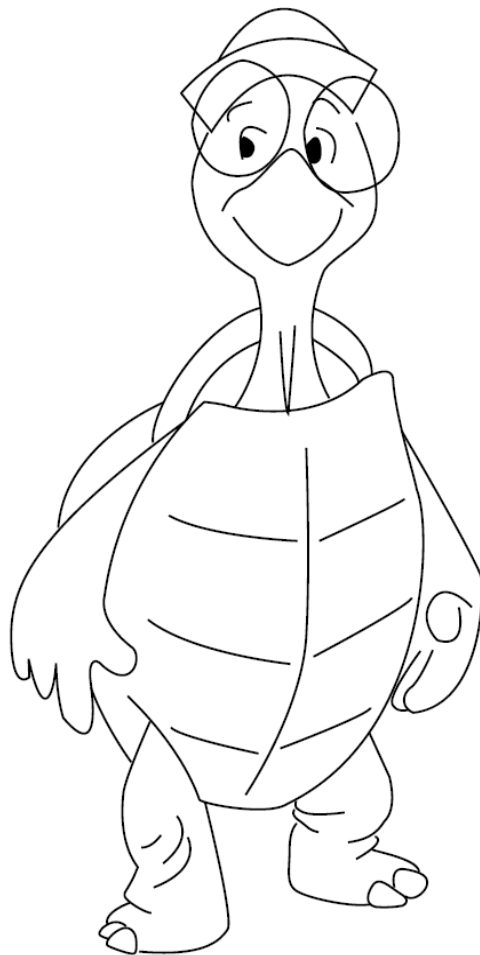


Informatik III

Thilo Beckmann, Marie Christ

21. Dezember 2006



Inhaltsverzeichnis

1	Organisatorisches	5
1.1	Vorlesung	5
1.1.1	Ausnahme:	5
1.2	Dozent	5
1.2.1	Termin	5
1.3	Homepage	5
1.4	Übungsbetrieb	6
1.4.1	Organisatoren	6
1.4.2	Übungsgruppen	6
1.5	Klausur	6
1.6	Literatur	6
2	Wozu theoretische Informatik?	7
2.1	Antworten auf folgende Fragen:	7
2.2	Game of Life:	7
3	Kapitel 1 - Theoretische Berechenbarkeit	8
3.1	(anschaulich)	8
3.1.1	Erklärungen	8
3.2	Satz 1.1	9
3.3	Zwei Ansätze zur Präzisierung	10
3.4	Die μ -rekursiven Funktionen	11
3.4.1	Zu A: Anfangsfunktionen	11
3.4.2	Zu B: <u>Schemata</u>	11
3.5	Definition	11
3.5.1	Beispiele	12
3.6	Prädikate	13
3.6.1	Beispiel	13
3.6.2	charakteristische Funktion	13
3.6.3	Lemma	13
3.7	Lemma:	14
3.8	μ -Operator	14
3.9	Definition	15
3.9.1	Satz	15
3.10	Definition	15
3.10.1	Satz	16
3.11	Rekapitulation	16
3.11.1	1. Ansatz	17
3.12	Turing-Maschinen	18
3.12.1	Modell der Turing-Maschine (TM)	18
3.12.2	Definition	19
3.12.3	Beispiel	20

3.12.4	Definition	21
3.13	Rekapitulation	21
3.14	Definition	22
3.15	Definition	22
3.16	Satz	22
3.16.1	Beweis	23
3.17	Lemma	23
3.17.1	Beweis:	23
3.18	Lemma:	24
3.18.1	Beweis:	24
3.19	Lemma:	25
3.19.1	Beschreibung von H:	26
3.20	Lemma:	27
3.20.1	Beweis:	27
3.21	Lemma 1.8:	28
3.22	Theorem 1.6	29
3.22.1	Beweis:	29
3.23	Lemma 1.9	30
3.24	Fortführung des Beweises von Theorem 1.6	30
3.25	Beschränkter μ -Operator:	31
3.25.1	Beweis	31
3.26	Lemma 1.10	31
3.27	Beweis von Lemma 1.10:	32
3.27.1	Abkürzungen:	32
3.27.2	Beweis von Lemma 1.8	33
3.28	Beweis:	33
3.28.1	Definition:	33
3.29	Beweis von Lemma 1.8	33
3.29.1	Definition:	34
3.30	Beweis von Lemma 1.9	34
3.31	Folgerungen	35
3.31.1	Beispiel:	36
3.32	Definition	36
3.33	starke Version von Kleene's Normalformtheorem	37
3.34	Entscheidbarkeit	37
3.34.1	Gödelnummer	38
3.34.2	Korollar (Unentscheidbarkeit des allgemeinen Halteproblems)	39
3.35	Weitere Beispiele unentscheidbarer Probleme	39
3.35.1	Wortproblem	39
3.35.2	Game of Life	40
3.35.3	10-tes Hilbert-Problem	40
3.35.4	Domino-Problem	40
3.36	Definition (rekursiv/entscheidbar)	41

3.36.1	Lemma	41
3.36.2	Theorem	41
3.36.3	Weltbild	42
3.36.4	Lemma	42
3.37	Was haben folgende Probleme gemeinsam?	42
3.37.1	k-Clique	42
3.37.2	Satisfiability (SAT) - Erfüllbarkeit	43
3.37.3	Traveling Salesman Problem (TSP)	43
3.37.4	Gemeinsamkeiten	43
3.38	nichtdeterministische Turingmaschinen	44
3.38.1	Definition	44
3.38.2	Rechenzeit	44
3.39	Definition NP (nichtdeterministisch polynomiell)	45
3.39.1	Beispiele:	45
3.40	Definition (polynomielle Reduktion)	47
3.41	Definition (NP-hart/vollständig)	47
3.42	Theorem (Cook)	47
3.42.1	Definition (3 SAT)	50
3.42.2	Satz	50
3.42.3	Bemerkung	51
3.43	Theorem (Clique ist NP-vollständig)	51
3.44	Vertex Cover	52
3.44.1	Theorem	52
3.45	(0,1)-Integer Programming	53
3.45.1	LP	54
3.45.2	Theorem	54
3.46	weitere NP-vollständige Probleme	56
3.46.1	Hamiltonian Circle (HC)	56
3.46.2	Theorem (HC ist NP vollständig)	56
3.46.3	Folgerung:	60
3.46.4	Theorem (TSP NP-vollständig)	61
3.46.5	Bemerkungen	61
3.46.6	Minimum Weight Triangulation	62
3.46.7	Minimum Dilation Graph	62
3.46.8	Partition:	62
3.46.9	Warehouseman- Problem	63
3.47	Wie geht man mit NPvollständigen Problemen um?	63
3.47.1	Prüfe, ob Problemkomplexität abnimmt, wenn einen oder mehrere Parameter beschränkt.	63
3.47.2	Prüfe ob eine Approximation des Optimums ausreicht	65
3.47.3	Versuche, den Suchraum zu begrenzen (Branch and Band)	67
3.48	Weltbild	69
3.49	RAM	69

3.49.1	Was kann RAM?	70
3.49.2	Operanden für RAM-Befehle:	70
3.49.3	Wert eines Operanden: $a: v(a)$	71
3.49.4	Befehle:	71
3.50	Kostenmaß e	71
3.50.1	Einheitskostenmaß	71
3.50.2	Logarithmisches Kostenmaß	71
3.50.3	Kosten der RAM-Befehle im logarithmischen Maß: . .	71
3.50.4	Wie verhalten sich RAM und TM zueinander?	72
4	Kryptographie	74
4.1	Situation	74
4.2	Ansatz:	74
4.3	Private-Key-Verfahren:	75
4.3.1	Problem:	75
4.4	Public-Key-Verfahren:	75
4.4.1	prominentestes Beispiel: RSA-Verfahren	76
4.4.2	Vorbereitung: Euler'sche ϕ -Funktion	76
4.4.3	76
4.4.4	Sätzchen (Kleiner Fermat)	76
4.4.5	RSA-Verfahren:	77
4.5	Mathematische Grundlagen	79
4.5.1	Gruppe	79
4.5.2	$\mathbb{Z}/n\mathbb{Z}$ - kommutativer Ring der Äquivalenzklassen, Gruppe bzgl. \odot und $\bar{1}$, abelsch	79
4.5.3	Euklidischer Algorithmus	80
4.5.4	Lemma: (ggT - ganzzahlige Linearkombination)	80
4.5.5	Kongruenz	80

1 Organisatorisches

1.1 Vorlesung

Montag 11 - 13 Uhr HS D
Mittwoch 11 - 13 Uhr HS D

1.1.1 Ausnahme:

Am 27.11 und am 29.11 fallen die Vorlesungen aus.

Sie werden nachgeholt am: 20.11 9 - 11 Uhr HS D
11.12 17 - 19 Uhr HS D

Am 18.12 und am 20.12 fallen die Vorlesungen aus.

Sie werden im nächsten Jahr angeholt.

1.2 Dozent

Professor Dr. Rolf Klein
Institut für Informatik I, Raum N310, Römerstraße 164, 53117 Bonn
rolf.klein@uni-bonn.e
0228-734321

1.2.1 Termin

bei Frau Knepper Telefon 0228-734333

1.3 Homepage

<http://web.informatik.uni-bonn.de/I/Lehre/Vorlesungen/InfoIII-WS0607/index.html>

1.4 Übungsbetrieb

- mindestens 50 % der Punkte für Klausurzulassung
- mindestens 2mal vorgerechnet haben

1.4.1 Organisatoren

Name	eMail
Ansgar Grüne	gruene@cs.uni-bonn.de
Martin Köhler	koehler@cs.uni-bonn.de

1.4.2 Übungsgruppen

Montag	13-15 Uhr	N 1
	15-17 Uhr	N 1
Dienstag	13-15 Uhr	N 327
Mittwoch	9-11 Uhr	N 327
	11-13 Uhr	N 1
Freitag	9-11 Uhr	N 1
	11-13 Uhr	N 1
	12-14 Uhr	N 327

1.5 Klausur

Es wird eine 90 bis 120 minütige Klausur etwa 2 Wochen nach Beginn der vorlesungsfreien Zeit (14.2.2007) statt finden.

1.6 Literatur

- allgemein *Norbert Blum* Theoretische Informatik, 2. Auflage, Oldenburg
 Ingo Wegener Theoretische Informatik
 Uwe Schöning Theoretische Informatik u.a.
 Katrin Elk, Lutz Priese Theoretische Informatik, Springer 2000
 Norbert Blum Einführung in Formale Sprachen, Berechenbarkeit,
 Informations- und Lerntheorie, Oldenbourg 2006
- P-NP *Garey Johnson* Computers and Intractability: A Guide to NP-
 Completeness, Freeman & Co, '79
- Kryptographie *F. Bauer* Entzifferte Geheimnisse: Methoden und Maximen der
 Kryptologie
 J.vz Gathen BIT

2 Wozu theoretische Informatik?

2.1 Antworten auf folgende Fragen:

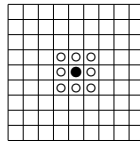
1. Was ist prinzipiell auf Computern berechenbar?
2. Wie effizient lassen sich wichtige, häufig wiederkehrende Probleme lösen?
 - (a) sortieren
 - (b) Travelling Salesman
 - (c) ...
 - (d) Faktorisieren von großen ganzen Zahlen
3. Wie geht man mit Problemen um, die sich aus prinzipiellen Gründen nicht effizient lösen lassen?
 - (a) ignorieren
 - (b) approximieren
 - (c) randomisieren
4. Erwerb eines Werkzeugkastens mit grundlegenden, häufig wiederkehrenden algorithmischen Prinzipien (Greedy, Divide & Conquer, dynamisches Programmieren, Sweep, Las Vegas, Monte-Carlo...)
5. Training für Denkmuskel
 Beweise, Gegenbeispiele ↔ Fehlersuche in Software
 Strukturierung von Texten ↔ Modularisierung von Softwaresystemen

6. Spaß

zu wissen, dass die eigene Lösung besser/optimal ist als die der Konkurrenz

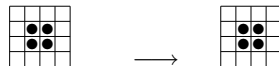
7. Anwendungsrelevanz

2.2 Game of Life:



- Jede Zelle hat 8 Nachbarn
- Generation $G_0, G_1, G_2, \dots, G_i, G_{i+1}, \dots$
 - G_0 = Anfangspopulation: endlich viele lebendige Zellen
- G_{i+1} aus G_i durch simultane Anwendung folgender Regeln:
 1. Eine lebende Zelle mit 2 oder 3 lebenden Nachbarn in G_i bleibt in G_{i+1} lebendig
 2. Eine tote Zelle mit genau 3 lebenden Nachbarn in G_i wird in G_{i+1} lebendig

Beispiele:



natürliche Frage: Was wird aus einer Anfangspopulation G_0 ? (genauer: stirbt G_0 aus?)

Diese Frage ist unentscheidbar!

D.h. man kann kein Programm schreiben¹, das von einer beliebigen Anfangspopulation G_0 entscheiden kann, ob G_0 ausstirbt.

¹unscharf → muss präzisiert werden

3 Kapitel 1 - Theoretische Berechenbarkeit

3.1 (anschaulich)

ist $f : \begin{cases} \mathbb{N}_0 \times \mathbb{N}_0 \times \mathbb{N}_0 \rightarrow \mathbb{N}_0 \\ (a, b, c) \mapsto (a + b)c \end{cases}$ berechenbar?

intuitiv: ja, denn wir kennen einen Algorithmus, der f berechnet (schriftl. Addition und Multiplikation)

3.1.1 Erklärungen

\mathbb{N}_0 die Menge der natürlichen Zahlen $\{0, 1, 2, 3, \dots\}$
 $f : \mathbb{N}_0^r \rightarrow \mathbb{N}_0$ heißt $f : \begin{cases} \mathbb{N}_0 \times \mathbb{N}_0 \times \dots \times \mathbb{N}_0 \rightarrow \mathbb{N}_0 \\ \xi = (x_1, x_2, \dots, x_r) \mapsto f(x_1, x_2, \dots, x_r) \end{cases}$

Algorithmus² - Rechenvorschrift

- besteht aus Text endlicher Länge
- Berechnung erfolgt schrittweise durch Ausführung elementarer Operationen (= Rechenvorschriften)
- zu jedem Zeitpunkt steht eindeutig fest, welche Rechenvorschrift als nächstes anzuwenden ist.

[!Achtung!] deckt auch randomisierte Verfahren ab! Denn: Ein Rechenschritt könnte z.B. ein Münzwurf sein.

- Der nächste Rechenschritt hängt ab von
 - o Algorithmus
 - o Eingabe
 - o die zuvor ausgeführten Schritte

- **Beispiel:** Kochrezept

Eingabe \longrightarrow *Algorithmus* \longrightarrow *Ausgabe*

Eine Funktion $f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$ heißt (intuitiv) berechenbar, wenn es einen Algorithmus gibt, der f berechnet.

3.2 Satz 1.1

Es gibt nicht-berechenbare Funktionen $f : \mathbb{N}_0^r \rightarrow \{0, 1\}$.

²al-Khawarizmi: arabischer Mathematiker 285 n. Chr.

Beweis:

f berechenbar \Rightarrow es gibt Algorithmus A , der f berechnet.

A ist ein Wort endlicher Länge über einem endlichen Alphabet (z.B. $\Sigma = \{a, b, c, \dots, z\}$).

Es gibt (nur) abzählbar unendlich viele³ Wörter endlicher Länge über Σ .

\Rightarrow es gibt nur abzählbar unendlich viele Algorithmen

\Rightarrow es gibt nur abzählbar unendlich viele berechenbare Funktionen $f : \mathbb{N}_0^r \rightarrow \mathbb{N}_0$

Aber: Die Menge aller Funktionen $f : \mathbb{N}_0^r \rightarrow \{0, 1\}$ ist überabzählbar!

(indirekter) **Beweis:**

Angenommen, die Menge dieser Funktionen wäre doch abzählbar. (sei $r=1$)

Sei $f_0(x), f_1(x), \dots, f_i(x), \dots$ eine Aufzählung.

Definiere $f(x)$ folgendermaßen: $f(i) := \begin{cases} 1, & f_i(i) = 0 \\ 0, & f_i(i) = 1 \end{cases}$

$\Rightarrow f$ kann in der Aufzählung $f_0(x), f_1(x), \dots, f_i(x), \dots$ nicht vorkommen, denn f unterscheidet sich von jedem f_j beim Wert $f(j)$

\Rightarrow Widerspruch \Rightarrow Behauptung.

qed

Fazit: Es gibt nicht-berechenbare Funktionen.

Wäre nicht schlimm, wenn es sich dabei um "Exoten" handeln würde.

Aber:

Game of Life ist nicht berechenbar. (schlecht für Biologen)

$$f : \{G_0 \subseteq \mathbb{Z} \times \mathbb{Z} \text{ endlich}\} \mapsto \{0, 1\}$$

$$G_0 \mapsto \begin{cases} 1, & \text{falls } G_0 \text{ überlebt} \\ 0, & \text{falls } G_0 \text{ ausstirbt} \end{cases}$$

³Eine Menge S heißt abzählbar unendlich,
 \Leftrightarrow es gibt eine surjektive Abbildung $\mathbb{N}_0 \rightarrow S$
 \Leftrightarrow es gibt eine injektive Abbildung: $S \rightarrow \mathbb{N}_0$.

Unentscheidbarkeit des Halteproblems⁴ ist nicht berechenbar. (schlecht für die Informatik)

$$f : \{A; A \text{ ist endlicher Algorithmus über } \Sigma\} \rightarrow \begin{cases} 1, & \text{falls } A \text{ bei Input } x \text{ jemals anhält} \\ 0, & \text{falls nicht} \end{cases}$$

3.3 Zwei Ansätze zur Präzisierung

Was heißt berechenbar eigentlich genau?

3.4 Die μ -rekursiven Funktionen

Idee: A Definiere einige wenige Grund-(Anfangs-)Funktionen
 B Definition einiger Schemata, um aus vorhandenen Funktionen neue zu bilden.

3.4.1 Zu A: Anfangsfunktionen

$$\text{- Konstante Funktionen } C_s^r : \begin{array}{ccc} \mathbb{N}_0^r & \rightarrow & \mathbb{N}_0 \\ \xi = (x_1, x_2, \dots, x_r) & \mapsto & s \end{array}$$

$$\text{- Nachfolgerfunktionen } N : \begin{array}{ccc} \mathbb{N}_0 & \rightarrow & \mathbb{N}_0 \\ x & \mapsto & x + 1 \end{array}$$

$$\text{- Projektionsfunktion } p_i^r : \begin{array}{ccc} \mathbb{N}_0^r & \rightarrow & \mathbb{N}_0 \\ \xi = (x_1, \dots, x_i, \dots, x_r) & \mapsto & x_i \end{array}$$

⁴Das heißt: Die semantische Korrektheit eines beliebigen Programmes kann man nicht algorithmisch testen.

semantische Korrektheit: tut das Programm, was es tun soll?

Im Gegensatz dazu: Syntaktische Korrektheit: Entspricht das Programm den Regeln der Sprache? (sehr wohl effizient entscheidbar, Compiler)

3.4.2 Zu B: Schemata

- Substitution: Zu $g_1, \dots, g_r : \mathbb{N}_0^m \rightarrow \mathbb{N}_0$
und $g : \mathbb{N}_0^r \rightarrow \mathbb{N}_0$ definiere

$$\mathbb{N}_0^m \rightarrow \mathbb{N}_0$$

$$h : \xi \mapsto g(g_1(\xi), g_2(\xi), \dots, g_r(\xi))$$

- Primitive Rekursion: Zu $g : \mathbb{N}_0^r \rightarrow \mathbb{N}_0$ und $f : \mathbb{N}_0^{r+2} \rightarrow \mathbb{N}_0$ definiere

$$h : \mathbb{N}_0^{r+1} \rightarrow \mathbb{N}_0$$

$$\text{mit } h(\xi, 0) := g(\xi)$$

$$h(\xi, y+1) := f(\xi, y, h(\xi, y))$$

$$\text{und } \xi \in \mathbb{N}_0^r$$

damit:

$$\begin{aligned} h(\xi, 0) &= g(\xi) \\ h(\xi, 1) &= f(\xi, 0, g(\xi)) \\ h(\xi, 2) &= f(\xi, 1, f(\xi, 0, g(\xi))) \end{aligned}$$

3.5 Definition

1. Die Klasse \mathcal{P} der primitiv rekursiven Funktionen ist die kleinste Funktionenklasse, die

- alle Anfangsfunktionen enthält
- gegen Substitution und primitive Rekursion abgeschlossen ist

2. Eine Funktion f heißt primitiv rekursiv $:\Leftrightarrow f \in \mathcal{P}$

3.5.1 Beispiele

1. $a(x, y) = x + y$

Frage $a \in \mathcal{P}$? ja! denn a entsteht auf folgende Weise:

$$\left. \begin{aligned} a(x, 0) &= P_1^1(x) \\ \underbrace{a(x, y+1)}_{(x+y)+1} &= N(P_3^3(x, y, a(x, y))) \end{aligned} \right\} \text{primitive Rekursion}$$

2. $m(x, y) = x \cdot y$

definiere $f(x, y, z) := a(P_3^3(x, y, z), P_1^1(x, y, z))$, dies ist eine Substitution $\Rightarrow f \in \mathcal{P}$

damit:

$$\left. \begin{aligned} m(x, 0) &:= C_0^1(x) \\ m(x, y+1) &:= f(x, y, m(x, y)) \\ &= m(x, y) + x = x \cdot y + x = x(y+1) \end{aligned} \right\} \Rightarrow m \in \mathcal{P}$$

$$3. \ l = x^y \in \mathcal{P} \quad \left. \begin{aligned} l(x, 0) &:= C_1^1(x) \\ l(x, y+1) &:= m(x, l(x, y)) \\ &= m(P_3^3(x, y, l(x, y)), P_1^3(x, y, l(x, y))) \end{aligned} \right\} \Rightarrow l \in \mathcal{P}$$

$$4. \text{ Vorgänger } V(x) := \begin{cases} 0, & x = 0 \\ x-1, & x > 0 \end{cases}$$

$$\left. \begin{aligned} v(0) &:= C_0^1(x) \\ v(y+1) &:= P_1^2(y, V(y)) \end{aligned} \right\} \text{ primitive Rekursion } \Rightarrow V \in \mathcal{P}$$

$$5. \text{ Modifizierte Differenz } d(x, y) := \begin{cases} x-y & \text{falls } y \leq x \\ 0 & \text{sonst} \end{cases}$$

$$\left. \begin{aligned} d(x, 0) &:= P_1^1(x) \\ d(x, y+1) &:= V(P_3^3(x, y, d(x, y))) \end{aligned} \right\} \text{ prim. Rekursion } \Rightarrow d \in \mathcal{P}$$

Schreibweise: $d(x, y) = x \overset{\bullet}{-} y$

6. Fakultät $h(x) = x!$ ist primitiv rekursiv

$$h: \mathbb{N}_0$$

$\rightarrow \mathbb{N}$

$$h(x) = x \cdot h(x-1)$$

$$h(0) = 1 \quad \Rightarrow g = C_1^0 = 1$$

jetzt wird f gesucht:

$$\begin{aligned} h(x+1) &= (x+1) \cdot h(x) \\ &= m(x+1, h(x)) \\ &= m(N(x), h(x)) \\ &= m(N(P_1^2(x, h(x))), P_2^2(x, h(x))) \end{aligned}$$

\Rightarrow primitiv rekursiv, da Projektion/Nachfolgerfunktion/Multiplikation primitiv rekursiv und deren Substitutionen auch.

