Prof. Dr. André Schulz

Kurs 01658

Grundlagen der Theoretischen Informatik B

LESEPROBE

Fakultät für
Mathematik und Informatik

FernUniversität in Hagen

Kurseinheit 1

Unentscheidbare Sprachen

Im Kursteil A haben wir verschiedene Rechenmodelle untersucht. Dies waren unter anderem der endliche Automat, der Kellerautomat und die Turingmaschine. Jedem dieser Modelle haben wir eine (oder auch mehrere) Sprachklassen zugeordnet: Dem endlichen Automaten die regulären Sprachen, dem Kellerautomaten die kontextfreien Sprachen und der Turingmaschine die entscheidbaren und erkennbaren/aufzählbaren[1] Sprachen. Bei der Untersuchung dieser Sprachklassen haben wir festgestellt, dass die regulären Sprachen eine echte Teilmenge der kontextfreien Sprachen sind, und diese wiederum eine echte Teilmenge der entscheidbaren Sprachen. Der Fokus im Teil A lag darauf, die „Berechenbarkeit“ von Sprachen (bzw. Entscheidungsproblemen) nachzuweisen. Am Anfang des Teils B werden wir uns im Gegensatz dazu auf den Nachweis der „Nichtberechenbarkeit“ konzentrieren. Wir versuchen zudem, die Grenze zwischen Nichtberechenbaren und Berechenbaren gut zu bestimmen.

Wir wissen bereits, dass es mit $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$ eine nichtreguläre und mit $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$ eine nichtkontextfreie Sprache gibt. Um dies zu zeigen, hatten wir als Werkzeuge das Pumpinglemma (kontextfrei/regulär) und den Satz von Myhill–Nerode zur Verfügung. Wir wissen noch nicht, ob es Sprachen gibt, welche nicht entscheidbar oder nicht aufzählbar sind. Aus der Definition der aufzählbaren und der entscheidbaren Sprachen folgt zwar direkt, dass $\mathbb{E} \subseteq \mathbb{A}$, doch bislang haben wir noch nicht untersucht, ob dies eine *echte* Teilmengenbeziehung ist.

1.1 Existenz von nichtaufzählbaren Sprachen

Wir beginnen unsere Untersuchung mit einer Existenzaussage. Wir wollen beweisen, dass es weniger Turingmaschinen gibt als Sprachen über dem Alphabet $\{\mathtt{0}, \mathtt{1}\}$. Da es ja maximal so viele aufzählbare (also erkennbare) Sprachen geben kann, wie es Turingmaschinen gibt, würde daraus folgen, dass es Sprachen gibt, die nicht aufzählbar sind. Nun ist es jedoch so, dass es ja unendlich viele Turingmaschinen gibt und auch unendlich viele Sprachen. Wir müssen also eine Methode erarbeiten, mit welcher wir die „Größen“ von unendlichen Mengen in sinnvoller Weise miteinander vergleichen können. Um solche

[1] Beachten Sie, dass wir die Begriffe *aufzählbar* und *erkennbar* synonym benutzen, siehe dazu auch Kursteil A.

Vergleiche vornehmen zu können, definieren wir den Begriff der **Gleichmächtigkeit**.

Definition 1.1
Zwei Mengen X und Y heißen *gleichmächtig*, falls eine Bijektion zwischen X und Y existiert.

Ein einfaches Beispiel zweier gleichmächtiger Mengen sind die Mengen $X_1 = \{\mathsf{a}, \mathsf{b}\}$ und $X_2 = \{\mathtt{0}, \mathtt{1}\}$, denn wir können eine Funktion $f_1 \colon X_1 \to X_2$ angeben, die eine Bijektion ist; zum Beispiel:

$$f_1(x) := \begin{cases} \mathtt{0} & \text{falls } x = \mathsf{a} \\ \mathtt{1} & \text{falls } x = \mathsf{b}. \end{cases}$$

Es liegt auf der Hand, dass zwei endliche Mengen genau dann gleichmächtig sind, wenn sie die gleiche Kardinalität haben. Aber auch unendliche Mengen können gleichmächtig sein. Ein einfaches Beispiel hierfür sind die Mengen $Y_1 = \{i \in \mathbb{Z} \mid i \geq 0\}$ und $Y_2 = \{i \in \mathbb{Z} \mid i \geq 1\}$. Zwischen diesen Mengen können wir die folgende Bijektion $f_2 \colon Y_1 \to Y_2$ angeben:

$$f_2(x) := x + 1$$

Wir sehen, dass f_2 keine zwei Zahlen auf eine gemeinsame Zahl abbildet, damit ist f_2 injektiv. Des Weiteren gibt es für jedes $y \in Y_2$ mit $y' = y - 1$ eine Zahl, sodass $f_2(y') = y$. Daraus folgt, dass f_2 surjektiv ist, und demnach eine Bijektion darstellt.

