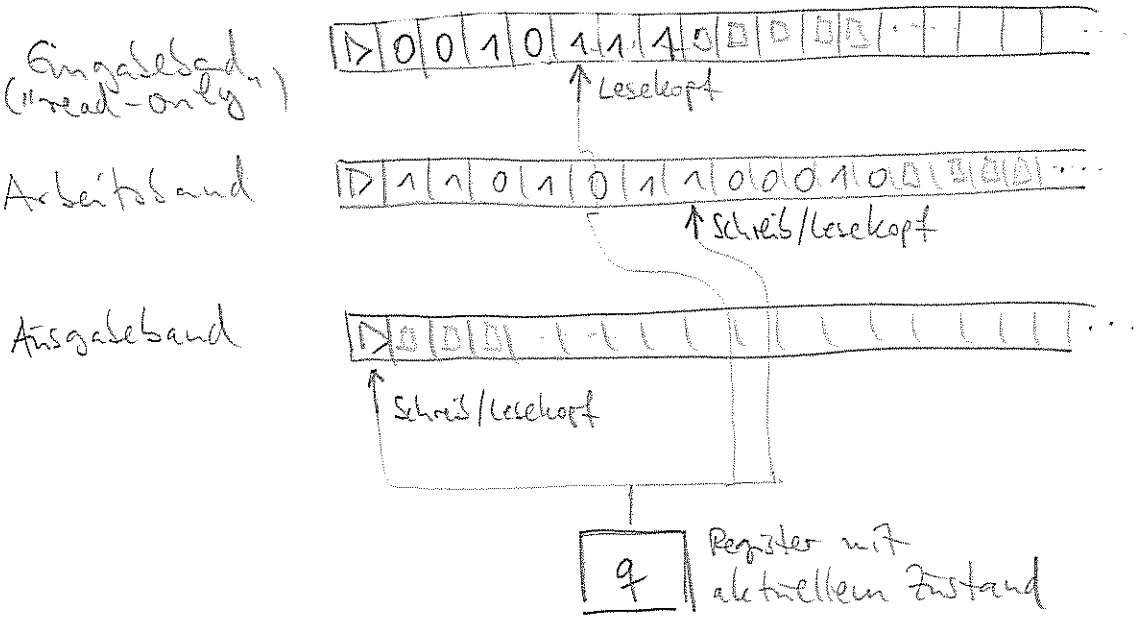
Kapitel 1:
Das Berechnungsmodell – und warum Details der Definition nicht wichtig sind

1.1 Turingmaschinen

k-Band-TM: (hier für $k=3$)



Definition 1.1

Sei $k \geq 2$. Eine (deterministische) (k-Band) Turingmaschine M (kurz: TM) wird durch ein Tupel (Γ, Q, δ) beschrieben, wobei

- Γ eine endliche Menge von Symbolen ist, die in den Zellen der k Bänder von M stehen können. Dabei gilt $\Gamma \supseteq \{0, 1, \triangleright, \square\}$, wobei $\square$ das Blank-Symbol ist und $\triangleright$ ein besonderes Start-Symbol, das in der linkesten Zelle jedes Bandes steht. Γ heißt das Alphabet von M

- Q eine endliche Menge von Zuständen ist, die sich im Register von M befinden können.

  Dabei gilt $Q \supseteq \{q_{start}, q_{halt}\}$, wobei q_{start} der Startzustand und q_{halt} der Haltezustand ist.

- Einer Funktion $\delta: Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{L, S, R\}^k$

  (L: left, S: stay, R: right)

  der sog. Transitionsfunktion von M.

Ist M in Zustand $q \in Q$ und liest die Symbole $(\sigma_1, \dots, \sigma_k)$ an den Schreib/Leseköpfen der Bänder $1, \dots, k$ und ist $\delta(q, \sigma_1, \dots, \sigma_k) = (q', \sigma_2', \dots, \sigma_k', B)$ mit $B \in \{L, S, R\}^k$, so werden im nächsten Berechnungsschritt die an den Schreib-/Leseköpfen der Bänder $2, \dots, k$ stehenden Symbole $\sigma_2, \dots, \sigma_k$ durch die Symbole $\sigma_2', \dots, \sigma_k'$ ersetzt, der Zustand q im Register wird durch den Zustand q' ersetzt, und die Schreib-/Leseköpfe der Bänder $1, \dots, k$ bewegen sich um einen Schritt in die durch B angegebene Richtung (d.h.: der Kopf auf Band i bewegt sich nach links / nach rechts / gar nicht, falls $B_i = L$, $B_i = R$, $B_i = S$).

Falls ein Schreib/Lesekopf, der auf dem linksten Band-symbol steht, um einen Schritt nach links bewegt werden soll, dann bleibt er stattdessen einfach auf der

linkesten Bandposition.

