VORLESUNGSMODUL THEORETISCHE INFORMATIK - VORLMOD THINF -

MATTHIAS ANSORG

ZUSAMMENFASSUNG. Studentische Mitschrift der ersten Veranstaltung zur Vorlesung Theoretische Informatik bei Prof. Dr. Lutz Eichner (Wintersemetser 2004/2005) im Studiengang Informatik an der Fachhochschule Gießen-Friedberg. Diese Vorlesung führt bis zum Beweis des Gödelschen Unvollständigkeitssatzes.

- **Bezugsquelle:** Die vorliegende studentische Mitschrift steht im Internet zum Download bereit. Quelle: Persönliche Homepage Matthias Ansorg :: InformatikDiplom http://matthias.ansorgs.de/InformatikAusbldg/.
- **Lizenz:** Diese studentische Mitschrift ist public domain, darf also ohne Einschränkungen oder Quellenangabe für jeden beliebigen Zweck benutzt werden, kommerziell und nichtkommerziell; jedoch enthält sie keinerlei Garantien für Richtigkeit oder Eignung oder sonst irgendetwas, weder explizit noch implizit. Das Risiko der Nutzung dieser studentischen Mitschrift liegt allein beim Nutzer selbst. Einschränkend sind außerdem die Urheberrechte der angegebenen Quellen zu beachten.
- **Korrekturen und Feedback:** Fehler zur Verbesserung in zukünftigen Versionen, sonstige Verbesserungsvorschläge und Wünsche bitte dem Autor per e-mail mitteilen: Matthias Ansorg <mailto:matthias@ansorgs.de>.
- **Format:** Die vorliegende studentische Mitschrift wurde mit dem Programm LyX (graphisches Frontend zu LaTeX) unter Linux geschrieben und mit pdfLaTeXals pdf-Datei erstellt. Grafiken wurden mit dem Programm xfig unter Linux erstellt und als pdf-Dateien exportiert.
- **Dozent:** Prof. Dr. Lutz Eichner.
- **Verwendete Quellen:** <quelle> {<quelle>}.
- **Klausur:**

INHALTSVERZEICHNIS

1. ORGANISATION

- Zur Veranstaltung »Theoretische Informatik« gibt es keine Voraussetzungen, auch nicht die Veranstaltung »Automaten und Formale Sprachen« nicht, trotz dass dies im Modulkatalog so angegeben ist.

Date: 11. Oktober 2004 bis 28. März 2005.

- Übung und Vorlesung sind eine kombinierte Veranstaltung, es gibt manchmal auch Hausübungen. Diese ergeben jedoch keine Bonuspunkte für die Klausur und sind auch keine Teilnahmevoraussetzung für die Klausur.
- Es gibt keine formalen Teilnahmevoraussetzungen für die Klausur.

2. Einleitung

2.1. **Wiederholungen.**

Definition 1. Algorithmus

Definition. Ein Algorithmus ist eine Vorschrift, die

- zur Behandlung von Problemen einer Klasse gleichartiger Probleme (Allgemeinheit)
- in endlichem Text abgelegt ist (Endlichkeit; endliche Laufzeit dagegen ist nicht gefordert)
- den nächsten Schritt eindeutig vorherbestimmt (Determiniertheit; es muss eine Gebrauchsanweisung sein, bei der nie Zweifel über den nächsten Schritt bestehen)

Turing überlegte sich eine Maschine, die jeden Algorithmus ausführen kann: die Turing-Maschine. Church dagegen überlegte den Lambda-Kalkül (was später zu Lisp führte). Später konnte bewiesen werden, dass das Lambda-Kalkül nicht mächtiger als die Turing-Maschine ist. Mit der Church'schen These wird heute (nicht beweisbar, aber allgemein akzeptiert) unterstellt, dass die Turing-Maschine äquivalent zu allen anderen Darstellungsformen des Algorithmus ist:

Theorem 1. *These von Church*

Theorem. *Eine Vorschrift V ist ein Algorithmus $\Leftrightarrow$ (»genau dann, wenn«) V ist durch eine Turing-Maschine realisierbar.*

