Komplexitätstheorie

Kapitel 6: Zeitkomplexität, reloaded

Einleitung

Weitere Themen rund um Zeitkomplexität:

- Schaltkreiskomplexität: mehr Struktur unterhalb von P
- P-Härte
- Die polynomielle Hierarchie: mehr Struktur zwischen NP und PSpace

Kapitel 6

Schaltkreiskomplexität

Motivation

Wir betrachten Schaltkreise als alternatives Berechnungsmodell

Damit lassen sich weitere Komplexitätsklassen definieren

Dies ist von Bedeutung, denn es

- schafft mehr Struktur innerhalb von P und erfasst wichtige, natürliche Probleme
- liefert Modell für massiv parallele Berechnungen (mehrere Millionen Prozessoren)
- führt auf natürliche Weise nicht-Uniformität als wichtigen theoretischen Aspekt ein

Schaltkreise

Definition (Boolscher) Schaltkreis

Boolscher Schaltkreis C ist Tupel $(V, E, \omega, x_1, \ldots, x_n, o)$ wobei

- (V, E) gerichteter azyklischer Graph,
- $x_1, \ldots, x_n \in V$ *Eingabeknoten* mit Eingangsgrad 0
- $o \in V$ *Ausgabeknoten* mit Ausgangsgrad 0
- $\omega : V \setminus \{x_1, \ldots, x_n\} \to \{\neg, \wedge, \vee, 0, 1\}$ Knotenbeschriftung so dass
  - $\omega(v) = \neg$ impliziert Eingangsgrad$(v) = 1$
  - $\omega(v) \in \{\wedge, \vee\}$ impliziert Eingangsgrad$(v) = 2$
  - $\omega(v) \in \{0, 1\}$ impliziert Eingangsgrad$(v) = 0$

Bei *Eingabe* $w \in \{0, 1\}^n$ ist der *Wert* jedes Knoten (induktiv) in der offensichtlichen Weise definiert. Die *Ausgabe* $C(w)$ von C ist der Wert des Ausgabeknoten

Schaltkreise

Schaltkreis-Terminologie:

- die Nicht-Eingabeknoten eines Schaltkreises werden *Gates* genannt
- der Eingangsgrad von Gates wird *Fan-In* genannt

Größe $|C|$ von Schaltkreis C ist die Anzahl seiner Gates

Definition Boolsche Funktion

Sei $n \in \mathbb{N}$.

n*-äre Boolsche Funktion* ist Funktion $f : \{0,1\}^n \to \{0,1\}$.

Schaltkreis C mit n Eingabeknoten:
berechnet n-äre Boolsche Funktion f_C mit $f_C(w) = C(w)$ für alle $w \in \{0,1\}^n$

Schaltkreise und Boolsche Funktionen

Definition Schaltkreiskomplexität

Schaltkreiskomplexität einer Boolschen Funktion f ist $|C|$ für kleinsten Schaltkreis C mit $f_C = f$

Durch direktes Implementieren der Wertetabelle:
Jede n-äre Boolsche Funktion hat Schaltkreiskomplexität $2^{\mathcal{O}(n)}$

Es ist nicht immer möglich, polynomiell große Schaltkreise zu finden:

Theorem (Shannon)

Für alle $n > 1$ gibt es Boolsche Funktion $f : \{0,1\}^n \to \{0,1\}$ mit Schaltkreiskomplexität $> \frac{2^n}{10n}$.

Interessanterweise kennt man keine "natürliche" Boolsche Funktion, die mehr als linear viele Gates benötigt!

Kapitel 6

Schaltkreise und Sprachen

Schaltkreise und Sprachen

Erkennen von Sprache:

- Wir beschränken uns o.B.d.A. auf Sprachen $L \subseteq \{0,1\}^*$
  (Sprachen über anderen Alphabeten können “umkodiert” werden)
- Jeder Schaltkreis erkennt nur Eingaben fester Länge, darum verwenden wir *Familie* von Schaltkreisen $(C_n)_{n\in\mathbb{N}} = (C_1, C_2, \dots)$
  (ein Schaltkreis für jede Eingabelänge)

Definition Schaltkreise und Sprachen

Familie $(C_n)_{n\in\mathbb{N}}$ von Schaltkreisen *definiert* L wenn jedes C_n die Einschränkung von L auf Worte der Länge n definiert:

