Komplexitätstheorie

SoSe 2018
Thomas Schneider

Kapitel 6: Schaltkreise

Homepage der Vorlesung: `http://tinyurl.com/ss18-kt`

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 6.1 (Schaltkreis)

Ein *(Boolescher) Schaltkreis* C ist Tupel $(V, E, \omega, x_1, \ldots, x_n, o)$ mit

- (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:
  - Wenn $\omega(v) = \neg$, dann Eingangsgrad$(v) = 1$.
  - Wenn $\omega(v) \in \{\wedge, \vee\}$, dann Eingangsgrad$(v) = 2$.
  - Wenn $\omega(v) \in \{0, 1\}$, dann Eingangsgrad$(v) = 0$.

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

T6.1

Schaltkreise

Schaltkreis-Terminologie:

- Die Nicht-Eingabeknoten eines Schaltkreises werden *Gates* genannt.
- Der Eingangsgrad von Gates wird *Fan-In* genannt.

Die *Größe* $|C|$ eines Schaltkreis C ist die Anzahl seiner Gates.

Definition 6.2 (Boolesche Funktionen und Schaltkreise)

Eine *n-äre Boolesche Funktion (BF)* ist eine Funktion $f : \{0,1\}^n \to \{0,1\}$.

Ein Schaltkreis C mit n Eingabeknoten berechnet die n-äre BF f_C
mit $f_C(w) = C(w)$ für alle $w \in \{0,1\}^n$.

T6.2

Schaltkreise und Boolesche Funktionen

Definition 6.3 (Schaltkreiskomplexität)

Die *Schaltkreiskomplexität* einer Booleschen Funktion f ist $|C|$ für kleinsten Schaltkreis C mit $f_C = f$.

T6.3

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

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

Theorem 6.4 (Shannon 1949)

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

T6.4

Interessanterweise kennt man keine „natürliche“ Boolesche Funktion, die mehr als linear viele Gates benötigt!

Kapitel 6

NEXT

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 6.5 (Schaltkreise und Sprachen)

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

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

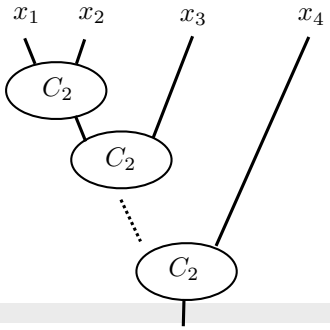
Schaltkreise und Sprachen

Paritäts-Beispiel (T6.1) 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 (T6.1)
- C_n, $n > 2$: Zusammenschalten mehrerer Kopien von C_2 :



Schaltkreise und Sprachen

Boolesche Funktion berechnen:
Ein einziger Schaltkreis; Schaltkreiskomplexität ist Zahl

Sprache erkennen:
Familie von Schaltkreisen;
Schaltkreiskomplexität ist Funktion von Eingabelänge auf Schaltkreisgröße
(analog zu Zeit- und Platzkomplexität von TMs)

Definition 6.6 (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

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

Kapitel 6

Polynomielle Schaltkreiskomplexität

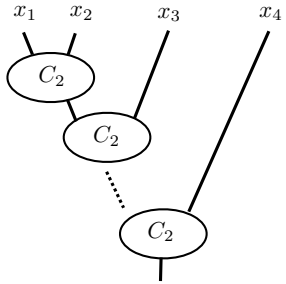
Jedes Gate eines Schaltkreises trägt „einen Schritt“ zur Berechnung bei.

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

Definition 6.7

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

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



Polynomielle Schaltkreiskomplexität

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

- 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 sind Schaltkreisfamilien kein effektives(!) Berechnungsmodell:

- Bei Eingabe w mit $|w| = n$ müsste zunächst C_n berechnet werden.
- Unsere Definition garantiert diese Berechenbarkeit aber nicht.

Theorem 6.8

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

T6.5

Also trivialerweise: $\text{P}_{/\text{poly}} \neq \text{P}$

Uniformität

Definition 6.9 (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-P$_{/\text{poly}}$ ist definiert wie P$_{/\text{poly}}$, aber mit polyzeit-uniformen Familien.

Intuitiv ist Uniform-P$_{/\text{poly}}$ sehr ähnlich zu P:

deterministische Modelle mit polynomiellen Ressourcen (Zeit und Platz)

In der Tat gilt:

Theorem 6.10

P = Uniform-P$_{/\text{poly}}$

Schaltkreise liefern uns also eine alternative Charakterisierung von P.

