

Komplexitätstheorie 4

Montag und Donnerstag
14:15 – 15:45 Uhr
in C-221

noch mehr Hierarchien

In der Komplexitätstheorie wie auch der Berechenbarkeitstheorie kennt und unterscheidet man vielfältige Klassifizierungen durch Hierarchien von Sprachklassen oder Klassen von Funktionen:

- a. Polynomielle Hierarchie (Meyer/Stockmeyer 1973)
(*NP-hart aber nicht mehr in NP*)
- b. Arithmetische Hierarchie (auch: Kleene/Mostowski Hierarchie)
(*Rekursionstheorie: Grade der Unentscheidbarkeit*)
- c. Analytische Hierarchie (Kleene)
(*Rekursionstheorie: Quantifizierung zweiter Ordnung*)

Nötig dazu das Konzept der Orakel Turingmaschine [A.M. Turing: "Systems of logic based on ordinals", Proc. London Math. Soc., 42 (1939) pp 230-265]

Die Orakel Turing-Maschine

Definition:

Eine Orakel-Turingmaschine M^A ist eine Turingmaschine (NTM oder DTM) mit drei zusätzlichen Zuständen „JA“, „NEIN“ und „?“, sowie dem sog. „Orakelband“. Eine Orakel-TM M^A akzeptiert eine Sprache relativ zu dem Orakel $A \subseteq \Sigma^*$, wobei A eine beliebige, auch unentscheidbare Menge sein kann!

Bei Eingabe von $w \in \Sigma^*$ rechnet M^A zunächst wie gewöhnlich. Gerät M^A in den Zustand „?“, so ist der Folgezustand „JA“, falls die augenblickliche Inschrift rechts des LSK in der Menge A ist, andernfalls „NEIN“.

Akzeptiert wird w , falls M^A in einen Endzustand gerät.

Wenn statt der Orakelmenge A eine unendliche Zeichenkette $x \in \{0, 1\}^\omega$ benutzt wird, so ist diese als charakteristische Funktion einer Menge $B \in \mathcal{P}(\mathbb{N})$ anzusehen:

Das n -te bit von x ist 1, genau dann, wenn $n \in B$.

Turing-Reduktion und many-one Reduktion

Definition:

Für $A, B \subseteq \Sigma^*$ sagen wir, dass A rekursiv aufzählbar in B ist, wenn es eine Orakel-Turingmaschine M^B gibt mit $A = L(M^B)$.

Wenn die Orakel-TM M^B stets hält, so nennen wir A Turing reduzierbar auf B , notiert als $A <_T B$. (Man sagt auch: A ist rekursiv in B .)

$A <_m B$ ist sog. *many-one* Reduktion von $A \subseteq \Sigma^*$ auf $B \subseteq \Gamma^*$, wenn es berechenbare Funktion $g : \Sigma^* \rightarrow \Gamma^*$ gibt, mit $x \in A \leftrightarrow g(x) \in B$.

Zur Durchführung einer Turing-Reduktion wird sozusagen eine externe, bei verschiedenen Elementen der Sprache A möglicherweise unterschiedliche, Prozedur verwendet (B ist i.A. unendliche Menge). Bei der *many-one* Reduktion, wird immer der gleiche Algorithmus „berechne g “ benutzt.

Turing-Reduktion

Satz:

1. Turing-Reduktionen sind transitiv.
2. $<_m$ verfeinert $<_T$, d.h.: $A <_m B$ impliziert $A <_T B$.
3. $<_T$ ist eine echte Vergrößerung von $<_m$.

Beweis:

1. folgt leicht aus der Definition.
2. einfache Übung.
3. Sei H die Menge, die das Halteproblem codiert.
Dann gilt für $\bar{H} := \{0,1\}^* \setminus H$ eben $\bar{H} \not<_m H$ aber $\bar{H} <_T H$!
Jede Menge A ist Turing-reduzierbar auf ihr Komplement: Mit dem Orakel für A fragt man dieses, ob $x \in A$ ist, und akzeptiert bei Antwort **nein** bzw. verwirft bei Antwort **ja**.

Die Polynomielle Hierarchie (0)

Definition:

$$P^A := \{L \subseteq \Sigma^* \mid L = L(M^A) \text{ für eine polynomzeit-beschränkte Orakel-DTM } M^A\}$$

$$= \{L \subseteq \Sigma^* \mid L \text{ ist } \mathbf{deterministisch} \text{ in Polynomzeit Turing-reduzierbar auf } A\}$$

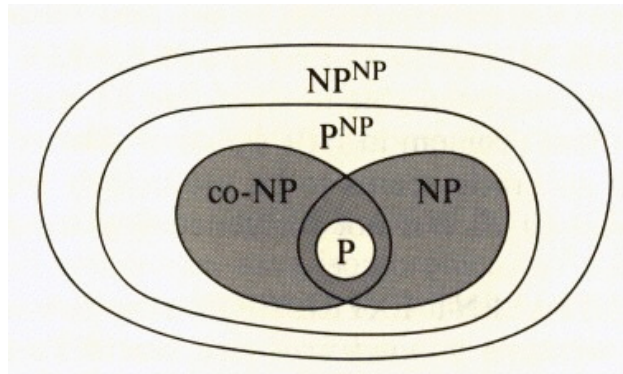
$$NP^A := \{L \subseteq \Sigma^* \mid L = L(M^A) \text{ für eine polynomzeit-beschränkte Orakel-NTM } M^A\}$$

$$= \{L \subseteq \Sigma^* \mid L \text{ ist } \mathbf{nichtdeterministisch} \text{ in Polynomzeit Turing-reduzierbar auf } A\}$$

Die Polynomielle Hierarchie (1)

Fakten:

1. $P^P = P$ und $NP^P = NP$.
2. $NP^A = NP^{\bar{A}}$.
3. $NP \cup \text{co-NP} \subseteq P^{SAT} \subseteq PSPACE$.
4. $A \in \mathcal{R}ec \rightarrow NP^A \in \mathcal{R}ec$.
5. $P^A \subseteq NP^A$



PSPACE einmal anders

Satz:

$$PSPACE \subseteq P^{QBF} \subseteq NP^{QBF} \subseteq NPSPACE = PSPACE$$

also:

$$PSPACE = P^{QBF} = NP^{QBF}$$

Beweis:

1. \subseteq Weil QBF vollständig ist für $PSPACE$, kann jedes $L \in PSPACE$ deterministisch in polynomieller Zeit dadurch entschieden werden, dass L auf QBF reduziert wird, und dann das Orakel einmal für " $w \in L?$ " befragt wird.
2. \subseteq Das ist trivial, denn $P \subseteq NP$.
3. \subseteq Jede Orakel-NTM M^{QBF} kann durch $PSPACE$ -beschränkte NTM simuliert werden, die die Orakelfragen an QBF selbst beantwortet.