3.6 Prädikate

⁵ zur Beschreibung von Eigenschaften

$P \subseteq \mathbb{N}_0^r$ heißt ein r-stelliges Prädikat.

(Intuitiv: "ξ hat Eigenschaft P" $\Leftrightarrow \xi \in P$)

⁵Prädikat \Leftrightarrow Relation

3.6.1 Beispiel

$r = 1, P = \{p \in \mathbb{N}_0; p \text{ Primzahl}\}$

3.6.2 charakteristische Funktion

Zu Prädikat P gehört Funktion

$$\chi_p : \mathbb{N}_0^r \rightarrow \{0, 1\}$$

$$\xi \mapsto \begin{cases} 1 & \xi \in P \\ 0 & \text{sonst} \end{cases}$$

χ_p heißt charakteristische Funktion des Prädikats P , Prädikat P heißt primitiv rekursiv $:\Leftrightarrow \chi_p \in \mathcal{P}$.

3.6.3 Lemma

Mit P und Q sind auch die Prädikate

$$\begin{aligned} P \wedge Q &:= P \cap Q \\ P \vee Q &:= P \cup Q \\ \neg P &:= \mathbb{N}_0 \setminus P \end{aligned}$$

primitiv rekursiv.

Beweis:

$$\begin{aligned} \chi_{P \wedge Q} &= \chi_P \cdot \chi_Q \\ \chi_{P \vee Q} &= sg(\chi_P + \chi_Q) \\ \chi_{\neg P} &= 1 - \chi_P \end{aligned}$$

⁶ Primitiv rekursive Prädikate lassen sich gut zu Definitionen mit Fallunterscheidung verwenden:

3.7 Lemma:

Seien P_1^r, \dots, P_k^r primitiv rekursive Prädikate, die als Mengen paarweise disjunkt sind.

Seien $f_1, \dots, f_k : \mathbb{N}_0^r \rightarrow \mathbb{N}_0 \in \mathcal{P}$

Dann ist auch folgende Funktion g in \mathcal{P} :

$$g(\xi) := \begin{cases} f_1(\xi), & \text{falls } P_1(\xi) \\ \vdots \\ f_k(\xi), & \text{falls } P_k(\xi) \\ 0 & \text{sonst} \end{cases}$$

⁶ $sg(x) := \begin{cases} 1 & x > 0 \\ 0 & x = 0 \end{cases} \quad sg \in \mathcal{P}$

Beweis:

$$g(\xi) = \sum_{i=1}^k f_i(\xi) \chi_p(x) = a(f_1(\xi) \cdot \chi_{P_1}(\xi), a(f_2(\xi) \cdot \chi_{P_2}(\dots)))$$

Frage: Gibt es “berechenbare“ Funktionen, die nicht in \mathcal{P} liegen?

Parallele zum Programmieren

Beim Programmieren: \mathcal{P} = Klasse der Funktionen, die man mit FOR-Schleifen berechnen kann.

Fehlt: Mächtigkeit von WHILE (o. ä.)

3.8 μ -Operator

Dazu neues Schema: Sei $f : \mathbb{N}_0^{r+1} \rightarrow \mathbb{N}_0$, dann definiere:

$$\mu f : \begin{array}{l} \mathbb{N}_0^r \rightarrow \mathbb{N}_0 \\ \xi \mapsto \begin{cases} \text{das kleinste } y \text{ mit } f(\xi, y) = 0, \text{ falls dies existiert} \\ \text{undefiniert sonst} \end{cases} \end{array}$$

falls gilt: $\forall \xi \exists y : f(\xi, y) = 0$, dann sagen wir:

μf entsteht aus f durch Anwendung des μ -Operators im Normalfall

3.9 Definition

1. \mathcal{F}_μ ist die kleinste Klasse von Funktionen, die
 - alle Anfangsfunktionen enthält⁷
 - abgeschlossen ist gegen Substitution, primitive Rekursion⁸
 - abgeschlossen gegen μ -Operator im Normalfall
2. $\mathcal{F}_\mu^{par(tiell)}$: ebenso, aber ohne Normalfall (diese Funktionen sind i. a. nicht überall definiert)

3.9.1 Satz

$$\mathcal{P} \neq \mathcal{F}_\mu.$$

Idee: Konstruiere eine Funktion, die “schneller wächst als alle Funktionen in \mathcal{P} “

⁷ \mathcal{P}

⁸ \mathcal{P}

Idee: Extrapoliere folgendes Schema:

$$\begin{aligned} f_1(x, y) &= x + y \\ f_2(x, y) &= x \cdot y \\ f_3(x, y) &= x^y \\ f_4(x, y) &= x^{x^{x^{x^x}}} \} \text{ (y-mal)} \\ &\vdots \\ f_{n+1}(x, y+1) &= f_n(x, f_{n+1}(x, y)) \end{aligned}$$

Beobachtung: Das schnelle Wachstum hängt nicht so sehr vom Wert von x ab ($x = 2$)!
Deshalb kompaktere Definition:

$$\left. \begin{aligned} f_0(y) &:= y + 1 \\ f_{n+1}(0) &:= f_n(1) \\ f_{n+1}(y+1) &:= f_n(f_{n+1}(y)) \end{aligned} \right\} \text{ primitive Rekursion für } f_{n+1}$$

Man kann leicht durch Induktion über n beweisen: $\forall n : f_n \in \mathcal{P}$.

3.10 Definition

$A(x, y) := f_x(y)$ Ackermann-Funktion

3.10.1 Satz

$A \notin \mathcal{P}$

Beweis beruht auf folgendem

Hauptlemma $\forall f \in \mathcal{P} \quad \exists k \in \mathbb{N}_0 \quad \forall \xi : f(\xi) \leq A(k, \Sigma\xi)^9$

Beweis des Hauptlemmas: Induktion über den Aufbau von \mathcal{P} .

Beweis des Satzes aus dem Hauptlemma

(indirekt) Angenommen, $A \in \mathcal{P}$

\Rightarrow Hauptlemma $\exists k \in \mathbb{N}_0 \forall x : f(x) \leq A(k, x)^{10}$

Diagonalisierung

\Rightarrow für $x := k : A(k, k) + 1 \leq A(k, k)$ Widerspruch

$\Rightarrow A \notin \mathcal{P}$.

Frage $A \in F_\mu$? Ja! Beweis später.

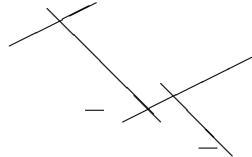
Klar Ackermann-Funktion wirkt etwas künstlich.

⁹ $\xi = (x_1, \dots, x_r)$

$\Sigma\xi := x_1 + \dots + x_r$

¹⁰ $f(x) = A(x, x) + 1$

Aber: Sie „kommt in der Natur vor“
 betrachte m Liniensegmente im \mathbb{R}^2



Die von unten sichtbaren Abschnitte,
 die untere Kontur besteht aus $O(n\alpha(n))$ vielen Abschnitten, wobei $\alpha(n) :=$
 das kleinste x mit $A(x, x) \geq n$ extrem langsam wachsende Funktion.

3.11 Rekapitulation

Frage: Was heißt "berechenbar"?

3.11.1 1. Ansatz

Definition von Funktionenklassen

\mathcal{P}	\subset \neq	\mathcal{F}_μ
Klasse der primitiv rekursiven Funktionen		Klasse der μ -rekursiven Funktionen
Konstante C_i^r , Nachfol- ger N , Projektion P_i^r , abgeschlossen gegen Substitution und primi- tive Rekursion	zum Beweis: Konstruktion der Ackermann- Funktion A , Nachweis $A \notin \mathcal{P}$	$\mathcal{P} + \mu$ -Operator

zur Erinnerung an μ -Operator

Lemma: Sei $f \in \mathcal{F}_\mu$ bijektiv. Dann liegt auch f^{-1} in \mathcal{F}_μ .

Beweis: Sei h definiert durch $h(v, w) := \left(v \dot{-} f(w) \right) + \left(f(w) \dot{-} v \right)$
 $\Rightarrow h \in \mathcal{F}_\mu$ und $\forall v, w : h(v, w) = 0$ genau dann, wenn $v = f'(w)$
 Setze $g(w) := \mu h(v) =$ das kleinste w mit $h(v, w) = 0$
 $=f$ injektiv dasjenige w mit $v = f'(w)$

- zu jedem Band gehört Schreib/Lesekopf; er kann
 - das Zeichen auf dem aktuellen Feld lesen
 - das Zeichen auf dem aktuellen Feld überschreiben
 - sich um ein Feld nach rechts oder links bewegen, aber nicht über das linke Bandende hinaus

1 Schritt \triangleq Ausführung einer Instruktion, z. B.

wenn Zustand = q und sich unter dem i -ten Kopf das Zeichen $s_i \in \sigma$ befindet, für $1 \leq i \leq k$

dann – schreibe mit dem i -ten Kopf das Zeichen s'_i ,
 – bewege den i -ten Kopf um $m_i \in \{0, -1, 1\}$,
 – gehe in den Folgezustand q' über.

(simuliert Verhalten von Magnetbändern)

präzise:

k -Halbband TM ,

M ist ein Tupel $(Q, \Sigma, \delta, q_0, F)$

Σ endliches Bandalphabet,

s_1, s_2, \dots, s_k aktuelle Zeichen

s'_1, s'_2, \dots, s'_k aktuelle Zeichen

Q endliche Zustandsmenge,

q_0 Anfangszustand,

q aktueller Zustand,

q' Folgezustand

$F = \{q \in Q; \forall a \in \Sigma^k : \delta(q, a) \text{ undefiniert}\}$ Menge der Endzustände

m_1, \dots, m_k Kopfbewegungen

δ Übergangsfunktion $\delta : \begin{array}{ccc} Q \times \Sigma^k & \longrightarrow & Q \times \Sigma^k \times \{-1, 0, 1\}^k \\ (q, s_1, \dots, s_k) & \mapsto & (q', s'_1, \dots, s'_k, m_1, \dots, m_k) \end{array}$

[!Achtung!] δ könnte nur eine partielle Funktion sein

Zahlen werden binär dargestellt, also durch

$bin : \mathbb{N}_0 \rightarrow \{0, 1\}^+ =$ Menge der endlichen Worte über 0/1 der Länge ≥ 1

$n = \sum_{i=1}^s w_i 2^{s-1} \mapsto w_1, w_2, \dots, w_s$, mit $w_i \in \{0, 1\}$

- von rechts nach links
- speichert Übertrag im Zustand q_2 : Übertrag 1
 q_3 : übertrag 0

falls zum Schluss (an der Stelle ganz links) "bertrag 1 (d.h. input $bin(x) = 1 \dots 1$)

- schreibe 1 links auf Band 1

Kopiere Band 2 hinter Inhalt von Band 1 (Band 1 := Band 1 · Band 2)

lösche Band 2

bewege Köpfe nach links.

$$M = (\{q_0, q_1, \dots, q_7\}, \{\$, 0, 1, -, \#\}, q_0, \{q_7\})$$

q	s_1	s_2	q'	s'_1	s'_2	m_1	m_2	
-----	-------	-------	------	--------	--------	-------	-------	--

q_0	1/0		q_0	-	1/0	+1	+1	Band 2:= Band 1, Band 1 löschen
-------	-----	--	-------	---	-----	----	----	------------------------------------

q_0	-		q_1			-1	-1	
-------	---	--	-------	--	--	----	----	--

q_1	-		q_1	-		-1		Kopf 1 nach links
-------	---	--	-------	---	--	----	--	-------------------

q_1	\$		q_2			+1		
-------	----	--	-------	--	--	----	--	--

q_2		1	q_2		0		-1	addiere 1 zu Band 2 von links nach rechts $q_2 =$ Übert- rag 1, $q_3 =$ Über- trag 0
-------	--	---	-------	--	---	--	----	--

q_2		0	q_3		1		-1	
-------	--	---	-------	--	---	--	----	--

q_3		1/0	q_3		1/0		-1	
-------	--	-----	-------	--	-----	--	----	--

q_3		\$	q_4		\$		+1	
-------	--	----	-------	--	----	--	----	--

q_2		\$	q_4	1	\$	+1	+1	
-------	--	----	-------	---	----	----	----	--

q_4		1/0	q_4	1/0	-	+1	+1	hänge Band 2 an Band 1 an
-------	--	-----	-------	-----	---	----	----	------------------------------

q_4		-	q_5			-1	-1	
-------	--	---	-------	--	--	----	----	--

q_5	1/0		q_5			-1		Kopf 1 nach links
-------	-----	--	-------	--	--	----	--	-------------------

q_5	\$		q_6			+1		
-------	----	--	-------	--	--	----	--	--

q_6		-	q_6				-1	Kopf 2 nach links
-------	--	---	-------	--	--	--	----	-------------------

q_6	\$		q_7			+1		
-------	----	--	-------	--	--	----	--	--

⇒ rein theoretisches Modell

3.12.4 Definition

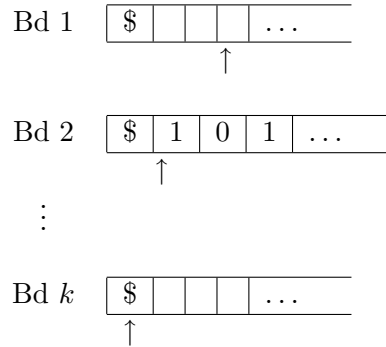
$\mathcal{F}_{TM}^{(par)} := \{f : \mathbb{N}_0^r \longrightarrow (\mathbb{N}_0 \text{ (partiell)}); \text{ es existiert TM } M \text{ mit } M \text{ berechnet } f\}$

Ziel

$$\mathcal{F}_{TM} = \mathcal{F}_\mu$$

3.13 Rekapitulation

k-Band - Turingmaschine $M = (Q, \Sigma, \delta, q_0, F)$

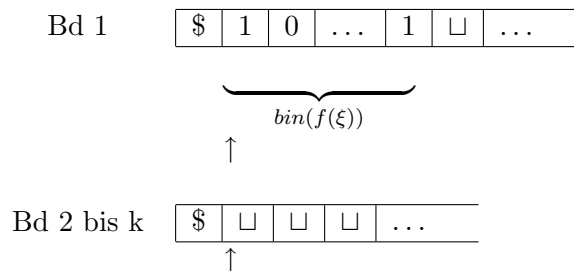


Sei $f : D(f) \rightarrow \mathbb{N}_0$ eine n-stellige Funktion mit Definitionsbereich $D(f)$.

3.14 Definition

M berechnet $f \Leftrightarrow \forall \xi \in \mathbb{N}_0 : \xi = (x_1, x_2, \dots, x_r)$

- $e \notin D(f) \Rightarrow$ M, angesetzt auf ξ (wie oben), stoppt nicht
- $\xi \in D(f) \Rightarrow$ M, angesetzt auf ξ stoppt nach endlich vielen Schritten in Endzustand $q \in F$ mit der Bandinschrift



3.15 Definition

$\mathcal{F}_{TM} := \{f : \mathbb{N}_0^r \rightarrow \mathbb{N}_0, r \geq 0 \text{ und es gibt Turingmaschine M mit M berechnet } f\}$
 = Klasse der totalen (d. h. überall definierten) turingberechenbaren Funktionen

$\mathcal{F}_{TM}^{par} := \{f : D(f) \rightarrow \mathbb{N}_0 \text{ und } r \geq 0 \text{ existiert Turingmaschine M mit M berechnet } f\}$
 = Klasse der partiellen turingberechenbaren Funktionen.

Ziel: $\mathcal{F}_{TM} = \mathcal{F}_\mu$, $\mathcal{F}_{TM}^{par} = \mathcal{F}_\mu^{par}$ ¹²

3.16 Satz

$$\mathcal{F}_\mu^{(par)} \subseteq \mathcal{F}_{TM}^{(par)}$$

3.16.1 Beweis

durch Induktion über den Aufbau von \mathcal{F}_μ ($\mathcal{F}_\mu^{(par)}$ entsprechend).
wir zeigen:

- alle Anfangsfunktionen sind in \mathcal{F}_{TM}
- \mathcal{F}_{TM} ist abgeschlossen gegen Substitution, primitive Rekursion und μ -Operator

3.17 Lemma

Sei $h : \mathbb{N}_0^m \rightarrow \mathbb{N}_0$, $\xi \mapsto f(g_1(\xi), \dots, g_r(\xi))$ und seien f, g_1, \dots, g_r durch k -Band-Turingmaschinen F, G_1, \dots, G_r berechenbar. Dann läßt sich h durch eine $(k+2)$ -Band-Turingmaschine H berechnen.

3.17.1 Beweis:

durch schematische Beschreibung des Programms von H .

zunächst: Beschreibung von Maschinen G'_i , $1 \leq i \leq r$.

G'_i : Band 3 := Band 1:

lasse G_i auf den Bändern 3, 4, \dots , $k+2$ laufen

für $i = 1$: Band 2 := Band 3;

$i = 2, \dots, r$: Band 2 := Band 2 # Band 3

H: $G'_1; G'_2; \dots; G'_r$

2 : $\underbrace{bin(g_1(\xi))}_{\text{von } G'_1} \# \underbrace{bin(g_2(\xi))}_{\text{von } G'_2} \# \dots \# \underbrace{bin(g_r(\xi))}_{\text{von } G'_r}$

lasse F auf den Bändern 2, 3, \dots , $k+2$ laufen;

Band 1 := Band 2;

lösche Band 2;

Köpfe nach vorn;

Bemerkung: $f : \mathbb{N}_0^r \rightarrow \mathbb{N}_0 \in \mathcal{P}(\in \mathcal{F}_\mu, \mathcal{F}_\mu^{par})$ und π Permutation von r Elementen

$\Rightarrow f_\pi(\xi) := f(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(r)})$ ebenfalls in $\mathcal{P}(\in \mathcal{F}_\mu, \mathcal{F}_\mu^{par})$.

¹²Abkürzung $\mathcal{F}_\mu^{(par)}$

Beweis: $f_\pi(\xi) = f(P_{\pi(1)}^r(\xi), P_{\pi(2)}^r(\xi), \dots, P_{\pi(r)}^r(\xi))$

Fazit: Auf Variablenreihenfolge kommt es nicht an.

3.18 Lemma:

Sei

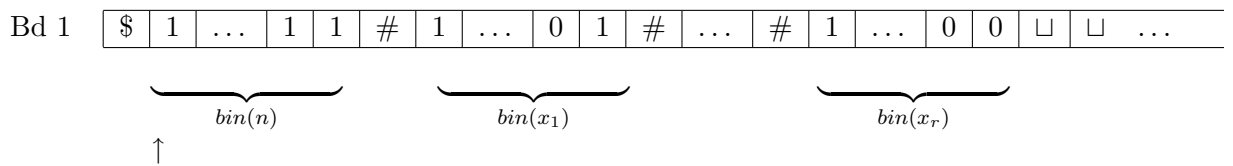
$$\begin{aligned} h : \mathbb{N}_0^{r+1} &\rightarrow \mathbb{N}_0 \\ (0, \xi) &\mapsto g(\xi) \\ (n+1, \xi) &\mapsto f(n, h(n, \xi), \xi) \end{aligned}$$

primitive Rekursion und seien F, G k -Band Turingmaschinen zur Berechnung von f und G .

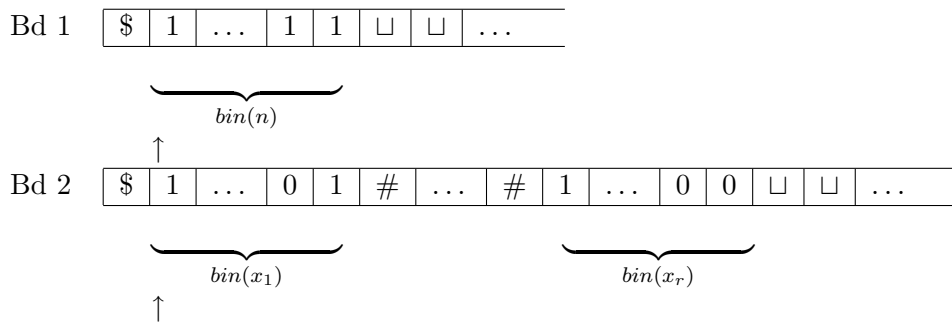
Dann gibt es eine $(k+4)$ -Band Turingmaschine H zur Berechnung von h .