Uniform P$_{/\text{poly}}$

Uniform-P$_{/\text{poly}}$ ⊆ P ist offensichtlich: verwende Schaltkreisauswertung

Definition 6.11 (CVP)

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

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

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

Theorem 6.12

Uniform-P$_{/\text{poly}}$ ⊆ P

T6.6

Uniform P$_{/\text{poly}}$

P ⊆ Uniform-P$_{/\text{poly}}$ ist aufwändiger: Ähnlichkeit mit Satz von Cook

Theorem 6.13

P ⊆ Uniform-P$_{/\text{poly}}$

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 $w = b_1 \cdots b_n \in \{0,1\}^n$ gdw. $C_n(w) = 1$
- Stelle Berechnung wieder als $\big((p(n)+2) \times (p(n)+1)\big)$-Matrix dar:

▷	q_0, b_1	b_2	$\cdots$	b_n	⊥	$\cdots$	⊥
▷	a	q, b_2	$\cdots$	a_n	⊥	$\cdots$	⊥
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- Kodiere jeden möglichen Inhalt einer Zelle mittels $c = |\Gamma| + |Q|$ Bits.

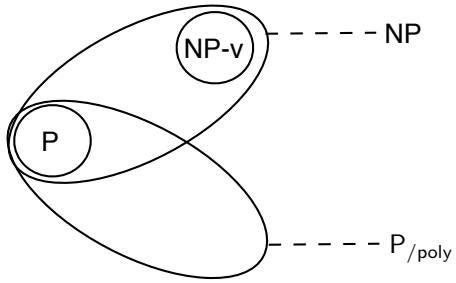
T6.7

Nicht-Uniformität

Wir haben gesehen:

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

Interessanterweise ist unbekannt, ob NP ⊆ P$_{/\text{poly}}$



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

Es gibt konkrete technische Resultate, die dafür Indizien liefern.

Kapitel 6

Massiv parallele Berechnungen

Massiv parallele Rechenmodelle:

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

Schaltkreise

- erlauben Parallelität:
  Gates (= Prozessoren), die sich gegenseitig nicht erreichen können, arbeiten unabhängig
- taugen daher als abstraktes Modell für massiv parallele Berechnungen („massiv“: Anzahl 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)

T6.8

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) zu 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 6.14 (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 6.15 (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 $\mathrm{NC} := \bigcup_{i\geq 0} \mathrm{NC}^i$

NC steht für „Nick’s Class“, nach Nicolas Pippenger

NC

Lemma 6.16

PARITY $\in$ NC^1 $\subseteq$ NC

T6.9

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 6.17

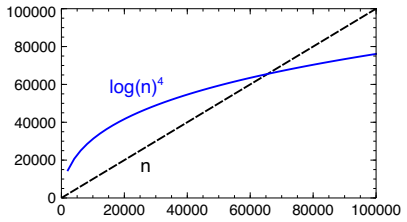
NC $\subseteq$ Uniform-P$_{/poly}$ $=$ P

NC und Parallelisierbarkeit

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

Etwas Vorsicht ist aber geboten:

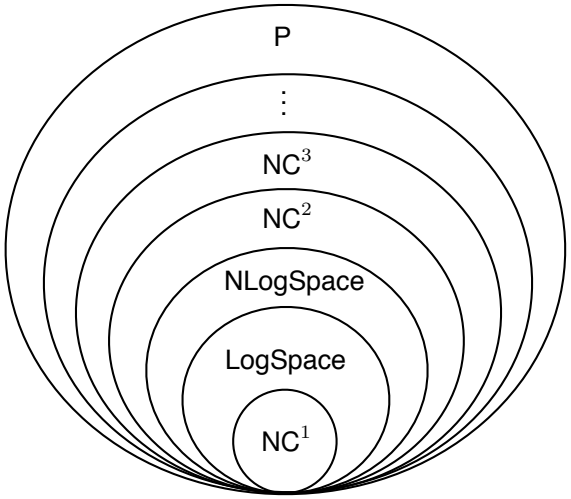
- Schon für recht kleine Werte von i (z. B. $i = 4$) wächst $\log(n)^i$ nur für sehr große Werte von n deutlich langsamer als n.