Relativierung

Aus dem letzten Ergebnis folgt, dass sich P und NP relativ zu QBF nicht unterscheiden!

Ist damit auch " $P=NP$?" entschieden?

Nein:

Satz:

$$\exists B \in \text{Rec} : P^B \neq NP^B$$

Idee:

Die entscheidbare Sprache B 'diagonalisiert' über alle DTM -n und ihre eigenen Anfänge. B wird für jedes n höchstens ein Wort der Länge n enthalten. $L_B := \{0^n \mid \exists w \in B : |w| = n\}$ ist offensichtlich ein Element von NP^B .

Zu zeigen ist also nur noch, dass es entscheidbare Sprache B gibt, mit $L_B \notin P^B$.

(Deren Definition ist nicht ganz einfach und in
Papadimitriou: *Computational Complexity* nachzulesen!)

Die Polynomielle Hierarchie (2)

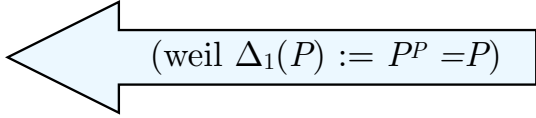
Definition: [Meyer & Stockmeyer, (1973)]

Die *Polynomielle Hierarchie* ist folgende Hierarchie von Sprachfamilien:

1. $\Delta_0(P) := \Sigma_0(P) := \Pi_0(P) = P$.
2. $\Delta_{i+1}(P) := P^{\Delta_i(P)}$ für $i \geq 0$,
3. $\Sigma_{i+1}(P) := NP^{\Sigma_i(P)}$ für $i \geq 0$, und
4. $\Pi_{i+1}(P) := \text{co-}NP^{\Sigma_i(P)}$ für $i \geq 0$, ergibt zusammen:

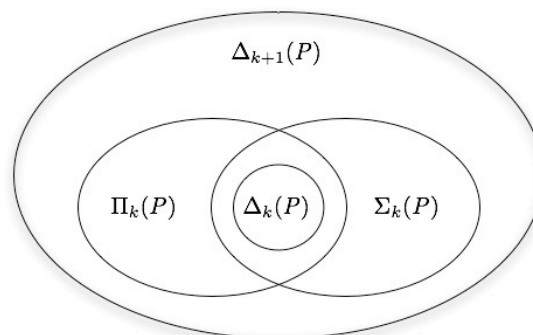
$$PH := \bigcup_{i \geq 0} \Sigma_i(P)$$

Eigenschaften

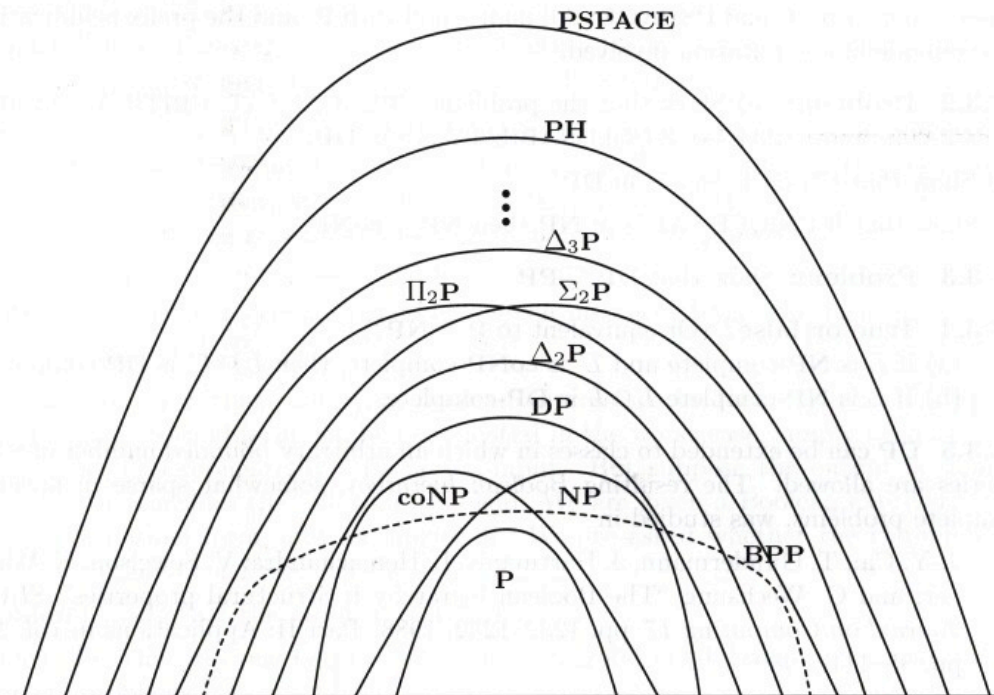
- a) $\Delta_1(P) = P$,  (weil $\Delta_1(P) := P^P = P$)
- b) $\Sigma_{k+1}(P) = NP^{\Pi_k(P)} = NP^{\Delta_k(P)}$,
- c) $\Delta_{k+1}(P) = P^{\Pi_k(P)}$,
- d) $\Delta_k(P) = \text{co-}\Delta_k(P)$,
- e) $P^{\Delta_k(P)} = \Delta_k(P)$,
- f) $\Sigma_k(P) \cup \Pi_k(P) \subseteq \Delta_{k+1}(P)$,
- g) $\Delta_k(P) \subseteq \Sigma_{k+1}(P) \cap \Pi_{k+1}(P)$,
- h) alle Stufen sind unter \cap , \cup und $<_{\text{pol}}$ abgeschlossen!
- i) $\Sigma_k(P) \subseteq \Pi_k(P) \rightarrow \Sigma_k(P) = \Pi_k(P)$

Die k-te Stufe

k-te Stufe von polynomieller Hierarchie PH



die polynomielle Hierarchie (anschaulich)



(Wir schrieben statt \mathbf{P} stets (P) und statt \mathbf{coNP} auch $co-NP$)

Zielrichtung:

Die Idee dabei: Wenn *downward separation* gilt, dann muss nur die Echtheit der Polynomiellen Hierarchie gezeigt werden, um $P \neq NP$ zu beweisen. Dass das schwierig wird, zeigten die letzten Orakel-Resultate.

downward separation \cong *upward equality*...

