Eine Zusammenstellung aus Prüfungsprotokollen
bei Professor Weihrauch mit (fast) allen Fragen zur

# Prüfungsvorbereitung
# Vordiplomprüfung
# Grundlagen theoretische Informatik

Thomas Schwarze
Thomas.Schwarze@FernUni-Hagen.de

9. September 2000

# Fragen zu theoretischer Informatik


**1 Teil A, Berechenbarkeit** **4**

1.1 Definition für berechenbare Zahlenfunktionen . . . . . 4
1.2 Definition für berechenbare Wortfunktionen . . . . . 4
1.3 Turing–berechenbar = Band–berechenbar . . . . . 4
1.4 Zeitbedarf Bandmaschine im Vergleich zu Turingmaschine . . . . . 5
1.5 Hilfssymbol–Lemma . . . . . 5
1.6 Bandmaschine: *Datenmenge*, *Befehlssatz*, *Konfigurationsmenge* . . . . . 5
1.7 Unterschied „heisst“ berechenbar ↔ „ist“ berechenbar . . . . . 5
1.8 Churchsche–These . . . . . 6
1.9 Beispiel für nicht berechenbare Funktion oder Menge . . . . . 6
1.10 Zusammenhang zwischen berechenbaren Wort– und Zahlfunktionen . . . . . 6
1.11 Register–berechenbar $\Longrightarrow$ Turing–berechenbar . . . . . 6
1.12 Zusammenhang Funktion und Programme? . . . . . 7
1.13 Was sind die Programme bei $\varphi$ ? . . . . . 7
1.14 Zwei wesentliche Anforderungen an Programmiersprachen . . . . . 7
1.15 Was besagt der Rekursionssatz von *Kleene* . . . . . 8
1.16 Eine Anwendung des Rekursionssatzes von Kleene? . . . . . 8
1.17 Gibt es andere, bessere Nummerierungen? . . . . . 9
1.18 Berechnung auf rationalen Zahlen . . . . . 9
1.19 Berechnung auf reellen Zahlen . . . . . 9
1.20 Typ–2 Maschine . . . . . 9

**2 Teil A, rekursive und r.a. Mengen** **10**

2.1 Definitionen für rekursive (entscheidbare) Mengen . . . . . 10
2.2 Definitionen für rekursiv–aufzählbare Mengen . . . . . 10
2.3 Abgeschlossenheit für rekursive und r.a. Mengen . . . . . 11
2.4 Beispiele für eine Menge, die rekursiv–aufzählbar aber nicht rekursiv ist . . . . . 11
2.5 Satz von Rice . . . . . 12
2.6 Berechenbarkeit anderer Mengen als Zahlen und Alphabet . . . . . 12
2.7 $(\nu, \nu)$–Berechenbarkeit . . . . . 12

**3 Teil B, formale Sprachen** **13**

3.1 Definition reguläre (Typ–3) Mengen (Sprachen) . . . . . 13
3.2 Beispiel für eine nicht reguläre Menge . . . . . 14
3.3 Zusammenhang (Überführung) nichtdeterm. Automaten ↔ determ. Automaten 15

3.4 Regelmenge der rechtslinearen Normalformgrammatik . . . 15
3.5 Wie erhält man die Normalform? . . . 15
3.6 Von einer vom endlichen Automaten akzeptierten Sprache zum regulären Ausdruck 15
3.7 Definition kontextfreie (Typ–2) Sprachen . . . 16
3.8 Womit erkennt man kontextfreie Sprachen? . . . 16
3.9 Beispiel für eine Menge die nicht kontextfrei ist . . . 16
3.10 Chomsky–Normalform der kontextfreien Grammatik . . . 17
3.11 Transformation von kontextfreien Grammatiken in Chomsky-Normalform . . . 17
3.12 PASCAL bzw. „normale“ Programmiersprachen kontextfrei? . . . 17
3.13 Diverse *Mustersprachen* und Aussagen dazu $\{a^n b^m\}, \{a^n b^n\}, \{ww^R\}, \{w\$w^R\}, \{ww\}, \{w\$w\}, \{w \in \{0,1\}^* \mid 010 \text{ ist Teilwort}\}, \{a^n b^n c^n\}$ . . . 17

**4 Teil B, Komplexitätstheorie 18**
4.1 Zusammenhang Platzbedarf ($s_M$) und Zeitbedarf ($t_M$) . . . 18
4.2 Definition P, NP . . . 18
4.3 Deterministische Abschätzung von NP . . . 18
4.4 NP-vollständig . . . 19
4.5 P–NP Problem . . . 19
4.6 Bedeutung von $A \in \text{ZEIT}(n^2)$ rekursiv? . . . 19
4.7 Bedeutung von $A \in \text{NZEIT}(n^2)$ rekursiv? . . . 19
4.8 Bedeutung von $A \in \text{BAND}(n^2)$ rekursiv? . . . 19
4.9 Satz von Savitch . . . 20
4.10 Kontrollturingmaschine . . . 20
4.11 Erkannte Sprache einer Kontrollturingmaschine . . . 20

# 1 Teil A, Berechenbarkeit

## 1.1 Definition für berechenbare Zahlenfunktionen

**Register–berechenbar** Eine Funktion $f :\subseteq \mathbb{N}^k \longrightarrow \mathbb{N}$ heisst berechenbar genau dann, wenn es eine $k$–stellige Registermaschine $M$ mit $f = f_M$ gibt.

**WHILE–berechenbar** Eine Funktion $f :\subseteq \mathbb{N}^k \longrightarrow \mathbb{N}$ heisst WHILE–berechenbar genau dann, wenn $f = \mathrm{AC} \circ \tau(P) \circ \mathrm{EC}$ für ein WHILE–Programm $P$ gilt. Eine Funktion $f$ ist WHILE–berechenbar genau dann, wenn sie Registerberechenbar ist.

**$\mu$–Rekursion** Es seien Prk, $\tilde{\mu}$ und Sub die Operationen der primitiven Rekursion, $\mu$–Rekursion und Substitution. Eine Funktion $f :\subseteq \mathbb{N}^k \longrightarrow \mathbb{N}$ heisst $\mu$–rekursiv genau dann, wenn sie sich in endlich vielen Schritten aus der Menge der rekursiven Grundfunktionen durch wiederholtes Anwenden der Operatoren Sub, Prk und $\tilde{\mu}$ erzeugen lässt.

