Das Rechenmodell namens „Turing-Maschine“

Hans U. Simon (RUB)

Email: simon@lmi.rub.de

Homepage: http://www.ruhr-uni-bochum.de/lmi

Die Turingmaschine

- DTM = Deterministische Turingmaschine
- NTM = Nichtdeterministische Turingmaschine
- TM = DTM oder NTM

Intuitiv gilt:

- DTM = (DFA + dynamischer Speicher)
- NTM = (NFA + dynamischer Speicher)
- Der dynamische Speicher ist ein (in Zellen unterteiltes) zweiseitig unendliches Band versehen mit einem Lese–Schreibkopf. Es enthält anfangs die Eingabe, dient aber auch als Arbeitsspeicher.
- Die Einschränkung, den Kopf auf dem Band nur von links nach rechts bewegen zu dürfen, wird fallen gelassen.
- Die Einschränkung, den Arbeitsspeicher kellerartig organisieren zu müssen, wird ebenfalls fallen gelassen.

DTM (formale Definition)

Eine DTM M besteht aus den folgenden Komponenten:

- Z, die Zustandsmenge (eine endliche Menge)
- Σ, das Eingabealphabet (ebenfalls endlich)
- $\Gamma \supset \Sigma$, das Arbeitsalphabet (ebenfalls endlich)
- $\delta : Z \times \Gamma \to Z \times \Gamma \times \{L, R, N\}$, die partiell definierte Überführungsfunktion
- $z_0 \in Z$, der Startzustand
- $\Box \in \Gamma \setminus \Sigma$, das Blank (auch Leerzeichen genannt)
- $E \subseteq Z$ die Menge der Endzustände:
  $\delta(z_e, A)$ ist undefiniert für alle $z_e \in E$ und alle $A \in \Gamma$.

Arbeitsweise der DTM

- Anfangs befindet sich M im Startzustand z_0, ihr Band enthält das Eingabewort $w \in \Sigma^*$ (umrahmt von Blanks) und der Kopf steht auf dem ersten Zeichen von w (bzw. auf einem Blank, falls $w = \varepsilon$).
- Falls sich M im Zustand $z \in Z$ befindet, der Kopf das Zeichen $A \in \Gamma$ liest und

$$\delta(z, A) = (z', A', d) \in Z \times \Gamma \times \{L, R, N\} \ ,$$

  dann geht M in den Zustand z' über und ersetzt A durch A'. Für $d = L$ bzw. $d = R$ erfolgt zusätzlich eine Kopfbewegung auf die linke bzw. rechte Nachbarzelle des Bandes.
- M stoppt **gdw** M in einem Zustand z ist, ein Symbol A liest und $\delta(z, A)$ undefiniert ist.
- Die Eingabe wird akzeptiert **gdw** M im Laufe der Rechnung in einen Endzustand gerät (und dann automatisch stoppt).

NTMs

Eine NTM M ist analog definiert.

Unterschied: Die Überführungsfunktion δ einer NTM hat die Form

$$\delta : Z \times \Gamma \to \mathcal{P}(Z \times \Gamma \times \{L, R, N\}) \ ,$$

wobei (in Analogie zu DTMs)

$$\forall z_e \in E, A \in \Gamma : \delta(z_e, A) = \emptyset \ .$$

Arbeitsweise von NTMs

Wie bei nicht-deterministischen Maschinen üblich hat die NTM die „Qual der Wahl“ (i.A. mehrere mögliche nächste Rechenschritte):

- Anfangs befindet sich M im Startzustand z_0, ihr Band enthält das Eingabewort $w \in \Sigma^*$ (umrahmt von Blanks) und der Kopf steht auf dem ersten Zeichen von w (bzw. auf einem Blank, falls $w = \varepsilon$).
- Falls sich M im Zustand $z \in Z$ befindet, der Kopf das Zeichen $A \in \Gamma$ liest und

  $$(z', A', d) \in \delta(z, A) \subseteq Z \times \Gamma \times \{L, R, N\} \ ,$$

  dann darf M in den Zustand z' übergehen, A durch A' ersetzen und die $d \in \{L, R, N\}$ entsprechende Kopfbewegung ausführen.
- Die Eingabe wird akzeptiert **gdw** M durch geeignete Wahl der möglichen Rechenschritte in einen Endzustand geraten kann.