Startkonfiguration bei Eingabe $x \in \{0,1\}^*$:

- Zustand q_{start}
- Die Schreib/Leseköpfe der k Bänder stehen jeweils auf der linkesten Bandposition
- Auf dem Eingabeband befindet sich ganz links das Startsymbol $\triangleright$, auf den nächsten $|x|$ Bandpositionen steht das Wort x, und alle anderen Bandpositionen sind mit dem Blanksymbol $\square$ beschriftet
- Auf jedem der Bänder $2, \dots, k$ befindet sich ganz links das Startsymbol $\triangleright$; alle anderen Bandpositionen sind mit dem Blanksymbol $\square$ beschriftet.

Berechnungsschritte:

Jeder Berechnungsschritt erfolgt durch Anwenden der Transitionsfunktion δ, wie oben beschrieben

Ende der Berechnung:

δ muss die folgende Eigenschaft haben: Ist M in Haltezustand q_{halt}, so wird weder die Bandbeschriftung noch der aktuelle Zustand geändert

Die Beschriftung des k-ten Bandes zwischen dem Start-symbol $\triangleright$ und dem ersten Symbol aus $\Gamma \setminus \{0,1\}$ ist dann die Ausgabe von M bei Eingabe x, kurz: $M(x)$.

Beispiel 1.2 (Palindrome)

Sei $PAL : \{0,1\}^* \to \{0,1\}$ wie folgt definiert:

F.a. $x \in \{0,1\}^*$ ist $PAL(x) := \begin{cases} 1 & \text{falls } x \text{ ein Palindrom ist} \\ 0 & \text{sonst} \end{cases}$

(Zur Erinnerung: $x = x_1 \cdots x_n$ ist ein Palindrom, falls $x_1 \cdots x_n = x_n \cdots x_1$).

Eine 3-Band TM M, die PAL berechnet, kann wie folgt vorgehen:

(1) Kopiere die Eingabe auf das Arbeitsband (Band 2)

(2) Bewege den Lesekopf des Eingabebandes wieder ganz nach links zum Startsymbol

(3) Lies parallel das Eingabeband von links nach rechts und das Arbeitsband von rechts nach links. Falls dabei auf den beiden Bändern irgendwann verschiedene Symbole gelesen werden, so schreibe "0" auf Ausgabeband und gehe in Zustand q_{halt}

(4) Falls der Kopf des Eingabebandes am ersten Blanksymbol $\square$ und der Kopf des Arbeitsbandes am Startsymbol $\triangleright$ angekommen ist, so schreibe "1" auf Ausgabeband und gehe in Zustand q_{halt}

Details: $M = (\Gamma, Q, \delta)$ mit

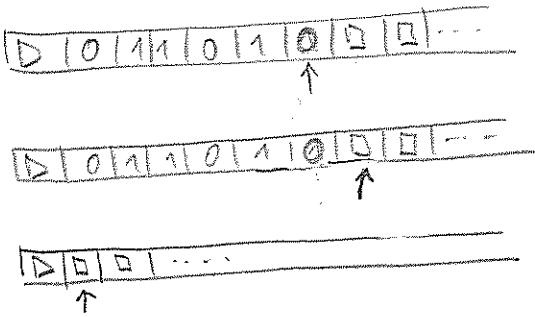
- $\Gamma = \{0, 1, \triangleright, \square\}$,
- $Q = \{q_{start}, q_{copy}, q_{left}, q_{test}, q_{halt}\}$
- δ wie folgt:

(1) $\delta(q_{start}, \triangleright, \triangleright, \triangleright) = (q_{copy}, \triangleright, \triangleright, R, R, R)$

$\delta(q_{copy}, a, \square, \square) = (q_{copy}, a, \square, R, R, S)$ für $a \in \{0,1\}$

$\delta(q_{copy}, \square, \square, \square) = (q_{left}, \square, \square, L, S, S)$

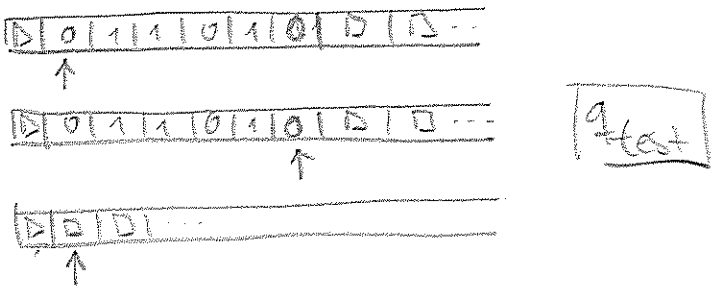
Situation: wenn erstmals in q_{left}



(2): $\delta(q_{left}, a, \square, \square) = (q_{left}, \square, \square, L, S, S)$ für $a \in \{0,1\}$

