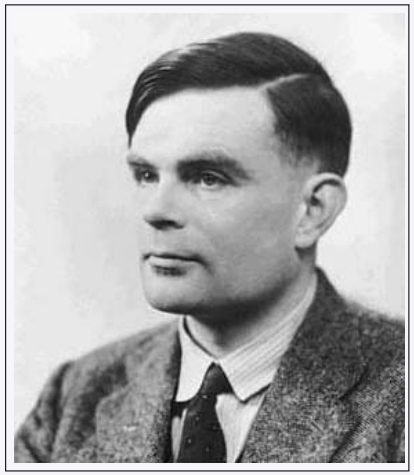
Turing: prinzipielle Berechenbarkeit → Abschnitt 4.2



Alan Turing (1912 – 1954)

Pionier der modernen Theorie der Berechenbarkeit

prinzipielle Grenzen und Möglichkeiten

Turingmaschinen: DTM → Abschnitt 4.2

DTM = DFA + unbeschränkter Lese/Schreibzugriff

Eingabe-/Arbeitsspeicher: unbeschränkte Folge von Zellen als "Band" mit Lese/Schreibkopf

Konfiguration bestimmt durch
- Zustand ($q \in Q$)
- Position auf dem Band
- Bandbeschriftunng

Übergang in Nachfolgekonfiguration abhängig von
- Zustand
- aktuell gelesenem Bandsymbol

Übergang resultiert in
- Zustandswechsel
- Schreiben
- Kopfbewegung ($<, \circ, >$)

DTM $\mathcal{M} = (\Sigma, Q, q_0, \delta, q^+, q^-)$

Q Zustandsmenge

$q_0 \in Q$ Anfangszustand

$q^+/q^- \in Q$ akzeptierender/verwerfender Endzustand, $q^- \neq q^+$

δ Übergangsfunktion

$$\delta\colon Q \times (\Sigma \cup \{\Box\}) \to (\Sigma \cup \{\Box\}) \times \{<, \circ, >\} \times Q$$

Konfigurationen:
$C = (\alpha, q, x, \beta) \in (\Sigma \cup \{\Box\})^* \times Q \times (\Sigma \cup \{\Box\}) \times (\Sigma \cup \{\Box\})^*$

α: Bandinhalt links vom Kopf

x: Bandinhalt in Kopfposition

β: Bandinhalt rechts vom Kopf

q: aktueller Zustand

Startkonfiguration auf Eingabe w: $C_0[w] := (\varepsilon, q_0, \Box, w)$

Nachfolgekonfiguration: $C \longmapsto C'$ gemäß δ ...

Endkonfigurationen: $q \in \{q^+, q^-\}$, akzeptierend/verwerfend

DTM: Akzeptieren und Entscheiden → Abschnitt 4.3

von DTM $\mathcal{M}$ akzeptierte Sprache

$L(\mathcal{M}) = \{w \in \Sigma^*\colon \text{ M akzeptiert } w\ \} = \{w \in \Sigma^*\colon w \xrightarrow{\mathcal{M}} q^+\}$

Entscheidung (des Wortproblems) von L

$\mathcal{M}$ *entscheidet* L falls für alle $w \in \Sigma^*$:

$$w \xrightarrow{\mathcal{M}} \begin{cases} q^+ & \text{für } w \in L \\ q^- & \text{für } w \notin L \end{cases} \qquad \text{definit!}$$

L entscheidbar (rekursiv):
L von einer DTM entschieden

L aufzählbar (rekursiv aufzählbar, semi-entscheidbar):
L von einer DTM akzeptiert

