4 Turing-Maschinen

Frage: Was kann von einem rechnenden Menschen (ggf. mit Hilfsmitteln) berechnet werden?

In der Arbeit „*On computable numbers and an application to the Entscheidungsproblem*“ von 1936 geht Alan Turing (1912-1954) davon aus, daß ein „Rechnender“

- sich Notizen auf beliebig viel Papier,
- sich endlich viel merken und
- nur einen begrenzten Teil der Notizen gleichzeitig überblicken und sich nur blätternd durch die Notizen bewegen kann.
- Außerdem arbeitet er mechanisch.

Aus diesen Überlegungen heraus definierte er die heute so genannte „Turing-Maschine“.

Definition. 1. Eine *nichtdeterministische Turing-Maschine* (NTM) ist ein Tupel $M = (Q, \Sigma, \Gamma, \iota, \delta, \square, F)$, wobei

- Q eine endliche Menge von „Zuständen“,
- Σ das „Eingabealphabet“,
- Γ das „Arbeitsalphabet“ mit $\Sigma \cup \{\square\} \subseteq \Gamma$,
- $\iota \in Q$ der „Initialzustand“,
- $\delta\colon Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{-1, 0, 1\})$ die “Überführungsfunktion“,
- $\square \in \Gamma \setminus \Sigma$ das „Leerzeichen“ und
- $F \subseteq Q$ die Menge der „Endzustände“ sind.

2. Eine *(deterministische) Turing-Maschine* (TM) ist eine NTM mit $|\delta(q, a)| \leq 1$ für alle $q \in Q$ und $a \in \Gamma$.

3. Eine *Konfiguration* der NTM M ist ein Tripel (q, p, B) mit $q \in Q$, $p \in \mathbb{Z}$ und $B\colon \mathbb{Z} \to \Gamma$ mit $B(i) = \square$ für fast alle $i \in \mathbb{Z}$.

4. Für zwei Konfigurationen (q, p, B) und (q', p', B') setzen wir

$$(q, p, B) \vdash (q', p', B')\,,$$

wenn es $(q', b, x) \in \delta(q, B(p))$ gibt mit $p' = p + x$ und

$$B'(i) = \begin{cases} b & \text{falls } i = p \\ B(i) & \text{sonst.} \end{cases}$$

5. Für Konfigurationen K und K' schreiben wir

$$K \vdash^* K'\,,$$

wenn es $n \in \mathbb{N}$ und Konfigurationen $K_0, K_1, \cdots, K_n$ gibt mit

$$K = K_0 \vdash K_1 \vdash \cdots \vdash K_n = K'\,.$$

6. Die NTM M *akzeptiert* das Wort $w = a_0a_1a_2 \cdots a_{n-1} \in \Sigma^*$, wenn es eine Konfiguration (f, p, B) mit $f \in F$ gibt, so daß

$$\delta(f, B(p)) = \emptyset \text{ und } (\iota, 0, B_w) \vdash^* (f, p, B)\,,$$

wobei

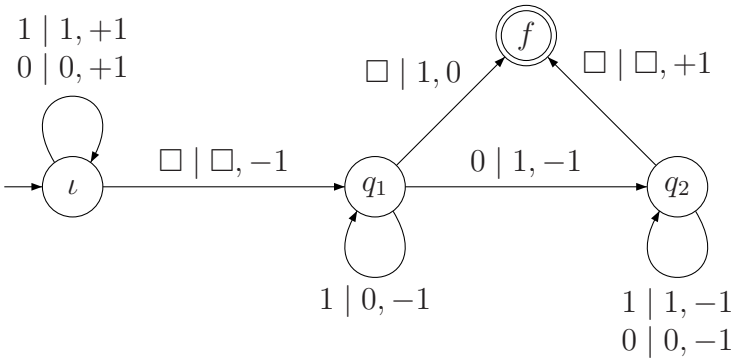
$$B_w(i) = \begin{cases} a_i & \text{falls } 0 \leq i < n \\ \square & \text{sonst} \end{cases}$$

für alle $i \in \mathbb{Z}$ gilt. Sei $L(M) \subseteq \Sigma^*$ die Menge der von M akzeptierten Wörter.

