Grundzüge der Theoretischen Informatik

21.1.22

Markus Bläser

Universität des Saarlandes

Kapitel 25: Eine universelle Turingmaschine

Goedelisierung von Turingmaschinen

- k-Band-TM $M = (Q, \{0,1\}, \Gamma, \delta, q_0, Q_{acc})$
- o.B.d.A $Q = \{0, 1, \ldots, s\}$ und $\Gamma = \{0, 1, \ldots, \ell\}$, ℓ ist das Blank.
- Kodiere q durch $\mathrm{bin}(q)$ und γ durch $\mathrm{bin}(\gamma)$.
- Kodiere $\delta(q, \gamma_1, \ldots, \gamma_k) = (q', \gamma_1', \ldots, \gamma_k', r_1, \ldots, r_k)$ durch

$$[\mathrm{bin}(q), \mathrm{bin}(\gamma_1), \ldots, \mathrm{bin}(\gamma_k), \mathrm{bin}(q'), \mathrm{bin}(\gamma_1'), \ldots, \mathrm{bin}(\gamma_k'), \hat{r}_1, \ldots, \hat{r}_k]$$

wobei

$$\hat{r}_\kappa = \begin{cases} 00 & \text{if } r_\kappa = S \\ 10 & \text{if } r_\kappa = L \\ 01 & \text{if } r_\kappa = R \end{cases}$$

- Falls $\delta(q, \gamma_1, \ldots, \gamma_k)$ undefiniert ist, kodiere dies durch

$$[\mathrm{bin}(q), \mathrm{bin}(\gamma_1), \ldots, \mathrm{bin}(\gamma_k), \mathrm{bin}(s+1), \varepsilon, \ldots, \varepsilon, \varepsilon, \ldots, \varepsilon]$$

Gödelisierung (2)

Injektive Abbildung $\mathrm{göd}_{TM}$ von der Menge aller TM nach $\{0,1\}^*$:

Konkatenation von

- $\mathrm{bin}(k)$, die Anzahl der Bänder
- $\mathrm{bin}(s+1)$, die Größe von Q
- $\mathrm{bin}(\ell+1)$, die Größe von Γ
- die Kodierungen von $\delta(q,\gamma_1,\ldots,\gamma_k)$, $q \in Q$ und $\gamma_1,\ldots,\gamma_k \in \Gamma$, in lexikographischer Ordnung
- $\mathrm{bin}(q_0)$, der Startzustand
- $\mathrm{bin}(|Q_{\mathrm{acc}}|)$, die Anzahl der akzeptierenden Zustände
- $\mathrm{bin}(q)$, $q \in Q_{\mathrm{acc}}$, in aufsteigender Reihenfolge

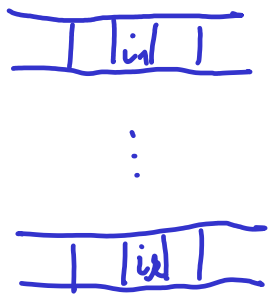
Eigenschaften:

- $\mathrm{göd}_{TM}$ ist injektiv, aber nicht bijektiv.
- das Bild von $\mathrm{göd}_{TM}$ ist entscheidbar (in Polyzeit)

Eine universelle Turingmaschine

- Eine universelle TM $\mathcal{U}_{TM}$ hat eine feste Anzahl von Bänder, ein festes Arbeitsalphabet und eine feste Zustandsmenge.
- Eine zu simulierende TM M kann mehr Bänder, ein großes Arbeitsalphabet und mehr Zustände haben.
- $\mathcal{U}_{TM}$ speichert alle Bänder von M auf einem.
- Wenn die i-te Zelle der k Bänder die Symbole $i_1, \ldots, i_k$ enthalten, dann ist der i-te Block dieses Bandes

$$\$ \# \operatorname{bin}(i_1) \# \operatorname{bin}(i_2) \# \ldots \# \operatorname{bin}(i_k) \$$$



Die Blöcke werden durch $\$$ separiert.

- Zu Beginn werden die Blöcke mit

$$\# \operatorname{bin}(x_j) \# \operatorname{bin}(\ell) \# \ldots \# \operatorname{bin}(\ell)$$

initialisiert (auf Eingabe $x = x_1 x_2 \ldots x_n$).