Konfigurationen einer TM

Die Konfiguration einer TM besteht aus

- dem aktuellen Zustand $z \in Z$,
- der Position des Kopfes auf dem Band.
- dem Bandinhalt $\gamma \in \Gamma^*$ (im Bereich der Eingabe sowie der im Laufe der Rechnung bereits besuchten Zellen)

Notation: $\alpha z \beta$, wobei $\gamma = \alpha\beta$ der aktuelle Bandinhalt und der Kopf auf dem ersten Zeichen von β positioniert ist

Anfangskonfiguration bei Eingabe w: $z_0 w$ (hier: $\alpha = \varepsilon, \beta = w$)

Akzeptierende Endkonfiguration: $\alpha z \beta$ für jedes $z \in E, \alpha, \beta \in \Gamma^*$

Stoppkonfiguration: $\alpha z A \beta'$ für jedes $z \in Z, \alpha, \beta' \in \Gamma^*$, $A \in \Gamma$ mit $\delta(z, A)$ ist undefiniert.

Beobachtung: Da δ auf Endzuständen undefiniert ist, ist jede akzeptierende Endkonfiguration auch eine Stoppkonfiguration.

Folgekonfigurationen

Eine „Rechnung“ einer TM lässt sich als Folge von Konfigurationen beschreiben.

Definition:

1. $\alpha' z' \beta'$ heißt unmittelbare Folgekonfiguration von $\alpha z \beta$ **gdw** $\alpha' z' \beta'$ aus $\alpha z \beta$ durch einen „Rechenschritt“ (einmalige Verwendung der Überführungsfunktion) resultieren kann.
   Notation: $\alpha z \beta \vdash \alpha' z' \beta'$.

2. $\alpha' z' \beta'$ heißt Folgekonfiguration von $\alpha z \beta$ **gdw** $\alpha' z' \beta'$ aus $\alpha z \beta$ durch eine (evtl. leere) Folge von Rechenschritten resultieren kann.
   Notation: $\alpha z \beta \vdash^* \alpha' z' \beta'$.
   Formal ist „$\vdash^*$” die reflexive–transitive Hülle von „$\vdash$“.

Im Falle einer DTM ist die unmittelbare Folgekonfiguration stets eindeutig bestimmt und es gibt nur eine mögliche Rechnung auf der Eingabe.

Beispiel

bin(n) bezeichne die Binärdarstellung einer Zahl $n \geq 0$.

Aufgabe: Implementiere einen Binärzähler, der, gestartet auf bin(n),

- bin($n+1$) berechnet,
- den Kopf auf dem ersten Zeichen von bin($n+1$) positioniert
- und sich dann in einen Endzustand begibt und stoppt.

Idee: Verwende vier Zustände für folgende Phasen der Berechnung:

- z_0: Suche das Bit am weitesten rechts.
- z_1: Inkrementiere den Zähler (unter Beachtung des Übertrages).
- z_2: Suche das Bit am weitesten links.
- z_e: Stoppe.

Beispiel (fortgesetzt)

Komponenten der „Binärzähler“–DTM:

- Zustandsmenge $\{z_0, z_1, z_2, z_e\}$
- Eingabealphabet $\{0, 1\}$
- Arbeitsalphabet $\{0, 1, \square\}$
- Überführungsfunktion δ (weiter unten spezifiziert)
- Startzustand z_0
- Blank $\square$
- Menge $\{z_e\}$ der Endzustände

Beispiel (fortgesetzt)

„Turing–Tafel“ von M (tabellarische Angabe von δ):

δ	0	1	$\square$
z_0	$(z_0, 0, R)$	$(z_0, 1, R)$	$(z_1, \square, L)$
z_1	$(z_2, 1, L)$	$(z_1, 0, L)$	$(z_e, 1, N)$
z_2	$(z_2, 0, L)$	$(z_2, 1, L)$	$(z_e, \square, R)$

`--- Macht das Sinn ?? Erläutere !! ---`

Sprache einer TM

Die folgende Definition der von der TM M erkannten Sprache $T(M)$ entspricht unserer Vereinbarung über das Akzeptieren mit Endzustand:

$$T(M) := \{w \in \Sigma^* | \ \exists z \in E, \alpha, \beta \in \Gamma^* : z_0 w \vdash^* \alpha z \beta\}$$

In Worten: Wort w gehört zur Sprache $T(M)$ **gdw** M durch Verarbeitung der Eingabe w aus der Anfangskonfiguration in eine akzeptierende Endkonfiguration gelangen kann.

