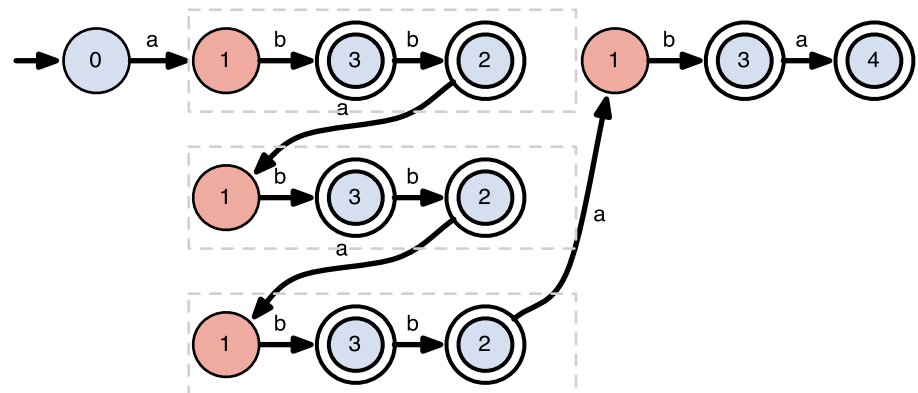


Berechenbarkeitstheorie

20. Vorlesung



Dr. Franziska Jahnke

Institut für Mathematische Logik und Grundlagenforschung

WWU Münster

Laufzeit einer Turingmaschine

- Laufzeit von M hängt von der Eingabe w ab

$$T(w \in \Sigma^*) := \# \text{ Schritte für Lauf von } M(w)$$

(det. 1-Band TM)

- Laufzeit wird meist in Bezug zur Eingabelänge gemessen

$$t(n \in \mathbf{N}) := \max_{w \in \Sigma^n} T(w)$$

- genaue Angabe der Laufzeit oft schwierig (und unnötig)
- wir greifen auf asymptotische Schranken zurück
($\rightarrow O, \Omega, \Theta$ -Notation)

Definition

Sei $f: \mathbf{N} \rightarrow \mathbf{N}$, dann bezeichnet

$$\text{TIME}(f(n)) = \{L \mid \exists \text{TM } M \text{ mit } t_M(n) = O(f(n))\}$$


die **Zeitkomplexitätsklasse** von $f(n)$.

Definition

Die Komplexitätsklasse P bezeichnet alle Probleme, die in Polynomialzeit von einer deterministischen TM entschieden werden können. Das heißt:

$$P := \bigcup_{k=1}^{\infty} \text{TIME}(n^k)$$

- Bsp.: $\{0^k 1^k \mid k \geq 0\} \in P$
- **Anmerkung** Simulation eines TM-äquivalenten Berechnungsmodells durch ein anderes kostet i.A. einen polynomiellen Faktor
→ Beschreibung von P relativ unabhängig vom Berechnungsmodell
- **Konvention:** Probleme aus P sind effizient lösbar
- Probleme müssen sinnvoll kodiert sein
 - $\langle k \rangle = \text{bin}(k) \leftrightarrow \langle k \rangle = 1^k$
 - exponentieller Unterschied der Länge
 - Laufzeit wird relativ zur Eingabelänge gemessen!

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache L** ist eine TM für die gilt:
 $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle \text{ wird von } V \text{ akzeptiert}\}$


z heißt **Zeuge** oder **Zertifikat** Laufzeit von V wird bzgl. $\langle w, z \rangle$ gemessen
- **Def.:** Ein Verifizierer V für L heißt **polynomiell**, gdw.
 - (1) $\exists k$, sodass $L = \{w \mid \exists z \in \Sigma^* : |z| \leq |w|^k \text{ und } \langle w, z \rangle \in L(V)\}$
 - (2) V hat polynomielle Laufzeit

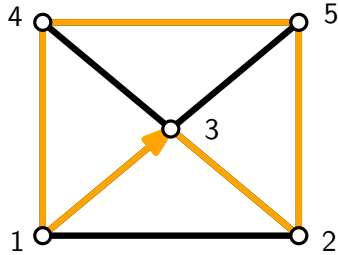
Definition

Die Komplexitätsklasse NP umfasst alle Sprachen, für die es einen polynomiellen Verifizierer gibt.

- Wenn L aus P dann ist der polynomielle Entscheider für L auch ein polynomieller Verifizierer für L (er ignoriert den Zeugen)
- Es folgt: $P \subseteq NP$

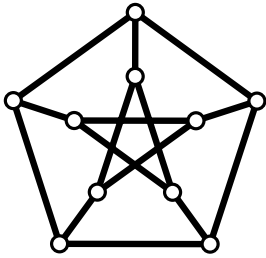
- **Bsp.:** $HC = \{ \langle G \rangle \mid G \text{ ist Graph mit Hamiltonkreis} \}$
- Hamiltonkreis ist ein Kreis, der jeden Knoten des Graphen genau 1x besucht.