$\delta(q_{left}, \triangleright, \square, \square) = (q_{test}, \square, \square, R, L, S)$

Situation wenn erstmals in q_{test}:



(3) $\delta(q_{test}, a, a, \square) = (q_{test}, a, \triangleright, R, L, S)$ für $a \in \{0,1\}$

$\delta(q_{test}, a, b, \triangleright) = (q_{halt}, b, 0, S, S, S)$ für $a, b \in \{0,1\}$ mit $a \neq b$

(4) $\delta(q_{test}, \square, \triangleright, \square) = (q_{halt}, \triangleright, 1, S, S, S)$

und $\delta(q, a, b, c) = (q, b, c, S, S, S)$ für alle noch nicht genannte weite $(q, a, b, c) \in Q \times \Gamma^3$.

1.2 Effizienz und Laufzeit

Ansatz: Zähle die Anzahl der Berechnungsschritte einer TM in Bezug auf die Länge n des Eingabewortes.

Definition 1.3 (Berechnung einer Funktion, Laufzeit)

Sei $f: \{0,1\}^* \to \{0,1\}^*$ und sei $T: \mathbb{N} \to \mathbb{N}$.

Sei M eine TM.

(a) Wir sagen: M berechnet f, falls $\forall x \in \{0,1\}^*$ gilt: Wird M in der Startkonfiguration bei Eingabe x gestartet, so gelangt M nach endlich vielen Berechnungsschritten in den Haltezustand q_{halt} und steht dann das Wort $f(x)$ auf dem Band.

(b) Wir sagen: M berechnet f in $T(n)$ Schritten, falls M f berechnet und für jede Eingabe $x \in \{0,1\}^*$ gilt: M gelangt nach höchstens $T(|x|+1)$ Berechnungsschritten in den Endzustand q_{end}.

Beispiel 1.4

Die TM aus Bsp 1.2 berechnet PAL in $3n$ Schritten.

Definition 1.5 (zeitkonstruierbarkeit)

Eine Funktion $T: \mathbb{N} \to \mathbb{N}$ heißt zeitkonstruierbar, falls gilt:

(1) $T(n) \geq n$ und

(2) Es gibt eine TM M, die die Funktion $f: \{0,1\}^* \to \{0,1\}^*$ mit $f(x) = \underline{T(|x|)}$ (f.a. $x \in \{0,1\}^*$) in Zeit $O(T(n))$ berechnet.

(Beachte: Wie in Kapitel 0 vereinbart ist $\underline{T(|x|)}$ die Binärdarstellung der Zahl $T(|x|)$

Fakt 1.6

z.B. die folgenden Funktionen sind zeitkonstruierbar:

(a) 2^n (genauer: $T: \mathbb{N} \to \mathbb{N}$ mit $T(n) = 2^n$ f.a. $n \in \mathbb{N}$)

(b) n^2

(c) n^c, für jedes $c \in \mathbb{N}$ mit $c \geq 2$

(d) $n \cdot \log n$

Beweis:

(a) Bei Eingabe der Länge n muss in $O(2^n)$ Schritten die Binärdarstellung der Zahl 2^n, d.h. der Bitstring 10^n berechnet werden. Das geht ganz leicht, sogar in $O(n)$ Schritten.

(b), (c), (d): Übung!

Frage: Ist $\underline{n}$ zeitkonstruierbar?
(d.h. die Funktion $T: \mathbb{N} \to \mathbb{N}$ mit $T(n) = n$ f.a. $n \in \mathbb{N}$)

1.3. Robustheit der Definition

Der in Definition 1.3 gegebene Begriff der "Laufzeit" ist — modulo polynomieller Faktoren — unabhängig von technischen Details des konkreten Berechnungsmodells.

Beispielsweise stellt es keine wesentliche Einschränkung dar, zu fordern, dass das Alphabet Γ einer TM genau die Menge $\{\triangleright, \square, 0, 1\}$ ist. Genauer:

Fakt 1.7

Für jedes $f: \{0,1\}^* \to \{0,1\}^*$ und jedes $T: \mathbb{N} \to \mathbb{N}$ gilt: Falls es eine TM M gibt, die f in $T(n)$ Schritten berechnet, so gibt es eine TM $\tilde{M}$ mit Alphabet $\{\triangleright, \square, 0, 1\}$, die f in $5 \cdot \log|\Gamma| \cdot T(n)$ Schritten berechnet, wobei Γ das Alphabet von M ist.

Beweisskizze: Sei M eine k-Band TM mit Alphabet Γ und Zustandsmenge Q, die f in $T(n)$ Schritten berechnet.

