

2. Algorithmen

Typen von Algorithmen:

Entscheidungs-, Aufzählungs- und Berechnungsverfahren

Anforderungen an Algorithmen

Berechenbarkeit, Entscheidbarkeit, Aufzählbarkeit:

Alternative Charakterisierungen und Vergleich

Ein *Algorithmus (Rechenvorschrift)* ist eine Vorschrift zur Lösung eines Problems.

Typen von Problemen / Algorithmen:

- Entscheidungsprobleme / Entscheidungsverfahren
- Aufzählungsprobleme / Aufzählungsverfahren
- Berechnungsprobleme / Berechnungsverfahren

2.1. Typen von Algorithmen: Entscheidungs-, Aufzählungs- und Berechnungsverfahren

Entscheidungsprobleme / Entscheidungsverfahren

Entscheidungsproblem: Stelle fest, ob ein aus einer Grundmenge von Daten gegebenes Datum eine gewisse Eigenschaft E hat.

Solch ein Problem kann durch eine Sprache L beschrieben werden: $L = \{x \in \Sigma^* : E(x)\}$

Entscheidungsverfahren (EV): Algorithmus zur Lösung eines Entscheidungsproblems

- Eingabe: $x \in \Sigma^*$
- Ausgabe: JA (falls $E(x)$) bzw. NEIN (falls $\neg E(x)$)

L ist *entscheidbar*, wenn es ein Entscheidungsverfahren für L gibt.

Anmerkungen:

- Ist \mathcal{E} ein EV für die Sprache L , so nennen wir L die von \mathcal{E} *akzeptierte* oder *erkannte Sprache*. Wir sagen, dass \mathcal{E} x *akzeptiert* (*verwirft*), falls das Verfahren die Antwort JA (NEIN) ausgibt.

- Oft bezeichnen wir Sprachen auch einfach als Mengen oder Probleme.

- Probleme über natürliche Zahlen identifizieren wir mit Sprachen, indem wir Zahlen unär kodieren:

$$\underline{n} := 0^{n+1} \hat{=} n$$

$$A \hat{=} \tilde{A} = \{\underline{n} : n \in A\} = \{0^{n+1} : n \in A\}$$

Aufzählungsprobleme / Aufzählungsverfahren

Aufzählungsproblem: Liste alle Daten aus einer Grundmenge mit einer gewissen Eigenschaft E auf.

Solch ein Problem kann (wie ein Entscheidungsproblem) durch eine Sprache L beschrieben werden: $L = \{x \in \Sigma^* : E(x)\}$

Aufzählungsverfahren (AV): Algorithmus zur Lösung eines Aufzählungsproblems

- Eingabe: keine
- Ausgabe: Alle Wörter mit Eigenschaft E , aufgelistet in beliebiger Reihenfolge (und eventuell mit Wiederholungen).

L ist *aufzählbar*, wenn es ein Aufzählungsverfahren für L gibt.

BEISPIEL

Die Menge

$$A = \{n : n \text{ gerade}\} = \{2n : n \geq 0\}$$

ist entscheidbar.

Entscheidungsverfahren für A :

- Eingabe: \underline{n}
- ```
w := n;
while |w| ≥ 2 do
 begin
 w := tail(tail(w))
 end;
if |w| = 1 then output(JA) else output(NEIN).
```

Hierbei ist die tail-Funktion durch  $tail(\lambda) = \lambda$  und  $tail(a\lambda) = \lambda$  definiert.

## Spezielle Aufzählungsverfahren

Ein *monotones AV* gibt die Wörter der Größe nach geordnet (bzgl. der Längen-lexikographischen Ordnung) aus.

$L$  ist *monoton aufzählbar*, wenn es ein monotones Aufzählungsverfahren für  $L$  gibt.

Die monotone Aufzählbarkeit für Zahlen ist entsprechend definiert.

## BEISPIEL

Die Menge

$$A = \{n : n \text{ gerade}\} = \{2n : n \geq 0\}$$

ist monoton aufzählbar.

Monotones Aufzählungsverfahren für  $\hat{A}$ :

- ```
w := 0 (= 0);
while 1 = 1 do
  begin
    output(w);
    w := w02
  end.
```

BEISPIEL

Die (2-stellige) Addition

$$f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

$$f(m, n) = m + n$$

ist berechenbar.

