

---

## 2. Algorithmen: Berechenbarkeit, Entscheidbarkeit und Aufzählbarkeit

---

Ein *Algorithmus (Rechenvorschrift)* ist eine Vorschrift zur Lösung eines Problems. Solch ein Problem wird i. Allg. unendlich viele Instanzen haben und der Algorithmus wird typischerweise zu einer gegebenen Eingabe die zugehörige Instanz lösen. Die Probleme können dabei *Entscheidungsprobleme* oder *Berechnungsprobleme* sein. D.h. die Aufgabe kann sein, für die Eingabe festzustellen, ob sie eine gewisse Eigenschaft hat (Bsp.: Primzahltest), oder die Eingabe in eine Ausgabe zu transformieren (Bsp.: Addition zweier Zahlen).

In diesem Abschnitt wollen wir kurz die wesentlichen Eigenschaften und Typen von Algorithmen untersuchen, um dann in den nächsten beiden Abschnitten den Algorithmusbegriff zu formalisieren.

Zu den Anforderungen an Algorithmen gehören:

1. Die Beschreibung des Algorithmus ist endlich. (*Finitheit*)
2. Das Ergebnis des Algorithmus hängt nur von den Eingaben und nicht von äußeren Einflüssen ab: Wird der Algorithmus mehrfach in der gleichen Anfangssituation gestartet, liefert er stets das gleiche Ergebnis. (*Determiniertheit*)
3. Die Ausführung des Algorithmus besteht aus einer Folge elementarer Schritte, die effektiv (= tatsächlich) ausführbar sind. (*Effektivität*)

Die Ausführung eines Algorithmus darf hierbei keine besondere Intelligenz oder Kreativität des Rechnenden voraussetzen, sondern sie muss rein mechanisch, d.h. im Prinzip von einer geeigneten Maschine durchführbar sein. Dies setzt voraus, dass in den einzelnen Schritten nur *elementare Operationen* auszuführen sind, die aus einer gegebenen endlichen Menge von Operationen stammen, die lokal also insbesondere in endlicher Zeit mit endlichem Speicherbedarf ausgeführt werden können. Jede dieser Operationen beschreibt eine eindeutige Abbildung, sodass eine eindeutige Festlegung der Schrittfolge ein eindeutiges Ein-/Ausgabeverhalten und damit die Determiniertheit garantiert. Ein Algorithmus, dessen Schrittfolge (durch die Eingabe) festgelegt ist, heißt *deterministisch*. Wird ein Schritt nach dem anderen ausgeführt, so spricht man von einem *sequentiellen Algorithmus*.

Bei nicht-sequentiellen, d.h. *parallelen* oder *verteilten* Algorithmen können mehrere Schritte gleichzeitig ausgeführt werden, wozu dann allerdings zur Ausführung mehrere Rechnende herangezogen werden müssen. Durch Parallelität kann die Ausführung der Aufgaben beschleunigt werden, es werden jedoch keine neuen Problemklassen hierdurch lösbar, da jeder parallele Algorithmus in einen sequentiellen Algorithmus mit demselben Ein-/Ausgabeverhalten überführt werden kann. Bei nicht-deterministischen Algorithmen können für einen Schritt mehrere Operationen zur Auswahl stehen. Wird diese Wahl zufällig durchgeführt, spricht man von *probabilistischen Algorithmen*. Solange der Algorithmus trotz dieser Mehrdeutigkeiten determiniert ist, können wir ihn

einfach deterministisch simulieren, indem wir jeweils die erstmögliche Operation ausführen. Sinnvoll werden nichtdeterministische und probabilistische Algorithmen in der Regel daher nur dann, wenn man die Forderung der Determiniertheit aufgibt. Man muss dann allerdings festlegen, welches der Ergebnisse das richtige sein soll. Bei probabilistischen Algorithmen geht das z.B. über eine Mehrheitsentscheidung, während sonst bei nichtdeterministischen Algorithmen mit JA/NEIN-Antworten das JA meist als dominierend angesehen wird, also die richtige Antwort ist, falls mindestens eine mögliche Schrittfolge zu diesem Ergebnis führt. Man kann auch hier wiederum das Verfahren durch einen klassischen, d.h. deterministischen und sequentiellen Algorithmus simulieren, indem man alle möglichen Rechnungen nacheinander ausführt.