- Die Kopfpositionen von M werden durch $*$ (statt $\#$) markiert.

Sei M $t(n)$ zeitbeschränkt und $s(n)$ platzbeschränkt.

Dann wird ein Schritt von M in Zeit

$O(|g| \cdot s(n))$ simuliert

Zeitbedarf von U_{TM}: $O(|g| \cdot s(n) \cdot t(n))$

Platzbedarf von U_{TM}: $|g| \cdot s(n)$

| Gödelisierung von M

Kapitel 26: Zeit- und Platzhierarchien

Ein technisches Lemma

Lemma (26.1)

Seien $s_1, s_2, t_1, t_2 : \mathbb{N} \to \mathbb{N}$ *mit* $s_1 = o(s_2)$ *und* $t_1 = o(t_2)$. *Sei* $s_2(n) \geq \log n$ *und* $t_2(n) \geq (1+\epsilon)n$ *für ein* $\epsilon > 0$. *Sei* s_2 *platz- und* t_2*-zeitkonstruierbar.*

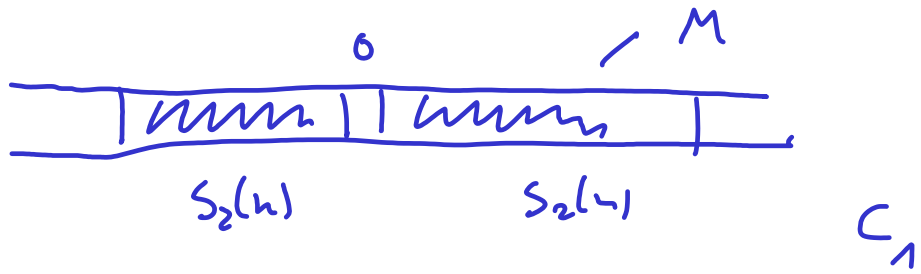
1. *Es gibt eine* s_2*-platzbeschränkte DTM* C_1*, so dass für jede* s_1*-platzbeschränkte Einband DTM* M $L(C_1) \neq L(M)$ *gilt.*
2. *Es gibt eine* t_2*-zeitbeschränkte DTM* C_2*, so dass für jede* t_1*-zeitbeschränkte Einband DTM* M $L(C_2) \neq L(M)$ *gilt.*

Beweis

$* \; bin(i_1) \# bin(i_2)$

- Sei g die Gödelnummer einer Einband-DTM M.
- Wir können M in Zeit $O(|g| \cdot t(n))$ und Platz $O(|g| \cdot s(n))$ simulieren.
  $\setminus$ $s(n)$ fällt nicht.
- Das i-te Symbol von M wird durch $\mathrm{bin}(i)$ (feste Länge) repräsentiert.
- Die Position des Kopfes von M muss nicht gespeichert werden.
- Die Simulation erfolgt Schritt für Schritt.

Konstruktion von C_1



Eingabe: $x \in \{0,1\}^*$, interpretiert als $[g, y]$ mit $g \in \operatorname{im} \text{göd}_{TM}$.

1. Falls x nicht diese Form hat, verwerfe.
2. Markiere $s_2(|x|)$ Symbole links und rechts der Zelle 0 auf dem ersten Band.
3. Simuliere $M := \text{göd}_{TM}^{-1}(g)$ auf x auf dem ersten Band.
4. Zähle die Schritte auf einem Extraband.
5. Falls die Simulation den markierten Platz verlässt, verwerfe.
6. Wenn mehr als $3^{s_2(|x|)}$ Schritte simuliert wurden, dann halte und ~~verwerfe~~. akzeptiere
7. Akzeptiere, falls M verwirft. Sonst verwerfe.

Beh.: Für jede Σ_1-prädikatbeschriebene Einband-DTM M gilt es eine Eingabe $x := [g, y]$, so dass sich C_1 auf $[g, y]$ anders verhält als M.

($g = g\ddot{o}d_{T_M}(M)$)

1. Sei $x \in L(C_1)$

Dann wird die Simulation von M beendet und M verwirft x (7) oder M macht mehr als $3^{Sz(1\times1)}$ viele Schritte. (6)

Im ersten Fall wird wie behauptet.