Beispiel: DTM für Palindrom

δ	$\square$	0	1
q_0	$(\square, >, q^?)$		
$q^?$	$(\square, \circ, q^+)$	$(\square, >, q^{\rightarrow 0})$	$(\square, >, q^{\rightarrow 1})$
$q^{\rightarrow 0}$	$(\square, <, q^{\leftarrow 0})$	$(0, >, q^{\rightarrow 0})$	$(1, >, q^{\rightarrow 0})$
$q^{\rightarrow 1}$	$(\square, <, q^{\leftarrow 1})$	$(0, >, q^{\rightarrow 1})$	$(1, >, q^{\rightarrow 1})$
$q^{\leftarrow 0}$	$(\square, \circ, q^+)$	$(\square, <, q^{\leftarrow})$	$(\square, \circ, q^-)$
$q^{\leftarrow 1}$	$(\square, \circ, q^+)$	$(\square, \circ, q^-)$	$(\square, <, q^{\leftarrow})$
$q^{\leftarrow}$	$(\square, >, q^?)$	$(0, <, q^{\leftarrow})$	$(1, <, q^{\leftarrow})$

intendierte Rolle der Zustände:

q_0 :	Startzustand	$q^?$:	Anfang abfragen
$q^{\rightarrow 0}$:	zum Ende, merke 0	$q^{\leftarrow 0}$:	vergleiche Ende mit 0
$q^{\rightarrow 1}$:	zum Ende, merke 1	$q^{\leftarrow 1}$:	vergleiche Ende mit 1
$q^{\leftarrow}$:	zum Anfang	q^+/q^-:	akzeptiere/verwerfe

Church-Turing These

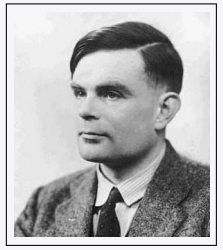
algorithmische Entscheidbarkeit = Turing-Entscheidbarkeit
algorithmische Erzeugbarkeit = Turing-Aufzählbarkeit
Berechenbarkeit = Turing-Berechenbarkeit

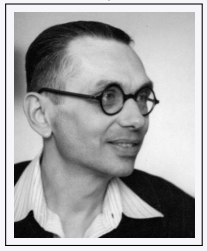
Belege:
- Erfahrung: alle akzeptierten Algorithmen lassen sich im Prinzip mit DTM simulieren
- Robustheit des TM Modells
- bewiesene Äquivalenz mit ganz unterschiedlichen alternativen Charakterisierungen

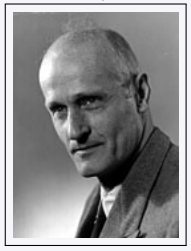
wichtig: idealisiertes Konzept von *prinzipieller* Machbarkeit im Ggs. zu *praktischer* Machbarkeit

einige Väter der Berechenbarkeitstheorie


Church (1903–1995)


Turing (1912–1954)


Gödel (1906–1978)


Kleene (1909–1994)

Erinnerung: Wortprobleme als Entscheidungsprobleme

Wortproblem zu $L \subseteq \Sigma^*$:

Eingabe: $w \in \Sigma^*$
Entscheide, ob $w \in L$

Lösung durch Algorithmus A mit

$$w \xrightarrow{A} \begin{cases} \text{“ja”} & \text{falls } w \in L & \textit{akzeptieren} \\ \text{“nein”} & \text{falls } w \notin L & \textit{verwerfen} \end{cases}$$

definit!

im Kontrast zu *(einseitigem) Akzeptieren* wie bei NFA/PDA oder *Erzeugen/Ableiten* wie bei Grammatik

- Zugehörigkeit zu L erkennen $\neq$ Zugehörigkeit entscheiden
- vgl. Problem des Komplement-Abschluss

Akzeptieren	**Entscheiden**
NFA $\mathcal{A}$ (nicht-deterministisch!): $w \in L$ gdw. eine akzeptierende Berechnung von $\mathcal{A}$ auf w *existiert* **existiert**	DFA $\mathcal{A}$: $w \xrightarrow{\mathcal{A}} \begin{cases} + & \text{für } w \in L \\ - & \text{für } w \notin L \end{cases}$
Grammatik G: $w \in L$ gdw. Ableitung von w in G *existiert* **existiert**	**?**
PDA $\mathcal{P}$ (nicht-deterministisch!): $w \in L$ gdw. eine akzeptierende Berechnung von $\mathcal{P}$ auf w *existiert* **existiert**	CYK-Algorithmus $w \xrightarrow{CYK} \begin{cases} + & \text{für } w \in L \\ - & \text{für } w \notin L \end{cases}$

vgl. auch **NP/P**

Entscheidbarkeit und Aufzählbarkeit

- L entscheidbar $\Leftrightarrow$ (L und $\overline{L} = \Sigma^* \setminus L$ aufzählbar)
- die Klasse der entscheidbaren Sprachen ist abgeschlossen unter
  Durchschnitt, Vereinigung *und* Komplement,
  Konkatenation, Stern, ...
- die Klasse der aufzählbaren Sprachen ist abgeschlossen unter
  Durchschnitt, Vereinigung (und *nicht* unter Komplement),
  Konkatenation, Stern, ...