Die obigen Überlegungen zeigen, dass wir uns bei Untersuchungen zur Reichweite der algorithmischen Methode auf die Untersuchung klassischer (d.h. deterministischer, sequentieller) Algorithmen beschränken können. Interessant werden die anderen Modelle erst bei Fragen der Komplexität, die wir jedoch erst später behandeln werden.

Im Folgenden werden also zunächst alle Algorithmen als sequentiell und – wenn nicht explizit anders festgelegt – als deterministisch angenommen.

Als Nächstes wollen wir uns die Aufgaben, die ein Algorithmus lösen soll, näher ansehen. Wir werden diese in drei Gruppen einteilen und dadurch drei entsprechende Typen von Algorithmen erhalten, nämlich *Berechnungsverfahren*, *Entscheidungsverfahren* und *Aufzählungsverfahren*.

Die erste typische Aufgabe für einen Algorithmus ist die Berechnung einer Funktion. Wohlbekannte Beispiele hierfür sind der Schulalgorithmus zur Addition zweier natürlicher Zahlen ( $+$  :  $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ ) und der Euklidische Algorithmus zur Bestimmung des größten gemeinsamen Teilers ( $ggT$  :  $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ ). Solche Algorithmen nennen wir *Berechnungsverfahren* (BV) und wir sagen, dass eine Funktion  $f : I \rightarrow O$  *berechenbar* ist, falls es ein Berechnungsverfahren für  $f$  gibt. Hierbei müssen die Ein- und Ausgaben *Daten* sein, d.h. endliche Beschreibungen von Objekten. So muss man beachten, dass zahlentheoretische Algorithmen nicht auf den Zahlen selbst, sondern auf deren Darstellungen operieren. So ist der o.g. Additionsalgorithmus für die Dezimaldarstellung entworfen und ist für die Unärdarstellung ungeeignet. (Hier erhält man die Summe zweier Zahlen durch deren Verkettung.)<sup>1</sup> Da Wörter ein universelles Beschreibungsmittel bieten, gehen wir davon aus, dass Daten stets Wörter sind. Wir betrachten also nur die Frage der Berechenbarkeit von Wortfunktionen  $f : \Sigma^* \rightarrow T^*$  und mehrstelligen Wortfunktionen  $f : \Sigma_1^* \times \dots \times \Sigma_n^* \rightarrow T^*$ , wobei wir uns in den folgenden Überlegungen meist auf den Fall 1-stelliger Funktionen beschränken. (Die Verallgemeinerung auf mehrstellige Funktionen ist in der Regel einfach und eine gute Übung!)

Eine Funktion  $f : \mathbb{N} \rightarrow \mathbb{N}$  nennen wir also *berechenbar*, wenn die zugehörige Abbildung der Unärdarstellungen ( $\hat{f} : \{1\}^* \rightarrow \{1\}^*$  mit  $\hat{f}(\text{Un}(n)) = \text{Un}(f(n))$ ) berechenbar ist. (Da es Berechnungsverfahren zum Übersetzen der Zahldarstellungen gibt, könnten wir hier genauso gut die Binär- oder Dezimaldarstellung wählen.)

Einem Berechnungsverfahren  $\mathcal{B}$  sieht man i. Allg. nicht an, ob es für eine Eingabe tatsächlich eine Ausgabe liefert. Wir erläutern dies an einem Beispiel. Bekanntlich kann man effektiv feststellen, ob eine Zahl eine Primzahl ist („Sieb des Eratosthenes“). Hieraus lässt sich ein Berechnungsverfahren gewinnen, das zu jeder Zahl  $n \geq 1$  die kleinste Primzahl  $npz(n)$ , die  $\geq 2n + 1$  ist, berechnet:

<sup>1</sup>Bei der Vorschrift zum g.g.T. wird diese Abhängigkeit nicht unmittelbar klar. Diese stützt sich jedoch auf andere Algorithmen (Multiplikation, Division) ab, die wiederum auf die Addition zurückgreifen. Die Beobachtung, dass nur die Algorithmen für einige Grundfunktionen darstellungsabhängig sind, führt zum Konzept des *Datentyps*. Dort nimmt man zu der Objektmenge gewisse Grundfunktionen hinzu und kann dann von der konkreten Darstellung abstrahieren (s. Vorlesung „Informatik 1“).

```

Eingabe(n)
p: = 2n+1;
while p ist keine Primzahl do p: = p+2;
Ausgabe(p)