$$C_n(w) = 1 \text{ gdw. } w \in L \quad \text{für alle } w \in \{0,1\}^n$$

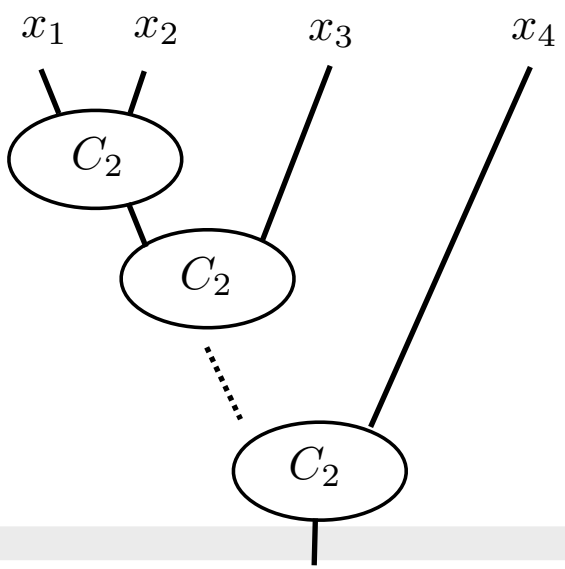
Schaltkreise und Sprachen

Paritäts-Beispiel leicht zu generalisieren zu Schaltkreisfamilie für

$$\text{PARITY} := \{w \in \{0,1\}^* \mid w \text{ hat Parität } 1\}$$

Im Detail:

- C_0: Konstantes 0-Gate liefert Ausgabe
- C_1: Ausgabe = (einziges) Eingabebit
- C_2: Schon gesehen
- $C > 2$: Zusammenschalten mehrerer Kopien von C_2:



Schaltkreise und Sprachen

Boolsche Funktion:

Ein einziger Schaltkreis, Schaltkreiskomplexität ist Zahl

Sprache:

Familie von Schaltkreisen, Schaltkreiskomplexität ist Funktion von Eingabelänge auf Schaltkreisgrösse
(Analog zu Zeit- und Platzkomplexität von TMs)

Definition Schaltkreiskomplexität von Sprachen

Sei $s : \mathbb{N} \to \mathbb{N}$ monoton wachsende Funktion.
Schaltkreisfamilie $(C_n)_{n\in\mathbb{N}}$ ist s-*größenbeschränkt* wenn $|C_n| \leq s(n)$ für alle $n \in \mathbb{N}$.

Definiere Komplexitätsklasse

$\text{Size}(s) := \{L \subseteq \Sigma^* \mid \exists \mathcal{O}(s)$-größenbeschränkte Familie $(C_n)_{n\in\mathbb{N}}$ von Schaltkreisen, die L definiert $\}$

Kapitel 6

Polynomielle Schaltkreiskomplexität und Uniformität

Polynomielle Schaltkreiskomplexität

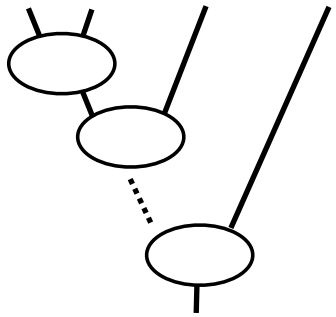
Jedes Gate eines Schaltkreises trägt “einen Schritt” zur Berechnung bei

In Analogie zu P ist es also natürlich, sich für polynomielle Schaltkreiskomplexität zu interessieren

Theorem

$\mathsf{P}_{/\mathsf{poly}} := \bigcup_{i \geq 1} \mathsf{Size}(n^i)$

Schon gesehen: PARITY $\in \mathsf{P}_{/\mathsf{poly}}$



Polynomielle Schaltkreiskomplexität

Essentieller Unterschied zwischen P und $\mathsf{P}_{/\mathsf{poly}}$:

- Jeder Sprache in P liegt eine TM zugrunde, die für Eingaben jeder Länge verwendet wird (uniformes Berechnungsmodell)
- Schaltkreise aus Familie $(C_n)_{n\in\mathbb{N}}$ sind unabhängig, C_i kann ganz anders sein als C_{i+1} (nicht-uniformes Berechnungsmodell)

Insbesondere: Schaltkreisfamilie kein effektives Berechnungsmodell:

- bei Eingabe w mit $|w| = n$ müßte erstmal C_n berechnet werden
- unsere Definition garantiert diese Berechenbarkeit aber nicht.

Theorem

$\mathsf{P}_{/\mathsf{poly}}$ enthält unentscheidbare Probleme.

Also trivialerweise $\mathsf{P}_{/\mathsf{poly}} \neq \mathsf{P}$.

Uniformität

Definition Polyzeit-Uniformität

Familie $(C_n)_{n\in\mathbb{N}}$ ist *polyzeit-uniform*, wenn es polyzeit-beschränkte DTM gibt, die bei Eingabe 1^n den Schaltkreis C_n ausgibt.

Uniform $\mathsf{P}_{/\mathsf{poly}}$ ist definiert wie $\mathsf{P}_{/\mathsf{poly}}$, aber mit polyzeit-uniformen Familien

Intuitiv ist uniform $\mathsf{P}_{/\mathsf{poly}}$ sehr ähnlich zu P:

deterministische Modelle mit polynomiellen Ressourcen (Zeit und Platz)

In der Tat gilt:

Theorem

P = uniform $\mathsf{P}_{/\mathsf{poly}}$

Schaltkreise liefern uns also eine alternative Charakterisierung von P.

Uniform P/poly

Uniform $\mathrm{P}_{/\mathrm{poly}} \subseteq \mathrm{P}$ ist offensichtlich: verwende Schaltkreisauswertung.

