

Komplexitätstheorie

SoSe 2018

Thomas Schneider

Kapitel 2: Turingmaschinen

Homepage der Vorlesung: <http://tinyurl.com/ss18-kt>

Einleitung

Wir möchten oft Aussagen über **alle** Algorithmen treffen (z. B. es gibt keinen, der ein bestimmtes Problem in Polyzeit löst).

Um in diesem Kontext formale Beweise führen zu können, müssen wir festlegen, was ein Algorithmus ist.

Prinzipiell gibt es viele Alternativen:

- Reale Modelle: z. B. Java-Programme oder C-Programme
- Mathematische Modelle, inspiriert durch Hardware:
z. B. Registermaschinen (RAMs)
- Rein mathematische Modelle: z. B. Turingmaschinen

Da wir **einfache Beweise** wollen, wählen wir ein Modell, das so einfach wie möglich ist: **Turingmaschinen**

Einleitung

Turingmaschinen...

- wurden in den 1930er Jahren von Alan Turing entwickelt
- sind sehr einfach aufgebaut
(Papadimitriou: „it is amazing how little we need to have everything“)
- sind **kein** realistisches Modell für reale Computer,
leisten jedoch interessanterweise genau dasselbe

Church-Turing-These

Die durch Turingmaschinen berechenbaren Probleme sind genau die im intuitiven Sinn berechenbaren Probleme.

„These“, denn diese Aussage lässt sich nicht beweisen
(„im intuitiven Sinn berechenbar“ kann man nicht formal fassen)

Einleitung

Die Church-Turing-These rechtfertigt noch **nicht** die Verwendung von Turingmaschinen in der **Komplexitätstheorie!**

Erweiterte Church-Turing-These

Problem kann mit Zeitverbrauch t in natürlichem und generellen Berechnungsmodell gelöst werden gdw. es mit Zeitverbrauch $p(t)$ von Turingmaschinen gelöst werden kann, für ein Polynom p .

Zeitverbrauch wird gemessen durch Anzahl elementarer Schritte (was das ist, hängt vom Berechnungsmodell ab)

Wenn „effizient berechenbar“ = „in polynomieller Zeit lösbar“:
Problem ist entweder in allen Modellen effizient berechenbar oder in keinem.

Der Grad des Polynoms kann sich aber u. U. unterscheiden.

NEXT



2.1 Turingmaschinen

2.2 Probleme lösen mittels Turingmaschinen

2.3 Mehrband-Turingmaschinen

2.4 Zeit-Komplexitätsklassen

2.5 Universelle Turingmaschinen

Turingmaschinen

Turingmaschine

- ist zu jeder Zeit in einem von **endlich** vielen Zuständen
- Arbeitet auf **einseitig unendlichem** Arbeitsband, das aus Kette von linear geordneten Zellen besteht
- Jede Zelle enthält eines von endlich vielen Symbolen
- Zu Anfang enthält das Band ganz links das **Sondersymbol** \triangleright gefolgt von der Eingabe, gefolgt von unendl. oft **Sondersymbol** \perp
- Die Maschine hat einen Schreib-/Lesekopf, der zu Anfang auf dem ersten (linken) Symbol der Eingabe steht
- In jedem Schritt kann die Maschine das Symbol der aktuellen Zelle lesen und schreiben, sowie den Kopf um eine Position verschieben
- Arbeitet feste Verarbeitungsvorschrift ab

T2.1

Turingmaschinen

Definition Turingmaschine

(*Deterministische*) Turingmaschine (kurz: (D)TM) ist Tupel

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

mit

- Q endlicher Menge von Zuständen
- Σ endliches Eingabealphabet
- Γ endliches Bandalphabet mit $\Sigma \subseteq \Gamma$ und $\{\perp, \triangleright\} \subseteq \Gamma \setminus \Sigma$
- $\delta : ((Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma) \rightarrow (Q \times \Gamma \times \{L, R, N\})$ Übergangsfunktion
- $q_0 \in Q$ Startzustand
- q_{acc} akzeptierender Endzustand
- q_{rej} verwerfender Endzustand