Beispiel (mit *padding*-Technik):

$$\mathcal{L} = \mathcal{NL} \rightarrow DSPACE(n) = \mathcal{Cs}.$$

upward equality und downward separation

Lemma: (*upward equality*)

$$\exists k \in \mathbb{N} : \Sigma_k(P) = \Sigma_{k+1}(P) \rightarrow \forall j \geq 0 : \Sigma_k(P) = \Sigma_{k+j}(P)$$

Beweis: (mit Induktion):

$$\Sigma_{k+2}(P) = NP^{\Sigma_{k+1}(P)} = NP^{\Sigma_k(P)} = \Sigma_{k+1}(P) = \Sigma_k(P)$$

Lemma: (*downward separation*)

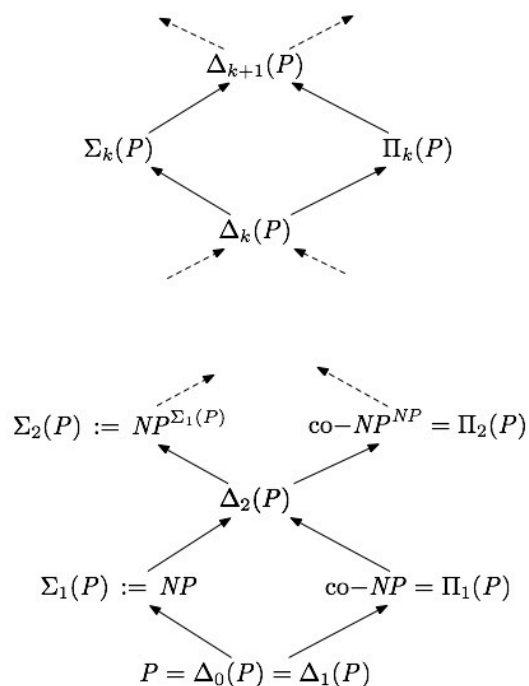
$$\exists k \in \mathbb{N} : \Sigma_k(P) \subsetneq \Sigma_{k+1}(P) \rightarrow P \neq NP$$

oder allgemeiner:

$$\exists k, j \in \mathbb{N} : \Sigma_k(P) \subsetneq \Sigma_j(P) \rightarrow P \neq NP$$

andere Darstellung der Polynomiellen Hierarchie

polynomielle Hierarchie



collapsing the PH-hierarchy

Folgerung:

$$P = NP \rightarrow \forall j \geq 0 : \Sigma_1(P) = \Sigma_j(P) = PH$$

Definition:

EQUIV

Eingabe:

Zwei boolesche Formeln F und G .

Frage:

Sind F und G equivalent?

Definition:

$MINIMAL := \left\{ F \text{ ist boolesche Formel} \mid \begin{array}{l} \text{keine boolesche Formel } G, \\ \text{die zu } F \text{ equivalent ist,} \\ \text{hat weniger Symbole als } F \end{array} \right\}$

einige Mengen aus PH

Lemma:

$$EQUIV \in \Pi_1(P) \text{ und } MINIMAL \in \Pi_2(P).$$

Für die Menge *MINIMAL* ist nicht bekannt, ob sie in einer tieferen Stufe der Polynomiellen Hierarchie liegt,

und auch nicht,

ob sie vielleicht sogar vollständig für $\Pi_2(P)$ ist!

Ein Problem in $\Sigma_2(P)$ ist dieses:

Gegeben ein boolescher Ausdruck A in *DNF* und eine Zahl b . Besitzt die *KNF* für A höchstens b Klauseln?

Es gibt tatsächlich auch vollständige Sprachen für die Σ -Stufen der Polynomiellen Hierarchie!

feste Alternierung

Definition:

- a) F_{QSAT_i} sei die Menge aller quantifizierten booleschen Formeln in Pränex-Normalform der Art:

$$\exists x_1 \in V_1 \forall x_2 \in V_2 \exists x_3 \in V_3 \cdots Q_i x_i \in V_i \Phi,$$

wobei $Q_n := \begin{cases} \forall, & \text{falls } i \text{ gerade,} \\ \exists, & \text{falls } i \text{ ungerade} \end{cases}$ und die sogenannte Matrix

Φ eine boolesche Formel über $\{0, 1, x_1, x_2, \dots, x_n\}$ in 3-KNF ist.

Hierbei ist die Variablenmenge V disjunkt zerlegt in

$$V = V_1 \uplus V_2 \uplus \cdots \uplus V_i.$$

- b) Die Menge $QSAT_i$ ist definiert als

$$QSAT_i := \{w \in F_{QSAT_i} \mid w \text{ besitzt erfüllende Belegung}\}.$$

Satz:

$QSAT_i$ ist vollständig für $\Sigma_i(P)$ für jedes $i \geq 1$.

vollständige Sprachen für PH ?

Satz:

Falls eine Sprache L vollständig für PH ist, dann fällt die Polynomielle Hierarchie bis zu einer festen Höhe zusammen.

Beweis: (mit Induktion):

Da $L \in PH$ gilt, gibt es $i \in \mathbb{N}$ mit $L \in \Sigma_i(P)$. Damit gilt für jede Sprache $L' \in \Sigma_{i+1}(P)$ auch $L' <_{\log} L$. Da jedes Level gegenüber Reduktionen abgeschlossen ist, folgt $\Sigma_i(P) = \Sigma_{i+1}(P)$. und damit also $\Sigma_i(P) = \Sigma_j(P)$ für jedes $j \geq i$!

PH und *PSPACE*

Lemma:

$$PH \subseteq PSPACE.$$

Aus $PH = PSPACE$ folgt, dass es ein vollständiges Problem für PH gibt ($PSPACE$ hat solche!) und die Hierarchie PH wird nicht “länger”, sondern “kürzer”!