## 1.2 Definition für berechenbare Wortfunktionen

**Turing–berechenbar** Eine Wortfunktion $f :\subseteq (\Sigma^*)^k \longrightarrow \Sigma^*$ heisst berechenbar genau dann, wenn $f = f_M$ für eine Turingmaschine $M$ gilt. (Wobei $\Sigma$ ein Alphabet und $k \in \mathbb{N}$ ist).

**Band–berechenbar** Eine Wortfunktion $f :\subseteq (\Sigma^*)^k \longrightarrow \Sigma^*$ ist Band–berechenbar genau dann, wenn sie Turing–berechenbar ist.

## 1.3 Turing–berechenbar = Band–berechenbar

Zu jeder Turingmaschine $M$ gibt es eine Bandmaschine $M'$ mit $f_M = f_{M'}$ bzw. $t_M = t_{M'}$. Zu jeder Bandmaschine $M'$ gibt es eine Turingmaschine $M$, die dasselbe Bandalphabet wie $M$ hat und für die $f_M = f_{M'}$ bzw. $t_M = t_{M'}$ gilt.
*Nachweis*: Band $\longrightarrow$ Turing trivial, Aufspaltung des einen Bandes.
Turing$\longrightarrow$ Band: „Zusammenkleben“ der Bänder $\Rightarrow$ Neues Arbeitsalphabet $\Gamma$, das aus $k+1$–Tupeln besteht, die jeweils als ein Symbol aufgefasst werden. Markieren der Kopfposition auf dem jeweiligen Band durch „überstreichen“ des jeweiligen Symbols.

## 1.4 Zeitbedarf Bandmaschine im Vergleich zu Turingmaschine

Quadratisch, da eventuell für jeden Befehl das ganze Band abgesucht werden muss.

## 1.5 Hilfssymbol–Lemma

Zu jeder Bandmaschine $M$ mit Ein–/Ausgabealphabet $\Sigma$ gibt es eine Bandmaschine $M'$ mit dem Arbeitsalphabet $\Gamma' = \Sigma \cup B$, so dass $f_M = f_{M'}$ und $t_M = t_{M'}$ gelten.
*Nachweisidee*: Simulation der Datenmenge, jedes Wort aus der ursprünglichen Datenmenge wird durch ein genügend großes Wort ($m \geq 2$ und $2^m \geq |\Gamma|$) der neuen Datenmenge verschlüsselt. Zusätzlich findet eine Anpassung der Eingabecodierung sowie der Ausgabecodierung statt, so dass alle Datenworte gleichviel Platz $m$ auf dem Band einnehmen. Für das Flussdiagramm werden zusätzliche Befehle der Form $L_m$, $R_m$ usw. eingeführt, die jeweils auf $m$–Felder des Bandes wirken. Das Flussdiagramm der Bandmaschine wird dann dementsprechend verfeinert.

## 1.6 Bandmaschine: Datenmenge, Befehlssatz, Konfigurationsmenge

### Datenmenge

$D := \Gamma^* \times \Gamma \times \Gamma^*$ wobei $\Sigma \subset \Gamma$ und $B \in \Gamma \setminus \Sigma$. Darstellung als 3–Tupel $(u, a, v)$ mit $u, v \in \Gamma^*$ und $a \in \Gamma$.

### Befehlssatz einer Bandmaschine

Kopfbewegung um ein Zeichen nach [L]inks oder [R]echts.
Schreibbefehl: $f^a$ mit $a \in \Gamma$
Testbefehl: $t^a$ mit $a \in \Gamma$.

### Konfigurationsmenge

2–Tupel aus (Marke im Flussdiagramm, (Datenelement)), wobei im Datenelement zusätzlich die Kopfposition(en) markiert ist(sind).

## 1.7 Unterschied „heisst“ berechenbar ↔ „ist“ berechenbar

Churchsche–These
„Heisst“ berechenbar = mathematische Definition (theoretische Berechenbarkeit, da Lösungsweg vorhanden).
„ist“ berechenbar = tatsächliche maschinelle Berechenbarkeit. Z.B. Speicher–/Zeitbedarf tatsächlich darstellbar.

## 1.8 Churchsche–These

Die Register–berechenbaren Funktionen sind genau die maschinell oder intuitiv berechenbaren Zahlenfunktionen.

## 1.9 Beispiel für nicht berechenbare Funktion oder Menge

- *Funktion:* Mit Diagonalbeweis z.B. $g(i) := \begin{cases} \text{div} & \text{falls } i \in \text{ Def}(\varphi_i) \\ 0 & \text{sonst} \end{cases}$
- *Menge:* $K_\varphi^{0-1} \mathrel{\widehat{=}} \mathbb{N}^2 \setminus K_\varphi^0$

## 1.10 Zusammenhang zwischen berechenbaren Wort– und Zahlfunktionen

Eine *Nummerierung* einer Menge $M$ ist eine (möglicherweise partielle) surjektive Funktion $\nu :\subseteq \mathbb{N} \to M$.
Es sei $\Sigma$ ein Alphabet, $\nu : \mathbb{N} \longrightarrow \Sigma^*$ eine Standardnummerierung von $\Sigma^*$ und $k \in \mathbb{N}$. Die Funktion $\vec{\nu} : \mathbb{N}^k \longrightarrow (\Sigma^*)^k$ sei definiert durch $\vec{\nu}(i_1, \ldots, i_k) := (\nu(i_1), \ldots, \nu(i_k))$ für $i_1, \ldots, i_k \in \mathbb{N}$. Es sei $g :\subseteq (\Sigma^*)^k \longrightarrow \Sigma^*$. Dann gilt:

$g$ berechenbar $\iff \nu^{-1} g \vec{\nu} :\subseteq \mathbb{N}^k \longrightarrow \mathbb{N}$ berechenbar.

