Universität Augsburg
Institut für Mathematik

Mathe-Camp 2023

Berechenbarkeit

Definition 1. Ein *Alphabet* Σ ist eine endliche Menge von Buchstaben. Ein *Wort* über einem solchen Alphabet ist eine endliche Folge von Buchstaben aus diesem Alphabet. Wir bezeichnet die Menge aller solcher Wörter mit Σ^*. Außerdem verwenden wir ϵ um das leere Wort zu beschreiben (also das Wort, das aus keinem Buchstaben besteht).

Eine *Sprache* S über einem Alphabet Σ ist eine (möglicherweise unendliche) Teilmenge $S \subseteq \Sigma^*$ (also eine Menge von Wörtern).

Definition 2. Sei S eine Sprache über einem Alphabet Σ^*. Das zu S gehörende *Entscheidungsproblem* lautet dann:

Eingabe: Ein Wort $w \in \Sigma^*$

Ausgabe: Ja, falls $w \in S$, und Nein, sonst

Wir sagen, dass eine Sprache S *entscheidbar* ist, wenn es einen Algorithmus (ein Computerprogramm) gibt, das das Entscheidungsproblem zu dieser Sprache löst.

Aufgabe 1. Wir betrachten die folgenden Sprachen:

- Die deutsche Sprache (im Sinn von: Die Menge aller Wörter, die um Duden stehen)
- Die Menge aller Palindrome (Wörter, die von vorne und hinten gelesen gleich sind)
- Die Menge aller natürlichen Zahlen
- Die Menge aller Terme aus natürlichen Zahlen sowie den Rechenzeichen $+$ und $-$
- Die Menge aller solcher Terme, die 0 ergeben
- Die Menge aller solcher Terme, die zusätzlich auch Klammern enthalten dürfen und korrekt geklammert sind
- Die Menge aller natürlichen Zahlen in einer festen Menge $S \subseteq \mathbb{N}$
- Die Menge aller Computerprogramme (in einer fest gewählten Programmiersprache, z.B. Python)
- Die Menge aller terminierenden Computerprogramme

Überlege dir jeweils, was ein geeignetes Alphabet für diese Sprachen ist.

Was denkst du: Welche dieser Sprachen ist entscheidbar?

Zusatzaufgabe 1. Zeige, dass es unentscheidbare Sprachen geben muss.

Hinweis 1: Überlege dir wie viele Computerprogramme es gibt und wie viele Sprachen.

Beobachtung 3. Wenn eine Sprache entscheidbar ist, ist klar wie wir das beweisen können: Wir müssen einfach nur ein entsprechendes Programm angeben (wobei es natürlich unter Umständen sehr schwer ist, so ein Programm zu finden).

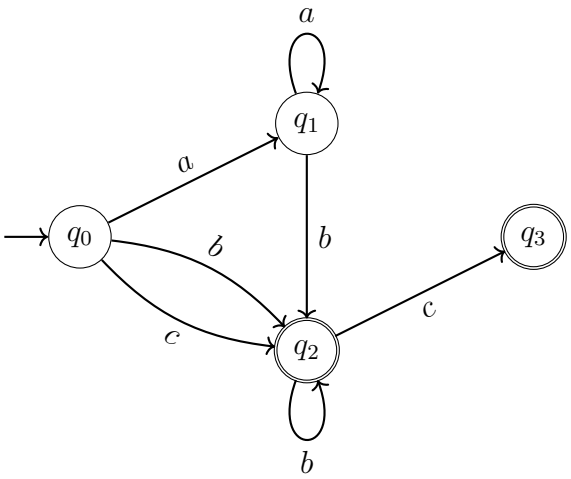
Aber wie können wir zeigen, dass eine Sprache unentscheidbar ist? Dafür müssen wir zeigen, dass es kein Programm geben kann, das diese Sprache entscheidet. Das klingt nach einer sehr schweren Aufgabe!

Im Folgenden wollen wir nun verschiedene Modelle für Computer kennenlernen und verstehen, was diese jeweils können (und nicht können).