Beispiel. TM, die alle Wörter über $\{0,1\}$ akzeptiert und dabei die Eingabe (als Binärzahl) um 1 erhöht:

- $Q = \{\iota, q_0, q_1, f\}$
- $\Gamma = \{0, 1, \square\}$
- $\Sigma = \{0, 1\}$
- $F = \{f\}$

Die Überführungsfunktion δ ist durch das folgenden Bild gegeben:



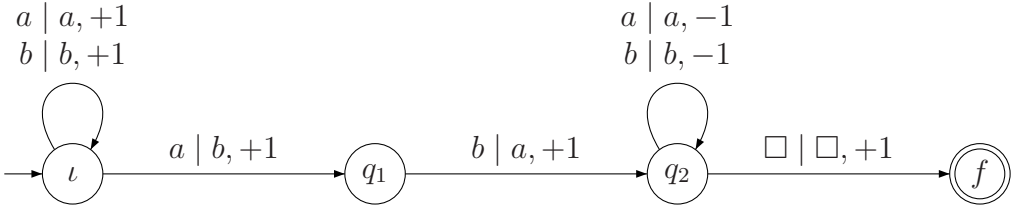
z.B. $\delta(\iota, 1) = \{(\iota, 1, +1)\}$, d.h. bei Lesen von 1 im Zustand ι bleibt die TM im Zustand ι, läßt die 1 auf dem Band stehen und bewegt den Kopf eine Stelle nach rechts.

Beispielrechnung:

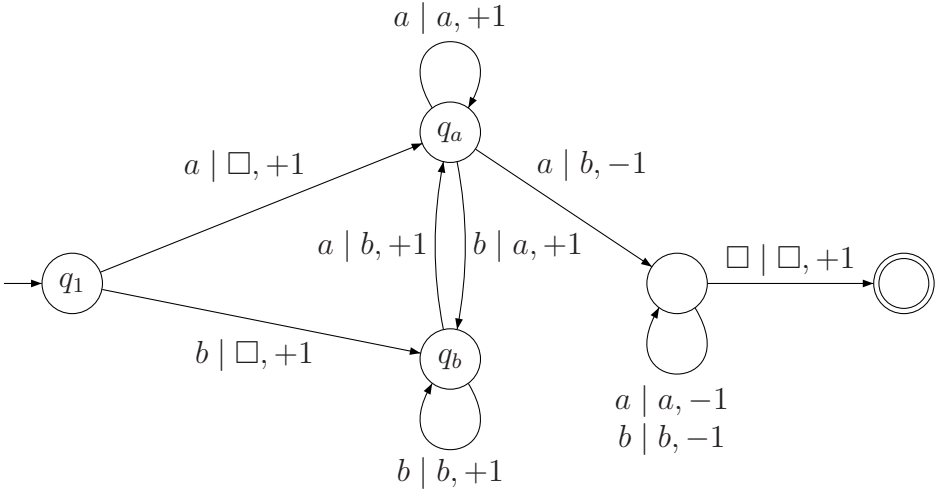
$$
\begin{array}{lll}
\overset{\iota}{\cdots\square\square\ 1\ 0011\square\square\cdots} & \vdash & \overset{\iota}{\cdots\square\square 1\ 0\ 011\square\square\cdots} \\
\vdash & \overset{q_1}{\cdots\square\square 1001\ \ 1\ \ \square\square\cdots} & \vdash \quad \overset{q_1}{\cdots\square\square 100\ \ 1\ \ 0\square\square\cdots} \\
\vdash & \overset{q_2}{\cdots\square\square 1\ \ 0\ \ 100\square\square\cdots} & \vdash \quad \overset{q_2}{\cdots\square\square\ \ 1\ \ 0100\square\square\cdots} \\
\vdash & \overset{f}{\cdots\square\square\ 1\ 0100\square\square\cdots} &
\end{array}
$$

$$
\vdash^* \quad \overset{\iota}{\cdots\square\square 10011\ \square\ \square\cdots}
\vdash \quad \overset{q_1}{\cdots\square\square 10\ \ 0\ \ 00\square\square\cdots}
\vdash \quad \overset{q_2}{\cdots\square\ \square\ 10100\square\square\cdots}
$$

