Theoretische Grundlagen der Informatik
Tutorium XII & XIII (SR 301)
Tut Nr. 8 – Üb7, Komplexitätstheorie

David Münch

Karlsruher Institut für Technologie
Institut für Informatik
IKS Müller-Quade

17. Dezember 2009

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Organisatorisches

Email: muenchdavid@gmail.com

`https://www.stud.uni-karlsruhe.de/~uhbro/`

Tutorium 12: Donnerstags 8:00 Uhr - Raum 301
Tutorium 13: Donnerstags 9:45 Uhr - Raum 301

Übungsblattabgabe Mittwochs 12:00 Uhr.

Organisatorisches

Deckblatt benutzen: `http://www.stud.uni-karlsruhe.de/~unbdh/deckblatt/index.php?course=6`

Gruppenarbeit erwünscht, aber jeder muss handschriftliche Lösung mit Namen aller Gruppenteilnehmer abgeben.

50% der Punkte sind notwendig für den Schein.

Organisatorisches

Nicht abgeholte Übungsblätter können ab sofort bei Nico Döttling R274 abgeholt werden.
`https://puck.iaks.uka.de/eiss/?id=167`

Was wollen wir heute erreichen?

Was wollen wir heute erreichen?

- Einführung in die Komplexitätstheorie.

Was wollen wir heute erreichen?

- Einführung in die Komplexitätstheorie.
- Komplexitätsklassen $\mathcal{P}$ und $\mathcal{NP}$

Was wollen wir heute erreichen?

- Einführung in die Komplexitätstheorie.
- Komplexitätsklassen $\mathcal{P}$ und $\mathcal{NP}$
- Reduktionen

Aufgabe 1

a) Sei $w \in \{0,1\}^n$. Das Zeichen 1 trete genau k mal auf in w. Geben Sie eine möglichst gute obere Schranke für $K(w)$ an.

Aufgabe 1

a) Sei $w \in \{0,1\}^n$. Das Zeichen 1 trete genau k mal auf in w. Geben Sie eine möglichst gute obere Schranke für $K(w)$ an.

b) Zeigen Sie: Die Kolmogorov-Komplexität ist bezüglich der Präfix-Ordnung nicht monoton, d.h. es gibt $x, y \in \{0,1\}^*$ sodass

$$K(xy) < K(x)$$

Lösung:

Es gibt insgesamt $\binom{n}{k}$ Wörter der Länge n, in welchen genau k-mal das Zeichen 1 auftaucht. Wir wählen i so, dass w das lexikographisch i-te Wort in $\{0,1\}^n$ ist, in welchem genau k-mal das Zeichen 1 vorkommt. Folgende Turingmaschine $\mathcal{M}$ berechnet w bei Eingabe n, k und i.

- Initialisiere einen Zähler $z = 0$
- Zähle alle Wörter $v \in \{0,1\}^n$ in lexikographischer Reihenfolge auf. Wann immer ein Wort v genau k Einsen enthält, setze $z = z + 1$.
- Falls $z = i$, breche die Zählschleife ab und gib v aus

Die Beschreibungsgröße von $\langle \mathcal{M} \rangle$ ist eine Konstante c, die unabhängig von n, k und i ist. Es gilt $i \in \{1, \ldots, \binom{n}{k}\}$, also können wir i mit $\log \binom{n}{k}$ Bits beschreiben. Insgesamt gilt also $K(w) \leq 2\log(n) + 2\log(k) + \log \binom{n}{k} + c$. Wir benötigen den Faktor 2 um Trennungen zwischen den einzelnen Worten zu kodieren.

Lösung:
Sei r ein nicht-komprimierbares Wort der Länge l, es gilt also $K(r) = l + c_1$ für eine Konstante c_1 die unabhänig von r ist. Wir betrachten r als Binärdarstellung einer Zahl $n \in \mathbb{N}$. Dann gilt $K(1^n) = l + c_2$ für eine Konstante c_2 die unabhängig von n ist. Wäre $K(1^n) < K(x) + c$ für beliebiges c, so so könnten wir eine kürzere Darstellung für x finden. Sei t die kleinste Zahl sodass $n < 2^t$. Es gilt also $2^t < 2n$ und damit $t \leq \log(2n) = l + \log(2)$. Nun ist 1^n ein Präfix von 1^{2^t} da $n < 2^t$. Es gilt aber $K(1^{2^t}) \leq \log(t) + c_3 \leq \log(l) + c_4$ wobei die Konstanten c_3 und c_4 unabhängig von n und t sind. Damit gilt für ein hinreichend großes l die Ungleichung $K(1^{2^t}) < K(1^n)$.