Berechnungsverfahren für \hat{f} mit $\hat{f}(\underline{m}, \underline{n}) = \underline{m + n}$:

- Eingabe: $x, y \in \{0\}^+$
- $z := \text{tail}(xy)$;
 $\text{output}(z)$.

Berechnungsprobleme / Berechnungsverfahren

Berechnungsproblem: Ordne den Daten aus einer Grundmenge andere Daten zu.

Solch ein Problem kann durch eine Wortfunktion $f : \Sigma^* \rightarrow \Sigma^*$ beschrieben werden.

Berechnungsverfahren (BV): Algorithmus zur Lösung eines Berechnungsproblems

- Eingabe: $x \in \Sigma^*$
- Ausgabe: $f(x)$

f ist *berechenbar*, wenn es ein Berechnungsverfahren für f gibt.

BEISPIEL

Die (2-stellige) Multiplikation

$$g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

$$g(m, n) = m \cdot n$$

ist berechenbar.

Berechnungsverfahren für \hat{g} mit $\hat{g}(\underline{m}, \underline{n}) = \underline{m \cdot n}$:

- Eingabe: $x, y \in \{0\}^+$
- $z := 0 (= 0)$
while $|y| > 1$ do
 begin
 $z := z \circ \text{tail}(x)$;
 $y := \text{tail}(y)$
 end;
 $\text{output}(z)$.

Partielle Berechnungsverfahren

Eine *partielle* Wortfunktion $\varphi : \Sigma^* \rightarrow \Sigma^*$ ist eine Funktion $\varphi : D \rightarrow \Sigma^*$ mit $D \subseteq \Sigma^*$. D heisst der *Definitionsbereich* von φ und wird mit $Db(\varphi)$ bezeichnet.

Notation:

- $\varphi(x) \downarrow$ ($\varphi(x)$ *definiert*), falls $x \in Db(\varphi)$
- $\varphi(x) \uparrow$ ($\varphi(x)$ *undefiniert*), falls $x \notin Db(\varphi)$

Partielles Berechnungsverfahren (PBV): Algorithmus zur Lösung eines partiellen Berechnungsproblems

- Eingabe: $x \in \Sigma^*$
- Ausgabe: $\varphi(x)$, falls $\varphi(x) \downarrow$

φ ist *partiell berechenbar*, wenn es ein partielles Berechnungsverfahren für φ gibt.

BEISPIEL

Die Primzahlzwillingsfunktion $pzzf : \mathbb{N} \rightarrow \mathbb{N}$ ordnet jeder Zahl $n \geq 1$ die kleinste ungerade Zahl $m > 2n$ zu, sodass m und $m+2$ Primzahlen sind (d.h. $(m, m+2)$ ist der kleinste Primzahlzwilling oberhalb von $2n$).

Die Funktion $pzzf$ ist partiell berechenbar. **Es ist aber nicht bekannt, ob es unendlich viele Primzahlzwillinge gibt, und daher nicht bekannt, ob pzf total berechenbar ist.**

Partielles Berechnungsverfahren für \widehat{pzzf} (unter Verwendung des Primzahltests PZT; vgl. Übungen):

- Eingabe: $x (= \underline{n}) \in \{0\}^+$
- $y := xx (= 2n + 1)$;
while $\neg PZT(y) \vee \neg PZT(y+2)$ do
begin $y := y+2$ end;
output(y).

NB: Sollte es keinen Primzahlzwilling oberhalb von $2n + 1$ geben, wird die Schleife unendlich oft durchlaufen und nicht verlassen.

Partielle Berechnungsverfahren II

Bemerkungen:

- Für *totale* Funktionen fallen Berechenbarkeit und partielle Berechenbarkeit zusammen.
- Ein PBV für eine partielle Funktion φ muss für eine Eingabe $x \notin Db(\varphi)$ keine Ausgabe liefern. In diesem Fall muss das Verfahren nicht terminieren, d.h. darf unendlich lange laufen!

Mehrdimensionale Probleme

Die eingeführten Begriffe lassen sich leicht auf mehrdimensionale (homogene oder inhomogene) Probleme übertragen:

- $L \subseteq \Sigma^* \times \dots \times \Sigma^*$
- $L \subseteq \Sigma_1^* \times \dots \times \Sigma_n^*$
- $f : \Sigma^* \times \dots \times \Sigma^* \rightarrow \Sigma^*$
- $f : \Sigma_1^* \times \dots \times \Sigma_n^* \rightarrow \Sigma_{n+1}^*$

2.2. Anforderungen an Algorithmen

Anforderungen an Algorithmen II

- Die Rechnung besteht aus einer Folge von 'elementaren' Rechenschritten. (*sequentiell*)
- Die Reihenfolge der Schritte ist eindeutig festgelegt. (*deterministisch*)
- In jedem Einzelschritt wird eine elementare Operation (aus einer endlichen Menge von Operationen) ausgeführt. Jede dieser Operationen kann effektiv, in endlicher Zeit ausgeführt werden.