Analog gilt für jede Funktion $f :\subseteq \mathbb{N}^k \longrightarrow \mathbb{N}$:

$f$ berechenbar $\iff \nu f \vec{\nu}^{-1} :\subseteq (\Sigma^*)^k \longrightarrow \Sigma^*$ berechenbar.

## 1.11 Register–berechenbar $\Longrightarrow$ Turing–berechenbar

Es sei $\Sigma := \{0\}$ und $f :\subseteq \mathbb{N}^k \longrightarrow \mathbb{N}$ Register–berechenbar. Dann gibt es eine Turing–berechenbare Funktion $g :\subseteq (\Sigma^*)^k \longrightarrow \Sigma^*$ mit $g(0^{x_1}, \ldots, 0^{x_n}) = 0^{f(x_1, \cdots, x_n)} \quad \forall x_1, \ldots, x_n \in \mathbb{N}$.
*Nachweis* über geeignete Simulation. Je Register der Registermaschine ein Band der Turingmaschine. Codierung der Eingabe $x \in \mathbb{N}$ durch entsprechende Anzahl von Zeichen $a \in \Sigma$. Simulation der Registerbefehle durch entsprechende Bandoperationen.

## 1.12 Zusammenhang Funktion und Programme?

Definiton der Standardnummerierung $\varphi$

(1) Es sei eine Ordnungsfunktion $a : \{1, \ldots, 8\} \longrightarrow \Omega$ definiert durch

$$(a_1, a_2, \ldots, a_8) := (1, B, (, ), :, , , R, L).$$

Es sei $\nu_\Omega$ die aus $a$ abgeleitete Standardnummerierung von $\Omega^*$.

(2) Es sei $\nu_P : \mathbb{N} \longrightarrow \mathrm{BP}$ definiert durch

$$\nu_P(i) = \begin{cases} \nu_\Omega(i) & \text{falls } \nu_\Omega(i) \in \mathrm{BP} \\ \text{„(: } B, , )\text{“} & \text{sonst.} \end{cases}$$

(3) Es sei $\xi : \mathrm{BM} \longrightarrow P^{(1)}$ definiert durch $\xi(M) := \iota^{-1} f_M \iota$
(wobei $\iota : \mathbb{N} \longrightarrow \{1\}^*, \iota(i) := 1^i$).

(4) Dann sei $\varphi : \mathbb{N} \longrightarrow P^{(1)}$ definiert duch

$$\varphi_i := \varphi(i) := \xi \nu_M \nu_P(i)$$

für alle $i \in \mathbb{N}$. Die Nummerierung $\varphi$ heißt die *Standardnummerierung* von $P^{(1)}$.

Das heisst graphisch dargestellt:
$\mathbb{N} \xrightarrow{\nu_P} BP \subseteq \Omega^* \xrightarrow{\nu_M} BM \xrightarrow{\xi} P^{(1)} \subseteq R^{(1)}$ bzw. $\varphi(i) := \xi \nu_M \nu_P(i)$

## 1.13 Was sind die Programme bei $\varphi$ ?

Die natürlichen Zahlen, denen eine partielle Funktion aus $P^{(1)}$ zugeordnet sind.

## 1.14 Zwei wesentliche Anforderungen an Programmiersprachen

### utm–Theorem (universelle Turingmaschine)

Es sei $\varphi : \mathbb{N} \longrightarrow P^{(1)}$ die Standardnummerierung von $P^{(1)}$, und es sei $u_\varphi :\subseteq \mathbb{N}^2 \longrightarrow \mathbb{N}$ definiert durch $u_\varphi(i, x) := \varphi_i(x)$ (für alle $i, x \in \mathbb{N}$). Dann ist $u_\varphi$ berechenbar.
($u_\varphi$ heißt die *universelle Funktion* von $\varphi$.)
*Beweisidee*: Berechnet nach Eingabe von $(i, x)$ das Programm (die Funktion) $i$ für die Eingabe (an der Stelle) $x$. Beweis über $h(i, n, t)$–Funktion mit folgendem Flussdiagramm:

```
1: R0 := h(R1, R2, R3), 2;
2: R0  = 0, 3, 4;
3: R3 := R3 + 1, 1;
4: R0 := R0 - 1, 5;
5: HALT
```

Gesucht wird mit diesem „Programm“ die kleinste Zahl $t$ für die $h(i, n, t) \neq 0$ gilt.

### smn-Theorem (Übersetzungsfunktion)

Es sei $f \in P^{(2)}$ eine beliebige zweistellige berechenbare Funktion. Dann gibt es eine totalrekursive Funktion $r \in R^{(1)}$ mit

$$(\forall i, j \in \mathbb{N}) f(i,j) = \varphi_{r(i)}(j).$$

Bestimmt aus gegebenen Band–Programmen neue Band–Programme mit vorgegebenen Eigenschaften.
utm– und smn–Theorem sind fundamentale Anforderungen an Programmiersprachen.

#### $h(i,n,t)$–Funktion ist berechenbar

Die wie folgt definierte Funktion $h : \mathbb{N}^3 \longrightarrow \mathbb{N}$ ist berechenbar.

$$h(i,n,t) = \begin{cases} 1 + \varphi_i(n) & \text{falls } \Phi_i(n) \text{ existiert und } \Phi_i(n) \leq t \\ 0 & \text{sonst.} \end{cases}$$

mit $\Phi$ als *Standardkomplexität* zu $\varphi$.

#### $h(i,x)$: Unlösbarkeit des Halteproblems

Es gibt keine Funktion $h \in R^{(2)}$ mit

$$h(i,x) = \begin{cases} 1 & \text{falls } \varphi_i(x) \text{ existiert} \\ 0 & \text{sonst.} \end{cases}$$

## 1.15 Was besagt der Rekursionssatz von Kleene

Zu jeder Funktion $f \in R^{(1)}$ gibt es eine Zahl $n$ mit $\varphi_n = \varphi_{f(n)}$.
Jede totale berechenbare Programmtransformation bildet mindestens ein Programm in ein äquivalentes ab.

## 1.16 Eine Anwendung des Rekursionssatzes von Kleene?

**Selbstreproduktion** Es gibt eine Zahl $n$, so dass $\varphi_n(x) = n$ für alle $x \in \mathbb{N}$.
*Beweis* durch Anwendung des Rekursionssatzes von Kleene.

