



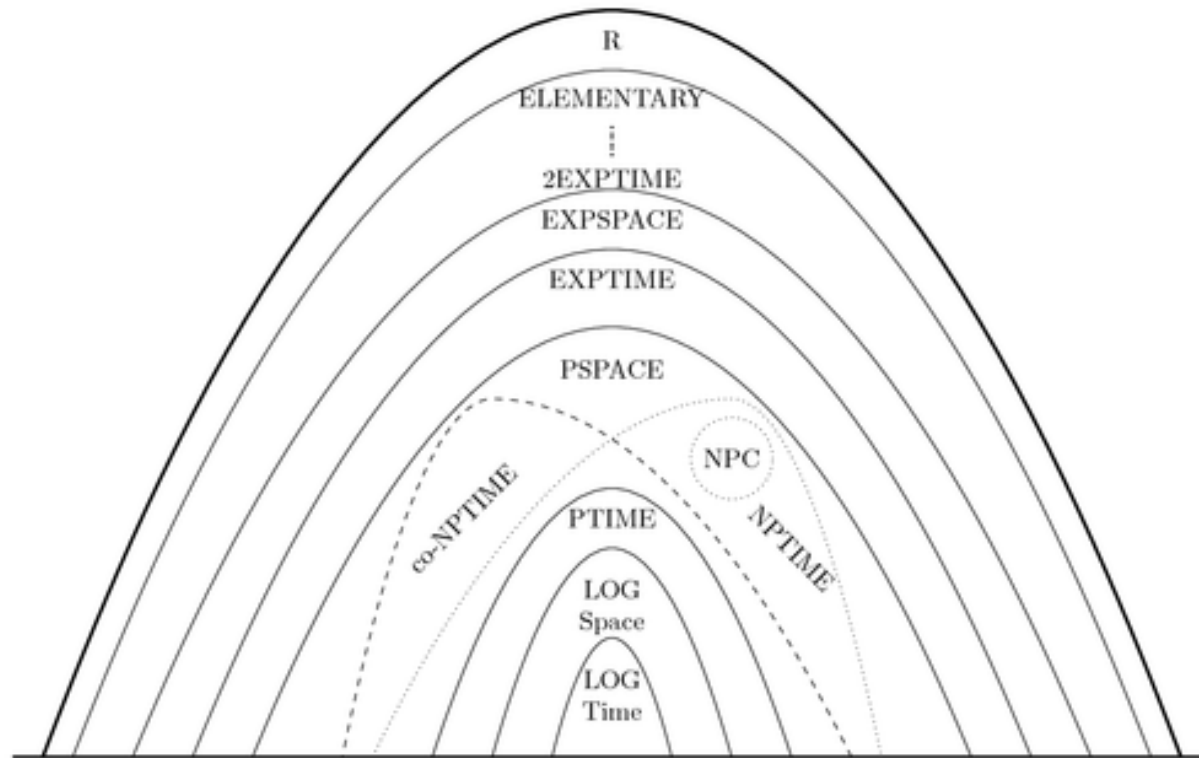
Technische
Universität
Braunschweig



Theoretische Informatik 2

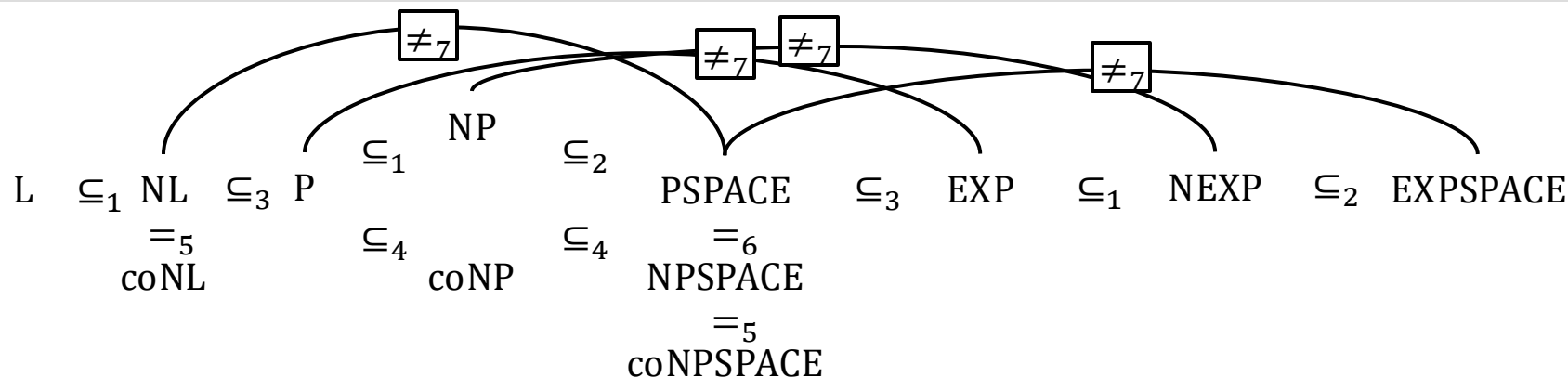
Arne Schmidt

Letzte Woche



Kapitel 8 – Komplexitätskarte

Komplexitätskarte



- | | |
|--|---|
| 1. Nicht-determinismus ist potentiell Mächtiger (Lemma 7.21) | 4. coNP erfüllt dieselben Inklusionsbeziehungen wie NP (Lemma 7.37) |
| 2. Jede Zeitschranke für NTMs ist auch Platzschranke. (Lemma 7.25) | 5. Satz von Immermann & Szelepcsényi (Theorem 2.1) |
| 3. Platzschranke für NTMs liefert exponentiell größere Laufzeit für DTMs. (Lemma 7.30) | 6. Satz von Savitch (Kapitel 12) |
| | 7. Hierarchiesätze (Kapitel 13) |

Wichtige Fragen

Wie stehen
deterministische Zeit- und
nicht-deterministische
Zeitkomplexitäten
zueinander?

Wie stehen Zeit- und
Platzkomplexitäten
zueinander?

Später im Detail: P vs NP

Charakteristische Probleme

L: Arithmetische Probleme, bspw. gegebene Zahlen x, y, z in binär, gilt $x + y = z$?

NL: Suchprobleme in Graphen („Existiert ein Pfad von s nach t “)

P: CVP, das Auswertungsproblem für Schaltkreise

NP: SAT, die Erfüllbarkeit einer booleschen Formel

PSPACE: QBF, die Auswertung quantifizierter boolescher Formeln.

Man vermutet, dass all die genannten Probleme die „schwersten“ Probleme der Klasse sind.

Wir sehen später: Haben wir einen Algorithmus, der die Zugehörigkeit des Problems zur Klasse beweist, haben wir einen Algorithmus für jedes Problem aus derselben Klasse!

Kapitel 9 – L und NL

L und NL



Zur Wiederholung: Ein Problem liegt in $(N)L$, wenn es sich (nicht-)deterministisch mit logarithmischem Platz lösen kann.

Man glaubt, dass $NL \neq L$ gilt.

Wir werden Probleme in NL identifizieren, von denen man glaubt, dass sie nicht in L liegen. Diese werden durch den Begriff „Vollständigkeit für NL “ charakterisiert.