Unentscheidbarkeit des Halteproblems

Satz 4.3.4

ein konkretes beweisbar (Turing-)unentscheidbares Problem

arbeite mit Kodierung $\mathcal{M} \mapsto \langle \mathcal{M} \rangle \in \Sigma^*$

Halteproblem: Eingabe $\langle \mathcal{M} \rangle$,
Entscheide, ob $\mathcal{M}$ auf Eingabe $\langle \mathcal{M} \rangle$ terminiert

$$\mathbf{H} = \{\langle \mathcal{M} \rangle \in \mathbf{\Sigma^*}\colon \langle \mathcal{M} \rangle \xrightarrow{\mathcal{M}} \mathrm{STOP}\}$$

Satz 4.3.4: H ist nicht entscheidbar
H aufzählbar, $\overline{H}$ nicht aufzählbar

Beweis: Unmöglichkeitsbeweis durch "*Diagonalisierung*" (!)

Konsequenzen:

Nachweis der prinzipiellen algorithmischen Unlösbarkeit vieler interessanter Entscheidungs- und Berechnungsprobleme

Trennung von Typ 1, Typ 0, und beliebigen Sprachen (s.u.)

Halteproblem: Diagonalisierung

Satz 4.3.4

$$\mathbf{H} = \{\langle \mathcal{M} \rangle \in \mathbf{\Sigma^*}\colon \langle \mathcal{M} \rangle \xrightarrow{\mathcal{M}} \mathrm{STOP}\}$$

Annahme, $\mathcal{M}_0$ entscheide **H**:

für alle $\mathcal{M}$: $\langle \mathcal{M} \rangle \xrightarrow{\mathcal{M}_0} \begin{cases} q^+ & \text{falls } \langle \mathcal{M} \rangle \xrightarrow{\mathcal{M}} \mathrm{STOP} \\ q^- & \text{falls } \langle \mathcal{M} \rangle \xrightarrow{\mathcal{M}} \infty \end{cases}$

$\mathcal{M}_1$ aus $\mathcal{M}_0$: nicht-terminierende Schleife statt q^+

dann: $\langle \mathcal{M}_1 \rangle \xrightarrow{\mathcal{M}_1} \infty \Leftrightarrow \langle \mathcal{M}_1 \rangle \xrightarrow{\mathcal{M}_1} \mathrm{STOP}$ **Widerspruch!**

zurück zur Chomsky-Hierarchie → Abschnitt 4.4

Typ 0 = Aufzählbarkeit (Satz 3.4.17)

Für $L \subseteq \Sigma^*$ sind äquivalent:

(i) L von Grammatik erzeugt (Typ 0): $L = L(G)$.
(ii) L von einer DTM $\mathcal{M}$ akzeptiert: $L = L(\mathcal{M})$.

Typ 1 Sprachen sind entscheidbar (Satz 3.4.18)

Jede kontextsensitive Sprache (Typ 1)
hat ein entscheidbares Wortproblem.

Bemerkung: nicht jede entscheidbare Sprache ist Typ 1,
aber es gibt ein genau entsprechendes NTM-Niveau

Chomsky-Hierarchie

Typ 3 $\subsetneq$ Typ 2 $\subsetneq$ Typ 1 $\subsetneq$ Typ 0 $\subsetneq$ bel.