3.18.1 Beweis:

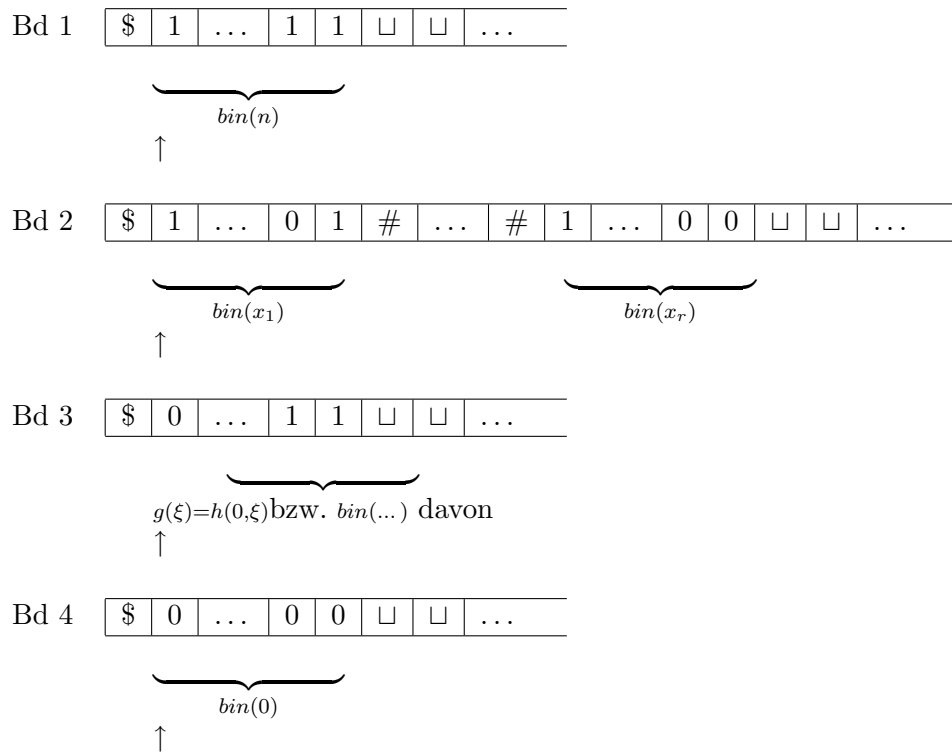
Beschreibung von H :



kopiere den hinter dem ersten # stehenden Inhalt vom Band 1 nach Band 2
 lösche diesen Inhalt auf Band 1



$n = 0 \left\{ \begin{array}{l} \text{Band 3} := \text{Band 2;} \\ \text{lasse } G \text{ auf den Bändern } 3, 4, \dots, k+2 \text{ laufen} \\ \text{Band 4} := 0; \end{array} \right.$



Algorithmus:

WHILE Band 1 \neq 0

DO

Band 5 := Band 4 # Band 3 # Band 2

lasse F auf den Bändern 5, 6, ... , k+4 laufen;

Band 3 := Band 5;

lösche Band 5;

Band 1 := Band 1 - 1;

Band 4 := Band 4 + 1;

OD

Band 1 := Band 3;

lösche Band 2, 3, 4;

Köpfe nach vorn;

3.19 Lemma:

(μ -Operator)

$$h = \mu f : \mathbb{N}_0 \rightarrow \mathbb{N}_0$$

Sei $\xi \mapsto \begin{cases} \text{das kleinste } n \text{ mit} \\ \cdot f(n, \xi) = 0 \text{ und} \\ \cdot f(i, \xi) \text{ definiert und } \neq 0 \text{ für } i = 0, 1, \dots, n-1 \\ [\text{undefiniert, sonst}] \end{cases}$

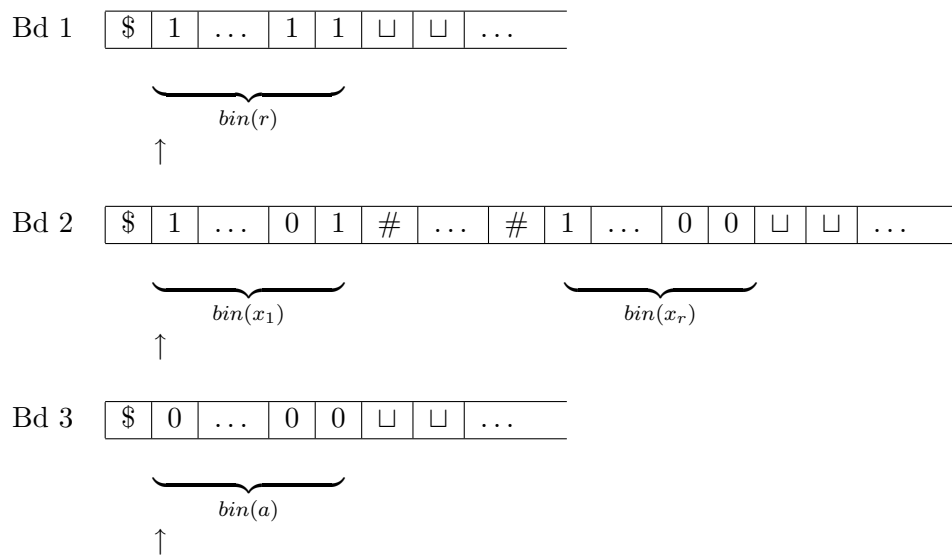
und sei F eine k -Band Turingmaschine zur Berechnung von f . Dann gibt es eine $(k+3)$ -Band Turingmaschine H zur Berechnung von h .

3.19.1 Beschreibung von H :

Band 2 := Band 1;

Band 1 := 1;

Band 3 := 0;



Algorithmus:

WHILE Band 1 \neq 0

DO

Band 4 := Band 3 # Band 2;

lasse auf Band 4, ..., $k+3$ laufen;

Band 1 := Band 4;

lösche Band 4;

Band 3 := Band 3 + 1;

OD

Band 1 := Band 3-1;

lösche Bänder; Köpfe nach vorn;

Damit: $\mathcal{F}_\mu \subseteq \mathcal{F}_{TM}$

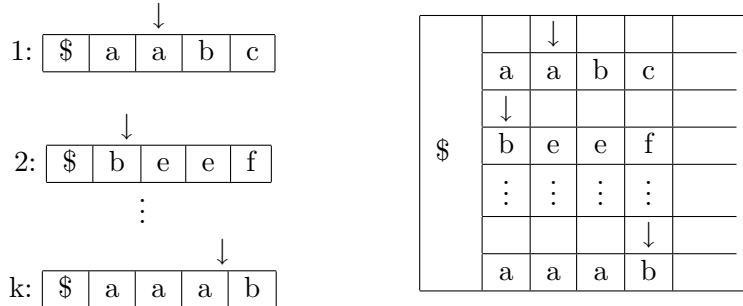
jetzt: $\mathcal{F}_\mu \supseteq \mathcal{F}_{TM}$

3.20 Lemma:

Sei $f : \mathbb{N}_0^r \rightarrow \mathbb{N}_0$ auf einer k -Band Turingmaschine F berechenbar. Dann lässt sich F auch auf einer 1-Band Turingmaschine F_1 berechnen.

3.20.1 Beweis:

Verwende ein Band mit $2k$ "Spuren":



Wie simuliert F_1 die Turingmaschine F ?

- suche nach den k Feldern mit \downarrow und speichere die Inschriften dieser Felder im Zustand
- Verwende Übergangsfunktion δ von F , um
 - neuen Zustand,
 - neue Bandinhalte,
 - Bewegungen der k Köpfe von F auszurechnen;
- besuche noch einmal alle Felder mit \downarrow ,
 - ändere Bandinschriften
 - verschiebe \downarrow

qed.

nächster Schritt Darstellung der Konfiguration einer 1-Band Turingmaschine in folgender Art:

$$K = \$a_1a_2 \dots a_{k-1}qa_ka_{k+1} \dots a_r$$

bedeute Maschine ist im Zustand q

Bandinhalt:

\$	a_1	a_2	\dots	a_{k-1}	a_k	a_{k+1}	\dots	a_r
----	-------	-------	---------	-----------	-------	-----------	---------	-------

K heißt Endkonfiguration $:\Leftrightarrow q \in F$ falls K keine Endkonfiguration besitzt und etwa

$$\delta(q, a_k) = (q', a'_k, +1)$$

$$\begin{array}{ccc} \$a_1 a_2 \dots a_{k-1} q a_k \dots a_r & \mapsto & \$a_1 a_2 \dots a_{k-1} a'_k q' a_{k+1} \dots a_r \\ K & \xrightarrow{\Delta} & K' \end{array}$$

Wir wollen jetzt

1. Konfigurationen (=Zeilenketten) durch Zahlen darstellen
2. Übergangsfunktionen Δ durch Funktion $\tilde{\Delta}$ auf Zahlen simulieren.
= Gödelisierung (Kurt Gödel)

Heute: $\mathcal{F}_{TM} \subseteq \mathcal{F}_\mu$ (“ \supseteq “ schon gezeigt.)

schon gesehen: Konfiguration $K = \$a_i \dots a_{k-1} q a_k \dots a_t$

drückt aus: Maschine in Zustand q , Bandinhalt:

\$	a_1	a_2	...	a_{k-1}	a_k	a_{k+1}	...	a_t
----	-------	-------	-----	-----------	-------	-----------	-----	-------

falls Turingmaschine in K durch Übergang $\delta(q, a_k) = (q', a'_k, +1)$ in Folgekonfiguration $K' = \$a_1 \dots a_{k-1} a'_k q' a_{k+1} \dots a_t$ übergehen kann, sagen wir: $K' = \Delta(K)$

jetzt:

1. Stelle Konfigurationen durch Zahlen dar.

zu (1): Sei $Q \cup \Sigma = \{b_1, b_2, \dots, b_q\}$

Definition:

$$\begin{aligned} \psi : (Q \cup \Sigma)^* &\rightarrow \mathbb{N}_0 \text{ bijektiv} \\ \varepsilon &\mapsto 0 \\ b_i &\mapsto i \end{aligned}$$

$$\text{für } v_1, \dots, v_s \in Q \cup \Sigma: v_1 \dots v_s \mapsto \sum_{j=1}^s \underbrace{\psi(v_j)}_{\in \{0,1,\dots,p\}} (p+1)^{s-j} \quad \begin{array}{l} \text{(p+1)-adische} \\ \text{Darstellung} \\ \text{natürlicher Zah-} \\ \text{len; eindeutig} \end{array}$$

Gödelisierung

2. “Simuliere“ Δ durch eine Funktion $\tilde{\Delta}$ auf Zahlen.

zu (2):

3.21 Lemma 1.8:

Zur Turingmaschine M gibt es Funktionen $\tilde{\Delta}$ und $END \in \mathcal{P}$ mit folgenden Eigenschaften:

$$\begin{aligned} \tilde{\Delta}(x) &= \begin{cases} \psi(K'), & \text{falls } x = \psi(K) \text{ und } K' \text{ die Fol-} \\ & \text{gekonfiguration von } K \text{ ist} \\ 0, & \text{sonst} \end{cases} \\ \text{END}(x) &= \begin{cases} 0, & \text{falls } x = \psi(K) \text{ f\"ur eine End-} \\ & \text{konfiguration } K \\ 1, & \text{sonst} \end{cases} \end{aligned}$$

Das heit, $\tilde{\Delta}$ macht folgendes Diagramm kommutativ:

$$\begin{array}{ccc} K & \xrightarrow{\Delta} & K' \\ \psi \downarrow & & \downarrow \psi \\ \psi(K) & \xrightarrow{\tilde{\Delta}} & \psi(K') \end{array}$$

Beweis spter

3.22 Theorem 1.6

$$\mathcal{F}_{TM} \subseteq \mathcal{F}_\mu \text{ und } \mathcal{F}_{TM}^{par} \subseteq \mathcal{F}_\mu^{par}.$$

3.22.1 Beweis:

Sei $f \in \mathcal{F}_{TM}$ eine r -stellige Funktion

\Rightarrow es gibt eine 1-Band Turingmaschine $M = (Q, \Sigma, \delta, q_0, F)$ mit M berechnet f .

Definiere

$$\begin{aligned} D(0, x) &:= x \\ D(n+1, x) &:= \tilde{\Delta}(D(n, x)) \quad \Rightarrow D \in \mathcal{P} \\ \text{und } D(n, \psi(k)) &= \begin{cases} \text{G\"odelnummer } \psi(K') \text{ derjenigen} \\ \text{Konfiguration } K', \text{ die entsteht,} \\ \text{wenn } M \text{ in } K \text{ startet und } n \text{ Re-} \\ \text{chenschritte macht (= \"Uberg\"ange)} \\ \text{falls das m\"oglich ist,} \\ 0 \text{ sonst} \end{cases} \end{aligned}$$

Definiere

$$\begin{aligned} g(n, x) &:= \text{END}(D(n, x)) \quad \Rightarrow g \in \mathcal{P} \\ A(x) &:= \mu g(x) = \text{kleinstes } n \text{ mit } g(n, x) = 0 \\ &= \text{kleinstes } n \text{ mit } D(n, x) \text{ ist Kodifikat} \\ & \quad (= \text{G\"odelnummer}) \text{ einer Endkonfigu-} \\ & \quad \text{ration.} \end{aligned}$$

f total \Rightarrow M erreicht immer eine Endkonfiguration
 $\Rightarrow A$ entsteht durch Anwendung des μ -Operators im Normalfall
 $\Rightarrow A \in \mathcal{F}_\mu$ (f partiell $\Rightarrow A \in \mathcal{F}_\mu^{par}$)
 und $A(\psi(K)) = \begin{cases} \text{Anzahl der Rechenschritte, die } M, \\ \text{gestartet in } K, \text{ macht, bis sie in} \\ \text{Endkonfiguration gerät, falls sie das} \\ \text{je tut} \\ \text{undefiniert, sonst} \end{cases}$

Angenommen M , angesetzt auf K , hält nach $A(\psi(K))$ Schritten in einer Endkonfiguration $K' = \$qbin(f(\xi))$ (nach Voraussetzung: M berechnet f).

Benötigen Funktion F , die aus $\psi(K')$ den Wert $f(\xi)$ extrahiert, müssen außerdem Startkonfiguration $q_0bin(x_1)\#bin(x_2)\#\dots\#bin(x_r)$ kodieren.

3.23 Lemma 1.9

Es gibt $E, F \in \mathcal{P}$ mit

$$E(x_1, \dots, x_r) = \psi(q_0bin(x_1)\#\dots\#bin(x_r))$$

$$F\left(\psi\left(\underbrace{\$qbin(x)}_{f(x)}\right)\right) = x$$

Beweis später!

3.24 Fortführung des Beweises von Theorem 1.6

$f(x_1, \dots, x_r)$ definiert $\Leftrightarrow M$, angesetzt auf $q_0bin(x_1)\#\dots\#bin(x_r)$, hält in Endzustand
 $\Leftrightarrow A(E(x_1, \dots, x_r))$ ist definiert und gibt Anzahl der Rechenschritte von M an.

in diesem Fall:

$$\begin{aligned}
 f(x_1, \dots, x_r) &= F(\psi(\text{Endkonfiguration})) \\
 &= F\left(D\left(\underbrace{A(E(x_1, \dots, x_r))}_{\text{Anzahl der Schritte}}, \underbrace{E(x_1, \dots, x_r)}_{\psi(K_0)}\right)\right) \\
 &\quad \underbrace{\hspace{10em}}_{\psi(K'), K' \text{ Endkonfiguration}}
 \end{aligned}$$

$\Rightarrow f \in \mathcal{F}_\mu$.

qed

3.25 Beschränkter μ -Operator:

Mit $f : \mathbb{N}_0^{r+1} \rightarrow \mathbb{N}_0$ ist auch

$$\mu_b f : \mathbb{N}_0^{r+1} \rightarrow \mathbb{N}_0$$

$$(\xi, y) \mapsto \begin{cases} \text{das kleinste } x \leq y \text{ mit} \\ f(\xi, x) = 0, \text{ falls es existiert} \\ 0 \text{ sonst} \end{cases}$$

primitiv rekursiv.

3.25.1 Beweis

:

$$\mu_b \cdot f(\xi, 0) = 0$$

$$\mu_b \cdot f(\xi, y + 1) = \begin{cases} y + 1, & \text{falls } f(\xi, y + 1) = 0 \\ & \text{und } \mu_b \cdot f(\xi, y) = 0 \text{ und} \\ & f(\xi, 0) \neq 0 \\ \mu_b \cdot f(\xi, y), & \text{sonst} \end{cases}$$

primitive Rekursion mit Fallunterscheidung

Bleibt der Beweis der Lemmata 1.8 und 1.9 Dazu.

3.26 Lemma 1.10

Es gibt primitiv rekursive Funktionen

L, CONCAT, PREFIX, SUFFIX, FIRST, LAST, SELECT,

so dass für alle $b, c \in (Q \cup \Sigma)^{ast}$ und alle i mit $0 \leq i \leq |b|$ gilt:

$$L(\psi(b)) = |b|$$

$$CONCAT(\psi(b), \psi(c)) = \psi(bc)$$

$$PREFIX(\psi(b), i) = \begin{cases} \psi(b_1, b_2, \dots, b_i), & \text{falls } i > 0 \\ 0, i = 0 & = \psi(\varepsilon) \end{cases}$$

$$b = b_1, b_2, \dots, b_i, \dots, b_{|b|}$$

$$SUFFIX(\psi(b), i) = \begin{cases} \psi(b_i b_{i+1}, \dots, b_{|b|}), & \text{falls } i > 0 \\ \psi(0), i = 0 \end{cases}$$

beachte $i \leq |b|$

$$FIRST(\psi(b)) = \psi(b_1)$$

$$LAST(\psi(b)) = \psi(b_{|b|})$$

$$SELECT(\psi(b), i) \begin{cases} \psi(b_i), & \text{falls } i > 0 \\ \psi(\varepsilon), & \text{sonst} \end{cases}$$

$$\text{Zur Erinnerung: } \psi(b_1, b_2, \dots, b_s) = \sum_{i=1}^s \psi(b_i) \cdot (p+1)^{S-i}$$

3.27 Beweis von Lemma 1.10:

$$L(x) := \min\{m, m \leq x \text{ und } (p+1)^m > x\}^{13}$$

beschränkter μ -Operator $\Rightarrow L \in \mathcal{P}$

$$CONCAT(x, y) := x \cdot (p+1)^{L(y)} + y$$

$$PREFIX(x, i) := x \cdot \text{div}(p+1)^{L(x)-i}, \text{ denn}$$

$$\psi(b) = \psi(b_1, \dots, b_{|b|}) = \psi(b_1) \cdot (p+1)^{|b|-1} + \underbrace{\psi(b_2) \cdot (p+1)^{|b|-2} + \dots + \psi(b_i) \cdot (p+1)^{|b|-i}}_{\text{niedrigere Terme}}$$

$$\Rightarrow PREFIX \in \mathcal{P}$$

$$SUFFIX(x, i) := \begin{cases} x \bmod (p+1)^{L(x)-i+1}, & \text{falls } i > 0 \\ 0, & i = 0 \end{cases}$$

$$\text{denn } \psi(b) = \psi(b_1, b_2, \dots, b_i, \dots, b_{|b|}) = \text{höchste Terme} + \underbrace{\psi(b_{i-1})(p+1)^{|b|-i+1} + \psi(b_i)(p+1)^{|b|-i} + \dots + \psi(b_{|b|})(p+1)^0}_{SUFFIX}$$

$$FIRST(x) := PREFIX(x, 1)$$

$$LAST(x) := SUFFIX(x, L(x))$$

$$SELECT(x, i) := \begin{cases} FIRST(SUFFIX(x, i)) & \text{falls } i > 0 \\ 0, & i = 0 \end{cases}$$

qed.

3.27.1 Abkürzungen:

$$X_{(i)} := SELECT(x, i)$$

$$[x, y] := CONCAT(x, y)$$

$$[x, y, z] := CONCAT(CONCAT(x, y), z)$$

¹³d.h. $(p+1)^{m-1}$ ist die größte in $x = \psi(b)$ vorkommende Potenz von $p+1$

3.27.2 Beweis von Lemma 1.8

Zu zeugen: es gibt $\tilde{\Delta}$, END mit:

$$\begin{array}{ccc}
 K & \xrightarrow{\Delta} & K' \\
 \psi \downarrow & & \downarrow \psi \\
 \psi(K) & \xrightarrow{\tilde{\Delta}} & \psi(K')
 \end{array}$$

kommutativ.

$$\text{END}(\psi(K)) = \begin{cases} 0, & \text{falls } K \text{ Endkonfiguration} \\ 1, & \text{sonst} \end{cases}$$

3.28 Beweis:

$\psi_Q := \{\psi(q), q \in Q\}$ endlich $\Rightarrow \chi_{\psi_Q} \in \mathcal{P}$ (endliche Fallunterscheidung)¹⁴

$q(x) := \min\{i \leq L(x) \mid x_i \in \psi_Q\}$

für $x = \psi(K)$ mit $K = \$a_1 \dots a_{k-1} q a_k \dots a_r$

ist $q(x) = k + 1$ = Position von "q" in K
 = Position des Schreib-/Lesekopfes von M .

3.28.1 Definition:

$u(x) := \text{PREFIX}(x, q(x) - 2) \Rightarrow u(\psi(K)) = \psi(\$a_1 \dots a_{k-2})$

$v(x) := \text{SUFFIX}(x, q(x) + 2) \Rightarrow v(\psi(K)) = \psi(\$a_{k+1} \dots a_1)$

$w(x) := [x_{q(x)-1}, x_{q(x)+1}]$, $w(\psi(K)) = \psi(\underbrace{a_{k-1} q a_k}_{\text{Action hier!}})$

3.29 Beweis von Lemma 1.8

$\psi(v_1, \dots, v_s) := \sum_{i=1}^2 \psi(v_i)(p+1)^{s-i}$ wobei $\Sigma \cup Q = \{b_1, b_2, \dots, b_p\}$

$v_i \in \Sigma \cup Q, v_i = b_j : \psi(v_i) = j, \psi(\epsilon) = 0$

Hatten Funktionen $u, v, w \in \mathcal{P}$ definiert mit

$K = \$a_1 \dots$	$\underbrace{a_{k-1} q a_k}_{\text{Fenster auf die interessante Stelle}}$	$\dots a_k:$	$u(\psi(K)) = \psi(\$a_1 \dots a_{k-2})$
			$w(\psi(K)) = \psi(a_{k-1} q a_k)$
			$v(\psi(K)) = \psi(a_{k+1} \dots a_t)$ Entscheidung

über nächsten Rechenschritt erfolgt lokal:

¹⁴ χ_{ψ_Q} = charakteristische Funktion, d.h. $\chi_{\psi_Q}(x) = 1 \Leftrightarrow x = \psi(q)$ für einen Zustand q .