Es gibt Orakel,
für die $PH \neq PSPACE$
und solche,
für die $PH = PSPACE$ gilt!

Zusammenhang mit *BPP*

Ein weiteres interessantes Resultat ist folgendes Ergebnis:

Satz:

Wenn das Erfüllbarkeitsproblem SAT mit booleschem Schaltkreis polynomieller Größe entschieden werden kann, so gilt $PH = \Sigma_2(P)$.

Exakte Problemvariante zu NP

Im *Cliquen-Problem* wir danach gefragt, ob in einem Graph $G=(V,E)$ zu einer Zahl k eine Clique mit mindestens k Knoten existiert.

Dieses Problem ist in NP

Im Komplement des *Cliquen-Problems* wir danach gefragt, ob in einem Graph $G=(V,E)$ zu einer Zahl k eine Clique mit höchstens k Knoten existiert.

Dieses Problem ist in co-NP

Das *exakte Cliquen-Problem* ist die Frage, ob in einem Graph $G=(V,E)$ zu einer Zahl k eine Clique mit genau k Knoten existiert.

Dieses Problem ist in $DP := NP \wedge co-NP$

mit

$NP \wedge co-NP := \{A \cap B \mid A \in NP, B \in co-NP\}$.

die Differenz zu NP

$DP := \{A \cap B \mid A \in NP, B \in co-NP\}$ bedeutet, dass für $\bar{B} := \Sigma^* \setminus B$ eben $A \cap B = A \setminus \bar{B}$ mit $\bar{B} \in NP$ gilt.

Somit ist

$$DP = \{L_1 \setminus L_2 \mid L_1, L_2 \in NP\}.$$

Die Klasse DP ist gegenüber Durchschnittsbildung abgeschlossen, genauso wie NP und P .

Gegenüber Vereinigung (und damit Komplement) ist die Familie DP wahrscheinlich nicht abgeschlossen, aber man kann zunächst die sog.

Boolesche Hierarchie, BH,
definieren.

die Boolesche Hierarchie

$$BH_0 := P$$

$$BH_1 := NP$$

⋮

$$BH_k := \begin{cases} BH_{k-1} \wedge co-NP, & \text{falls } k \geq 2 \text{ gerade.} \\ BH_{k-1} \vee NP, & \text{falls } k \geq 2 \text{ ungerade.} \end{cases}$$

⋮

$$BH := \bigcup_{k \geq 0} BH_k$$

Wegen $\emptyset, \Sigma^* \in NP$ und $\emptyset, \Sigma^* \in co-NP$, ist

$$BH_{k-1} \subseteq BH_k \quad \text{und} \quad co-BH_{k-1} \subseteq co-BH_k$$

Boolescher Abschluss

Definition:

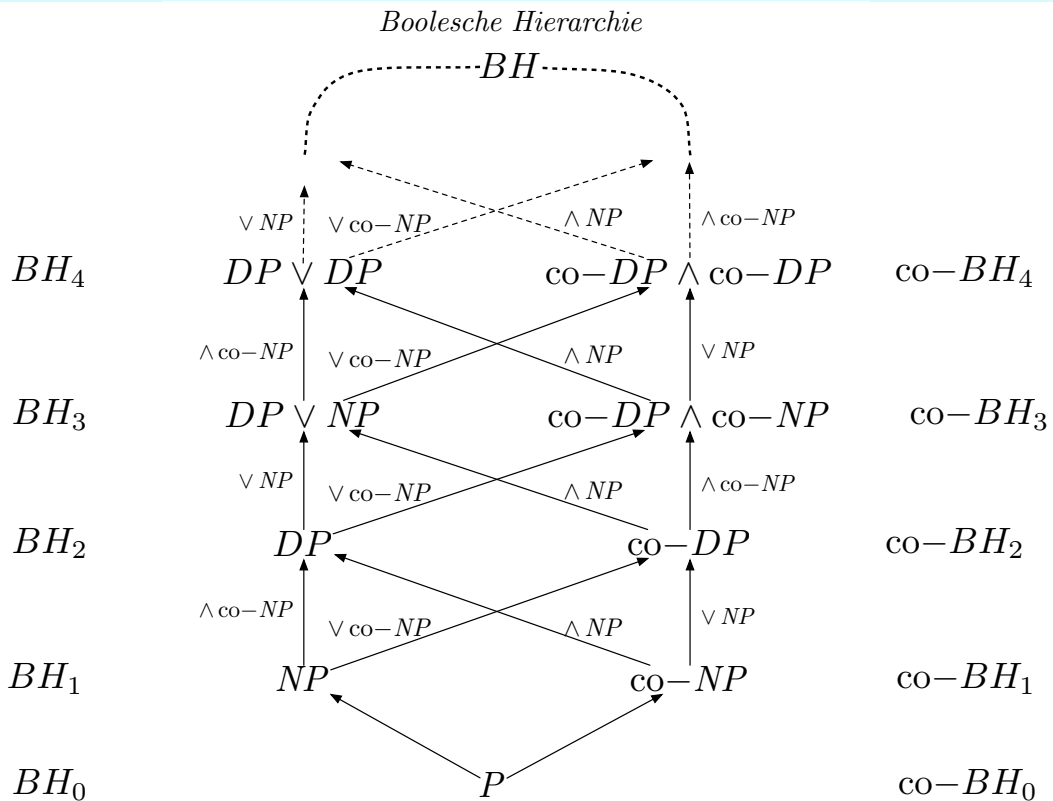
Zu einer beliebigen Familie von Mengen C sei deren Boolescher Abschluss, $BC(C)$, definiert als kleinste Klasse von Mengen, die C enthält und abgeschlossen ist gegenüber:

1. Vereinigung
2. Durchschnitt
3. Komplementbildung

Satz:

$$BH = BC(NP)$$

Das Gesamtbild als Hasse Diagramm



Vollständige Probleme für BH_k

$L_{BH_2} := \{(F_1, F_2) \mid F_1 \text{ ist erfüllbar und } F_2 \text{ ist unerfüllbar}\}$

$L_{BH_3} := \{(F_1, F_2, F_3) \mid (F_1 \text{ ist erfüllbar und } F_2 \text{ ist unerfüllbar})$
 oder $F_3 \text{ ist erfüllbar}\}$

...

