

NTM, die RUCKSACK entscheidet

Algorithmus NTM für RUCKSACK

Eingabe: W, P, b, k

- 1 Erzeuge nichtdeterministisch einen Zeugen $I \subseteq [n]$.
 - 2 Falls $\sum_{i \in I} w_i \leq b$ und $\sum_{i \in I} p_i \geq k$, akzeptiere.
 - 3 Sonst lehne ab.
- D.h. NTM erzeugt sich im Gegensatz zum Verifizierer ihren Zeugen I selbst.
 - Laufzeit: Schritt 1: $\mathcal{O}(n)$, Schritt 2: $\mathcal{O}(n \cdot \log(\max\{w_i, p_i, b, k\}))$.
 - D.h. die Laufzeit ist polynomiell in der Eingabelänge.

\mathcal{NP} mittels NTMs

Satz

\mathcal{NP} ist die Klasse aller Sprachen, die von einer NTM in polynomieller Laufzeit entschieden wird, d.h.

$$\mathcal{NP} = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k).$$

Zeigen:

- \exists polynomieller Verifizierer für L
- $\Leftrightarrow \exists$ NTM N , die L in polynomieller Laufzeit entscheidet.

Verifizierer \Rightarrow NTM

" \Rightarrow ": Sei V ein Verifizierer für L mit Laufzeit $\mathcal{O}(n^k)$ für ein festes k .

Algorithmus NTM N für L

Eingabe: w mit $|w| = n$.

- 1 Erzeuge nicht-deterministisch einen Zeugen c mit $|c| = \mathcal{O}(n^k)$.
- 2 Simuliere V mit Eingabe (w, c) .
- 3 Falls V akzeptiert, akzeptiere. Sonst lehne ab.

- Korrektheit:
 $w \in L \Leftrightarrow \exists c \text{ mit } |c| = \mathcal{O}(n^k) : V \text{ akzeptiert } (w, c) \text{ in Zeit } \mathcal{O}(n^{k'})$.
 $\Leftrightarrow N \text{ akzeptiert die Eingabe } w \text{ in Laufzeit } \mathcal{O}(n^{\max\{k, k'\}})$.
- Damit entscheidet N die Sprache L in polynomieller Laufzeit.

NTM \Rightarrow Verifizierer

" \Leftarrow ": Sei N eine NTM, die L in Laufzeit $\mathcal{O}(n^k)$ entscheidet.

Algorithmus Verifizierer

Eingabe: w, c

- 1 c ist Kodierung eines Berechnungspfades von N bei Eingabe w .
- 2 Simuliere N auf Eingabe w auf dem Berechnungspfad c .
- 3 Falls N akzeptiert, akzeptiere. Sonst lehne ab.

Korrektheit:

$$\begin{aligned} w \in L &\Leftrightarrow \exists \text{ akzeptierender Berechnungspfad } c \text{ von } N \text{ f\u00fcr } w \\ &\Leftrightarrow V \text{ akzeptiert } (w, c). \end{aligned}$$

Laufzeit:

- L\u00e4ngster Berechnungspfad von N besitzt L\u00e4nge $\mathcal{O}(n^k)$.
- D.h. die Gesamtlaufzeit von V ist ebenfalls $\mathcal{O}(n^k)$.

Boolesche Formeln

Definition Boolesche Formel

- Eine Boolesche Variable x_i kann Werte aus $\{0, 1\}$ annehmen, wobei $0 \cong$ falsch und $1 \cong$ wahr.
- Jede Boolesche Variable x_i ist eine Boolesche Formel.
- Sind ϕ, ϕ' Boolesche Formeln, so auch $\neg\phi, \phi \wedge \phi', \phi \vee \phi', (\phi)$.
- Wir ordnen die Operatoren nach absteigender Priorität: $()$, \neg , \wedge , \vee .
- ϕ ist erfüllbar $\Leftrightarrow \exists$ Belegung der Variablen in ϕ , so dass $\phi = 1$.

Bsp:

- $\phi = \neg(x_1 \vee x_2) \wedge x_3$ ist erfüllbar mit $(x_1, x_2, x_3) = (0, 0, 1)$.
- $\phi' = x_1 \wedge \neg x_1$ ist eine nicht-erfüllbare Boolesche Formel.

Satisfiability SAT

Definition SAT

$\text{SAT} := \{\phi \mid \phi \text{ ist eine erfüllbare Boolesche Formel.}\}$

Kodierung von ϕ :

- Kodieren Variable x_i durch $\text{bin}(i)$.
- Kodieren ϕ über dem Alphabet $\{0, 1, (,), \neg, \wedge, \vee\}$.

SAT ist polynomiell verifizierbar.