Aufgabe 2

a) Geben Sie ein Modell für die folgende aussagenlogische Formel ϕ an.

$$\begin{aligned}\phi =&(\forall x \forall y \forall z \forall u \forall v \forall w : P(u,x,v) \wedge P(v,y,z) \wedge P(w,x,y) \\ &\Rightarrow P(u,w,z)) \wedge (\exists x \forall y \exists z : P(x,y,z) \wedge P(x,z,y))\end{aligned}$$

Aufgabe 2

a) Geben Sie ein Modell für die folgende aussagenlogische Formel ϕ an.

$$\begin{aligned}\phi =&(\forall x \forall y \forall z \forall u \forall v \forall w : P(u,x,v) \wedge P(v,y,z) \wedge P(w,x,y) \\ &\Rightarrow P(u,w,z)) \wedge (\exists x \forall y \exists z : P(x,y,z) \wedge P(x,z,y))\end{aligned}$$

b) Zeigen Sie: $\mathrm{Th}(\mathbb{Z},+)$ ist entscheidbar.

Lösung:
Wir wählen als Modell $(\mathbb{Z}, +)$, also $\mathcal{U} = \mathbb{Z}$ und
$P(x, y, z) :\Leftrightarrow x = y + z$.
Die erste Klausel ist wegen dem Assoziativgesetz in $(\mathbb{Z}, +)$ erfüllt

$$(u = x + v) \wedge (v = y + z) \wedge (w = x + y)$$
$$\Rightarrow (u = x + y + z) \wedge (w = x + y) \Rightarrow (u = w + z)$$

Die zweite Klausel ist erfüllt weil $0 \in \mathbb{Z}$ neutrales Element bezüglich der Addition ist. Wir wählen deshalb $x = 0$ und $z = -y$, damit

$$0 = y + (-y)$$
$$0 = (-y) + y$$

Also ist auch die zweite Klausel erfüllt und $(\mathbb{Z}, +)$ ist ein Modell für ϕ.

Lösung:
Wir reduzieren das "Wortproblem" von $\mathrm{Th}(\mathbb{Z}, +)$ auf das Wortproblem von $\mathrm{Th}(\mathbb{N}, +)$, von welchem wir wissen, dass es entscheidbar ist. Sei also ϕ eine prädikatenlogische Formel in $\mathrm{Th}(\mathbb{Z}, +)$. Wir trennen jedes vorkommende $z \in \mathbb{Z}$ in ϕ in einen "Positivteil" $z_1 \in \mathbb{N}$ und einen Negativteil $z_2 \in \mathbb{N}$ auf, es gelte also $z = z_1 - z_2$. Wir ersetzen jede Quantifizierung Qz in ϕ durch Qz_1Qz_2. Setzen wir für jede in ϕ vorkommende Variable x: $x = x_1 - x_2$, so können wir für ein Auftreten von $x = y + z$ in ϕ schreiben: $x_1 - x_2 = y_1 - y_2 + z_1 - z_2$. Dies lässt sich umformulieren zu

$$x_1 + y_2 + z_2 = y_1 + z_1 + x_2 \tag{1}$$

Da in $(\mathbb{N}, +)$ nur das Prädikat $P(x, y, z) :\Leftrightarrow x = y + z$ existiert müssen wir (1) umformulieren zu

$$\forall v \forall w \forall w : ((u = x_1 + v) \wedge (v = y_2 + z_2) \wedge (w = z_1 + x_2) \Rightarrow (u = y_1 + w))$$

Wir erhalten durch besagte Ersetzungsregeln eine Formel ϕ' welche genau dann in $(\mathbb{N}, +)$ liegt wenn ϕ in $(\mathbb{N}, +)$ liegt.

Komplexitätsklassen

Sei M eine TM mit Eingabealphabet Σ. Die Rechenzeit von M wird durch die Funktion $t_M : \Sigma^* \to \mathbb{N}_0$ beschrieben.

$$t_M(\alpha) = \begin{cases} \text{minimale Zahl von Rechenschritten,} & \text{falls } \alpha \in L(M) \\ \text{die } M \text{ braucht, um } \alpha \text{ zu akzeptieren,} & \\ 0, & \text{falls } \alpha \notin L(M). \end{cases}$$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:
DTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

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

Mit diesen Rechenzeiten kommen wir nun zu folgenden Komplexitätsklassen:

DTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine deterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

NTIME$(f(n)) =$
$\{L \subset \Sigma^* | L = L(M)$, wobei M eine nichtdeterministische TM mit $t_M(\alpha) = \mathcal{O}(f(|\alpha|))\}$

P $= \bigcup_{k=1}^{\infty} DTIME(n^k)$

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

EXPTIME $= \bigcup_{k=1}^{\infty} DTIME(2^{n^k})$

Aufgabe

Die Komplexitätsklasse **co-P** sei definiert als die Menge der Sprachen $\mathcal{L}$, deren Komplementsprache $\overline{\mathcal{L}}$ in der Komplexitätsklasse **P** liegt.
Erinnerung: Zu einer Sprache $\mathcal{L}$ über einem Alphabet Σ ist die Komplementsprache $\overline{\mathcal{L}} = \Sigma^* \setminus \mathcal{L}$.