Test 1.1
Zeigen Sie, dass die Mengen $Z_1 = \{i \in \mathbb{N} \mid i \geq 1\}$ und $Z_2 = \{p \in \mathbb{N} \mid p \text{ ist Primzahl}\}$ gleichmächtig sind.

Von besonderem Interesse ist es, die Mengen zu bestimmen, die gleichmächtig zu den natürlichen Zahlen sind. Wir nehmen an, dass die 0 keine natürliche Zahl ist, alle Argumente des Kurses funktionieren aber genauso gut, wenn wir die 0 als natürliche Zahl zulassen würden.

Definition 1.2 — abzählbar.
Eine Menge X heißt *abzählbar*, falls (i) X endlich ist oder (ii) X und die Menge der natürlichen Zahlen $\mathbb{N}$ gleichmächtig sind.

Aus unseren bisherigen Beobachtungen folgt, dass zum Beispiel die Menge $\{p \in \mathbb{N} \mid p \text{ ist Primzahl}\}$ abzählbar ist. Wenn eine Menge X abzählbar ist, heißt dies also, dass wir alle ihre Elemente mit den natürlichen Zahlen durchnummerieren können. Eine Bijektion $f \colon \mathbb{N} \to X$ beschreibt die Vergabe der Nummern. Wir bezeichnen die dazugehörige Folge $f(1), f(2), f(3), \ldots$ auch als eine **Nummerierung** der Menge X. Beachten Sie aber, dass *abzählbar* und *aufzählbar* zwei verschiedene Eigenschaften sind.

Test 1.2
Zeigen Sie, dass wenn X abzählbar, dann ist auch jede Teilmenge $Y \subseteq X$ abzählbar.

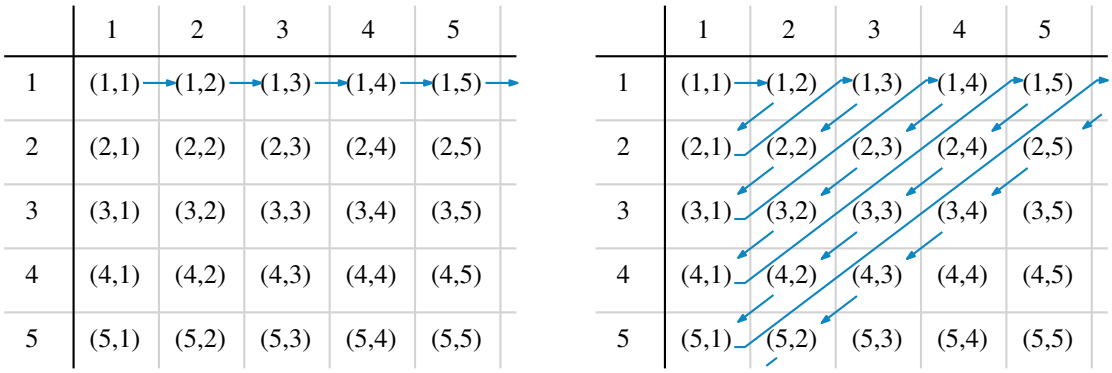

Abbildung 1.1: Strategie beim Durchnummerieren der Einträge einer Tabelle: Zeilenweises Abzählen (links), Cantornummerierung (rechts).

Lemma 1.1
Die Menge der ganzen Zahlen $\mathbb{Z}$ ist abzählbar.

Beweis. Wir geben eine Funktion $f : \mathbb{Z} \to \mathbb{N}$ an als

$$f(x) = \begin{cases} 2|x| + 1 & \text{falls } x \leq 0 \\ 2x & \text{falls } x > 0 \end{cases}$$

Wir erkennen, dass f die positiven ganzen Zahlen auf die geraden natürlichen Zahlen und die nichtpositiven ganzen Zahlen auf die ungeraden natürlichen Zahlen abbildet. Es ist leicht zu sehen, dass es keine ganzen Zahlen gibt, die auf die gleiche natürliche Zahl abgebildet werden. Des Weiteren gibt es für jede Zahl $i \in \mathbb{N}$ eine ganze Zahl x mit $f(x) = i$. Also ist f surjektiv und injektiv und damit eine Bijektion. ■

In den bisherigen Beispielen war es einfach, Gleichmächtigkeit (bzw. Abzählbarkeit) durch eine Bijektion anzugeben. Wir wollen uns nun ein weiteres, sehr interessantes Beispiel ansehen, bei welchem der Nachweis der Abzählbarkeit nicht so offensichtlich ist.