Satz

$\text{SAT} \in \mathcal{NP}$.

Beweis

Algorithmus Polynomieller Verifizierer

EINGABE: $(\phi(x_1, \dots, x_n), \mathbf{c})$, wobei $\mathbf{c} = (c_1, \dots, c_n) \in \{0, 1\}^n$.

- Falls $\phi(c_1, \dots, c_n) = 1$, akzeptiere. Sonst lehne ab.

Korrektheit:

- $\phi(x_1, \dots, x_n) \in \text{SAT} \Leftrightarrow \exists$ Belegung $\mathbf{c} \in \{0, 1\}^n : \phi(\mathbf{c}) = 1$

Laufzeit:

- Belegung von ϕ mit \mathbf{c} : $\mathcal{O}(|\phi|)$ auf RAM.
- Auswertung von ϕ auf \mathbf{c} : $\mathcal{O}(|\phi|^2)$ auf RAM.

Konjunktive Normalform

Definition Konjunktive Normalform (KNF)

Seien x_1, \dots, x_n Boolesche Variablen und ϕ eine Boolesche Formel.

- *Literale* sind Ausdrücke der Form x_i und $\neg x_i$.
- *Klauseln* sind disjunktive Verknüpfungen von Literalen.
- ϕ ist in KNF, falls ϕ eine Konjunktion von Klauseln ist.
- Eine KNF Formel ϕ ist in 3-KNF, falls jede Klausel genau 3 Literale enthält.

Bsp:

- $\neg x_1 \vee x_2$ und x_3 sind Klauseln.
- $(\neg x_1 \vee x_2) \wedge x_3$ ist in KNF.
- $(\neg x_1 \vee x_2 \vee x_2) \wedge (x_3 \vee x_3 \vee x_3)$ ist in 3-KNF.

Die Sprache 3-SAT

Definition 3SAT

3SAT := $\{\phi \mid \phi \text{ ist eine erfüllbare 3-KNF Boolesche Formel.}\}$

- Offenbar gilt $3\text{SAT} \subset \text{SAT}$.

Satz

$3\text{SAT} \in \mathcal{NP}$.

Beweis

Algorithmus NTM für 3SAT

Eingabe: $\phi(x_1, \dots, x_n) \in 3\text{-KNF}$

- 1 Rate nicht-deterministisch eine Belegung $(c_1, \dots, c_n) \in \{0, 1\}^n$.
- 2 Falls $\phi(c_1, \dots, c_n) = 1$, akzeptiere. Sonst lehne ab.

- Laufzeit Schritt 1: $\mathcal{O}(n) = \mathcal{O}(|\phi|)$, Schritt 2: $\mathcal{O}(|\phi|)$.
- D.h. die Laufzeit ist polynomiell in der Eingabelänge $|\phi|$.

Simulation von NTMs durch DTMs

Satz Simulation von NTM durch DTM

Sei N eine NTM, die die Sprache L in Laufzeit $t(n)$ entscheidet. Dann gibt es eine DTM M , die L in Zeit $\mathcal{O}(2^{t(n)})$ entscheidet.

Sei $B(w) = (V, E)$ der Berechnungsbaum von N bei Eingabe w .

Algorithmus DTM M für L

- 1 Führe Tiefensuche auf $B(w)$ aus.
 - 2 Falls akzeptierender Berechnungspfad gefunden wird, akzeptiere.
 - 3 Sonst lehne ab.
- Tiefensuche auf $B(w)$ benötigt Laufzeit $\mathcal{O}(|V| + |E|) = \mathcal{O}(|V|)$.
 - Berechnungspfade in $B(w)$ besitzen höchstens Länge $t(n)$.
 - D.h. $B(w)$ besitzt höchstens $2^{t(n)}$ Blätter.
 - Damit besitzt $B(w)$ höchstens $|V| \leq 2 \cdot 2^{t(n)} - 1$ viele Knoten.
 - D.h. die Gesamtlaufzeit ist $\mathcal{O}(2^{t(n)})$.

Polynomielle Reduktion

Definition Polynomiell berechenbare Funktion

Sei Σ ein Alphabet und $f : \Sigma^* \rightarrow \Sigma^*$. Eine Funktion f heißt polynomiell berechenbar gdw. eine DTM M existiert, die für jede Eingabe w in Zeit polynomiell in $|w|$ den Wert $f(w)$ berechnet.