- Für sehr große Eingaben ist aber die Annahme „polynomiell viele Prozessoren“ unrealistisch.
- Also wäre NC^1 oder NC^2 vielleicht realistischer (das sind aber sehr kleine Klassen).

NC im Kontext

Wir setzen nun NC in Beziehung zu unseren bisherigen Klassen:



NC versus LogSpace

Theorem 6.18

NC^1 $\subseteq$ LogSpace

Vorbemerkung:

Wir repräsentieren Knoten in Schaltkreisen als binäre Zahlen;
o. B. d. A. 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)

(Die „Umkehrung“ gilt ebenfalls!)

T6.10

NC versus NLogSpace

Theorem 6.19

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

Idee für Konstruktion von Schaltkreis C_n:

- Wir repräsentieren Konfigurationen α 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 mittels "teile und herrsche" (ähnlich wie im Satz von Savitch)
- Nur Basisfall $\alpha \vdash_M \alpha'$ hängt von Eingabe ab

T6.11

Kapitel 6

AC

Es gibt eine weitere Hierarchie

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

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

T6.12

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

Theorem 6.20 (Furst, Saxe, Sipser 1981; Ajtai 1983)

PARITY $\notin$ AC^0

Daraus folgt offensichtlich AC$^0 \subsetneq$ NC^1, also auch AC$^0 \subsetneq$ 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 NC^1
  Samuel R. Buss.
  The Boolean Formula Value Problem Is in ALogTime.
  In Proc. of STOC, 1987: 123–131.
  http://doi.acm.org/10.1145/28395.28409
- Multiplikation, Division, Potenzieren, ganzer Zahlen in NC^1
  P. Beame, S. Cook, and J. Hoover.
  Log depth circuits for division and related problems.
  SIAM Journal on Computing 15:994–1003, 1986.
  https://doi.org/10.1137/0215070
- Erreichbarkeit in gitterförmigen Graphen ist in AC^0
  D. A. M. Barrington, C.-J. Lu, P. B. Miltersen, and S. Skyum.
  Searching constant width mazes captures the AC^0 hierarchy.
  In Proc. of STACS, 1998.
  https://eccc.weizmann.ac.il/eccc-reports/1997/TR97-044/

Kapitel 6

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 wieder 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 6.21 (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

T6.13

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

Theorem 6.22

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

(Beweis ähnlich der Abgeschlossenheit von LogSpace unter Komposition.)

P-Härte

Das *Circuit Value Problem* ist das „protoytpische“ P-vollständige Problem:

Theorem 6.23 (Ladner)

CVP ist P-vollständig.

T6.14

Weitere P-vollständige Probleme z. B.:

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

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

Kapitel 6

NP versus $P_{/poly}$: ein Zugang zu $P \neq NP$?

Vor ca. 30 Jahren hat man geglaubt, das Problem $P \stackrel{?}{=} NP$ lösen zu können

Ein Zugang über Schaltkreiskomplexität ...

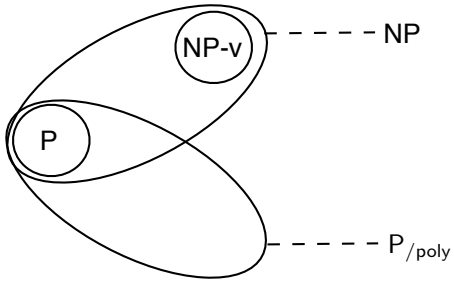
- wurde lange erforscht
- lieferte viele interessante und anspruchsvolle Resultate
- aber hat bisher nicht zum Ziel geführt

Es folgt: ein kurzer Abriss dieses Forschungsprogramms

NP versus $P_{/poly}$: ein Zugang zu $P \neq NP$?

Zur Erinnerung:
$P =$ uniform $P_{/poly} \subseteq P_{/poly}$

Also gilt:
wenn $NP \not\subseteq P_{/poly}$
dann $P \neq NP$



Man hat lange versucht, $NP \not\subseteq P_{/poly}$ zu zeigen

NP versus $P_{/poly}$: ein Zugang zu $P \neq NP$?

Definition 6.24 (schwere Funktionen)

Familie $(f_n)_{n\in\mathbb{N}}$ Boolescher Funktionen mit $f_n : \{0,1\}^n \to \{0,1\}$ ist *schwer,* wenn es kein $k \in \mathbb{N}$ gibt,
so dass die Schaltkreiskomplexität von f höchstens n^k ist.