Für den 2. Fall berechnen wir, dass M auf x maximal ^ machen kann, wenn M hält.

$$c^{s_1(|x|)} \cdot (s_1(|x|)+2)(|x|+2)$$

D.h. wir wird fertig, als $3^{s_2(|x|)}$

$$c^{s_1(|x|)} \cdot (s_1(|x|)+2)(|x|+2)$$

Das ist äquivalent zu

$$\underbrace{(\log 3)}_{1} \cdot s_2(|x|) > \log c \cdot \underbrace{s_1(|x|)}_{\in o(s_2(|x|))} + \underbrace{\log(s_1(|x|+2)}_{\in o(s_1(|x|))} + \underbrace{\log(|x|+2)}_{\in s_2(|x|)}$$

≈ 1.5

Da $s_1 \in o(s_2)$, ist die linke Seite größer, falls x lang genug

2. Fall: $x \notin L(C_1)$

Dann verlässt M den markierten Platz oder M hält und akz.

In zweiter Fall wird wir fertig.

Der erste Fall kann nicht eintreten falls x lang genug ist, denn wenn M s_1-platzbeschränkt ist, benutzt die Simulation lgl. $s_1(|x|) \vee \leq s_2(|x|)$ falls $|x|$ groß genug (Platz).

C_1 ist s_2-platzbeschränkt, da die Eingabe und die Zähler in $s_2(|x|)$ Platz realisiert werden. □

Hierarchiesätze

Theorem (26.2 Deterministischer Platzhierarchiesatz)

Seien $s_2(n) \geq \log n$ *platzkonstruierbar und* $s_1(n) = o(s_2(n))$. *Dann gilt*

$$\mathsf{DSpace}(s_1) \subsetneq \mathsf{DSpace}(s_2).$$

$t_1 \log t_1 = o(t_2)$

Theorem (26.3 Deterministischer Zeithierarchiesatz)

Seien t_2 *zeitkonstruierbar und* $t_1^2 = o(t_2)$. *Dann gilt*

$$\mathsf{DTime}(t_1) \subsetneq \mathsf{DTime}(t_2).$$

Weitere Hierarchiesätze (ohne Beweis)

Theorem (26.6, Fürer)

Seien $k \geq 2$, t_2 *zeitkonstruierbar und* $t_1 = o(t_2)$. *Dann gilt:*

$$\mathsf{DTime}_k(t_1) \subsetneq \mathsf{DTime}_k(t_2).$$

Theorem (26.7, Borodins Lückensatz)

Seien f *eine rekursive Funktion* $\mathbb{N} \to \mathbb{N}$ *mit* $f(n) \geq n$ *für alle* n. *Dann gibt es totale rekursive Funktionen* $s, t : \mathbb{N} \to \mathbb{N}$ *mit* $s(n) \geq n$ *und* $t(n) \geq n$, *so dass*

$$\mathsf{DTime}(f(t(n))) = \mathsf{DTime}(t(n)),$$
$$\mathsf{DSpace}(f(s(n))) = \mathsf{DSpace}(s(n)).$$