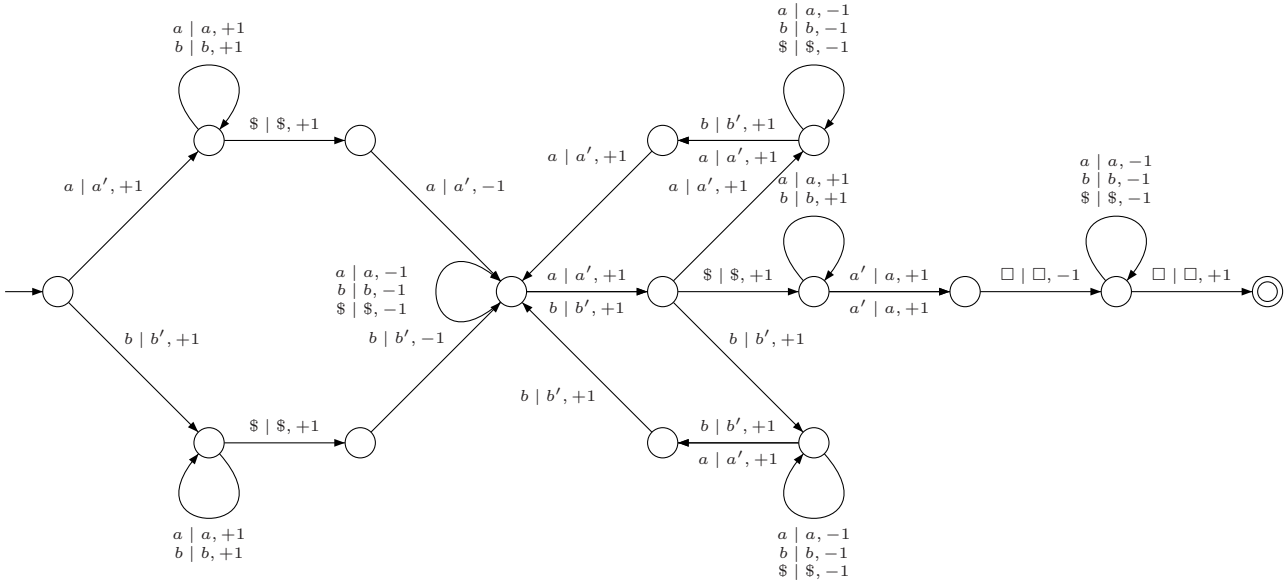
Beispiel. NTM mit $L(M) = \{a, b\}^*\{ab\}\{a, b\}^*$, die beim Akzeptieren ein Vorkommen von ab durch ba ersetzt:



Beispiel. NTM mit $L(M) = \{a, b\}^*\{aa\}\{a, b\}^*$, die beim Akzeptieren ein Vorkommen von aa durch b ersetzt:



Beispiel. Die folgende TM akzeptiert die Menge der Wörter $w\$w$ mit $w \in \{a, b\}^*$. Wir haben gesehen, daß diese Sprache nicht kontextfrei ist.



Definition. Eine Sprache $L \subseteq \Sigma^*$ heißt *semi-entscheidbar*, wenn es eine det. TM M gibt mit $L = L(M)$. Sie heißt *entscheidbar*, wenn M zusätzlich so gewählt werden kann, daß sie bei jeder Eingabe irgendwann hält (d.h. für jedes Wort $w \in \Sigma^*$ existiert eine Konfiguration (q, p, B) mit $(\iota, 0, B_w) \vdash^* (q, p, B)$ und $\delta(q, B(p)) = \emptyset$).

Bemerkung. Es ist leicht zu sehen, daß jede reguläre Sprache entscheidbar ist (betrachte den DFA als TM, die niemals den Inhalt des Bandes ändert).

Etwas weniger einfach ist zu sehen, daß jede kontextfreie Sprache entscheidbar ist (verwende einen zusätzlichen Abschnitt des Bandes, um den CYK-Algorithmus auszuführen).