Definition CVP

Das *Schaltkreisauswertungsproblem (Circuit Value Problem, CVP)*:

$$\mathrm{CVP} := \{(C, w) \mid C \text{ } n\text{-ärer Schaltkreis}, w \in \{0,1\}^n, C(w) = 1\}$$

Leicht zu sehen: CVP ist in P (Azyklizität ausnutzen)

Theorem

Uniform $\mathrm{P}_{/\mathrm{poly}} \subseteq \mathrm{P}$.

Uniform P/poly

Theorem

P $\subseteq$ uniform $\mathsf{P}_{/\mathsf{poly}}$

Ideen:

- Sei $L \in$ P, M p-zeitbeschränkte DTM M mit $L(M) = L$
- Für jede Eingabelänge n, konstruiere in Polyzeit Schaltkreis C_n so dass: M akzeptiert $w \in \{0,1\}^n$ gdw. $C_n(w) = 1$
- Stelle Berechnung wieder als $(p(n)+2) \times (p(n)+1)$-Matrix dar:

$\triangleright$	q_0, a_0	a_1	$\cdots$	a_n	$\perp$	$\cdots$	$\perp$
$\triangleright$	b	q, a_1	$\cdots$	a_n	$\perp$	$\cdots$	$\perp$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

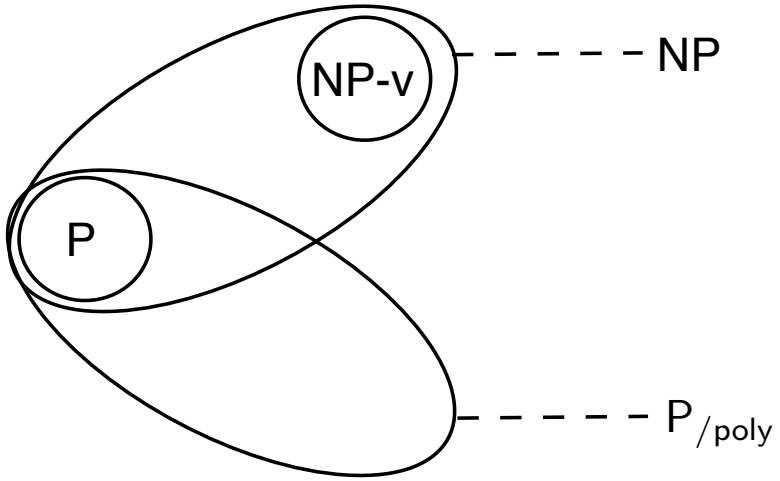
- Kodiere jeden möglichen Feldinhalt mittels $c = |\Gamma| + |Q|$ bits

Uniformität

Wir haben gesehen:

(nicht-uniformes) $P_{/poly}$ enthält Probleme, die weder in P noch in NP sind

Interessanterweise ist unbekannt,
ob $NP \subseteq P_{/poly}$



Man vermutet, dass Uniformität ein irrelevanter Aspekt für P vs NP ist, nämlich dass $NP \subseteq P$ gdw. $NP \subseteq P_{/poly}$.

Es gibt konkrete technische Resultate, die dafür Indizien liefern

Kapitel 6

Die Klasse NC und Massiv Parallele Berechnungen

Massiv Parallele Berechnungen

Massiv parallele Rechenmodelle:

- sehr viele (hundertausende) sehr einfache Prozessoren
- Prozessoren arbeiten unabhängig, kommunizieren über direkte Links oder Bus

Schaltkreise

- erlauben Parallelität, denn Gates (=Prozessoren), die sich wechselseitig nicht erreichen können, arbeiten unabhängig
- taugen daher als abstraktes Modell für massiv parallele Berechnungen (massiv: die Anzahl der Gates/Prozessoren steigt mit Eingabelänge!)
- wenn man Rechenzeit jedes Prozessors mit 1 ansetzt ist Rechenzeit des Schaltkreises C dessen Tiefe $d(C)$ (Länge des längsten Pfades)

Massiv Parallele Berechnungen

Ziel von parallelen Berechnungen:

Rechenzeit signifikant verkürzen, inbesondere exponentieller Speedup von linearer Zeit auf logarithmische Zeit

Beachte:

- Eine TM kann in logarithmischer Zeit nicht mal die Eingabe lesen
- Ein Schaltkreis braucht die Eingabe gar nicht (sequentiell) lesen, bekommt sie parallel zur Verfügung gestellt

Man interessiert sich also für Schaltkreise mit logarithmischer Tiefe und polynomieller Größe (damit Anzahl Prozessoren nicht absurd wird)

NC

Definition LogSpace-Uniformität

Familie $(C_n)_{n\in\mathbb{N}}$ ist *LogSpace-uniform*, wenn es LogSpace-Transduktor gibt, der bei Eingabe 1^n den Schaltkreis C_n ausgibt.