Aufgrund der These von Church steht die Turing-Maschine im Mittelpunkt der Betrachtungen der Theoretischen Informatik - alle Probleme der Berechenbarkeit können mit der Turing-Maschine dargestellt werden. Ebenso können Algorithmen durch Programme dargestellt werden. Programme (im Modell: alle Algorithmen) arbeiten auf Zeichenketten (8bit-Muster ohne Vorgabe einer Interpretation) als kleinsten Einheiten. Da es nur um Darstellung geht (physikalische Größen, mit denen Zeichenketten dargestellt werden) setzt man hier nicht »Information« oder »Daten«, weil dadurch eine Semantik mt eingeschlossen wäre. Ein Programm interessiert sich nicht für die Bedeutung einer Zeichenkette, es verarbeitet sie nur entsprechend von Anweisungen.

Probleme einer Problemklasse werden als Zeichenketten dargestellt (Wörter). Die Bedeutung der Zeichenketten, d.i. die Probleme für die sie stehen, interessiert dann während der Bearbeitung nicht mehr. Die Menge der Zeichenketten ist die Sprache.

Definition 2. Alphabet

Definition.

- Ein Alphabet Σ ist eine endliche Menge.
- $x \in \Sigma$ heißt Zeichen (auch: Symbol).
- $x_1x_2 \dots x_n$ mit $x_i \in \Sigma$ heißt Wort über Σ.
- ε heißt Leerwort und enthält kein Zeichen (z.B. die leere Menge in der Mengenlehre)
- Σ^* ist die Menge aller Wörter über Σ, einschließlich ϵ
- $\Sigma^+ := \Sigma^* - \{\varepsilon\}$
- Sei $w := x_1x_2x_3 \dots x_n$, $x_i \in \Sigma$; $|w| := n$ heißt Länge von w. Es gilt: $|\varepsilon| = 0$.
- $A \subseteq \Sigma^*$ heißt Sprache über Σ

Die Turing-Maschine wird oft unterschiedlich dargestellt, um bestimmte Probleme besser darstellen zu können. Die Steuereinheit der Turing-Maschine hat nur endlich viele Zustände, das sind endlich viele Belegungsmöglichkeiten von endlich vielen Speicherzellen.

2.2. **Turing-Maschine.** Siehe Abbildung **??**.

Definition 3. Turing-Maschine

Definition. Die Turing-Maschine besteht aus:

Turing-Band:

- Ein beidseitig unendliches Band. Auch eine Turing-Maschine mit einseitig unendlichem Band kann dasselbe, die zweiseitig unendliche ist aber komfortabler.
- Das Turing-Band ist in Zellen eingeteilt. Die Menge der Zellen ist abzählbar unendlich, also nicht überabzählbar! Die Menge der Zellen ist also mit natürlichen Zahlen nummerierbar.
- Nur endlich viele Zellen sind beschriftet, d.h. sie enthalten ein sog. Eingabezeichen (oder Hilfszeichen[1]). Alle übrigen Zellen sind leer, man sagt: sie enthalten das Leerzeichen $\square$, genannt »Blank«. Denn: in endlicher Zeit nach dem Einschalten der Turing-Maschine kann man nur endlich viele Zellen beschriften.
- Die Gesamtheit aller Zeichen (Eingabezeichen, Hilfszeichen und $\square$) heißt Bandalphabet $\Gamma = \{\mathrm{a}_0, a_1, a_2, \ldots, a_n\}$ wobei $a_0 = \square$.

Steuereinheit:

- Sie nimmt Zustände $q \in Q = \{q_0, q_1, \ldots, q_n\}$ an; q_0 heißt Startzustand.
- Eine Zeichenkette, die verarbeitet werden soll, wird vor dem Start der Turing-Maschine auf das Band gebracht. In der Turing-Maschine wird dann q_0 eingestellt und der Lese-/Schreibkopf wird auf das erste Zeichen der Zeichenkette gesetzt. Das nennt man »die Turing-Maschine auf eine Inschrift ansetzen«.