Repräsentiere jedes Symbol $\tau \in \Gamma$ durch einen Bitstring der Länge $\log|\Gamma|$.

$\tilde{M}$ hat $k+1$ Bänder: Band 1 ist das "read-only" Eingabeband, das genauso genutzt wird wie das Eingabeband von M.
Die Bänder $2, \dots, k$ sind die Arbeitsbänder, wobei jede Zelle des i-ten Bandes von M (für $2 \le i \le k$) durch $\log|\Gamma|$ Zellen des i-ten Bandes von $\tilde{M}$ repräsentiert wird.

Jeder einzelne Schritt von M wird durch $\tilde{M}$ wie folgt simuliert:

(1) Nutze $\log|\Gamma|$ Schritte, um auf jedem der Bänder $2, \dots, k$ die entsprechenden $\log|\Gamma|$ Zellen zu lesen, die die Repräsentation der aktuell von M auf den Bändern $2, \dots, k$ gelesenen Symbole aus Γ enthalten.
Nutze den aktuellen Zustand, um
- den aktuellen Zustand von M,
- einen Zähler $i \in \{1, \dots, \log|\Gamma|\}$,
- den auf jedem der Bänder $2, \dots, k$ in den ersten i solchen Schritten gelesenen Bitstring (für $i \in \{1, \dots, \log|\Gamma|\}$)

zu speichern. Beachte: Dafür reichen $|Q| \cdot (\log|\Gamma|) \cdot \left(2^{\log|\Gamma|}\right)^k \le |Q| \cdot (\log|\Gamma|) \cdot (2|\Gamma|)^k$ Zustände.

(2) Nach diesen $\log|\Gamma|$ Schritten hat $\tilde{M}$ im Zustand gespeichert, welche Symbole M liest und in

welchem Zustand M ist. Die Transitionsfunktion von M gibt an, welchen neuen Zustand M annimmt, welche Symbole geschrieben werden sollen und in welche Richtung sich die Köpfe von M bewegen sollen. Dies kann wieder im aktuellen Zustand von $\tilde{M}$ gespeichert werden (ähnlich wie in (1)).

(3) In den nächsten $\log|T|$ Schritten läuft $\tilde{M}$ auf den Bändern $2,\dots,k$ jeweils nach links und schreibt dabei die Repräsentation des von M zu schreibenden Symbols ans Band.

(4) Danach macht $\tilde{M}$ nochmal $\log|T|$ Schritte, um die Köpfe der Bänder $2,\dots,k$ an den Anfang des richtigen Blocks der Länge $\log|T|$ zu setzen.

Insgesamt macht $\tilde{M}$ also $3\log|T|$ Schritte, um einen Schritt von M zu simulieren.

Nach $T(n)$ Schritten von M, also $3\log|T|\cdot T(n)$ Schritten von $\tilde{M}$ steht auf dem k-ten Band von $\tilde{M}$ die Repräsentation der Ausgabe von M. Diese hat die Länge

$\leq 3\log|T|\cdot T(n)$.

$\tilde{M}$ bewegt nun noch den Kopf des k-ten Bandes ganz nach links und liest dann nochmal den Inhalt des k-ten Bandes und erzeugt dabei auf Band $k+1$ den Bitstring, der die "Originalausgabe" von M ist. Insgesamt ist $\tilde{M}$ nach $\leq 9\cdot\log|T|\cdot T(n)$ Schritten fertig. □

Auch die Verfügbarkeit von k Bändern – im Gegensatz zu einem einzigen Arbeitsband – macht das Berechnungsmodell nicht wesentlich stärker.

Genauer:

Definition 1.8 (1-Band TM)

Eine 1-Band TM ist eine Turingmaschine, die ein einziges Band besitzt, von dem gelesen und auf das geschrieben werden kann. Zu Beginn der Berechnung ist auf diesem Band die Eingabe gegeben; am Ende der Berechnung soll dieses Band die Ausgabe enthalten.

Fakt 1.9

Für jedes $f: \{0,1\}^* \to \{0,1\}^*$ und jedes $T: \mathbb{N} \to \mathbb{N}$ gilt: Falls es eine k-Band TM M gibt, die f in $T(n)$ Schritten berechnet, so gibt es auch eine 1-Band TM $\tilde{M}$, die f in $\mathcal{O}(T(n)^2)$ Schritten berechnet.

Beweisidee: $\tilde{M}$ simuliert die k Bänder von M, indem sie ihr Band in k "Spuren" aufteilt. $\tilde{M}$ simuliert einen einzelnen Schritt von M, indem sie 4 mal den gesamten Inhalt ihres Bandes liest und modifiziert (1x v.l.n.r, um zu ermitteln, was M gerade liest,

2 weitere Male, um die von M geschriebenen Symbole und die Markierungen für M's aktuelle Kopfpositionen zu setzen, und 1 weiteres Mal um den Kopf wieder ganz nach links zu setzen).