1 Endliche Automaten

Ein endlicher Automat funktioniert grob gesprochen wie folgt: Er liest ein Eingabewort Buchstabe für Buchstabe ein und sagt am Ende entweder Ja oder Nein. Dazu hat der Automat eine endliche Menge von Zustanden zwischen denen er während dem Einlesen hin und her wechseln kann. Die Ausgabe am Ende hängt davon ab, in welchem Zustand sich der Automat am Ende befindet.

Solche endlichen Automaten kann man gut als gerichtete Graphen darstellen, zum Beispiel:



Hierbei sind q_0, q_1, q_2, q_3 die *Zustände* des Automaten, q_0 ist der Startzustand und q_2 und q_3 sind die (akzeptierenden) Endzustände. Der Automat funktioniert nun wie folgt: Er beginnt im Startzustand q_0 und liest den ersten Buchstaben des Eingabewortes. Dann läuft er entlang einer Kante, auf der dieser Buchstabe steht, zum nächsten Zustand und wiederholt den Vorgang hier mit dem zweiten Buchstaben des Wortes. Kommt der Automat in einen Zustand, von dem aus es keine ausgehende Kante mit dem aktuellen Buchstaben gibt, so verwirft er das Wort (antwortet also mit Nein). Schafft er es das Wort bis zum Ende zu lesen, so antwortet er mit Ja, falls er sich in diesem Moment in einem Endzustand befindet, und mit Nein sonst.

Beispiel 4. Wir legen unserem Automaten das Wort *aabc* vor. Der Automat liest den ersten Buchstaben (a) und wechselt zum Zustand q_1. Der Automat liest den zweiten Buchstaben (a) und landet erneut im Zustand a. Danach liest er den dritten Buchstaben (b) und wechselt in den Zustand q_2 und schließlich den letzten Buchstaben (c) und wechselt in den Zustand q_3. Da dies ein Endzustand (und das Wort zu Ende) ist, antwortet der Automat mit Ja.

Aufgabe 2. Welche der folgenden Worte akzeptiert unser Automat:

$$b, \quad cbb, \quad aaa, \quad aab, \quad aac, \quad cc$$

Mit etwas Überlegen erkennen wir, dass die Sprache aller Wörter, die dieser Automat akzeptiert wie folgt lautet:

$$\{a^n b^m, a^n b^m c, b^n, b^n c, cb^k, cb^k c \,|\, m, n \geq 1, k \geq 0\}$$

Hierbei steht beispielsweise $a^n b^m$ für das Wort, das aus n as gefolgt von m bs besteht. Also zum Beispiel $a^3 b^5 = aaabbbbb$.

Aufgabe 3. Welche der folgenden Sprachen über dem Alphabet $\{a, b, c\}$ kann von einem endlichen Automaten entschieden werden?

- Alle Wörter, die mit a beginnen und mindestens ein b enthalten.
- Alle Wörter, die nicht mit c enden.
- Alle Wörter, die mit aa beginnen und mit b enden.
- Alle Wörter, die genau zwei b enthalten.
- Alle Wörter, die eine durch 3 teilbare Anzahl an a enthalten.
- Die Sprache $\{\epsilon, ab, aabb, aaabbb\}$.
- Die Sprache $\{a^k b^k | k \geq 0\}$.

Wie können wir beweisen, dass eine Sprache von keinem endlichen Automaten entschieden werden kann? Eine Antwort hierauf liefert das Pumping-Lemma:

Lemma 5. *Sei M ein endlicher Automat und S die von ihm entschiedene Sprache. Dann gibt es eine natürliche Zahl $n \in \mathbb{N}$ mit der folgenden Pumping-Eigenschaft:*

Für jedes Wort $w \in S$ aus n oder mehr Buchstaben gibt es eine Zerlegung von w in drei Teilworte u, v und x, sodass uv höchstens n Buchstaben enthält, v mindestens einen Buchstaben und für jedes $i \in \mathbb{N}_0$ auch das Wort $uv^i x$ zur Sprache S gehört.