Lese-/Schreibkopf:

- kann einen Zelleninhalt ändern
- kann eine Zelle nach links oder rechts gehen oder die Position beibehalten

Zu einem bestimmten Zeitpunkt befindet sich die Turing-Maschine in einem bestimmten Zustand und unter dem Lese-/Schreibkopf befindet sich ein Zeichen. Mit jedem neuen Takt[2] ändert die Turing-Maschine das aktuelle Zeichen und ihren aktuellen Zustand, nur abhängig vom aktuellen Zeichen und dem aktuellen Zustand. Zusätzlich kann der Lese-/Schreibkopf bewegt werden. Diese eindeutige Vorherbestimmung ist der Determinismus der Turing-Maschine.

Definition 4. Arbeitsweise der Turing-Maschine

Definition.

Start:

- Auf dem Turing-Band befinden sich endlich viele Zeichen $\neq \square$.
- Die Steuereinheit ist im Zustand q_0.
- Der Lese-/Schreibkopf wird auf (irgend-)ein Feld gesetzt. Üblicherweise aber auf das erste (am meisten linke) Zeichen eines Eingabe-Wortes gesetzt.

Arbeitsschritt: Abhängig vom Zustand q und dem Zeichen a (aktuelles Zeichen unter dem Lese-/Schreibkopf) führt die Turing-Maschine folgende Aktionen aus:

- Die Steuereinheit geht in eien Zustand q' über. Es kann $q = q'$ sein.
- Der Lese-/Schreibkopf ersetzt a durch a'. Es kann $a = a'$ sein.
- Der Lese-/Schreibkopf führt genau eine Bewegung $y \in \{L, R, N\}$ aus: Verschiebung um eine Zelle nach links, nach rechts, oder keine Positionsänderung.
- Der Arbeitsschritt kann als Funktion geschrieben werden: $\delta(q, a) = (q', a', y)$, $y \in \{L, R, N\}$.

[1] Man kann alle Beweise auch ohne Hilfszeichen führen.

[2] Man darf sich durchaus vorstellen, die Turing-Maschine sei getaktet.

- Jede spezielle Turing-Maschine ist ein Programm: jede Problemklasse wird durch eine unterschiedliche Turing-Maschine (d.i. einen anderen Algorithmus) dargestellt.

Definition 5. Deterministische Turing-Maschine (abstrakte Beschreibung)

Definition. Eine deterministische Turing-Maschine (DTM) ist ein Gebilde[3] $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ mit:

Q**:** Zustandsmenge
Σ**:** Eingabealphabet (enthält nur die Zeichen, die zur Beschreibung der Probleme einer Problemklasse notwendig sind)
Γ**:** Bandalphabet (auch: Arbeitsalphabet); $\Sigma \subset \Gamma$
$q_0 \in Q$**:** Startzustand (der Name ist egal, es muss lediglich einen ausgezeichneten Startzustand geben)
$F \subseteq Q$**:** Menge der Endzustände. Eigentlich unnötig. Sie dienen dazu, algorithmisch entschiedene Eigenschaften von Eingabewörtern mtzuteilen. Hat man lediglich Rechenergebnisse (Zahlen), braucht man keine Endzustände. Aber auch sonst braucht man keine Endzustände: Eigenschaften (z.B.: ob eine Zahl durch 4 teilbar ist) können auch mitgeteilt werden durch das Zeichen, über dem der Lesekopf schließlich hält.
$\square \in \Gamma - \Sigma$**:** Das sog. Leerzeichen (»Blank«), zu unterscheiden vom leeren Zeichen.
$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$**:** Überangsfunktion. Diese Abbildung beschreibt das Verhalten der Maschine. Es ist eine partielle Funktion: sie liefert entweder ein eindeutiges oder gar kein Ergebnis, d.h. manche Urbilder haben keine Bilder. Diese Definitionslücken sind notwendig, um das Halten der Maschine beschreiben zu können.

Das Turing-Band enthält zu jedem Zeitpunkt nur endlich viele Zeichen $\neq \square$. Computer entsprechen der universellen Turing-Maschine: eine Maschine, die alle anderen (speziellen) Turing-Maschinen und auch sich selbst interpretieren kann.