Haltebereich einer DTM

Wir definieren den Haltebereich einer DTM M wie folgt:

$$H(M) := \{w \in \Sigma^* \mid \exists z \in Z, \alpha, \beta' \in \Gamma^*, A \in \Gamma : z_0 w \vdash^* \alpha z A \beta',\ \delta(z, A) \text{ ist undefiniert}\}$$

In Worten: Wort w gehört zum Haltebereich $H(M)$ **gdw** M durch Verarbeitung der Eingabe w aus der Anfangskonfiguration in eine Stoppkonfiguration gelangt.

Beobachtung: Da jede Endkonfiguration auch eine Stoppkonfiguration ist, gilt $L(M) \subseteq H(M)$.

DTMs versus realistischere Rechnermodelle

Die „Programmiersprache“ für eine Turingmaschine ist

- leicht zu erlernen,
- aber wenig problemorientiert und daher mühselig zu handhaben.

Wir werden an einem späteren Punkt der Vorlesung folgendes aufzeigen:

1. DTMs sind „universelle“ Rechnermodelle: alles was in einem intuitiven Sinne berechenbar ist, ist auch durch eine DTM berechenbar.
2. DTMs können ohne wesentlichen Effizienzverlust realistischere Modelle moderner Rechner simulieren.

Da die Angabe von Turing–Tafeln sehr mühselig ist, werden wir im folgenden die Strategie einer TM zur Lösung eines Problems mehr informell beschreiben (eine Vorgehensweise, die später durch den Nachweis der Universalität der DTM gerechtfertigt wird).

NTMs und ihr Konfigurationsdigraph

Zu einer NTM M betrachten wir den folgenden sogenannten Konfigurationsdigraphen G_M:

- Die Knoten sind die Konfigurationen von M.
- Wir ziehen eine Kante von Konfiguration K zu Konfiguration K', falls $K \vdash_M K'$.

Die Knotenmenge (= Menge aller Konfigurationen) ist unendlich!
Allerdings spielen bei einer konkreten Eingabe w nur die von Startkonfiguration $K_0(w) = z_0 w$ erreichbaren Konfigurationen eine Rolle.

Nichtdeterminismus und „Raten eines Beweises“

- Jede NTM M kann (ohne Abänderung der von ihr erkannten Sprache) so modifiziert werden, dass sie in jedem Schritt genau zwei Wahlmöglichkeiten hat.
- Im Konfigurationsdigraph G_M hat dann jeder Knoten zwei ausgehende Kanten, sagen wir die „0-Ausgangskante“ die „1-Ausgangskante“.
- Wir können uns anschaulich vorstellen, dass eine NTM in jedem Schritt ein Bit, 0 oder 1, rät und dann den aktuellen Berechnungspfad entsprechend fortsetzt.
- Jeder Berechnungspfad der Länge t ist daher eindeutig durch einen Binärstring der Länge t (die Folge der geratenen Bits) beschreibbar.
- Eine akzeptierende Rechnung auf Eingabe w entspricht dann einem Pfad von Anfangskonfiguration $K_0(w) = z_0w$ zu einer Endkonfiguration. Den zugehörigen Binärstring (Folge der geratenen Bits) können wir als einen Beweis für $w \in T(M)$ ansehen.

Raten unterwegs oder ganz am Anfang — egal

Raten unterwegs: Die NTM nach unserer bisherigen Definition rät in jeder Konfiguration ein Bit, 0 oder 1, und setzt ihren Berechnungspfad dann entsprechend fort.