```

Da es unendlich viele Primzahlen gibt, wird die while-Schleife schließlich verlassen und  $p = npz(n)$  ausgegeben. Ersetzt man die while-Schleife durch

```

while (p oder p+2 ist keine Primzahl) do p: = p+2;

```

so erhält man ein Verfahren zur Berechnung des kleinsten Primzahl-Zwilling oberhalb von  $2n$ . Hier weiß man allerdings nicht, ob es unendlich viele solche Zwillinge gibt, also für jede Eingabe  $n$  die while-Schleife verlassen und eine Ausgabe geliefert wird.

Wir sagen, dass der Algorithmus  $\mathcal{A}$  bei Eingabe  $x$  *terminiert*, falls er bei Eingabe  $x$  eine Ausgabe liefert. Der Algorithmus  $\mathcal{A}$  ist *total*, falls er bei jeder Eingabe terminiert.

Wir werden später sehen, dass die Frage der Terminierung eines Algorithmus ein grundsätzliches Problem ist, das selbst nicht algorithmisch entschieden werden kann („Unlösbarkeit des Halteproblems“). Für eine systematische Betrachtung der algorithmischen Berechenbarkeit können wir uns daher nicht auf totale Funktionen beschränken, sondern müssen partielle Funktionen betrachten.

Eine *partielle Funktion*  $\varphi : I \rightarrow O$  auf  $I$  ist eine Funktion, deren Definitionsbereich  $Db(\varphi)$  in  $I$  und deren Wertebereich  $Wb(\varphi)$  in  $O$  enthalten ist. Dabei ist

$$\begin{aligned}
 Db(\varphi) &= \{x \in I : \varphi(x) \text{ ist definiert}\} \\
 Wb(\varphi) &= \{\varphi(x) : x \in Db(\varphi)\}
 \end{aligned}$$

Ist  $\varphi(x)$  definiert, so sagen wir auch, dass  $\varphi(x)$  *konvergiert* und schreiben  $\varphi(x) \downarrow$ ; andernfalls *divergiert*  $\varphi(x) : \varphi(x) \uparrow$ . Ist  $Db(\varphi) = I$ , so heißt  $\varphi$  *total*. Mit  $\varphi(x) = y$  bezeichnen wir, dass  $\varphi(x)$  definiert ist und den Wert  $y$  annimmt. Die Gleichheit  $\varphi(x) = \psi(y)$  bedeutet, dass entweder  $\varphi(x)$  und  $\psi(y)$  beide undefiniert sind oder beide definiert sind und denselben Wert annehmen. Gilt  $\varphi(x) = \psi(x)$  für alle  $x \in I$ , so schreiben wir  $\varphi = \psi$ . In der Regel werden wir kleine griechische Buchstaben (insbesondere  $\phi, \psi$ ) für partielle und kleine lateinische Buchstaben (insbesondere  $f, g, h$ ) für totale Funktionen verwenden. Wir sagen, dass  $\varphi$  *partiell berechenbar* ist, wenn es ein Berechnungsverfahren für  $\varphi$  gibt. Für totales  $\varphi$  sind also Berechenbarkeit und partielle Berechenbarkeit identisch.

Die zwei weiteren Typen von Algorithmen, die wir betrachten, charakterisieren Mengen. Da wir generell nur Wörter als Daten betrachten, sind diese also Darstellungen von Sprachen.

Ein *Entscheidungsverfahren* (EV)  $\mathcal{E}$  für eine Sprache  $L \subseteq \Sigma^*$  ist ein Algorithmus, der bei Eingabe eines Wortes  $x \in \Sigma^*$  feststellt, ob dieses zu  $L$  gehört (Ausgabe: JA) oder nicht (Ausgabe: NEIN). Wir sagen entsprechend, dass  $\mathcal{E}$  die Eingabe  $x$  *akzeptiert* bzw. *verwirft*. Ein Beispiel für ein EV ist der oben bereits angesprochene Primzahltest. Manchmal betrachtet man auch *partielle Entscheidungsverfahren* (PEV) für eine Sprache  $L$ , die bei Eingabe  $x$  im positiven Falle wiederum mit JA antworten, im negativen Falle entweder wiederum mit NEIN antworten oder aber keine Antwort liefern, d.h. nicht terminieren. Gibt es ein (partielles) Entscheidungsverfahren für eine Sprache, so heißt diese (*partiell*) *entscheidbar*.