→ "klassischer Algorithmus"

Anforderungen an Algorithmen I

Ein Algorithmus ist eine Rechenvorschrift, die im Prinzip rein mechanisch ausgeführt werden kann, und u.a. folgenden Anforderungen genügt:

- *Fintheit*: endliche Beschreibung
- *Determiniertheit*: Ergebnis hängt nur von der Eingabe ab
- *Effektivität*: Verfahren tatsächlich durchführbar

Anforderungen an Algorithmen III

Eine weitere, häufig genannte Anforderung an Algorithmen ist, dass sie *terminieren*, d.h. nach endlich vielen Schritten stoppen.

Entscheidungs- und Berechnungsverfahren müssen dieser Forderung genügen.

Partielle Berechnungsverfahren und Aufzählungsverfahren terminieren dagegen i.a. nicht. Ein Aufzählungsverfahren für eine unendliche Menge KANN NICHT terminieren, da es eine unendliche Aufgabe erfüllen muss.

Folgerung aus unseren Anforderungen:

Alles was ein Algorithmus in einer vorgegebenen Anzahl von Schritten macht, kann man effektiv erkennen. D.h. die folgenden Probleme sind entscheidbar (\mathcal{A} EV oder BV, \mathcal{A}' AV):

$\{(x, n) : \mathcal{A} \text{ terminiert bei Eingabe } x \text{ in } \leq n \text{ Schritten}\}$

$\{(x, y, n) : \mathcal{A} \text{ terminiert bei Eingabe } x \text{ in } \leq n \text{ Schritten und gibt } y \text{ aus}\}$

$\{(x, n) : \mathcal{A}' \text{ gibt innerhalb der ersten } n \text{ Schritte } x \text{ aus}\}$

NB: Dagegen ist nicht klar, ob man für einen Algorithmus entscheiden kann, ob er bei einer Eingabe terminiert! (Wir werden später sehen, dass dies in der Tat nicht entscheidbar ist!)

- *Probabilistische Algorithmen*: Schrittfolge wird durch Eingabe nicht eindeutig festgelegt. Der jeweils nächste Rechenschritt wird durch ein Zufallsexperiment (Münzwurf, Würfeln) festgelegt.

Probabilistisches EV: Determiniertheit erhält man dadurch, dass man Akzeptanz durch Mehrheitsentscheidung definiert.

Weitere nichtklassische Rechenverfahren: Quantenalgorithmen, "Bio-Algorithmen", "DNA-Algorithmen", ...

NB: Man überlegt sich leicht, dass die oben angegeben nicht-klassischen Algorithmen nicht mächtiger sind als die klassischen Algorithmen. Interessant sind sie, da sie möglicherweise *effizienter (schneller)* sind! (→ Komplexitätstheorie)

EXKURS: Nichtklassische Algorithmen

- *Parallele und verteilte Algorithmen*: Der Algorithmus wird von mehreren "Rechnern" gleichzeitig bearbeitet. (*nicht sequentiell*)

- *Nichtdeterministische Algorithmen*: Schrittfolge wird durch Eingabe nicht eindeutig festgelegt.

Nichtdeterministische Algorithmen sind meist nichtdeterminierte EVs. Um ein eindeutig bestimmtes Ergebnis zu erhalten, legt man fest: Eine Eingabe x wird akzeptiert (d.h. $x \in L$), wenn mindestens eine der möglichen Rechnungen akzeptiert.

2.3. Berechenbarkeit, Entscheidbarkeit, Aufzählbarkeit:

Alternative Charakterisierungen und Vergleich

Als nächstes wollen wir die grundlegenden Begriffen der Berechenbarkeitstheorie, d.h. die

- Entscheidbarkeit
- Aufzählbarkeit
- (partieller) Berechenbarkeit

nochmals etwas näher betrachten und die bestehende Beziehungen zwischen diesen Konzepten aufzeigen.

