Z0	1	Z0	1	rechts
Z0	ϵ	Z1	1	—
Z1	1	Z1	1	links
Z1	ϵ	Z2	ϵ	rechts
Z2	Endzustand			

ϵ = leer

Zustandgraph:



Die Angabe ”1;0;R” bedeutet, dass wenn die Maschine im aktuellen Zustand eine 1 liest, sie diese durch eine 0 ersetzt und anschließend den Lesekopf um eine Stelle nach rechts versetzt. Das Leerzeichen ist durch ein leeres Quadrat angedeutet.

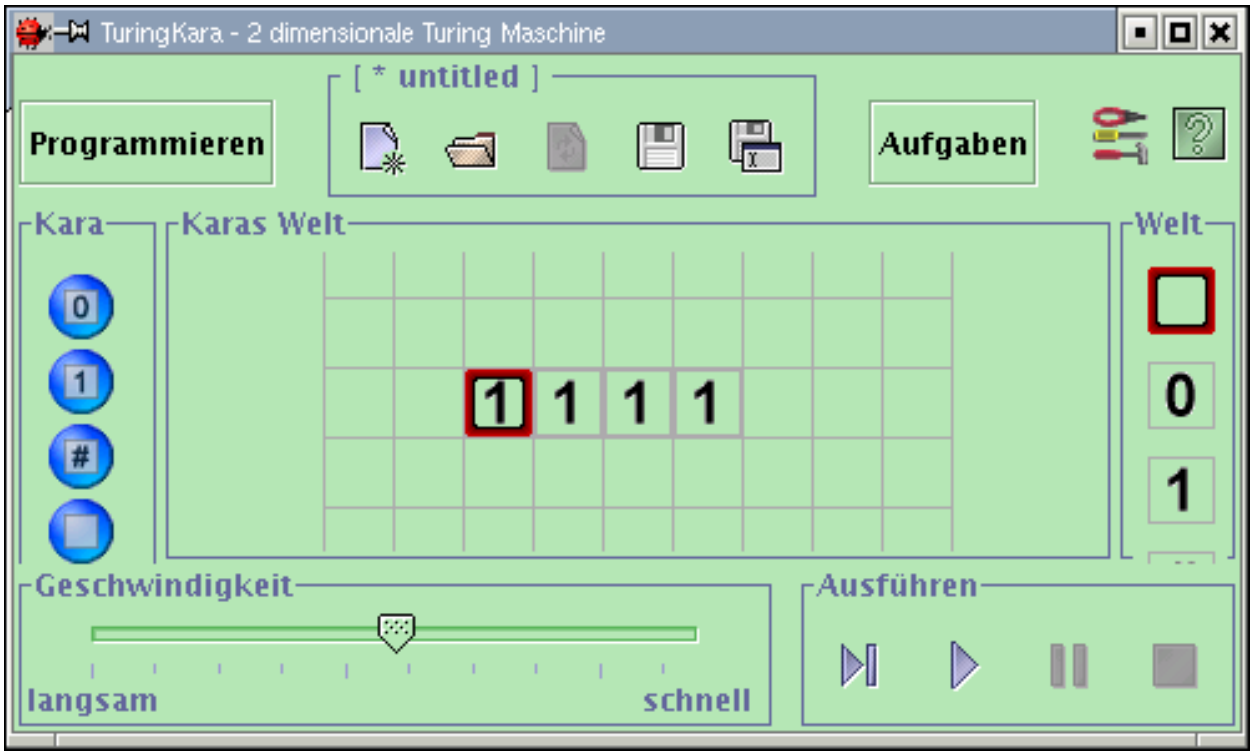
1.2 Das Kara-Turing-Modell

Die an der ETH Zürich entwickelte KARA-Familie bietet mit TuringKara ein Modul an, das eine zustandsorientierte Programmierung ohne viel Vorwissen erlaubt. Vorhandene Kenntnisse aus anderen KARA-Modellen lassen sich dabei problemlos übertragen. Das Kara-Modell kann unter `http://www.swisseduc.ch/informatik/karatojava` heruntergeladen werden.