Turingmaschinen: Beispiel

Beispiel

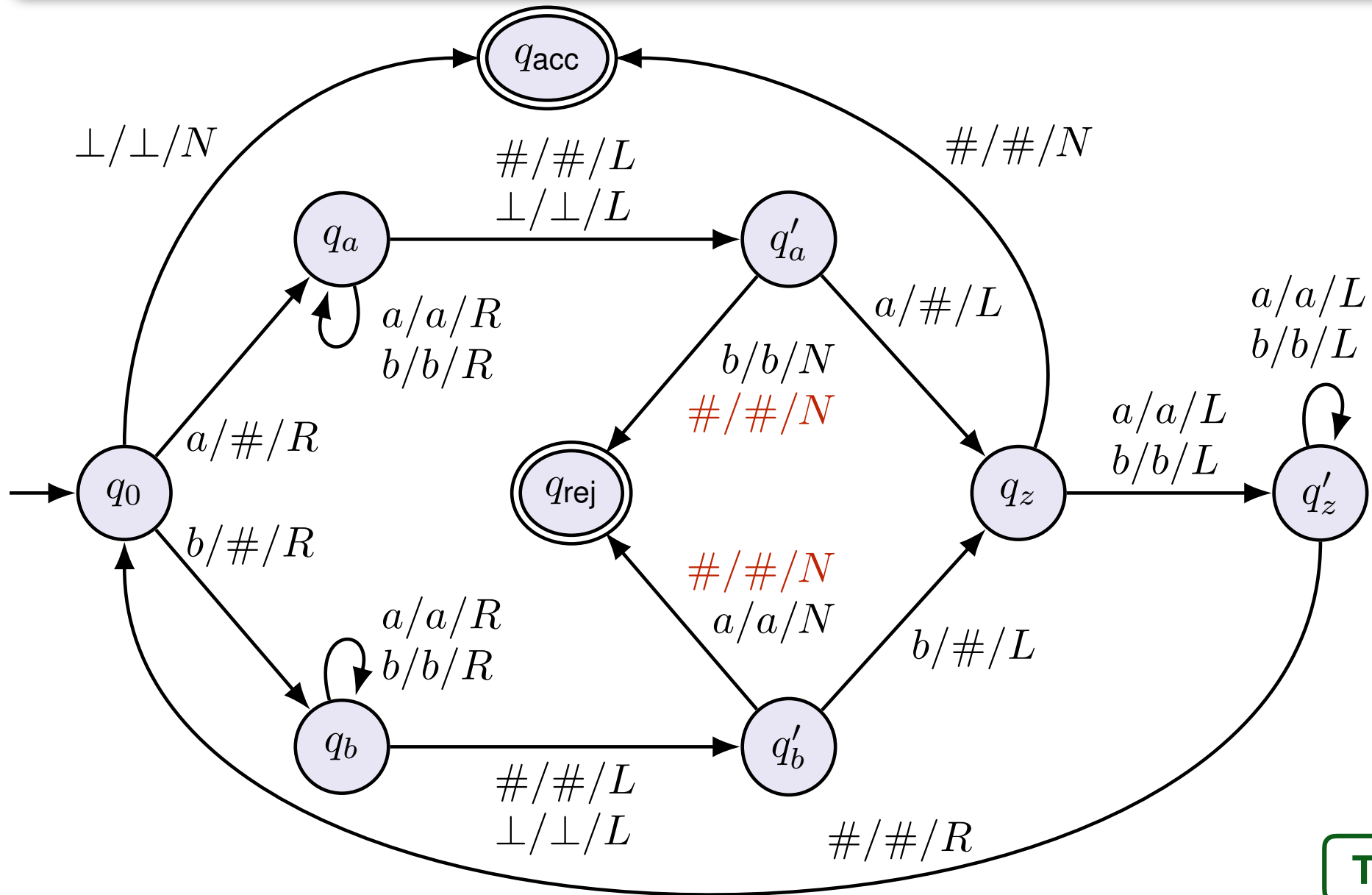
TM M , die die Eingabe $w \in \{a, b\}^*$ akzeptiert
gdw. $w = u\overleftarrow{u}$ mit $u \in \{a, b\}^*$ und $\overleftarrow{u} = u$ rückwärts gelesen

Idee

- Ersetze Symbol unter Kopf durch # und „merke“ (im Zustand), ob a oder b überschrieben wurde.
- Laufe nach rechts bis \perp oder # und gehe 1 Schritt nach links.
- Vergleiche Symbol unter Kopf mit „gemerktem“ Symbol. Wenn unterschiedlich, dann **verwirf**.
- Überschreibe aktuelles Symbol mit #.
- Laufe ganz nach links.
- Laufe nach rechts bis zum ersten Nicht-#-Symbol. Ist dieses ein \perp , dann **akzeptiere**. Sonst beginne von vorn.