Bsp.1

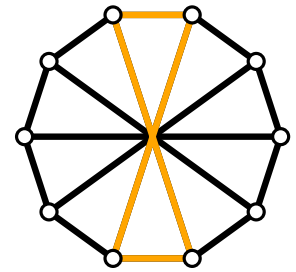


- Graph hat einen Hamiltonkreis
Zeuge: $(3,2,5,4,1)$

Bsp.2



- Graph hat keinen Kreis der Länge ≤ 4
- Angenommen es gibt Hamiltonkreis
- nicht-kreuzende "Diagonalen" erzeugen Kreis der Länge ≤ 4
- verbleibenden Kanten kreuzen paarweise
- einzige Möglichkeit hat Kreis der Länge 4



- $HC \in NP$
- Zeuge: Permutation der Knoten die Hamiltonkreis beschreibt
- Verifikation in polynomieller Zeit möglich, Größe des Zeugen ist Polynom

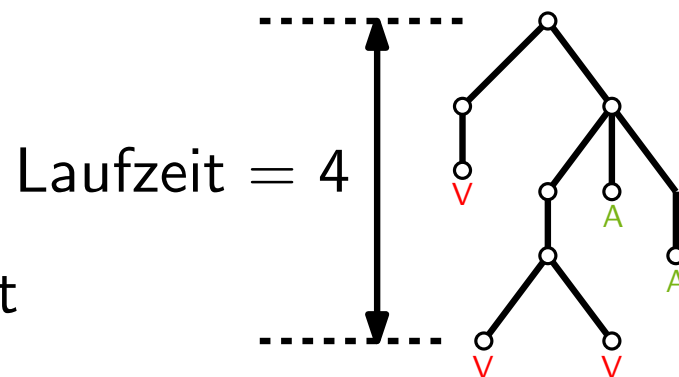
- Verifizierer V für Hamiltonkreis

$V(\langle G, z \rangle)$

1. Prüfe ob G eine korrekte Kodierung eines Graphen ist
2. Prüfe ob z eine Permutation der Knoten von G ist
3. - Bearbeite Knoten in der zyklischen Reihenfolge der Permutation
 - Sind je zwei aufeinander folgende Knoten durch eine Kante verbunden – akzeptiere
 - Ansonsten verwerfe

- V ist polynomieller Verifizierer, was belegt das $HC \in NP$
- Anmerkung: Gibt es für folgende Sprache einen Verifizierer?
 $\overline{HC} = \{\langle G \rangle \mid G \text{ hat keinen Hamiltonkreis}\}$
- Im Beispiel war es aufwendiger zu bezeugen, dass es keinen Hamiltonkreis gibt
- Existenz eines polynomiellen Verifizierers zu \overline{HC} ist ein offenes Problem

- Definition der Laufzeit einer nichtdeterministischen Turingmaschine
- Für NTM M und Eingabe w existiert ein Berechnungsbaum
- Laufzeit von $M(w)$ ist Höhe des Berechnungsbaumes von $M(w)$
- Es ist egal ob, das Blatt zu einer akz. oder verw. Konfiguration gehört
- $T(w) = \text{Laufzeit von } M(w)$
- $t(n) = \max_{w \in \Sigma^n} T(w)$



Definition

Sei $f: \mathbb{N} \rightarrow \mathbb{N}$, dann bezeichnet

$$\text{NTIME}(f(n)) = \{L \mid \exists \text{NTM } N \text{ mit } t_N(n) = O(f(n))\}$$

die **nichtdeterministische Zeitkomplexitätsklasse** von $f(n)$.

Satz 37

$$\text{NP} = \bigcup_{i=1}^{\infty} \text{NTIME}(n^k)$$

Beweis

\Rightarrow -Richtung: $L \in \text{NP} \Rightarrow L \in \text{NTIME}(n^k)$ für ein k

- wenn L aus NP, dann existiert ein polynomieller Verifizierer V für L (d.h. V benutzt Zeugen der Länge $\leq |w|^k$, für ein $k \in \mathbf{N}$)
- betrachte die NTM $N(w)$

$N(w)$

1. Erzeuge ein beliebiges Wort $z \in \Sigma^*$ mit Länge $\leq n^k$
2. Simuliere $V(\langle w, z \rangle)$ und gebe das Ergebnis aus

- Laufzeit von N ist polynomiell
- wir kennen nicht unbedingt das k , aber ein k existiert, und somit existiert die entsprechende NTM N