Hier ist eine entsprechende Komplexitätsklasse

Definition NC

Problem L ist in NC^i, $i \geq 1$, wenn es LogSpace-uniforme Familie $(C_n)_{n\in\mathbb{N}}$ gibt, die L erkennt und so dass:

- es gibt $k \in \mathbb{N}$ mit $|C_n| \in \mathcal{O}(n^k)$
- $d(C_n) \in \mathcal{O}(\log(n)^i)$

Nun ist NC $:= \bigcup_{i\geq 0}$ NC^i

NC steht für Nick's Class, nach Nicolas Pippenger

NC

Lemma

PARITY $\in NC^1 \subseteq NC$

Beachte: $NC^1 \subseteq NC^2 \subseteq \cdots NC$ ist unendliche Hierarchie in NC (und in P!)

Echtheit der Inklusionen unbekannt!

Die gesamte unendliche NC-Hierarchie ist trivialerweise unterhalb von P:

Theorem

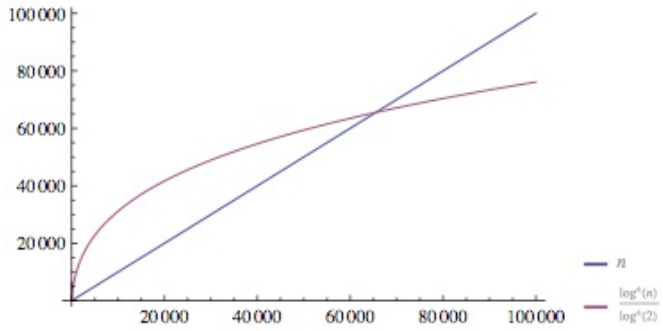
$NC \subseteq$ uniform $P_{/poly} \subseteq P$

NC

$L \in$ NC with oft mit "L effizient parallelisierbar" gleichgesetzt.

Etwas Vorsicht ist aber geboten:

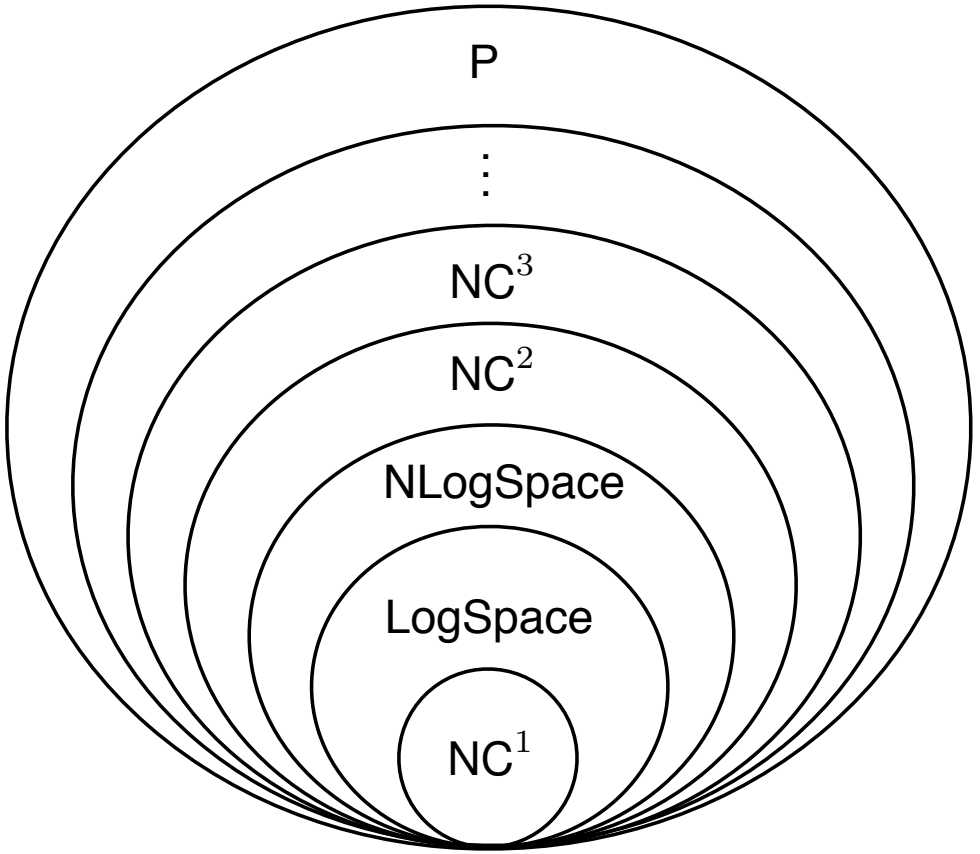
- Schon für recht kleine Werte von i (z.B. $i = 4$) wächst $\log(n)^i$ nur für sehr größe Werte von n deutlich langsamer als n



- Für sehr große Eingaben ist aber die Annahme "Poly viele Prozessoren" unrealistisch
- Also wäre NC^1 oder NC^2 vielleicht realistischer (das ist aber sehr kleine Klasse)

NC

Wir setzen nun NC in Beziehung zu unseren bisherigen Klassen:



NC

Theorem

$\mathrm{NC}^1 \subseteq \mathrm{LogSpace}$

Vorbemerkung:

Wir repräsentieren Knoten in Schaltkreisen als binäre Zahlen;
w.l.o.g. ist Knoten 1 der Ausgabeknoten

Wenn M LogSpace-Transduktor ist, der Schaltkreisfamilie $(C_n)_{n \in \mathbb{N}}$ konstruiert, dann kann folgendes in LogSpace berechnet werden:

- gegeben (k, n) mit $n \geq 0$ und k Knoten in C_n, den Typ von k in C_n
- gegeben (k, j, n) mit $n \geq 0$, k Knoten in C_n und $j \in \{1, 2\}$, den j-ten Nachfolger von k in C_n (oder $\perp$, wenn er nicht existiert)

(NB: Die Umkehrung gilt ebenfalls!)