Details: [AB], Seite 18
bzw. Kapitel 5 in Prof Schnitgers Skript zur Veranstaltung "Algorithmentheorie".

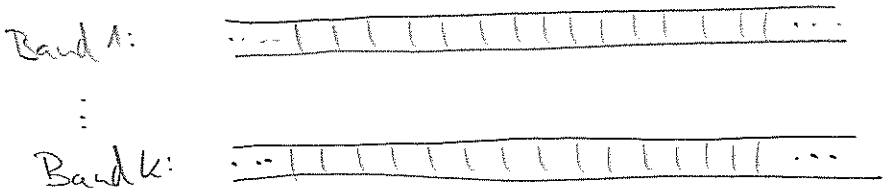
Bemerkung 1.10:
Obige Beweisidee lässt sich so präzisieren, dass man eine so genannte stereotype Turingmaschine (engl: oblivious TM) $\tilde{M}$ erhält, d.h. eine TM, deren Kopfbewegungen nur von der Länge der Eingabe, nicht aber von der konkreten Eingabe selbst abhängen.
Details: Übungsblatt 1.

(Bei Beweisen in späteren Kapiteln wird es manchmal nützlich sein, dass wir o.B.d.A stereotype TMs betrachten können.)

Auch die Tatsache, dass die Bänder unserer TMs einseitig beschränkt sind, stellt keine wesentliche Einschränkung dar:

Definition 1.11 (bidirektionale TM)

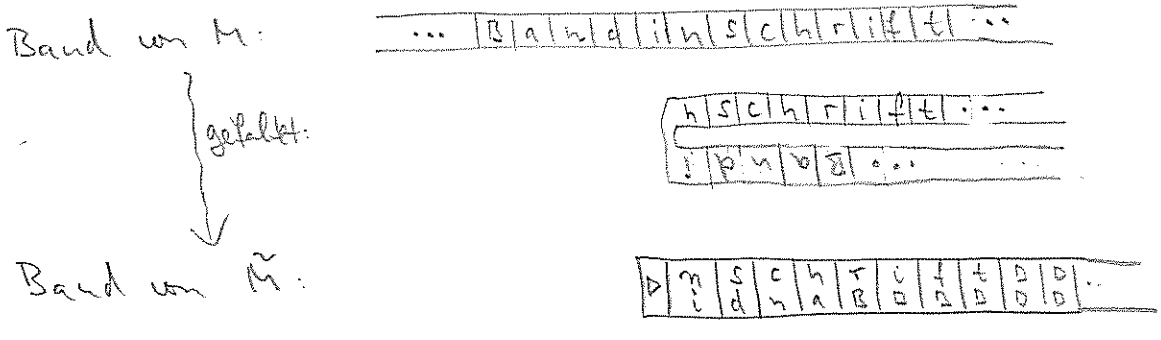
Eine bidirektionale TM ist eine TM, deren Bänder nach beiden Seiten unendlich lang sind.



Fakt 1.12

Für jedes $f: \{0,1\}^* \to \{0,1\}^*$ und jedes $T: \mathbb{N} \to \mathbb{N}$ gilt:
Falls es eine bidirektionale TM M gibt, die f in $T(n)$ Schritten berechnet, so gibt es auch eine TM $\tilde{M}$ (im Sinne von Definition 1.1), die f in $O(T(n))$ Schritten berechnet.

Beweisidee: Falte jedes Band von M in der Mitte zusammen



Alphabet von $\tilde{M}$: $\Gamma \times \Gamma$, wobei Γ das Alphabet von M ist.
Details: Übung. □

Aus der Veranstaltung "Algorithmentheorie" kennen Sie auch ein weiteres Berechnungsmodell, das in polynomiell vielen Schritten durch TMs simuliert werden kann: die so genannten

Registermaschinen.

Hausaufgabe: Lesen Sie Kapitel 5 des Skripts zur Veranstaltung "Algorithmentheorie".

1.4 Repräsentation von TMs durch Worte — und die Universelle Turingmaschine

Turingmaschinen können auf naheliegende Weise durch Worte repräsentiert werden (einfach indem man die Komponenten (T, Q, δ) hinschreibt und auf geeignete Weise durch einen Bitstring repräsentiert). Wir werden im Folgenden davon ausgehen, dass unsere Repräsentation von TMs durch Bitstrings folgende Eigenschaften hat:

(1) Jedes Wort in $\{0,1\}^*$ repräsentiert irgendeine TM.
(Dies kann man leicht dadurch erreichen, dass man jedem Bitstring, der keine "sinnvolle" Kodierung einer TM darstellt, die "triviale" TM zuweist, die — unabhängig von der Eingabe — direkt anhält und "0" ausgibt.)