Definition 6. Bandinschrift

Definition. Sei B der kleinste zusammenhängende Teilbereich des Turingbandes, der

- alle Zeichen $\neq \square$ enthält
- alle früher besuchten Zellen (auch leere)
- die aktuelle Zelle (d.h. die Zelle unter dem Lese-/Schreibkopf)

enthält. B heißt Bandinschrift.

Example 1. Bandinschrift

Example.

$\ldots$	$\square$	$\square$	a	c	$\square$	b	b	$\square$	$\square$	$\square$	$\ldots$
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}

$x_0, x_1, x_8, x_9, x_{10}, x_{11}$**:** Hier war der Lese-/Schreibkopf noch nie.
x_2**:** Hier war der Lese-/Schreibkopf früher.
x_5**:** Hier befindet sich der Lese-/Schreibkopf jetzt.

Die Bandinschrift B, die Position des Lese-/Schreibkopfes und der Zustand der Steuereinheit zu einem bestimmten Zeitpunkt heißt Konfiguration der Turing-Maschine.

[3] 7-Tupel

2.3. **Darstellung von Konfigurationen.** Es gibt mehrere Möglichkeiten:

-
- kürzer:

$$\begin{array}{cccccccc} b_1 & b_2 & \dots & b_i & b_{i+1} & \dots & b_k \\ & & & & q & & \end{array}$$

- im Fließtext:

$$b_1 b_2 \dots b_i q b_{i+1} \dots b_k$$

Definition 7. Konfiguration

Definition. Sei Γ das Bandalphabet einer Turing-Maschine M. $\alpha q \beta$ heißt Konfiguration von M, wenn gilt:

- $\alpha\beta$ ist die Bandinschrift ($\alpha \in \Gamma^*$, $\beta \in \Gamma^+$)
- q ist der Zustand von M
- der Lese-/Schreibkopf besondert sich über dem 1. Zeichen von β

Die Arbeitsweise einer Turing-Maschine ist nun als Folge von Konfigurationen beschreibbar.

Definition 8. Direkte Folgekonfiguration

Definition. Sei $\alpha q \beta$ eine Konfiguration von M. Sei a das 1. Zeichen von β. Fälle:

(1) Sei $\delta(q, a) = (q', a', Y)$ mit $Y \in \{R, L, N\}$. Aus $\alpha q \beta$ entsteht dann eine Konfiguration $\alpha' q' \beta'$. $\alpha' q' \beta'$ heißt direkte Folgekonfiguration von $\alpha q \beta$, in Zeichen:

$$\alpha q \beta \vdash \alpha' q' \beta'$$

(Lies: »geht direkt über in«).

(2) $\delta(q, a) = \perp$. Das heißt: $\delta(q, a)$ ist undefiniert. In Zeichen:

$$\alpha q \beta \nvdash$$

Lies: »$\alpha q \beta$ besitzt keine Folgekonfiguration« oder kürzer »Mstoppt in $\alpha q \beta$«.

Darstellung einer Turing-Maschine Mdurch die Turing-Tabelle (auch: durch ein Programm): durch die Angabe von Σ, F und eine Tabelle der Form:

$\Gamma \rightarrow$	a_0	a_1	a_2	$\dots$	a_n
q_0					
q_1					
$\vdots$					
q_n					

Vereinbarung, sofern nicht anders angegeben: $a_0 = \square$, q_0 ist Startzustand. An Stelle von q_0 schreibt man auch, wenn mehrere Maschinen unterschieden werden, für die Maschine M: q_M. Anders als in vielen Büchern werden hier (außer $\square$) keine Sonderzeichen verwendet. Sie wurden nur zur bequemeren Darstellung und Verarbeitung eingeführt. Vorläufig sei also: $\Sigma = \{a_1, \dots, a_n\}$, $\Gamma = \Sigma \cup \{\square\}$.