Satz:

L_{BH_2} ist DP -vollständig (= BH_2 -vollständig)

Satz:

Für $k \geq 1$ gilt:

L_{BH_k} ist BH_k -vollständig

Beweis für L_{BH_2}

- $L_{BH_2} \in DP$ ist klar.
- Sei $L \in DP$, d.h. $L = L_1 \cap L_2$ für $L_1 \in NP$, $L_2 \in co-NP$.

Also gibt es Reduktionen f_1, f_2 mit
 $x \in L_1 \Leftrightarrow f_1(x) \in SAT$ und $x \in L_2 \Leftrightarrow f_2(x) \in co-SAT$.

mit Abbildung f , die x auf $(f_1(x), f_2(x))$ abbildet ist dann

$$\begin{aligned}x \in L &\Leftrightarrow x \in L_1 \cap L_2 \Leftrightarrow x \in L_1 \text{ und } x \in L_2 \\ &\Leftrightarrow f_1(x) \in SAT \text{ und } f_2(x) \in co-SAT \\ &\Leftrightarrow (f_1(x), f_2(x)) \in L_{BH_2}\end{aligned}$$

die gesuchte Polynomialzeit-Reduktion.

- Für die Sprachen L_{BH_k} geht das dann analog.

Orakelmaschinen mit verschiedenen Abfragemodi

Definition:

Sei FP die Klasse der polynomzeit berechenbaren, totalen Funktionen von Σ^* nach Σ^*

Wir schreiben $A \leq_{tt}^p B$ genau dann, wenn

$\exists f \in FP : f(x) = \langle \tau, q_1, q_2, \dots, q_k \rangle$, so dass

$$x \in A \iff \tau(\chi_B(q_1), \chi_B(q_2), \dots, \chi_B(q_k)) = 1.$$

Hierbei ist τ eine boolesche Funktion, die über einen Schaltkreis beschrieben und berechnet wird.

Die Zahl k der Orakelfragen ist polynomiell in $|x|$, da f in FP ist!

Definition:

Die \leq_{tt}^p Hülle einer Klasse \mathcal{C} ist:

$$P_{tt}^{\mathcal{C}} := \{A \mid \exists B \in \mathcal{C} : A \leq_{tt}^p B\}$$

Query Hierarchie

Definition:

Die Query-Hierarchie über NP ist definiert durch

$$P^{NP[k]} := \left\{ L \mid \begin{array}{l} L = L(M^{SAT}) \text{ für DPOTM } M, \text{ die maximal} \\ k \text{ Turing-Anfragen an } SAT \text{ stellt.} \end{array} \right\}$$

$$P^{NP[O(1)]} := \bigcup_{k \in \mathbb{N}} P^{NP[k]} \quad \text{und} \quad P^{NP[O(\log)]} := \bigcup_{k(\log)} P^{NP[k]}$$

$$P_{k\text{-tt}}^{NP} := \left\{ L \mid \begin{array}{l} L = L(M^{SAT}) \text{ für DPOTM } M, \text{ die maximal} \\ k \text{ truth-table-Anfragen an } SAT \text{ stellt.} \end{array} \right\}$$

Boolesche Hierarchie ist in Polynomieller Hierarchie enthalten!

Recht kompliziert zu beweisen ist das folgende Theorem,
aus dem obige Aussage folgt:

Theorem:

1. $P_{k\text{-tt}}^{NP} = P^{A[1]}$ für eine \leq_m^p -vollständige Menge $A \in BH_k$.
2. $P^{NP[k]} = P_{(2^k-1)\text{-tt}}^{NP}$
3. $BH_k \cup \text{co-}BH_k \subseteq P_{k\text{-tt}}^{NP} \subseteq BH_{k+1} \cup \text{co-}BH_{k+1}$
4. $BH_{2^k-1} \cup \text{co-}BH_{2^k-1} \subseteq P^{NP} \subseteq BH_{2^k} \cup \text{co-}BH_{2^k}$

Korollar:

$$BH = P^{NP[O(1)]} = \bigcup_{k \in \mathbb{N}} P_{k\text{-tt}}^{NP} = BC(NP)$$

Der Satz von Kadin

Satz:

Falls es $k \in \mathbb{N}$ gibt, mit $BH_k = \text{co-}BH_k$, dann kollabiert die Polynomielle Hierarchie auf der dritten Stufe zusammen, d.h., dann gilt $PH = \Sigma_3 \cap \Pi_3$.

Dies Originalergebnis von Kadin benutzt sog. „Dünne Mengen“ (*sparse sets*) mit Ergebnissen von Yap (1983) unter Verwendung einer Technik, die als „*easy-hard-technique*“ bezeichnet wird. Yap zeigte, diesen Zusammenbruch der Polynomiellen Hierarchie, sobald es eine dünne Menge S gibt, mit $\text{co-NP} \subseteq NP^S$.

Spätere Arbeiten von Wagner (1987,1989) und Chang und Kadin (1991) verbesserten das Ergebnis letztlich zum Zusammenbruch bei $PH = BC(\Sigma_2)$.

Dünne Mengen

Definition:

Eine Menge $L \subseteq \Sigma^*$ heißt *dünn*, wenn

$$\left| \{w \in L \mid n = |w|\} \right| \leq p(n)$$

für ein Polynom $p(n)$.

Die Menge B , mit der $P^B \neq NP^B$ gezeigt wurde, ist eine dünne Menge. Mahaney zeigte (1982), dass unter der Voraussetzung $P \neq NP$ keine dünne Menge NP -schwer sein kann.

Satz (Mahaney):

Wenn $A \in NP^S$ für eine dünne Menge S gilt, dann folgt $A \in P^{NP[\log]}$.

Der wichtige Satz für den Beweis von Kadin stammte von Yap (1983):

Das Lemma von Yap

Lemma:

Falls es eine dünne Menge S gibt, mit $co-NP \subseteq NP^S$, dann kollabiert die Polynomielle Hierarchie auf der dritten Stufe zusammen, d.h., dann gilt $PH = \Sigma_3 \cap \Pi_3$.

