11c
NP-Vollständigkeit

PD Dr. Jan Johannsen

Institut für Informatik

Stand: 17. Juni 2026
Basierend auf Folien von Prof. Dr. David Sabel und Prof. Dr. Jasmin Blanchette

Wiederholung: P

Definition

Für eine Funktion $f : \mathbb{N} \to \mathbb{N}$ sei die Klasse $TIME(f(n))$ genau die Menge der Sprachen L, für die es eine stets anhaltende Mehrband-DTM M gibt mit $L(M) = L$ und $time_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$.

Definition

Die Klasse P ist definiert als

$$\mathrm{P} := \bigcup_{p \text{ Polynom}} TIME(p(n))$$

Wiederholung: NP

Definition

Für eine Funktion $f : \mathbb{N} \to \mathbb{N}$ sei die Klasse $NTIME(f(n))$ genau die Menge der Sprachen L, für die es eine stets anhaltende Mehrband-NTM M gibt mit $L(M) = L$ und $ntime_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$.

Definition

Die Klasse NP ist definiert als

$$\text{NP} := \bigcup_{p \text{ Polynom}} NTIME(p(n))$$

Wiederholung: P vs. NP

Die Frage „Gilt $\mathsf{P} = \mathsf{NP}$ oder $\mathsf{P} \neq \mathsf{NP}$?“ ist bis heute ungelöst.

- $\mathsf{P} \subseteq \mathsf{NP}$ ist klar.
- Es gibt gute Gründe, $\mathsf{P} \neq \mathsf{NP}$ zu vermuten.

Bedeutung von P vs. NP

Obwohl man die P-vs.-NP-Frage nicht geklärt hat,
will man wissen, wie schwer ein Problem ist:

- Wenn man weiß, dass das Problem in P liegt,
  dann existiert ein effizienter Algorithmus.
- Wenn man nur weiß, dass das Problem in NP liegt,
  dann kennt man nur Algorithmen, die in deterministischer Exponentialzeit laufen.

Heute: NP-Vollständigkeit:
Zeige, dass ein gegebenes Problem zu den schwersten Problemen in NP zählt.

Polynomialzeit-Reduktion

Definition

Seien $L_1 \subseteq \Sigma_1^*$ und $L_2 \subseteq \Sigma_2^*$ Sprachen.
Dann sagen wir L_1 ist auf L_2 polynomiell reduzierbar (geschrieben $L_1 \leq_p L_2$), falls es eine totale und in deterministischer Polynomialzeit berechenbare Funktion $f : \Sigma_1^* \to \Sigma_2^*$ gibt, sodass für alle $w \in \Sigma_1^*$ gilt: $w \in L_1$ g.d.w. $f(w) \in L_2$.
Die Funktion f nennt man Polynomialzeit-Reduktion.

Die Definition von $L_1 \leq_p L_2$ ist analog zu der von $L_1 \leq L_2$, mit dem Zusatz, dass f in deterministischer Polynomialzeit berechenbar sein muss.

Analogie:

$L_1 \leq L_2$ und L_2 (semi-)entscheidbar	$L_1 \leq_p L_2$ und $L_2 \in$ (N)P
$\Longrightarrow L_1$ (semi-)entscheidbar	$\Longrightarrow L_1 \in$ (N)P

Nachweis der Zugehörigkeit zu P

Lemma

Falls $L_1 \leq_p L_2$ und $L_2 \in \mathsf{P}$, dann gilt $L_1 \in \mathsf{P}$.

Beweis Seien $L_1 \leq_p L_2$ und f in Polynomialzeit berechenbar.

Sei M_f die DTM, die f in Polynomialzeit berechnet.

Seien $L_2 \in \mathsf{P}$ und M_2 eine DTM, sodass $L(M_2) = L_2$,
wobei M_2 stets in deterministischer Polynomialzeit anhält.

Sei $M_f; M_2$ die Hintereinanderausführung von M_f und M_2. Dann gilt $L(M_f; M_2) = L_1$.

$M_f; M_2$ hält stets in deterministischer Polynomialzeit.

Daher gilt $L_1 \in \mathsf{P}$. □

Nachweis der Zugehörigkeit zu NP

Lemma

Falls $L_1 \leq_p L_2$ und $L_2 \in \mathsf{NP}$, dann gilt $L_1 \in \mathsf{NP}$.

Der Beweis ist analog:

Beweis Seien $L_1 \leq_p L_2$ und f in Polynomialzeit berechenbar.

Sei M_f die DTM, die f in Polynomialzeit berechnet.