Raten ganz am Anfang: Jede NTM M, die pro Schritt zwei Wahlmöglichkeiten hat, kann modifiziert werden (ohne dabei die Sprache $N(M)$ abzuändern) wie folgt:
Sie rät am Anfang der Rechnung einen Binärstring $u \in \{0,1\}^*$, den sie auf die Zellen $-1, \ldots, -|u|$ schreibt. Danach rechnet sie $|u|$ Schritte lang deterministisch, indem sie u auf die offensichtliche Weise als Wegbeschreibung durch den Konfigurationsdigraphen interpretiert. Genau dann, wenn sie dabei zu einer Endkonfiguration gelangt, akzeptiert sie ihre Eingabe.

Die NTM mit „Raten ganz am Anfang“ arbeitet nach dem sogenannten „Rate-Verifikationsprinzip“: rate einen „Beweis“ u für $w \in T(M)$ und verifiziere mit Hilfe von u anschließend deterministisch, dass die Eingabe w akzeptabel ist.

Determinismus versus Nondeterminismus

Jede DTM kann als Spezialfall einer NTM aufgefasst werden. Es gilt aber auch umgekehrt der

Satz: Jede NTM M kann von einer DTM M' simuliert werden.

Determinismus versus Nondeterminismus (fortgesetzt)

Beweis: Wir dürfen annehmen, dass M gemäß dem Rate-Verifikationsprinzip vorgeht. M' arbeitet wie folgt:

- M' probiert alle Strings $u = \varepsilon, 0, 1, 00, 01, 10, 11, 000 \ldots$ der Reihe nach aus.
- Für jeden festen String u rechnet M' so wie M in ihrer deterministischen Verifikationsphase mit „Beweis“ u. Falls dabei eine Endkonfiguration erreicht wird, akzeptiert M' und stoppt die Simulation.

Es ist offensichtlich, dass folgendes gilt:

- Wenn M eine akzeptierende Rechnung auf Eingabe w besitzt, dann wird w auch von M' nach endlich vielen Schritten akzeptiert.
- Auf Eingaben $w \notin N(M)$ rechnet M' endlos (ohne zu akzeptieren).

Ein Beispiel

Die Sprache der Nicht-Primzahlen kann nach dem Rate-Verifikationsprinzip von einer NTM M erkannt werden wie folgt:

- Rate zwei ganze Zahlen $a, b \geq 2$ (bzw. die Bits ihrer Binärdarstellung).
- Verifiziere deterministisch, durch Multiplizieren von a und b, dass die Eingabezahl n sich gemäß $n = ab$ zerlegen lässt (und somit keine Primzahl ist).
- Das Zahlenpaar (a, b) ist in diesem Beispiel der Beweis dafür, dass die Eingabezahl n keine Primzahl ist. (Natürlich besitzen nur Nicht-Primzahlen einen solchen Beweis.)
- Die deterministische Simulation würde alle denkbaren Beweise (hier: Zahlenpaare (a, b)) der Reihe nach durchprobieren.

Mehrspurenmaschinen

Zu einem gegebenen Alphabet Γ können wir „Supersymbole“ aus Γ^k betrachten. Wenn das Arbeitsalphabet einer TM ein Supersymbol $(A_1, \ldots, A_k)$ enthält, dann ist es anschaulich sich vorzustellen, dass

- das Band in k „Spuren“ zerlegt werden kann,
- und beim Abspeichern von $(A_1, \ldots, A_k)$ in einer Zelle, das Symbol A_i in der i–ten Spur der Zelle steht.

Beachte: Mehrspurenmaschinen haben zwar ein unkonventionelles Arbeitsalphabet (welches k-Tupel enthält), entsprechen aber unserer Standarddefinition einer TM (**kein** neues Modell).

Mehrbandmaschinen

Definition: Unter einer k-Band TM verstehen wir eine TM mit k Bändern und einem Kopf pro Band. Die insgesamt k Köpfe können sich in einem Rechenschritt in verschiedene Richtungen bewegen. Die Überführungsfunktion δ hat nun die Form

$$\delta : Z \times \Gamma^k \to Z \times \Gamma^k \times \{R, L, N\}^k$$

mit der offensichtlichen Interpretation.

Mehrbandmaschinen sind nicht mächtiger als das Standardmodell wie der folgende sogenannte Bandreduktionssatz zeigt:

Satz: Eine k-Band TM M kann von einer 1-Band TM M' simuliert werden. Ist dabei M eine DTM, so auch M'.