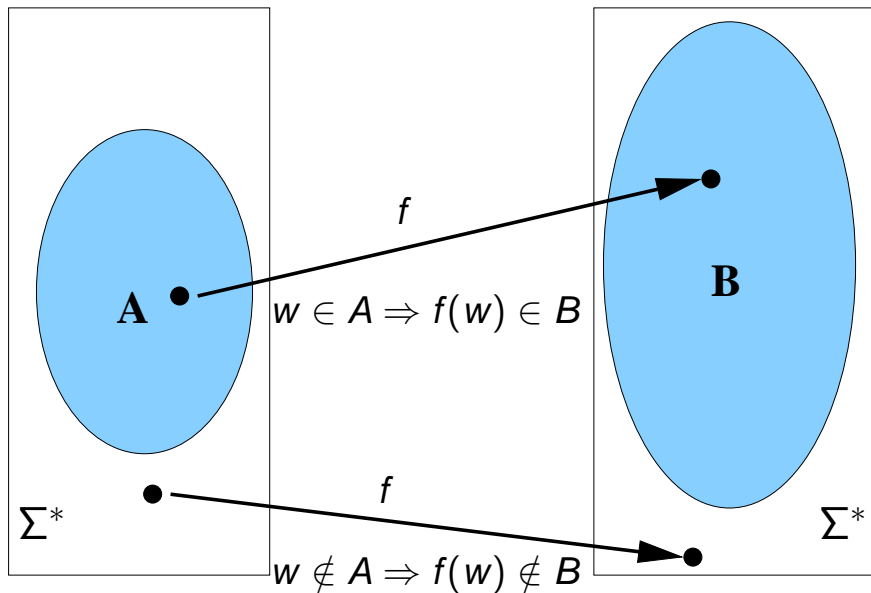
Definition Polynomielle Reduktion

Seien $A, B \subseteq \Sigma^*$ Sprachen. A heißt *polynomiell reduzierbar auf B* , falls eine polynomiell berechenbare Funktion $f : \Sigma^* \rightarrow \Sigma^*$ existiert mit

$$w \in A \Leftrightarrow f(w) \in B \quad \text{für alle } w \in \Sigma^*.$$

Wir schreiben $A \leq_p B$ und bezeichnen f als *polynomielle Reduktion*.

Graphische Darstellung $w \in A \Leftrightarrow f(w) \in B$



A ist nicht schwerer als B.

Satz \mathcal{P} -Reduktionssatz

Sei $A \leq_p B$ und $B \in \mathcal{P}$. Dann gilt $A \in \mathcal{P}$.

- Wegen $B \in \mathcal{P}$ existiert DTM M_B , die B in polyn. Zeit entscheidet.
- Wegen $A \leq_p B$ existiert DTM M_f , die f in polyn. Zeit berechnet.

Algorithmus DTM M_A für A

Eingabe: w

- 1 Berechne $f(w)$ mittels M_f auf Eingabe w .
- 2 Falls M_B auf Eingabe $f(w)$ akzeptiert, akzeptiere. Sonst lehne ab.

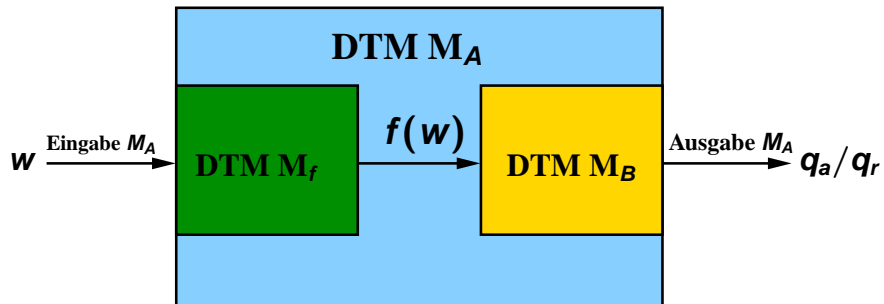
Korrektheit:

- M_A akzeptiert $w \Leftrightarrow M_B$ akzeptiert $f(w) \Leftrightarrow f(w) \in B \Leftrightarrow w \in A$.

Laufzeit:

- $T(M_A) = \mathcal{O}(T(M_f) + T(M_B))$, d.h. polynomiell in $|w|$.

Graphische Darstellung des Reduktionsbeweises



Transitivität polynomieller Reduktionen

Satz Transitivität von \leq_p

Seien $A, B, C \subseteq \Sigma^*$ Sprachen mit $A \leq_p B$ und $B \leq_p C$. Dann gilt $A \leq_p C$.

- Sei f die polynomielle Reduktion von A auf B , d.h.
$$w \in A \Leftrightarrow f(w) \in B \quad \text{für alle } w \in \Sigma^*.$$
- Sei g die polynomielle Reduktion von B auf C , d.h.
$$v \in B \Leftrightarrow g(v) \in C \quad \text{für alle } v \in \Sigma^*.$$
- Dann gilt insbesondere $w \in A \Leftrightarrow f(w) \in B \Leftrightarrow g(f(w)) \in C$.
- Damit ist die Komposition $g \circ f$ eine Reduktion von A auf C .
- $g \circ f$ kann in polynomieller Zeit berechnet werden durch Hintereinanderschaltung der polynomiellen DTMs für f und g .