- **Trennung durch Pumping Lemmata**
- **trennende Beispiele H und $\overline{\text{H}}$**

Abschlusseigenschaften

Typ	abgeschlossen unter $\cup$	$\cap$	$^-$	$\cdot$	$*$
3	+	+	+	+	+
2	+	−	−	+	+
1	+	+	+	+	+
0	+	+	−	+	+
bel. Σ-Sprachen	+	+	+	+	+

das Wichtigste aus Kapitel 4

Berechnungsmodelle

PDA und kontextfreie Sprachen

Turingmaschinen als **universelles Berechnungsmodell**

Aufzählbarkeit und **Entscheidbarkeit**

Striktheit der Chomsky-Hierarchie

Exkurs: zwei algorithmische Anwendungsideen

Textsuche (string matching) → KMP Algorithmus (5.1)

gesucht: guter Algorithmus für einfache Textsuche:

Eingabe: Suchwort $w \in \Sigma^*$ $|w| = m$
Text $t \in \Sigma^*$ $|t| = n$

Ausgabe: alle Stellen i, $1 \leqslant i \leqslant n - m + 1$
mit $t_{i,i+m-1} = w$

Verifikation mit Automaten: model checking (5.1)

gesucht: Entscheidungsverfahren für das Überprüfen
von Systemspezifikationen:

Eingabe: Eigenschaft E (Spezifikation)
und Systementwurf $\mathcal{S}$

Ausgabe: $\begin{cases} \text{“ja”} & \text{falls } \mathcal{S} \models E \\ \text{“nein” (+Information)} & \text{falls } \mathcal{S} \not\models E \end{cases}$

Textsuche naiv

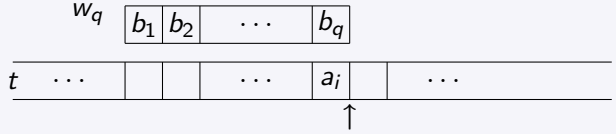
Suchwort $w = b_1 \dots b_m$:
längs Text $t = a_1 \dots a_n$ ermittle in jeder Position i,
ob $t_{i,i+m-1} = w$

bis zu $\leqslant m(n-m)$ Vergleiche/Schritte

Textsuche verbessert DFA + dynamisches Programmieren

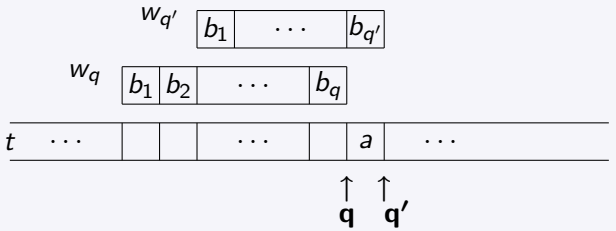
Suchwort $w = b_1 \dots b_m$:
längs Text $t = a_1 \dots a_n$ ermittle in jeder Position i

den *maximalen Präfix* $w_q := w_{1,q} = b_1 \dots b_q$ von w,
der an dieser Stelle passt ($t_{i-q+1,i} = b_1 \dots b_q$)



Vorrücken um einen Buchstaben im Text: simuliere passenden DFA

Textsuche: DFA-Simulation



simuliere den DFA $\mathcal{A}_w = (\Sigma, \{0, \dots, m\}, 0, \delta, \{m\})$

mit $\delta(q, a) = \max\{k\colon w_k \text{ Suffix von } w_q a\}$

Knuth, Morris, Pratt

Berechnung von δ-Werten aus (vorab) tabellierten Daten
zu Selbstüberlappungen in w
$\rightarrow$ Gesamtlaufzeit linear in $n + m$ statt in $n \cdot m$

model checking (grobe Idee)

Systementwurf $\mathcal{S}$ (Transitionssystem)	$\longrightarrow$	Sprache $L_\mathcal{S}$: die Zustandsfolgen in Läufen von $\mathcal{S}$
Spezifikation: Eigenschaft E, von allen Läufen gefordert	$\longrightarrow$	Sprache L_E: die Zustandsfolgen mit Eigenschaft E

Reduktion auf Leerheitsproblem

$\mathcal{S} \models E$ gdw. $L_\mathcal{S} \subseteq L_E$ gdw. $L_\mathcal{S} \cap \overline{L_E} = \emptyset$
$\mathcal{S}$ erfüllt E **Leerheitsproblem**