Lemma 1.2
Die Menge $\mathbb{N}^2 = \mathbb{N} \times \mathbb{N}$ ist abzählbar.

Beweis. Bevor wir die Abzählbarkeit von $\mathbb{N}^2$ beweisen, werden wir kurz darauf eingehen, welcher Ansatz nicht funktioniert. Die Menge $\mathbb{N} \times \mathbb{N}$ ist die Menge der geordneten Paare von natürlichen Zahlen. Diese Menge kann man sich gut in einer Tabelle vorstellen, bei welcher der Eintrag der i-ten Zeile und der j-ten Spalte das Tupel (i, j) enthält. Unser Ziel ist es, jedem Eintrag dieser Tabelle genau eine natürliche Zahl zuzuordnen. Am einfachsten wäre es natürlich, dass wir die Zahlen $1, 2, 3, \ldots$ Zeile für Zeile über die Tabelle verteilen. Dies funktioniert jedoch nicht. In der ersten Zeile gibt es ja unendlich viele Einträge, deshalb würden wir bereits für die erste Zeile alle natürlichen Zahlen „aufbrauchen“. Auch wenn wir spaltenweise vorgehen, haben wir das gleiche Problem. Es bedarf also einer anderen Strategie.

Anstatt die Nummern zeilen- oder spaltenweise zu vergeben, werden wir sie diagonal verteilen. Diese Idee (siehe Abbildung 1.1) ist so einfach wie genial. Wir bezeichnen mit der k-ten Gegendiagonalen der Tabelle alle Einträge (i, j) mit $i+j = k$. Die k-te Gegendiagonale

hat also $k-1$ Einträge, welche wir nach aufsteigender Zeilennummer sortieren. Zum Beispiel erhalten wir für die vierte Gegendiagonale die Folge $(1,3),(2,2),(3,1)$. Wir werden nun die Gegendiagonalen in aufsteigender Nummer abarbeiten. Dabei vergeben wir der Reihe nach die Nummern von $\mathbb{N}$ für die Einträge der Tabelle. Haben wir eine Gegendiagonale durchnummeriert, fahren wir mit der nächsten fort und beginnen das Nummerieren dort mit der nun folgenden Nummer. Auf diese Weise bekommen alle Einträge der Tabelle eine Nummer aus $\mathbb{N}$. Die so erhaltene Nummerierung beginnt demnach mit

$$(1,1),(1,2),(2,1),(1,3),(2,2),(3,1),(1,4),(2,3),(3,2),(4,1),(1,5),(2,4),\ldots.$$

Die Vergabe der Nummern impliziert eine Bijektion zwischen $\mathbb{N}$ und $\mathbb{N}^2$. Wir weisen hierbei der Zahl i das mit i beschriftete Paar der Tabelle zu. Offensichtlich ist diese Funktion injektiv und surjektiv. ■

Die Strategie, die Elemente von $\mathbb{N}\times\mathbb{N}$ in der im Beweis diskutierten Art zu nummerieren, ist als **Cantornummerierung** bekannt (nach ihrem Entdecker, dem deutschen Mathematiker Georg Cantor). Die daraus abgeleitete Funktion, die jedem Tabelleneintrag eine natürliche Zahl zuordnet, nennt man **cantorsche Paarungsfunktion**.

> **Test 1.3**
>
> Bestimmen Sie eine geschlossene Darstellung (Formel) für die cantorsche Paarungsfunktion.

Korollar 1.1
Die Menge der rationalen Zahlen $\mathbb{Q}$ ist abzählbar.

Beweis. Jede Zahl aus $\mathbb{Q}$ ist als Bruch p/q darstellbar. Diese Brüche tragen wir in eine Tabelle ein, sodass p/q in der Zelle (p,q) eingetragen wird. Nun nummerieren wir die Tabelleneinträge mit der Cantornummerierung. Daraus ergibt sich eine Folge F von Brüchen, die wie folgt beginnt:

$$1/1, 1/2, 2/1, 1/3, 2/2, 3/1, 1/4, 2/3, \ldots$$

Wir streichen in dieser Folge alle Brüche, die wir kürzen können, zum Beispiel 3/9. Die daraus entstandene Folge nennen wir F'. Die ersten Einträge von F' sind

$$1, 1/2, 2, 1/3, 3, 1/4, 2/3, 3/2, 4, 1/5, 5, \ldots.$$

Wir können nun die folgende Bijektion zum Nachweis der Gleichmächtigkeit zu $\mathbb{N}$ benutzen:

$$f(i) := i\text{-ter Eintrag in } F' \text{ als rationale Zahl.}$$

■

Man könnte den Eindruck bekommen, dass alle Mengen von Zahlen abzählbar sind. Das folgende Lemma zeigt aber, dass dem nicht so ist. Eine Menge, die nicht abzählbar ist, nennen wir **überabzählbar**.