## 1.17 Gibt es andere, bessere Nummerierungen?

**Äquivalenzsatz von Rogers** Es sei $\psi : \mathbb{N} \longrightarrow P^{(1)}$ eine Nummerierung von $P^{(1)}$. Dann sind die folgenden Eigenschaften äquivalent:

(1) Es gibt Funktionen $g, h \in R^{(1)}$ („Übersetzer“) mit $\varphi = \psi \circ g$ und $\psi = \varphi \circ h$ („$\varphi$ und $\psi$ sind äquivalent“).

(2) $\psi$ erfüllt das utm-Theorem und das snm-Theorem:

$$\begin{aligned} utm(\psi) :\Longleftrightarrow\ & \text{die universelle Funktion } u_\psi :\in \mathbb{N}^2 \longrightarrow \mathbb{N} \\ & \text{von } \psi \text{ ist berechenbar (wobei } u_\psi(i,x) := \psi_i(x)) \\ smn(\psi) :\Longleftrightarrow\ & (\forall f \in P^{(2)})(\exists s \in R^{(1)})(\forall i,x)\psi_{s(i)}(x) = f(i,x). \end{aligned}$$

## 1.18 Berechnung auf rationalen Zahlen

$\nu_{\mathbb{Q}} : \mathbb{N} \longrightarrow \mathbb{Q}, \nu_{\mathbb{Q}}\langle i,j,k\rangle := \frac{i-j}{1+k}$ für alle $i,j,k \in \mathbb{N}$;
Berechnung dann über eine Funktion $g :\subseteq \mathbb{N} \longrightarrow \mathbb{N}$ mit $f :\subseteq \mathbb{Q} \longrightarrow \mathbb{Q}$ über $f := \nu_{\mathbb{Q}} g \nu_{\mathbb{Q}}^{-1}$

## 1.19 Berechnung auf reellen Zahlen

Für $w \in \mathrm{Def}(\nu_{rat})$ sei $\bar{w} := \nu_{rat}(w)$. Die Cauchy–Darstellung $\rho :\subseteq \Sigma^w \longrightarrow \mathbb{R}$ der reellen Zahlen sei wie folgt definiert:

$$\begin{aligned} \rho(p) = x :\Longleftrightarrow\ & \text{es gibt } u_0, u_1, \ldots \in \mathrm{Def}(\nu_{rat}), \text{ so dass } p = u_0 \# u_1 \# \ldots, \\ & (\forall k \in \mathbb{N})(\forall i > k)|\bar{u}_i - \bar{u}_k| \leq 2^{-k} \text{ und } x = \lim_{k\to\infty} \bar{u}_k. \end{aligned}$$

## 1.20 Typ–2 Maschine

Modifikation der Turingmaschine. Eine $k$–stellige Typ–2 Maschine hat $k$ unendliche Eingabebänder für Eingaben $p_1, p_2, \ldots, p_k \in \Sigma^w$, ein überall unendliches *Einweg*ausgabeband, und endlich viele normale Turingbänder als Arbeitsbänder.

### $(\rho, \rho)$–Berechenbarkeit

Eine Funktione $f :\subseteq \mathbb{R}^n \longrightarrow \mathbb{R}$ heißt berechenbar genau dann, wenn sie $(\rho, \ldots, \rho, \rho)$–berechenbar ist.

### Berechenbare reelle Funktionen

$x^2$, $x$, $x + y$, $x \cdot y$, $\frac{1}{x}$, $\sqrt{x}$, sin, log, Exponentialfunktion $e^x$.

# 2 Teil A, rekursive und r.a. Mengen

## 2.1 Definitionen für rekursive (entscheidbare) Mengen

### Charakteristische Funktion

Eine Menge $A \subseteq \mathbb{N}^k (k \geq 1)$ heißt *rekursiv*, gdw. die charakteristische Funktion $\mathrm{cf}_A : \mathbb{N}^k \longrightarrow \mathbb{N}$, definiert durch

$$\mathrm{cf}_A(x) := \begin{cases} 1 & \text{falls } x \in A \\ 0 & \text{sonst,} \end{cases}$$

berechenbar ist. Rekursive Mengen nennt man auch *entscheidbar*.

### über eine totale Funktion $f$

$A \subseteq \mathbb{N}^k$ ist *rekursiv*, gdw. es eine totale berechenbare Funktion $f \in R^{(k)}$ gibt mit $A = f^{-1}\{0\}$.

### Bild einer wachsenden totalen Funktion

Es sei $A \subseteq \mathbb{N}$ unendlich. Dann ist $A$ *rekursiv*, gdw. es eine wachsende Funktion $f \in R^{(1)}$ gibt mit $A = \mathrm{Bild}(f)$.

### $\mathbb{N} \setminus A$ rekursiv–aufzählbar

Eine Menge $A \subseteq \mathbb{N}^k$ ist rekursiv, gdw. $A$ und $\mathbb{N}^k \backslash A$ rekursiv–aufzählbar sind.

## 2.2 Definitionen für rekursiv–aufzählbare Mengen

### Definitonsbereich einer partiellen Funktion

Eine Menge $A \subseteq \mathbb{N}^k$ heißt *rekursiv–aufzählbar*, gdw. es eine partielle berechenbare Funktion $f :\subseteq \mathbb{N}^k \longrightarrow \mathbb{N}$ gibt mit $A = \mathrm{Def}(f)$. Analog für Wortmengen. $A \subseteq \Sigma^*$.

### Bildbereich einer totalen Funktion

Eine Menge $A \subseteq \mathbb{N}$ ist *rekursiv–aufzählbar* gdw. $A = \emptyset$ oder $A = \mathrm{Bild}(g)$ für ein $g \in R^{(1)}$.