(2) Jede TM wird durch unendlich viele Bitstrings repräsentiert.

(Dies kann man z.B. dadurch erreichen, dass ans Ende jeder "sinnvollen" Kodierung einer TM beliebig viele 1en angehängt werden dürfen.)

Konvention 1.13

(a) Wir schreiben im Folgenden $\langle M \rangle$ um einen Bitstring zu bezeichnen, der die TM M repräsentiert

(b) Für $\alpha \in \{0,1\}^*$ schreiben wir M_α, um die TM zu bezeichnen, die durch α repräsentiert wird.

Die Repräsentation von TMs durch Bitstrings ermöglicht es uns, einer TM die Kodierung einer anderen TM als Eingabe zu übergeben. Dadurch ist sozusagen die Unterscheidung zwischen "Hardware" (Bänder, Zustände), "Software" (Transitionsfunktion) und "Daten" (Bitstring als Eingabe) aufgehoben, und wir können TMen betrachten, die beliebige andere TMen simulieren: so genannte Universelle Turingmaschinen

Theorem 1.14 (Effiziente universelle TM)

Es gibt eine TM U s.d. f.a. $x \in \{0,1\}^*$ und f.a. $\alpha \in \{0,1\}^*$ gilt: $U(\langle x, \alpha \rangle) = M_\alpha(x)$.

D.h.: Bei Eingabe (der Kodierung) des Tupels $\langle x, \alpha \rangle$ gibt U genau das aus, was die von α repräsentierte TM M_α bei Eingabe x ausgibt.

Außerdem gilt: Falls M_α bei Eingabe x nach t Schritten anhält, so hält U bei Eingabe $\langle x, \alpha \rangle$ nach $c \cdot t \cdot \log t$ Schritten an, wobei die Zahl c nur von der Alphabetgröße und der Anzahl der Bänder von M_α abhängt, nicht aber vom Eingabewort x. Ferner ist $c \leq \text{poly}(|\alpha|)$ — bei geeigneter Wahl der Repräsentation von TMen durch Bitstrings.

Beweisidee: Der Beweis geht auf Hennie und Stearns (1966) zurück; Details finden sich in Abschnitt 1.7 von [AB]

Wir beweisen hier eine etwas schwächere Version des Theorems, die 1965 von Hartmanis und Stearns gezeigt wurde, und in der U an Stelle von $c \cdot t \cdot \log t$ Schritten $c \cdot t^2$ Schritte zur Verfügung hat.

U geht bei Eingabe $\langle x, \alpha \rangle$ wie folgt vor:

(1) U transformiert α in eine Repräsentation einer zu M_α äquivalenten TM M, deren Alphabet $\{\triangleright, \sqcup, 0, 1\}$ ist und die neben dem Eingabe- und dem Ausgabeband nur ein weiteres Arbeitsband benutzt. Dann geht U vor wie in den

Beweisen von Fakt 1.7 und Fakt 1.9.

Beachte: Wenn M_a bei Eingabe x t Schritte macht, so macht M bei Eingabe x dann $\leq c \cdot t^2$ Schritte, für eine Zahl c, die nur von M_a, nicht aber von x abhängt.

Außerdem hängt die Anzahl der Schritte, die U in dieser Phase (1) macht, nur von a, nicht aber von x ab.

(2) U benutzt Eingabeband, Ausgabeband und ein Arbeitsband, um die entsprechenden Bänder von M bei Eingabe x zu simulieren.

Außerdem benutzt U 2 weitere Arbeitsbänder: eins, um M's aktuellen Zustand zu speichern, und eins, um eine Repräsentation der Transitionsfunktion von M zu speichern.

U simuliert jeden einzelnen Schritt von M bei Eingabe x, indem sie zunächst das Arbeitsband, das die Repräsentation von M's Transitionsfunktion nach der als nächstes auszuführenden Aktion durchsucht und dann den entsprechenden neuen Zustand von M auf das andere Arbeitsband schreibt und die entsprechende Symbole auf die Bänder schreibt, die die Bänder von M simulieren und die Köpfe dieser Bänder entsprechend bewegt.
Somit kann jeder Schritt von M durch eine Anzahl d von Schritten von U simuliert werden, wobei d nur von M (also a), nicht aber von x abhängt

Insgesamt macht U also $\leq d \cdot c \cdot t^2$ Schritte in Phase (2).