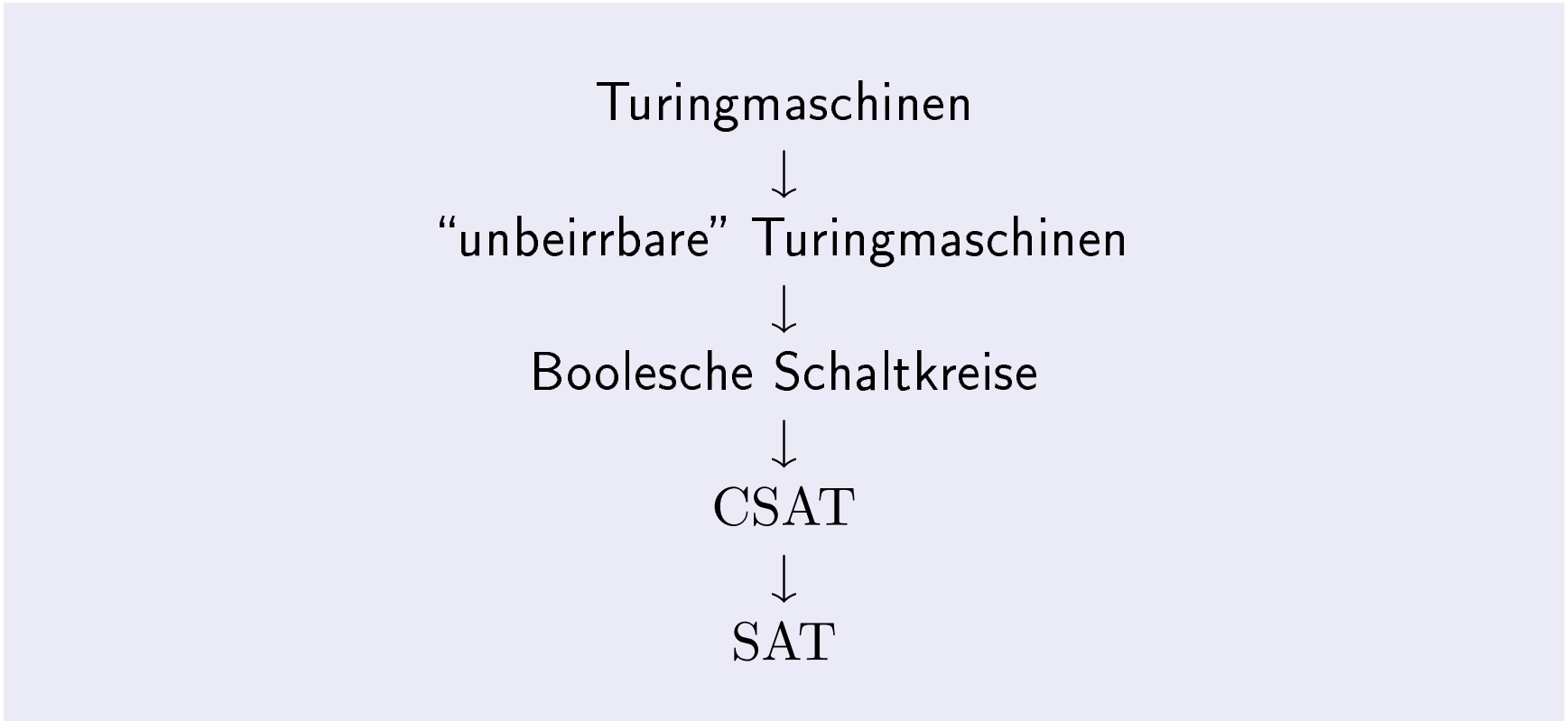
$F(n) = 2^n$ $\quad 2^{2^{2^n}}$ $\quad \mathsf{DTime}(2^{t(n)}) = \mathsf{DTime}(t(n))$

Die Schranke war nicht zeitkonstruierbar...

Kapitel V: Beweis des Cook–Karp–Levin-Theorems (Skizze)

Reduktionsschema

- Gegeben: NTM M und $x \in \Sigma^*$
- Gesucht: Formel $\phi_{M,x}$ in CNF
- Eigenschaft: M akzeptiert $x \iff \phi_{M,x}$ ist erfüllbar



Boolesche Schaltkreise

Unbeirrbare Turingmaschinen

Definition (V.3)