Die *.jar*-Datei kann unter Windows durch einen Doppelklick im Explorer gestartet werden. Funktioniert dies nicht, dann fehlt eine Verknüpfung der Endung mit java. In diesem Fall kann das Programm über die Konsole gestartet werden:

java -jar allkara.jar

Zunächst öffnet sich das „Weltfenster", in dem sich das Eingabeband befindet. Die Umgebung eignet sich auch für zweidimensionale Turingmaschinen, soll aber zunächst nur mit einem Band (eine Zeile) benutzt werden. Dazu können mit der Maus aus dem rechten Bereich Symbole auf die Felder gezogen werden. Das rote Feld markiert die Anfangsposition der Turingmaschine.



Weltfenster (mit Eingabeband)

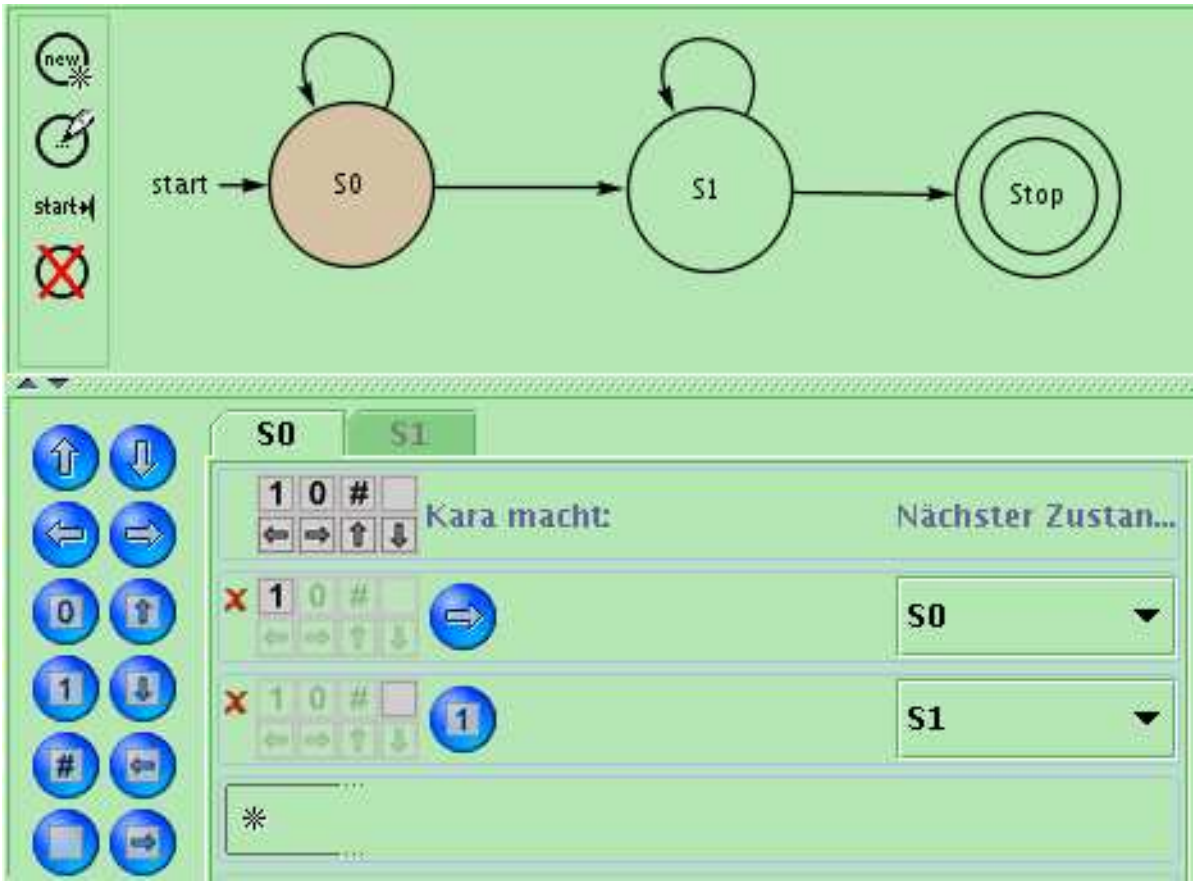
Ein Programm wird über die unteren, rechten Tasten gestartet. neben einem Einzelschrittmodus empfiehlt sich auch ein verzögerter Ablauf, wobei beide Fenster (Welt- und Programmierfenster) sichtbar sein sollten. Der aktuelle Zustand wird dabei im Programm farblich hervorgehoben.