Ein *Aufzählungsverfahren* (AV)  $\mathcal{A}$  für eine Sprache  $L \subseteq \Sigma^*$  ist ein Algorithmus, der (ohne Eingabe) die Elemente von  $L$  in beliebiger Reihenfolge und eventuell mit Wiederholungen ausgibt. Erfolgt die Ausgabe in der richtigen Reihenfolge bzgl. der Längen-lexikographischen Ordnung, so heißt das Verfahren *monoton*. Entsprechend

heißt  $L$  (*monoton*) *aufzählbar*,<sup>2</sup> wenn es ein (monotones) Aufzählungsverfahren für  $L$  gibt.

Zwischen Berechnungsverfahren, Entscheidungsverfahren und Aufzählungsverfahren gibt es enge Zusammenhänge, die sich in den Beziehungen zwischen den abgeleiteten Begriffen der Berechenbarkeit, Entscheidbarkeit und Aufzählbarkeit widerspiegeln. Im Folgenden geben wir die wichtigsten Beziehungen an.

Für alle diese Untersuchungen ist es wesentlich, dass die Abwicklung eines Algorithmus aus einer Folge endlicher Schritte besteht. Wir können daher jeden Algorithmus  $\mathcal{A}$  mit einem Schrittzähler ausstatten und effektiv feststellen, was der Algorithmus innerhalb einer vorgegebenen Schrittzahl geleistet hat. Insbesondere sind für ein Berechnungs- (oder (partielles) Entscheidungs-)verfahren  $\mathcal{B}$  und ein Aufzählungsverfahren  $\mathcal{A}$  folgende Mengen entscheidbar:

$$\begin{aligned} & \{(x, w_n) : \mathcal{B} \text{ terminiert bei Eingabe } x \text{ in } \leq n \text{ Schritten}\} \\ & \{(x, y, w_n) : \mathcal{B} \text{ terminiert bei Eingabe } x \text{ in } \leq n \text{ Schritten und gibt } y \text{ aus}\} \\ & \{(x, w_n) : \mathcal{A} \text{ gibt innerhalb der ersten } n \text{ Schritte } x \text{ aus}\} \end{aligned}$$

Diese Beobachtung wird in den folgenden Überlegungen an vielen Stellen implizit verwendet.

Wie man leicht sieht, kann man ein EV  $\mathcal{E}$  für eine Sprache  $L \subseteq \Sigma^*$  stets in ein AV  $\mathcal{A}$  für  $L$  umformen:  $\mathcal{A}$  geht hierbei alle Wörter  $w$  von  $\Sigma^*$  der Reihe nach (bzgl.  $\leq_{ll}$ ) durch, testet mit Hilfe von  $\mathcal{E}$  für jedes  $w$ , ob  $w \in L$  gilt und gibt im positiven Fall  $w$  aus. Dieses Verfahren ist sogar monoton, weshalb wir gezeigt haben:

2.1 LEMMA. *Jede entscheidbare Sprache ist monoton aufzählbar, also insbesondere aufzählbar.*  $\square$

Für *monotone* Aufzählungen gilt auch die Umkehrung. Der Beweis benutzt die folgende Beobachtung.

2.2 LEMMA. *Jede endliche Menge  $L$  ist entscheidbar.*

BEWEIS. Ein EV für  $L$  benutzt eine (endliche) Tabelle mit den Elementen von  $L$ . Die Eingabe  $w$  wird mit den Einträgen der Tabelle verglichen. Wird eine Übereinstimmung gefunden, wird akzeptiert, sonst verworfen.  $\square$

2.3 SATZ.  $\dagger$  *Eine Sprache  $L$  ist genau dann entscheidbar, wenn sie monoton aufzählbar ist.*

BEWEIS. Wegen Lemma 2.1 genügt es aus einem monotonen AV  $\mathcal{A}$  für  $L$  ein EV  $\mathcal{E}$  für  $L$  abzuleiten. Dabei dürfen wir wegen Lemma 2.2 davon ausgehen, dass  $L$  unendlich ist. Das Verfahren  $\mathcal{E}$  arbeitet bei Eingabe  $x$  wie folgt: Es simuliert  $\mathcal{A}$  bis dieses erstmals ein Wort  $y$  mit  $x <_{ll} y$  ausgibt. Ist  $x$  bei den von  $\mathcal{A}$  zuvor ausgegebenen Wörtern, so gibt  $\mathcal{E}$  JA aus, sonst NEIN.  $\square$