Oder in Formeln:

$$\exists n \in \mathbb{N} : \forall w \in S : |w| \geq n \implies \exists u, v, x \in \Sigma^* : w = uvx, |uv| \leq n, |v| \geq 1, \forall i \in \mathbb{N}_0 : uv^i x \in S.$$

Beweisidee. Wähle als n die Anzahl der Zustände von M. Lesen wir ein Wort w der Länge n oder mehr ein, so muss M nach dem Einlesen der ersten n Buchstaben mindestens einen Zustand doppelt besuchen (Schubfachprinzip!). Sei q dieser Zustand. Dann definieren wir u als den Teil von w, den M gelesen hat als es q zum ersten Mal besucht, v als den Teil bis zum nächsten Besuch von q und z als den Rest. Man kann sich nun überlegen, dass diese Zerlegung alle gewünschten Eigenschaften erfüllt. □

Anschaulich gesprochen besagt das Pumping-Lemma, dass in jeder Sprache, die von einem endlichen Automaten entschieden wird, alle ausreichend langen Worte ein Mittelstück besitzen, das man beliebig oft auf- oder abpumpen kann.

Für sich genommen ist das keine besonders interessante Eigenschaft. Wir können sie aber dazu nutzen, um zu zeigen, dass eine gegebene Sprache nicht von einem endlichen Automaten entschieden werden kann: Dazu müssen wir nur zeigen, dass diese Sprache die obige Pumping-Eigenschaft nicht hat.

Aufgabe 4. Zeige mit Hilfe des Pumping-Lemmas, dass es für folgende Sprachen keinen endlichen Automaten gibt:

- $\{a^k b^k | k \geq 0\}$
- $\{a^m b^k | k \geq m \geq 0\}$
- $\{a^m b^k | m \geq k \geq 0\}$
- $\{a^{k^2} | k \geq 0\}$

Zusatzaufgabe 2. Untersuche, ob die folgenden beiden Sprachen von einem endlichen Automaten entschieden werden können:

a) Die Sprache aller Wörter aus a und b, in denen die Buchstabenfolge ab und ba gleich oft vorkommen.

b) $S \coloneqq \{a^p | p \text{ Primzahl}\}$

Hinweis: Zahlen der Form $n! + 2$ könnten hier hilfreich sein.

Zusatzaufgabe 3. Ein *un*endlicher Automat ist ein Automat, der genauso funktioniert wie ein endlicher Automat, aber unendlich viele Zustände haben kann. Überlege dir, ob diese Automaten mächtiger sind als endliche Automaten. Gibt es Sprachen, die auch ein unendlicher Automat nicht entscheiden kann?

2 Abschlusseigenschaften

Lemma 6. *Seien S_1 und S_2 zwei Sprachen über einem gemeinsamen Alphabet Σ, die beide von einem endlichen Automaten entschieden werden. Dann werden auch die folgenden Sprachen von einem endlichen Automaten entschieden:*

- $S_1 \cap S_2$
- $S_1 \cup S_2$
- $\Sigma^* \setminus S_1$

Beweisidee. Seien M_1 und M_2 zwei endliche Automaten für S_1 und S_2. Für die ersten beiden Sprachen verwenden wir den sogenannten Produktautomaten: Dieser hat einen Zustand (q, p) für jeden Zustand q von M_1 und Zustand p von M_2. Es gibt nun einen Übergang a von (q, p) nach (q', p') falls M_1 einen Übergang a von q nach q' hat und M_2 einen Übergang a von p nach p'. Für die erste Sprache ist ein Zustand (q, p) akzeptierend, falls sowohl q als auch p akzeptierend sind. Für die zweite Sprache, falls mindestens q oder p akzeptierend sind.

Für die dritte Sprache überlegen wir uns, dass es leicht möglich ist M_1 so zu erweitern, dass es jedes Wort komplett einlesen kann (also nie stecken bleibt). Dann genügt es akzeptierende und nicht-akzeptierende Zustände zu vertauschen. □

Zusatzaufgabe 4. Wir betrachten die Sprache

$$\{a^m b^k c^k | m, k \geq 1\} \cup \{b^m c^k | m, k \geq 0\}.$$

Diese Sprache kann von keinem endlichen Automaten entschieden werden.

a) Warum können wir das nicht direkt mit Hilfe des Pumping-Lemmas zeigen?

b) Verwende eine geeignete Abschlusseigenschaft um diese Aussage zu zeigen.