□

Bemerkung 1.15 (Universelle TM mit Zeitbeschränkung)

Manchmal ist es nützlich, eine Variante der univ. TM zu betrachten, deren Eingabe aus einem Tripel $\langle x, \alpha, t \rangle$ mit $x, \alpha \in \{0,1\}^*$ und $t \in \mathbb{N}$ besteht, und die

- $M_\alpha(x)$ ausgibt, falls M_α bei Eingabe x nach $\leq t$ Schritten anhält
- ein spezielles "Fehlersymbol" ausgibt, sonst.

Obiger Beweis von Theorem 1.14 lässt sich leicht modifizieren, um eine solche univ. TM mit Zeitbeschränkung zu erhalten: U bekommt einfach ein zusätzliches Arbeitsband, auf dem der Schrittzähler von t nach und nach runtergezählt wird, bis 0 erreicht ist. Danach bricht die Maschine die Berechnung einfach ab und erzeugt die entsprechende Ausgabe.

1.5 Kurzer Exkurs zum Thema "Unentscheidbarkeit"

Theorem 1.16

Es gibt eine Funktion $UC: \{0,1\}^* \to \{0,1\}$, die von keiner TM berechnet werden kann.

Beweis: Wähle $UC: \{0,1\}^* \to \{0,1\}$ wie folgt:

F.a. $\alpha \in \{0,1\}^*$ ist

$$UC(\alpha) := \begin{cases} 0 & \text{falls } M_\alpha(\alpha) = 1 \\ 1 & \text{sonst (d.h. falls } M_\alpha \text{ bei Eingabe } \alpha \text{ nie anhält oder einen Wert} \neq 1 \text{ ausgibt)} \end{cases}$$

Angenommen, UC wird von einer TM M berechnet.

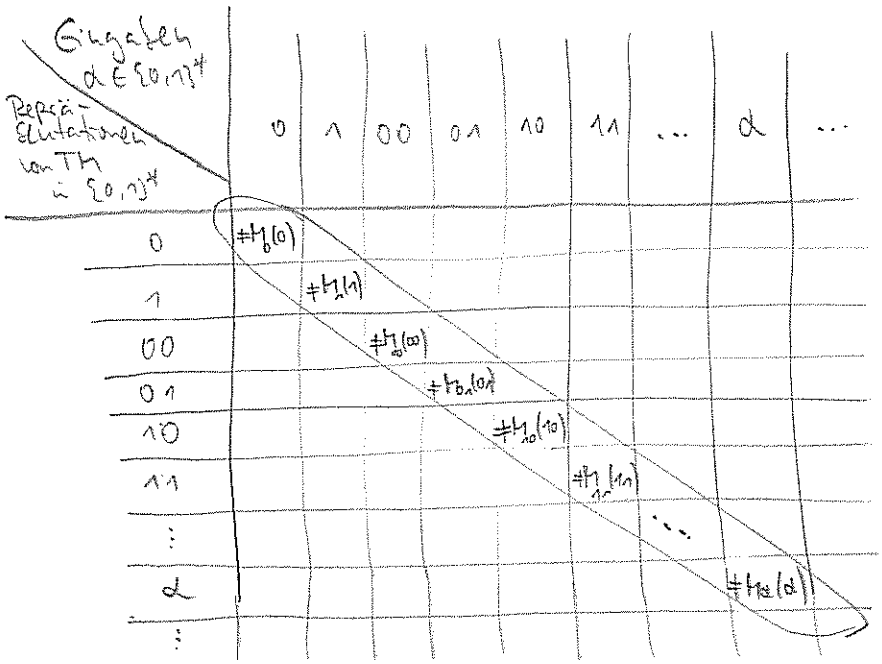
Dann gilt: $M(\alpha) = UC(\alpha)$, f.a. $\alpha \in \{0,1\}^*$.

Also gilt insbes. für $\alpha := \langle M \rangle$: $M(\langle M \rangle) = UC(\langle M \rangle)$.

Aber gemäß der Definition von UC gilt: $M(\langle M \rangle) \neq UC(\langle M \rangle)$.

↯ □

Dieses Verfahren wird Diagonalisierung genannt. Skizze:



UC ist entsprechend der Diagonalen definiert s.d. $UC(\alpha) \neq M_\alpha(\alpha)$ f.a. $\alpha \in \{0,1\}^*$ gilt.

Insbes. kann es daher kein $\alpha \in \{0,1\}^*$ geben, s.d. UC von M_α berechnet wird (da $UC(\alpha) \neq M_\alpha(\alpha)$ ist!)

Das Halteproblem

Definition 1.17 (HALT)

$$\text{HALT}: \{0,1\}^* \times \{0,1\}^* \to \{0,1\} \quad \text{wobei, } \forall\, \alpha, x \in \{0,1\}^*:$$

$$\text{HALT}(\alpha, x) := \begin{cases} 1 & \text{falls } M_\alpha \text{ bei Eingabe } x \text{ nach endlich vielen Schritten anhält} \\ 0 & \text{sonst.} \end{cases}$$

Theorem 1.18

Es gibt keine TM, die HALT berechnet.