Mengen \longrightarrow (partielle) Funktionen

- Charakteristische Funktion einer Menge (Sprache) A :

$$c_A(x) = \begin{cases} 1 & \text{falls } x \in A \\ 0 & \text{falls } x \notin A \end{cases}$$

- Partielle charakteristische Funktion einer Menge (Sprache) A :

$$\chi_A(x) = \begin{cases} 1 & \text{falls } x \in A \\ \uparrow & \text{falls } x \notin A \end{cases}$$

(Partielle) Funktionen \longrightarrow Mengen

- Graph einer partiellen Funktion φ :

$$Graph(\varphi) = G_\varphi = \{(x, \varphi(x)) : \varphi(x) \downarrow\}$$

Mengen vs. Funktionen

Da wir es sowohl mit Mengen (Entscheidbarkeit, Aufzählbarkeit) als auch mit (partiellen) Funktionen ((partielle) Berechenbarkeit) zu tun haben, erinnern wir uns zunächst wie man allgemein Mengen durch Funktionen und Funktionen durch Mengen darstellen kann.

2.3.1. Entscheidbare Sprachen

A endlich $\Rightarrow A$ entscheidbar

Lege eine endliche Tabelle der Elemente von A an und bei Eingabe x prüfe, ob x in der Tabelle vorkommt.

A entscheidbar $\Rightarrow \bar{A}$ entscheidbar

Vertausche in EV für A JA- und NEIN-Antworten.

A entscheidbar $\Leftrightarrow c_A$ berechenbar

Ersetze in EV für A JA- und NEIN-Antworten durch 1 bzw. 0 (und umgekehrt).

A entscheidbar $\Leftrightarrow A$ und \bar{A} aufzählbar

(\Rightarrow) Diese Richtung folgt aus den vorhergehenden Beobachtungen:

A entscheidbar $\Rightarrow A$ monoton aufzählbar $\Rightarrow A$ aufzählbar

A entscheidbar $\Rightarrow \bar{A}$ entscheidbar

(\Leftarrow) Um zu entscheiden, ob $x \in A$, simuliere parallel die Aufzählungsverfahren \mathcal{A} und \mathcal{A}' für A bzw. \bar{A} bis eines der Verfahren x ausgibt. Gebe JA aus, falls \mathcal{A} x ausgibt, sonst NEIN.

NB. Es folgt insbesondere, dass jede entscheidbare Sprache aufzählbar ist. Wir werden später zeigen, dass die Umkehrung nicht gilt, d.h. dass es aufzählbare Sprachen gibt, die unentscheidbar sind.

A entscheidbar $\Leftrightarrow A$ monoton aufzählbar

(\Rightarrow)

```
n := 0;
while n ≥ 0 do
  begin
    if wn ∈ A then output(wn);
    n := n+1
  end.
```

(\Leftarrow) O.B.d.A. ist A unendlich. Um zu entscheiden, ob $x \in A$, simuliere das monotone AV für A bis es erstmals x (\Rightarrow JA) oder ein Wort $y > x$ (\Rightarrow NEIN) ausgibt.

NB: Für den Beweis von (\Leftarrow) ist es wesentlich, dass es ein *monotones* AV gibt!

Charakterisierungen Entscheidbarer Sprachen (Zusammenfassung)

A entscheidbar

$\Leftrightarrow \bar{A}$ entscheidbar

$\Leftrightarrow c_A$ berechenbar

$\Leftrightarrow A$ monoton aufzählbar

$\Leftrightarrow A$ und \bar{A} aufzählbar

2.3.2. Aufzählbare Sprachen

Die aufzählbaren Sprachen lassen sich auf viele Arten beschreiben. Wir beginnen mit der partiellen Entscheidbarkeit.

Eine Sprache A ist *partiell entscheidbar* (oder *einseitig entscheidbar*), wenn sie ein *partielles Entscheidungsverfahren* (PEV) besitzt, d.h. es einen Algorithmus gibt, der bei Eingabe x die Ausgabe JA liefert, falls x in A liegt, und andernfalls *keine Ausgabe* liefert (und evtl. auch *nicht terminiert*).

N.B. A entscheidbar $\Rightarrow A$ partiell entscheidbar.

A aufzählbar $\Leftrightarrow A$ partiell entscheidbar

(\Rightarrow) Um $x \in A$? partiell zu entscheiden, simuliere das AV von A bis dieses x ausgibt. Wenn dies geschieht, gebe JA aus und stoppe. (Liegt x nicht A , so terminiert dieses Verfahren nicht und liefert keine Ausgabe.)