Beobachtung 7. Diese Aufgabe zeigt also, dass das Pumping-Lemma keine vollständige Charakterisierung der durch endliche Automaten entschiedenen Sprachen liefert. Insbesondere kann es also vorkommen, dass eine Sprache die Pumping-Eigenschaft hat und trotzdem nicht von einem endlichen Automaten entschieden werden kann.

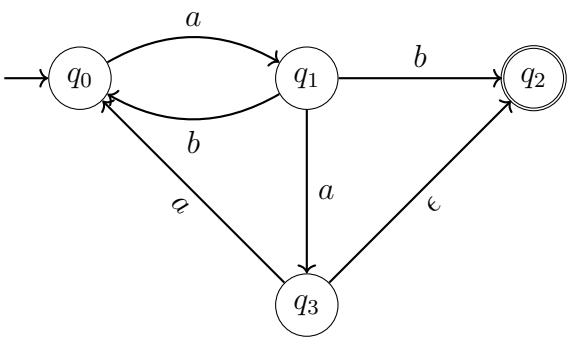
3 Nichtdeterministische endliche Automaten

Bisher haben wir nur sogenannte *deterministische* endliche Automaten betrachtet. Das heißt, für jedes feste Eingabewort gibt es nur einen einzigen Weg wie der Automat dieses einlesen kann. Insbesondere bedeutet das, dass jeder Zustand nur höchstens eine ausgehende Kante zu jedem Buchstaben des Alphabets haben darf.

Definition 8. Ein *nichtdeterministischer endlicher Automat* ist ein endlicher Automat, bei dem Zustände mehrere ausgehende Kanten mit dem gleichen Buchstaben haben dürfen. Außerdem kann es ϵ-Kanten geben. Das sind Kanten, über die der Automat den Zustand wechseln kann ohne dabei einen Buchstaben einzulesen.

Ein nichtdeterministischer endlicher Automat akzeptiert ein Eingabewort, wenn es wenigstens einen Weg gibt dieses Wort einzulesen und in einem akzeptierenden Zustand zu enden.

Aufgabe 5. Betrachte den folgenden nicht-deterministischen Automaten:



Welche Sprache entscheidet dieser Automat?

Die Akzeptanzbedingung von nichtdeterministischen Automaten kann man so interpretieren, dass der Automat beim Einlesen immer richtig rät, wann immer er mehrere Möglichkeiten zur Auswahl hat. Das klingt nun so als wären nichtdeterministische endliche Automaten deutlich mächtiger als deterministische. Überraschenderweise ist das aber nicht der Fall:

Lemma 9. *Sei S eine Sprache, die von einem nichtdeterministischen endlichen Automaten entschieden wird. Dann gibt es auch einen deterministischen endlichen Automaten, der diese Sprache entscheidet.*

Beweisidee. Das Vorgehen hier ist ähnlich wie für Lemma 6 nur, dass wir diesmal die Potenzmenge der Zustandsmenge als neue Zustandsmenge verwenden. □

Aufgabe 6. Wandle den nichtdeterministischen Automaten aus der vorherigen Aufgabe in einen deterministischen um.

Aufgabe 7. Seien S_1 und S_2 zwei Sprachen, die von einem endlichen Automaten entschieden werden. Dann kann auch die folgende Sprache von einem endlichen Automaten entschieden werden:

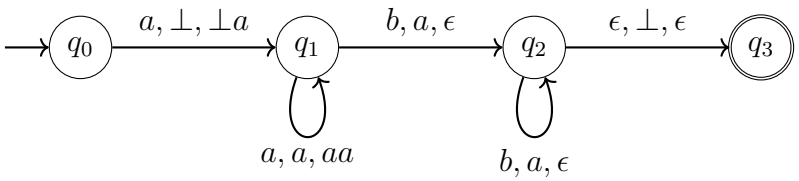
$$S_1 \cdot S_2 \coloneqq \{ww' \mid w \in S_1, w' \in S_2\}.$$

4 Kellerautomaten

Ein Problem von endlichen Automaten ist, dass sie nicht (beliebig) zählen können. Das liegt daran, dass sie keinen wirklichen Speicher haben (außer den endlich vielen Zuständen). Wir wollen unsere Automaten daher jetzt um einen einfachen Speicher erweitern: Einen *Keller* (oder auch *Stack*).