Beweisen Sie: **co-P** $=$ **P**.

Aufgabe

Gegeben seien ein ungerichteter Graph $G = (V, E)$ mit Knoten $v \in V$ und Kanten $e = (v_1, v_2) \in E$ mit $v_1, v_2 \in V$
und zwei Farben A und B.
Beweisen Sie: Die Sprache 2-COLOR $=$
$\{G \mid G = (V, E)$ ungerichteter Graph mit $\exists$ totale Funktion $g : V \to \{A, B\} : \forall\, (v_1, v_2) \in E : g(v_1) \neq g(v_2)\}$ liegt in der Komplexitätsklasse **P**.

Aufgabe

Das Problem 2-SAT ist folgendermaßen definiert:

2-SAT

Gegeben eine in ihrer Größe polynomiell beschränkte aussagenlogische Formel F in konjuktiver Normalform, wobei jede Klausel genau 2 Literale enthält. F hat also die Form

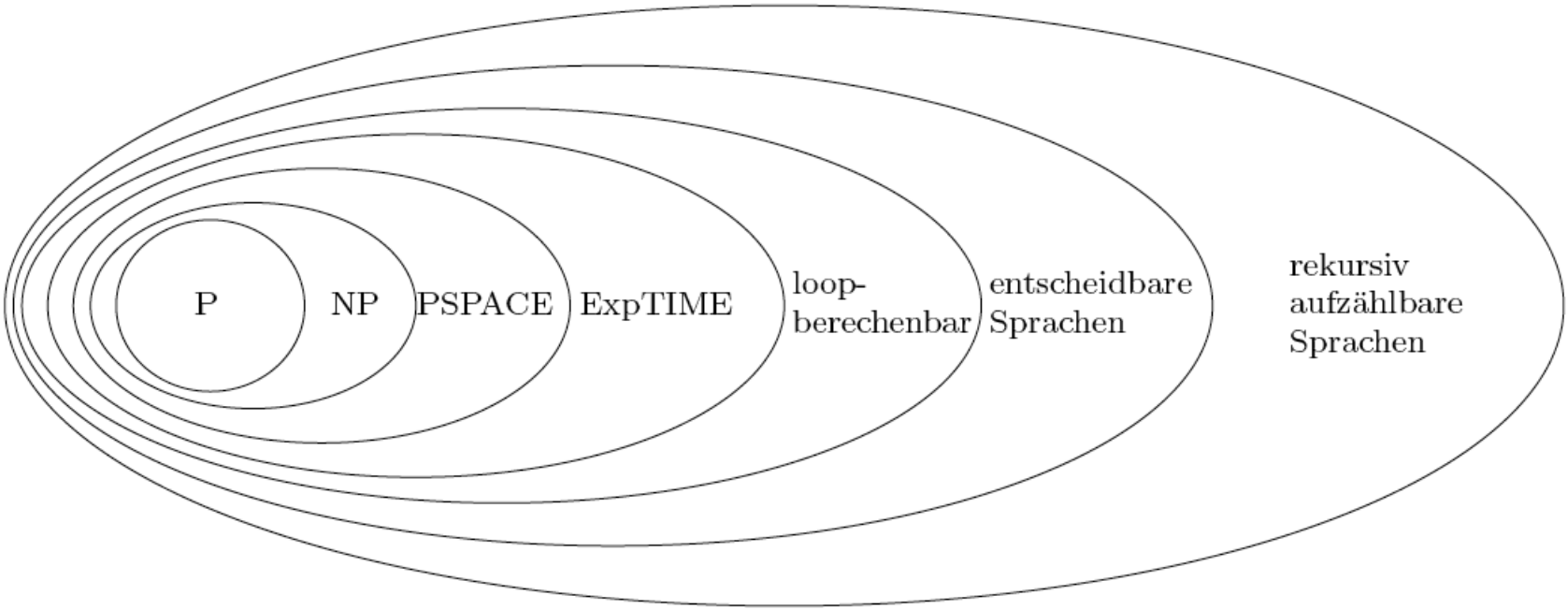
$$F = \Lambda_{i=1}^{n}(L_i \vee N_i),$$

wobei L_i und N_i Literale sind, also von der Form X oder $\neg X$ für eine Variable X sind.
Gibt es eine erfüllende Belegung für F?