Man kann auch zeigen, daß jede NTM M in eine äquivalente TM umgewandelt werden kann (die eine Breitensuche in der Menge der Konfigurationen von M durchführt und daher „sehr viel langsamer“ als M ist).

Church-Turing-These *Eine Sprache $L \subseteq \Sigma^*$ kann genau dann durch einen Algorithmus akzeptiert werden, wenn es eine Turing-Maschine M gibt mit $L(M) = L$.*

mögliches Killerargument gegen die These:

- ein Algorithmus, der einen Sprache akzeptiert, die von keiner Turing-Maschine akzeptiert wird.

allg. akzeptierte Argumente für die These:

- Turings Analyse des Rechnens eines Menschen.
- Man kann von vielen Sprachen zeigen, daß sie von einer TM akzeptiert werden.
- Verschiedenste alternative Berechnungsmodelle (z.B. Registermaschinen, Markov-Algorithmen und -Tabellen, Gödels rekursive Funktionen, Churchs λ-Kalkül, Programmiersprachen, ...) haben sich als nicht stärker als Turing-Maschinen erwiesen.
- Niemand hat in den vergangenen 80 Jahren eine algorithmisch erkennbare Sprache angegeben, die von keiner Turingmaschine akzeptiert wird.

  Die TM-Lösbarkeit ist daher nach aller Erfahrung eine notwendige Bedingung für die algorithmische Lösbarkeit eines Problems.

4.1 Unentscheidbare Probleme

Ziel: Viele (natürliche) Probleme sind nicht entscheidbar, aber wenigstens semi-entscheidbar.

Dazu nehmen wir in diesem Kapitel für alle Turingmaschinen $M = (Q, \Sigma, \Gamma, \iota, \delta, \square, F)$ an:

- $Q = \{1^n \mid 1 \leq n \leq N\}$
- $\Sigma = \{0, 1\}$
- $\Gamma = \{0, 1, 2, 3\}$, mit $3 = \square$
- $\iota = 1$, $F = \{1^n \mid f \leq n \leq N\}$

Dies ist keine (wesentliche) Einschränkung, denn jede Turing-Maschine kann in eine „äquivalente“ umgewandelt werden, die diese Eigenschaften hat.

Codes/Namen der Turing-Maschine M

1. Seien $1 \leq m, n \leq N$, $a, b \in \Gamma$ und $x \in \{-1, 0, 1\}$. Der Code von $(1^m, a, 1^n, b, x)$ ist das Wort
$$001^m01^{a+1}01^n01^{b+1}01^{2+x}\,.$$

2. jede Verkettung aller Codes von $(1^m, a, 1^n, b, x)$ mit $(1^n, b, x) \in \delta(1^m, a)$ ist ein Code von δ

3. jedes Wort $w0001^f$, wobei w ein Code von δ ist, ist ein Code von M. Sei $L_{TM} \subseteq \{0, 1\}^*$ die Menge aller Codes von TM. Ist $w \in L_{TM}$ ein Code der TM M, so setze $M_w = M$.

Bemerkung. Die Sprache L_{TM} ist entscheidbar, d.h. es gibt eine TM M_{TM}, die sich bei Eingabe von $w \in \{0, 1\}^*$ wie folgt verhält:

- Gilt $w \in L_{TM}$, so hält M_{TM} in einem akzeptierenden Zustand.
- Gilt $w \notin L_{TM}$, so hält M_{TM} in einem nicht-akzeptierenden Zustand.

Insbes. hält die TM M_{TM} bei jeder Eingabe.

Lemma 4.1. *Das spezielle Halteproblem*

$$K = \{w \in \{0,1\}^* \mid w \in L_{TM},\ M_w \textit{ hält bei Eingabe von } w\}$$

ist semi-entscheidbar.

BewIdee:
Wir werden eine TM S konstruieren, die sich bei Eingabe eines Wortes w wie folgt verhält:

- Gilt $w \in L_{TM}$ und hält die TM M_w bei Eingabe von w irgendwann, so akzeptiert S das Wort w (d.h. hält in einem akzeptierenden Zustand).
- Ansonsten hält S bei Eingabe von w nicht.