Beweis

- M' besitzt für jeden Zustand z von M einen entsprechenden Zustand z' (und weitere Zustände).
- M' simuliert
  - einen Schritt von M mit Zustandswechsel von z_1 nach z_2
  - durch eine Folge von Schritten, welche im Zustand z'_1 startet und im Zustand z'_2 endet

Nach diesem Schema verlaufende Simulationen heißen „Schritt für Schritt Simulation“.

Beweis (fortgesetzt)

Die wesentliche Schwierigkeit besteht darin, die Beschriftung der k-Band TM M auf einem einzigen Band unterzubringen. M' benutzt dazu ein Band mit k Spuren. Dabei soll stets gelten:

(1) Spur i des Bandes von M' enthält die Beschriftung von Band i von M $(1 \leq i \leq k)$.

(2) Zelle 1 von M' enthält genau die k Symbole, auf denen die k Köpfe von M positioniert sind.

(3) Zu Beginn der Simulation des nächsten Rechenschrittes von M befindet sich der Kopf von M' auf Zelle 1.

Bedingungen (2) und (3) sorgen dafür, daß M' die von M gelesenen k Symbole kennt.

Beweis (fortgesetzt)

Um einen Schritt von M zu simulieren, geht M' vor wie folgt:

- Wenn M Symbole $a_1, \ldots, a_k$ durch $b_1, \ldots, b_k$ ersetzt, ersetzt M' in Zelle 1 $(a_1, \ldots, a_k)$ durch $(b_1, \ldots, b_k)$.
- Wenn M Kopf i nach rechts (bzw. links) bewegt, so verschiebt M' die Inschrift von Spur i um eine Position in die entgegengesetzte Richtung (positioniert aber im Anschluss den Kopf wieder auf Zelle 1).
- Wenn M in Zustand z übergeht, geht M' in Zustand z' über.

Hierdurch bleiben Bedingungen (1), (2) und (3) erhalten und die Simulation ist korrekt.

Offensichtlich arbeitet M' deterministisch, falls M deterministisch arbeitet.

Zusätzliche Beobachtung

Wenn M auf Eingaben der Länge n

- maximal $S(n)$ Zellen ihrer Bänder besucht
- und maximal $T(n)$ Schritte rechnet,

dann

- besucht M' ebenfalls maximal $S(n)$ Zellen
- und rechnet maximal $O(S(n) \cdot T(n)) = O(T(n)^2)$ Schritte (da jeder Schritt von M in $O(S(n))$ Schritten von M' simuliert werden kann).

Wir merken kurz an, dass es eine raffinierte 2-Band Simulation einer k-Band DTM gibt, die statt $O(T(n)^2)$ Schritte lediglich $O(T(n) \log T(n))$ Schritte benötigt.

Eine binäre Kodierung von Turing–Maschinen

Arbeitsalphabet und Zustandsmenge können stets so gewählt werden, dass jedes Symbol und jeder Zustand eine Nummer erhält:

$$\begin{aligned} \Gamma &= \{A_0, \ldots, A_{r-1}\} \\ Z &= \{z_0, \ldots, z_{s-1}\} \end{aligned}$$

Dabei ist z_0 der Startzustand und oBdA gibt es nur einen Endzustand, und das ist z_{s-1}.

Ebenso können die Richtungsangaben nummeriert werden:

$$d_0 = L, d_1 = R, d_2 = N$$

Ein Eintrag $\delta(z_i, A_j) = (z_{i'}, A_{j'}, d_k)$ der Turing–Tafel kann dann durch den String

$$\#\#bin(i)\#bin(j)\#bin(i')\#bin(j')\#bin(k)$$

kodiert werden.

Die Turing–Tafel ist dann kodiert durch die Konkatenation der Codewörter ihrer Einträge (wobei diese, sagen wir, zeilenweise durchlaufen werden).

Die komplette TM ist schließlich kodiert durch die Konkatention von:

- Präambel bin(r)#bin(s)
- Kodierung der Turing-Tafel
- Endmarkierung ###, welche dafür sorgt, dass kein Codewort Präfix eines anderen Codewortes ist.