<sup>2</sup>Man beachte den Unterschied zwischen „abzählbar“ und „aufzählbar“. Offensichtlich ist jede aufzählbare Menge abzählbar, die Umkehrung gilt i. Allg. aber nicht (s. weiter unten). Abzählbarkeit einer Menge  $M$  bedeutet anschaulich, dass die Elemente von  $M$  in (i. Allg. unendliche) Listenform gebracht werden können. Aufzählbarkeit bedeutet, dass man solch eine Liste effektiv beschreiben kann.

Wir werden später mit Hilfe der Formalisierung des Algorithmusbegriffs zeigen, dass in Satz 2.3 die Forderung der *Monotonie* der Aufzählung wesentlich ist: Es gibt aufzählbare Mengen, die nicht entscheidbar sind. Sind jedoch eine Menge  $L$  und ihr Komplement beide aufzählbar, so ist die Menge  $L$  entscheidbar, wie wir als Nächstes zeigen werden. Wir beobachten hierzu zunächst:

2.4 LEMMA. *Eine Sprache  $L$  ist genau dann entscheidbar, wenn ihr Komplement  $\bar{L}$  entscheidbar ist.*

BEWEIS. Aus einem EV für  $L$  ( $\bar{L}$ ) erhält man ein EV für  $\bar{L}$  ( $L$ ), indem man JA- und NEIN-Antworten vertauscht.  $\square$

2.5 SATZ. *Eine Sprache  $L \subseteq \Sigma^*$  ist genau dann entscheidbar, wenn  $L$  und  $\bar{L}$  aufzählbar sind.*

BEWEIS. Aus den Lemmata 2.1 und 2.4 folgt, dass die Entscheidbarkeit von  $L$  die Aufzählbarkeit von  $L$  und  $\bar{L}$  impliziert. Für die Umkehrung seien  $\mathcal{A}$  und  $\mathcal{A}'$  Aufzählungsverfahren für  $L$  bzw.  $\bar{L}$ . Da für jedes Wort  $x \in \Sigma^*$  entweder  $x \in L$  oder  $x \in \bar{L}$ , wird  $x$  entweder von  $\mathcal{A}$  oder von  $\mathcal{A}'$  ausgegeben. Damit ergibt sich folgende Idee für ein EV  $\mathcal{E}$  für  $L$ : Bei Eingabe  $x$ , lässt  $\mathcal{E}$  die Verfahren  $\mathcal{A}$  und  $\mathcal{A}'$  parallel laufen, bis eines der Verfahren  $x$  ausgibt. Gibt  $\mathcal{A}$   $x$  aus, so gibt  $\mathcal{E}$  JA aus; sonst NEIN.  $\square$

Ähnlich können wir zeigen, dass jede aufzählbare Menge  $L$  partiell entscheidbar ist. Hierzu simuliert das PEV  $\mathcal{P}$  für  $L$  bei Eingabe  $x$  ein AV  $\mathcal{A}$  für  $L$ , bis dieses  $x$  ausgibt, und terminiert dann mit Ausgabe JA. (Wird  $x$  nicht aufgezählt, terminiert  $\mathcal{P}$  nicht, was ja bei  $x \notin L$  zulässig ist.)

2.6 LEMMA. *Jede aufzählbare Sprache ist partiell entscheidbar.*  $\square$

Hier gilt wiederum auch die Umkehrung:

2.7 LEMMA. *Jede partiell entscheidbare Sprache  $L \subseteq \Sigma^*$  ist aufzählbar.*

BEWEIS. Aus einem PEV  $\mathcal{P}$  für  $L$  erhalten wir ein AV  $\mathcal{A}$  für  $L$  wie folgt.  $\mathcal{A}$  führt die folgenden Schrittfolgen  $\mathcal{A}_n$  für alle  $n \in \mathbb{N}$  induktiv aus. Dabei ist jede Folge  $\mathcal{A}_n$  endlich, so dass alle  $\mathcal{A}_n$  tatsächlich durchlaufen werden. In der Schrittfolge  $\mathcal{A}_n$  wird für jedes der ersten  $n$  Wörter  $w_i$ ,  $0 \leq i < n$ , der Reihe nach das PEV  $\mathcal{P}$  für Eingabe  $w_i$  maximal  $n$  Schritte simuliert. Akzeptiert  $\mathcal{P}$   $w_i$  innerhalb dieser Schrittzahl, so wird  $w_i$  ausgegeben.

