

# $\mathcal{NP}$ -Vollständigkeit

Prof. Dr. Jasmin Blanchette

Lehr- und Forschungseinheit für  
Theoretische Informatik

Stand: 4. Juli 2023

Folien ursprünglich von PD Dr. David Sabel



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

Für eine Funktion  $f : \mathbb{N} \rightarrow \mathbb{N}$  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 (Klasse $\mathcal{P}$ )

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

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

### Definition (Klasse $NTIME(f(n))$ )

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

### Definition (Klasse $\mathcal{NP}$ )

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

# 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.

## 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  $f : \Sigma_1^* \rightarrow \Sigma_2^*$  gibt, sodass für alle  $w \in \Sigma_1^*$  gilt:  $w \in L_1 \iff f(w) \in L_2$ .

## 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  $f : \Sigma_1^* \rightarrow \Sigma_2^*$  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

## 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  $f : \Sigma_1^* \rightarrow \Sigma_2^*$  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:

$L_1 \leq L_2$  und  $L_2$  (semi-)entscheidbar

$\implies L_1$  (semi-)entscheidbar

Komplexitätstheorie:

$L_1 \leq_p L_2$  und  $L_2 \in (\mathcal{N})\mathcal{P}$

$\implies L_1 \in (\mathcal{N})\mathcal{P}$



# Polynomialzeit-Reduktion (Eigenschaften)

---

## Lemma

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

Beweis:

# Polynomialzeit-Reduktion (Eigenschaften)

## Lemma

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

Beweis:

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.

# Polynomialzeit-Reduktion (Eigenschaften)

## Lemma

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

Beweis:

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.

# Polynomialzeit-Reduktion (Eigenschaften)

## Lemma

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

Beweis:

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.
- ▶ Sei  $L_2 \in \mathcal{P}$ ,  $L(M_2) = L_2$ , wobei  $M_2$  stets in Polynomialzeit anhält.

# Polynomialzeit-Reduktion (Eigenschaften)

## Lemma

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

Beweis:

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.
- ▶ Sei  $L_2 \in \mathcal{P}$ ,  $L(M_2) = L_2$ , wobei  $M_2$  stets in Polynomialzeit anhält.
- ▶ Sei  $M_f; M_2$  die Hintereinanderausführung von  $M_f$  und  $M_2$ . Dann gilt  $L(M_f; M_2) = L_1$ .

# Polynomialzeit-Reduktion (Eigenschaften)

## Lemma

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

Beweis:

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.
- ▶ Sei  $L_2 \in \mathcal{P}$ ,  $L(M_2) = L_2$ , wobei  $M_2$  stets in 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 Polynomialzeit.

# Polynomialzeit-Reduktion (Eigenschaften)

## Lemma

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

Beweis:

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.
- ▶ Sei  $L_2 \in \mathcal{P}$ ,  $L(M_2) = L_2$ , wobei  $M_2$  stets in 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 Polynomialzeit.
- ▶ Es gilt  $L_1 \in \mathcal{P}$ . □

## Polynomialzeit-Reduktion (Eigenschaften) (2)

---

### Lemma

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

Beweis (analog):



## Polynomialzeit-Reduktion (Eigenschaften) (2)

### Lemma

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

Beweis (analog):

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.

## Polynomialzeit-Reduktion (Eigenschaften) (2)

### Lemma

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

Beweis (analog):

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.

## Polynomialzeit-Reduktion (Eigenschaften) (2)

### Lemma

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

Beweis (analog):

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.
- ▶ Sei  $L_2 \in \mathcal{NP}$ ,  $L(M_2) = L_2$ , wobei  $M_2$  stets in Polynomialzeit anhält.

## Polynomialzeit-Reduktion (Eigenschaften) (2)

### Lemma

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

Beweis (analog):

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.
- ▶ Sei  $L_2 \in \mathcal{NP}$ ,  $L(M_2) = L_2$ , wobei  $M_2$  stets in Polynomialzeit anhält.
- ▶ Sei  $M_f; M_2$  die Hintereinanderausführung von  $M_f$  (deterministisch) und  $M_2$  (nichtdeterministisch). Dann gilt:  $L(M_f; M_2) = L_1$ .

## Polynomialzeit-Reduktion (Eigenschaften) (2)

### Lemma

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

Beweis (analog):

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.
- ▶ Sei  $L_2 \in \mathcal{NP}$ ,  $L(M_2) = L_2$ , wobei  $M_2$  stets in 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.

## Polynomialzeit-Reduktion (Eigenschaften) (2)

### Lemma

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

Beweis (analog):

- ▶ Sei  $L_1 \leq_p L_2$  und  $f$  in Polynomialzeit berechenbar.
- ▶ Sei  $M_f$  die DTM, die  $f$  in Polynomialzeit berechnet.
- ▶ Sei  $L_2 \in \mathcal{NP}$ ,  $L(M_2) = L_2$ , wobei  $M_2$  stets in 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.
- ▶ 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  $L_1 \leq_p L_3$ .

Komposition von zwei Polynomen bleibt Polynom.

## 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  $\mathcal{NP}$  lösen kann.



# $\mathcal{NP}$ -Vollständigkeit beweisen

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

► **Zugehörigkeit zu  $\mathcal{NP}$ :**

Gib 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 sind, 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  $L' \in \mathcal{NP}$ .

## $\mathcal{NP}$ -Vollständigkeit beweisen (2)

---

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

Berechenbarkeitstheorie:

$L_0 \leq L$  und  $L_0$  unentscheidbar  
 $\implies L$  unentscheidbar

Komplexitätstheorie:

$L_0 \leq_p L$  und  $L_0$   $\mathcal{NP}$ -schwer  
 $\implies L$   $\mathcal{NP}$ -schwer

## Bedingung für $\mathcal{P} = \mathcal{NP}$

---

### Satz

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

## Bedingung für $\mathcal{P} = \mathcal{NP}$

### Satz

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

Beweis ( $\Rightarrow$ ):

- ▶ 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  $\mathcal{P} = \mathcal{NP}$ . □

## Bedingung für $\mathcal{P} = \mathcal{NP}$

### Satz

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

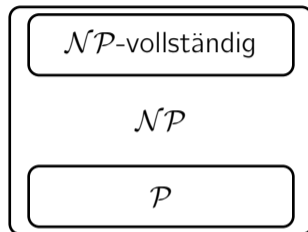
Beweis ( $\Rightarrow$ ):

- ▶ 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 zu können.

## Vermutete Lage der Probleme

---



Es ist bekannt (Ladner 1975):

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

Möglicher Kandidat:

Graph-Isomorphismus-Problem. Weder ein polynomieller Algorithmus noch dessen  $NP$ -Vollständigkeit bekannt.

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 eine Auswahl an  $\mathcal{NP}$ -vollständigen Problemen kennen.