$$
\begin{array}{lll}
f(1) & =?? . & \underline{9}\ 1\ 3\ 3\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\
f(2) & =?? . & 1\ \underline{5}\ 6\ 3\ 7\ 1\ 2\ 1\ 2\ 1\ 2 \\
f(3) & =?? . & 1\ 1\ \underline{1}\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \\
f(4) & =?? . & 4\ 5\ 1\ \underline{1}\ 0\ 1\ 9\ 3\ 3\ 1\ 2 \\
f(5) & =?? . & 2\ 2\ 2\ 2\ \underline{2}\ 0\ 0\ 0\ 0\ 0\ 0 \\
f(6) & =?? . & 2\ 2\ 2\ 2\ 2\ \underline{0}\ 0\ 0\ 0\ 0\ 0 \\
f(7) & =?? . & 1\ 4\ 1\ 5\ 9\ 2\ \underline{6}\ 5\ 3\ 5\ 8 \\
\\
r = 0 . & & 0\ 6\ 2\ 2\ 3\ 1\ 7\ \cdots
\end{array}
$$

Abbildung 1.2: Konstruktion einer reellen Zahl r, welche in der Nummerierung der reellen Zahlen durch f fehlt.

Lemma 1.3
Die Menge der reellen Zahlen $\mathbb{R}$ ist überabzählbar.

Beweis. Wir führen diesen Beweis als Widerspruchsbeweis. Dazu nehmen wir an, dass es eine Bijektion $f : \mathbb{N} \to \mathbb{R}$ gibt. Wir nutzen zum Beweis eine Tabelle T, in der wir die Nachkommastellen aller reellen Zahlen in Dezimaldarstellung „eintragen“. Die Tabelle hat unendlich viele Spalten und Zeilen. In der Zelle (i, j) tragen wir die j-te Nachkommastelle der reellen Zahl $f(i)$ ein. Wir schreiben für den Inhalt der Zelle (i, j) von T kurz $T[i, j]$.

Wir werden nun eine reelle Zahl definieren, welche nicht in der Tabelle als Zeile aufgelistet ist. Dazu nutzen wir die Funktion

$$
d(i, j) := \begin{cases} 0 & \text{falls } T[i, j] = 9 \\ T[i, j] + 1 & \text{sonst.} \end{cases}
$$

Diese Funktion garantiert, dass $T[i, j] \neq d(i, j)$ für alle $(i, j) \in \mathbb{N}^2$. Des Weiteren sind die Funktionswerte von d nicht großer als 9. Sei r nun folgende reelle Zahl in Dezimaldarstellung angegeben:

$$
r = 0.d(1, 1)d(2, 2)d(3, 3)d(4, 4)d(5, 5) \ldots
$$

Ein Beispiel für diese Konstruktion ist in Abbildung 1.2 zu sehen.

Wir nehmen nun an, dass es ein i gibt mit $f(i) = r$. Das heißt, dass die Nachkommastellen von r in der Tabelle T in Zeile i eingetragen wurden. Die i-te Nachkommastelle von r ist $d(i, i)$, der Eintrag in der Tabelle T an dieser Stelle ist aber $T[i, i] \neq d(i, i)$. Das ist ein Widerspruch, folglich ist die Zahl r nicht in T eingetragen worden, und damit fehlt r in der Nummerierung. Wir sehen also, dass $\mathbb{R}$ nicht abzählbar ist. ■

An dieser Stelle wollen wir uns die Zeit nehmen, über den Beweis von Lemma 1.3 noch etwas nachzudenken. Die Idee hinter dem Beweis ist sehr elegant, trotzdem kommt es hier oft zu Verwirrungen. Wir hatten gezeigt, dass in unserer Nummerierung der reellen Zahlen die Zahl r nicht vorkommen kann. Natürlich ist es möglich, eine andere Nummerierung zu finden, die r enthält. So könnte man zum Beispiel allen Zahlen eine um eins größere Nummer geben und dann r die Nummer 1 zuordnen. Bei dieser neuen Abbildung fehlt dann aber wieder eine (andere) Zahl.