Über die Schaltfläche „*Programmieren*" gelangt man in den Programmiereditor. Hier werden zunächst die benötigten Zustände angelegt, wobei ein Zustand als Startzustand zu markieren ist. Anschließend werden für jeden Zustand die benötigten Übergänge definiert. Dazu klickt man auf den Stern im unteren Bereich. Aus den möglichen Bandbelegungen (1,0,#,Leer, Pfeiltasten) wählt man die geeigneten aus und zieht die Aktion, die die Turingmaschine beim Lesen der markierten Zeichen erledigen soll mit der Maus in den freien Bereich. Zum Schluss muss noch der Folgezustand ganz rechts ausgewählt werden. Die definierten Übergänge erscheinen im oberen Zustands-

diagramm durch Pfeile (ohne Beschriftung).

Das folgende Bild zeigt die ausgewählten Aktionen im Startzustand $S0$. Die erste Zeile beschreibt die Aktion(en), die durchgeführt werden, wenn auf dem Band eine 1 gelesen wird. In diesem Fall muss nur der Kopf um eine Stelle nach rechts verschoben werden (Pfeil nach rechts). Die Maschine verbleibt im Zustand $s0$.

In der zweiten Zeile ist das Leerzeichen (Quadrat) ausgewählt. Hier soll eine 1 eingetragen werden, ohne weitere Bewegungen. Als Folgezustand ist $S1$ ausgewählt. Entsprechend enthält die Karteikarte für den Zustand $S1$ definierte Aktionen für die möglichen Eingabewerte. Ein Wert verweist von dort auf den Endzustand.



Programmierfenster (mit Sicht auf Zustand S0)

1.3 Allgemeine Beschreibung einer Turingmaschine

Die folgende Beschreibung stellt eine allgemeingültige Definition einer Turingmaschine dar. Sie kann insbesondere im Anfangsunterricht weggelassen werden, bietet aber im Leistungskurs eine gute Möglichkeit, auf bekannte formale Beschreibungen (endliche Automaten, Sprachen, etc.) bezug zu nehmen.

Unter einer deterministischen Turingmaschine mit beidseitig unendlichem Band versteht man ein Tupel

$$T = (Z, A, u, s, F, \epsilon)$$

wobei

- Z die Zustandsmenge,
- A das Bandalphabet,
- u eine Übergangsfunktion ($u : Z \times A \rightarrow Z \times A \times \{\triangleleft, \triangleright\}$)
- s der Startzustand,
- F mögliche Endzustände und
- ϵ das Leerzeichen des Bandes ist.

Die Zeichen $\triangleleft$ und $\triangleright$ geben die Bewegung des Kopfes nach links und rechts an.

1.4 Weitere Implementierungen

Das Karamodell bietet für Schüler eine motivierende Oberfläche und einen raschen Zugriff auf Lösungsansätze. Allerdings bringt Kara auch Einschränkungen mit sich. Insbesondere das doch sehr reduzierte Bandalphabet führt bei der Übertragung informatischer Fragestellungen zu Problemen. Hier bieten andere Umgebungen teils mächtigere Lösungsansätze.