Seien $L_2 \in \mathsf{NP}$ und M_2 eine NTM, sodass $L(M_2) = L_2$,
wobei M_2 stets in nichtdeterministischer Polynomialzeit anhält.

Sei $M_f; M_2$ die Hintereinanderausführung von M_f (deterministisch) und
M_2 (nichtdeterministisch). Dann gilt: $L(M_f; M_2) = L_1$.

$M_f; M_2$ hält stets in nichtdeterministischer Polynomialzeit.

Daher gilt $L_1 \in \mathsf{NP}$. □

Transitivität der Polynomialzeit-Reduktion

Lemma

Die Relation $\leq_p$ ist transitiv, d.h. wenn $L_1 \leq_p L_2$ und $L_2 \leq_p L_3$, dann gilt auch $L_1 \leq_p L_3$.

Beweis Die Komposition von zwei Polynomen bleibt ein Polynom. □

NP-Vollständigkeit

Definition

Eine Sprache L heißt NP-vollständig, wenn gilt

1. $L \in \mathsf{NP}$ und
2. L ist NP-schwer: für alle $L' \in \mathsf{NP}$ gilt $L' \leq_p L$.

NP-vollständige Probleme sind die schwierigsten Probleme in NP.

NP-Schwere besagt, dass man mit dem NP-vollständigen Problem alle anderen Probleme aus NP lösen kann.

NP-schwer wird manchmal auch NP-hart genannt.

Nachweis der NP-Vollständigkeit

Nachweis der NP-Vollständigkeit einer Sprache L:

1. Zugehörigkeit zu NP:
   Gib eine Polynomialzeit-beschränkte NTM an, die L entscheidet.
   (Alternativ: Gib eine Polynomialzeit-Reduktion von $L \leq_p L_1$ an mit $L_1 \in \mathsf{NP}$.)
2. NP-Schwere:
   Statt jedes mal neu zu beweisen, dass alle Probleme aus NP auf L polynomiell reduzierbar sind, wähle ein NP-schweres Problem L_0 und zeige $L_0 \leq_p L$.

   Da L_0 NP-schwer, gilt $L' \leq_p L_0$ für alle $L' \in \mathsf{NP}$ und damit $L' \leq_p L_0 \leq_p L$ und mit Transitivität: $L' \leq_p L$ für alle $L' \in \mathsf{NP}$.

   Daher ist L NP-schwer.

Nachweis der NP-Schwere

Analog zum Vorgehen wie bei der Unentscheidbarkeit, wesentlicher Unterschied: Polynomialzeit-Reduktion:

$L_1 \leq L_2$ und L_1 unentscheidbar
$\Longrightarrow L_2$ unentscheidbar

$L_1 \leq_p L_2$ und L_1 NP-schwer
$\Longrightarrow L_2$ NP-schwer

Bedingung für P = NP

Satz

Sei L ein NP-vollständiges Problem. Dann gilt $L \in \mathsf{P}$ g.d.w. $\mathsf{P} = \mathsf{NP}$.

Beweis

$\Longleftarrow$ Offensichtlich.

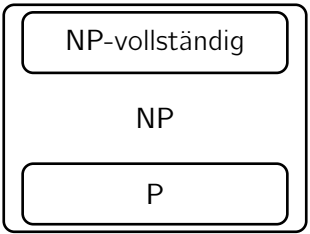
$\Longrightarrow$ Sei L NP-vollständig und $L \in \mathsf{P}$.

Aus NP-Schwere von L folgt:
Für alle $L' \in \mathsf{NP}$: $L' \leq_p L$ und damit $L' \in \mathsf{P}$.

Da dies für alle $L' \in \mathsf{NP}$ gilt, folgt $\mathsf{P} = \mathsf{NP}$. □

Also: Es reicht aus nachzuweisen, dass ein NP-vollständiges Problem in P bzw. nicht in P liegt, um die P-vs.-NP-Frage ein für allemal beantworten zu können.

Vermutete Lage der Probleme



Unter der Annahme $\mathsf{P} \neq \mathsf{NP}$ gibt es Probleme in NP,
die nicht in P liegen und nicht NP-vollständig sind (Ladner 1975).

Ausblick

Was fehlt noch? Ein erstes Problem L_0, dass man direkt als NP-vollständig beweist. Ein solches L_0 und den NP-Vollständigkeitsbeweis sehen wir in der nächsten Vorlesung.

Danach können wir NP-Vollständigkeit von L zeigen durch

1. $L \in \mathsf{NP}$
2. $L_0 \leq_p L$.

Danach lernen wir eine Auswahl an NP-vollständigen Problemen kennen.