Die Beweistechnik in Lemma 1.3 trägt den Namen **Diagonalisierung** und ist ein sehr mächtiges Werkzeug. Wir werden auch andere Sätze mit dieser Technik beweisen. Bei

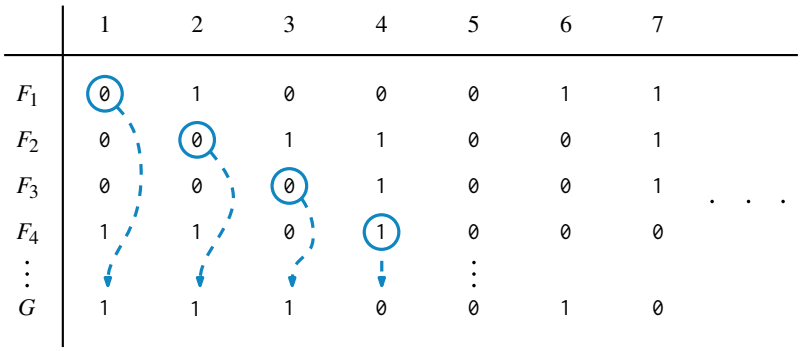

Abbildung 1.3: Allgemeines Prinzip der Diagonalisierung.

der Diagonalisierung geht es darum, nachzuweisen, dass eine Folge in einer Nummerierung von Folgen $(F_i)_{i\in\mathbb{N}}$ fehlt. Im vorigen Beweis ergab sich zum Beispiel F_i aus den Nachkommastellen der reellen Zahl $f(i)$. Man trägt die Folgen F_i als Zeilen in eine unbeschränkte Tabelle ein. Hierbei wird die Folge F_i in die i-te Zeile geschrieben. Nun nimmt man sich die Diagonale der Tabelle als neue Folge und verändert jeden Wert in dieser Folge. Dadurch erreicht man, dass die so konstruierte Folge G sich von allen Folgen der Tabelle unterscheidet. Konkret unterscheidet sich G von der Folge F_i an der i-ten Stelle (Tabelleneintrag (i,i)). Dieses Prinzip wird noch einmal in Abbildung 1.3 veranschaulicht.

Der Diagonalisierungsbeweis beruht auf dem Umstand, dass zwei Folgen schon dann unterschiedlich sind, wenn sie sich an einer Stelle unterscheiden. Wir konnten die erste Stelle benutzen, um die Ungleichheit zur ersten Folge zu erzwingen, die zweite Stelle für die Ungleichheit zur zweiten Folge, und so weiter.

Mit dem Lemma 1.3 haben wir gezeigt, dass es zwei unendliche Mengen gibt, die nicht gleichmächtig sind. In diesem Sinne konnten wir also nachweisen, dass es mehr reelle Zahlen als natürliche Zahlen gibt. Es gilt nun diese Ideen umzuformulieren, sodass man zeigen kann, dass es mehr Sprachen gibt als Turingmaschinen. Bislang haben wir nur Mengen von Zahlen verglichen. Wir wollen nun auch Sprachen und Mengen von Funktionen vergleichen.

Lemma 1.4
Die Menge Σ^* ist abzählbar.

Beweis. Wir haben bereits im Kursteil A die Standardnummerierung von Σ^* vorgestellt. Zur Erinnerung: Man sortiert alle Zeichen in Σ^* entsprechend ihrer Länge in aufsteigender Reihenfolge, wobei man Wörter mit gleicher Länge lexikographisch sortiert. Eine Bijektion f zwischen $\mathbb{N}$ und Σ^* ergibt sich direkt als

$$f(i) := \text{ das } i\text{-te Wort in der Standardaufzählung von } \Sigma^*.$$

■

Lemma 1.5
Die Menge $\{L \subseteq \Sigma^*\}$ aller Sprachen über Σ ist überabzählbar.

Beweis. Wir führen den Beweis als Diagonalisierungsbeweis. Wir nehmen also an, dass

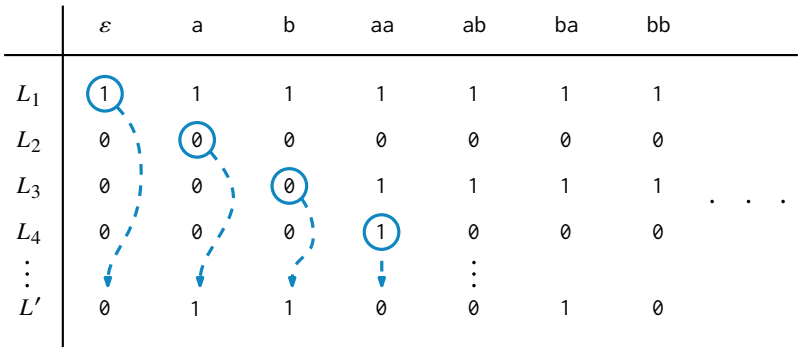

Abbildung 1.4: Ausführung des Beweises von Lemma 1.5 als Diagonalisierungsbeweis. Im Beispiel wurde $L_1 = \Sigma^*$, $L_2 = \emptyset$, $L_3 = \Sigma^2$, $L_4 = \{\mathtt{aa}\}$ gewählt.