NC

Theorem

NLogSpace $\subseteq$ NC^2 $\subseteq$ NC

Idee für Konstruktion von Schaltkreis C_n:

- Wir repräsentieren Konfigurationen durch uqv für Arbeitsband plus Kopfposition auf Eingabeband (aber nicht dessen Inhalt)
- Dann ist Konfigurationsmenge nur von n abhängig, nicht von genauer Eingabe (genauso wie C_n)
- Der Schaltkreis berechnet den transitiven Abschluss von "$\vdash_M$" auf dieser Konfigurationsmenge, ähnlich wie im Satz von Savitch
- Nur Basisfall $\alpha \vdash_M \alpha'$ hängt von Eingabe ab

Kapitel 6

Die Klasse AC

AC

Es gibt eine weitere Hierarchie

$$\mathrm{AC}^0 \subseteq \mathrm{AC}^1 \subseteq \cdots \subseteq \mathrm{AC} \subseteq \mathrm{P}$$

die exakt wie NC definiert ist, ausser, dass bei $\wedge$- und $\vee$-Gates das Fan-in unbeschränkt ist.

Interessanterweise konnte folgendes negatives Resultat bewiesen werden (ohne Beweis):

Theorem

$\mathrm{PARITY} \notin \mathrm{AC}^0$

Daraus folgt offensichtlich $\mathrm{AC}^0 \subsetneq \mathrm{NC}^1$, also auch $\mathrm{AC}^0 \subsetneq \mathrm{P}$.

NC / AC

Weitere Probleme in NC / AC:

- Erreichbarkeit in gerichteten Graphen ist in NC (denn in NLogSpace)
- Das Auswertungsproblem für AL-Formeln ist in NC1

Kapitel 6

P-Härte

P-Härte

Identifizierte Teilklassen von P werfen neue Fragen auf:

- Gilt P = NC, also: ist jedes Polyzeitproblem effizient parallelisierbar?
- Gilt sogar P = LogSpace?

Beides ist unbekannt, aber man vermutet, dass das nicht der Fall ist.

Um Kandidaten für "echte" P-Probleme zu finden, benötigen wir

Begriffe von Härte und Vollständigkeit für P

Polynomialzeit-Reduktionen sind hier nicht sinnvoll, da

für alle $L, L' \in$ P mit L' nicht-trivial: $L \leq_p L'$

(*nicht-trivial*: es gibt positive Instanzen und negative Instanzen)

P-Härte

Definition P-Härte, P-Vollständigkeit

Problem L ist

- *P-hart* wenn $L' \leq_{\log} L$ für alle $L' \in$ P;
- *P-vollständig* wenn L P-hart und in P.

Also: wenn Problem L P-vollständig, dann

1. L nicht in LogSpace, außer wenn LogSpace = P

2. L nicht in NC (= nicht effizient parallelisierbar), außer wenn NC = P

Für 2. brauchen wir allerdings noch (ohne Beweis):

Theorem

Wenn $L \in$ NC und $L' \leq_{\log} L$, dann $L' \in$ NC.

P-Härte

Das Circuit Value Problem ist das "protoytpische" P-vollständige Problem:

Theorem (Ladner)

CVP ist P-vollständig.

Weitere P-vollständige Probleme z.B.:

- das Leerheitsproblem für kontextfreie Grammatiken
- monotones CVP (Schaltkreise ohne Negation)
- Linear programming (= Integer Programming mit rationalen Lösungen)
- Erfüllbarkeit von AL-Formeln in Horn-Form

$$p_1 \wedge \cdots \wedge p_n \rightarrow p, \qquad p_1 \wedge \cdots \wedge p_n \rightarrow \bot, \qquad p$$

Kapitel 6

Die polynomielle Hierarchie

PH

(N)LogSpace, NC, AC, etc: reiche Struktur innerhalb von P

Die polynomielle Hierarchie liefert Struktur zwischen P und PSpace

Wichtiges Problem für Schaltkreisentwurf:

Definition Minimal Circuit (MC)

Schaltkreis C ist minimal wenn $|C'| \geq |C|$ für alle C', die *äquivalent* zu C sind, also gleiche Anzahl n von Eingabebits und

$$C(w) = C'(w) \text{ für alle } w \in \{0,1\}^n$$

MC ist Menge aller minimalen Schaltkreise.

Was ist die ”richtige” Komplexitätsklasse (Vollständigkeit!) für dieses Problem?