3.29.1 Definition:

Durch endliche, primitive rekursive Fallunterscheidung $\Rightarrow \tilde{\delta} \in \mathcal{P}$

$$\tilde{\delta}(x) := \begin{cases} \psi(a_{k-1}a'_kq'), & \text{falls } x = \psi(a_{k-1}qa_k) \text{ und } \delta(q, a_k) = (q', a'_k, +1) \\ \psi(a_{k-1}q'a'_k), & \text{falls } x = \psi(a_{k-1}qa_k) \text{ und } \delta(q, a_k) = (q', a'_k, 0) \\ \psi(q'a_{k-1}a'_k), & \text{falls } x = \psi(a_{k-1}qa_k) \text{ und } \delta(q, a_k) = (q', a'_k, -1) \end{cases}$$

Damit :

$\tilde{\Delta}(x) := [u(x), \tilde{\delta}(w(x)), v(x)] \Rightarrow \tilde{\Delta} \in \mathcal{P}$ und leistet das Verlangte!

$$END(x) = \begin{cases} 0, & \text{falls } X_{(q(x))} \in \{\psi(e); e \in F \text{ (Endzustände von M)}\} \\ 1 & \text{sonst} \end{cases}$$

$\Rightarrow END \in \mathcal{P}$

3.30 Beweis von Lemma 1.9

Zu zeigen: es gibt $E, F \in \mathcal{P}$ mit

a) $E(X_1, \dots, X_r) = \psi(\$q_0 \text{bin}(x_1) \# \text{bin}(X_2) \# \dots \# \text{bin}(X_r))$

b) $F(\psi(\$q \text{bin}(y))) = y$

zu a) Sei $x = w_1 2^{s-1} + w_2 2^{s-2} + \dots + w_{s-i+1} 2^{i-1} + w_{s-1+2} 2^{i-2} + \dots + w_{s-1} 2^1 + w_s$ die Binärdarstellung von x. Dann: $w_{s-i+1} = (x \text{ mod } 2^i) \text{ div } 2^{i-1} =: B(i, x)$

$$\Rightarrow B \in \mathcal{P} \text{ und } P(n, x) := \sum_{i=1}^n \psi(B(i, x)) \cdot (p+1)^{i-1}$$

$$\Rightarrow P \in \mathcal{P} \text{ und } \Rightarrow P \in P \text{ und } P(L(x), x) = \psi(\text{bin}(x))$$

$L(x)$ = Länge von x in (p+1)-adischen Darstellung

Damit:

$$E(x_1, \dots, x_r) := [\psi(\$), \psi(q_0), P(L(x_1), x_1), \psi(\#), \dots, \psi(\#), P(L(x_r), x_r)]$$

$\Rightarrow E \in \mathcal{P}$ und leistet, was es soll.

zu b) zunächst: $G(x) := \text{SUFFIX}(x, q(x) + 1)^{15}$

$\Rightarrow G \in \mathcal{P}$

und $G(\psi(\$q \text{bin}(y))) = \psi(\text{bin}(y))$

ohne Einschränkung sei $\psi(0) = 1$ und $\psi(1) = 2$

$$\text{Sei } H(x) := \sum_{i=1}^{L(x)} \left(\underbrace{(x \text{ mod } (p+1)^i) \text{ div } (p+1)^{i-1}}_{\text{Koeffizient bei } (p+1)^{i-1}} - 1 \right) \cdot 2^i$$

$\Rightarrow H(\psi(\text{bin}(y))) = y$ und $H \in \mathcal{P}$ Setze $F(x) := H(G(x)) \Rightarrow F \in \mathcal{P}$ und

¹⁵ $q(x)$ = "die Stelle vom Zustandin x

$$F(\psi(\$qbin(y))) = H(bin(y)) = y$$

damit fertig:

$$\mathcal{F}_\mu = \mathcal{F}_{TM}, \quad \mathcal{F}_\mu^{par} = \mathcal{F}_{TM}^{par}$$

Frage: Was hat man davon?

3.31 Folgerungen

1. Ackermann-Funktion

$$\begin{aligned} A(0, y) &= y + 1 \\ A(x + 1, 0) &= A(x, 1) \\ A(x + 1, y + 1) &= A(x, A(x + 1, y)) \end{aligned}$$

wissen: $A \notin P$

zeigen jetzt: $A \in \mathcal{F}_\mu$

genügt zu zeigen: $A \in \mathcal{F}_{TM}$, d.h. wir brauchen nur eine TM zu bauen, die A berechnet

$$\text{Darstellung: } A(x, y) \bar{\wedge} \dots \underbrace{\|\dots\|}_x A \underbrace{\|\dots\|}_y \dots$$

$$\text{Vorbereitung: } \$bin(x)\#bin(y) \sqcup \dots \Rightarrow \$ \underbrace{\|\dots\|}_x A \underbrace{\|\dots\|}_y \sqcup \dots$$

solange ein A auf dem Band steht, führt M folgende "Makros" aus:
(abhängig von der Umgebung des am weitesten rechts stehenden A)

i links vom rechten A steht \$:

$$\begin{aligned} & \$ \underbrace{A \|\dots\|}_y \sqcup \\ & \underbrace{\hspace{1.5cm}}_{A(0,y)} \\ & \Rightarrow \$ \underbrace{\|\dots\|}_{y+1} \sqcup \end{aligned}$$

links vom rechten A steht A:

$$\begin{aligned} & \dots A \underbrace{A \|\dots\|}_y \sqcup \\ & \Rightarrow A \underbrace{\|\dots\|}_{y+1} \sqcup \end{aligned}$$

ii links vom rechten A steht |, rechts \sqcup :

$$\dots | A \sqcup \dots$$

$$\dots A | \sqcup \dots$$

iii links und rechts vom rechten A steht ein |:
mit $L \in \{A, \$\}$:

$$\begin{aligned} & \dots L \underbrace{|| \dots ||}_{x+1} A \underbrace{| \dots ||}_{y+1} \sqcup \\ \Rightarrow & \dots L \underbrace{| \dots |}_x A \underbrace{|| \dots ||}_{x+1} A \underbrace{| \dots |}_y \sqcup \end{aligned}$$

geht!

Zum Schluß: Rückverwandlung:

$$\underbrace{\$ | \dots |}_{A(x,y)} \sqcup \Rightarrow \$ \text{bin} A(x, y) \sqcup$$

3.31.1 Beispiel:

$$\begin{aligned} & A(1, 2) = ? \\ & \downarrow \\ & \$ | A | \sqcup \\ & \downarrow \text{(iii)} \\ & \$ A | A | \sqcup \\ & \downarrow \text{(iii)} \\ & \$ A A | A \sqcup \\ & \downarrow \text{(ii)} \\ & \$ A A A | \sqcup \\ & \downarrow \text{(i)} \\ & \$ A A | \sqcup \\ & \downarrow \text{(i)} \\ & \text{dollar} A | \sqcup \\ & \downarrow \text{(i)} \\ & \text{dollar} | \sqcup \\ & \downarrow \\ & A(1, 2) = 4 \end{aligned}$$

3.32 Definition

Seien f, g r -stellige Funktionen in \mathcal{F}_μ^{par} .
 $f \cong g \Leftrightarrow D(f) = D(g)$ und $\forall \xi \in D(f) : f(\xi) = g(\xi)$.

2. Kleene'sches Normalformtheorem: (schwache Version) Zu jedem $f \in \mathcal{F}_\mu^{(par)}$ gibt es $p, q \in \mathcal{P}$, so dass $\forall \xi : f(\xi) \cong g(\mu p(\xi), \xi)$

Beweis:

Wir hatten beim Beweis von $\mathcal{F}_{TM}^{(par)} \subseteq \mathcal{F}_\mu^{(par)}$ gezeigt:

es gibt Funktionen F, D, A, E mit $F, D, E \in \mathcal{P}$, $A(x) = \mu g(x)$ mit $g \in \mathcal{P}$ und $\forall \xi : f(\xi) = F(D(A(E(e), E(e))))$

Beachte: F, D, E, A hängen von f ab.

Folgerung hieraus:

- Jedes $f \in \mathcal{F}_\mu^{(par)}$ kann durch einmalige Anwendung des μ -Operators auf primitive rekursive Funktionen erzeugt werden.
 \Rightarrow beim Programmieren genügt im Prinzip eine einzige WHILE-Schleife
 - $f \in \mathcal{F}_\mu^{(par)}$ und $D(f) = \mathbb{N}_0^r$ (d.h. f total) $\Rightarrow f \in \mathcal{F}_\mu$
3. Stütze für These von Church (" \mathcal{F}_μ sind die berechenbaren Funktionen ")

3.33 starke Version von Kleene's Normalformtheorem

(statt - etwa bei $\tilde{\delta}$ in $\tilde{\Delta}$ - die Turingmaschine M "fest einzubauen", kann man auch ihre Gödelnummer $k = \ulcorner M \urcorner$ als Parameter übergeben)

Damit:

$\exists U, T, \varphi \in \mathcal{P} : \forall f \in \mathcal{F}_\mu^{(par)} \exists k \forall \xi : f(\xi) \cong U(\mu T(k, \varphi, \xi)) \cong [k](\xi)$

Also: Abhängigkeit von f steckt nur in $k = \ulcorner M \urcorner$.

Man kann eine Turingmaschine M bauen, welche die Funktion $(h, \xi) \mapsto U(\mu T(k, \varphi(\xi)))$ berechnet

$\Rightarrow M$ ist eine universelle Turingmaschine, $\hat{=}$ Interpreter für $k = \ulcorner M \urcorner$

Damit: Programme sind Daten: M^* bekommt als Input k und ξ und berechnet $[k](\xi)$ (v. Neumann-Rechner)

3.34 Entscheidbarkeit

Σ endliches Alphabet.

Sprache $L \subseteq \Sigma^*$ heißt "entscheidbar" : $\Leftrightarrow \chi_L \in \mathcal{F}_\mu$.

Entscheidbarkeit bedenkt: es gibt ein effektives Verfahren, um zu entscheiden, ob ein beliebiges $w \in \Sigma^*$ zu L gehört.

Wir können eine Turingmaschine M bauen, die χ_L berechnet, und sie auf w ansetzen.

$\Rightarrow M$ muss nach endlich vielen Schritten mit Ausgabe $\begin{cases} 1, & \text{falls } w \in L \\ 0 & \text{falls } w \notin L \end{cases}$

anhalten.

Redeweise:

“M akzeptiert w “ \Leftrightarrow M stoppt mit Ausgabe 1
 $\Leftrightarrow \chi_L(w) = 1$
 $\Leftrightarrow w \in L$

man könnte auch \mathcal{F}_{TM} für Funktionen auf Σ^* betrachten.

Klar: $f : \Sigma^* \rightarrow \mathbb{N}_0 \in \mathcal{F}_{TM}$

$\Leftrightarrow \mathbb{N}_0 \xrightarrow{\pi^{-1}} \Sigma^* \xrightarrow{f} \mathbb{N}_0$ in \mathcal{F}_μ , wobei π irgendeine vernünftige Gödelisierung von Σ^*

Nicht-Entscheidbarkeit Hatten gesehen: es gibt nicht-berechenbare Funktionen $f : \mathbb{N}_0 \rightarrow \{0, 1\}$
 \Rightarrow es gibt nicht-entscheidbare Mengen, z.B. $\{x \in \mathbb{N}_0 : f(x) = 1\}$

Problem: Wir kennen keine nicht-entscheidbare Menge.

3.34.1 Gödelnummer

Zunächst: Kodierung einer Turingmaschine $= (Q, \Sigma, \delta, q_0, F)$ durch Strings, d.h. Wörter über $\{0, 1, \#\}$

Sei $Q = \{q_0, q_1, \dots, q_{p-1}, F = \{q_{p-|F|}, \dots, q_{p-1}, \Sigma = \{a_p, a_{p+1}, \dots, a_r\}$

Sei etwa $\delta(q_i, a_j) = (q'_i, a'_j, \underbrace{m}_{\in \{0, -1, 1\}}) : \sigma = \#\#bin(i)\#bin(j)\#bin(i')\#bin(j')\#bin(c)$

mit $c = m + 1$

$$\langle M \rangle := \underbrace{\#\#\#\#bin(0)\#\dots\#bin(p-1)\#\#\#}_{Q} \underbrace{bin(p)\#\dots\#bin(r)}_{\Sigma}$$

$$\#\#\#bin(p-|F|)\#\dots\#bin(p-1)\#\#\#\sigma_1\#\sigma_2\#\dots\#\sigma_k\#\#\#\# \in \{0, 1, \#\}^*$$

(Durch Anwendung von ψ könnte man aus $\langle M \rangle$ eine echte Gödelnummer machen)

Kodierung möglicher Eingaben $w = a_{i_1}a_{i_2}\dots a_{i_n} \in \Sigma^*$:

$$\langle w \rangle := bin(i_1)\#bin(i_2)\#\dots\#bin(i_n)$$

Theorem $H_l := \{x \in \{0, 1, \#\}^* : \text{es gibt Turingmaschine } M \text{ mit } x = \langle M \rangle \text{ und } M \text{ hält bei Eingabe von } \langle M \rangle\}$ ist unentscheidbar. (“Halten bei Eingabe der eigenen Kodierung/Gödelnummer”)

Beweis: Angenommen H_l wäre entscheidbar.

\Rightarrow es existiert Turingmaschine M mit M berechnet χ_{H_l} .

\Rightarrow es gibt Turingmaschine M' zu M mit:

M berechnet 0 bei Eingabe $x \Rightarrow M'$ hält bei Eingabe x

M berechnet 1 bei Eingabe $x \Rightarrow M'$ hält bei Eingabe x nicht!

Betrachte was M' bei Eingabe von $\langle M' \rangle$ macht:

M' hält bei Eingabe von $\langle M' \rangle$ $\stackrel{\Leftrightarrow}{\text{Def. } M'} M$ berechnet 0 bei Eingabe von $\underbrace{\langle M' \rangle}_x$

$\stackrel{\Leftrightarrow}{M\text{ber. } \chi_{H_l}} \langle M' \rangle \neg \in H_l$

$\stackrel{\Leftrightarrow}{\text{Def. } H_l} M'$ angesetzt auf $\langle M' \rangle$ hält nicht. Widerspruch

\Rightarrow (Diagonalschluß) H_l ist nicht entscheidbar.

3.34.2 Korollar (Unentscheidbarkeit des allgemeinen Halteproblems)

Sei $H : \{x \in \{0, 1, \#\}^*; x = \langle M \rangle \langle w \rangle \text{ für eine Turingmaschine } M \text{ und Input } w \text{ und } M \text{ hält bei Eingabe von } w.\}$

$\Rightarrow H$ ist unentscheidbar

Beweis: Andernfalls wäre auch H_l entscheidbar, denn für $x = \langle M \rangle$ gilt:

$x \in H_l \Leftrightarrow M$ angesetzt auf x hält $\Leftrightarrow x' := \langle M \rangle \langle \langle M \rangle \rangle \in H$

Also: Semantische Verifikation von beliebigen Programmen ist prinzipiell unmöglich.

3.35 Weitere Beispiele unentscheidbarer Probleme

3.35.1 Wortproblem

Grammatik $G = (T, N, S, R)^{16}$ mit $T = \{a, b, c, \dots\}$, $N = \{S, A, B, \dots\}$ und Übergangsregeln R der Art $\alpha \rightarrow \beta$ mit $\alpha, \beta \in (T \cup N)^*$

\Rightarrow transitiver Abschluß von $\rightarrow L(G) := \{w \in T^* \text{ mit } S \Rightarrow w\}$.

Beispiel: $T = \{a, b\}$, $N = \{S\}$, $R : \{S \rightarrow aSb, S \rightarrow \varepsilon\}$

$L(G) = \{a^n b^n, n \geq 0\}$

$S \rightarrow \varepsilon : \varepsilon$

$S \rightarrow aSb \rightarrow aaSbb \rightarrow aaaSbbb \rightarrow a^3b^3$

¹⁶Terminalsymbole, Nichtterminalsymbole, Startzeichen, Übergangsregeln; alles endlich

Aber: für beliebiges G ist das Wortproblem:

Gegeben $w \in \Sigma^*$. Frage $w \in L(G)$?

nicht entscheidbar.

Erfreulicherweise verwendet man bei Programmiersprachen spezielle G mit entscheidbarem Wortproblem.

speziell: Post'sches Korrespondenzproblem

Gegeben: zwei Folgen v_1, v_2, \dots, v_m und w_1, w_2, \dots, w_m von Wörtern in Σ^*

Frage: Gibt es Folge von Indizes i_1, i_2, \dots, i_n mit

$$v_{i_1} v_{i_2} \dots v_{i_n} = w_{i_1} w_{i_2} \dots w_{i_n}$$

unentscheidbar

3.35.2 Game of Life

in $\mathbb{Z} \times \mathbb{Z}$

3.35.3 10-tes Hilbert-Problem

Sei $p(x_1, \dots, x_n)$ ein Polynom mit Koeffizienten in \mathbb{Z} , also z.B. $13x_1^3x_2 - 27x_2^2x_3^4 + 17$.

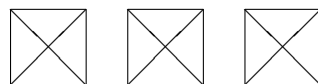
Gibt es $x_1, \dots, x_n \in \mathbb{Z}$ mit $p(x_1, \dots, x_n) = 0$?

unentscheidbar

(die Frage nach $x_1, \dots, x_n \in \mathbb{R}$ mit $p(x_1, \dots, x_n) = 0$ ist entscheidbar)

3.35.4 Domino-Problem

Gegeben sind endlich viele Typen von Steinen folgender Art:



Frage: Kann man die Ebene mit Translaten dieser Steine so pflastern, dass benachbarte Farben gleich sind?

3.36 Definition (rekursiv/entscheidbar)

- $P \subseteq \mathbb{N}_0$ heißt rekursiv $:\Leftrightarrow P$ ist entscheidbar
- $P \subseteq \mathbb{N}_0$ heißt rekursiv aufzählbar $:\Leftrightarrow P \neq \emptyset$ oder es existiert $f \in \mathcal{F}_\mu$ mit $P = \{f(x); x \in \mathbb{N}_0\}$

3.36.1 Lemma

P rekursiv aufzählbar \Leftrightarrow die Funktion c_P mit $c_P = \begin{cases} 1, & \text{falls } y \in P \\ \text{undefiniert} & \text{sonst} \end{cases}$ in \mathcal{F}_μ^{par} liegt.

Beweis: Sei $P = \{f(x); x \in \mathbb{N}_0 \text{ mit } f \in \mathcal{F}_\mu\}$

$$c_P = 1 - \underbrace{|f(\mu x \cdot f(x) = y) - y|}_{\Leftrightarrow Def_{y \in P}} \Rightarrow c_P \in \mathcal{F}_\mu^{par}$$

Umgekehrt: Sei $c_P \in \mathcal{F}_\mu^{par}$ Kleene \Rightarrow es gibt $q, s \in \mathcal{P}$ mit $c_P(y) \cong q(\mu s(y), y)$

Also: (*) $y \in P \Leftrightarrow c_P(y)$ ist definiert \Leftrightarrow es gibt z mit $s(y, z) = 0$

Trick: zu $K(x, y) = 2^x 3^y$ gibt es $d_1, d_2, K \in \mathcal{P}$ mit $d_1(K(x, y)) = x, d_2(K(x, y)) = y$

Sei $p \in P$ fest gewählt.

Definition: $f(x) := \begin{cases} d_1(x), & \text{falls } s(d_1(x), d_2(x)) = 0 \\ p, & \text{sonst} \end{cases} \Rightarrow f \in \mathcal{P}$, und es

gilt $P = \{f(x); x \in \mathbb{N}_0\}$

“ \supseteq “ $p \in P$: Falls $s(d_1(x), d_2(x)) = 0 \stackrel{(*)}{\Rightarrow} P \ni d_1(x) = f(x)$

“ \subseteq “ $y \in P \stackrel{(*)}{\Rightarrow}$ es gibt z mit $s(y, z) = 0$. Setze $x := K(y, z)$, dann folgt: $s(d_1(x), d_2(x)) = 0$

Def. $f(x) = d_1(x) = y$

qed.

3.36.2 Theorem

$\{P \subseteq \mathbb{N}_0, P \text{ rekursiv}\} \subsetneq \{P \subseteq \mathbb{N}_0, P \text{ rekursiv aufzählbar}\}$

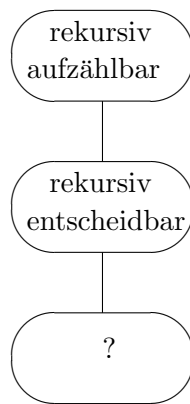
Beweis

“ \subset “ P rekursiv $\Rightarrow \chi_P \in \mathcal{F}_\mu \Rightarrow c_P(y) := \begin{cases} 1, & \chi_P(y) = 1 \\ \text{undefiniert} & \text{sonst} \end{cases} \in \mathcal{F}_\mu^{par} \stackrel{\Rightarrow}{\text{Lemma}} P \text{ rekursiv aufzählbar.}$

warum " \subsetneq " Betrachte $\tilde{H} := \{y \in \mathbb{N}_0, y = \psi(\langle M \rangle \langle w \rangle)\}$ mit Turingmaschine M , Eingabe w und M hält bei Eingabewert w
 unentscheidbar, also nicht rekursiv. Aber rekursiv aufzählbar, $c_{\tilde{H}} := 1 - (1 - (\text{Anzahl der Rechenschritte nach denen } M \text{ angesetzt auf } w \text{ hält})) \in \mathcal{F}_{\mu}^{par}$
 analog zu unserem Beweis $\mathcal{F}_{TM} \subsetneq \mathcal{F}_{\mu}$.

3.36.3 Weltbild

geometrisch arithmetisches Weltbild



3.36.4 Lemma

Sei P und das Komplement P^C rekursiv aufzählbar. Dann ist P rekursiv.