eine Nummerierung aller Sprachen über Σ existiert. Sei dann L_i die i-te Sprache in dieser Nummerierung. Wir tragen alle Sprachen in eine Tabelle T ein. Hierbei werden wir L_i in die i-te Zeile eintragen. Die Spalten von T sind mit den Wörtern aus Σ^* beschriftet, sagen wir in der Reihenfolge der Standardnummerierung. Die Beschriftung der i-ten Spalte bezeichnen wir mit w_i. Wir markieren die Wörter, welche in L_i enthalten sind, indem wir in der korrespondierenden Zelle eine 1 eintragen. Wörter aus Σ^*, die nicht in L_i enthalten sind, markieren wir mit 0. Man könnte auch sagen, dass wir die charakteristische Funktion der Menge L_i in die i-te Zeile eintragen.

Wir konstruieren nun eine Sprache L', welche in der Nummerierung der L_i fehlt. Dazu „negieren wir die Diagonale der Tabelle“ und übernehmen die so erzeugte Sequenz als charakteristische Funktion von L' (siehe Abbildung 1.4). Das heißt konkret: Wenn $w_i \in L_i$, dann nehmen wir w_i nicht zu L' hinzu. Ist jedoch $w_i \notin L_i$, dann fügen wir w_i zu L' hinzu. Formal können wir definieren, dass

$$L' := \{w_i \mid w_i \notin L_i\}.$$

Die Sprache L' fehlt in der Nummerierung der L_i, da sie sich von jeder dieser Sprachen unterscheidet, da

$$w_i \in L_i \iff w_i \notin L'.$$

Somit haben wir durch den Diagonalisierungsbeweis gezeigt, dass $\{L \subseteq \Sigma^*\}$ überabzählbar ist.

■

Den letzten Beweis kann man natürlich auch kürzer führen und zum Beispiel auf die Verwendung einer expliziten Tabelle verzichten. Die Verwendung der Tabelle macht jedoch sehr deutlich, warum es sich bei diesem Beweis um einen Diagonalisierungsbeweis handelt.

Satz 1.1
Es gibt eine Sprache, die nicht aufzählbar ist.

Beweis. Für den Beweis fixieren wir ein Alphabet Σ. Es gibt höchstens so viele aufzählbare (erkennbare) Sprachen über Σ, wie es Turingmaschinen gibt. Jede Turingmaschine können wir als Wort über $\Sigma' = \{0, 1\}$ kodieren. Nach Lemma 1.4 ist $(\Sigma')^*$ abzählbar. Deshalb gibt es nur abzählbar viele Turingmaschinen und damit nur abzählbar viele erkennbare Sprachen.

Da es aber nach Lemma 1.4 überabzählbar viele Sprachen über Σ gibt, können nicht alle diese Sprachen erkennbare Sprachen sein. ■

Das Argument aus dem Beweis von Satz 1.1 ist eigentlich noch stärker. Man sieht sogar, dass es nicht nur eine nichtaufzählbare Sprache gibt sondern unendlich viele. Man könnte mit etwas mehr Aufwand sogar zeigen, dass fast alle Sprachen nicht aufzählbar sind.

1.2 Konstruktion von nichtaufzählbaren Sprachen

Wir wissen nun, dass es Sprachen gibt, die nicht aufzählbar sind. Unser nächstes Ziel ist es, eine solche Sprache auch konstruktiv zu bestimmen. Wir werden dies über einen kleinen Umweg machen, indem wir zuerst eine unentscheidbare Sprache konstruieren.

Satz 1.2
Die Sprache $A_{\text{TM}} := \{\langle M, w\rangle \mid M \text{ ist Turingmaschine die } w \text{ akzeptiert}\}$ ist nicht entscheidbar.

Beweis. Wir führen diesen Beweis als Widerspruchsbeweis. Dazu nehmen wir an, dass A_{TM} entscheidbar ist. Das heißt, es existiert eine Turingmaschine die A_{TM} erkennt und auf jeder Eingabe hält. Wir nennen diese Turingmaschine H. Es gilt also:

$$H(\langle M, w\rangle) = \begin{cases} \text{akzeptiert} & \text{M akzeptiert } w \\ \text{verwirft} & \text{M akzeptiert } w \text{ nicht} \end{cases}$$

Wir geben nun eine Turingmaschine $D(\langle M\rangle)$ an, welche wie folgt arbeitet.