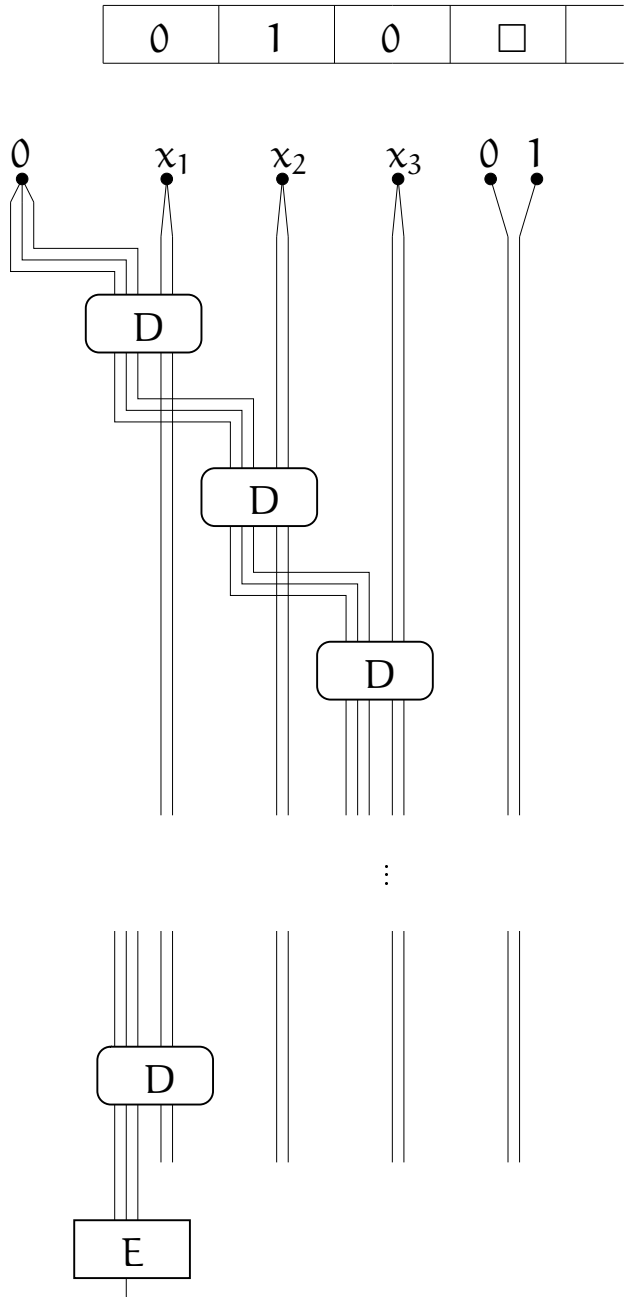
Eine TM heißt *unbeirrbar*, falls die Kopfbewegungen die gleichen sind für alle Eingaben der Länge n. (Insbesondere macht sie die gleiche Anzahl von Schritten.)

Lemma (V.4)

Sei t *zeitkonstruierbar. Für jede* t*-zeitbeschränkte DTM* M *gibt es eine unbeirrbare* $O(t^2)$*-zeitbeschränkte Einband-DTM* S *mit* $L(M) = L(S)$.

Von Turingmaschinen zu Schaltkreisen

- $M = (Q, \Sigma, \Gamma, \delta, q_0, Q_{acc})$, unbeirrbar.
- $Q \subseteq \{0,1\}^d$ für ein festes d, $q_0 = 0 \ldots 0$.
- $\Gamma \subseteq \{0,1\}^c$ für ein festes c.
- $\Sigma = \{0,1\}$, 0 entspricht $0 \ldots 0$ und 1 entspricht $1 \ldots 1$.
- Schaltkreis D
  $D : \{0,1\}^d \times \{0,1\}^c \to \{0,1\}^d \times \{0,1\}^c$
  berechnet δ (ohne Richtung).



Beispiel:

- Eingabe 010
- $Q = \{0,1\}^3$
- $\Gamma = \{0,1\}^2$

Der Beweis

CSAT

Gegeben: (Kodierung eines) Booleschen Schaltkreises C.
Frage: Gibt es ein $\xi \in \{0,1\}^*$ mit $C(\xi) = 1$?

Theorem (V.6)

CSAT *ist* NP-*schwer.*

Lemma

$\mathrm{CSAT} \leq_{\mathrm{P}} \mathrm{SAT}$.

Kapitel 27: Mehr zu NP

NP und co-NP

$P = co\text{-}P$

- co-NP $= \{L \mid \bar{L} \in NP\}$
- Frage: NP $=$ co-NP?

negative Antwort $\Rightarrow$ $P \neq NP$

- UNSAT: Gegeben eine Formel ϕ in CNF, ist ϕ unerfüllbar?
- TAUT: Gegeben ϕ in DNF, erfüllt jede Belegung ϕ?

$\bigvee \ldots (x_i \wedge x_j \wedge x_k)$

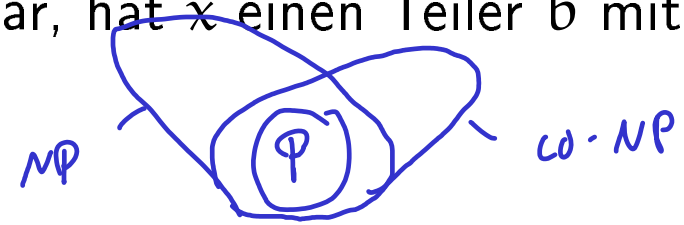
Lemma

TAUT *und* UNSAT *sind* co-NP*-vollständig.*

Theorem

Falls co-NP *ein* NP*-vollständiges Problem enthält, dann ist* NP $=$ co-NP*.*

- FACTOR: Gegeben x und c in binär, hat x einen Teiler b mit $2 \leq b \leq c$?
- FACTOR $\in$ NP $\cap$ co-NP.