Beweis :

P rekursiv aufzählbar \Rightarrow es existiert $f \in \mathcal{F}_{\mu} : P = \{f(x); x \in \mathbb{N}_0\}$

P^C rekursiv aufzählbar \Rightarrow es existiert $g \in \mathcal{F}_{\mu} : P^C = \{g(x); x \in \mathbb{N}_0\}$

setze $h(x) := \mu x(f(x) = y \text{ oder } g(x) = y)$

$\Rightarrow \mu$ -Operator im Normalfall

$\Rightarrow h \in \mathcal{F}_{\mu} \Rightarrow \chi_P \in \mathcal{F}_{\mu}$.

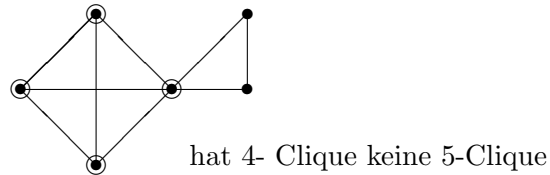
3.37 Was haben folgende Probleme gemeinsam?

3.37.1 k-Clique

Gegeben Graph $G(V, E)$, Zahl $k \in \mathbb{N}$.

Frage: Gibt es in G k Knoten, die paarweise direkt verbunden sind? "k-Clique"

z.B.



3.37.2 Satisfiability (SAT) - Erfüllbarkeit

Gegeben aussagenlogische Formel α in konjunktiver Normalform, z. B.

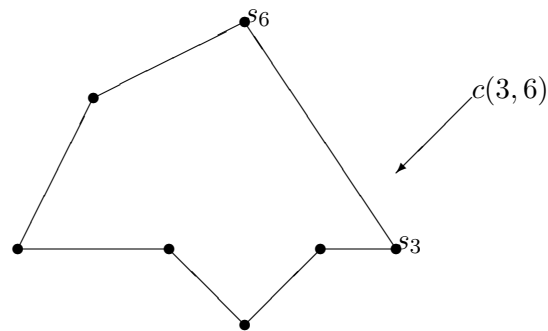
$$\alpha \equiv (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$$

Frage: Gibt es eine Belegung der bool'schen Variablen x_i , für die α den Wert 1 bekommt? hier: $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$

3.37.3 Traveling Salesman Problem (TSP)

Gegeben n Städte s_1, s_2, \dots, s_n und für je zwei die Kosten $c(i, j)$ der direkten Reise von s_i nach s_j . Ferner k .

Frage: Gibt es eine Rundreise, die jede Stadt besucht und Gesamtkosten $\leq k$ hat?



3.37.4 Gemeinsamkeiten

- wichtig
- gleich schwierig (!)¹⁷
- niemand weiß wie schwierig
- wenn ein Lösungskandidat gegeben ist, kann man leicht feststellen, ob er die gewünschte Eigenschaft hat.

¹⁷Kommentar: es ist relativ schwierig zu sehen, dass alle gleich schwierig sind

Deshalb: "Lösungssatz "

- * Rate einen Lösungskandidaten (Keine Wahrscheinlichkeiten)
- * Stelle fest, ob er das Problem löst

3.38 nichtdeterministische Turingmaschinen

$M = (Q, \Sigma, q_0, \delta, F)$ mit $\delta : Q \times \Sigma^k \rightarrow \underbrace{\mathfrak{R}}_{\text{Potensmenge}} (Q \times \Sigma^k \times \{-1, 0, 1\}^k)$

Bedeutung im Zustand q ist bei Bandinschriften s_1, s_2, \dots, s_k jeder Übergang $(q', s'_1, \dots, s'_k, m_1, \dots, m_l)$ aus der Menge $\delta(q, s_1, \dots, s_k)$ möglich.

3.38.1 Definition

Σ endliches Alphabet, $w \in \Sigma^*$, M nichtdeterministische Turingmaschine mit $F = \{q_-, q_+\}$

M akzeptiert $w : \Leftrightarrow$ es gibt einen Rechengang von M , angesetzt auf w , der in q_+ endet.

q_+ : akzeptierender Zustand von M .

Sei $L \subseteq \Sigma^*$

M akzeptiert $L : \Leftrightarrow L = \{w \in \Sigma^*; M \text{ akzeptiert } w\} = L(M)$

Frage Was ändert sich durch Einführung nichtdeterministischer Turingmaschinen?

"Ganz oben: nichts"

$\{L; \text{ es gibt nichtdet. } M' \text{ mit } M' \text{ akzeptiert } L\} \stackrel{?}{\equiv} \{L; \text{ es gibt det. } M \text{ mit } M \text{ akzeptiert } L\}$
 $= \{L \text{ rekursiv}\}$

" \subset " gilt, weil man eine deterministische M bauen kann, welche alle möglichen Rechengänge einer nichtdeterministischen M' nach und nach ausführt.
 ABER DAS KOSTET ZEIT !

\Leftrightarrow wollen jetzt auch Rechenzeit betrachten

- als Funktion der Länge der Eingabe, (w)
- asymptotisch, z. B. $13 \cdot n^3 - 2 \cdot n^2 + 28 \in \Theta(n^3)$

3.38.2 Rechenzeit

$f, g : \mathbb{N}_0 \rightarrow \mathbb{R}_{\geq 0}$

$O(f) = \{g, \text{ ex. } C > 0, n_0 \text{ mit } \forall n \geq n_0 :$

$f(n) \leq C \cdot g(n)\}$

$g \in \Omega(f) \Leftrightarrow f \in O(g)$

$\Theta(f) = O(f) \cap \Omega(f)$

det. Turingmaschinen $M = (Q, \Sigma, \delta, q_0, F)$ und $p(x)$ Polynom
 M akzeptiert L in polynomieller Zeit $p(x) :\Leftrightarrow \forall w \in \Sigma^*$
 M angesetzt auf w , hält nach endlich vielen $p(|w|)$ Schritten,

im Zustand q_+ , falls $w \in L$

“ Akzeptor“

im Zustand q_- , falls $w \notin L$.

$\mathcal{P} := \{L; \text{es gibt det. TM } M, \text{ Polynom } p(x) \text{ mit: } M \text{ akzeptiert } L \text{ in polynomieller Zeit } p(x)\}$

Beispiele: $L = \{bin(n); n \text{ ist durch } 2 \text{ teilbar}\}$ in Zeit $|w| = \log n$

$L = \{(bin(a), bin(b), bin(c)), c = a \cdot b\}$ in Zeit $\log a \cdot \log b$

$[L = \{bin(p); p \text{ Primzahl}\}$ (!) nicht Gegenstand der Vorlesung, das Problem wurde erst vor ein paar

Modellierung von “ Raten “ M nichtdet. TM, $p(x)$ Polynom

M akzeptiert w in polynomieller Zeit $p(x) :\Leftrightarrow$ es gibt einen Rechengang von M , angesetzt auf w , der nach höchstens $p(|w|)$ Schritten in Zustand q_+ endet.

3.39 Definition NP (nichtdeterministisch polynomiell)

$NP := \{L; \text{es gibt nichtdet. TM } M \text{ und Polynom } p(x) \text{ mit}$
 $M \text{ akzeptiert } L \text{ in polynomieller Zeit } p(x)\}$

\Updownarrow

$L = \{w, M \text{ akzeptiert } w \text{ in Zeit } p(w)\}$

Klar: $P \subset NP$

Offen: $P \stackrel{?}{=} NP$, man vermutet $P \subsetneq NP$

3.39.1 Beispiele:

Clique

Eingabe: Graph $G = (V, E)$, Zahl $k \in \mathbb{N}_0$

¹⁷(eine Ausgabe ist nicht nötig, der Zustand reicht als Ergebnis aus)

Behauptung: Clique $\in NP$

Input: $bin(|V|)\# \underbrace{\text{Liste der Kanten}}_{\text{Adjazenzlisten}} \# bin(k)$ über $\Sigma = \{0, 1, \#, (,)\}$
 $:= w(G, k)$

Clique = $\{x \in \Sigma^*; \text{es gibt Graphen } G \text{ und eine Zahl } k \text{ mit } x = w(G, k) \text{ und } G \text{ hat eine } k\text{-Clique}\}$ Clique $\in NP$: baue eine nichtdet. TM M , die

- teste, ob $k \leq |v|$, sonst gehe nach q_{-}
- eine Teilmenge $v' \subseteq V$ der Größe k rät
- prüft, ob je zwei Knoten von v' direkt durch Kante verbunden sind

weil $k \leq |v|$:

Laufzeit von $M \leq C \cdot k^2$ polynomiell in der Länge der Adjazenzlisten

5-Clique $\in P$

Denn es gibt nur $\binom{n}{5}$ mögliche Teilmengen v' von v , die aus 5 Elementen bestehen. (falls $|v| = n$)

$\binom{n}{5} = \frac{n!}{5!(n-5)!} = \frac{1}{5!}(n-4)(n-3) \dots n \leq n^5$ Polynomiell in Länge der Eingabe

Argument gilt für variables k nicht! Denn z. B. $k = \frac{n}{2} : \binom{n}{\frac{n}{2}} = \frac{n!}{(\frac{n}{2})!^2} \in \Theta(\frac{2^n}{\sqrt{n}})$ exponentiell in n

Ebenso: SAT $\in NP$

- rate Belegung der x_i mit 0 oder 1 (2^n Möglichkeiten)
- teste, ob α Wert 1 bekommt (einfach in polynomieller Zeit)

und Traveling Salesman $\in NP$

- rate eine Reihenfolge der n Städte ($n!$ Möglichkeiten)
- teste, ob bei dieser Reihenfolge Gesamtkosten $\leq k$ (einfach in polynomieller Zeit)

Wie geht man das P/NP - Problem an?

- versuche, in NP Probleme maximaler Schwierigkeit zu identifizieren (sehr erfolgreich)
 - versuche zu beweisen, dass ein solches Problem in P liegt. (erfolglos)
- ($\Rightarrow P = NP$)

Was sind Probleme maximaler Schwierigkeit in NP?

3.40 Definition (polynomielle Reduktion)

Seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Gamma^*$ zwei Sprachen.

Wir sagen, L_1 lässt sich polynomiell auf L_2 reduzieren $:\Leftrightarrow$ es gibt $f : \Sigma^* \rightarrow \Gamma^*$ und Polynom $p(x)$, so dass für alle $w \in \Sigma^*$

- i $f(w)$ ist (von einer det. TM) in Zeit $p(|w|)$ aus w berechenbar item[ii]
 $f(w) \in L_2 \Leftrightarrow w \in L_1$

Klar: Wenn man L_2 schnell lösen könnte, so auch L_1
 polynomiell

$\Rightarrow L_2$ ist mindestens so schwierig wie L_1 : Abkürzung $L_1 \leq_{pol} L_2$

3.41 Definition (NP-hart/vollständig)

$L \subseteq \Sigma^*$ heißt

- NP-hart, falls gilt: $\forall L' \in NP : L \leq_{pol} L'$
- NP-vollständig, falls L NP-hart und $L \in NP$

NP vollständige Probleme sind Probleme maximaler Schwierigkeit in NP.

Frage Wie findet man NP vollständige Probleme?

3.42 Theorem (Cook)

SAT ist NP vollständig. (Gegeben $\alpha = \gamma_1 \wedge \gamma_2 \wedge \dots \wedge \gamma_r$ mit $\gamma_i = \gamma_{i1} \vee \gamma_{i2} \vee \dots \vee \gamma_{is_i}$ mit $\gamma_{ij} \in \{x_1, \neg x_1, \dots, x_k, \neg x_k\}$)

Frage Gibt es Belegung von x_1, \dots, x_k in $\{0, 1\}$, so dass α Wert 1 bekommt?)

Beweis Sei $L \in P$ beliebig. Zu zeigen $L \leq_{pol} SAT$

$L \in NP \Rightarrow$ es gibt nichtdet. TM M und Polynom $p(x)$ mit $\forall w \in \Sigma^* :$

$(w \in L \Leftrightarrow$ es gibt einen Rechengang von M , angesetzt auf w , der nach $p(|w|)$ Schritten in q_+ endet,
 M fest.

Definiere zu w einen Ausdruck $\alpha_M(w)$ mit:

- $\alpha_M(w)$ ist in konjunktiver Normalform
- $\alpha_M(w)$ lässt sich in Zeit $q(|w|)$ konstruieren, für ein Polynom $q(x)$, das nur von M , aber nicht von w abhängt
- $\alpha_M(w)$ ist erfüllbar $\Leftrightarrow w \in L$.

Die Abbildung $w \mapsto \alpha(w)$ ist dann eine polynomielle Reduktion von L auf SAT.

Annahmen über M: nur 1 Band, $F = \{q_-, q_+\}$, falls $\delta(q, a_i) = \emptyset$: füge $\delta(q, a_i) := \{(q, a_i, 0)\}$ hinzu um Anzahl der Rechenschritte auf $p(|w|)$ auffüllen zu können.

$$Q = \{q_0, q_1, \dots, q_{s-1}, q_s = q_+\}$$

$$\Sigma = \{a_1 = \sqcup, a_2, \dots, a_r\}$$

M hat insgesamt m Übergänge: 5-Tupel¹⁸ d_1, d_2, \dots, d_n .

Sei $w \in \Sigma^*$ und $|w| = n$ nach Definition: $w \in L \Leftrightarrow$ es gibt Folge $K_0, K_1, \dots, K_{p(n)}$ von Konfigurationen von M mit:

K_0 Startkonfiguration von M bei Eingabe von α

K_{i+1} ist eine Folgekonfiguration von K_i , für $1 \leq i \leq p(n) - 1$

der Zustand von M in $K_{p(n)}$ ist q_+

Bei Konstruktion von $\alpha_M(w)$ werden die folgenden booleschen Variablen verwendet:

$q_{t,k}$	für	$0 \leq t \leq p(n)$ $0 \leq k \leq s$	$q_{t,k} = 1$:	q_k ist Zustand von K_t (Zeit = Schrittzahl)	
$a_{t,i,j}$	für	$0 \leq t \leq p(n)$ $1 \leq j \leq r$	$a_{t,i,j} = 1$	a_j ist Inhalt des i - ten Feldes in K_t Klar: nur für $i \leq t$ kann $a_j \neq \sqcup$ sein	$\alpha_M(w)$ besteht
$s_{t,i}$	für	$0 \leq t \leq p(n)$	$s_{t,i} = 1$	Kopf steht in K_t auf Feld i	
$b_{t,l}$	für	$0 \leq t \leq p(n)$ $1 \leq l \leq m$	$b_{t,l} = 1$	Tupel d_l bewirkt Übergang von K_t zu K_{t+1}	

aus mehreren Teilen

S (Startbedingung)

q_0 ist Zustand von K_0

L/S-Kopf steht auf Feld 1

$\underbrace{w}_{n \text{ Zeichen}}$ $\sqcup^{p(n)-n}$ ist Bandinhalt

max benutzte Bandlänge: $p(n)$

$$S := q_{0,p} \wedge S_{0,1} \wedge a_{0,1,j_1} \wedge a_{0,2,j_2} \wedge \dots \wedge a_{0,n+1,1} \wedge \dots \wedge a_{0,p(n),1} \text{ für } w = a_{j_1} a_{j_2} \dots a_{j_n} [O(p(n))]$$

¹⁸ (q, a, q', a', m)

R: (Randbedingung) für alle $K_t : 0 \leq t \leq p(n)$:

M ist in genau einem Zustand, Kopf steht auf genau einem Feld, jedes Feld enthält genau ein Zeichen, beim Übergang zu K_{t+1} kommt genau ein Tupel d_j zur Anwendung.

“ **genau ein** “ bedeutet: mindestens ein und nicht mindestens zwei

$$\begin{aligned} \text{genau_ein}(x_1, \dots, x_l) & \equiv (x_1 \vee x_2 \vee \dots \vee x_l) \wedge \neg((x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee \dots \vee (x_{l-1} \wedge x_l)) \\ & \sim (x_1 \vee \dots \vee x_l) \wedge \bigwedge_{1 \leq i < j \leq l} (\neg x_i \vee \neg x_j) \end{aligned}$$

$$[O(l^2)]$$

$$R_{\text{Zustände}}(t) \equiv \text{genau_ein}(q_{t,0}, q_{t,1}, \dots, q_{t,s}) [O(1)]$$

$$R_{\text{Position}}(t) \equiv \text{genau_ein}(s_{t,0}, s_{t,1}, \dots, s_{t,p(n)}) [O(p(n)^2)]$$

$$R_{\text{Band}}(t) \equiv \bigwedge_{1 \leq i \leq p(n)} \text{genau_ein}(a_{t,i,1}, \dots, a_{t,i,r}) [O(p(n))]$$

$$R_{\text{Übergang}}(t) \equiv \text{genau_ein}(b_{t,1}, \dots, b_{t,n}) [O(1)]$$

$$\mathbf{R} \equiv \bigwedge_{0 \leq t \leq p(n)} R_{\text{Zustände}}(t) \wedge R_{\text{Position}}(t) \wedge R_{\text{Band}}(t) \wedge R_{\text{Übergang}}(t) [O(p(n)^3)]$$

U Übergangsverhalten: für $0 \leq t < p(n)$ ergeben sich Zustand und Kopfposition und Bandinschrift von K_{t+1} auf K_t durch Anwendung von $d_l = (q_{k_l}, a_{j_l}, q_{\tilde{k}_l}, a_{\tilde{j}_l}, m_l)$

$$U(t)$$

$$\begin{aligned} & \equiv \bigwedge_{0 \leq i \leq p(n)} \left[\bigwedge_{1 \leq j \leq r} (\neg s_{t,i} \wedge a_{t,i,j} \rightarrow a_{t+1,i,j}) \right. \\ & \quad \sim (s_{t,i} \vee \neg a_{t,i,j} \vee a_{t+1,i,j} \text{ KNF}) \\ & \quad (s_{t,i} \wedge b_{t,l} \rightarrow q_{t,k_l}) \wedge \\ & \quad (s_{t,i} \wedge b_{t,l} \rightarrow a_{t,i,j_l}) \wedge \\ & \quad (s_{t,i} \wedge b_{t,l} \rightarrow q_{t+1,\tilde{k}_l}) \wedge \\ & \quad (s_{t,i} \wedge b_{t,l} \rightarrow a_{t+1,i,\tilde{j}_l}) \wedge \\ & \quad \left. (s_{t,i} \wedge b_{t,l} \rightarrow s_{t+1,i+n_l}) \wedge \right] \end{aligned}$$

$$\mathbf{U} \equiv \bigwedge_{0 \leq t \leq p(n)} U(t) [O(p(n)^2)]$$

Damit

$$\alpha_M(w) \equiv S \wedge R \wedge U \wedge \underbrace{q_{p(1),S}}_{\text{inderletztenKonfigurationistderZustand } q_+ = q_s}$$

$\alpha_M(w)$ ist in KNF; Länge $O(\underbrace{p(n)^3}_{\text{Anzahl der Literale}} \cdot \underbrace{\log n}_{\text{Länge der Literale}})$

$\alpha_M(w)$ erfüllbar $\stackrel{!}{\Leftrightarrow} M$ akzeptiert w in $p(|w|)$ Schritten

Damit: SAT ist NP-vollständig

Klar: Sei L_0 ein NP-vollständiges Problem. Dann gilt:

a) $L_0 \in P \Leftrightarrow P = NP$ Klar

b) $\forall L_1 \in NP : (L_0 \leq_{pol} L_1 \Rightarrow L_1 \text{ ebenfalls NP-vollständig})$

Beweis: Sei $L \in NP$ beliebig $\underbrace{\Rightarrow}_{L_0 \text{ NP-vollst.}} L \leq_{pol} L_0 \leq_{pol} L_1 \underbrace{\Rightarrow}_{\text{Einsetzen}} L \leq_{pol} L_1$

3.42.1 Definition (3 SAT)

wie SAT, aber Klausellänge ≤ 3 : $\underbrace{(x_1 \vee \neg x_2 \vee x_3)}_{\leq 3} \wedge (\neg x_2 \vee \neg x_3 \vee x_4) \dots$

3.42.2 Satz

3 SAT ist NP-vollständig

Beweis: Wir zeigen $SAT \leq_{pol} 3SAT$.

Sei also β ein Ausdruck in KNF, und sei $\kappa \equiv Z_1 \vee Z_2 \vee \dots \vee Z_t$ eine Klausel von β mit $t > 3$, mit $Z_i \in \{x_1, x_2, \dots, x_n\} \cup \{\neg x_1, \neg x_2, \dots, \neg x_n\}$.

Seien y_1, \dots, y_t neue boolesche Variablen,

ersetze κ durch $\lambda \equiv y_1 \wedge (Z_1 \vee (y_1 \rightarrow Z_2)) \vee \underbrace{(Z_2 \vee (y_2 \rightarrow y_3))}_{Z_2 \vee \neg y_2 \vee y_3} \wedge \dots \wedge (Z_{t-1} \vee$

$(y_{t-1} \rightarrow y_t)) \wedge (Z_t \vee \neg y_t)$

$\Rightarrow \lambda \in 3SAT$

Behauptung: Sei ψ eine Bewertung (mit 0/1) der Variablen Z_i . Dann gilt: ψ lässt sich so auf die y_1, \dots, y_t fortsetzen, dass

$$\psi(\lambda) = 1 \Leftrightarrow \psi(\kappa) = 1$$

Beweis:

“ \Rightarrow “ Angenommen, ψ ist so fortgesetzt, dass $\psi(\lambda) = 1$

Zeige $\psi(\kappa) = 1$.