1. D simuliert $H(\langle M, \langle M\rangle\rangle)$
2. D gibt das entgegengesetzte Ergebnis der Simulation zurück

Wir sehen, dass die Eingabe von D als Kodierung einer Turingmaschine aufgefasst wird. Im ersten Schritt wird die Maschine H als Unterprogramm mit der geforderten Eingabe aufgerufen. Offensichtlich existiert die Turingmaschine D, wenn die Turingmaschine H existiert. In diesem Fall ist D sogar ein Entscheider, hält also auf allen Eingaben.

Wir zeigen nun, dass D nicht existieren kann. Die Turingmaschine D liefert folgendes Ergebnis:

$$D(\langle M\rangle) = \begin{cases} \text{akzeptiert} & \text{wenn } M(\langle M\rangle) \text{ verwirft} \\ \text{verwirft} & \text{wenn } M(\langle M\rangle) \text{ akzeptiert} \end{cases}$$

Das heißt aber insbesondere, dass

$$D(\langle D\rangle) = \begin{cases} \text{akzeptiert} & \text{wenn } D(\langle D\rangle) \text{ verwirft} \\ \text{verwirft} & \text{wenn } D(\langle D\rangle) \text{ akzeptiert .} \end{cases}$$

Dies ist offensichtlich ein Widerspruch. Somit kann D nicht existieren und deshalb kann H nicht existieren. Unsere Annahme war daher falsch und demnach gibt es keinen Entscheider für A_{TM}. ■

Der letzte Beweis beruhte schon wieder auf einem Diagonalisierungsargument, auch wenn dieses etwas im Verborgenen bleibt. Die Turingmaschine H definiert uns eine Tabelle.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$
M_1	akzeptiert	akzeptiert	akzeptiert	akzeptiert
M_2	verwirft	akzeptiert	verwirft	verwirft
M_3	verwirft	verwirft	verwirft	verwirft
M_4	verwirft	akzeptiert	akzeptiert	verwirft
$\vdots$			$\vdots$	
D	verwirft	verwirft	akzeptiert	akzeptiert

Abbildung 1.5: Das dem Beweis von Satz 1.2 zu Grunde liegende Diagonalisierungsargument.

Die Zeilen dieser Tabelle sind mit den Turingmaschinen M_i beschriftet, wobei es für jede mögliche Turingmaschine eine Zeile gibt. Die Spalten sind mit den Wörtern über Σ beschriftet, und zwar so, dass die i-te Spalte mit $\langle M_i \rangle$ beschriftet ist. Der Eintrag in Zelle $(M_i, \langle M_j \rangle)$ entspricht dem Ergebnis von $H(\langle M_i, \langle M_j \rangle \rangle)$. Siehe hierzu auch Abbildung 1.5. Die Maschine D übernimmt nun das Diagonalisieren. Dazu kehrt sie die Antworten auf der Diagonale um. Der Punkt ist, dass die zu D zugeordnete Sprache $L(D)$ entscheidbar ist, wenn H entscheidbar ist. Wir haben aber $L(D)$ von allen $L(M_i)$ unterschiedlich gemacht. Deshalb kann $L(D)$ nicht entscheidbar sein. Dadurch erreichen wir den Widerspruch zur Annahme, dass H existiert.

Der folgende Satz ist eine direkte Konsequenz aus Satz 1.2.

Satz 1.3
Das Komplement der Sprache A_{TM} ist nicht aufzählbar.

Beweis. Wir wissen nach Satz 1.2, dass $A_{\text{TM}} \notin \mathbb{E}$. Im Kursteil A haben wir zudem gesehen, dass A_{TM} von der universellen Turingmaschine erkannt wird, also ist $A_{\text{TM}} \in \mathbb{A}$. Ein weiterer Satz des A-Teils besagte, dass wenn $X \in \mathbb{A}$ und $X \in \text{co-}\mathbb{A}$, dann auch $X \in \mathbb{E}$. Angenommen $A_{\text{TM}} \in \text{co-}\mathbb{A}$, dann würde nach diesem Satz gelten, dass $A_{\text{TM}} \in \mathbb{E}$, was jedoch nicht gilt. Somit ist also $A_{\text{TM}} \notin \text{co-}\mathbb{A}$ oder anders gesagt, das Komplement von A_{TM} ist nicht aufzählbar. ■

