Kapitel 8: Der PCP-Satz

Der PCP-Satz ist eins der zentralen (und überraschenden) Resultate der Komplexitätstheorie der 1990er Jahre.
Es gibt 2 (äquivalente) Sichten auf den PCP-Satz:

(1) Er gibt an, dass viele interessante Optimierungsprobleme schwer zu approximieren sind.

(2) Er gibt eine Verbindung zwischen NP und probabilistisch überprüfbaren Beweisen an.

Die Abkürzung "PCP" steht für "Probabilistically Checkable Proofs". Der PCP-Satz wurde 1992 von Arora, Lund, Motwani, Sudan und Szegedy bewiesen.

In diesem Kapitel werden wir beide o.g. Sichten auf den PCP-Satz kennenlernen (jeweils ohne Beweis).

8.1 Approximierbarkeit

Definition 8.1 (MAX-3SAT)

MAX-3SAT ist das Problem, zu einer gegebenen aussagenlogischen Formel φ in 3CNF eine Belegung zu finden, von der so viele Klauseln von φ wie möglich wahr gemacht werden.

Definition 8.2 (Approximation von MAX-3SAT)

(a) Sei φ eine aussagenlogische Formel in 3CNF.

Der Wert von φ, kurz: $val(\varphi)$, ist

$$\max_{\beta: \beta \text{ ist eine Belegung für } \varphi} \left(\frac{\text{Anzahl der Klauseln von } \varphi, \text{ die von } \beta \text{ wahr gemacht werden}}{\text{Anzahl aller Klauseln von } \varphi} \right)$$

Beachte: φ erfüllbar $\Leftrightarrow$ $val(\varphi) = 1$

(b) Sei $g \leq 1$.

Ein Algorithmus A berechnet eine g-Approximation für MAX-3SAT, falls f.a. $m \geq 1$ und alle 3CNF-Formeln φ, die aus genau m Klauseln bestehen, gilt: A gibt bei Eingabe von φ eine Belegung $A(\varphi)$ aus, die mindestens

$$g \cdot val(\varphi) \cdot m$$ Klauseln von φ wahr macht

Beispiel 8.3 ($\frac{1}{2}$-Approximationsalgorithmus für MAX-3SAT)

Wir beschreiben im Folgenden einen Polynomialzeit-Algorithmus, der eine $\frac{1}{2}$-Approximation für MAX-3SAT berechnet:

Sei φ die eingegebene 3CNF-Formel.
Sei $x_1, \ldots, x_n$ eine Liste aller in φ vorkommenden Variablen
Für $i := 1, \ldots, n$ tue Folgendes:

↓

- Belege x_i mit demjenigen Wert $\beta(x_i) \in \{0,1\}$, der mehr Klauseln von φ wahrmacht.
- Entferne aus φ alle Klauseln, die die Variable x_i enthalten

Behauptung: Die so erhaltene Belegung $\beta: \{x_1, \dots, x_n\} \to \{0,1\}$ macht mindestens die Hälfte aller Klauseln von φ wahr.

Beweis:

Seien $C_1^{(1)}, \dots, C_{m_1}^{(1)}$ alle Klauseln von φ, in denen die Variable x_1 vorkommt. Für alle $i \geq 2$ seien $C_1^{(i)}, \dots, C_{m_i}^{(i)}$ alle Klauseln von φ, in denen die Variable x_i aber keine der Variablen $x_1, \dots, x_{i-1}$ vorkommt.

Insgesamt besteht φ also aus $m_1 + m_2 + \dots + m_n$ Klauseln.

Der obige Algorithmus macht mindestens $\frac{m_1}{2} + \frac{m_2}{2} + \dots + \frac{m_n}{2}$ Klauseln wahr. □

Bemerkung 8.4

Mit raffinierteren Methoden kann man sogar zeigen, dass es für jedes $\varepsilon > 0$ einen Polynomialzeit-Algorithmus gibt, der eine $\left(\frac{7}{8} - \varepsilon\right)$-Approximation für MAX-3SAT berechnet.