T2.2

Turingmaschinen: Beispiel



T2.3

Alle fehlenden Übergänge wie z. B. $\delta(q'_z, \perp)$ werden auf (q_{rej}, \perp, N) gesetzt.

Turingmaschinen

Konvention: Wenn $\delta(q, a) = (q', a', B)$, dann verlangen wir, dass

- wenn $a = \triangleright$, dann $a' = \triangleright$ und $B = R$;
- wenn $a \neq \triangleright$, dann $a' \neq \triangleright$.

Also: \triangleright bleibt stets am linken Bandende und steht sonst nirgendwo.

Definition Konfiguration

Konfiguration einer TM M hat Form uqv , wobei

- $u \in \Gamma^*$ Bandbeschriftung links des Kopfes,
- $q \in Q$ der momentane Zustand,
- $v \in \Gamma^*$ Bandbeschriftung ab (inkl.) Kopf nach rechts
- $uv \in \{\triangleright\} \cdot (\Gamma \setminus \{\triangleright\})^*$

Die *Länge* $|\alpha|$ einer Konfiguration $\alpha = uqv$ ist $|uv|$.

Turingmaschinen

Mehr Begriffe:

- Für Eingabe $w \in \Sigma^*$ ist $\triangleright q_0 w$ die *Startkonfiguration*.
- β ist *Folgekonfiguration* von α ,
wenn β aus α in **einem** Schritt hervorgeht.
Wir schreiben $\alpha \vdash_M \beta$ (formale Def. als Übung).
Folgekonfigurationen von DTMs sind eindeutig!
- Konfiguration uqv ist *akzeptierend*, wenn $q = q_{\text{acc}}$,
und *verwerfend*, wenn $q = q_{\text{rej}}$.
- *Berechnung* auf Eingabe w ist Folge von Konfigurationen $\alpha_0, \alpha_1, \dots$
(endl. oder unendl.) mit α_0 Startkonfig. für w und $\alpha_i \vdash_M \alpha_{i+1}$ für $i \geq 0$.
- Endl. Berechnung $\alpha_0, \dots, \alpha_k$ ist *akzeptierend*, wenn α_k akzeptierend,
und *verwerfend*, wenn α_k verwerfend.

T2.5

Turingmaschinen

Mehr Begriffe:

- DTM *akzeptiert* Eingabe w wenn die (maximale) Berechnung auf w endlich und akzeptierend ist, sonst *verwirft* M die Eingabe w
- Von TM M *erkannte* Sprache ist

$$L(M) := \{w \in \Sigma^* \mid M \text{ akzeptiert } w\}$$

Anm.: Es gibt zwei Wege, auf denen DTM Eingabe verwerfen kann:

- Berechnung endet in verwerfendem Zustand
- Berechnung ist unendlich (TM terminiert nicht)

Kapitel 2

NEXT



2.1 Turingmaschinen

2.2 Probleme lösen mittels Turingmaschinen

2.3 Mehrband-Turingmaschinen

2.4 Zeit-Komplexitätsklassen

2.5 Universelle Turingmaschinen

Entscheidungsprobleme

Eingabe für TM ist (endl.) Wort aus Σ^* .

Andere Eingaben (Graphen, Zahlen, etc) müssen kodiert werden.

Definition Problem

(*Entscheidungs*)problem ist Teilmenge $L \subseteq \Sigma^*$, für ein endliches Alphabet Σ .

Idee: das Problem ist die Menge aller „Ja-Instanzen“, z. B.:

- GAP ist die Menge aller Tripel (G, v, v') mit $G = (V, E)$ Graph und $v, v' \in V$ (geeignet kodiert), so dass v' von v erreichbar in G
- CLIQUE ist die Menge aller Paare (G, k) so dass G eine k -Clique hat

Wir geben die Kodierung in der Regel nicht explizit an.

Entscheidungsprobleme

Wir setzen „Turingmaschine“ mit „Algorithmus“ gleich

Wir interessieren uns für Algorithmen, die auf **jeder** Eingabe **anhalt**en (denn nur solche Algorithmen **lö**sen ein Entscheidungsproblem).

Definition Entscheidungsverfahren

TM M *entscheidet* Problem L : M terminiert auf jeder Eingabe und $L(M) = L$. Dann ist M *Entscheidungsverfahren* für L .

Also: wenn M Entscheidungsverfahren für L ist, dann

- akzeptiert M jedes $w \in \Sigma$ mit $w \in L$ durch halten in q_{acc} ;
- verwirft M jedes $w \in \Sigma$ mit $w \notin L$ durch halten in q_{rej} .