Es ist klar, dass  $\mathcal{A}$  nur Wörter, die von  $\mathcal{P}$  akzeptiert werden, d.h. die in  $L$  liegen, ausgibt. D.h. für die von  $\mathcal{A}$  aufgezählte Sprache  $L(\mathcal{A})$  gilt  $L(\mathcal{A}) \subseteq L$ . Zum Nachweis der Umkehrung sei  $x \in L$ . Dann gibt es  $n, m \in \mathbb{N}$ , so dass  $x = w_n$  und  $\mathcal{P}$  gibt  $x$  innerhalb der ersten  $m$  Schritte aus. Für jedes  $k \geq n + 1, m$  gibt  $\mathcal{A}_k$  dann aber  $x$  aus, weshalb  $x \in L(\mathcal{A}_k) \subseteq L(\mathcal{A})$  gilt.  $\square$

2.8 SATZ. *Eine Sprache  $L$  ist genau dann aufzählbar, wenn sie partiell entscheidbar ist.*  $\square$

Der Begriff der Aufzählbarkeit lässt sich schließlich auf den der Entscheidbarkeit zurückführen. Hierzu stellt man aufzählbare Mengen als Projektionen (mehrstelliger) entscheidbarer Mengen dar. Hierbei heißt eine  $n$ -stellige Menge  $A \subseteq I^n$  die *Projektion* der  $(n + 1)$ -stelliger Menge  $B \subseteq I^{n+1}$ , falls

$$\forall \vec{x} \in I_n (\vec{x} \in A \Leftrightarrow \exists y \in I ((\vec{x}, y) \in B)).$$

2.9 SATZ. Eine Sprache  $L \subseteq \Sigma^*$  ist genau dann aufzählbar, wenn sie Projektion einer entscheidbaren Sprache  $L' \subseteq \Sigma^* \times \Sigma^*$  ist.

BEWEISIDEE. „ $\Rightarrow$ “ Für ein AV  $\mathcal{A}$  von  $L$  definiert man

$$L' = \{(x, w_n) : \mathcal{A} \text{ zählt } x \text{ innerhalb der ersten } n \text{ Schritte auf}\}$$

„ $\Leftarrow$ “ Ist  $L$  die Projektion der entscheidbaren Menge  $L'$ , so erhält man ein PEV für  $L$ , indem man bei Eingabe  $x$  mit Hilfe eines EV für  $L'$  induktiv für  $n = 0, 1, \dots$  prüft, ob  $(x, w_n) \in L'$  gilt, und akzeptiert, sobald solch ein  $w_n$  gefunden wird.  $\square$

Satz 2.9 erlaubt eine aufzählbare Menge  $L$  als ein *unbeschränktes Suchproblem* zu verstehen, dessen Lösungsraum *entscheidbar* ist, d.h. bei dem die Korrektheit einer möglichen Lösung effektiv überprüfbar ist. (Hierbei lässt sich die Größe einer möglichen Lösung i. Allg. nicht effektiv beschränken. Siehe Übungen.)

Mit Hilfe der charakteristischen Funktion einer Menge können wir die Entscheidbarkeit auf die Berechenbarkeit zurückführen. Für die Umkehrung müssen wir die Graphen einer Funktion betrachten.