Definition 10. Ein Kellerautomat funktioniert im Wesentlichen wie ein endlicher Automat, hat aber zusätzlich Zugriff auf einen Keller. Dieser Keller enthält zu Beginn nur ein einziges Symbol $\perp$, welches nicht Teil des Eingabealphabets ist. Bei einem Übergang von einem Zustand zum nächsten liest ein Kellerautomat jetzt nicht nur den aktuellen Buchstaben des Eingabewortes, sondern auch das oberste Symbol im Keller. Danach löscht er dieses Symbol aus dem Keller und fügt eine beliebige (endliche!) Anzahl von neuen Buchstaben oben im Keller hinzu (insbesondere ist es auch erlaubt kein Symbol hinzuzufügen oder das gerade gelöschte Symbol wieder hinzuzufügen).

Einen Kellerautomat können wir nun wie folgt darstellen:



Auf den Kanten stehen nun immer drei Dinge: Der gelesene Buchstabe des Eingabewortes, der gelesene Buchstabe aus dem Keller und die Buchstaben, die neu in den Keller gelegt werden (oder ϵ, wenn nichts in den Keller gelegt wird).

Aufgabe 8. Welche Sprache entscheidet der obige Kellerautomat?

Auch für die von Kellerautomaten entschiedenen Sprachen gibt es ein Pumping-Lemma:

Lemma 11. *Sei S eine Sprache, die von einem Kellerautomaten entschieden wird. Dann gibt es eine Zahl $n \in \mathbb{N}$ mit folgender Eigenschaft:*

Für jedes $w \in S$ mit $|w| \geq n$ gibt es eine Zerlegung $w = uvxyz$, sodass $|vxy| \leq n$, $|vy| \geq 1$ und $uv^ixy^iz \in S$ für alle $i \in \mathbb{N}_0$.

Aufgabe 9. Welche der folgenden Sprachen kann von einem Kellerautomaten entschieden werden?

- Alle Wörter über dem Alphabet $\{a, b\}$ die gleich viele a und b enthalten.
- Alle Palindrome über einem festen Alphabet.
- $\{a^k b^k c^k | k \geq 0\}$
- Die Menge aller Wörter über dem Alphabet $\{(,)\}$, die einer korrekten Klammerung entsprechen.

Aufgabe 10. Finde zwei Sprachen S_1 und S_2 mit folgender Eigenschaft: Es gibt Kellerautomaten, die S_1 und S_2 entscheiden, aber keinen Kellerautomaten, der $S_1 \cap S_2$ entscheidet.

Zusatzaufgabe 5. Wir betrachten die Sprache aller Wörter der Form wcw, wobei w ein beliebiges Wort aus den Buchstaben a und b sein kann. Gibt es einen Kellerautomaten, der diese Sprache entscheidet?

5 Turingmaschinen

Eine Turingmaschine funktioniert ebenfalls ähnlich wie ein endlicher Automat, hat aber zwei zusätzliche Fähigkeiten:

- Sie kann die Eingabe nicht nur einmal von links nach rechts lesen, sondern beim Lesen sowohl vor als auch zurück springen.
- Sie kann die Eingabe nicht nur lesen, sondern auch überschreiben.

Definition 12. Eine Turingmaschine startet mit einem unendlichen langen Band, auf dem zu Beginn das Eingabewort und sonst nur Leerzeichen (ein spezielles Symbol $\textvisiblespace$, welches nicht Teil des Eingabealphabets ist) stehen. Sie hat einen Schreib-/Lesekopf, der zu Beginn auf den ersten Buchstaben des Eingabewortes zeigt. Sie hat endlich viele Zustände, von denen einer ein Startzustand ist und einer oder mehrere akzeptierende Endzustände. Außerdem kann es auch verwerfende Endzustände geben.