Kapitel 9.1 – Reduktionen und Vollständigkeit

Reduktionen

Definition 9.1

Sei R eine Klasse von Funktionen. Eine Sprache (oder Problem) $A \subseteq \Sigma_1^*$ heißt **R-many-one-reduzierbar** auf eine Sprache $B \subseteq \Sigma_2^*$, falls es eine Funktion $f: \Sigma_1^* \rightarrow \Sigma_2^*$ gibt, sodass für alle $x \in \Sigma_1^*$ gilt:

$$x \in A \Leftrightarrow f(x) \in B$$

Wir nennen f die **Reduktion** und schreiben $A \leq_m^R B$.

Bemerkung 9.3

Die Reduktionen im ersten Teil der Vorlesung, waren R-many-one-Reduktionen, wobei R die Menge der totalen berechenbaren Funktionen waren.

Für die Komplexitätstheorie werden hauptsächlich

- Polynomialzeitreduktionen (R ist die Menge aller polynomiellen Funktionen; $\leq_m^{poly}$), und
- Log-Space-Reduktionen (R ist die Menge aller log-space-Funktionen; $\leq_m^{log}$) verwendet.

Diese definieren wir gleich etwas präziser.

Schwere und Vollständigkeit

Definition 9.2

Sei $\mathcal{C}$ eine Komplexitätsklasse, R eine Menge von Funktionen und B eine Sprache.

- a) Die Sprache B heißt $\mathcal{C}$ -schwer bzgl. R -many-one Reduktionen, falls sich alle $A \in \mathcal{C}$ mit R -many-one-Reduktionen auf B reduzieren lassen:

$$\forall A \in \mathcal{C}: A \leq_m^R B$$

- b) Die Sprache B heißt $\mathcal{C}$ -vollständig bzgl. R -many-one Reduktionen falls
- $B \in \mathcal{C}$ („Membership“, „untere Schranke“ der Schwere)
 - B ist $\mathcal{C}$ -schwer bzgl. R -many-one Reduktionen („Hardness“, „obere Schranke“ der Schwere)

Intuitiv:

- a) bedeutet, dass B mind. so schwer wie jedes Problem in $\mathcal{C}$ ist.
b) bedeutet, dass B tatsächlich das schwerste Problem in $\mathcal{C}$ ist.

Bedingungen an die Reduktionen

Die Reduktionen sollten schwächer als die Klasse selbst sein, d.h. $\mathcal{C}$ ist unter R-many-one-Reduktionen abgeschlossen.