PH

Offensichtliches “Teilproblem” ist CEQ $:= \{(C, C') \mid C$ äquivalent zu $C'\}$

Lemma

CEQ ist co-NP-vollständig.

Betrachte wieder $\overline{\text{MC}}$:

- Ist in NP wenn wir einen CEQ-Algorithmus als Unterprozedur ohne Zeitverbrauch verwenden
- Wir können beide Algorithmen nicht zu einem NP-Algorithmus vereinigen, weil der NP-Algorithmus einen co-NP-Algorithmus aufruft ($\exists\forall$-Charakteristik)

Derartige Probleme sind offensichtlich in PSpace. Man kann deren Komplexität aber noch exakter bestimmen

Orakel

Orakel: Unterprogramm, dessen Zeitverbrauch ausgeblendet wird, dargestellt als formale Sprache

Definition Orakel-TM

Eine *Orakel-TM (OTM)* M^O ist eine (deterministische oder nicht-deterministische) TM M ausgestattet mit einem Orakel $O \subseteq \Sigma^*$. OTM hat

- ein zusätzliches *Orakelband*
- drei spezielle Zustände $q_?, q_+, q_-$.

Für q_+ und q_- sind normale Transitionen definiert. Der Folgezustand von $q_?$ ist q_+ wenn das momentane Wort auf dem Orakelband in O ist und q_- sonst. Kopfposition und Bandinhalte bleiben dabei unverändert.

Also schon gesehen: $\overline{\text{MC}}$ wird von Polyzeit-beschränkter ONTM akzeptiert wenn O=CEQ

Orakel

Orakel-TM ist ebensowenig realistisches Berechnungsmodell wie nicht-deterministische TMs

Dennoch können mittels Orakel-TMs natürliche Komplexitätsklassen definiert werden (natürlich = erfassen viele natürliche Probleme)

Orakel-TMs können auch verwendet werden, um komplexitäts-theoretische Annahmen zu formalisieren, z.B.:

- wenn SAT in konstanter Zeit lösbar wäre, welche anderen Problem wären dann effizient lösbar (in Polyzeit mit SAT-Orakel)?
- wenn das Halteproblem für Turingmaschinen H entscheidbar wäre, welche anderen Probleme wären dann entscheidbar (von TM mit H-Orakel)

Orakel

Definition Orakel-Komplexitätsklassen

Sei $O \subseteq \Sigma^*$ ein Orakel. Dann:

- $\mathrm{P}^O := \{L \mid L$ wird von ODTM M^O in poly-Zeit entschieden $\}$
- $\mathrm{NP}^O := \{L \mid L$ wird von ONTM M^O in poly-Zeit entschieden $\}$

Sei $\mathcal{C}$ Komplexitätsklasse. Dann:

$$\mathrm{P}^{\mathcal{C}} := \bigcup_{O \in \mathcal{C}} \mathrm{P}^O \qquad \mathrm{NP}^{\mathcal{C}} := \bigcup_{O \in \mathcal{C}} \mathrm{NP}^O$$

Schon gezeigt: $\overline{\mathrm{MC}} \in \mathrm{NP}^{\mathrm{CEQ}}$, also in $\overline{\mathrm{MC}} \in \mathrm{NP}^{\text{co-NP}}$

Leicht zu sehen: $\mathrm{NP}^{\text{co-NP}} = \mathrm{NP}^{\mathrm{NP}}$

PH

Einige Beispiele:

- $\mathrm{P}^{\mathrm{P}} = \mathrm{P}$, $\mathrm{NP}^{\mathrm{P}} = \mathrm{NP}$;

  (Integriere Orakel in OTM)

- $\mathrm{NP}^{\mathrm{NP}} = \mathrm{NP}$ ist hingegen nicht klar, denn co-NP $\subseteq \mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{NP}^{\mathrm{NP}}$

- $\mathrm{P}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{SAT}}$ und ebenso für jedes andere NP-vollständige Problem

  (Für $\mathrm{P}^{O} \subseteq \mathrm{P}^{\mathrm{SAT}}$ mit O NP-vollständig, integriere Reduktion $O \leq_p$ SAT in OTM)

- co-$\mathrm{NP}^{\mathcal{C}}$ verwenden wir für $\overline{\mathrm{NP}^{\mathcal{C}}}$