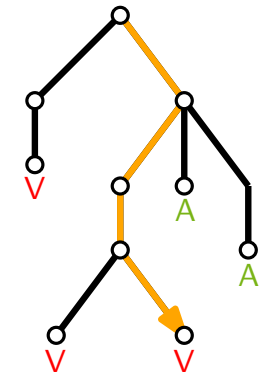
$$w \in L \iff \exists z: (|z| \leq |w|^k): V(\langle w, z \rangle) \text{ akz.} \iff N(w) \text{ akz.}$$

⇐-**Richtung:** $L \in \text{NTIME}(n^k)$ für ein $k \Rightarrow L \in \text{NP}$

- Sei N die NTM, die L in nichtdet. Polynomialzeit entscheidet
- $w \in L$, gdw. es gibt akz. Konfiguration im Berechnungsbaum zu $N(w)$
- Zeuge kodiert die Adresse eines Knotens im Berechnungsbaum
- $z = a_1 \# a_2 \# \dots \# a_k$, und a_i ist die Nummer der Alternative der i ten Folgekonfiguration

$V(\langle w, z \rangle)$

1. Suche im Berechnungsbaum Blatt mit Adresse z , falls es existiert
2. Akzeptiere, gdw. gefundenes Blatt akz. Konfiguration



Adr.: 2#1#1#2

- $V(\langle w, z \rangle)$ arbeitet in Polynomialzeit
- $w \in L \iff \exists$ akz. Berechnungspfad der Länge $\leq |w|^k$
 $\iff \exists z (|z| \leq |w|^k) : V(\langle w, z \rangle)$ akz.
- V ist polynomieller Verifizierer für L

□

SAT (Erfüllbarkeit)

Eingabe: Boolesche Formel ϕ in den Variablen x_i

Frage: Gibt es eine ϕ erfüllende Belegung der Variablen x_i ?

Bsp. $(x_1 \wedge x_2) \vee (x_3 \vee \neg x_2) \vee \neg x_3$

→ erfüllbar mit z.B. $x_1 = 1, x_2 = 1$ und $x_3 = 1$

$x_1 \wedge (x_2 \vee \neg x_1) \wedge (\neg x_2 \vee \neg x_1)$

→ nicht erfüllbar (Wahrheitstabelle ausfüllen)

- Zeuge ist der Vektor der Variablenbelegung

Verifizierer zu SAT

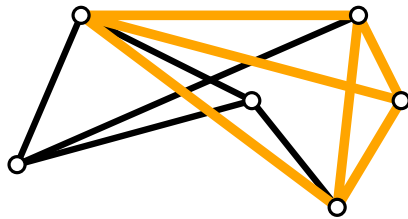
1. Ersetze alle Variablen durch ihre Belegung
2. Suche eine atomare Verbindung mit \wedge, \vee, \neg und ersetze sie (zum Beispiel ersetze $1 \vee 0$ durch 1)
3. Wiederhole bis Ausdruck ausgewertet und akzeptiere, wenn Ergebnis 1

CLIQUE

Eingabe: Graph G und natürliche Zahl k

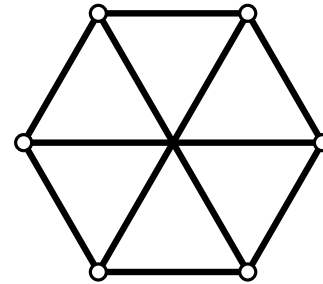
Frage: Hat G eine k -Clique (d.h., K_k als Teilgraph)?

Bsp.



$k = 4$

Graph besitzt 4-Clique



$k = 3$

Graph besitzt keine 3-Clique

- Zeuge ist Auswahl der Knoten der Clique

Verifizierer zu CLIQUE

1. Prüfe für jedes Paar von ausgewählten Knoten (des Zeugen), ob zwischen diesen Knoten eine Kante existiert
2. Ist jedes solche Paar durch eine Kante verbunden, akzeptiere

SUBSET-SUM

Eingabe: Multimenge $S = \{X_1, X_2, \dots, X_n\}$, $X_i \in \mathbf{N}$ und $B \in \mathbf{N}$

Frage: Existiert $S' \subseteq S$ mit $\sum_{X \in S'} X = B$

Bsp.