JFLAP

Als Universalpaket bietet sich zunächst JFLAP an (`www.jflap.org`). Hier handelt es sich ebenfalls um ein in Java geschriebenes Programm, das alle wichtigen Maschinen der theoretischen Informatik enthält, leicht zu bedienen ist und an vielen Universitäten zur Modellierung eingesetzt wird. Das Programm ist frei erhältlich, allerdings nur in englischer Sprache. Der erste oben abgebildete Zustandsgraph wurde mit diesem Programm erstellt.

Mit der implementierten Turingmaschine können auch Blöcke (Makros) wiederkehrender Probleme erstellt und in eigene Programme integriert werden. Damit werden größere Programme übersichtlicher.

xTuringMaschineLab

Quelle: `http://math.hws.edu/TMCM/java/labs/xTuringMachineLab.html`

Das Java-Applet ermöglicht eine tabellarische Bearbeitung der Übergänge. Einige Aufgabenstellungen sind per Menü auswählbar. Der Programmablauf wird durch eine einfache Animation unterstützt. Auf den Seiten finden sich eine Reihe möglicher Aufgaben zu Turingmaschinen (auf englisch).

Matheprisma

Quelle: `http://www.matheprisma.uni-wuppertal.de/Module/Turing/`

Matheprisma bietet eine Lerneinheit zu Turingmaschinen an. Diese Einheit enthält auch ein Simulationsprogramm mit vorgefertigten Problemen inklusive Lösungen. Sie enthält umfangreiche Beschreibungen zur Turingmaschine, zum Programmieren und Theorie einer TM.

Visuelle Turingmaschine

Quelle: `http://www.dbg.rt.bw.schule.de/lehrer/ritters/info/turing/vtur.htm`

Andreas Rittershofer stellt ein Windows-programm zur Simulation von Turingmaschinen zur Verfügung. Auf seinen Seiten beschreibt er exemplarisch den Einsatz.

tm (FH-Wiesbaden)

Quelle: `http://wwwsys.informatik.fh-wiesbaden.de/weber1/turing/tm.html`

Das Applet bietet eine übersichtliche Oberfläche und enthält einige Beispielprogramme. Eigene Programme lassen sich erstellen, aber nicht direkt speichern. Im Internet findet man zahlreiche vergleichbare Turingmaschinen.

Allgemeine Beschreibung: `http://de.wikipedia.org/wiki/Turingmaschine`

1.5 Arbeiten mit Binärzahlen

Das (unendlich lange) Turingband enthält im Folgenden nur die Ziffern 1 und 0, deren Folge als Binärzahl interpretiert wird. Für die Zahl 11 sieht das etsprechende Band dann so aus:

```
Turingband:   ----------------------------
              . . . . . 1 0 1 1 . . . . .
              ----------------------------
                        | Startzustand
```

Es soll nun ein Programm entwickelt werden, das diese Zahl um 1 erhöht (Inkrementiermaschine). Zu Beginn soll die Maschine auf den Anfang der Binärzahl (most significant bit - msb) verweisen. Nach Ablauf des Programms soll die Maschine ebenfalls wieder am Anfang der Binärzahl stehen.

Lösungsidee:

- Zunächst muss der Kopf an das Ende der Zahl bewegt werden.
- Dann wird schrittweise von rechts nach links gelesen.
- Bei einer ’0’ wird diese durch ’1’ ersetzt und die Maschine wird auf den Anfang zurückgesetzt (Endzustand).
- Beim Lesen einer ’1’ wird diese durch ’0’ überschrieben. Die Maschine verbleibt in diesem Zustand, der Lesekopf wird aber um eine Stelle nach links verschoben.
- Trifft die Maschine auf ein leeres Zeichen, so wird dieses durch eine ’1’ ersetzt und die Maschine gerät in den Endzustand.

1.6 Zweidimensionale Turing-Maschinen

Eine der Stärken des Kara-Turingmodells liegt in der integrierten zweidimensionalen Welt. Das lineare Eingabeband kann problemlos auf eine Eingabeebene erweitert werden. Damit gleicht die Eingabe einem kariierten Papier, wobei jedes Karo für ein mögliches Feld steht.