Für eine Menge  $M \subseteq I$  ist die *charakteristische Funktion*  $c_M : I \rightarrow \Sigma_2$  von  $M$  definiert durch

$$c_M(x) = \begin{cases} 1 & \text{falls } x \in M \\ 0 & \text{sonst} \end{cases}$$

Mitunter betrachten wir auch die *partielle charakteristische Funktion*  $\chi_M : I \rightarrow \Sigma_1$ , die durch

$$\chi_M(x) = \begin{cases} 1 & \text{falls } x \in M \\ \uparrow & \text{sonst} \end{cases}$$

definiert ist. Der *Graph*  $G_\varphi$  einer (partiellen) Funktion  $\varphi : I \rightarrow O$  ist die Menge

$$G_\varphi = \{(x, \varphi(x)) : x \in \text{Db}(\varphi)\} \subseteq I \times O$$

2.10 SATZ. Eine Sprache  $L$  ist genau dann entscheidbar, wenn ihre charakteristische Funktion  $c_L$  berechenbar ist.

BEWEIS. Ersetzt man in einem EV für  $L$  die JA- und NEIN-Antworten durch 1 bzw. 0, so erhält man ein BV für  $c_L$  (und umgekehrt).  $\square$

Entsprechend kann man die Aufzählbarkeit einer Menge durch die partielle Berechenbarkeit ihrer partiellen charakteristischen Funktion beschreiben. Hierzu zeigt man zunächst, dass der Definitionsbereich einer partiell berechenbaren Funktion aufzählbar ist.

2.11 LEMMA. Für partiell berechenbares  $\varphi : \Sigma^* \rightarrow T^*$  ist  $\text{Db}(\varphi)$  aufzählbar.

BEWEIS. Für ein BV  $\mathcal{B}$  von  $\varphi$  ist die Menge

$$D = \{(x, w_n) : \text{Bei Eingabe } x \text{ terminiert } \mathcal{B} \text{ in } \leq n \text{ Schritten.}\}$$

entscheidbar und  $\text{Db}(\varphi)$  ist die Projektion von  $D$ .  $\square$

2.12 SATZ. *Eine Sprache  $L$  ist genau dann aufzählbar, wenn ihre partielle charakteristische Funktion partiell berechenbar ist.*

BEWEIS. Ist  $L$  aufzählbar und  $\mathcal{A}$  ein Aufzählungsverfahren, so erhält man ein Berechnungsverfahren  $\mathcal{B}$  für  $\chi_L$  wie folgt: Bei Eingabe  $x$  simuliert  $\mathcal{B}$   $\mathcal{A}$  bis dieses  $x$  ausgibt.  $\mathcal{B}$  terminiert dann mit Ausgabe 1. Wird  $x$  von  $\mathcal{A}$  nie ausgegeben, terminiert  $\mathcal{B}$  nicht.

Da  $L = Db(\chi_L)$  folgt die Umkehrung aus Lemma 2.11.  $\square$

Da  $L$  gerade der Definitionsbereich der partiellen charakteristischen Funktion von  $L$  ist, ergibt sich unmittelbar aus Lemma 2.11 und Satz 2.12 der folgende weitere Zusammenhang zwischen Aufzählbarkeit und (partieller) Berechenbarkeit.

2.13 KOROLLAR. *Eine Sprache ist genau dann aufzählbar, wenn sie der Definitionsbereich einer partiell berechenbaren Funktion ist.*  $\square$

In den nächsten beiden Sätzen charakterisieren wir die totale bzw. partielle Berechenbarkeit mit Hilfe der Entscheidbarkeit bzw. Aufzählbarkeit.

2.14 SATZ. *Eine totale Funktion  $f : \Sigma^* \rightarrow T^*$  ist genau dann berechenbar, wenn ihr Graph  $G_f \subseteq \Sigma^* \times T^*$  entscheidbar ist.*

BEWEIS. „ $\Rightarrow$ “ Aus einem BV  $\mathcal{B}$  für  $f$  erhält man ein EV  $\mathcal{E}$  für  $G_f$  wie folgt: Bei Eingabe  $(x, y)$  berechnet  $\mathcal{E}$  durch Simulation von  $\mathcal{B}$  den Wert  $f(x)$  und akzeptiert g.d.w.  $f(x) = y$ .

„ $\Leftarrow$ “ Aus einem EV  $\mathcal{E}$  für  $G_f$  erhält man ein BV  $\mathcal{B}$  für  $f$  wie folgt. Bei Eingabe  $x$ , simuliere sukzessive  $\mathcal{E}$  für Eingabe  $(x, w_i)$  für wachsendes  $i = 0, 1, \dots$  bis  $\mathcal{E}$  diese Eingabe erstmals akzeptiert. Gebe dann das entsprechende  $w_i$  aus.  $\square$

2.15 SATZ. *Eine partielle Funktion  $\varphi : \Sigma^* \rightarrow T^*$  ist genau dann partiell berechenbar, wenn ihr Graph  $G_\varphi \subseteq \Sigma^* \times T^*$  aufzählbar ist.*

BEWEIS. „ $\Rightarrow$ “ Wie im Beweis der entsprechenden Aussage von Satz 2.14 erhält man aus einem BV  $\mathcal{B}$  für  $\varphi$  ein *partielles* EV für  $G_\varphi$ . Die Behauptung folgt dann mit Satz 2.8.