Selbstreduzierbarkeit

TSP: Gegeben ein kantengewichteter Graph G, Schranke b

Entscheidungsproblem: Gibt es eine Tour der Länge $\leq$ b?

Berechnungsproblem: Wie lang ist die kürzeste Tour?

Konstruktionsproblem: Gib eine Tour minimaler Länge aus!

Binärsuche

log in oberer Schranke

Beobachtung:

- P = NP gibt einen effizienten Algorithmus für das Entscheidungsproblem.
- Aber: wir wollen das Konstruktionsproblem lösen!

Hilfe! Mein Problem ist NP-schwer

- Exakte Algorithmen mit exponentieller Worst-Case-Laufzeit, aber akzeptabler Laufzeit auf vielen Eingaben, z.B. Sat-Solver
- Heuristiken, die keine optimale Lösung finden, aber eine akzeptable.
- ...

Zwei Methoden, bei denen man etwas beweisen kann:

- Approximationsalgorithmen
- Parametrisierte Algorithmen

Approximationsalgorithmen

Nicht-baren → $2^{p(n)} \cdot n$

$w: \binom{V}{2} \to \mathbb{N}^+$ Kanten → 1

Theorem

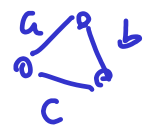
Falls es eine Polynomialzeit-DTM A *gibt, die gegeben* $G = (V, \binom{V}{2}, w)$ *eine Hamiltonsche Tour* H *mit*

$$w(H) < 2^{p(n)} \cdot OPT(G)$$

Länge einer optimalen Ham. Tour

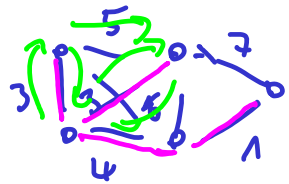
ausgibt für ein Polynom p*, dann gilt* $P = NP$.

Metrisches TSP: w erfüllt die Dreiecksungleichung



$c \leq a+b$

Approximationsalgorithmus:



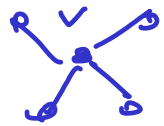
1. Sei T ein MST von G.
2. Ordne die Knoten nach den Besuchszeiten einer DFS.
3. Die Hamiltonsche Tour hat Länge $\leq 2 \cdot OPT(G)$.

1.5

Parametrisierte Algorithmen

Vertex-Cover:

- Jeder Vertex-Cover muss v oder alle Nachbarn $N(v)$ von v enthalten.



Gegeben (G, k), gibt es einen Vertex-Cover der Größe $\leq k$?

Binärer Suchbaum:

1. Falls $k < 0$, gebe nein zurück.
2. Gebe ja zurück, falls der Graph keine Kanten hat.
3. Gebe nein zurück, wenn $k = 0$.
4. Fahre rekursiv fort auf $(G - \{v\}, k - 1)$ und $(G - N(v), k - \deg(v))$.
5. Falls ein rekursiver Aufruf ja ausgibt, gebe ja zurück. Sonst nein.

Laufzeit: $O(2^k \operatorname{poly}(n))$

Für Clique gibt das vermutlich nicht n^k

Starke NP-Härte und pseudopolynomielle Algorithmen

Zahlprobleme: Eingaben sind Tupel von Zahlen

Definition

Ein Zahlproblem ist *stark* NP-*schwer*, falls es NP-schwer ist, wenn die Zahlen unär kodiert werden.

- Subset-Sum und Partition haben pseudopolynomielle Algorithmen
- pseudopolynomiell: Laufzeit polynomiell in der Größe der Zahlen, nicht in der Länge der Binärdarstellung.

Theorem

Falls ein stark NP-*schweres Zahlproblem* L *einen pseudopolynomiellen Algorithmus hat, dann* $\mathrm{P} = \mathrm{NP}$.