1. Vortest: die TM S verhält sich zunächst wie die TM M_{TM}. Wenn M_{TM} in einem nicht-akzeptierenden Zustand hält, so geht S in eine Endlosschleife über. Wenn M_{TM} in einem akzeptierenden Zustand hält, so rechnet S wie folgt weiter (auf dem Band steht weiterhin nur das Wort w).

2. Initialisierung: die TM S schreibt hinter die Eingabe w das Wort $0000100002w$
Idee: die simulierte TM M_w ist im Initialzustand 1^1 und auf ihrem Band steht das Wort w und der Kopf der simulierten TM steht auf der ersten Position von w

3. Simulieren: in jeder „Runde“ sucht die TM S im Wort w den Code $001^p01^{a+1}01^q01^{b+1}01^{2+x}$ einer Anweisung $(q, b, x) \in \delta(p, a)$, wobei p der augenblickliche Zustand der simulierten Maschine M_w und a das von der simulierten Maschine M_w augenblicklich gelesene Zeichen ist.
Dann ersetzt die TM S den Zustand p durch q (dabei muß u.U. viel hin-und-her geschoben werden), das gelesene Zeichen a durch b und bewegt den simulierten Kopf (d.h. die 2) um x.
Ist $p \in F$ und existiert kein Code der Form $001^p01^{a+1}01^q01^{b+1}01^{2+x}$, so wechselt die TM S in einen akzeptierenden Zustand und hält an. Ansonsten geht S in die nächste Runde.
□

Lemma 4.2. *Das spezielle Halteproblem K ist nicht entscheidbar.*

BewIdee:
Angenommen, E wäre eine TM, die K entscheidet, d.h. E hält bei jeder Eingabe und $L(E) = K$.
Wir konstruieren aus E eine neue TM M:

- Wenn E akzeptiert (d.h. in einem akzeptierenden Zustand anhält), so bewegt M ihren Kopf immer weiter nach rechts (insbes. hält M nicht an).
- Wenn E in einem nicht-akzeptierenden Zustand anhält, so geht M in einen Finalzustand und hält an.

Sei w ein Code von M, d.h. $w \in L_{TM}$ mit $M_w = M$. Dann gilt:

M hält bei Eingabe von w gdw. E akzeptiert w nicht
gdw. M_w hält bei Eingabe von w nicht,
im Widerspruch zu $M = M_w$. □

Folgerung. *Das spezielle Halteproblem K ist semi-entscheidbar, sein Komplement $\Sigma^* \setminus K$ ist nicht semi-entscheidbar.*

BewIdee:
Da K nach Lemma 4.1 semi-entscheidbar ist, existiert eine TM S mit $L(S) = K$.

Wäre $\Sigma^* \setminus K$ semi-entscheibar, so gäbe es eine TM $\overline{S}$ mit $L(\overline{S}) = \Sigma^* \setminus K$.

Dann könnte man eine TM M konstruieren, die S und $\overline{S}$ „parallel“ ausführt.

Diese TM M entscheidet K, was nach Lemma 4.2 nicht möglich ist. □

Diese Beweisidee kann man verallgemeinern zu einem Beweis von:

Satz. *Sei $A \subseteq \Sigma^*$. Dann sind äquivalent*

- *A ist entscheidbar.*
- *A und $\Sigma^* \setminus A$ sind beide semi-entscheidbar.*

Satz 4.3. *Das Halteproblem*

$$H = \{w\#x \mid w \in L_{TM}, M_w \textit{ hält bei Eingabe von } x\}$$

ist nicht entscheidbar.

BewIdee:
Angenommen, H wäre entscheidbar. Dann gäbe es eine TM M mit $L(M) = H$, die bei jeder Eingabe anhält.

Wir konstruieren hieraus eine neue TM M', die zunächst hinter ihre Eingabe w das Wort $\#w$ schreibt und dann die TM M startet.

Die TM M' hält bei jeder Eingabe.