Der Beweis über die sog. *easy-hard technique* verlangt weitere Definitionen und ist hier weggelassen. Die bislang stärkste Kopplung zwischen Boolescher Hierarchie und dem Kollaps der Polynomiellen Hierarchie ist wohl von E. und L. Hemaspaandra & Hempel (1989) gleichfalls Reith & Wagner (2001).

Das bisher stärkste und allgemeinste Resultat:

E. und L. Hemaspaandra und H. Hempel zeigten 2001:

Satz:

$$\forall k > 0 : \forall i > 1 :$$

$$P_{k-tt}^{\Sigma_i} = P_{(k+1)-tt}^{\Sigma_i} \implies BH_k(\Sigma_i) = co-BH_k(\Sigma_i)$$

Dazu sei die benutzte Notation erklärt:

Definition:

Mit $A_1 - A_2 - \dots - A_{k-1} - A_k := A_1 - (A_2 - (\dots - (A_{k-1} - A_k \dots)))$
sei für eine Menge \mathcal{C}

$$BH_k(\mathcal{C}) := \left\{ L \mid \begin{array}{l} L = A_1 - A_2 - \dots - A_k \text{ für Mengen } A_i \in \mathcal{C} \\ 1 \leq i \leq k \text{ mit } A_k \subseteq A_{k-1} \subseteq \dots \subseteq A_1 \end{array} \right\}$$

und

$$BH(\mathcal{C}) := BC(\mathcal{C})$$

die Arithmetische Hierarchie

Definition:

Die *Arithmetische Hierarchie* ist folgende Hierarchie von Sprachfamilien:

1. $\Sigma_0 := \Delta_0 := \Pi_0 = \mathcal{R}ec$.
2. $\Sigma_{i+1} := \mathcal{R}e^{\Sigma_i}$ für $i \geq 0$,
3. $\Pi_i := \text{co-}\Sigma_i$ für $i \geq 0$,
4. $\Delta_i := \Sigma_i \cap \Pi_i$ für $i \geq 0$.

Fakten:

1. $\Sigma_1 = \mathcal{R}e$
2. $\forall i \geq 0 : \Sigma_i \subseteq \Sigma_{i+1}$ und $\Pi_i \subseteq \Pi_{i+1}$.
3. $\forall i \geq 0 : \Sigma_i \cup \Pi_i \subseteq \Delta_{i+1}$.
4. $A \in \Sigma_{i+1} \leftrightarrow A \in \mathcal{R}e^{\Sigma_i}$.
5. $\Delta_{i+1} = \mathcal{R}ec^{\Sigma_i}$

alternative Charakterisierung der Arithmetischen Hierarchie

Definition:

- a) Es gilt $L \in \Sigma_n$ genau dann, wenn es entscheidbares, $n + 1$ -stelliges Prädikat R gibt, mit

$$L = \{y \mid \exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n : R(y, x_1, \dots, x_n)\},$$

$$\text{wobei } Q_n := \begin{cases} \forall, & \text{falls } n \text{ gerade,} \\ \exists, & \text{falls } n \text{ ungerade.} \end{cases}$$

- b) Es gilt $L \in \Pi_n$ genau dann, wenn es entscheidbares, $n + 1$ -stelliges Prädikat R gibt, mit

$$L = \{y \mid \forall x_1 \exists x_2 \forall x_3 \cdots Q_n x_n : R(y, x_1, \dots, x_n)\}$$

$$\text{wobei } Q_n := \begin{cases} \exists, & \text{falls } n \text{ gerade,} \\ \forall, & \text{falls } n \text{ ungerade.} \end{cases}$$

einige Sprachen in der Arithmetischen Hierarchie

Definition und Einstufung:

1. $EMPTY := \{M \text{ ist DTM} \mid L(M) = \emptyset\} \in \Pi_1,$
2. $TOTAL := \{M \text{ ist DTM} \mid L(M) \in Rec\} \in \Pi_2,$
3. $FINITE := \{M \text{ ist DTM} \mid L(M) \text{ ist endlich}\} \in \Sigma_2,$
4. $COFIN := \{M \text{ ist DTM} \mid \Sigma^* \setminus L(M) \text{ ist endlich}\} \in \Sigma_3.$

Begründung:

1. $EMPTY = \{M \mid \forall w \forall t : M \text{ akzeptiert } w \text{ nicht in } t \text{ Schritten}\},$
2. $TOTAL = \{M \mid \forall w \exists t : M \text{ akzeptiert } w \text{ in } t \text{ Schritten}\},$
3. $FINITE = \{M \mid \exists n \forall w \forall t : |w| \leq n \text{ oder } M \text{ akzeptiert } w \text{ nicht in } t \text{ Schritten}\},$
4. $COFIN = \{M \mid \exists n \forall w \exists t : |w| \leq n \text{ oder } M \text{ akzeptiert } w \text{ in } t \text{ Schritten}\}.$

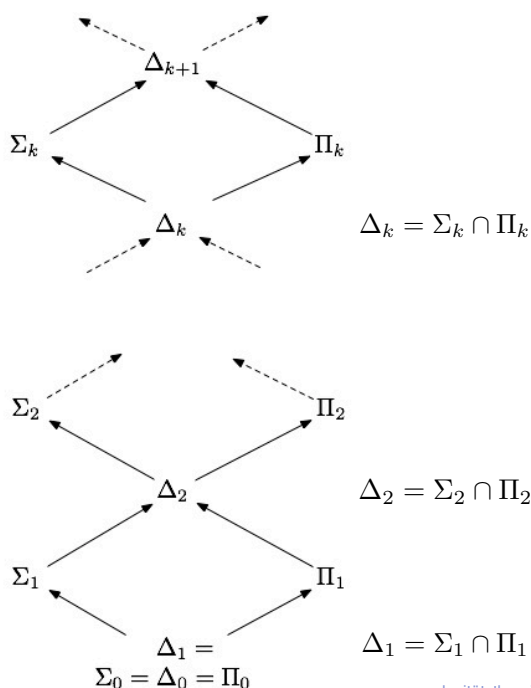
Arithmetische Hierarchie, anschaulich

arithmetische Hierarchie

Alle Inklusionen sind echt!

Alle Klassen sind abgeschlossen gegenüber \cap , \cup und $<_{\text{pol}}$.

Die Klassen Δ_k sind dazu gegenüber Komplement abgeschlossen!