Zur Erinnerung:

Theorem 6.4 (Shannon 1949)

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

Folgerung:

Es gibt schwere Funktionenfamilien.

(Es gilt sogar: Die meisten Funktionenfamilien sind schwer.)

NP versus $P_{/poly}$ mittels schwerer Funktionen

Versuch, $NP \not\subseteq P_{/poly}$ zu zeigen:
Finde schwere Funktionenfamilie, die in NP berechenbar ist.

Man begann, Funktionen zu finden,
die schwer bezüglich eingeschränkter Schaltkreisklassen sind:

- konstante Tiefe
- konstante Tiefe plus Zählgatter
- monotone Schaltkreise

Funktionenfamilie $(f_n)_{n\in\mathbb{N}}$ ist *schwer für Schaltkreisklasse $\mathcal{C}$,*
wenn $\mathcal{C}$-Schaltkreiskomplexität von $(f_n)_{n\in\mathbb{N}}$ größer als poly(n) ist,
d. h. $(f_n)_{n\in\mathbb{N}}$ kann nicht durch poly große $\mathcal{C}$-Schaltkreise berechnet werden.

Schaltkreise konstanter Tiefe

Zur Erinnerung:

Theorem 6.25 (Furst, Saxe, Sipser 1981; Ajtai 1983)

PARITY $\notin AC^0$

Das bedeutet: die Funktionenfamilie $(f_n)_{n\in\mathbb{N}}$ mit

$$f_n(x_1,\ldots,x_n) = \text{Parität von } x_1\cdots x_n$$

ist schwer für Schaltkreise konstanter Tiefe

Nächster Schritt: Untere Schranken für allgemeinere Schaltkreisklassen?

Schaltkreise konstanter Tiefe mit Zählgattern

Erlauben zusätzlich MOD_m-Gatter:
berechnen die Funktion $MOD_m(x_1,\ldots,x_n) = \begin{cases} 0, \text{falls } \sum_{i=1}^n x_i \equiv 0 \bmod m \\ 1, \text{sonst} \end{cases}$
(Klar: PARITY $= MOD_2$)

Definition 6.26 (ACC^0)

$ACC^0(m_1,\ldots,m_k) = \{L \mid L$ wird von einer Schaltkreisfamilie konstanter Tiefe und poly. Größe akzeptiert, die nur Gatter $\wedge, \vee, \neg, MOD_{m_1},\ldots,MOD_{m_k}$ verwendet$\}$

$ACC^0 = \bigcup_{k\geq 0}\bigcup_{m_1>1}\cdots\bigcup_{m_k>1} ACC^0(m_1,\ldots,m_k)$

T6.15

MOD_p ist schwer für SKe konstanter Tiefe mit MOD_q-Gattern (o. Bew.):

Theorem 6.27 (Razborov 1987; Smolensky 1987)

Für je zwei verschiedene Primzahlen p, q ist MOD_p nicht in $ACC^0(q)$.

monotone Schaltkreise

Betrachten Schaltkreisklassen ohne Größenbeschränkung:

Definition 6.28 (Monotonie)

Ein Schaltkreis heißt *monoton*, wenn er nur $\wedge$- und $\vee$-Gatter enthält.

Für alle n-Tupel $\overline{x}, \overline{y} \in \{0,1\}^n$ mit $\overline{x} = (x_1,\ldots,x_n)$ und $\overline{y} = (y_1,\ldots,y_n)$
gelte $\overline{x} \preceq \overline{y}$, wenn für alle $i \leq n$ gilt: $x_i = 1 \Rightarrow y_i = 1$.

Eine Boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ heißt *monoton*,
wenn für alle $\overline{x}, \overline{y} \in \{0,1\}^n$ gilt: $\overline{x} \preceq \overline{y} \Rightarrow f(\overline{x}) \leq f(\overline{y})$

Relativ leicht zu sehen:

1. Jeder monotone Schaltkreis berechnet eine monotone Funktion.
2. Jede monotone Fkt. kann von einem (hinreichend großen) monotonen Schaltkreis berechnet werden.