In dieser Umgebung kann z. B. die Addition von (Binär-)Zahlen gemäß den in der Grundschule geübten Regeln für das addieren mehrerer Zahlen implementiert werden. Das Buch „*Programmieren mit Kara*"[1] beschreibt eine Lösung für das Addieren.

Dabei werden die zu addierenden Zahlen untereinander geschrieben, mit führenden Nullen ergänzt und durch Rauten begrenzt. Die Turingmaschine kommt dann mit vier Zuständen aus. Dabei befindet sich Kara auf der letzten Ziffer der obersten Zahl. Zunächst addiert Kara die beiden obersten Zahlen. Dabei überschreibt die Maschine die zweite Zahl. Anschließend wiederholt Kara diesen Vorgang, bis alle Zeilen addiert wurden.

Beispiel: (Ausgangszustand)

```
                                | Startzustand
                                v
Eingabefläche:    . # 0 1 0 1 1 1 0 #  .
                  . # 0 0 0 0 1 0 1 # .
                  . # 0 0 1 0 0 0 1 # .
                  . # 0 0 0 0 0 1 1 # .
                  . # . . . . . . . # .
```

Zwischenzustände nach jeweils einer zeilenweise durchgeführten Addition:

```
nach der
1. Addition          2. Addition          3. Addition

# . . . . . . . #   # . . . . . . . #   # . . . . . . . #
# 0 1 1 0 0 1 1 #   # . . . . . . . #   # . . . . . . . #
# 0 0 1 0 0 0 1 #   # 1 0 0 0 1 0 0 #   # . . . . . . . #
# 0 0 0 0 0 1 1 #   # 0 0 0 0 0 1 1 #   # 1 0 0 0 1 1 1 #
# . . . . . . . #   # . . . . . . . #   # . . . . . . . #
```

Es kann vorkommen, dass das Ergebnis durch Überläufe größer wird als der markierte Bereich. In diesem Fall hilft nur, den Bereich zwischen den Rauten zu vergrößern. Dies kann im Unterricht an konkreten Datentypen anderer Programmiersprachen aufgegriffen und problematisiert werden (was macht Java, wenn zwei Zahlen vom Typ int addiert werden sollen, deren Ergebnis aber größer als der darstellbare Bereich einer Ganzzahl ist?).

Weitere Beispiele, die sich für eine zweidimensioanle Turingmaschine anbieten:

- XOR-Maschine
- Multiplikationsmaschine
- Simulation boolscher Operatoren
- Alle Felder eines Labyrinthes finden und markieren

1.7 Turingmaschinen und Berechenbarkeit

Alan Turing beschäftigte sich bei der Entwickelte seiner Turingmaschine mit der Berechenbarkeit von Funktionen. *Alonso Church* fasste das Ergebnis seiner Arbeiten in folgender These zusammen:

> *Die Menge der Turing-berechenbaren Funktionen ist genau die Menge der im intuitiven Sinne berechenbaren Funktionen.*

Alle gängigen Programmiersprachen sind in diesem Sinne vollständig. Diese Aussage gilt streng genommen allerdings nur, wenn unbegrenzt Speicher zur Verfügung steht (entsprechend dem unendlich langen Turingband).

R. Baumann präzisierte den Begriff des Algorithmus mit Hilfe einer Turingmaschine[1]:

> *Ein Algorithmus ist ein Verfahren, das von einer Turingmaschine durchgeführt werden kann.*

Vorteile des Turingmaschinenmodells [2]:

- Jeder bis heute bekannte Formalismus kann damit simuliert werden.
- Die Berechnungsschritte sind einfach und können unmittelbar mechanisch umgesetzt werden.
- Die Beschreibung ist sehr einfach und anschaulich.

Nachteile des Turingmaschinenmodells:

- Unendlich langes Band ist nicht realisierbar.
- Programmierung ist kompliziert, da die Sprache sehr primitiv ist.
- Der Ablauf eines Programms wird aus der Struktur nicht ersichtlich.
- Die Laufzeit eines Turingprogramms ist sehr schlecht.