(\Leftarrow) Sei \mathcal{P} ein partielles EV für A . Ein AV für A arbeitet wie folgt:

```
n := 0;
while n ≥ 0 do
  begin
    for m = 0, ..., n do
      begin
        if (P akzeptiert w_m in ≤ n Schritten) then output(w_m)
      end;
    n := n+1
  end.
```

“Dove Tailing Argument”

A aufzählbar $\Leftrightarrow \chi_A$ partiell berechenbar

Da A genau dann aufzählbar ist, wenn A partiell entscheidbar ist, genügt es ein PEV für A in ein PBV für χ_A umzuwandeln (und umgekehrt). Hierzu genügt es die Ausgabe JA durch die Ausgabe 1 zu ersetzen (und umgekehrt).

A aufzählbar $\Leftrightarrow A$ Definitionsbereich einer partiell berechenbaren Funktion

(\Rightarrow) $A = Db(\varphi)$

(\Leftarrow) Sei $A = Db(\varphi)$ und \mathcal{B} ein PBV für φ . Man erhält hieraus ein partielles Berechnungsverfahren für χ_A , indem man alle Ausgaben durch 1 ersetzt.

A aufzählbar $\Leftrightarrow A$ Projektion einer (2-dim) entscheidbaren Menge B

Hierbei ist die n -st. Menge A die *Projektion* der $(n+1)$ -st. Menge B , falls gilt:

$$\vec{x} \in A \Leftrightarrow \exists y ((\vec{x}, y) \in B)$$

(\Rightarrow) Für ein AV \mathcal{A} von A definiere

$$B = \{(x, \underline{n}) : x \text{ wird von } \mathcal{A} \text{ innerhalb der ersten } n \text{ Schritte aufgezählt}\}.$$

(\Leftarrow) (Idee) Sei A die Projektion der entscheidbaren Menge B . Dann ist A der Definitionsbereich der partiell berechenbaren Funktion

$$\varphi(x) = \begin{cases} \text{kleinstes } y \text{ mit } (x, y) \in B & \text{falls ex.} \\ \uparrow & \text{sonst} \end{cases}$$

BEISPIELE

- Halteproblem für ein EV: \mathcal{E}
Suche den Schritt s , bei dem \mathcal{E} bei Eingabe x terminiert.
- Beweisbarkeitsproblem für eine mathematische Theorie:
Suche einen Beweis B für den gegebenen Satz φ .
- Postsches Korrespondenzproblem:
Gegeben eine Folge von Wortpaaren $(x_1, y_1), \dots, (x_n, y_n)$
suche eine Folge k_1, \dots, k_m mit $x_{k_1} \dots x_{k_m} = y_{k_1} \dots y_{k_m}$.

Die Charakterisierung der aufzählbaren Probleme als Projektionen entscheidbarer Mengen zeigt: Aufzählbare Probleme sind **Suchprobleme**, bei denen der Suchraum **unbeschränkt** ist und die Korrektheit von **Lösungen entscheidbar** ist.

Aufzählbares Problem
=
Unbeschränktes effektives Suchproblem

BEISPIELE (Fortsetzung)

- Lösbarkeit Diophantischer Gleichungssysteme:
Gegeben eine endliche Folge von k -stelligen Polynomgleichungen mit ganzzahligen Koeffizienten

$$P_1(\vec{x}) = 0, \dots, P_m(\vec{x}) = 0$$

suche eine gemeinsame ganzzahlige Lösung $\vec{x} \in \mathbb{Z}^k$.
(= 10. Hilbertsches Problem)

Dies sind alles Beispiele für unbeschränkte effektive Suchprobleme, d.h. aufzählbare Probleme. Keines dieser Probleme ist jedoch **entscheidbar**! (Wir werden dies z.T. später zeigen.)

A aufzählbar $\Leftrightarrow A = \emptyset$ oder A ist der Wertebereich einer (totalen) berechenbaren Funktion f .

(\Rightarrow) Ist A endlich aber nicht leer, d.h. $A = \{x_0, \dots, x_m\}$, definiert man f durch $f(\underline{n}) = x_n$ für $n \leq m$ und $f(\underline{n}) = x_m$ sonst. Ist A unendlich, definiert man $f(\underline{n}) = n$ -tes von A ausgegebenes Wort, wobei A ein AV für A ist.