2.1 Turingmaschinen

2.2 Probleme lösen mittels Turingmaschinen

NEXT



2.3 Mehrband-Turingmaschinen

2.4 Zeit-Komplexitätsklassen

2.5 Universelle Turingmaschinen

Mehrband-TM

(Deterministische) Mehrband-TM hat mehrere Bänder (endlich viele):

- wie Einband-TM ist sie zu jedem Zeitpunkt in einem Zustand (**nicht** ein Zustand pro Band!)
- Es gibt einen Schreib-/Lesekopf pro Band
- in einem Schritt werden alle Bänder gleichzeitig gelesen und geschrieben
- Die Köpfe bewegen sich unabhängig voneinander (z.B.: einer nach links, ein anderer nach rechts)
- Die Eingabe ist auf dem ersten Band (*Eingabeband*); alle anderen Bänder sind initial mit $\triangleright \perp \perp \perp \dots$ beschriftet

T2.6

Mehrband-TM

Formal, bei k Bändern:

- Übergangsfunktion hat jetzt die Form

$$((Q \setminus \{q_{\text{acc}}, q_{\text{rej}}\}) \times \Gamma^k) \rightarrow (Q \times \Gamma^k \times \{L, R, N\}^k)$$

- bei $\delta(q, a_1, \dots, a_k) = (q', a'_1, \dots, a'_k, B_1, \dots, B_k)$ ist
 a_i Symbol, das auf i -tem Band gelesen wird,
 a'_i Symbol, das auf i -tem Band geschrieben wird,
 B_i Bewegung des i -ten Kopfes
- Konfiguration hat Form $(q, u_1, v_1, \dots, u_k, v_k)$, mit
 u_i Beschriftung Band i links des Kopfes,
 v_i Beschriftung i -tes Band ab (inkl.) Kopfposition
- Akzeptanz ist wie für Einband DTMs definiert.

T2.7

Kapitel 2

2.1 Turingmaschinen

2.2 Probleme lösen mittels Turingmaschinen

2.3 Mehrband-Turingmaschinen

NEXT



2.4 Zeit-Komplexitätsklassen

2.5 Universelle Turingmaschinen

Komplexitätsklassen

Komplexitätsklasse ist Klasse von Problemen, die mit bestimmter „Menge“ einer bestimmten Ressource gelöst werden können.

Je nach betrachteter Ressource:

„Menge“ z. B.:

- Zeitkomplexitätsklassen
- Platzkomplexitätsklassen
- polynomiell viel
- exponentiell viel

Wesentliche Fragestellung der Komplexitätstheorie:

Wie ist der Zusammenhang zwischen verschiedenen K.-Klassen?

Zeitkomplexitätsklassen

Definition zeitbeschränkte Funktion, DTime

Für DTM M und $w \in \Sigma^*$ schreiben wir $\text{time}_M(w) = n$, wenn M auf w nach n Schritten hält.

Sei $t : \mathbb{N} \rightarrow \mathbb{N}$ monoton wachsende Funktion mit $t(n) \geq n$.

M ist *t -zeitbeschränkt*, wenn $\text{time}_M(w) \leq t(n)$ für alle w der Länge n .

Wir definieren *Zeitkomplexitätsklasse* abhängig von Bandzahl:

$$\text{DTime}_k(t) := \{L \subseteq \Sigma^* \mid \text{es gibt } \mathcal{O}(t)\text{-zeitbeschränkte } k\text{-Band-DTM } M \text{ mit } L(M) = L\}$$

Zum Beispiel:

$\text{DTime}_1(n^2)$: Menge der Probleme, die in quadratischer Zeit von 1-Band-TM entschieden werden können.

T2.8

Relevante Funktionen

Wichtige Funktionen t zur Definition von Zeitkomplexitätsklassen z.B.:

- n (linear), n^2 (quadratisch), n^3 (kubisch) und andere Polynome
- 2^n , 2^{n^2} , 2^{n^3} und andere Exponentialfunktionen
- $n^{\log(n)}$, $2^{\log(n)^2}$ und andere quasipolynomielle Funktionen
- $2^{\sqrt{n}}$ und andere subexponentielle Funktionen

Um einen Eindruck von deren Wachstum zu bekommen:

Wolfram Alpha is your friend. :)

Mehrband TMs

Viele Bänder vs. wenige:

Bandreduktion / Satz von Hennie und Stearns (1966)

Für alle $k \geq 1$ und t gilt:

1. $\text{DTime}_k(t) \subseteq \text{DTime}_1(t^2)$
2. Wenn es $\varepsilon > 0$ gibt mit $t(n) \geq (1 + \varepsilon)n$,
dann gilt $\text{DTime}_k(t) \subseteq \text{DTime}_2(t \cdot \log(t))$.