Angenommen, $\psi(Z_i) = 0$ für alle Z_i

$\Rightarrow \psi(\lambda) = 1$ $\psi(y_1) = 1$ und $\psi(y_1 \rightarrow y_2) = 1 \Rightarrow \psi(y_2) = 1$
und ... und

$\psi(y_{t-1} \rightarrow y_t) = 1 \Rightarrow \psi(y_t) = 1$ und $\psi(\neg y_t) = 1$ Widerspruch

⇒ Annahme war falsch ⇒ ex i: $\psi(X_i) = 1 \Rightarrow \psi(\kappa) = 1$

“⇐“ Sei $\psi(\kappa) = 1$. ⇒ $\psi(Z_1) = \dots = \psi(Z_{i-1}) = 0; \psi(Z_i) = 1$
 definiere Fortsetzung von ψ auf die y_j :

$$\begin{aligned} \psi(y_1) &:= 1, \dots, \psi(y_i) := 1 \\ \psi(y_{i+1}) &:= 0, \dots, \psi(y_t) := 0 \\ \underbrace{y_1}_{=1}, \underbrace{Z_1}_{=0} \vee (\underbrace{y_1}_{=1} \rightarrow \underbrace{y_2}_{=1}), \dots, \underbrace{Z_{i-1}}_{=0} \vee (\underbrace{y_{i-1}}_{=1} \rightarrow \underbrace{y_i}_{=1}) \\ &\quad \underbrace{Z_i}_{=1} \vee (\underbrace{y_i}_{=1} \rightarrow \underbrace{y_{i+1}}_{=0}) \\ \underbrace{Z_{i+1}}_{=?} \vee (\underbrace{y_{i+1}}_{=0} \rightarrow \underbrace{y_{i+2}}_{=0}), \dots, \underbrace{Z_{t-1}}_{=?} \vee (\underbrace{y_{t-1}}_{=0} \rightarrow \underbrace{y_t}_{=0}), \underbrace{Z_t}_{=1} \vee \underbrace{\neg y_t}_{=1} \end{aligned}$$

3.42.3 Bemerkung

$$\begin{aligned} \beta &\equiv (\underbrace{\neg x_1 \vee x_2}_{\leq 2}) \wedge (x_2 \vee \neg x_3) \wedge \wedge (x_3 \wedge x_4) \\ \mathbf{2 SAT} \in \mathcal{P} & \\ &\sim x_1 \rightarrow x_2 \quad x_3 \rightarrow x_2 \quad \neg x_3 \rightarrow x_4 \end{aligned}$$

max-2SAT NP-vollständig finde Belegung der x_i , so dass maximale Anzahl von Klauseln “1“ wird

3.43 Theorem (Clique ist NP-vollständig)

Clique¹⁹ ist NP-vollständig.

Beweis: Wir zeigen, dass $3SAT \leq_{pol} Clique$

Klar: $Clique \in NP$

Sei: $\alpha \equiv \gamma_1 \wedge \gamma_2 \wedge \dots \wedge \gamma_k$ mit Klauseln $\gamma_i \equiv \underbrace{x_{i_1}^{\beta_{i_1}} \vee x_{i_2}^{\beta_{i_2}} \vee x_{i_3}^{\beta_{i_3}}}_{\text{Literal}} \vee$ (zur Not

auf 3 Literale auffüllen) mit $x_{i_j}^{\beta_{i_j}} = \begin{cases} x_{i_j}, & \beta_{i_j} = 1 \\ \neg x_{i_j}, & \beta_{i_j} = 0 \end{cases}$

Definiere $G = (V, E)$ folgendermaßen:

- ein Knoten $x_{i_j}^{\beta_{i_j}}$ für jedes Literal $x_{i_j}^{\beta_{i_j}}$, welches in α auftritt
- eine Kante zwischen $x_{i_j}^{\beta_{i_j}}$ und $x_{l_m}^{\beta_{l_m}}$ genau dann, wenn

$$- x_{i_j}^{\beta_{i_j}} \text{ und } x_{l_m}^{\beta_{l_m}} \text{ gehören zu verschiedenen Klauseln, d.h. } i \neq l^{20}$$

¹⁹Clique: Gegeben Graph $G = (V, E)$, Zahl k
 Frage: Gibt es $v' \subset V$ mit $|v'| = k$ und $\forall a, b \in v' : (a, b) \in E$
²⁰ i und l Klauselindizes

$$- x_{i_j} = x_{l_m} \Rightarrow \beta_{i_j} = \beta_{l_m}$$

Klar: G läßt sich in polynomieller Zeit aus α konstruieren.

Behauptung: α erfüllbar $\Leftrightarrow G$ hat k -Clique

Beweis:

“ \Rightarrow “ Sei α erfüllbar \Rightarrow in jeder Klausel γ_i muss mindestens ein Literal den Wert 1 haben,

d.h. es gibt ein Literal $x_{i_j}^{\beta_{i_j}}$ mit Wert 1 in jedem γ_i

Im Graphen G sind diese Knoten paarweise durch eine Kante verbunden, denn

- sie stehen in verschiedenen Klauseln

- $(x_{i_j} = x_{l_m} \Rightarrow \beta_{i_j} = \beta_{l_m})$ ist ebenfalls immer erfüllt, denn diese Literale haben alle denselben Wahrheitswert 1

$\Rightarrow G$ hat k -Clique.

“ \Leftarrow “ Sei v' eine k -Clique von G . Weil in G nur Knoten mit Literalen aus verschiedenen Klauseln verbunden sind, folgt: V' enthält genau einen Knoten $x_{i_j}^{\beta_{i_j}}$ aus jedem γ_i

$$\text{setze Variablenbelegung } \psi(\underbrace{x_{i_j}}_{=x_r}) := \begin{cases} 1, & \beta_{i_j} = 1 \\ 0, & \beta_{i_j} = 0 \end{cases}$$

Wegen der Bedingung: “ $x_{i_j} = x_{l_m} \Rightarrow \beta_{i_j} = \beta_{l_m}$ “ ist ψ wohldefiniert, d.h. es hängt nicht davon ab aus welcher Klausel die Variable stammt.

$\Rightarrow \psi$ macht in jeder Klausel γ_i ein Literal zu 1 $\Rightarrow \psi(\alpha) = 1$

qed.

3.44 Vertex Cover

Gegeben: Graph $G = (V, E)$, Zahl k

Frage: Gibt es $v' \subseteq V$ mit $|v'| = k$ und \forall Kanten $(a, b) \in E$ gilt $a \in v'$ oder $b \in v'$ (oder beide)

3.44.1 Theorem

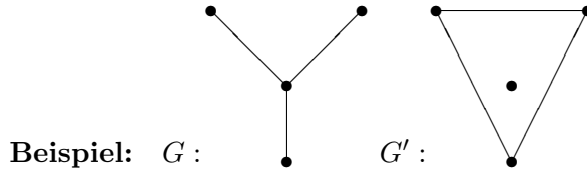
Vertex Cover ist NP-vollständig

Beweis Clique $\leq_{pol} \geq$ Vertex Cover: VertexCover $\in NP$

wir zeigen: " \leq_{pol} "

Sei $G = (V, E)$ und k ein Clique-Beispiel

Konstruiere den Komplementgraph $G' := (V, V \times V \setminus E)$



Behauptung: G hat k -Clique $\Leftrightarrow G'$ hat $(|v| - k)$ -VertexCover

" \Rightarrow " Sei W k -Clique von G .

Betrachte $W' := V \setminus W$ in G' . Klar: $|w'| = |v| - k$

W' ist Vertex Cover von G' . Denn Sei e Kante von G' .

Angenommen, kein Endpunkt von e liegt in W'

\Rightarrow beide Endpunkte liegen in W

\Rightarrow W Clique von G $e \in E$ Widerspruch zur Definition von G'

\Rightarrow mindestens ein Endpunkt von e liegt in $W' \Rightarrow W'$ VertexCover von G' .

" \Leftarrow " analog

3.45 (0,1)-Integer Programming

Zur Auswahl stehen 4 Beteiligungen:

	Preis	jährliche Auszahlung	
1.	5000,-	800,-	Ihr Kapital: 14.000 EUR
2.	7000,-	1100,-	
3.	4000,-	600,-	
4.	3000,-	400,-	

Rendite $\frac{800}{5000} = 16\%$ bei 1. am höchsten

schön wäre: 14.000 in 1. zu investieren. Geht nicht! Entweder ganz oder gar nicht.

\rightarrow maximiere $8 \cdot x_1 + 11 \cdot x_2 + 6 \cdot x_3 + 4 \cdot x_4$ (Zielfunktion = Auszahlung)

$$\left. \begin{array}{r}
 5 \cdot x_1 + 7 \cdot x_2 + 4 \cdot x_3 + 3 \cdot x_4 \leq 14 \\
 x_1 \leq 1 \\
 x_2 \leq 1 \\
 x_3 \leq 1 \\
 x_4 \leq 1
 \end{array} \right\} \text{ " lineares Programm " } \rightarrow \text{ lineares Programming (LP) }$$

Man könnte noch zusätzliche Bedingungen haben, z.B.

“höchstens 3 Beteiligungen erlaubt“: $x_1 + x_2 + x_3 + x_4 \leq 3$

“2. geht nur mit 4.“: $x_2 - x_4 \leq 0$

lässt sich für reelle x_1 leicht ausführen (z.B. in mittlerer Zeit $O(n \cdot d_i)$,
 $n = \#$ Nebenbedingungen, $d = \#$ Variablen)
damit ergibt sich in unserem Fall:

jährliche Auszahlung: 2142,85 bei $x_1 = 1, x_2 = 0.857, x_3 = 0, x_4 = 1$.

Bei unserem Problem brauchen wir ganzzahlige Lösungen x_i , sogar $\in \{1, 0\}$

optimale ganzzahlige Lösung

jährliche Auszahlung: 2100 bei $x_1 = 0, x_2 = x_3 = x_4 = 1$

\rightsquigarrow Integer (linear) Programming.

3.45.1 LP

LP bedeutet geometrisch:

jede Nebenbedingung definiert Halbraum

alle zusammen ein Konvexes Polyeder F

Maximierung der Zielfunktion

verschiebe eine Hyperebene so, dass sie zu F tangential wird.

Berührungspunkt ist die Lösung

3.45.2 Theorem

(0,1)-Integer Programming ist NP-vollständig.

Sogar schon die “einfache“ Feasibility-Version:

Gegeben Matrix $C \in M_{k \times m}(\mathbb{Z})$, Vektor $d \in \mathbb{Z}^k$

Frage Gibt es Vektor $e \in \{0, 1\}^m$ mit $C \cdot e \geq^{21} d$

²¹koeffizientenweise

Beispiel

$$\underbrace{\begin{pmatrix} -3 & 5 & 1 \\ 2 & -6 & 3 \end{pmatrix}}_C \underbrace{\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}}_e = \begin{pmatrix} -2 \\ 5 \end{pmatrix} \geq \underbrace{\begin{pmatrix} -3 \\ 5 \end{pmatrix}}_d$$

hier wird gefragt: lassen sich die Nebenbedingungen überhaupt durch Variablen $x_i = e_i$ erfüllen?

Wir zeigen: $SAT \leq_{pol} IP$ (=Integer Programming)

Sei $\alpha = \gamma_1 \wedge \dots \wedge \gamma_k$ in konjunktiver Normalform seien x_1, \dots, x_m die booleschen Variablen, die in α vorkommen.

Vorüberlegung Wenn ein γ_i sowohl x_j als auch $\neg x_j$ enthält, hat γ_i stets Wert 1 und kann weggelassen werden.

Also: In jedem γ_i steht entweder x_j oder $\neg x_j$ oder kein Vorkommen von x_j

Definiere Matrix $C \in M_{k \times n}(\mathbb{Z})$ durch $c_{ij} := \begin{cases} 1, & \text{falls } x_j \text{ Literal in } \gamma_i \\ -1, & \text{falls } \neg x_j \text{ Literal in } \gamma_i \\ 0, & \text{sonst} \end{cases}$

und Vektor $d \in \mathbb{Z}^k$ durch $d_i := 1 - \text{Anzahl der } \neg x_j \text{ in } \gamma_i$, für $1 \leq i \leq k$

Behauptung: α erfüllbar $\Leftrightarrow \exists e \in \{0, 1\}^m$ mit $C \cdot e \geq d$.

“ \Rightarrow “ Sei $\psi(\alpha) = 1$. Definiere $e \in \{0, 1\}^m$ durch $e = \begin{pmatrix} \psi(x_1) \\ \psi(x_2) \\ \vdots \\ \psi(x_n) \end{pmatrix}$

Zeige $C \cdot e \geq d$

Für jede Klausel $\gamma_i : \psi(\gamma_i) = 1 \Rightarrow \text{es gibt } j :$

$$x_j \text{ Literal in } \gamma_i \text{ und } \psi(x_j) = 1$$

oder

$$\neg x_j \text{ Literal in } \gamma_i \text{ und } \psi(x_j) = 0$$

$$\Rightarrow \sum_{x_j \text{ in } \gamma_i} \psi(x_j) \geq 1 \text{ oder } \sum_{\neg x_j \text{ in } \gamma_i} \psi(x_j) \leq \text{Anzahl der Literale } \neg x_j \text{ in } \gamma_i - 1 = d_i$$

\Rightarrow i-tes Element von

$$C \cdot e = \sum_{j=1}^m c_{ij} e_j \stackrel{\text{Def. } C, e}{=} \sum_{x_j \text{ in } \gamma_i} \psi(x_j) - \sum_{\neg x_j \text{ in } \gamma_i} \psi(x_j) \stackrel{!}{\geq} 1 - \sum_{\neg x_j \text{ in } \gamma_i} 1 = d_i$$

“ \Leftarrow “ Gelte $C \cdot e \geq d$ für Vektor $e \in \{0, 1\}^m$.

zu zeigen: es gibt ψ mit $\psi(\alpha) = 1$

Definiere $\psi(x_j) := e_j$. Zeige $\psi(\gamma_i) = 1$ für alle γ_i Angenommen, es gibt i mit $\psi(\gamma_i) = 0$

$$\Rightarrow \psi(x_j) = \begin{cases} 0, & \text{falls } x_j \text{ in } \gamma_i \\ 1, & \text{falls } \neg x_j \text{ in } \gamma_i \end{cases}$$

\Rightarrow i-tes Element von $C \cdot e$:

$$\sum_{x_j \text{ in } \gamma_i} \underbrace{\psi(x_j)}_{=0} - \sum_{\neg x_j \text{ in } \gamma_i} \underbrace{\psi(x_j)}_{=1} = - \sum_{\neg x_j \text{ in } \gamma_i} 1 < d_i$$

Widerspruch

qed.

3.46 weitere NP-vollständige Probleme

3.46.1 Hamiltonian Circle (HC)

Gegeben: Graph $G = (V, E)$

Frage: Gibt es eine Rundtour durch G , die jeden Knoten $\in V$ genau einmal besucht?

3.46.2 Theorem (HC ist NP vollständig)

Hamiltonian Circle ist NP-vollständig.

\in NP: klar.

Rate eine der $(|v| - 1)!$ vielen zyklischen Knotenreihenfolgen und teste, ob die zugehörigen Kanten in G existieren.

NP-hart: Wir zeigen $3SAT \leq_{pol} HC$

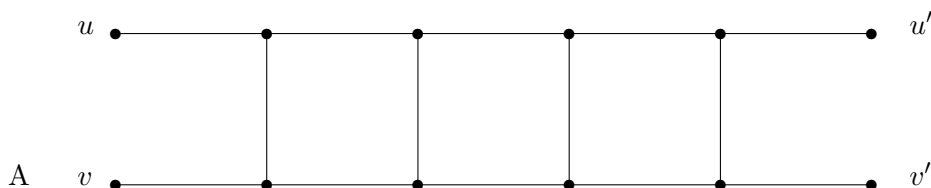
Sei $\alpha \equiv \gamma_1 \wedge \dots \wedge \gamma_k$ in KNF mit Klauseln γ_i der Länge 3.

Seien x_1, x_2, \dots, x_n die boole'schen Variablen, die in α auftreten.

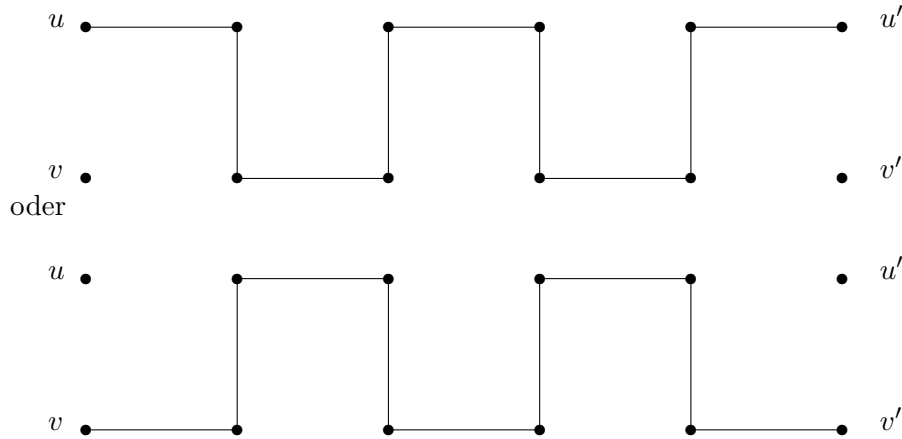
Ziel: Konstruiere zu α (in polynomieller Zeit) einen Graphen G mit

$$\alpha \text{ erfüllbar} \Leftrightarrow G \text{ hat einen HC}$$

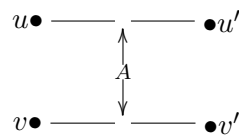
G setzt sich zusammen aus folgenden Teilgraphen ("Gadgets")



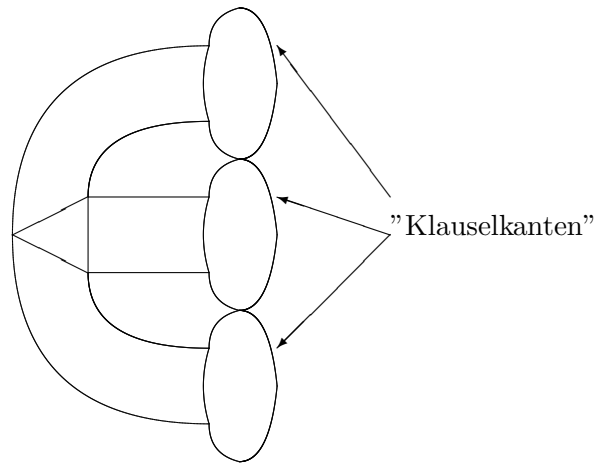
Falls G einen HC hat, kann A nur so durchlaufen werden:



(Bei anderen Routen: Mehrfachbesuche!)
symbolisch:



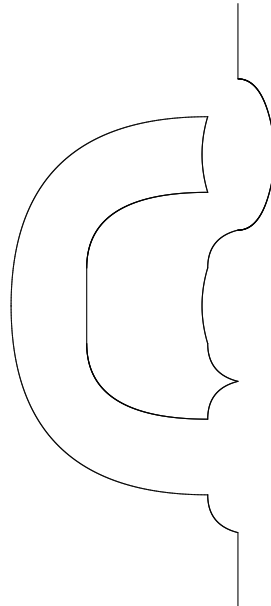
B :



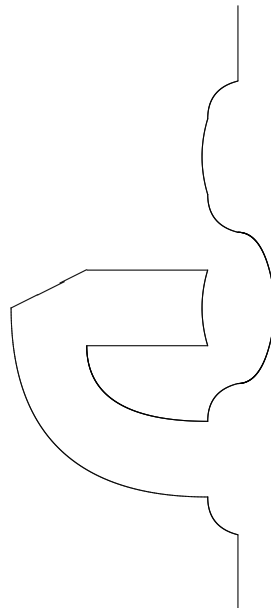
Behauptung: B hat folgende Eigenschaft:
Falls B Teilgraph von G und G einen HC besitzt, so kann dieser Hamilton-Kreis

- nicht alle drei Klauselkanten durchlaufen
- aber jede Teilmenge!

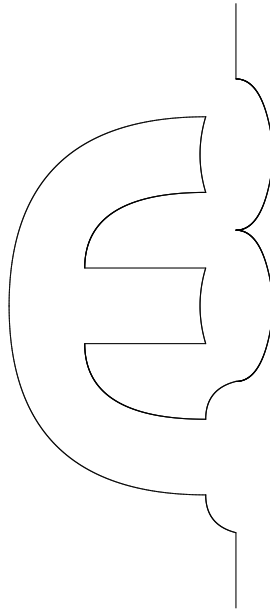
eine Klauselkante am Rand :



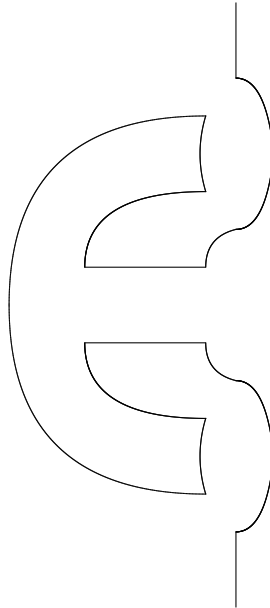
eine Klauselkante, in der Mitte :



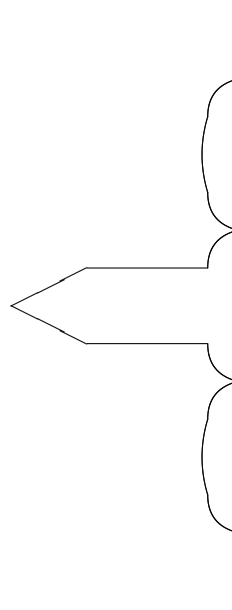
zwei Klauselkanten hintereinander :



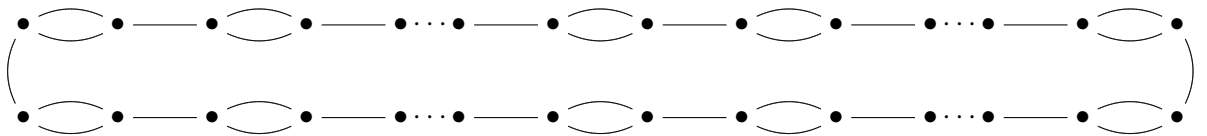
zwei Klauselkanten mit Lücke :



keine Klauselkante :



Konstruiere von G aus Gadgets A und B:



Behauptung: α erfüllbar $\Leftrightarrow G$ hat HC

Beweis:

“ \Rightarrow “ Sei $\psi(\alpha) = 1 \Rightarrow$ in jeder Klausel hat mindestens 1 Literal den Wert 1.
 Durchlaufe für jede Variable x_1 diejenige Literalkante mit Wert 1 und nicht die zugehörige Klauselkante, sondern die übrigen Kanten der B-Konstruktion zu denjenigen γ_j , in deren x_i (oder $\neg x_i$) vorkommt.
 Das geht wegen der B-Eigenschaft
 \Rightarrow Graph G hat HC

“ \Leftarrow “ Angenommen, G hat einen HC.
 Setze für jede Variable die Bewertung desjenigen Literale auf 1, dessen Literalkante von dem Kreis besucht wird.
 Eigenschaft von B

“ \Rightarrow “ zu jeder Klausel gibt es eine Klauselkante, die nicht besucht wird

“ \Rightarrow “ die zugehörige Literalkante wird besucht (wegen Eigenschaft A)

“ \Rightarrow “ Literal hat Wert 1

“ \Rightarrow “ α hat Wert 1

3.46.3 Folgerung:

Traveling Salesperson ist NP-vollständig

Gegeben: Graph $G = (V, E)$ und Kostenfunktion $c : E \rightarrow \mathbb{Q}_{\geq 0}$, Zahl $k \in \mathbb{Q}$

Frage: Gibt es Rundtour, die jeden Knoten einmal besucht und Gesamtkosten $\leq k$ hat?