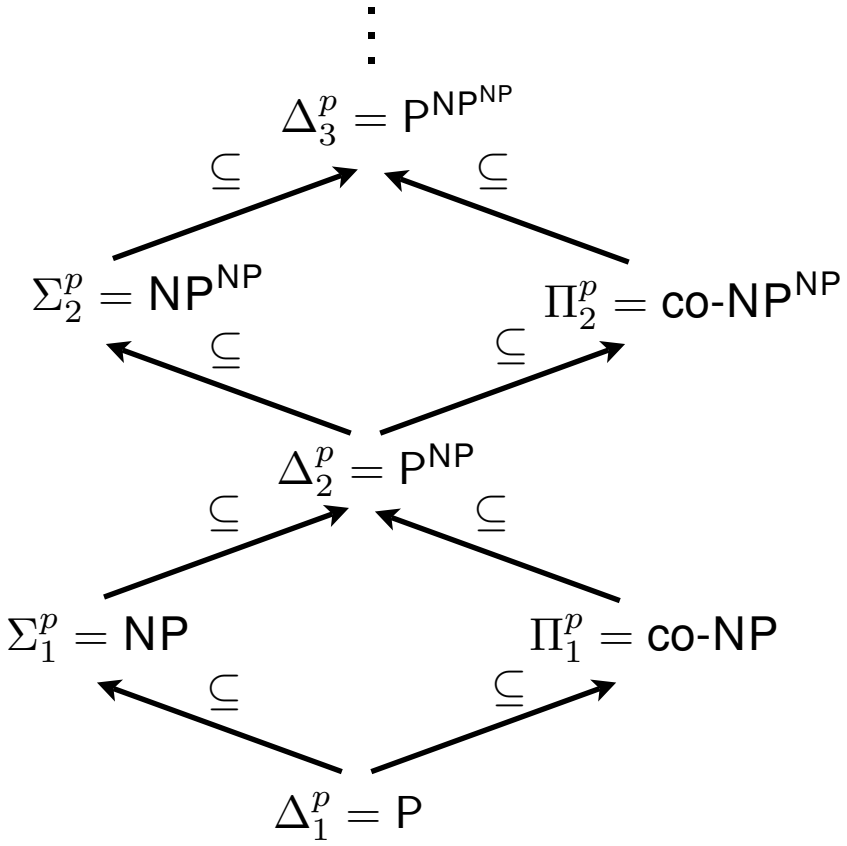
PH

Die polynomielle Hierarchie entsteht nun durch iteriertes Orakel-anwenden

Definition Polynomielle Hierarchie

- $\Sigma_1^p = \text{NP}$, $\Pi_1^p = \text{co-NP}$, $\Delta_1^p = \text{P}$
- Für $k \geq 1$ sei
  - $\Sigma_{k+1}^p = \text{NP}^{\Sigma_k^p}$
  - $\Pi_{k+1}^p = \text{co-}\Sigma_{k+1}^p$
  - $\Delta_{k+1}^p = \text{P}^{\Sigma_k^p}$

PH



Echtheit der Inklusionen ist unbekannt.

Lemma

Für alle $k \geq 1$ gilt: $\Delta_k^p \subseteq \Sigma_k^p \subseteq \Delta_{k+1}^p$ und $\Delta_k^p \subseteq \Pi_k^p \subseteq \Delta_{k+1}^p$

PH

Es gibt auch eine Klasse für die gesamt polynomielle Hierarchie:

Definition Polynomielle Hierarchie

$\mathrm{PH} = \bigcup_{k \geq 1} \Sigma_k^p$

Die polynomielle Hierarchie liegt zwischen P und PSpace:

Theorem

$\mathrm{PH} \subseteq \mathrm{PSPACE}$

PH

Viele Resultate in der Komplexitätstheorie beziehen sich auf die Echtheit der Inklusionen in der polynomiellen Hierarchie

Die polynomielle Hierarchie kollabiert wenn PH $= \Sigma_k^p$ für ein $k \geq 1$

Lemma

Wenn $\Sigma_k^p = \Sigma_{k+1}^p$, dann PH $= \Sigma_k^p$.

Also: $\Sigma_{k-1}^p \neq \Sigma_k^p$ schwächere Annahme als $\Sigma_k^p \neq \Sigma_{k+1}^p$

und P $\neq$ NP schwächste aller dieser Annahmen

Anders formuliert: PH kollabiert am ehesten weit oben!

Theorem

Wenn PH $=$ PSPACE, dann kollabiert PH.

Kapitel 6

Logische Charakterisierung der Polynomiellen Hierarchie

Charakterisierung PH

Folgende Charakterisierung generalisiert Definition von NP

Theorem

$L \in \Sigma_k^p$ gdw. es Polynom q und $L' \in$ P gibt so dass

$$L = \{w \mid \exists u_1 \in A. \forall u_2 \in A. \exists u_3 \in A \ldots Q u_k \in A : (w, u_1, \ldots, u_k) \in L'\}.$$

wobei $A = \{0,1\}^{q(|w|)}$ und Q der sich durch Alternierung ergebende Quantor.

Die Klassen der polynomiellen Hierarchie werden also mittels logischer Ausdrückbarkeit beschrieben

Frage nach Echtheit der Inklusionen in PH: liefern zusätzliche Quantoren-alternierungen zusätzliche Ausdrucksstärke?