Aus dem PCP-Satz folgt, dass dies nicht beliebig verbessert werden kann (vgl Folgerung 8.7 weiter unten).

Theorem 8.5 (PCP-Satz: Approximations-Sicht)

Es gibt eine Zahl $\varepsilon < 1$, so dass es für jedes Problem $L \in NP$ eine in Polynomialzeit berechenbare Funktion f gibt, die Worte $x \in \{0,1\}^*$ auf 3CNF-Formeln $f(x)$ abbildet, so dass f.a. $x \in \{0,1\}^*$ gilt:

- Falls $x \in L$, so $val(f(x)) = 1$
- Falls $x \notin L$, so $val(f(x)) < \varepsilon$.

(Hier ohne Beweis)

Bemerkung 8.6

Die Aussage von Theorem 8.5 kann als eine stärkere Variante des Satzes von Cook und Levin (NP-Vollständigkeit von 3SAT) angesehen werden.
Der Beweis von Theorem 8.5 ist sehr aufwändig (und strukturell sehr verschieden vom Beweis des Satzes von Cook und Levin).

Als einfache Anwendung von Theorem 8.5 erhalten wir die folgende Aussage, die besagt, dass es schwierig ist, MAX-3SAT zu approximieren:

Folgerung 8.7

Es gibt ein $\varrho < 1$, so dass folgendes gilt:
Falls $P \neq NP$, so gibt es keinen Polynomialzeit-Algorithmus, der eine ϱ-Approximation für MAX-3SAT berechnet.

Beweis:
Sei ϱ gemäß Theorem 8.5 gewählt.
Sei $L \in NP$ und sei f die von Theorem 8.5 gelieferte Funktion.

Angenommen, A ist ein Polynomialzeit-Algorithmus, der eine ϱ-Approximation für MAX-3SAT berechnet.

Dann wird L durch folgenden Polynomialzeit-Algorithmus B entschieden:

Bei Eingabe $x \in \{0,1\}^*$ tue Folgendes:

(1.) Berechne $\varphi := f(x)$

(2) Berechne $\beta := A(\varphi)$

(3) Werte φ unter Belegung β aus.
Sei m' die Anzahl der Klauseln von φ, die von β wahrgemacht werden,
sei m die Anzahl aller Klauseln von φ, und
sei $v := \frac{m'}{m}$

(4) Falls $v \geq \varrho$, so gib "ja" aus;
ansonsten gib "nein" aus.

Klar: B ist ein deterministischer Polynomialzeit-Algorithmus.

Behauptung: F.a. $x \in \{0,1\}^*$ gilt:
$x \in L \iff$ B gibt bei Eingabe x "ja" aus.

Beweis:

"$\Rightarrow$": Sei $x \in L$.
Gemäß Theorem 8.5 gilt dann: $val(\varphi) = 1$
Da Algo. A eine ϱ-Approximation für MAX-3SAT berechnet, gilt dann:
$\beta := A(\varphi)$ macht mindestens $\varrho \cdot val(\varphi) \cdot m = \varrho \cdot 1 \cdot m$ Klauseln von φ wahr.
Somit ist $m' \geq \varrho \cdot m$ und $v := \frac{m'}{m} \geq \varrho$.
Daher gibt B "ja" aus.

"$\Leftarrow$": Sei $x \in \{0,1\}^*$ so, dass B bei Eingabe x "ja" ausgibt.
Dann gilt: $v = \frac{m'}{m} \geq \varrho$, d.h. $\beta := A(\varphi)$ macht
$m' \geq \varrho \cdot m$ Klauseln von φ wahr.
Insbesondere ist $val(\varphi) \geq \frac{m'}{m} \geq \varrho$.

Angenommen, $x \notin L$. Dann muss gemäß Theorem 8.5 gelten: $val(\varphi) < \varrho$. ↯ Widerspruch

Somit ist $x \in L$.

□ Behauptung.

Insgesamt haben wir also gezeigt, dass $L \in P$ ist.