3.46.4 Theorem (TSP NP-vollständig)

TSP ist NP-vollständig!

$\in NP$: rate + teste

NP-hart: $HC \leq_{pol} TSP$ wie folgt:
Sei HC-Beispiel $G = (V, E)$ gegeben
definiere Kostenfunktion:

$$c : V \times V \rightarrow \mathbb{N}_0 \\ (v, w) \mapsto \begin{cases} 1, & (v, w) \in E \\ 2, & \text{sonst} \end{cases} \quad k := |V|$$

Klar: es existiert HC \Leftrightarrow es existiert TSP-Tour mit Gesamtkosten $\leq k$.

3.46.5 Bemerkungen

1. Eigentlich will man nicht wissen, ob es eine TSP-Tour mit $\underbrace{\text{Gesamtkosten} \leq k}_{\text{Entscheidungsproblem}}$ gibt, sondern man will eine $\underbrace{\text{optimale Tour}}_{\text{Optimierungsproblem}}$ finden.

anschaulich: entscheiden ist einfacher als optimieren

Aber: wenn z.B. die Werte der Kostenfunktion c in \mathbb{N} liegen, sind beide Probleme gleich schwierig!

Erinnerung: Binäre Suche

Gegeben: sortiertes Array

Suche nach einem Wert x , schaue in der Mitte des Array und vergleiche ob die Zahl kleiner als x ist, wenn ja, vergesse den rechten Teil des Arrays und wiederhole das verfahren im linken Teil des Arrays. Analoges Verfahren für die rechte Seite, wenn der Wert in der Mitte größergleich x ist.

Anzahl der benötigten Schritte $\tilde{\log} n$

Jetzt: Opt TSP für ganzzahligen Fall

$$Min := 0, Max := \sum_{e \in E} c(e) =: k$$

Solange Max-Min ≥ 1 :

$$t := \lfloor \frac{Min + Max}{2} \rfloor;$$

entscheide: Gibt es Rundreise mit Kosten $\leq t$?

2. Euklidischer TSP

Gegeben: Punkte p_1, \dots, p_n im \mathbb{R}^2 , Schranke k

Frage: Gibt es Rundreise mit Gesamtlänge $\leq k$?

man weiß: NP-hart

man weiß nicht: \in NP ?

Achtung: Turingmaschine kann keine Wurzeln ziehen
(aber sie könnte die Wurzel rational approximieren)

Aber: beim Euklidischen TSP müssen Summen von n Quadratwurzeln abgeschätzt werden.

Hierfür sind keine polynomiellen Approximationsverfahren bekannt.

3.46.6 Minimum Weight Triangulation

(ohne Beweis)

(Mulzer, Rote '06) war offen seit 1979

Gegeben: n Punkt in \mathbb{R}^2 Schranke k

Frage: Gibt es Triangulation der Punktmenge mit Gesamtkantenlänge $\leq k$?

Theorem NP-hart.

3.46.7 Minimum Dilation Graph

Gegeben: n Punkt im \mathbb{R}^2 , Schranken m und δ

Frage: Kann man die Punkte mit $\leq m$ Kanten so verbinden, dass alle Umwege $\leq \delta$ sind?

d.h. ob für alle Kosten $p \neq q$ gilt: $|\pi(p, q)| \leq \delta|pq|$

Theorem: NP-hart (Kutz RK '06)

3.46.8 Partition:

Gegeben: Zahlen $a_1, a_2, \dots, a_n \in \mathbb{N}$

Frage: Gibt es Indexmenge $I \subseteq 1, \dots, n$ mit $\sum_{i \in I} a_i = \sum_{i \in \{1, \dots, n\} \setminus I} a_i$

Theorem: NP-vollständig.

3.46.9 Warehouseman- Problem

Gegeben: rechtwinkliger Grundriss, n Rechtecke

Frage: lassen sich diese Rechtecke kollisionsfrei von einer Start- in eine Zielkonfiguration überführen?

Folgerung aus Partition: NP-hart

Beweis: Partition \leq_{pol} Warehouseman
 Gegeben a_1, a_2, \dots, a_n , Setze $A := \frac{a_1 + a_2 + \dots + a_n}{2}$
 Baue folgende Umgebung
 B lässt sich von oben nach unten bewegen

“ \Leftrightarrow “ die kleinen Hindernisse passen in die Ärme”

“ \Leftrightarrow “ Partition ist lösbar

Warehouseman ist sogar PSPACE-vollständig.

3.47 Wie geht man mit NPvollständigen Problemen um?

(Ein paar Ideen)

3.47.1 Prüfe, ob Problemkomplexität abnimmt, wenn einen oder mehrere Parameter beschränkt.

Beispiel 1: Vertex Cover (VC)

Gegeben: $G = (V, E)$, k .

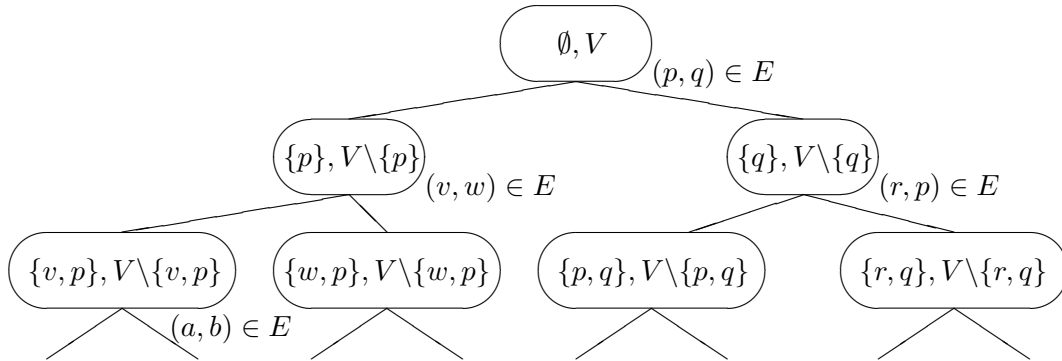
Frage: Gibt es $v' \subseteq V$ mit $|v'| \leq k$ und $\forall (p, q) \in E : p \in v' \text{ oder } q \in v'$?
 NP-vollständig

Angenommen, k ist beschränkt (z.B. = 183)

1. Ansatz alle $\binom{|v|}{k}$ viele Teilmengen²² von V testen, ob ein Vertex Cover darunter ist

²² $\in O(n^k)$, falls $n := |v|$ wächst bei festem k polynomiell in n .

2. Ansatz erzeuge einen "Cover-Tree"



- nimm stets Kante aus E hinzu, von der noch kein Endpunkt abgedeckt ist, und betrachte beide Möglichkeiten.
- fahre fort, bis Baum Tiefe k erreicht,
- Behauptung: Wenn es k -Cover gibt, steht es in einem der Blätter dieses Baumes

Laufzeit :

Anzahl der Knoten	·	Aufwand für Finden einer noch nicht abgedeckten Kante
$\sim 2^k$		$\sim n^2$
exponentiell in k , unabhängig von n		polynomiell in n , unabhängig von k

$O(2^k \cdot n^2)$ besser als $O(n^k)$ vom 1. Ansatz
 Vertex Cover ist **Fixed Parameter Tractable**
 (Für Clique weiß man das nicht)

Beispiel 2: Partition

Gegeben: n Zahlen $a_1, a_2, \dots, a_n \in \mathbb{N}$

Frage: Gibt es $I \subseteq \{1, 2, \dots, n\}$ mit $\underbrace{\sum_{i \in I} a_i}_{\frac{A}{2}} = \sum_{i \in \{1, \dots, n\} \setminus I} a_i$?

NP-vollständig

Algorithmus zur Lösung von Partition :

Baue ein boole'sches Array $T = (t_{i,j})$ für $1 \leq i \leq n$, $0 \leq j \leq \frac{A}{2}$ mit

$$t_{i,j} = \begin{cases} 1, & \text{falls es Teilmenge} \\ & \text{von } a_1, a_2, \dots, a_i \\ & \text{mit Summe } j \text{ gibt} \\ 0, & \text{sonst} \end{cases}$$

Initialisierung:

$$t_{1,j} := \begin{cases} 1, & \text{falls } j = 0 \text{ oder } a_1 = j \\ 0, & \text{sonst.} \end{cases}$$

$$t_{i+1,j} := \begin{cases} 1, & \text{falls } t_{i,j} = 1 \text{ oder } (a_{i+1} \leq j \text{ und } t_{i,j-a_{i+1}} = 1) \\ 0, & \text{sonst.} \end{cases}$$

Klar: • Array T läßt sich in $O(n \cdot \frac{A}{2})$ Schritten bauen.

• Partition lösbar $\Leftrightarrow t_{n, \frac{A}{2}} = 1$. Fertig!

$O(n \frac{A}{2})$ ist nicht polynomiell in der Länge der Eingabe $\sum \log(a_i)$ ($A = \sum a_i$)

Also: Wir haben nicht gezeigt, dass $P = NP$ ist, aber:

Falls Eingabewerte a_i des Partition-Problems beschränkt sind ($\leq K$)
so haben wir eine Lösung in Zeit $O(n \cdot n \cdot K) = O(n^2)$.
pseudo-polynomiell.

3.47.2 Prüfe ob eine Approximation des Optimums ausreicht**Beispiel: Traveling Salesperson**

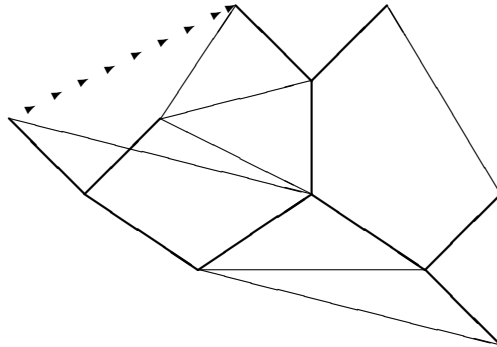
Gegeben: Graph $G = (V, E)$, $c : E \rightarrow \mathbb{O}_{>0}$ Kostenfunktion (=Kantengewichte)

Gesucht: Rundtour durch alle Knoten mit minimalen Gesamtkosten.
NP-vollständig.

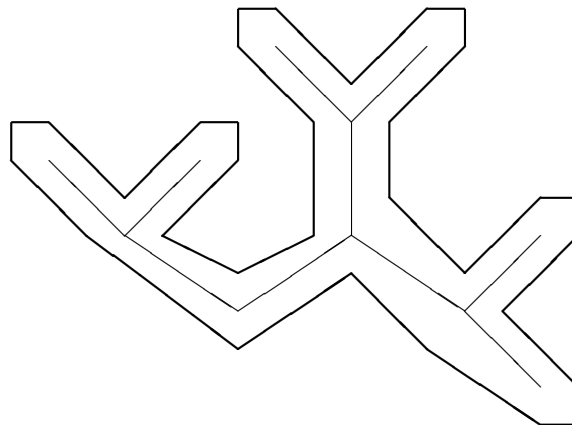
Ansatz: Berechne den minimalen Spannbaum (= Minimum Spanning Tree MST) von G bzgl. Kostenfunktion c .

= Baum mit Kanten aus E , der

- Jeden Knoten aus V enthält
- minimale Kostensumme hat



Den MST kann man bequem in Zeit $O(n^3)$ berechnen, $n = |v|$ (später)



Jetzt:
herum \rightsquigarrow Tour R

laufe einmal um MST

Theorem: $|R| \leq 2 \cdot |OPT|$, OPT = optimale TSP-Tour

Beweis: T entstehe aus OPT durch Weglassen einer Kante

- \Rightarrow T eine Kette
- \Rightarrow T Baum
- $\Rightarrow |MST| \leq |T|$
- $\Rightarrow |R| = 2|MST| \leq 2|T| < 2|OPT|$

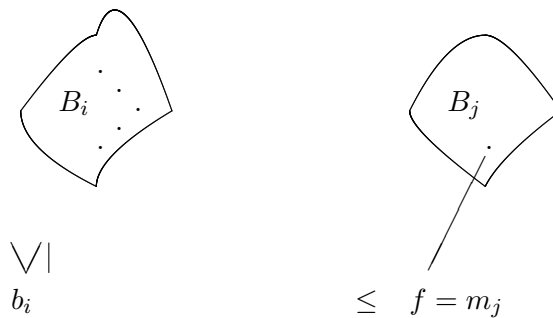
qed.

⇒ **zeigt:** Man kann optimale TSP-Tour leicht bis auf Faktor 2 approximieren.

3.47.3 Versuche, den Suchraum zu begrenzen (Branch and Bound)

Allgemein:

- Wollen Funktion f auf Bereich B maximieren
- teile rekursiv B in Teilbereiche B_i auf
- unterhalte für jedes B_i eine obere Schranke b_i für $\max_{\xi \in B_i} f(\xi)$
- Angenommen, für manche B_j ist $m_j := \max_{\xi \in B_j} f(\xi)$ bekannt.
- Falls für ein B_i gilt: $b_i \leq m_j$ für ein j



⇒ brauchen B_i nicht weiter zu betrachten (Bound = Prune)

Beispiel: $\{0, 1\}$ **Integer-Programming** maximiere

$$f(x_1, x_2, x_3, x_4) := 8 \cdot x_1 + 11 \cdot x_2 + 6 \cdot x_3 + 4 \cdot x_4$$

unter Nebenbedingung

$$5 \cdot x_1 + 7 \cdot x_2 + 4 \cdot x_3 + 3 \cdot x_4 \leq 14$$

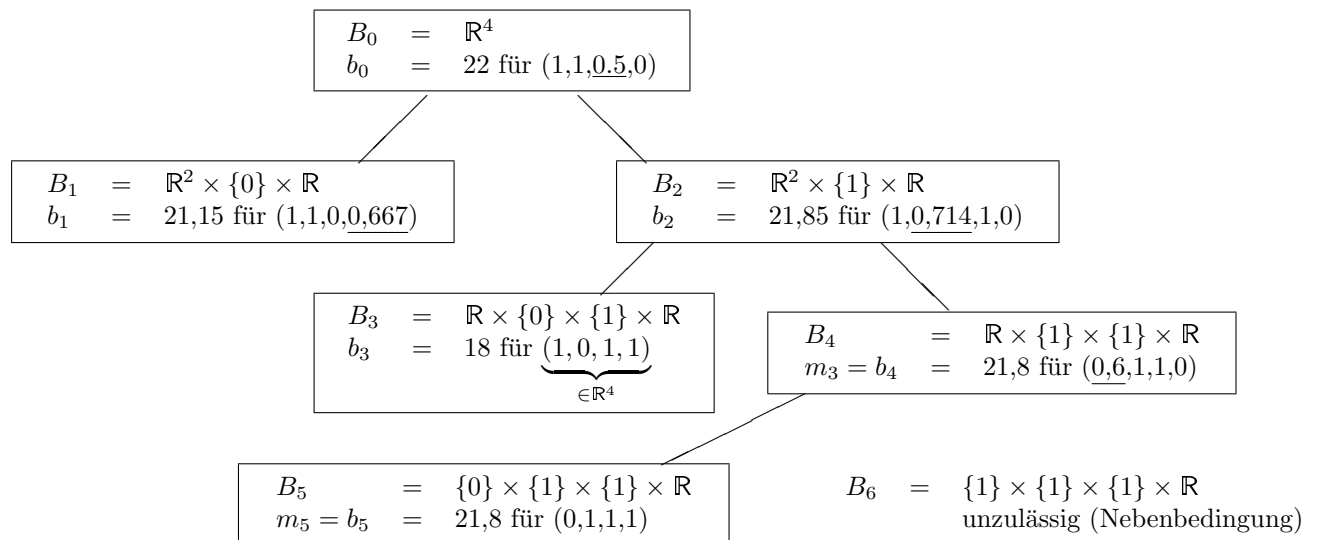
und

$$x_1, x_2, x_3, x_4 \in \{0, 1\}$$

NP-vollständig

Idee:

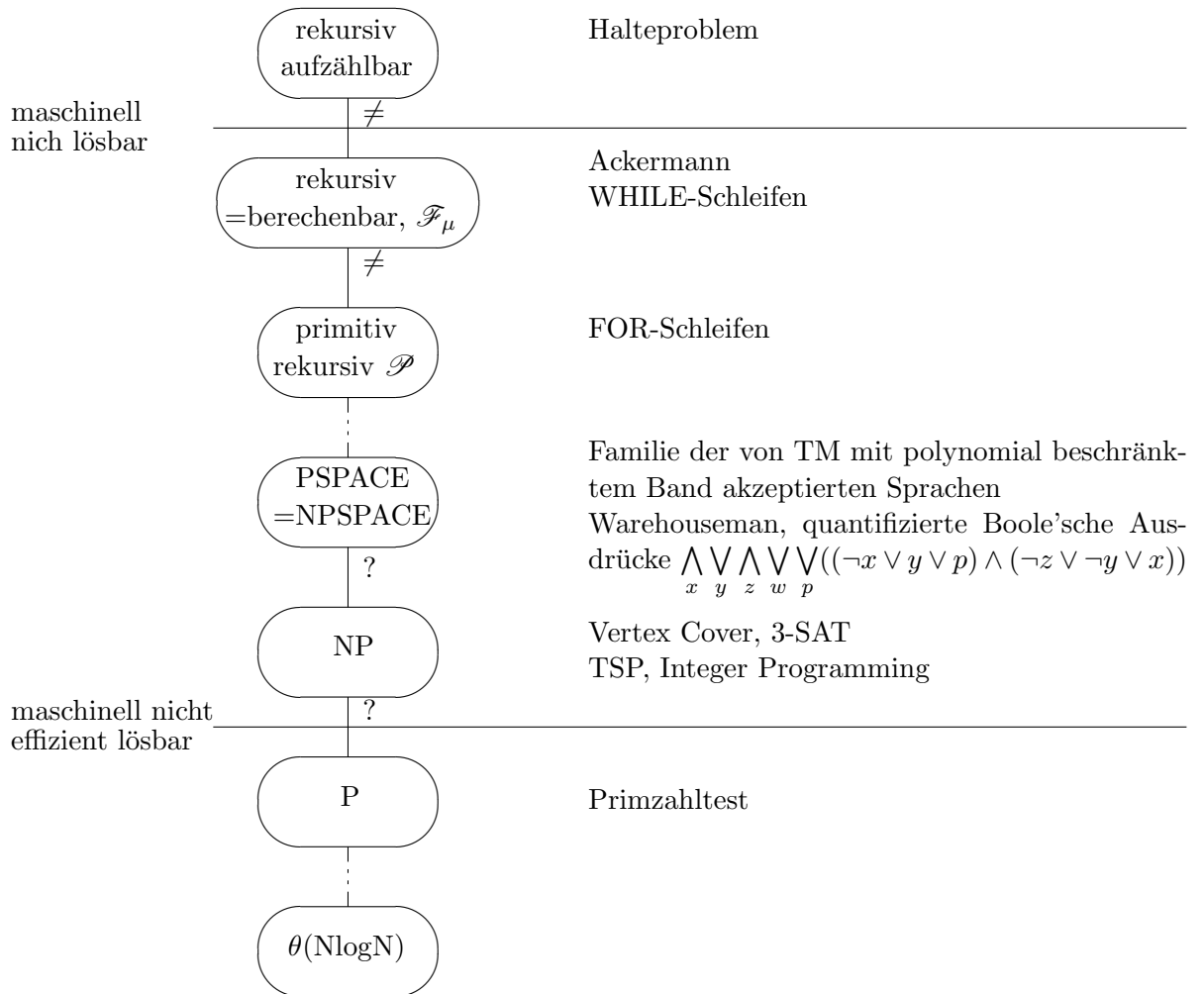
- Versuche, durch *B&B* beim Ausprobieren der x_i zu sparen
- finde zuerst reelle (rationale) Lösungen für x_1, \dots, x_4 (Relaxieren)



beachte: b_i sind obere Schranken für ganzzahlige Lösungen

B_1 nicht erforderlich = "Bound"