$S = \{2, 3, 4, 8, 19\}$ $B = 25$ $2 + 4 + 19 = 25$, also ja

$S = \{1, 1, 4, 8, 19\}$ $B = 30$ nicht möglich

- Zeuge ist Auswahl S'

Verifizierer zu SUBSET-SUM

1. Prüfe ob $S' \subseteq S$
2. Prüfe, ob $\sum_{X \in S'} X = B$
3. Wenn beides erfolgreich, akzeptiere

Satz 38

$$\text{NP} \subseteq \text{EXPTIME} = \bigcup_{k=1}^{\infty} \text{TIME} \left(2^{(n^k)} \right)$$

Beweis

- wir müssen zeigen, dass jede Sprache aus NP in $\text{TIME}(2^{(n^k)})$, für ein $k \in \mathbf{N}$
- sei $L \in \text{NP}$ mit polynomiellen Verifizierer V
- wir konstruieren Entscheider M für L wie folgt

$$M(w)$$

1. Zähle alle möglichen Zeugen auf (d.h. alle Worte aus Σ^*)
 \rightarrow für jeden Zeugen z simuliere $V(\langle w, z \rangle)$
2. akzeptiere, sobald V akzeptiert, ansonsten prüfe nächstes z

- es gibt $2^{O(p(n))}$ Zeugen, wobei p ein Polynom
- pro Zeugen z benötigt Simulation $p'(|\langle w, z \rangle|)$ Zeit, wobei p' Polynom
- L ist aus $\text{TIME}(2^{O(p(n))} p'(n + p(n))) \subset \text{EXPTIME}$



Definition

Seien $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Sigma'^*$ Sprachen. Die Funktion f heißt **polyzeit Reduktion von L_1 auf L_2** , gdw.

1. f ist berechenbar und total,
2. $\forall w \in \Sigma^* : w \in L_1 \iff f(w) \in L_2$,
3. f wird durch eine TM berechnet deren Laufzeit ein Polynom in der Eingabelänge ist

Definition

Eine Sprache L_1 ist **polyzeit reduzierbar** auf L_2 , gdw. es eine polyzeit Reduktion von L_1 auf L_2 gibt.

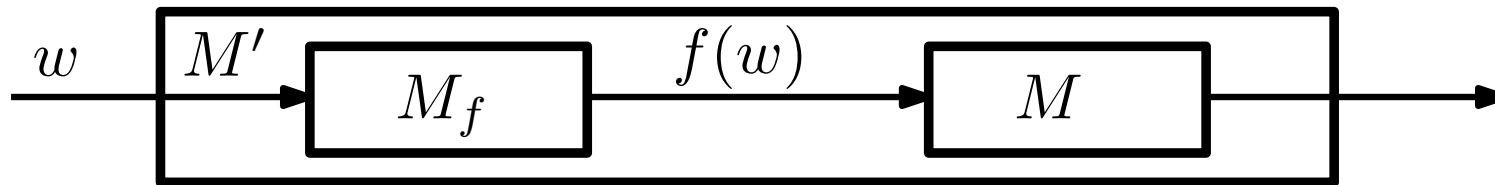
Schreibweise: $L_1 \leq_p L_2$

- modifizierter Reduktionsbegriff erlaubt es uns Probleme zu "übersetzen" unter Beibehaltung von polynomiellen Laufzeiten ihrer Entscheider

Wenn $B \in P$ und $A \leq_p B$, dann auch $A \in P$.

Beweis

- Sei M der polynomieller Entscheider für B und f eine polyzeit Reduktion von A auf B .
- Sei M_f TM, die f in polyzeit berechnet.
- Konstruiere TM M' , als Hintereinanderschaltung von M und M_f



$$M'(w) \text{ akz.} \iff M(f(w)) \text{ akz.} \iff f(w) \in B \iff w \in A$$

- Laufzeit von $M'(w)$ entspricht $t_f(|w|) + t_M(|f(w)|)$

\nearrow
 Laufzeit M_f

\nearrow
 Laufzeit M

- $|f(w)|$ ist beschränkt durch ein Polynom p und t_M durch ein Polynom p'
 $\rightarrow t_M(|f(w)|) \leq \underline{p'(p(n))}$ \blacktriangleleft Polynom
- Laufzeit von M ist beschränkt durch Polynom $\rightarrow A \in P$ □

Definition

Eine Sprache L heißt **NP-vollständig**, gdw.

1. $L \in \text{NP}$,
2. $\forall L' \in \text{NP}$ gilt $L' \leq_p L$.

- NP-vollständige Sprachen kann man als die schwersten Sprachen in der Klasse NP verstehen

$$\text{NPC} := \{L \mid L \text{ ist NP-vollständig}\}$$

- Gilt in der obigen Definition nur der Punkt 2., heißt L **NP-schwer**.

Satz 40

Wenn $B \in \text{P}$ und $B \in \text{NPC}$, dann gilt $\text{P} = \text{NP}$.

Beweis

- Für jedes $A \in \text{NP}$ gilt nach Definition $A \leq_p B$
- Aus Satz 39 folgt dann aber, dass $A \in \text{P}$
- Also gilt $\text{NP} \subseteq \text{P}$, und $\text{P} \subseteq \text{NP}$ wurde bereits gezeigt $\longrightarrow \text{P} = \text{NP}$