Beweis:

Angenommen, M_{HALT} ist eine TM, die HALT berechnet.
Wir nützen M_{HALT}, um eine TM M_{UC} zu konstruieren, die UC berechnet (— im Widerspruch zu Theorem 1.16):

Bei Eingabe α schreibt M_{UC} die Kodierung des Tupels $\langle \alpha, \alpha \rangle$ auf's Band und startet dann M_{HALT}. Falls M_{HALT} bei Eingabe $\langle \alpha, \alpha \rangle$ den Wert 0 ausgibt, so schreib 1 auf's Ausgabeband und halte an. Falls M_{HALT} bei Eingabe $\langle \alpha, \alpha \rangle$ den Wert 1 ausgibt, so starte die universelle TM U mit Eingabe (α, α), um M_α bei Eingabe α zu simulieren und den Wert $b := M_\alpha(\alpha)$ zu berechnen. Falls $b = 1$ ist, so gib 0 aus; ansonsten gibt 1 aus und halte an. □

Bemerkung: Wir haben hier UC auf HALT reduziert.

1.6 Die Klasse P

Eine Komplexitätsklasse ist eine Menge von Funktionen, die mit bestimmten Ressourcen berechnet werden können

Zumeist werden hier Boolesche Funktionen, d.h. Funktionen $f: \{0,1\}^* \to \{0,1\}$ betrachtet.
Diese beschreiben Entscheidungsprobleme oder Sprachen.

Definition 1.19:
Eine TM M entscheidet eine Sprache $L \subseteq \{0,1\}^*$, falls sie die Funktion $f_L: \{0,1\}^* \to \{0,1\}$ mit

$$f_L(x) = \begin{cases} 1 & \text{falls } x \in L \\ 0 & \text{sonst} \end{cases}$$

berechnet.

Definition 1.20 ($DTIME(T(n))$)
Sei $T: \mathbb{N} \to \mathbb{N}$ eine Funktion.

$$DTIME(T(n)) := \{ L \subseteq \{0,1\}^* : \text{es gibt eine TM } M, \text{ die } L \text{ in } O(T(n)) \text{ Schritten entscheidet} \}$$

(Bemerkung: Das "D" von "DTIME" steht für "deterministisch")

Definition 1.21 (Die Klasse P)

(a) $P := \bigcup_{c \geq 1} DTIME(n^c)$.

P ist die Klasse aller deterministisch in Polynomialzeit lösbaren Entscheidungsprobleme.

(b) Eine Polynomialzeit-TM ist eine TM, die eine Funktion $f: \{0,1\}^* \to \{0,1\}^*$ in $poly(n)$ Schritten berechnet (d.h. in $O(n^c)$ Schritten, für ein $c \in \mathbb{N}$).

Beispiele 1.22

(a): Erreichbarkeit $:= \{ \langle G, s, t \rangle :$ G ist ein Graph, $s, t \in V(G)$, und es gibt in G einen Weg von s nach t $\}$

und

Zusammenhang $:= \{ G :$ G ist ein zusammenhängender ungerichteter Graph $\}$

sind beides Probleme, die in P liegen

(b): Da P nur aus Entscheidungsproblemen besteht, macht es keinen Sinn, zu sagen

"die Multiplikation zweier nat. Zahlen liegt in P".

Stattdessen gilt aber:

$\{ \langle a,b,c \rangle : a,b,c \in \mathbb{N}$ und $a \cdot b = c \} \in P$.

Man beachte, dass die (polynomielle) Laufzeit hier in der Länge der Bitrepräsentation der Zahlen gemessen wird. Der Algorithmus "wiederholtes Addieren" aus Kapitel 0 liefert daher kein Polynomialzeit-

Verfahren — wohl aber der Algorithmus "Grundschulmultiplikation".

Bemerkung:

Unabhängig davon, welches der in Kapitel 1.3 betrachteten konkreten Berechnungsmodelle wir zu Grunde legen, gehören immer die gleichen Probleme zur Klasse P — die Klasse P ist also "robust" (d.h. unabhängig von der konkreten Wahl des Maschinenmodells).

Auch aus Sicht eines Programmierers scheint P eine sinnvolle Formalisierung des Begriffs "effizient berechenbar" zu sein:
Wenn wir annehmen, dass alles, was in Linearzeit oder quadratischer Zeit berechenbar ist, "effizient berechenbar" ist — und außerdem erlauben wollen, in Programmen "effizient berechenbare" Subroutinen zu benützen und dabei insgesamt nur "effizient oft" solche Subroutinen aufrufen, erhalten wir genau die Klasse P.