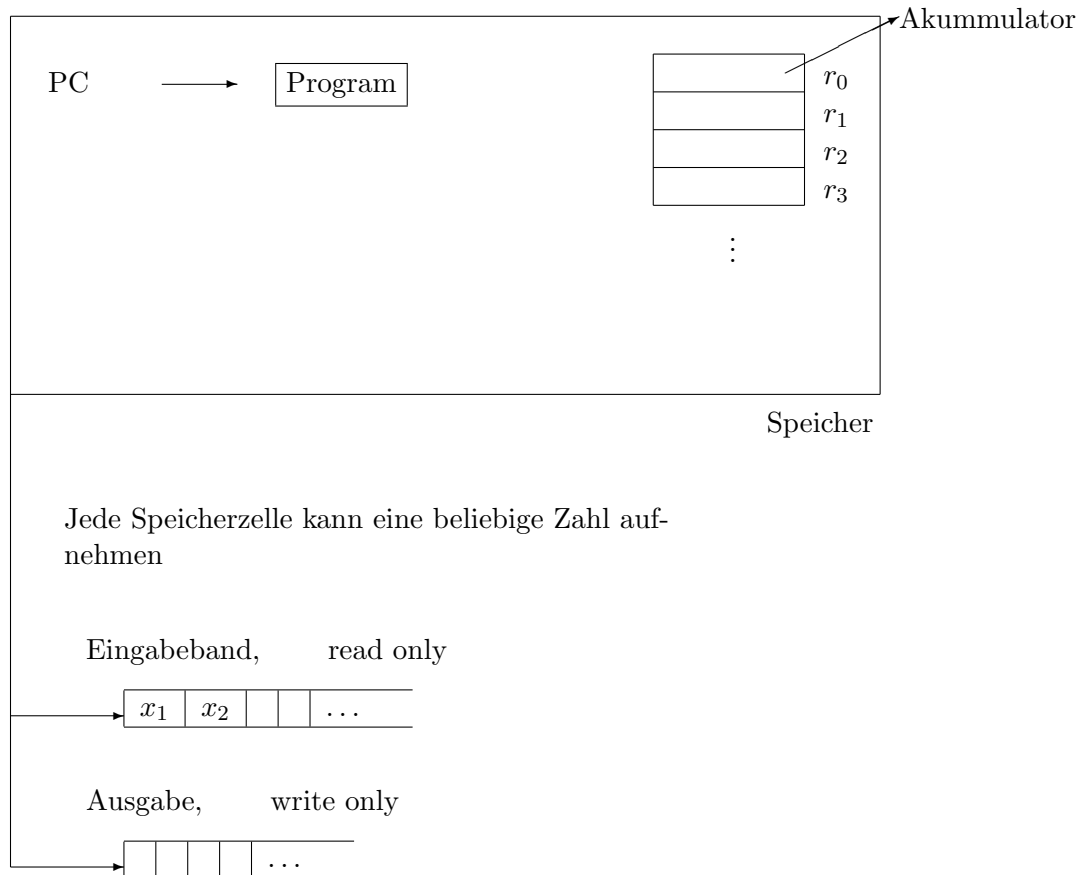
3.48 Weltbild



3.49 RAM

Neue Hardware Random Access Maschine (= RAM)

hat nichts mit Zufall zu tun, sondern mit wahlfreiem Zugriff



3.49.1 Was kann RAM?

- Eingabe, Ausgabe
- arithmetische Operationen
- indirekte Adressierung
- Sprünge

zur Abkürzung: $c(i)$: Inhalt von Register i

3.49.2 Operanden für RAM-Befehle:

- = i die Zahl i
- i Inhalt von Register r_i
- * i Inhalt von Register $r_{c(i)}$

3.49.3 Wert eines Operanden: a: v(a)

$v(=i) := i$
 $v(i) := c(i)$
 $v(*i) := c(c(i))$

3.49.4 Befehle:

LOAD a : $c(0) := v(a)$
 STORE i : $c(i) := c(0)$
 STORE i : $c(i) := c(0)$
 ADD a : $c(0) := c(0) + v(a)$
 SUB a : $c(0) := c(0) - v(a)$
 MULT a : $c(0) := c(0) \cdot v(a)$
 DIV a : $c(0) := \left\lfloor \frac{c(0)}{v(a)} \right\rfloor$
 READ i : $c(i) :=$ aktuelles Eingabesymbol (unter Lesekopf)
 READ *i : $c(c(i)) :=$ aktuelles Eingabesymbol (unter Lesekopf)
 WRITE a : drucke $v(a)$ auf Ausgabeband an Kopfposition
 JUMP b : setze Programmzähler auf Instruktion b
 JGTZ b : if $c(0) > 0$ then JUMP b
 JZERO b : if $c(0) = 0$ then JUMP b
 HALT : Beende Berechnung

3.50 Kostenmaß e **3.50.1 Einheitskostenmaß**

Zugriff / Operation auf einer Zahl $n \in \mathbb{N}$ kostet 1 Einheit

3.50.2 Logarithmisches Kostenmaß

$$L(n) := \begin{cases} 1 \text{ Einheit, falls } n=0 \\ \lceil \log_2 n + 1 \rceil, \text{ falls } n > 0 \end{cases}$$

Länge der Binärdarstellung von n

OK für Zugriff, Addition, Subtraktion

auch OK für Multiplikation von zwei k -stelligen Zahlen? Bestes bekanntes Verfahren: $O(k \log k \log \log k)$ Schönhage, Strassen '71

3.50.3 Kosten der RAM-Befehle im logarithmischen Maß:

Bereitstellung eines Operanden:

$$*i : L(i) + L(c(i)) + L(c(c(i)))$$

Beispiel: JGTZ b: $L(c(o)) + L(b)$
vergleiche Blum

3.50.4 Wie verhalten sich RAM und TM zueinander?

Lemma: Sei M eine t -zeitbeschränkte TM .
Dann gibt es eine RAM R , die M in Zeit

$O(t(n))$ im Einheitskostenmaß

$O(t(n) \log(t(n)))$ im logarithmischen Kostenmaß

simuliert.

Beweis: Angenommen, M ist 1 – Band TM .

Ansatz: Bilde Bandquadrat i auf Register Nr $i + d$ ab, d Konstante.

- Speichere Kopfposition von M in Register 1.
- Stelle Zeichen des Bandalphabets durch Zahlen dar.
- aue Programm von M in Programm von R ein. 1 Schritt von M : $\stackrel{wedge}{\equiv}$
 $\max c$ (Konstante) Anzahl von Schritten von R

Damit:

– im Einheitskostenmaß:

Rechenzeit von R in $O(t(n))$

– im logarithmischen Maß:

Rechenzeit von R in $\underbrace{O(t(n) \log(t(n)))}_{*}$

dann: M kann höchstens $t(n)$ viele Felder auf Band besuchen

\Rightarrow Registeradressen von R liegen in $O(t(n))$

\Rightarrow Registerzugriffskosten in $O(\log t(n))$ pro Zugriff

Satz: Sei f eine Funktion, die in Zeit $t(n)$ im logarithmischen Kostenmaß auf einer RAM R berechnet werden kann.

Dann kann man f in Zeit $O(t(n)2)$ auf einer (6-Band) TM M berechnen.

Beweis: Verwende je ein Band für

Eingabeband von R

Ausgabeband von R

Speicher

\$	###	i_1	c_1	##	i_2	...	###	□	□...
----	-----	-------	-------	----	-------	-----	-----	---	------

Akkumulator

Operand

Schmierpapier (S)

Simulation der Operationen von R auf M :

a) Speicherzugriff auf Register I :

lies Speicherband, bis $I = i_j$ oder letztes ### gefunden

falls $I = i_j$ gefunden:

bei LOAD I :

kopiere c_j auf Akkuband

bei STORE I :

kopiere alles rechts von c_j auf Band S ,

überschreibe c_j mit Inhalt vom Akkuband,

kopiere Inhalt von Band S dahinter

falls ### gefunden

bei LOAD I :

Fehler

bei STORE I :

überschreiben letztes # auf Speicherband mit $I\#c###$

(c = Inhalt von Akkuband)

klar: M simuliert R

für RAM R logarithmisches Kostenmaß verwendet

⇒ Gesamtlänge aller Zahlen in $O(t(n))$

⇒ Gesamtlänge der Bänder von M in $O(t(n))$

⇒ jeder (von $t(n)$ möglichen) Speicherzugriffen kostet M maximal $O(t(n))$ Schritte

⇒ maximal $O(t(n)^2)$ Schritte für Speicherzugriff.

b) Rechenoperationen:

Anmerkung: Operand steht schon auf Operandenband

ADD, SUB: $O(k_1 + k_2)$, wobei k_1, k_2 die Längen der Längen der beiden Zahlen

MULT, DIV: $O(k_1 \cdot k_2)$, wobei k_1, k_2 die Längen der Längen der beiden Zahlen, Schulmethode

RAM benötigt dafür mindestens $k_1 + k_2$ viel Zeit

$$\sum_{MULT, DIV} k_1 \cdot k_2 \underbrace{\leq}_{(a-b)^2 \geq 0} \sum_{MULT, DIV} k_1^2 + k_2^2 \leq \underbrace{\left(\sum_{MULT, DIV} k_1 + k_2 \right)^2}_{\leq c \cdot t(n)} \in O(t(n)^2)$$

qed.

!Achtung! Satz gilt nicht für Einheitskostenmaß.

Denn: betrachte z. B. $f(n) = n^{(2^n)}$

- braucht zur Darstellung des Resultats mindestens 2^n Felder auf TM
- lässt sich im Einheitskostenmaß auf RAM in Zeit $O(n)$ berechnen: durch sukzessives Quadrieren:

$$n, n^2, n^4, \dots, n^{2^k}, \dots, n^{2^n}$$

4 Kryptographie

Bisher: NP-Vollständigkeit als etwas schlechtes empfunden.

Frage: Kann sie auch nutzen?

Manchmal ja. Beispiel: Verschlüsselung.

4.1 Situation

Bob will an Alice eine geheime Nachricht über ein unsicheres Netz schicken.

4.2 Ansatz:

Verschlüsseln.

2 Möglichkeiten

4.3 Private-Key-Verfahren:

- Bob hat Schlüssel E_B , nur ihm bekannt
- Alice hat Schlüssel D_A , nur ihr bekannt

Um Klartext P zu übermitteln:

Bob sendet $E_B(P)$ an Alice

Alice entschlüsselt $D_A(E_B(P)) = P$, falls $D_A \cdot E_B = id$

! Nachricht im Netz nicht verändert (wird hier nicht betrachtet)

4.3.1 Problem:

Wie kommt Schlüssel D_A zu Alice?

- reitende Boten
- hierarchische Schlüsselsysteme
- Merkle-Verfahren:
 - Bob schickt an Alice eine große Anzahl von Nachrichten $\tilde{E}((D_A^i, i))$
(\tilde{E} ein recht leicht zu brechendes Verschlüsselungsverfahren,
 i Index, $i = 1, \dots, 10000$)
 - Alice wählt zufällig eine Nachricht $\tilde{E}(D_A^i, i)$ aus und bricht sie
schickt an Bob Index i zurück
 - Bob verwendet ab jetzt den passenden Verschlüsselungsschlüssel
 E_B^i
 - Alice verwendet ab jetzt D_A^i

Sicherheit beruht auf folgenden Tatsachen:

- Angreifer kennt nicht die Liste aller Schlüssel D_A^i (nur Bob kennt sie)
- Angreifer hat keine Zeit, alle Nachrichten zu brechen (relative Sicherheit)

Bekanntes Private-Key-Verfahren: DES (Data Encryption Standard) beruht auf Bit-Shuffling

4.4 Public-Key-Verfahren:

Jede Person A erzeugt einen öffentlich bekannten Schlüssel E_A

Jede Person A erzeugt einen privaten Schlüssel D_A

Will Bob an Alice eine Nachricht P schicken, so versendet er $E_A(P)$
Alice wendet darauf ihren Schlüssel D_A an und erhält $D_A(E_A(P)) = P$

4.4.1 prominentestes Beispiel: RSA-Verfahren

(Rivest, Shamir, Adleman) beruht auf der Schwierigkeit, ganze Zahlen in Primzahlen zu zerlegen.

Verwendet ein öffentlich bekanntes $N = p \cdot q$, $p \neq q$ zwei Primzahlen.

Ob eine Zahl N Primzahl ist, lässt sich polynomiell in der Anzahl der Stellen von N testen (Agrawal, Kayal, Saxena, 2002)

Ob Primfaktorzerlegung in P liegt oder NP-vollständig ist, weiß man nicht. (Die Sicherheit von RSA ist nicht klar)

4.4.2 Vorbereitung: Euler'sche ϕ -Funktion

$N \in \mathbb{N}$: $\phi(N)$ = Anzahl der zu N teilerfremden natürlichen Zahlen $< N$

Beispiel $\phi(5) = 4$ denn: $\{1, 2, 3, 4\}$ sind < 5 und teilerfremd.

$\phi(15) = 8$ denn: $\{1, 2, 4, 7, 8, 11, 13, 14\}$ sind < 15 und teilerfremd.

allgemein: $\phi(p) = p - 1$ falls p prim

$\phi(p^r) = (p - 1) \cdot p^{r-1}$ falls p prim

$\phi(a \cdot b) = \phi(a) \cdot \phi(b)$, falls a, b teilerfremd

4.4.3

$\mathbb{Z}/n\mathbb{Z}$ Ring mit Einheitengruppe $(\mathbb{Z}/n\mathbb{Z})^* = \{\bar{a}, a \text{ teilerfremd zu } n\} \stackrel{Def.}{=} \{\bar{a}, \exists b : ab \equiv 1 \text{ mod } n\}$

Beweis:

“ \supseteq “ Angenommen, es gibt b mit $ab \equiv 1 \text{ mod } n \Rightarrow$ es gibt $k : ab = 1 + kn$
Angenommen, p wäre echter gemeinsamer Teiler von $ab \text{ mod } n \Rightarrow p | ab - kn = 1$

“ \subseteq “ Angenommen, a und n sind teilerfremd $\Rightarrow \text{ggT}(a, n) = 1$

$$\begin{aligned} a, n \text{ teilerfremd} &\Rightarrow \text{ggT}(a, n) = 1 \\ &\Rightarrow \exists v, w \in \mathbb{Z} : 1 = va + wn \\ &\stackrel{\text{Lemma}(\text{ggT})}{\Rightarrow} a \equiv 1 \text{ mod } n \end{aligned}$$

$\Rightarrow (\mathbb{Z}/n\mathbb{Z})^*$ hat $\phi(n)$ Elemente.

4.4.4 Sätzchen (Kleiner Fermat)

G Gruppe der Ordnung k

$$\Rightarrow \forall g \in G : \underbrace{g \cdot g \cdot g \cdots g}_{k\text{-mal}} = g^k = 1$$

Beweis: für abelsche Gruppen: Betrachte die Abbildung

$$\begin{aligned} f: G &\rightarrow G \\ h &\mapsto h \cdot g \end{aligned}$$

g fest, ist bijektiv

Sei $x = g_1 \cdot g_2 \cdot \dots \cdot g_k$ das Produkt aller Gruppenelemente (auf Reihenfolge kommt es nicht an)

$$f(g_1) \cdot f(g_2) \cdot f(g_3) \cdot \dots \cdot f(g_k) = g_1 \cdot g_2 \cdot \dots \cdot g_k = x$$

\Updownarrow

$$g_1 \cdot g \cdot g_2 \cdot g \cdot \dots \cdot g_k \cdot g = g_1 \cdot g_2 \cdot \dots \cdot g_k \cdot g^k = x \cdot g^k \Rightarrow g^k = 1$$

f permutiert die Elemente von G

qed

4.4.5 RSA-Verfahren:

öffentlich: $E_A : (N, e)$ privat: $D_A : (N, d)$

wobei $N = \underbrace{p \cdot q}_{\text{Zerlegung nicht öffentlich}}$ mit $p \neq q$ prim, etwa gleich groß

$$e < \phi(N), \text{ teilerfremd zu } \phi(N)$$

$$d < \phi(N) \text{ und } ed \equiv 1 \pmod{\phi(N)}$$

Angenommen Bob will an Alice Nachricht x schicken, wobei $x \in \mathbb{N}$ und $x < N^{23}$

Bob sendet

$$x^e \pmod{N} =: y \in \{0, \dots, N-1\}$$

Alice entschlüsselt

$$y^d \pmod{N} =: x^* \in \{0, \dots, N-1\}$$

$$x^* = x$$

$$\text{Beweis: } ed \equiv 1 \pmod{\phi(N)} \stackrel{\text{Def} \equiv}{\Rightarrow} \exists k \in \mathbb{Z} : \begin{aligned} ed &= 1 + k \cdot \phi(N) \\ &= 1 + k \cdot (p-1)(q-1), \text{ da } N = p \cdot q \end{aligned}$$

1. Fall: x, p teilerfremd $\stackrel{\text{Kleiner Fermat}}{\Rightarrow} x^{p-1} \equiv 1 \pmod{p}$

$$\Rightarrow x^{k(p-1)(q-1)} \equiv 1 \pmod{p}$$

$$\Rightarrow x^{ed} = x^{1+k(p-1)(q-1)} \equiv x \pmod{p}$$

²³Keine Einschränkung, denn jede Nachricht P lässt sich als Folge solcher x schreiben

2. Fall: x, p nicht teilerfremd $\stackrel{prim}{\Rightarrow} p|x$

$$\Rightarrow x \equiv 0 \pmod{p}$$

$$\Rightarrow x^{ed} \equiv 0 \pmod{p}$$

$$\Rightarrow x^{ed} \equiv x \pmod{p}$$

Genauso: $x^{ed} \equiv x \pmod{q}$

Zusammen: $p|x^{ed} - x$ und $q|x^{ed} - x \stackrel{p \neq q}{\Rightarrow} \underbrace{pq}_N |x^{ed} - x$

$$\Rightarrow x^* = \underbrace{x^{ed}}_{y^d} \equiv x \pmod{N}$$

und $0 \leq x, x^* \leq N - 1 \Rightarrow x = x^*$

qed

Bemerkungen

1. Zu gegebenen e berechnet man d durch Anwendung des Euklidischen Algorithmus auf e und $\phi(N)$.
2. Das setzt Kenntnis vom $\phi(N)$ voraus.
3. Dies ist äquivalent zu Kenntnis der Zerlegung $N = p \cdot q$.

Beweis $\phi(N)$ aus Zerlegung $N = p \cdot q$ einfach: $\phi(N) = (p-1)(q-1)$
 p aus Kenntnis von $N, \phi(N)$:

$$\phi(N) = (p-1)(q-1) = pq - (p+q) + 1 = N - (p + \frac{N}{p}) + 1$$

$$\Rightarrow p + \frac{N}{p} = p + q = N + 1 - \phi(N)$$

$$\Rightarrow p^2 - (N + 1 - \phi(N))p + N = 0$$

$$\Rightarrow p = \frac{N+1-\phi(N) \pm \sqrt{(N+1-\phi(N))^2 - 4N}}{2} \quad (\text{quadratische Gleichung})$$

4. Berechnung der Potenzen $x^e \pmod{N}, y^d \pmod{N}$ durch fortgesetztes Quadrieren und sofortige Reduktion \pmod{N} :

$$x^e \equiv (x^{\frac{e}{2}} \pmod{N}) \cdot (x^{\frac{e}{2}} \pmod{N}) \pmod{N}, \text{ falls } e \text{ gerade}$$

$$\equiv (x^{\frac{e-1}{2}} \pmod{N}) \cdot (x^{\frac{e-1}{2}} \pmod{N}) (x \pmod{N}), \text{ falls } e \text{ ungerade}$$

Damit Reduktion in insgesamt $(\log x)^2$ Schritten in logarithmischem Kostenmaß.

4.5 Mathematische Grundlagen

4.5.1 Gruppe

Menge G mit Verknüpfung

$$\begin{aligned} \cdot : G \times G &\rightarrow G \\ (g, h) &\mapsto g \cdot h \end{aligned}$$

4.5.2 $\mathbb{Z}/n\mathbb{Z}$ - kommutativer Ring der Äquivalenzklassen, Gruppe bzgl. \oplus und $\bar{\cdot}$, abelsch

$a \sim b \Leftrightarrow n|a - b$ (in Worten: a kongruent zu b modulo n)
ist Äquivalenzrelation, d.h. \sim ist

reflexiv $a \sim a \forall a$,

symmetrisch $a \sim b \Rightarrow b \sim a$,

transitiv $a \sim b, b \sim c \Rightarrow a \sim c$
($n|a - b$ und $n|b - c \Rightarrow n|(a - b) + (b - c) = a - c$)

Zu a bilde $\bar{a} := \{b \in \mathbb{Z}, a \sim b\}$ Äquivalenzklasse von a .

Beispiel:

$$n = 11, a = 3 \Rightarrow \bar{a} = \{3, 3 + 11, 3 + 2 \cdot 11, 3 + 3 \cdot 11, \dots\}$$

Man kann mit den Äquivalenzklassen (fast) wie mit Zahlen rechnen:

$$\bar{a} \oplus \bar{b} := \overline{a + b}$$

$$\bar{a} \ominus \bar{b} := \overline{a - b}$$

$$\bar{a} \odot \bar{b} := \overline{a \cdot b}$$

Menge der Äquivalenzklassen \bar{a} wird mit $\oplus, (\ominus,)\odot$ zum kommutativen Ring $(\mathbb{Z}/n\mathbb{Z})$.

In jedem Ring R kann man die Gruppe der Einheiten betrachten: $R^* = \{r \in R; \exists s \in R : rs = 1_R\}$

Beispiel: $\mathbb{Z}^* = \{1, -1\}$

4.5.3 Euklidischer Algorithmus

am Beispiel 294, 203

$$\begin{array}{rcl}
 294 = & 1 \cdot 203 + & 91 \\
 & \swarrow & \searrow \\
 203 = & 2 \cdot 91 + & 21 \\
 & \swarrow & \searrow \\
 91 = & 4 \cdot 21 + & 7 \\
 & \swarrow & \\
 21 = & 3 \cdot 7 + & 0
 \end{array}$$

allgemeiner Schritt: $q_i = v \cdot q_{i+1} + r_{i+1}$ mit $v \in \mathbb{N}$, $r_{i+1} \in \{0, 1, \dots, q_{i+1}-1\}$
(Rest bei Division)

Klar

$$\begin{aligned}
 r_{i+1} &< q_{i+1} \leq q_i \\
 r_{i+1} &\leq \frac{1}{2} q_i
 \end{aligned}$$

\Rightarrow Euklidischer Algorithmus macht $O(\log n)$ viele Schritte

Behauptung

1. $7 \stackrel{!}{=} \text{ggT}(294, 203)$

- Durch Betrachtung der Gleichungskette rückwärts: 7 ist gemeinsamer Teiler von 294, 203
- Jeder gemeinsame Teiler von 294, 203 teilt 7: durch Betrachtung vorwärts.

$$\begin{aligned}
 7 &= 91 - 4 \cdot 21 \\
 &= 91 - 4 \cdot (203 - 2 \cdot 91) \\
 2. &= 9 \cdot 91 - 4 \cdot 203 \\
 &= 9 \cdot (294 - 203) - 4 \cdot 203 = 9 \cdot 294 - 13 \cdot 203
 \end{aligned}$$

4.5.4 Lemma: (ggT - ganzzahlige Linearkombination)

$\text{ggT}(a,b)$ lässt sich als ganzzahlige Linearkombination von a, b , schreiben.

4.5.5 Kongruenz

$$a \equiv b \pmod{n} \Leftrightarrow n \mid a - b \Leftrightarrow \bar{a} = \bar{b}$$