(\Leftarrow) Ein AV für A berechnet induktiv die Werte $f(x_0), f(x_1), f(x_2), \dots$ und gibt diese aus.

Ähnlich lässt sich zeigen:

A aufzählbar $\Leftrightarrow A$ endlich oder A ist der Wertebereich einer (totalen) berechenbaren injektiven Funktion f .

A aufzählbar $\Leftrightarrow A$ ist der Wertebereich einer partiell berechenbaren Funktion φ .

2.3.3. (Partiell) Berechenbare Funktionen

Charakterisierungen Aufzählbarer Sprachen (Zusammenfassung)

A aufzählbar

- $\Leftrightarrow A$ partiell (d.h. einseitig) entscheidbar
- $\Leftrightarrow \chi_A$ partiell berechenbar
- $\Leftrightarrow A$ Definitionsbereich einer part. berechenbaren Funktion
- $\Leftrightarrow A$ Projektion einer entscheidbaren (2-dim) Menge
- $\Leftrightarrow A$ Wertebereich einer part. berechenbaren Funktion
- $\Leftrightarrow A = \emptyset$ oder A Wertebereich einer totalen berechenbaren Funktion
- $\Leftrightarrow A$ endlich oder A Wertebereich einer totalen berechenbaren injektiven Funktion

Für partielle Funktionen φ gilt:

φ partiell berechenbar $\Leftrightarrow \text{Graph}(\varphi)$ aufzählbar

(\Rightarrow) Für ein PBV \mathcal{B} für φ ist die Menge

$\{(x, y, \underline{n}) : \mathcal{B} \text{ terminiert bei Eingabe } x \text{ in } \leq n \text{ Schritten und gibt } y \text{ aus}\}$
entscheidbar und $\text{Graph}(\varphi)$ ist die Projektion (entlang der 3. Komponente) dieser Menge.

(\Leftarrow) Ein AV \mathcal{A} für $\text{Graph}(\varphi)$ lässt sich in folgendes PBV \mathcal{B} für φ umwandeln:
Bei Eingabe x simuliere \mathcal{A} bis dieses ein Paar (x, y) mit erster Komponente x ausgibt. (Geschieht dies nie, so terminiert \mathcal{B} nicht und gibt nichts aus.) Gebe y aus.

Für totale Funktionen f gilt:

f berechenbar
 $\Leftrightarrow \text{Graph}(f)$ entscheidbar
 $\Leftrightarrow \text{Graph}(f)$ aufzählbar

(2 \Rightarrow 3) Trivial, da jede entscheidbare Menge aufzählbar ist.

(3 \Rightarrow 2) Ein AV \mathcal{A} für $\text{Graph}(f)$ lässt sich in folgendes EV für $\text{Graph}(f)$ umwandeln: Bei Eingabe (x, y) simuliere \mathcal{A} bis dieses ein Paar (x, y') mit erster Komponente x ausgibt. Gilt $y = y'$ so akzeptiere (x, y) . Andernfalls verwerfe (x, y) . (Man beachte, dass dieses Verfahren wegen der Totalität von f terminiert! Es gibt nämlich zu jedem x (genau) ein y mit $(x, y) \in \text{Graph}(f)$, nämlich $y = f(x)$.)

Rückführung der Entscheidbarkeit auf Aufzählbarkeit und Berechenbarkeit:

A entscheidbar $\Leftrightarrow A$ und \bar{A} aufzählbar

A entscheidbar $\Leftrightarrow c_A$ berechenbar

2.3.4. Beziehungen zwischen den Konzepten

(Zusammenfassung)

Rückführung der Aufzählbarkeit auf Berechenbarkeit und Entscheidbarkeit:

A aufzählbar $\Leftrightarrow A$ Definitionsbereich einer partiell berechenbaren Funktion
 $\Leftrightarrow A$ Wertebereich einer partiell berechenbaren Funktion
 $\Leftrightarrow A = \emptyset$ oder A Wertebereich einer berechenbaren Funktion

A aufzählbar $\Leftrightarrow A$ Projektion einer entscheidbaren Menge

Rückführung der Berechenbarkeit auf Aufzählbarkeit und Entscheidbarkeit:

$$\begin{aligned} f \text{ berechenbar} &\Leftrightarrow G_f \text{ entscheidbar} \\ &\Leftrightarrow G_f \text{ aufzählbar} \\ \varphi \text{ part. berechenbar} &\Leftrightarrow G_f \text{ aufzählbar} \end{aligned}$$