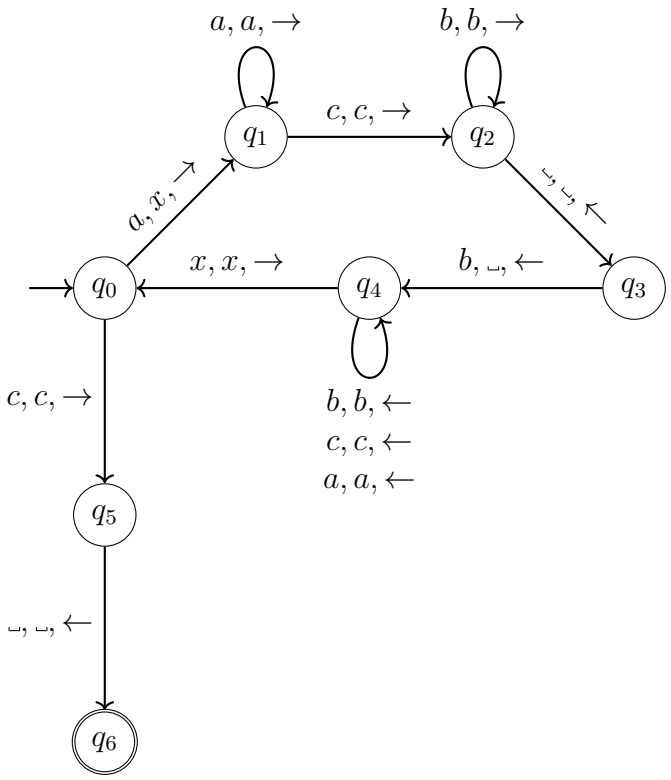
Ein Übergang zwischen zwei Zuständen besteht aus drei Dingen: Lese den aktuellen Buchstaben vom Band, überschreibe diesen mit einem neuen Buchstaben und bewege den Kopf dann nach links oder rechts.

Eine Turingmaschine akzeptiert ein Wort, wenn sie beim Einlesen des Wortes einen akzeptierenden Endzustand erreicht. Sie verwirft es, wenn sie einen verwerfenden Endzustand oder einen Zustand ohne geeignete ausgehende Kanten erreicht.

Beobachtung 13. Beim Einlesen eines Wortes durch eine Turingmaschine können drei Dinge passieren:

- Das Wort wird akzeptiert.
- Das Wort wird verworfen.
- Die Berechnung der Turingmaschine endet nicht.

Eine Turingmaschine können wir wie folgt darstellen:



Auf den Kanten stehen hier der gelesene Buchstabe, der geschriebene Buchstabe und schließlich die Richtung, in die sich der Kopf danach bewegt.

Aufgabe 11. Welche Wörter akzeptiert die obige Turingmaschine?

Aufgabe 12. Finde eine Turingmaschine, die die Sprache $\{wcw | w \in \{a, b\}^*\}$ entscheidet.

Aufgabe 13. Finde eine Turingmaschine, die auf manchen Eingaben nie hält.

Zusatzaufgabe 6. Finde eine Turingmaschine, die die Sprache $\{a^k b^k c^k | k \geq 0\}$ entscheidet.

Die *Churchsche These* besagt, dass eine Turingmaschine alles berechnen kann, was ein Algorithmus (Computerprogramm) berechnen kann. Eine solche Aussage kann man natürlich nicht beweisen. Bisher hat sie sich aber stets bewahrheitet. Insbesondere gilt sie also für alle bekannten Computermodelle und Programmiersprachen. Ein anderer interessanter Aspekt ist, dass eine Turingmaschine nicht mehr mächtiger wird, auch wenn wir ihr zusätzliche Fähigkeiten geben:

Definition 14. Eine Mehrband-Turingmaschine funktioniert wie eine (Einband-)Turingmaschine hat aber mehrere (endlich viele!) Bänder zur Verfügung und einen Schreib-/Lesekopf pro Band. Für einen Zustandsübergang liest und schreibt die Maschine je einen Buchstaben mit jedem Kopf auf dessen jeweiligen Band und bewegt die Köpfe dann (unabhängig voneinander) nach links oder rechts.