### Projektionssatz

Eine Menge $A \subseteq \mathbb{N}^k$ ist *rekursiv–aufzählbar* gdw. $A = \{x \in \mathbb{N}^k | (\exists t)(x,t) \in B\}$ für eine rekursive Menge $B \subseteq \mathbb{N}^{k+1}$
*Beispielsfunktionen*: $h(i,n,t) \in R$ und $h(i,n) \in P$

## 2.3 Abgeschlossenheit für rekursive und r.a. Mengen

### für rekursive Mengen

Schnitt, Vereinigung und Komplement
Es gilt:
$A \cap B$: Denn $\mathrm{cf}_{(A\cap B)}(x) = \min\{cf_A(x), cf_B(x)\}$
$A \cup B$: Denn $\mathrm{cf}_{(A\cup B)}(x) = \max\{cf_A(x), cf_B(x)\}$
$\mathbb{N} \setminus A$: Denn $\mathrm{cf}_{(\mathbb{N}\setminus A)}(x) = 1 - cf_A(x)$

### für rekursiv aufzählbare Mengen

Schnitt und Vereinigung.
Es gilt:

- $A \cap B$: Sei $f$ partielle Funktion von $\mathbb{N}^k$ nach $\mathbb{N}$ mit $f = f_A + f_B$. Dann ist $\mathrm{Def}(f) = \mathrm{Def}(f_A) \cap \mathrm{Def}(f_B)$
- $A \cup B$: „Parallele“ Ausführung der Berechung von $f_A$ und $f_B$. Hält eine der Berechnungen an, so ist $x \in A \cup B$ sonst nicht.

## 2.4 Beispiele für eine Menge, die rekursiv–aufzählbar aber nicht rekursiv ist

### Halteproblem ($K_\varphi$ bzw. $K^0_\varphi$)

$K_\varphi := \{i \in \mathbb{N} \mid \varphi_i(i) \text{ existiert}\}$ (Selbstanwendbarkeitsproblem)
$K^0_\varphi := \{(i,x) \in \mathbb{N}^2 | \varphi_i(x) \text{existiert}\}$ (Halteproblem)
Beide Mengen sind rekursiv aufzählbar aber nicht rekursiv, denn die Komplemente sind nicht rekursiv–aufzählbar.
*Beweisidee*: $K^0_\varphi$ ist rekursiv–aufzählbar, denn $K^0_\varphi = \mathrm{Def}(u_\varphi)$ und $u_\varphi \in P^{(2)}$.
$K_\varphi$ ist rekursiv–aufzählbar, da es ein Spezialfall für $x = i$ von $K^0_\varphi$ ist.
Wäre $K_\varphi$ rekursiv, so wäre $\mathbb{N} \setminus K_\varphi$ rekursiv–aufzählbar. Sei $i \in \mathbb{N}$ und (Def($\varphi_0$),Def($\varphi_1$),...) die Folge aller rekursiv–aufzählbaren Teilmengen von $\mathbb{N}$. Dann gilt mit Definition von $K_\varphi$:
$i \in \mathbb{N} \setminus K_\varphi \iff i \notin K_\varphi \iff i \notin \mathrm{Def}(\varphi_i) \implies \mathbb{N} \setminus K\varphi \neq \mathrm{Def}(\varphi_i)$ für alle $i \in \mathbb{N}$.
Da die Liste (Def($\varphi_0$),Def($\varphi_1$),...) alle rekursiv–aufzählbaren Teilmengen enthält, kann $\mathbb{N} \setminus K_\varphi$ nicht rekursiv–aufzählbar sein.
Wäre $K^0_\varphi$ rekursiv, dann müsste auch $K_\varphi$ rekursiv sein. Das ist aber nicht der Fall.

## 2.5 Satz von Rice

Falls $F \subseteq P^{(1)}$ und $F \neq \emptyset$ und $F \neq P^{(1)}$ ist, dann ist die Menge $\varphi^{-1}(F) = \{i \in \mathbb{N} | \varphi_i \in F\}$ nicht rekursiv.
Grob gesagt kann man keine nicht triviale Eigenschaft von Funktionen anhand ihrer Programme entscheiden.

### Korrektheitsprobem (nicht einmal rekursiv–aufzählbar)

Es gibt kein Programm, dass feststellt, ob ein anderes Programm fehlerfrei arbeitet.

### Diagonalisierung

Beweis Methode z.B. dafür, dass eine nicht berechenbare Funktion $g :\subseteq \mathbb{N} \longrightarrow \mathbb{N}$ existiert. Seien alle partiell berechenbaren Funktionen $f \in P^{(1)}$ durchnummeriert (das geht, da die Menge $P^{(1)}$ abzählbar unendlich ist), und sei $\varphi_i(i)$ der $i$–te Funktionswert an der Stelle $i$. Sei $g(i) := \varphi_i(i)+1$ falls $\varphi_i(i)$ existiert und 0 sonst. Dann ist $g$ offensichtlich berechenbar $\Longrightarrow \exists k \in \mathbb{N}$ mit $\varphi_k = g$. Da sich aber $g$ von jeder Funktion aus $\varphi_i$ an der Stelle $i$ unterscheidet, ist $g \notin P^{(1)}$ = Widerspruch.

## 2.6 Berechenbarkeit anderer Mengen als Zahlen und Alphabet

Mittels $(\nu, \nu)$–Berechenbarkeit.

## 2.7 $(\nu, \nu)$–Berechenbarkeit

Es seien $\nu_0 :\subseteq \mathbb{N} \longrightarrow M_0, \ldots \nu_k :\subseteq \mathbb{N} \longrightarrow M_k$ Nummerierungen. Eine Funktion $f :\subseteq M_1 \times \ldots \times M_k \longrightarrow M_0$ heißt $(\nu_1, \ldots, \nu_k, \nu_0)$–berechenbar, gdw. es eine berechenbare Funktion $g :\subseteq \mathbb{N}^k \longrightarrow \mathbb{N}$ gibt mit

$$f(\nu_1(i_1), \ldots, \nu_k(i_k)) = \nu_0 \circ g(i_1, \ldots, i_k)$$

für alle $(i_1, \ldots, i_k)$ mit $(\nu_1(i_1), \ldots, \nu_k(i_k)) \in \mathrm{Def}(f)$.
Wir sagen in diesem Falle auch: $f$ ist eine $(\nu_1, \ldots, \nu_k, \nu_0)$–Spur von $g$.