Extra: falls $\mathcal{S} \not\models E$, finde Zustandsfolge
$w \in L_\mathcal{S} \setminus L_E$ als Hinweis auf Ursache

Beispiel zur Übung

(1) gesucht: minimaler DFA für L(a(bc)* + (abc)*)
$\alpha = a(bc)^* + (abc)^* \in \mathrm{REG}(\Sigma)$, $\Sigma = \{a, b, c\}$
$\alpha = \alpha_1 + \alpha_2$, $\alpha_1 = a(bc)^*$, $\alpha_2 = (abc)^*$
$L(\alpha) = L(\alpha_1) \cup L(\alpha_2)$

- NFA $\mathcal{A}_i$ für $L(\alpha_i)$, $i = 1, 2$
- DFA $\mathcal{A}_i^{\text{det}}$ für $L(\alpha_i)$, $i = 1, 2$
- Produkt DFA für $L(\alpha)$
- NFA $\mathcal{A}$ für $L(\alpha)$
- DFA $\mathcal{A}^{\text{det}}$ für $L(\alpha)$
- Minimierung oder Nachweis der Minimalität

Beispiel zur Übung

(2) NFA/DFA für binäre Addition

für $u_1 = b_{1,1}\ldots b_{1,n}$ $u_2 = b_{2,1}\ldots b_{2,n}$ $u_3 = b_{3,1}\ldots b_{3,n}$ teste ob $u_3 = u_1 + u_2$ (als Binärzahlen)

$$\begin{array}{rl} & b_{1,1}\ldots b_{1,n} \\ + & b_{2,1}\ldots b_{2,n} \\ \hline & b_{3,1}\ldots b_{3,n} \end{array}$$

Alphabet (für Spalten in Addition):

$$\Sigma = (\{0,1\})^3 = \left\{ \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \ldots, \begin{bmatrix}1\\1\\1\end{bmatrix} \right\}$$

$$(u_1,u_2,u_3) \longmapsto w(u_1,u_2,u_3) = \begin{bmatrix}u_{1,1}\\u_{2,1}\\u_{3,1}\end{bmatrix} \ldots \begin{bmatrix}u_{1,n}\\u_{2,n}\\u_{3,n}\end{bmatrix} \in \Sigma^*$$

$L = \{w(u_1,u_2,u_3)\colon u_3 = u_1 + u_2\}$

gesucht: NFA/DFA für L

Beispiel zur Übung

(3) Abschlusseigenschaften

Sei L_0 regulär, L_1 kontextfrei.

- folgt, dass $L_0 \cup L_1$ kontextfrei ist?
- folgt, dass $L_0 \cap L_1$ kontextfrei ist?
- folgt, dass $L_0 \setminus L_1$ kontextfrei ist?
- folgt, dass $L_1 \setminus L_0$ kontextfrei ist?

- ist das Komplement einer Typ 0 Sprache stets Typ 0?
- ist Typ 0 abgeschlossen unter Durchschnitt/Vereinigung?

Beispiel zur Übung

(4) Automatenkonstruktion: rückwärtslesen

die Klasse der regulären Sprachen ist abgeschlossen unter Wortumkehr $L \mapsto L^{-1}$ (Begründungen?)

- wie gewinnt man aus einem DFA/NFA für L einen DFA/NFA für L^{-1} ?

Beispiel zur Übung

(5) Pumping Lemmata

$\Sigma = \{a, b\}$.

- $L = \{ww^{-1}\colon w \in \Sigma^*\}$.
- $L = \{ww\colon w \in \Sigma^*\}$.

allg. Struktur (PL): **wenn** L vom Typ 3/2 ist,
dann **existiert n** $\in \mathbb{N}$, so dass
für alle w $\in$ **L mit** $|\mathbf{w}| \geqslant \mathbf{n}$ gilt:
w lässt sich so zerlegen, dass ...

negative Anwendung:
wenn für alle n $\in \mathbb{N}$
existiert w $\in$ **L mit** $|\mathbf{w}| \geqslant \mathbf{n}$,
so dass **keine** Zerlegung von w ...
dann ist L nicht vom Typ 3/2