Andernfalls könnte man einen signifikanten Teil der Berechnung der zu reduzierenden Sprache bereits durch die Reduktion berechnen.

Die Reduktionen sollten transitiv sein.

Insbesondere sollte gelten:

A ist $\mathcal{C}$ -schwer und $A \leq_m^R B$, dann ist auch B $\mathcal{C}$ -schwer.

Logspace-Reduktionen

Definition 9.4

Eine Funktion $f: \Sigma_1^* \rightarrow \Sigma_2^*$ ist **logspace-berechenbar**, wenn es eine DTM mit speziellem Ein- und Ausgabeband gibt, wobei:

- Eingabeband mit Alphabet Σ_1^* ist read-only
- Ausgabeband mit Alphabet Σ_2^* ist write-only
- Arbeitsband mit Alphabet Γ ist read-write

Außerdem ist die DTM

- total,
- hält zu jeder Eingabe nach endlich vielen Schritten und hat $f(x) \in \Sigma_2^*$ auf das Ausgabeband geschrieben und
- der Speicherverbrauch ist durch $O(\log n)$ beschränkt.

In der Literatur findet man diese DTM auch unter dem Begriff **Logspace-Transducer**

Polyzeit-Reduktionen

Definition 9.4 (cont.)

Eine Funktion $f: \Sigma_1^* \rightarrow \Sigma_2^*$ ist **Polynomialzeit-berechenbar**, wenn es eine DTM mit speziellem Ein- und Ausgabeband gibt, wobei:

- Eingabeband mit Alphabet Σ_1^* ist read-only
- Ausgabeband mit Alphabet Σ_2^* ist write-only
- Arbeitsband mit Alphabet Γ ist read-write

Außerdem ist die DTM

- total,
- hält zu jeder Eingabe nach endlich vielen Schritten und hat $f(x) \in \Sigma_2^*$ auf das Ausgabeband geschrieben und
- **der Zeitverbrauch ist durch $O(n^k)$ beschränkt, wobei k eine vom Input unabhängige Konstante ist.**

Logspace- und Polytime-Reduktionen

Definition 9.6

Eine Sprache $A \subseteq \Sigma_1^*$ heißt **logspace-reduzierbar** auf eine Sprache $B \subseteq \Sigma_2^*$, wenn $A \leq_m^R B$ mit R gleich der Klasse der logspace-berechenbaren Funktionen gilt. Wir schreiben $A \leq_m^{\log} B$

Alternativ:

Eine Sprache $A \subseteq \Sigma_1^*$ heißt **logspace-reduzierbar** auf eine Sprache $B \subseteq \Sigma_2^*$, wenn es eine logspace-berechenbaren Funktion f gibt, sodass für alle $x \in \Sigma_1^*$ gilt $x \in A \Leftrightarrow f(x) \in B$.

Analog:

Eine Sprache $A \subseteq \Sigma_1^*$ heißt **polynomiell-reduzierbar** auf eine Sprache $B \subseteq \Sigma_2^*$, wenn $A \leq_m^R B$ mit R gleich der Klasse der Polynomialzeit-berechenbaren Funktionen gilt. Wir schreiben $A \leq_m^{\text{poly}} B$

Lemma 9.7 Falls $A \leq_m^{\log} B$, dann auch $A \leq_m^{\text{poly}} B$.

Logspace-Reduktion ist transitiv und zu mächtig für L

Lemma 9.8

Es seien $f: \Sigma_1^* \rightarrow \Sigma_2^*$ und $g: \Sigma_2^* \rightarrow \Sigma_3^*$ logspace-berechenbare Funktionen. Dann ist auch ihre Verkettung $g \circ f: \Sigma_1^* \rightarrow \Sigma_3^*$ logspace-berechnbar.

Insbesondere folgt dann aus $A \leq_m^{\log} B$ und $B \leq_m^{\log} C$ auch $A \leq_m^{\log} C$.

Beweis: Gleich an der Tafel!

Lemma 9.9

Sei Σ ein endliches Alphabet.

- a) Eine Sprache $\mathcal{L} \subseteq \Sigma^*$ ist in L, gdw. $\mathcal{L} \leq_m^{\log} \{1\}$
- b) Jede Sprache $\mathcal{L} \subseteq \Sigma^*$ in L mit $\mathcal{L} \neq \emptyset$ und $\mathcal{L} \neq \Sigma^*$ ist L-vollständig (bzgl. Logspace-Reduktionen)

Beweis: Selbst!

Abgeschlossenheit

Lemma 9.10

Sei $A \leq_m^{log} B$. Wenn B in L (bzw. NL , bzw. P) ist, dann ist auch A in L (bzw. NL , bzw. P).

Lemma 9.11

Sei A eine Sprache.

- a) Falls A NL -schwer bzgl. logspace-Reduktionen ist und A in L ist, dann folgt $NL = L$.
- b) Falls A P -schwer bzgl. logspace-Reduktionen ist und A in NL ist, dann folgt $NL = P$.