Charakterisierung PH

Lemma

Für $L \subseteq \Sigma^*$ gilt $L \in \Sigma_k^p$ gdw. es gibt Polynom p und Relation $R \subseteq \Sigma^* \times \Gamma^*$ so dass

- $(w, b) \in R$ impliziert $|b| \leq p(|w|)$
- $R \in \Pi_{k-1}^p$ (wobei $\Pi_0^p := \mathrm{P}$)
- $L = \{w \mid \exists b : (w, b) \in R\}$

Idee:

- Induktion über k
- Der Fall $k = 1$ folgt direkt aus Definition NP
- In "$\Rightarrow$" ist der Beweis b eine Berechnung der NTM zusammen mit Beweisen für die "ja"-Antworten des Orakels (induktiv)

Charakterisierung PH

Korollar

Für $L \subseteq \Sigma^*$ gilt $L \in \Pi_k^p$ gdw. es gibt Polynom p und Relation $R \subseteq \Sigma^* \times \Gamma^*$ so dass

- $(w, b) \in R$ impliziert $|b| \leq p(|w|)$
- $R \in \Sigma_{k-1}^p$ (wobei $\Sigma_0^p :=$ P)
- $L = \{w \mid \forall b \in \Gamma^*$ mit $|b| \leq p(|w|) : (w, b) \in R\}$

Beweis: Für $\overline{L} \in \Sigma_k^p$ gibt es R wie in vorigem Lemma, verwende für L:

$$\widehat{R} := \{(w, b) \in \Sigma^* \times \Gamma^* \mid (w, b) \notin R \text{ und } |b| \leq p(|w|)\}$$

Aus Lemma + Korollar folgt nun das ursprüngliche Theorem:
Ersetze wiederholt Σ_i^p und Π_i^p durch ihre Beweissysteme

Kapitel 6

Härte und Vollständigkeit in der polynomiellen Hierarchie

Vollständigkeit

Um Probleme korrekt in die polynomielle Hierarchie "einzuordnen", brauchen wir Vollständigkeitsbegriff

Definition

Für $k \geq 1$ ist Problem L

- Σ_k^p-*hart* wenn $L' \leq_p L$ für alle $L' \in \Sigma_k^p$;
- *NP-vollständig* wenn L sowohl Σ_k^p-hart als auch in Σ_k^p.

Für Π_k^p, Δ_k^p und PH analog (ausser für $\Delta_1^p = \mathrm{P}$)

Aber PH hat wahrscheinlich keine vollständigen Probleme:

Lemma

Wenn für PH vollständige Probleme existieren, kollabiert die Hierarchie

Vollständigkeit

QBF liefert uniforme Familie von "typischen" vollständigen Problemen

Für $\overline{V} = v_1, \ldots, v_n$ schreiben wir $\exists\overline{V}$ als Abkürzung für $\exists v_1 \cdots \exists v_n$

$\forall\overline{V}$ als Abkürzung für $\forall v_1 \cdots \forall v_n$

Definition k-QBF

QBF $Q_1\overline{V_1} \cdots Q_n\overline{V_n}\varphi$ heisst k-*QBF* wenn

- $n = k$
- $Q_1 = \exists$, $Q_2 = \forall$, $Q_3 = \exists$, etc. (Quantoren alternieren)

QBF_k ist die Menge aller gültigen k-QBFs.

Beispiel für 3-QBF: $\exists v_1 \exists v_2 \forall v_3 \exists v_4 \exists v_5 . \varphi$

Vollständigkeit

Theorem

Für alle $k \geq 1$ ist QBF$_k$ Σ_k^p-vollständig.

Idee:

- "in Σ_k^p": benutze logische Charakterisierung
- Härte: benutze logische Charakterisierung und Übersetzung von TM in AL-Formel analog zum Beweis von Cook's Theorem

Beginnt man die Quantorenalternierung mit "$\forall$", so ist k-QBF Π_k^p-vollständig.

Vollständigkeit

In der Logik gibt es verschiedene natürliche Probleme, die vollständig für Klassen der polynomiellen Hierarchie sind.

Definition MINSAT

Für zwei WZen π und π' schreiben wir $\pi \leq \pi'$ gdw.

$$\pi'(v) = 1 \text{ impliziert } \pi(v) = 1 \text{ für alle Variablen } V$$

π ist *minimales Modell* von AL-Formel φ wenn

- π erfüllt φ
- für alle π', die φ erfüllen, gilt $\pi \leq \pi'$

MINSAT ist die Menge aller Tripel (φ, v) mit φ AL-Formel und v Variable so daß $\pi(v) = 0$ in allen minimalen Modellen von φ.

Theorem

MINSAT ist Π_2^p-vollständig.

Vollständigkeit

Weiteres natürliches vollständiges Problem z.B.:

Äquivalenzproblem für kontextfreie Grammatiken über 1-elementigen (Terminal-)Alphabeten ist Π_2^p-vollständig.

Es wird vermutet, dass MC (Minimial Circuit) ebenfalls Π_2^p-vollständig ist, die Härte konnte aber bisher nicht bewiesen werden!

Für Klassen weit oben in der polynomiellen Hierarchie scheint es nur sehr wenig "natürliche" vollständige Probleme zu geben

Kapitel 6

Wie viele Klassen gibt es eigentlich?

Mehr Komplexitätsklassen

Abgesehen von den angegebenen Büchern:

http://qwiki.stanford.edu/wiki/Complexity_Zoo

http://www.math.ucdavis.edu/~greg/zoology/

Vollständigkeit für PH