Turingmaschinen wurden tatsächlich nie gebaut; sie stellen ein rein theoretisches Rechnermodell dar, mit dem theoretische Betrachtungen wie die Berechenbarkeit von Funktionen einfacher untersucht werden können.

Registermaschinen stellen ein Bindeglied zwischen Turingmaschinen und realem Rechner dar. Diese erlauben eine deutlich komfortablere Programmierung, da sie direkten Zugriff auf alle Register haben. Ihre Handhabung ist aber noch wesentlich mühseliger als die realer Rechner.

[1] Rüdiger Baumann: Informatik mit Pascal, Klett-Verlag, Stuttgart 1981
[2] Rechenberg, Pomberger: Informatikhandbuch, Hanser-Verlag, München, Wien 2002

1.8 Aufgaben

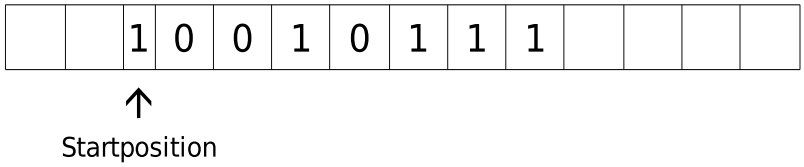
Bei den folgenden Aufgaben befindet sich der Lesekopf immer über der ersten besetzten Stelle links auf dem Band. Am Ende soll sich die Maschine wieder in diesem Zustand befinden.

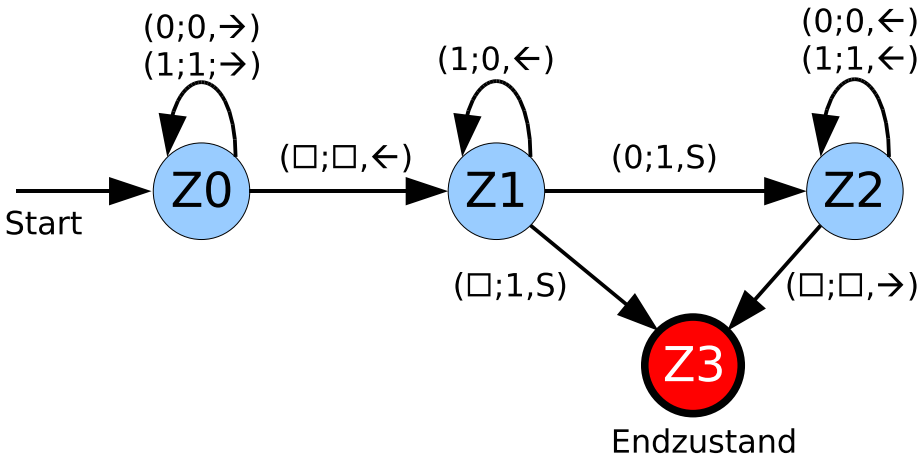
Übung 1: Entwerfen Sie eine Turingmaschine, die auf einem Band nach zwei aufeinander folgenden Nullen sucht (Leeres Feld = Suchen beenden). Der Lesekopf bleibt auf der ersten Null bzw. auf einem leeren Feld stehen.

Übung 2 Entwerfen Sie eine Turingmaschine, die Binärzahlen um 2 erhöht.

Übung 3 Eine Multipliziermaschine soll eine auf dem Band liegende Binärzahl mit 2 multiplizieren.

Aufgabe 1: Was macht folgende Maschine? Bilde diese Maschine mit Kara nach. Überprüfe mit unterschiedlichen Bandbelegungen.





Aufgabe 2: Gesucht ist eine Maschine, die solange nach rechts wandert, bis sie auf eine 1 oder ein leeres Feld trifft. Kommt sie zuerst zur 1, dann werden bis zur nächsten 1, höhstens aber bis zum nächsten Leerzeichen alle Felder gelöscht. Anschließend bleibt die Maschine auf der gefundenen zweiten 1 oder dem ersten Leerzeichen stehen.