$$\begin{array}{ccc} \mathbb{N} & \xrightarrow{g} & \mathbb{N} \\ {\scriptstyle \nu_1}\downarrow & & \downarrow{\scriptstyle \nu_2} \\ M_1 & \xrightarrow{f} & M_2 \end{array} \qquad \begin{array}{ccc} i & \xrightarrow{g} & g(i) \\ {\scriptstyle \nu_1}\downarrow & & \downarrow{\scriptstyle \nu_2} \\ x & \xrightarrow{f} & f(x) \end{array}$$

# 3 Teil B, formale Sprachen

## 3.1 Definition reguläre (Typ–3) Mengen (Sprachen)

**Reguläre Menge wird von rechtslinearen Grammatiken erzeugt** Reguläre Sprachen werden von Typ–3–(rechtslinearen)–Grammatiken $G = (\Pi, \Sigma, R, S)$ erzeugt, wobei

- $\Pi$ die Menge der Nichtterminale ist,
- $\Sigma$ ein Alphabet,
- S das Startnichtterminal und
- R eine Menge von Regeln (Zuordnungsvorschriften) (U,V) für die gilt:
  $\forall (U, V) \in \mathbb{R}.U \in \Pi \wedge V \in \Sigma^* \cup \Sigma^* \cdot \Pi$
  D.h. alle Regeln in $G$ haben die Form: $A \longrightarrow wB$, $A \longrightarrow w$ mit $A, B \in \Pi, w \in \Sigma^*$

**Reguläre Menge wird von endlichen Automaten erkannt** Ein endlicher Automat ist ein Quintupel $A = (\Sigma, Q, q_0, F, \delta)$ mit

- $\Sigma$ ist Alphabet,
- $Q$ ist die endliche Menge der Zustände,
- $q_0 \in Q$ ist der Anfangszustand,
- $F \subseteq Q$ ist die Menge der Endzustände und
- $\delta \subseteq Q \times \Sigma \times Q$ ist die Überführungsrelation.

Die von $A$ erkannte (akzeptierte) Sprache $L(A) := \{x \in \Sigma^* \mid \exists q \in F.(q_0, x, q) \in \delta^*\}$.
A heißt determiniert, wenn es für jeden Zustand $q$ *maximal* einen Folgezustand für alle $a \in \Sigma$ gibt.
A heisst vollständig, wenn es für jeden Zustand $q$ zu jedem $a \in \Sigma$ *mindestens* einen Folgezustand gibt.

**Reguläre Menge wird mit regulärem Ausdruck erzeugt: $\cup, \cdot, ^*$** Sei $\Sigma$ ein Alphabet.

(1) Die $\emptyset$ ist reguläre Menge über $\Sigma$.

Ist $a \in \Sigma$, so ist die Wortmenge $\{a\}$ eine reguläre Menge über $\Sigma$.

(2) Sind $X$ und $Y$ reguläre Mengen über $\Sigma$, so auch $X \cup Y$, $X \cdot Y$, $X^*$

(3) Jede reguläre Menge lässt sich durch (1) und (2) definieren.

### Pumping–Lemma für reguläre Sprachen

Sei $L \subseteq \Sigma^*$ regulär. Dann gilt: Es gibt ein $n \in \mathbb{N}$, so dass für alle $t, t', z \in \Sigma^*$ mit $tzt' \in L$ und $lg(z) = n$ gilt:

$$\exists u, v, w \in \Sigma^*.(z = uvw \wedge v \neq \varepsilon \wedge i \geq 0.tuv^iwt' \in L).$$

Für $n$ kann man die Zustandszahl eines endlichen Automaten $A$ mit $L(A) = L$ wählen.
Wenn $L$ regulär ist, gibt es also eine Konstante $n$ (die nur von $L$ abhängt), so dass man in jedem Teilwort $z$ der Länge $n$ eines jeden Wortes $x$ aus $L$ (der Länge $\geq n$) ein Teilwort $v$ mit der Eigenschaft findet, dass beim Ersetzen von $v$ in $x$ durch eine beliebig häufige Wiederholung von $v$ („Aufpumpen“) wiederum ein Wort aus $L$ entsteht.

## 3.2 Beispiel für eine nicht reguläre Menge

**$\{a^nb^n\}$**

a) Pumping–Lemma für reguläre Sprachen

Angenommen $L$ ist regulär. Dann gibt es ein $n$ mit den im Pumpinglemma genannten Eigenschaften. Sei $a^nb^n \in L$ und $t := \varepsilon, z = a^n, t' = b^n$. Dann kann $z$ in $u, v, w$ zerlegt werden mit $z = uvw$ und $v \neq \varepsilon$. Es gibt dann $k \in \mathbb{N}$ mit $1 \leq k \leq n$ und $v = a^k$. Mit $i = 0$ gilt dann $tuv^0wt' = tuwt' = a^{(n-k)}b^n \notin L$. Widerspruch zum Pumpinglemma $\Longrightarrow L$ ist nicht regulär.

b) Anschaulich, weil sich ein endlicher Automat ohne Keller nicht „merken“ kann wieviele beliebig viele $a$'s er gelesen hat. Im Gegensatz dazu Kellerautomat, der die $a$'s auf dem Kellerspeicher „zwischenlagert“.

## 3.3 Zusammenhang (Überführung) nichtdeterm. Automaten ↔ determ. Automaten

**Potenzautomaten** Zu jedem endlichen Automaten $A$ gibt es einen vollständigen deterministischen Automaten $B$ mit $L(A) = L(B)$.
Konstruktion über WHILE–Programm mit daraus resultierender Zustandstabelle $P, a, b, W \setminus V$, wobei $\Sigma = \{a, b\}$, $V$ die anfängliche leere Menge der erreichbaren Zustände des Potenzautomaten ist und $W$ sukzessive alle erreichbaren Zustände des Potenzautomaten enthält, wobei $|W| \geq |V|$ ist. Wenn $W$ und $V$ identisch sind, ist das WHILE–Programm beendet und $P$ enthält alle erreichbaren Marken.