Geben Sie eine polynomielle Reduktion von 2-COLOR auf 2-SAT an!

Zentrale Frage: $\mathcal{P} = \mathcal{NP}$?

Inklusion der Komplexitätsklassen



Definition: polynomial reduzierbar

Seien L_a und L_b Sprachen über Σ. Dann heißt L_a auf L_b polynomial reduzierbar, kurz $L_a \leq_p L_b$, falls es eine in polynomialer Laufzeit berechenbare totale Funktion $f : \Sigma^* \to \Sigma^*$ gibt mit $\forall a \in \Sigma^* : \alpha \in L_a \Leftrightarrow f(\alpha) \in L_b$.

Definition: polynomial reduzierbar

Seien L_a und L_b Sprachen über Σ. Dann heißt L_a auf L_b polynomial reduzierbar, kurz $L_a \leq_p L_b$, falls es eine in polynomialer Laufzeit berechenbare totale Funktion $f : \Sigma^* \to \Sigma^*$ gibt mit $\forall a \in \Sigma^* : \alpha \in L_a \Leftrightarrow f(\alpha) \in L_b$.

Lemma

Ist $L_a \leq_p L_b$ und L_b in $\mathcal{P}$ (oder $\mathcal{NP}$), ist auch L_a in $\mathcal{P}$ (oder $\mathcal{NP}$).

Definition: $\mathcal{NP}$-vollständig

Eine Sprache L heißt $\mathcal{NP}$-**vollständig**, falls gilt:

1. $L \in \mathcal{NP}$ und
2. für alle $L' \in \mathcal{NP}$ gilt: $L' \leq_p L$

SAT

Das Erfüllbarkeitsproblem **SAT** (satisfiability) ist das "erste" $\mathcal{NP}$-vollständige Problem (Steven Cook hat es 1971 bewiesen).

Gegeben: Menge U von Variablen, Menge C von Klauseln über U

Frage: Exisitert eine Wahrheitsbelegung von U, so dass C erfüllt wird, d.h. dass alle Klausel aus C den Wahrheitswert *wahr* annehmen?

Aufgabe

Geben Sie eine nicht-triviale Instanz von SAT an und überprüfen Sie, ob es eine Wahrheitsbelegung gibt.

Aufgabe

Geben Sie eine nicht-triviale Instanz von SAT an und überprüfen Sie, ob es eine Wahrheitsbelegung gibt.

Wer keine nicht-triviale Instanz im Kopf hat:

$$U = \{a, b, c, d, e, f\}$$

$$C = \{\bar{a} \vee \bar{b} \vee c \vee d \vee \bar{e} \vee f, \bar{a} \vee b \vee \bar{c} \vee \bar{d} \vee e \vee \bar{f}, \bar{a} \vee b \vee \bar{c} \vee d \vee \bar{e} \vee f,$$
$$a \vee b \vee \bar{c} \vee \bar{d} \vee e \vee \bar{f}, a \vee \bar{b} \vee c \vee d \vee \bar{e} \vee f, \bar{a} \vee b \vee \bar{c} \vee d \vee e \vee f,$$
$$a \vee b \vee c \vee d \vee e \vee f\}$$

Aufgabe

Wie könnte eine NTM, die das Entscheidungsproblem von SAT löst, arbeiten? Schätzen Sie grob den Arbeitsaufwand ab.

Aufgabe

Überlegen Sie, wie eine DTM, die das Entscheidungsproblem von SAT löst, arbeiten könnte. Schätzen Sie grob den Arbeitsaufwand ab.

Reflexion

Was haben wir heute gelernt?

Reflexion

Was haben wir heute gelernt?

- Einführung in die Komplexitätstheorie erhalten.

Reflexion

Was haben wir heute gelernt?

- Einführung in die Komplexitätstheorie erhalten.
- Unterschied zwischen $\mathcal{P}$ und $\mathcal{NP}$.

Reflexion

Was haben wir heute gelernt?

- Einführung in die Komplexitätstheorie erhalten.
- Unterschied zwischen $\mathcal{P}$ und $\mathcal{NP}$.
- Polynomiale Transformierbarkeit kennen gelernt.

Noch Fragen?

Vorschau

Vorschau

- Nächster Termin am **14.01.2010**!

Bis zum nächsten Mal