Aufgabe 3: Die Maschine „Doppelnull" sucht auf einem Band nach vorhandenen doppelten Nullen (Suchrichtung: rechts). Die Maschine soll auf der ersten der beiden Nullen anhalten. Es darf angenommen werden, dass das Band keine Leerzeichen innerhalb des Suchtextes enthält.

Aufgabe 4: Die „Dupliziermaschine" dupliziert eine Bandbelegung, die ausschließlich aus aufeinanderfolgenden # besteht. Enthält das Eingabeband etwa 3 Rauten, so befinden sich nach dem Programmablauf 6 Rauten auf dem Band. Zu Beginn und am Ende befindet sich die Maschine auf der linken, ersten Raute.

Diese Aufgabe kann zu einer „Kopiermaschine" verallgemeinert werden. Diese enthält eine beliebige Bandbelegung, die von der Maschine ans Ende kopiert wird. Optional kann zwischen Original und Kopie ein leeres Feld eingefügt werden.

```
vorher:          nachher:
...1011#01#...   ...1011#01#.1011#01#...
```

Aufgabe 5: Ein „Palindromfinder" untersucht, ob eine Bandbelegung von rechts und von links gelesen übereinstimmt. Zu Beginn befindet sich die Maschine auf dem ersten, linken Zeichen. Wenn es sich um ein Palindrom handelt, dann ist anschließend das Band leer, sonst nicht.

Aufgabe 6: Entwirf einen „Dekrementierer", der eine positive Binärzahl um 1 vermindert. Anfangs- und Endzustand: msb (Most Significant Bit).

Aufgabe 7: Die „Addiermaschine" kann zwei Binärzahlen auf dem Eingabeband addieren. Die beiden Zahlen sind durch die Raute (#) getrennt. Am Ende soll auf dem Band nur noch die Summe der beiden Binärzahlen stehen und die Maschine am Anfang des Ergebnisses stehen bleiben.

Beispiel: 13 + 3 = 16

```
Turingband (Anfang)              Turingband (Ende)
---------------------------      ---------------------------
. . . 1 1 0 1 # 1 1 . . .        . . . 1 0 0 0 0 ...
---------------------------      ---------------------------
      | Start                          | Ende
```

Aufgabe 8: Die „Rautenmaschine" liest von einem Band eine Binärzahl ein und ersetzt diese Binärzahl durch entsprechend viele Rauten. Am Ende verharrt die Maschine auf der ersten, linken Raute.

Beispiel:

```
Turingband (Anfang)              Turingband (Ende)
---------------------------      ---------------------------
. . . 1 0 0 1 . . .              . # # # # # # # # # . . .
---------------------------      ---------------------------
      | Start                      | Ende
```

Aufgabe 9: Eine einfache „Sortiermaschine" ordnet vorhandene Ziffern auf dem Eingabeband so an, dass zuerst alle vorhandenen Nullen, dann alle Einsen auf dem Band liegen.

Beispiel:

```
Turingband (Anfang)              Turingband (Ende)
---------------------------      ---------------------------
. . . 1 0 0 1 0 . . .            . . . 0 0 0 1 1 . . .
---------------------------      ---------------------------
      | Start                          | Ende
```

Aufgabe 10: – etwas für Tüfftler –

Bei der Gen-Analyse wird in Gen-Code-Schnipseln nach dem Vorkommen einer bestimmten Folge von Genen gesucht. Diese genetische Suche soll durch eine Turingmaschine simuliert werden. Dazu wird das zu suchende Wort in einer Zeile aufgelistet. Der zu durchsuchende Genstring steht eine Zeile darunter. Wird das Wort gefunden, so zeigt die Turing- maschine auf den Anfang des gefundenen Wortes. Ist die Suche erfolglos, so bleibt die Turingmaschine am Ende des Genstrings stehen.