Example 2. Eine Turing-Maschine M_1

Example. M_1:

	$\square$	0	1
q_0	$-$	$q_0, \square, R$	$q_0, \square, R$

$\Sigma = \{0, 1\}$, $Q = \{q_0\}$, $F = \{q_0\}$

Konfigurationen und Folgekonfigurationen, also die Arbeitsweise: M_1, auf das erste Zeichen von w angesetzt, löscht w:

$$\begin{array}{ccccccccccccccccc} 0 & 1 & 0 & \vdash & \square & 1 & 0 & \vdash & \square & \square & 0 & \vdash & \square & \square & \square & \square & \nvdash \\ q_0 & & & & & q_0 & & & & & q_0 & & & & & q_0 & \end{array}$$

Example 3. Eine Turing-Maschine M_2

Example. M_2:

	$\square$	0	1
q_0	$-$	$q_1 0 R$	$q_0 1 R$
q_1	$-$	$q_0 0 N$	$q_0 1 L$

$F = \{q_1\}$

Startkonfiguration sei $q_0 w$ mit $w \in \Sigma^+$. Konfigurationen und Folgekonfigurationen:

(1) Für $w = 00$:

$$\underset{q_0}{0}\ \underset{}{0}\ \vdash\ 0\ \underset{q_1}{0}\ \vdash\ \underset{q_0}{0}\ 0\ \vdash\ 0\ \underset{q_1}{0}\ \square\ \nvdash$$

Man sagt: M_2 stoppt akzeptierend, d.h. in einem Endzustand ($q_1 \in F$).

(2) Für $w = 11$:

$$\underset{q_0}{1}\ \underset{}{1}\ \vdash\ 1\ \underset{q_0}{1}\ \vdash\ \underset{q_0}{1}\ 1\ \square\ \nvdash$$

Man sagt: M_2 stoppt verwerfend, d.h. in einem Zustand, der nicht Endzustand ist ($\notin F$).

(3) Für $w = 01$:

$$\underset{q_0}{0}\ \underset{}{1}\ \vdash\ 0\ \underset{q_1}{1}\ \vdash\ \underset{q_0}{0}\ 1\ \vdash\ \dots$$

Man sagt: M_2 stoppt nie.

Definition 9. Übergang in

Definition. Für $\alpha_1 \vdash \alpha_2 \vdash \dots \vdash \alpha_n$ schreibt man $\alpha_1 \overset{*}{\vdash} \alpha_n$ (lies: »geht über in«). $\alpha_1 \vdash \alpha_2$ dagegen liest man: »geht direkt über in«.

Definition 10. Turing-Maschinen als Berechnungsverfahren

Definition. Sei Σ ein Alphabet, $f : \Sigma^* \to \Sigma^*$ eine partielle Funktion. f heißt Turing-berechenbar[4] :⇔ es esistiert eine deterministische Turing-Maschine $M\,(Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ mit: Für $w, v \in \Sigma^*$ gilt: $f(w) = v \Leftrightarrow q_0 w \overset{*}{\underset{M}{\vdash}} \square \dots \square q v \square\square \dots \square$ mit $q \in F$ und $\square \dots \square q v \square\square \dots \square \nvdash$ (s.h. die Maschine stoppt).

Literatur

[1] Schöningh: »Theoretische Informatik kurzgefasst«. Das Buch, nach dem sich diese Vorlesung richtet; enthält allen Stoff dieser Vorlesung, jedoch mit zu kurzen Beweisen, die Herr Eichner so erweitert hat dass man sie verstehen kann.

[2] Sipsev: »Computation«. Das Buch, das in der Veranstaltung »Automaten und Formale Sprachen« bei Prof. Metz verwendet wurde. Einige Beispiele in »Theoretische Informatik« stammen aus diesem Buch, dort immer zum Bereich »Automaten und Formale Sprachen«. Die sonstigen Beispiele in »Theoretische Informatik« stammen aus dem Bereich »Berechenbarkeit«, so dass sich die Veranstaltungen nicht überschneiden. Die Definitionen in dieser Vorlesung richten sich nach der Formulierung in diesem Buch.

[3] Hermes. Dieses Buch ist für Mathematiker, inkl. der »wahren« Variante der Ackermann-Funktion. Herr Eichner hat Inhalte hieraus für Informatiker übersetzt.

[4] Unter Voraussetzung der Church'schen These einfach »berechenbar«.