## 3.4 Regelmenge der rechtslinearen Normalformgrammatik

Rechtslineare Normalformgrammatiken von Chomsky haben die Form:
$A \longrightarrow aB, A \longrightarrow \varepsilon$ mit $A, B \in \Pi$ und $a \in \Sigma$

## 3.5 Wie erhält man die Normalform?

**Umwandlung von rechtslinearen Grammatiken in rechtslineare Normalformgrammatiken mittels Transformation**

NT Normieren der Terminalregeln $A \longrightarrow w$ wird zu $A \longrightarrow wE, E \longrightarrow \varepsilon$

VR Verkürzen der Regellängen: $A \longrightarrow awB$ wird zu $A \longrightarrow aC, C \longrightarrow wB$

ELR Eliminieren längentreuer Regeln: $A \longrightarrow B, B \longrightarrow aC$ wird zu $A \longrightarrow aC$

## 3.6 Von einer vom endlichen Automaten akzeptierten Sprache zum regulären Ausdruck

Über die Rekursionsformel $(L_{ij}^k)$
Allg. $L(A) = \bigcup_{j \in F} L_{0j}^{|Q|}$.
Wobei $L_{ij}^k = L_{ij}^{k-1} \cup L_{ik}^{k-1} \cdot (L_{kk}^{k-1})^* \cdot L_{kj}^{k-1}$ gilt.
Anschaulich: Ein Wort $w$ überführt den Zustand $i$ in den Zustand $j$, wobei als Zwischenzustände höchstens $1, \ldots, k$ vorkommen.

## 3.7 Definition kontextfreie (Typ–2) Sprachen

**Kontextfreie Sprache wird von kontextfreier Grammatik erzeugt** Kontextfreie Grammatiken bestehen aus rechtslinearen bzw. regulären Grammatiken, bei denen auf der rechten Seite der Regeln beliebige Worte zugelassen sind.
Eine Grammatik $G = (\Pi, \Sigma, R, S)$ heißt kontextfrei, gdw. $R \subseteq \Pi \times (\Sigma \cup \Pi)^*$.
Sprache $L \subseteq \Sigma^*$ heißt kontextfrei, gdw. es eine kontextfreie Grammatik $G$ gibt mit $L = L(G)$.

**Kontextfreie Sprache wird von Kellerautomaten erkannt** $A = (\Sigma, \Gamma, \#, Q, q_0, Q_f, \delta)$ heißt Kellerautomat, wenn

| | |
|---|---|
| $\Sigma$ | eine endliche Menge (*Eingabealphabet*) |
| \$ | ein Symbol $\notin \Sigma$ (*Markierung für Eingabeende*) |
| $\Gamma$ | eine endliche Menge (*Kelleralphabet*) |
| $\# \in \Gamma$ | (*Kellerstartsymbol*) |
| Q | eine endliche Menge (*Zustandsmenge*) |
| $q_0 \in Q$ | (*Anfangszustand*) |
| $Q_f \subseteq Q$ | (*Menge der Endzustände*) und |

$\delta \subseteq (Q \times \Sigma \times \Gamma) \times (\{N, R\} \times \Gamma^* \times Q) \cup (Q \times \{\$\} \times \Gamma) \times (\{N\} \times \Gamma^* \times Q)$
wobei der Eingabekopf stehen bleibt ($N$) oder nach rechts bewegt wird ($R$), eine endliche Menge ist.

## 3.8 Womit erkennt man kontextfreie Sprachen?

Mit einem Kellerautomaten (Pushdownautomat PDA).

### Kontextfreie Grammatik für $\{a^n b^n\}$

$S \longrightarrow aSb \,|\, \varepsilon$

## 3.9 Beispiel für eine Menge die nicht kontextfrei ist

$\{a^n b^n c^n\}$
Nachweis über Pumpinglemma für kontextfreie Sprachen

### Pumpinglemma für kontextfreie Sprachen

Sei $L$ kontextfrei. Dann gibt es ein $p \in \mathbb{N}$, so dass man jedes Wort $z \in L$ mit $lg(z) \geq p$ so in Teilworte $uvwxy$ zerlegen kann, dass gilt:

$$z = uvwxy, vx \neq \varepsilon, \lg(vwx) \leq p, \forall i \geq 0. uv^i wx^i y \in L$$

Ist $L = L(G)$ mit einer Chomsky-Normalformgrammatik $G = (\Pi, \Sigma, R, S)$, so gilt die Aussage für jedes $p \geq 2^{\#\Pi}$.

## 3.10 Chomsky–Normalform der kontextfreien Grammatik

Allgemein $R$ von kontextfreien Grammatiken. $\forall(U,V) \in R.U \in \Pi^1$
Chomsky-Normalform:

(1) Enthält keine $\varepsilon$–Regeln mit Ausnahme von $S \longrightarrow \varepsilon$ zur Erzeugung des leeren Wortes.

(2) $R \subseteq \Pi \times (\Pi^2 \cup \Sigma)$, d.h. $V$ besteht entweder aus einem Terminal oder aus 2 Nichtterminalen.

## 3.11 Transformation von kontextfreien Grammatiken in Chomsky-Normalform

(1) ET Ersetzen der Terminalregeln $A \longrightarrow ab$ wird zu $A \longrightarrow A'B', A' \longrightarrow a, B' \longrightarrow b$

(2) VR Verkürzen der Regellängen: $A \longrightarrow ABC$ wird zu $A \longrightarrow AD, D \longrightarrow BC$

(3) ELR Eliminieren längentreuer Regeln: $A \longrightarrow B, B \longrightarrow b$ wird zu $A \longrightarrow b$

## 3.12 PASCAL bzw. „normale“ Programmiersprachen kontextfrei?

Backus Naur Form der PASCAL-Syntax ist kontextfrei.
Aber da jede Variable vor ihrer Verwendung deklariert werden muss, ist PASCAL als Programmiersprache *nicht* kontextfrei.

## 3.13 Diverse Mustersprachen und Aussagen dazu

Ist die Sprache $L_i$ kontextfrei / regulär?