Erfolgreiche Suche:

A	T	T	C																	
T	C	A	T	G	T	T	A	T	G	A	T	T	C	C	G	C	A	T	C	
										↑										

Erfolglose Suche:

A	T	T	C																	
T	C	A	T	G	T	T	A	T	G	A	T	T	C	C	G	C	A	T	C	
																				↑

Da Kara die Buchstaben A,C,B,T nicht kennt, werden diese durch die vier Pfeile ersetzt. Vorsicht: Die pfeilsymbole dürfen nicht mit den Bewegungspfeilen verwechselt werden. Alle Symbole sind bei Kara durch einen hellen Hintergrund erkennbar.

Die Lösungen zu den Aufgaben können unter der Adresse `http://www.pns-berlin.de/fortbildungen/ibbb07/loesungen.zip` *heruntergeladen werden.*

1.9 Literatur und Verweise

Literatur

[1] Reichert, R. Nievergelt, J. Hartmann, W.: Programmieren mit Java. Springer Berlin, Heidelberg, New York 2005

Das Buch beschreibt die unterschiedlichen Einsatzmöglichkten des Kara-Modells im Schulunterricht. Dabei werden die unterschiedlichen Kara-Modelle exemplarisch vorgestellt.

[2] Modrow, E.: Theoretische Informatik mit Delphi. emu-online Scheden 2005.(`www.emu-online.de`)

Ein neues Schulbuch mit einer vollständigen Darstellung schulrelevanter Themen (Automaten, Turingmaschinen, Sprachen). Als Programmiersprache wird Delphi verwendet.

[3] Lehmann, E.: Die Turing-Maschine im Anfangsunterricht. LOGIN 1999, Heft 6, S.44-52

Ein Bericht von den ersten Stunden eines Informatikkurses in Klasse 11

Eine umfassendere Literaturliste finden Sie auf den Seiten des Berliner Bildungsservers.

Beschreibung von Turingmaschinen im Internet

`http://de.wikipedia.org/wiki/Turingmaschine`

Allgemeine Informationen zu Turingmaschinen.

`http://www.oberstufeninformatik.de/theorie`

Startseite von H. Gierhardt zur theoretischen Informatik. Zahlreiche Verweise zu weiterführenden Seiten und Software.

`http://www.pns-berlin.de/prof{}ilkurs/kara`

Unterrichtseinheit im Profilkurs Informatik mit Turingkara an der Paul-Natorp-Oberschule. Beispiele aus dem Leistungskurs mit JFLAP sind unter `http://www.pns-berlin.de/lk/ti/turing.html` zu finden.

`http://de.wikipedia.org/wiki/Fleißiger_Biber`

Das Busy-Beaver-Problem beschreibt eine Turingmaschinen, die zu einer vorgegebenen Zahl von Zuständen möglichst viele Einsen auf ein leeres Band schreibt und dabei zum Stehen kommt. (siehe auch `http://www.dbg.rt.bw.schule.de/lehrer/ritters/info/turing/biber.htm` und `http://www.fmi.uni-stuttgart.de/ti/projects/beaver/bbb.html`)

Software

- www.swisseduc.ch/informatik/karatojava/
- www.jflap.org
- http://math.hws.edu/TMCM/java/labs/xTuringMachineLab.html
- www.matheprisma.uni-wuppertal.de/Module/Turing/
- www.dbg.rt.bw.schule.de/lehrer/ritters/info/turing/vtur.htm
- wwwsys.informatik.fh-wiesbaden.de/weber1/turing/tm.html

Für Rückfragen, Anregungen und Kritik können Sie sich gerne an mich wenden. Die Materialien dieses Workshops finden Sie unter der angegebenen Internetadresse. Die Lösungen können über den (nicht sichtbaren) Link herunterladen.

Walter Gussmann, Paul-Natorp-Oberschule

Internet: http://www.pns-berlin.de/fortbildungen/ibbb07

Lösungen: http://www.pns-berlin.de/fortbildungen/ibbb07/loesungen.zip

Email: wagu1@web.de