T6.16

CLIQUE braucht große monotone Schaltkreise

Betrachten Boolesche Funktion $\text{CLIQUE}_{k,n} : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ mit

$\text{CLIQUE}_{k,n}(\overline{x}) = 1 \Leftrightarrow$
Der Graph mit n Knoten, dessen Adjazenzmatrix durch $\overline{x}$ gegeben ist, hat eine k-Clique

Es folgt sofort: $\text{CLIQUE}_{k,n}$ ist monoton **T6.17**

Die zugehörige Funktionenfamilie ist schwer für monotone Schaltkreise (ohne Beweis):

Theorem 6.29 (Razborov 1985; Andreev 1985; Alon und Boppana 1987)

$\exists \varepsilon > 0 \ \forall k \leq n^{\frac{1}{4}}$: es gibt keinen monotonen Schaltkreis der Größe $< 2^{\varepsilon\sqrt{k}}$, der $\text{CLIQUE}_{k,n}$ berechnet

T6.18

Verallgemeinerung auf beliebige (nicht-monotone) Schaltkreisklassen offen

Zurück zu *uneingeschränkt* schweren Funktionen

Versuch, NP $\not\subseteq$ $P_{/poly}$ zu zeigen:
Finde schwere Funktionenfamilie, die in NP berechenbar ist.

Versuch blieb erfolglos –
beste bisher bekannte untere Schranke ist linear (ohne Beweis):

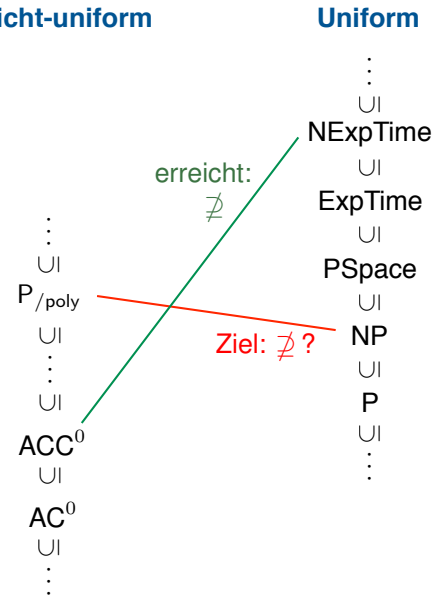
Theorem 6.30 (Iwama, Morizumi 2002)

Es gibt eine Familie $(f_n)_{n\in\mathbb{N}}$ von polyzeit-konstruierbaren Funktionen $f_n : \{0,1\}^n \to \{0,1\}$,
deren Schaltkreiskomplexität mindestens $5n - o(n)$ beträgt.

Schaltkreiskomplexität: bekannte untere Schranken

Ziel: Gilt NP $\not\subseteq$ P_{poly} ?

Bisher erreicht:
NEXPTIME $\not\subseteq$ ACC^0
(Williams 2011)



You can't always get what you want ...

Siehe auch:
http://blog.computationalcomplexity.org/2014/11/favorite-theorems-circuit-lower-bounds.html

Offene Fragen für ACC^0

Bekannt: PARITY $\notin AC^0$ $\Rightarrow$ $AC^0 \subsetneq NC^1$ und $AC^0 \subsetneq P$

Offene Fragen

- $ACC^0 \subsetneq NC^1$?
- $ACC^0 \subsetneq P$?
- CLIQUE $\notin ACC^0(6)$?

Schaltkreise nicht konstanter Tiefe

Beschränkung der Tiefe wird gelockert.

Einfachste SK-Klasse mit nicht konstanter Tiefe: $O(\log n)$ Tiefe und $O(n)$ Größe

Offene Fragen

- Finde eine Boolesche Funktionenfamilie $f(n)_{n\in\mathbb{N}}$, so dass f_n nicht durch SKe der Tiefe $O(\log n)$ und Größe $O(n)$ berechnet werden kann.
- Finde eine nicht-Boolesche Funktionenfamilie mit dieser Eigenschaft (also $f_n : \{0,1\}^n \to \{0,1\}^n$).

(Mittels eines Zählarguments kann man leicht beweisen, dass es eine solche Funktionenfamilie geben muss.)

Zusammenfassung untere Schranken

Forschungsprogramm war erfolgreich für einige eingeschränkte Schaltkreisklassen:

- konstante Tiefe
- monotone Schaltkreise

Für diese sind nichttriviale untere Schranken bekannt.

Die ursprüngliche Frage „NP $\not\subseteq$ $P_{/poly}$?“ bleibt offen:

Bisher kennt man für keine explizite Funktionenfamilie eine super-lineare untere Schranke für die Schaltkreiskomplexität.

Übersicht Vorlesung