- $a^n b^m$ regulär (Angabe eines Automaten). Regulärer Ausdruck: $a^*b^*$
- $a^n b^n$ kontextfrei, Regel $S \longrightarrow aSb \,|\, \varepsilon$, nicht regulär.
- $ww^R$ nichtdeterministisch kontextfrei, Regel $S \longrightarrow 0S0 \,|\, 1S1 \,|\, \varepsilon$, Angabe eines Kellerautomaten, der Mitte „rät“, nicht regulär
- $w\$w^R$ deterministisch kontextfrei, nicht regulär
- $ww$ nicht kontextfrei
- $w\$w$ nicht kontextfrei, Beweis mittels Pumping Lemma möglich
- $\{w \in \{0,1\}^* \,|\, 010 \text{ ist Teilwort}\}$ regulär, Angabe eines endlichen Automaten
- $a^n b^n c^n$ nicht kontextfrei

# 4 Teil B, Komplexitätstheorie

## 4.1 Zusammenhang Platzbedarf ($s_M$) und Zeitbedarf ($t_M$)

(1) $s_M(x) \leq t_M(x) + k$

*Beweisidee*: Pro Zeit kann nur ein Speicher angesprochen werden.

(2) $t_M(x) \leq c^{\tilde{s}_M(lg(x))}$, falls $s_M$ mindestens logarithmischen Platzbedarf hat.

*Beweisidee*: Anzahl der Konfigurationen der Bandmaschine. Da auf beschränktem Bandplatz nur endliche Anzahl von unterschiedlichen Konfigurationen entstehen können. Wird zweimal dieselbe Konfiguration erreicht, befindet sich das Programm in einer Endlosschleife.

## 4.2 Definition P, NP

Menge, der in polynomialer Zeitomplexität

- deterministisch *entscheidbaren* Sprachen (P)
- nichtdeterministisch *beweisbaren* Sprachen (NP)

$$\mathrm{P} := \bigcup_{k\in\mathbb{N}} \mathrm{ZEIT}(n^k) \quad (\mathrm{ZEIT}(n^k) := \{L \mid \exists \text{ TM } M \text{ mit } L(M) = L \text{ und } \tilde{t}_M \in \mathrm{O}(n^k)\})$$

$$\mathrm{NP} := \bigcup_{k\in\mathbb{N}} \mathrm{NZEIT}(n^k) \quad (\mathrm{NZEIT}(n^k) := \{L \mid \exists \text{ KTM } M \text{ die } L \text{ beweist und } \tilde{t}_M \in \mathrm{O}(n^k)\})$$

## 4.3 Deterministische Abschätzung von NP

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \bigcup_{c\in\mathbb{N}} \mathrm{ZEIT}(2^{n^c})$$

## 4.4 NP-vollständig

Eine Menge $X \subseteq \Sigma^*$ heißt NP–vollständig, gdw. sie NP-hart ist und $X \in$ NP gilt.

### NP–hart

Eine Menge $X \subseteq \Sigma^*$ heißt NP–hart, gdw. $\forall L \in \text{NP}.L \leq_{pol} X$ gilt. D.h. $\exists f \in \text{FP}$, so dass $x \in L \Longleftrightarrow f(x) \in X$ ist.

## 4.5 P–NP Problem

P $\subseteq$ NP, aber NP $\subseteq$ P und damit P $\stackrel{?}{=}$ NP ist ungelöst.

### Beispiele für NP–vollständige Menge

- 2D-Domino
- SAT
- 3SAT
- CLIQUE
- Hamiltonkreisproblem
- TravelingsSalesman
- Rucksackproblem

## 4.6 Bedeutung von $A \in$ ZEIT($n^2$) rekursiv?

Es gibt eine Turingmaschine, die $A$ in O($n^2$) Zeit akzeptiert. $A$ ist rekursiv. TM hält immer.

## 4.7 Bedeutung von $A \in$ NZEIT($n^2$) rekursiv?

Es gibt eine KTM $M$, die $x \in A$ in O($n^2$) Zeit bei geeignetem Beweis akzeptiert.
$A$ ist aber nicht in NZEIT($n^2$) entscheidbar, sondern nur in ZEIT($c^{n^2}$), wegen
NZEIT($n^2$) $\subseteq \bigcup_{c \in \mathbb{N}}$ ZEIT($c^{n^2}$).
$A$ ist dennoch rekursiv.

## 4.8 Bedeutung von $A \in$ BAND($n^2$) rekursiv?

$A \in$ BAND($n^2$) bedeutet, dass es eine Turingmaschine gibt, die $A$ mit einem Platzbedarf von maximal $n^2$ entscheidet, ob $x \in A$ gilt oder nicht. Wobei lg($x$) = $n$.
Die Menge $A$ ist rekursiv.

## 4.9 Satz von Savitch

Für alle Funktionen $f : \mathbb{N} \longrightarrow \mathbb{N}$ mit $\log \in \mathrm{O}(f)$ gilt: $\mathrm{NBAND}(f) \subseteq \mathrm{BAND}(f(n)^2)$.
Jedes Akzeptanzproblem in $\mathrm{NBAND}(f)$ liegt auch in $\mathrm{BAND}(f(n)^2)$.

## 4.10 Kontrollturingmaschine

### Definition Kontrollturingmaschine (Aufbau und Funktionsweise)

Eine k–Band Kontrollturingmaschine $M$ ist eine Turingmaschine mit

- Eingabealphabet $\Sigma \cup \{0, 1\}$, Arbeitsalphabet $\Gamma$,
- einem Eingabeband (Band 1) für $x$,
- einem *Einweg*eingabeband (Band 0) für den zu $x$ gehörigen „Beweis“,
- $k$ Arbeitsbändern,
- einem aktzeptierenden Endzustand $q+$
- einem verwerfenden Endzustand $q-$

## 4.11 Erkannte Sprache einer Kontrollturingmaschine

$L_M = \{x \in \Sigma^* \mid \exists y \in \{0, 1\}.\tau_M(x, y) = 1\}$ wobei $\tau_M$ der von $M$ berechnete Test ist, und $y$ als Hilfseingabe den nichtdeterministischen Anteil darstellt. $y$ ist der „Beweis“ für die Eingabe auf Band 1, wenn der Automat mit $q+$ anhält.