LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Formale Sprachen und Komplexität
Theoretische Informatik für Medieninformatiker
Sommersemester 2022

$\mathcal{NP}$-Vollständigkeit

Prof. Dr. David Sabel

LFE Theoretische Informatik

Letzte Änderung der Folien: 10. Juli 2022

Wiederholung: $\mathcal{P}$

Definition (Klasse $TIME(f(n))$)

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 deterministische, stets anhaltende, Mehrband-TM M gibt, mit $L(M) = L$ und $time_M(w) \leq f(|w|)$ für alle $w \in \Sigma^*$

Definition (Komplexitätsklasse $\mathcal{P}$)

Die Klasse $\mathcal{P}$ ist definiert als

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

Wiederholung: $\mathcal{NP}$

Definition

Für eine Funktion $f : \mathbb{N} \to \mathbb{N}$ bezeichne $NTIME(f(n))$ die Klasse aller Sprachen L, für die es eine nichtdeterministische Mehrband-TM M gibt mit $L(M) = L$ und für alle $w \in \Sigma^*$ gilt $ntime_M(w) \leq f(|w|)$.

Definition

Die Klasse $\mathcal{NP}$ ist definiert als

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

Wiederholung: Die millionenschwere Frage

Gilt $\mathcal{P} = \mathcal{NP}$ oder $\mathcal{P} \neq \mathcal{NP}$?

- bis heute ungelöst
- $\mathcal{P} \subseteq \mathcal{NP}$ ist klar
- gute Gründe $\mathcal{P} \neq \mathcal{NP}$ zu vermuten

Obwohl man die Frage nicht geklärt hat, will man wissen, wie schwer ein Problem ist:

- Wenn man weiß: Problem liegt in $\mathcal{P}$, dann: Effizienter Algorithmus existiert
- Wenn man nur weiß: Problem liegt in $\mathcal{NP}$, dann:
  Man kennt nur Algorithmen, die in **deterministischer Exponentialzeit laufen**
- Heute: $\mathcal{NP}$-Vollständigkeit:
  Zeige, dass ein Problem zu den schwersten Problemen in $\mathcal{NP}$ zählt.

Polynomialzeit-Reduktion

Definition (Polynomialzeit-Reduktion (einer Sprache auf eine andere))

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: $w \in L_1 \iff f(w) \in L_2$.

Analogie zu Reduktionen in der Berechenbarkeitstheorie $L_1 \leq L_2$:

Zusatz hier: Polynomialzeit!
Nächste Analogie:

Berechenbarkeitstheorie:	Komplexitätstheorie:
$L_1 \leq L_2$ und L_2 (semi-) entscheidbar	$L_1 \leq_p L_2$ und $L_2 \in (\mathcal{N})\mathcal{P}$
$\implies$ L_1 (semi-) entscheidbar	$\implies$ $L_1 \in (\mathcal{N})\mathcal{P}$

Polynomialzeit-Reduktion (Eigenschaften)

Lemma

Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{P}$, dann gilt $L_1 \in \mathcal{P}$.
Ebenso gilt: Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{NP}$, dann gilt $L_1 \in \mathcal{NP}$.

Beweis (für $\mathcal{P}$):

- Sei $L_1 \leq_p L_2$ und f in Polynomialzeit berechenbar
- Sei M_f die DTM, die f in Polynomialzeit durch p beschränkt berechnet.
- Sei $L_2 \in \mathcal{P}$, $L(M_2) = L_2$, wobei Schritte von M_2 auf $w \leq q(|w|)$
- 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 Polynomialzeit: $|f(w)| \leq |w| + p(|w|)$
  (da M_f nicht mehr in $p(|w|)$-Schritten schreiben kann)
  $M_f; M_2$ macht höchstens $r(|w|) := p(|w|) + q(|w| + p(|w|))$ Schritte
- Es gilt $L_1 \in \mathcal{P}$.

Polynomialzeit-Reduktion (Eigenschaften) (2)

Lemma

Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{P}$, dann gilt $L_1 \in \mathcal{P}$.
Ebenso gilt: Falls $L_1 \leq_p L_2$ und $L_2 \in \mathcal{NP}$, dann gilt $L_1 \in \mathcal{NP}$.

Beweis (für $\mathcal{NP}$): Analog:

- Sei $L_1 \leq_p L_2$ und f in Polynomialzeit berechenbar
- Sei M_f die DTM, die f in Polynomialzeit durch p beschränkt berechnet.
- Sei $L_2 \in \mathcal{NP}$, $L(M_2) = L_2$, wobei max. Schritte von M_2 auf $w \leq q(|w|)$
- 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: $|f(w)| \leq |w| + p(|w|)$
  (da M_f pro nicht mehr in $p(|w|)$-Schritten schreiben kann)
  $M_f; M_2$ macht auf maximalen Pfad höchstens $r(|w|) := p(|w|) + q(|w| + p(|w|))$ Schritte
- Es gilt $L_1 \in \mathcal{NP}$.