Clique

Definition Clique

Sei $G = (V, E)$ ein ungerichteter Graph. $C \subseteq V$, $|C| = k$ heißt k -Clique in G , falls je zwei Knoten in C durch eine Kante verbunden sind.

$$\text{CLIQUE} := \{(G, k) \mid G \text{ enthält eine } k\text{-Clique.}\}$$

Satz

$3\text{SAT} \leq_p \text{CLIQUE}$

Zu zeigen: Es gibt eine Reduktion f mit

- 1 f ist eine polynomiell berechenbare Funktion
- 2 $\phi \in 3\text{SAT} \Leftrightarrow f(\phi) = (G, k) \in \text{CLIQUE}$

Idee für die Reduktion: Konstruiere (G, k) derart, dass

- ϕ erfüllbar $\Leftrightarrow \exists$ erfüllende Belegung B für ϕ .
- $\Leftrightarrow B$ setzt in jeder der n Klauseln mind. ein Literal wahr.
- \Leftrightarrow Wahre Literale entsprechen einer n -Clique in G .

Die Reduktion f

Algorithmus M_f für f

Eingabe: $\phi = (a_{11} \vee a_{12} \vee a_{13}) \wedge \dots \wedge (a_{n1} \vee a_{n2} \vee a_{n3})$

- 1 Wahl der Knotenmenge V von G
 - ▶ Definiere $3n$ Knoten mit Labeln a_{i1}, a_{i2}, a_{i3} für $i = 1, \dots, n$.
- 2 Wahl der Kantenmenge E : Setze Kante $(u, v) \in E$ **außer** wenn
 - ▶ u, v entsprechen Literalen derselben Klausel, denn die Clique soll aus Literalen verschiedener Klauseln bestehen.
 - ▶ Label von u ist Literal x und Label von v ist $\neg x$, denn x soll nicht gleichzeitig auf wahr und falsch gesetzt werden (Konsistenz).
- 3 Wahl von k .
 - ▶ Setze $k = n = \frac{|V|}{3}$, denn alle Klauseln sollen erfüllt werden.

Ausgabe: (G, k)

zu zeigen: f ist polynomiell berechenbar.

- Laufzeit Schritt 1: $\mathcal{O}(n)$, Schritt 2: $\mathcal{O}(n^2)$, Schritt 3: $\mathcal{O}(1)$.
- Gesamtlaufzeit $\mathcal{O}(n^2)$ ist polynomiell in der Eingabelänge.

Korrektheit der Reduktion

Zeigen zunächst: $\phi \in 3\text{SAT} \Rightarrow f(\phi) = (G, k) \in \text{CLIQUE}$

- Sei $\phi \in 3\text{SAT}$. Dann besitzt ϕ eine erfüllende Belegung B .
- Damit setzt B in jeder Klausel $(a_{i1} \vee a_{i2} \vee a_{i3})$, $i = 1, \dots, n$ mindestens ein Literal $a_{i\ell_i}$, $\ell_i \in [3]$ auf wahr.
- Die n Knoten mit Label $a_{i\ell_i}$ in G sind paarweise verbunden, da
 - ▶ die Literale $a_{i\ell_i}$ aus verschiedenen Klauseln stammen.
 - ▶ B ist eine konsistente Belegung, d.h. dass die Literale $a_{i\ell_i}$ von B alle konsistent auf wahr gesetzt werden.
- Die n Knoten mit Label $a_{i\ell_i}$ bilden eine n -Clique in G .
- D.h. $f(\phi) = (G, n) \in \text{CLIQUE}$

Korrektheit von f : Rückrichtung

Zeigen: $f(\phi) = (G, n) \in \text{CLIQUE} \Rightarrow \phi \in 3\text{SAT}$

- Sei $f(\phi) = (G, n) \in \text{CLIQUE}$. Dann besitzt G eine n -Clique v_1, \dots, v_n .
- Nach Konstruktion der Kantenmenge von E gilt:
 - 1 v_1, \dots, v_n korrespondieren zu Variablen in verschiedenen Klauseln.
 - 2 $\nexists v_i, v_j$ mit Labeln x und $\neg x$.
- Sei B diejenige Belegung, die die Label von v_1, \dots, v_n wahr setzt.
 - 1 B setzt in jeder Klausel ein Literal v_i auf wahr.
 - 2 B ist eine konsistente Belegung.
- Damit ist B eine erfüllende Belegung für ϕ .
- D.h. $\phi \in 3\text{SAT}$.