Sei $w \in L_{TM}$. Dann gilt

$$\begin{array}{lll} w \in L(M') & \text{gdw.} & w\#w \in H \\ & \text{gdw.} & M_w \text{ hält bei Eingabe von } w \\ & \text{gdw.} & w \in K, \end{array}$$

d.h. $L(M') = K$ und M' hält bei jeder Eingabe, im Widerspruch zu Lemma 4.2. Also ist das Halteproblem H nicht entscheidbar. □

Der Beweis von Satz 4.3 verwendet die folgende Idee:

(1) Wir konstruieren eine TM F, die aus einer Eingabe w für K eine Eingabe $f(w) = w\#w$ für H berechnet und zeigen, daß $w \in K \iff f(w) \in H$ gilt.

Eine solche Funktion f heißt *Reduktion von K auf H*. Sie muß

- von einer TM berechnet werden können und
- $w \in K \iff f(w) \in H$ erfüllen.

Wir schreiben $K \leq H$ für „es gibt eine Reduktion von K auf H“.
Der Beweis geht dann wie folgt weiter:

(2) Angenommen, H wäre entscheidbar. Dann gäbe es eien TM M, die H entscheidet. Die TM M' führt dann die TM F u'nd M nacheinander aus und entscheidet K.

Da aber K nicht entscheidbar ist, kann eine solche TM M' nicht existieren. Also ist H nicht entscheidbar.

Allgemeiner erhalten wir

Satz 4.4. *Seien $A, B \subseteq \Sigma^*$ mit $A \leq B$.*

- *Ist B entscheidbar, so auch A (d.h. ist A unentscheidbar, so auch B).*
- *Ist B semi-entscheidbar, so auch A (d.h. ist A nicht semi-entscheidbar, so auch B).*

Hiermit kann man viele weitere Unentscheidbarkeiten zeigen:

Satz. *Die folgenden Probleme sind nicht entscheidbar:*

1. *Das* Halteproblem bei leerer Eingabe

$$H_0 = \{w \in \{0,1\}^* \mid w \in L_{TM} \Longrightarrow M_w \textit{ hält bei Eingabe von } \varepsilon\}$$

   (analog für jede andere Eingabe v)

2. *Das* Regularitätsproblem

$$\{w \in \{0,1\}^* \mid w \in L_{TM} \Longrightarrow L(M_w) \textit{ ist regulär}\}$$

allgemeiner gilt der ***Satz von Rice.***
Sei $\mathcal{P}$ eine Eigenschaft von Sprachen und seien N und P TM, so daß $L(P)$ $\mathcal{P}$ erfüllt und $L(N)$ nicht. Dann ist

$$\{w \in \{0,1\}^* \mid w \in L_{TM} \Longrightarrow L(M_w) \text{ erfüllt } \mathcal{P}\}$$

nicht entscheidbar.

3. *Das Disjunktheitsproblem, das Inklusionsproblem, das Äquivalenzproblem und das Regularitätsproblem für kontextfreie Sprachen sind ebenfalls nicht entscheidbar (unter jeder „sinnvollen" Kodierung von kontextfreien Grammatiken durch Wörter über $\{0,1\}$).*

4. *Die Menge der multivariaten Polynome $p(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ mit ganzzahligen Koeffizienten, die eine ganzzahlige Nullstelle besitzen.*

5. *Die Menge der SQL-Anfragen, die in jeder Datenbankinstanz die leere Antwort erzeugen.*

6. *Das Äquivalenzproblem für Programme in Ihrer Lieblingssprache (wobei wir einen unendlichen Speicher voraussetzen).*

Zusammenfassung

- Die Church-Turing These erlaubt es zu untersuchen, inwiefern Probleme durch Algorithmen lösbar oder nicht lösbar sind.
- Jedes entscheidbare Problem ist semi-entscheidbar, aber nicht umgekehrt.
- Es gibt Problem, die nicht semi-entscheidbar (und damit nicht entscheidbar) sind.
- Reduktionen erlauben es, die Unentscheidbarkeit eines Problems aus der Unentscheidbarkeit eines anderen herzuleiten.
- Sehr viele natürliche Probleme sind nicht entscheidbar, oft aber wenigstens semi-entscheidbar.