Polynomialzeit-Reduktion (Eigenschaften) (3)

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 $L_1 \leq_p L_3$

Analog zum vorherigen Beweis:
Komposition von zwei Polynomen bleibt Polynom.

$\mathcal{NP}$-Vollständigkeit

Definition ($\mathcal{NP}$-Vollständigkeit)

Eine Sprache L heißt $\mathcal{NP}$-vollständig, wenn gilt

1. $L \in \mathcal{NP}$ und
2. L ist $\mathcal{NP}$-schwer (manchmal auch $\mathcal{NP}$-hart genannt):
   Für alle $L' \in \mathcal{NP}$ gilt $L' \leq_p L$

$\mathcal{NP}$-vollständige Probleme sind die schwierigsten Probleme in $\mathcal{NP}$:

$\mathcal{NP}$-Schwere besagt, dass man mit dem $\mathcal{NP}$-vollständigen Problem alle anderen Probleme aus $\mathcal{NP}$ (in zusätzlicher deterministischer Polynomialzeit) lösen kann

$\mathcal{NP}$-Vollständigkeit beweisen

Nachweis der $\mathcal{NP}$-Vollständigkeit von L

- Zugehörigkeit zu $\mathcal{NP}$: Gebe polynomiell Laufzeit-beschränkte NTM an, die L entscheidet (alternativ: Polynomialzeit reduktion von $L \leq_p L_1$ mit $L_1 \in \mathcal{NP}$)
- **$\mathcal{NP}$-Schwere:** Statt jedes mal neu zu beweisen, dass alle Probleme aus $\mathcal{NP}$ auf L polynomiell reduzierbar, wähle ein $\mathcal{NP}$-schweres Problem L_0 und zeige $L_0 \leq_p L$.

  Dann folgt L ist $\mathcal{NP}$-schwer:

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

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

Berechenbarkeitstheorie:	Komplexitätstheorie:
$L_0 \leq L$ und L_0 unentscheidbar	$L_0 \leq_p L$ und L_0 $\mathcal{NP}$-schwer
$\Longrightarrow$ L unentscheidbar	$\Longrightarrow$ L $\mathcal{NP}$-schwer

Satz

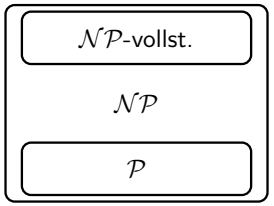
Sei L ein $\mathcal{NP}$-vollständiges Problem. Dann gilt $L \in \mathcal{P} \iff \mathcal{P} = \mathcal{NP}$.

Beweis:

- Sei L $\mathcal{NP}$-vollständig und $L \in \mathcal{P}$
- Aus $\mathcal{NP}$-Schwere von L folgt:
  Für alle $L' \in \mathcal{NP}$: $L' \leq_p L$ und damit $L' \in \mathcal{P}$.
- Da dies für alle $L' \in \mathcal{NP}$ gilt, folgt $\mathcal{P} = \mathcal{NP}$.

Also: Es reicht aus für ein $\mathcal{NP}$-vollständiges Problem nachzuweisen, dass es in $\mathcal{P}$ bzw. nicht in $\mathcal{P}$ liegt, um die $\mathcal{P}$-vs-$\mathcal{NP}$-Frage ein für allemal beantworten.

Vermutete Lage der Probleme



Es ist bekannt (Ladner 1975):

Unter der Annahme $\mathcal{P} \neq \mathcal{NP}$ gibt es Probleme in $\mathcal{NP}$ gibt, die nicht in $\mathcal{P}$ liegen und nicht $\mathcal{NP}$-vollständig sind.

Möglicher Kandidat:

Graph-Isomorphismus-Problem. Weder ein polynomieller Algorithmus noch dessen $\mathcal{NP}$-Vollständigkeit bekannt.

Ausblick

Was fehlt noch?

Ein erstes Problem L_0, dass man direkt als $\mathcal{NP}$-vollständig beweist.

(ein solches L_0 und den $\mathcal{NP}$-Vollständigkeitsbeweis sehen wir beim nächsten Mal)

Danach kann man $\mathcal{NP}$-Vollständigkeit von L zeigen durch:

- $L \in \mathcal{NP}$
- $L_0 \leq_p L$

Danach: Lerne einen Satz an $\mathcal{NP}$-vollständigen Problemen kennen.