Lemma 15. *Alles, was eine Mehrband-Turingmaschine kann, können wir auch mit einer Einband-Turingmaschine machen.*

Zusatzaufgabe 7. Eine Keller-Turingmaschine ist eine (Einband-)Turingmaschine, die zusätzlich einen Keller zur Verfügung hat (der genauso funktioniert wie bei einem Kellerautomaten).

Zeige, dass Einband-Turingmaschinen alles können, was Keller-Turingmaschinen können.

Definition 16. Wir sagen, dass eine Turingmaschine M eine Sprache S *entscheidet*, wenn sie genau die Wörter aus S akzeptiert und alle anderen verwirft (und, insbesondere, also immer terminiert).

Wir sagen dann auch, dass diese Sprache *entscheidbar* ist.

Ein interessantes Problem in der Betrachtung von Turingmaschinen ist das sogenannte *Halteproblem*:

Eingabe: Eine Turingmaschine M und ein Wort w

Ausgabe: Ja, falls M bei Eingabe w irgendwann hält, und Nein sonst

Wir fragen uns nun, ob es eine Turingmaschine gibt, die dieses Problem entscheidet. Etwas formaler fragen wir uns, ob die folgende Sprache durch eine Turingmaschine entscheidbar ist:

$$\{\langle M, w\rangle \mid M \text{ ist eine Turingmaschine, die bei Eingabe von } w \text{ hält}\}$$

Dabei bezeichnet $\langle M, w\rangle$ eine geeignete Kodierung eines Tupels aus Turingmaschine und Eingabewort als Wort.

Aufgabe 14. Es gibt eine Turingmaschine, die bei Eingabe von $\langle M, w\rangle$ die Turingmaschine M auf dem Wort w simuliert (also einfach genau das macht, was die Turingmaschine M bei Eingabe von w machen würde).

Überlege dir, ob wir eine solche Turingmaschine verwenden können um das Halteproblem zu entscheiden.

Satz 17. *Das Halteproblem ist unentscheidbar.*

Beweisidee. Wir zeigen dies durch Widerspruch. Wir nehmen also an es gäbe eine Turingmaschine H, die das Halteproblem löst. Aus dieser können wir nun eine neue Turingmaschine P konstruieren, die folgendes tut:

Bei Eingabe eines Wortes w prüft P zunächst, ob w die Kodierung einer Turingmaschine ist. Falls nicht, wird w verworfen. Ansonsten sei M diese Turingmaschine.

Nun simuliert P die Maschine H um herauszufinden, ob M bei Eingabe von $w = \langle M\rangle$ hält. Falls dem so ist, startet P eine Endlosschleife. Ansonsten, akzeptiert P.

Wir geben dieser Turingmaschine P nun als Eingabewort ihre eigene Kodierung $\langle P\rangle$. Was passiert dann? Hält P oder nicht? Eines davon muss passieren, wir stellen aber fest, dass beides zu einem Widerspruch führt. Also kann es die Turingmaschine P nicht geben.

Die einzige Annahme für unsere Konstruktion von P war aber, dass es eine Turingmaschine H gibt, die das Halteproblem löst. Also kann es eine solche Turingmaschine nicht geben. □

Zusatzaufgabe 8. Wir betrachten die Sprache aller Kodierungen von Turingmaschinen, die das Wort, das ihrer eigenen Kodierung entspricht, nicht akzeptieren.

Zeige, dass diese Sprache unentscheidbar ist.

Zusatzaufgabe 9. Da Turingmaschinen das Band auch beschreiben dürfen, kann man den Inhalt des Bandes am Ende der Berechnung auch als Ausgabe interpretieren. Wir können also auch Turingmaschinen bauen, die eine bestimmte Ausgabe haben.

a) Konstruiere eine Turingmaschine, die als Eingabe eine Zahl in Binärdarstellung erhält und diese dann um eins erhöht.

b) Konstruiere eine Turingmaschine, die zwei Zahlen (in Binärdarstellung) erhält und diese zusammen addiert.

Hinweis: Es macht die Beschreibung der Turingmaschinen evtl. einfacher, wenn du annimmst, dass sie mehrere Bänder haben. Du musst die Turingmaschine außerdem nicht zwangsläufig komplett formal angeben.