Alternierendes Akzeptieren

Alternierung ist Verallgemeinerung des Konzepts des Nichtdeterminismus bei Turingmaschinen oder endlichen Automaten!

Bisher ist nichtdeterministisches Akzeptieren existentiell definiert:

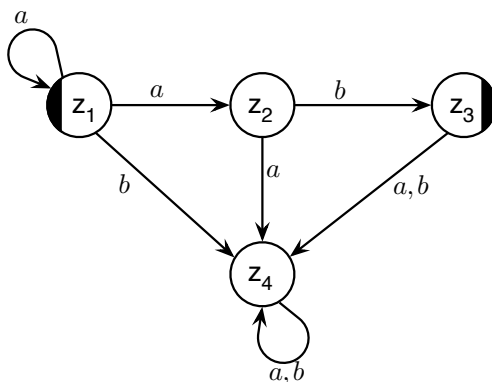
$w \in L$ wird akzeptiert,
wenn mindestens eine Erfolgsrechnung existiert!

existentiell

$w \in L$ wird akzeptiert,
wenn alle Rechnungen Erfolgsrechnungen sind!

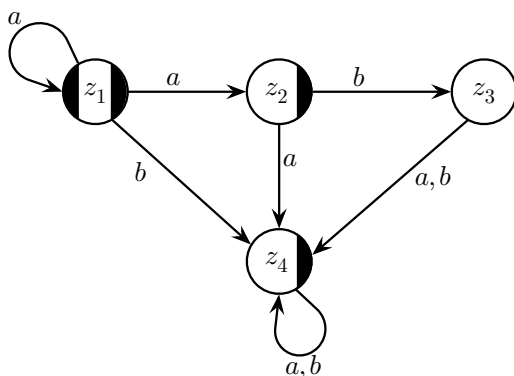
universell

universelles und existentielles Akzeptieren bei *NFA*



existentiell: $\{a\}^+\{b\}$

universell: \emptyset



existentiell: $(a+b)^*$

universell: $\{a, b\}^* \setminus \{a\}^+\{b\}$

AFA = alternierende NFA

Alternierende NFA akzeptieren nicht mehr als alle NFA:

die Potenzautomatenkonstruktion wird bei den universellen Zuständen variiert angewendet, so dass ein äquivalenter DFA erzeugt wird.

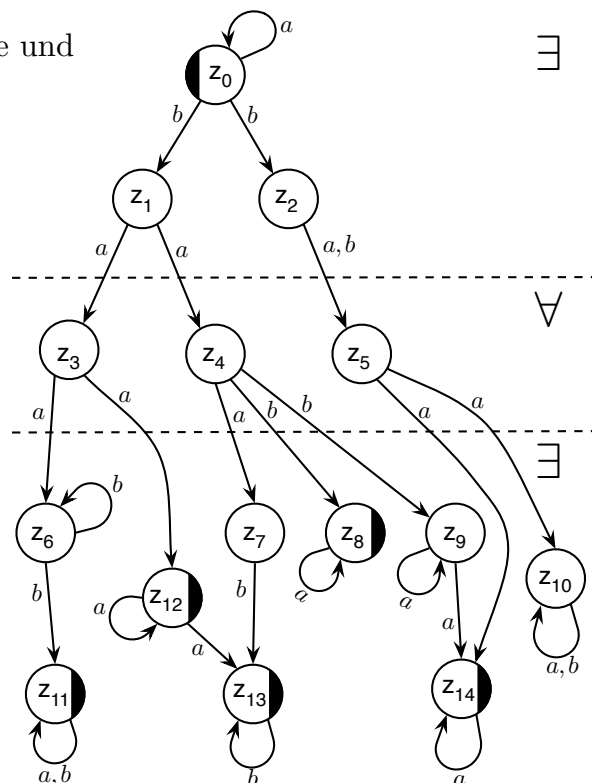
SFI Working Paper Abstract
2000
Title: New Results on Alternating and Non-Deterministic Two-Dimensional Finite-State Automata
Author(s): Jarkko Kari and Cristopher Moore
Files: [\[zipped postscript\]](#) [\[postscript\]](#)
Paper #: 00-09-051
Abstract: We resolve several long-standing open questions regarding the power of various types of finite-state automata to recognize "picture languages," i.e. sets of two-dimensional arrays of symbols. We show that the languages recognized by four-way alternating finite-state automata (AFAs) are incomparable to the so-called tiling-recognizable languages. Specifically, we show that the set of acyclic directed graphs is AFA-recognizable but not tiling-recognizable, while the set of non-acyclic directed graphs is tiling-recognizable but not AFA-recognizable. More generally, the complement of an AFA-recognizable language is tiling-recognizable, and therefore the AFA-recognizable languages are not closed under complement. We also show that the set of languages recognized by four-way NFAs is not closed under complement, and that NFAs are more powerful than DFAs, even for languages over one symbol.
Keywords: picture languages, finite-state automata, image recognition

Man kennt auch Arbeiten zu AFA, die mit Zeitverhalten gekoppelt sind.

alternierende Turingmaschinen: ATM

Eine ATM hat akzeptierende, nicht akzeptierende sowie sog. existentielle und universelle Zustände.

Ein Zustandswechsel von existentiell zu universell nennt man eine Alternierung.
 Die Zeit, die eine Berechnung zum Akzeptieren eines Wortes w benötigt wird gemessen in der Tiefe des alternierenden Akzeptierungsbaums:



$$L = a^*baba^+ \cup a^*baab^+$$

ein Beispiel zur alternierenden Akzeptanz

Definition:

$USAT := \{F \mid F \text{ ist boolesche Formel mit genau einer erfüllenden Belegung}\}$

$USAT$ ist von ATM $\exists\forall$ -akzeptierbar!

rate existentiell eine Variablenmenge X

rate universell eine Variablenmenge $Y \neq X$

prüfe ob $F(X) = true$ und $F(Y) = false$ gilt.
wenn ja, akzeptiere Belegung $X \cup Y$

zur Erinnerung: Charakterisierung der Arithmischen Hierarchie

Definition:

- a) Es gilt $L \in \Sigma_n$ genau dann, wenn es entscheidbares, $n + 1$ -stelliges Prädikat R gibt, mit

$$L = \{y \mid \exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n : R(y, x_1, \dots, x_n)\},$$

wobei $Q_n := \begin{cases} \forall, & \text{falls } n \text{ gerade,} \\ \exists, & \text{falls } n \text{ ungerade.} \end{cases}$