T2.9

Überblick Beweis von 1.:

- Repräsentiere k Bänder als $2k$ „Spuren“ auf einem Band:
 k Spuren für Bandbeschriftung, k Spuren für Kopfmarker
- Simuliere jeden Schritt der k -Band TM
durch zweimaliges Ablaufen des Bandes.

Daher ist folgende Definition natürlich:

Definition DTime

Für $t : \mathbb{N} \rightarrow \mathbb{N}$ monoton wachsend mit $t(n) \geq n$ definieren wir:

$$\text{DTime}(t) := \bigcup_{k \geq 1} \text{DTime}_k(t)$$

Im Folgenden

- ... sind alle weiteren deterministischen Zeitkomplexitätsklassen mittels DTime definiert
- ... verwenden wir also **beliebige (endliche) Bandzahl** je nach Bedarf

Kapitel 2

2.1 Turingmaschinen

2.2 Probleme lösen mittels Turingmaschinen

2.3 Mehrband-Turingmaschinen

2.4 Zeit-Komplexitätsklassen

NEXT



2.5 Universelle Turingmaschinen

Universelle Turingmaschinen

Wir werden von Zeit zu Zeit **universelle Turingmaschinen** verwenden, die **alle anderen DTMs** simulieren können.

Sie sind sozusagen **Interpreter** für DTMs, implementiert als DTM.

Zunächst: jede DTM lässt sich als ein Wort kodieren.

Einige vereinfachende Annahmen (ohne wesentliche Einschränkung):

- alle DTMs verwenden das Eingabealphabet $\Sigma = \{a, b\}$
- alle DTMs verwenden nur Symbole aus dem Alphabet $\{a_1, a_2, a_3, \dots\}$, wobei $a_1 = a$, $a_2 = b$, $a_3 = \triangleright$ und $a_4 = \perp$
- alle DTMs verwenden nur Zustände aus der Menge $\{q_1, q_2, q_3, \dots\}$, wobei q_1 immer der Startzustand ist, $q_2 = q_{\text{acc}}$ und $q_3 = q_{\text{rej}}$

Kodierung von DTMs

DTM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$ dargestellt als Wort:

- Transition $t = \begin{matrix} L \\ (q_i, a_j) \mapsto (q_{i'}, a_{j'}, R) \\ N \end{matrix}$

dargestellt als

$$\text{code}(t) = \begin{matrix} a \\ a^i b a^j b a^{i'} b a^{j'} b \\ aa \quad bb \\ aaa \end{matrix}$$

- Besteht δ aus Transitionen t_1, \dots, t_k so wird M dargestellt als:

$$\text{code}(M) = \text{code}(t_1) \cdots \text{code}(t_n) b$$

- Konfiguration $K = a_{i_1} \cdots a_{i_k} q_j a_{i_{k+1}} \cdots a_{i_\ell}$ darstellbar als

$$\text{code}(K) = a^{i_1} b \cdots b a^{i_k} b a^j b b a^{i_{k+1}} b \cdots b a^{i_\ell}$$

Universelle Turingmaschinen

Theorem

Es gibt eine *universelle DTM* U , d. h. U erfüllt folgende Eigenschaft.

Für alle DTMs M und Eingaben w für M gilt:

$$U \text{ akzeptiert } \text{code}(M)w \quad \text{gdw.} \quad M \text{ akzeptiert } w$$

Beweisidee: (siehe auch Theor. Inf. 2)

- Erzeuge zunächst $\text{code}(M)\text{code}(q_0w)$.
- Wenn M bei Eingabe w die Berechnung $q_0w = K_1 \vdash_M K_2 \vdash_M \dots$ ausführt, erzeuge sukzessive

$$\text{code}(M)\text{code}(K_1), \text{code}(M)\text{code}(K_2), \text{code}(M)\text{code}(K_3), \dots$$

- Jeder einzelne Schritt wird durch mehrmaliges Hin- und Herlaufen implementiert.

Übersicht Vorlesung

Kapitel 1: Einführung

Kapitel 2: Turingmaschinen

NEXT



Kapitel 3: P vs. NP

Kapitel 4: Mehr Ressourcen, mehr Möglichkeiten?

Kapitel 5: Platzkomplexität

Kapitel 6: Schaltkreise

Kapitel 7: Orakel