„ $\Leftarrow$ “ Aus einem AV  $\mathcal{A}$  für  $G_\varphi$  erhält man ein BV  $\mathcal{B}$  für  $\varphi$  wie folgt: Bei Eingabe  $x$  simuliere  $\mathcal{A}$  bis erstmals ein Paar  $(x, y)$  mit 1. Komponente  $x$  ausgegeben wird. Gebe dann das zugehörige  $y$  aus. (Wird solch ein Paar  $(x, y)$  nie aufgezählt, terminiert  $\mathcal{B}$  nicht.)  $\square$

(Die Sätze 2.14 und 2.15 zeigen, dass für Graphen totaler Funktionen die Entscheidbarkeit und Aufzählbarkeit gleichwertig sind. Für beliebige Menge gilt dies – wie oben bereits erwähnt – jedoch nicht.)

Zuletzt charakterisieren wir noch die Aufzählbarkeit durch die Berechenbarkeit. Hierzu definieren wir zunächst: Eine *Aufzählungsfunktion*  $f$  für eine Menge  $L \subseteq \Sigma^*$  ist eine totale berechenbare Funktion  $f : \mathbb{N} \rightarrow \Sigma^*$  mit  $Wb(f) = L$ .

2.16 SATZ. *Eine nichtleere Menge  $L \subseteq \Sigma^*$  ist genau dann aufzählbar, wenn sie eine Aufzählungsfunktion besitzt. Ist  $L$  unendlich, kann diese injektiv gewählt werden.*

BEWEIS. „ $\Rightarrow$ “ Ist  $L$  endlich, etwa  $L = \{v_0, \dots, v_n\}$ , so ist  $f$  mit  $f(m) = v_m$  für  $m \leq n$  und  $f(m) = v_n$  für  $m > n$  eine Aufzählungsfunktion von  $L$ . Für unendliches  $L$  wählt man  $f(n)$  als das  $(n+1)$ -te Wort, das ein Aufzählungsverfahren für  $L$  ausgibt.

„ $\Leftarrow$ “ Ist  $f$  Aufzählungsfunktion von  $L$ , so erhält man ein Aufzählungsverfahren für  $L$ , indem man  $f$  sukzessive für alle  $n \geq 0$  berechnet und  $f(n)$  ausgibt.  $\square$

Da die monoton aufzählbaren Mengen gerade die entscheidbaren Mengen sind, erhält man analog eine Beschreibung der Entscheidbarkeit über monotone Aufzählungsfunktionen. Dabei ist  $f: \Sigma^* \rightarrow T^*$  *schwach* [*streng*] *monoton*, falls

$$\forall x, y \in \Sigma^* (x <_U y \Rightarrow f(x) \leq_U f(y) [f(x) <_U f(y)]).$$

2.17 SATZ. Eine nichtleere Menge  $L \subseteq \Sigma^*$  ist genau dann entscheidbar, wenn sie eine schwach monotone Aufzählungsfunktion besitzt. Ist  $L$  unendlich, kann diese streng monoton gewählt werden.  $\square$

Die oben nachgewiesenen Beziehungen zwischen den Begriffen Berechenbarkeit, Entscheidbarkeit und Aufzählbarkeit zeigen, dass man diese Begriffe wechselseitig aufeinander zurückführen kann. Wir werden dies bei der Formalisierung des Algorithmusbegriffs benutzen und nur den Begriff des Berechnungsverfahrens formalisieren. Hierzu werden wir zunächst im nächsten Abschnitt das allgemeine Konzept einer mathematischen Maschine zur Beschreibung des schematischen Ablaufs von Algorithmen einführen. Durch geeignete Wahl der Grundoperationen werden wir dann im übernächsten Abschnitt einen konkreten Typus mathematischer Maschinen angeben, der als adäquate Formalisierung des intuitiven Algorithmusbegriffes allgemein akzeptiert ist.