- b) Es gilt $L \in \Pi_n$ genau dann, wenn es entscheidbares, $n + 1$ -stelliges Prädikat R gibt, mit

$$L = \{y \mid \forall x_1 \exists x_2 \forall x_3 \cdots Q_n x_n : R(y, x_1, \dots, x_n)\}$$

wobei $Q_n := \begin{cases} \exists, & \text{falls } n \text{ gerade,} \\ \forall, & \text{falls } n \text{ ungerade.} \end{cases}$

Die Alternierungsklassen

Definition:

1. $ATIME(t) :=$ Familie von Sprachen, die von t -zeitbeschränkter ATM akzeptiert werden,
2. $ASPACE(s) :=$ Familie von Sprachen, die von s -platzbeschränkter ATM akzeptiert werden,
3. $ATIME - SPACE(t, s) :=$ Familie von Sprachen, die von simultan t -zeitbeschränkter und s -platzbeschränkter ATM akzeptiert werden,
4. $ATIME - ALT(t, a) :=$ Familie von Sprachen, die von simultan t -zeitbeschränkter und a -alternierungsbeschränkter ATM akzeptiert werden,
5. $AALT - SPACE(a, s) :=$ Familie von Sprachen, die von simultan a -alternierungsbeschränkter und s -platzbeschränkter ATM akzeptiert werden.

Die Alternierungs-Einstufung:

$$\begin{aligned} EXPTIME &= DTIME \left(2^{n^{O(1)}} \right) = ASPACE \left(n^{O(1)} \right) \\ &\cup \\ PSPACE &= DSPACE \left(n^{O(1)} \right) = ATIME \left(n^{O(1)} \right) \\ &\cup \\ P &= DTIME \left(n^{O(1)} \right) = ASPACE \left(\log n \right) \\ &\cup \\ L &= DSPACE \left(\log n \right) \supseteq ATIME \left(\log n \right) \end{aligned}$$

Das alles folgt aus folgenden Einzelergebnissen:

Satz:

$$DSPACE(f(n)) \subseteq ATIME \left((f(n))^2 \right).$$

Beweisidee

Wenn $L \in DSPACE(f(n))$ ist, dann hat jedes $w \in L$ eine akzeptierende Rechnung einer Länge $2^{O(f(n))}$. Eine rekursive Prozedur prüft nun die Erreichbarkeit einer Endkonfiguration von der Startkonfiguration:

β ist von α in höchstens 2^{t+1} Schritten erreichbar, wenn es eine Konfiguration γ gibt, für die gilt:

- a) γ ist in 2^t Schritten von α aus erreichbar
- und
- b) β ist in 2^t Schritten von γ aus erreichbar.

Für platzkonstruierbare Funktion f können $f(n)$ Speicherzellen markiert werden und in diesem Platz

existentiell die Konfiguration γ geraten werden.

Dann werden die Eigenschaften a) und b) *universell* verifiziert.

Die Rekursionstiefe ist $O(f(n))$ und in jedem Level ist die Zeit ebenfalls $O(f(n))$ und damit gilt $DSPACE(f(n)) \subseteq ATIME(f^2(n))$.

noch zum Beweis

Sollte f nicht platzkonstruierbar sein, so rät die *ATM* *existentiell* eine unäre Darstellung eines Wertes $f(n) = 1, 2, 3, \dots$ wenn die *ATM* dann w akzeptiert, so benötigt sie ebenfalls nur eine Tiefe des Akzeptierungsbaums von $O(f^2(n))$.

Folgerung:

Sei $f(n) \geq \log n$, dann gilt:

$$DSPACE(f(n)) \subseteq ATIME - SPACE \left((f(n))^2, f(n) \right)$$

weitere Ergebnisse (ohne Beweis)

Proposition:

Sei $t(n) \geq \log n$, dann gilt:

$$ATIME(t(n)) \subseteq DSPACE(t(n))$$

Proposition:

Sei $s(n) \geq \log n$, dann gilt:

$$ASPACE(s(n)) \subseteq DTIME(2^{O(s(n))})$$

Proposition:

Sei $t(n) \geq \log n$, dann gilt:

$$DTIME(t(n)) \subseteq ASPACE(\log(t(n)))$$

eine neue Hierarchie?

Das Hauptergebnis war:

Alternierende Polynomzeit = polynomieller Platz

Definition:

Eine $A\Sigma_k$ - (bzw. $A\Pi_k$ -) ATM ist eine ATM, die in existentiellen (bzw. universellen) Zuständen beginnt, und höchstens $(k - 1)$ -mal alterniert.

Die *Alternierungshierarchie* ist folgende Hierarchie von Sprachfamilien:

1. $A\Sigma_i(P) := \left\{ L \mid L \text{ wird von einer } A\Sigma_i\text{-ATM} \right.$
in Polynomzeit akzeptiert $\left. \right\}$,
2. $A\Pi_i(P) := \left\{ L \mid L \text{ wird von einer } A\Pi_i\text{-ATM} \right.$
in Polynomzeit akzeptiert $\left. \right\}$.

wirklich eine neue Hierarchie?

Offensichtlich
gilt:

Lemma:

$$A\Sigma_i(P) = \text{co-}A\Pi_i(P)$$

$$A\Sigma_i(P) \subseteq A\Sigma_{i+1}(P)$$

Was sind das für Klassen?

$$A\Sigma_0(P) = P$$

$$A\Sigma_1(P) = NP$$

$$A\Sigma_2(P) = ?$$

Nix Neues durch Alternierung!

Satz:

$$\forall k \geq 0 : A\Sigma_k(P) = \Sigma_k(P)$$

$$\forall k \geq 0 : A\Pi_k(P) = \Pi_k(P)$$

Alternierungs Hierarchie = Polynomielle Hierarchie

Wie geht's weiter

- logarithmische Alternierungsklassen
- mehr zu Schaltkreisen
- Kolmogorov Komplexität
- Mehr zu NP-Vollständigkeitsbeweisen durch Reduktionen
- NC (Nicks Class) und logarithmische parallele Berechnungen
- Randomisierte Berechnungen
- Approximationsverfahren