An dieser Stelle wollen wir die Konsequenzen aus den Sätzen 1.2 und 1.3 diskutieren. Die Sprache A_{TM} ist nicht irgendeine künstlich konstruierte Sprache. Das dieser zur Sprache zu Grunde liegende Entscheidungsproblem hat eine hohe praktische Relevanz. Im Entscheidungsproblem fragen wir, ob eine Turingmaschine M eine Eingabe w akzeptiert. Etwas allgemeiner gefasst könnte man sagen, wir wollen wissen, ob ein gegebener Algorithmus A bei Eingabe x, die Ausgabe y produziert. Dieses Problem beschreibt die Verifikation eines Algorithmus/Programms. Die Aussage aus den beiden letzten Sätzen kann man so interpretieren, dass es keinen Algorithmus gibt, mit Hilfe dessen wir andere Algorithmen verifizieren können. Zwar können wir positive Ergebnisse verifizieren, da $A_{\text{TM}} \in \mathbb{A}$, wir werden aber nicht in allen Fällen die negativen Ergebnisse feststellen können, da $A_{\text{TM}} \notin \text{co-}\mathbb{A}$. Es ist natürlich möglich, für ausgewählte Algorithmen einen Nachweis zu erbringen, dass sie korrekt arbeiten. Was wir gezeigt haben ist, dass es kein allgemeines Verfahren für die Verifikation geben kann.

Ein zu A_{TM} eng verwandtes Problem ist das Problem

$$HALT := \{\langle M, w\rangle \mid \text{ die TM } M \text{ hält bei Eingabe } w\}.$$

Mit den Ideen aus Satz 1.2 kann man zeigen, dass auch $HALT \notin \mathbb{E}$.

Satz 1.4
Die Sprache *HALT* ist nicht entscheidbar und nicht co-aufzählbar.

Beweis. Der Beweis des Satzes ist völlig analog zum Beweis der Sätze 1.2 und 1.3. Wir nehmen an, dass *HALT* durch eine Maschine H entschieden wird und konstruieren daraufhin eine Turingmaschine D, welche bei Eingabe $\langle M\rangle$ zuerst $H(\langle M, \langle M\rangle\rangle)$ simuliert. Verwirft H die Eingabe, stoppt D akzeptierend. Akzeptiert H die Eingabe, gehen wir in eine Endlosschleife. Nun gilt

$$D(\langle D\rangle)\text{zykelt} \iff H(\langle D, \langle D\rangle\rangle)\text{akzeptiert} \iff D(\langle D\rangle) \text{ stoppt.}$$

Somit kann eine solche Turingmaschine H nicht existieren.

Als Nächstes zeigen wir, dass *HALT* erkennbar ist. Die Turingmaschine U' die *HALT* erkennt, arbeitet wie die universelle Turingmaschine, mit dem einzigen Unterschied, dass alle Übergänge in den verwerfenden Zustand nun in den akzeptierenden Zustand führen. Die Modifikation bewirkt, dass die Eingabe $\langle M, w\rangle$ von U' immer akzeptiert wird, wenn $M(w)$ stoppt (egal wie). Falls $M(w)$ nicht stoppt, wird $\langle M, w\rangle$ nicht akzeptiert, da U' auf dieser Eingabe zykeln wird. Da somit *HALT* erkennbar aber nicht entscheidbar ist, muss folgen, dass *HALT* nicht co-erkennbar (siehe Beweis zu Satz 1.3). ■

1.3 Entscheidbarkeit des universellen Wortproblems

Mit dem Wortproblem bezeichneten wir das Problem, zu entscheiden, ob ein Eingabewort w aus einer gegebenen Sprache ist, oder nicht. Alle Entscheidungsprobleme, werden bei uns als Wortprobleme modelliert. Für die (algorithmische) Lösung des Wortproblems ist es natürlich wichtig, in welcher Form die Sprache gegeben ist. Wir haben bislang im A-Teil verschiedene Möglichkeiten kennengelernt, wie man eine nichtendliche Sprache, durch ein endliches Wort beschreiben kann. Dazu wählt man sich ein Modell und gibt dann eine Kodierung einer konkreten Instanziierung (ein Programm, Beschreibung) an.

Es ist wünschenswert, alle Wortprobleme einer Sprachklasse *universell* lösen zu können. Damit ist gemeint, dass es einen Algorithmus gibt, der für jede Modellinstanz mit Eingabewort das Wortproblem löst. Ein solcher Algorithmus ist in diesem Sinne universell für das entsprechende Berechnungsmodell. Wir haben bereits gesehen, dass für das Berechnungsmodell Turingmaschine, kein solcher Algorithmus exisitieren kann, da $A_{\text{TM}} \notin \mathbb{E}$. Für die Modelle endlicher Automat und kontextfreie Grammatik existiert aber ein universeller Algorithmus für das Wortproblem. Dies wurde am Ende des A-Teils (Satz 7.5) gezeigt. Demnach gilt also, dass

$$A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ ist CFG und } w \in L(G)\} \in \mathbb{E}, \text{ und}$$
$$A_{\text{DEA}} = \{\langle M, w\rangle \mid M \text{ ist DEA und } w \in L(M)\} \in \mathbb{E}.$$