Schließlich erhalten wir ein binäres Codewort durch die Substitutionen

$$0 \mapsto 00, 1 \mapsto 01, \# \mapsto 11 \ .$$

Binäre Kodierung von Turing–Maschinen (fortgesetzt)

Es ist nicht schwer zu zeigen, dass

$$G := \{w \in \{0,1\}^* | \ w \text{ ist Codewort einer DTM}\}$$

entscheidbar ist.

- Falls $w \in G$, dann bezeichne M_w die von w kodierte DTM.
- Falls $w \notin G$, dann bezeichne M_w eine (beliebig aber fest ausgewählte) „Default–DTM“.

Damit haben wir erreicht, dass jedes Binärwort als DTM interpretierbar ist!

Universelle Turing–Maschine

Die universelle Sprache ist definiert wie folgt:

$$U := \{w\#x |\ x \in T(M_w)\}$$

Eine DTM heißt Universelle Turing–Maschine (UTM) **gdw** sie ein Akzeptor von U ist.

Eine UTM ist eine Art „General Purpose Computer“, der auf Eingaben der Form $w\#x$ vorgeht wie folgt:

- Simuliere M_w auf x.
- Akzeptiere $w\#x$ **gdw** M_w ihre Eingabe x akzeptiert.

Konstruktion einer UTM

Satz: Es gibt eine universelle Turing-Maschine (UTM).

Folgerung: U ist semi–entscheidbar.

Beweis des Satzes: Wir konstruieren eine UTM mit 3 Bändern, die wie folgt eingesetzt werden:

- Band 1 dient als „Programmspeicher“. Es enthält dauerhaft den String w, der als Code der DTM M_w interpretiert wird.
- Band 2 enthält die Binärkodierung $\mathrm{bin}(i)$ des aktuellen Zustands z_i von M_w.
- Band 3 dient als „Rechenspeicher“. Es enthält die aktuelle Bandinschrift der 1-Band DTM M_w in kodierter Form. Dabei wird das Symbol $A_j \in \Gamma$ kodiert als String $\mathrm{bin}(j)\$\ldots\$ \in \{0,1,\$\}^\ell$ mit $\ell = \lceil \log r \rceil$, wobei r die Größe des Bandalphabets Γ von M_w bezeichnet. Durch die „Auspolsterung“ mit $\$\ldots\$$ wird erreicht, dass alle Codewörter für Bandalphabetssymbole die selbe Länge ℓ haben.

Konstruktion einer UTM (fortgesetzt)

Die UTM vollzieht eine Schritt–für–Schritt Simulation von M_w. Ein Rechenschritt von M_w gemäß $\delta(z_i, A_j) = (z_{i'}, A_{j'}, R)$ (analog für die anderen Richtungswechsel des Kopfes) verändert die Konfiguration der Rechnung von M_w wie folgt:

$$\cdots z_i A_j A_k \cdots \vdash_{M_w} \cdots A_{j'} z_i A_k \cdots \quad .$$

Die UTM (mit bin(i) auf Band 2 und dem ersten Symbol von bin(j) unter dem Lesekopf von Band 3) simuliert diesen Schritt wie folgt:

1. Finde im „Programm“ w auf Band 1 den Teilstring für den Eintrag $\delta(z_i, A_j) = (z_{i'}, A_{j'}, R)$ der Turing-Tafel.
2. Ersetze auf Band 2 „bin(i)“ durch „bin(i')“.
3. Ersetze auf Band 3 „bin(j)\$ . . . \$“ durch „bin(j')\$ . . . \$“ und bewege den Kopf auf das erste Bit dahinter (also das erste Bit von „bin(k)\$ . . . \$“).

Auf diese Weise kann die Schritt–für–Schritt Simulation in Gang gehalten werden.

Zusätzliche Beobachtung

- Die beschriebene UTM kann eine DTM mit Zeitschranke $T(n)$ in $O(T(n)^2)$ Schritten simulieren.
- Der „quadratische Blow-up“ rührt daher, dass k Bänder der simulierten DTM auf *einem* Band der UTM untergebracht werden.
- Wenn wir die k Bänder auf *2 Bändern* der UTM unterbringen, dann kann die Simulation in $O(T(n) \log T(n))$ Schritten erfolgen.