□

Bemerkung 8.7': Mit stärkeren Methoden kann man sogar folgendes zeigen: Falls $P \neq NP$, so gibt es kein $\varepsilon > 0$, so dass in Polynomialzeit eine $(\frac{7}{8}+\varepsilon)$-Approximation für MAX-3SAT berechnet werden kann (vgl. dazu Bemerkung 8.4!)

8.2 Probabilistisch überprüfbare Beweise

Zur Erinnerung: $L \in NP \Leftrightarrow$

Es gibt eine (det.) Polyn.zeit TM V ("Verifizierer"), s.d. f.a. Eingaben $x \in \{0,1\}^*$ gilt:

- Falls $x \in L$, so gibt es einen "Beweis" π (polyn. Länge), s.d. $V(x,\pi) = 1$
- Falls $x \notin L$, so gilt für alle "Beweiskandidaten" π (polyn. Länge), dass $V(x,\pi) = 0$.

Jetzt: Eine etwas andere Sichtweise:

- Schreibe $V^\pi(x)$ statt $V(x,\pi)$.
  V^π ist dann ein Verifizierer, der Zugriff auf das "Zertifikat" π hat.
- Der Verifizierer darf ein randomisierter Algorithmus sein.
- Das Zertifikat π darf beliebig lang sein.
- Der Verifizierer hat "random access"-Zugriff auf das Zertifikat π. D.h. Er hat ein spezielles Adressband, auf das er im Laufe seiner Berechnung einen Bitstring schreiben kann, der eine Zahl i repräsentiert. Wenn er dann in den speziellen Zustand q_π geht, ist er im nächsten Schritt im Zustand q_0 bzw q_1, je nach dem ob das i-te Zeichen von π eine 0 oder eine 1 ist.

Definition 8.8 (PCP-Verifizierer)

Sei $L \subseteq \{0,1\}^*$, sei $q: \mathbb{N} \to \mathbb{N}$ und sei $r: \mathbb{N} \to \mathbb{N}$.

L hat einen $(r(n), q(n))$-PCP-Verifizierer,

falls es einen randomisierten Polynomialzeit-Algorithmus V gibt, so dass f.a. $n \in \mathbb{N}$ und alle $x \in \{0,1\}^n$ gilt:

(1) Bei Eingabe von x und eines Worts $\pi \in \{0,1\}^*$ mit $|\pi| \leq q(n) \cdot 2^{r(n)}$ hat V "random-access"-Zugriff auf π.
V benutzt höchstens $r(n)$ Münzwürfe, schaut sich höchstens $q(n)$ Stellen von π an, macht dann $\text{poly}(n)$ deterministische Berechnungsschritte (die vom Ausgang der vorherigen Münzwürfe und den $q(n)$ gesehenen Zeichen von π abhängen dürfen) und gibt dann entweder 1 (für "akzeptieren") oder 0 (für "verwerfen") aus und hält an.

$V^\pi(x)$ bezeichnet die Zufallsvariable, die die Ausgabe von V bei Eingabe von x und π angibt.

(2) Falls $x \in L$, so gibt es ein $\pi \in \{0,1\}^*$ mit $|\pi| \leq q(n) \cdot 2^{r(n)}$, so dass

$$\Pr[V^\pi(x) = 1] = 1 \quad \text{ist.}$$

Wir bezeichnen dieses π als "korrekten Beweis für x".

(3) Falls $x \notin L$, so gilt für jedes $\pi \in \{0,1\}^*$ mit $|\pi| \leq q(n) \cdot 2^{r(n)}$, dass

$$P_c\left[V^{\pi}(x) = 1\right] \leq \frac{1}{2}.$$

Beispiel 8.9 (Ein (poly(n), 1)-PCP-Verifizierer für Graph-Nicht-Isomorphie)

Das Problem GNI ist wie folgt definiert:

$$GNI := \{ \langle G_0, G_1 \rangle : G_0, G_1 \text{ sind zwei endliche gerichtete Graphen mit Knotenmenge } \{1,\dots,n\} \text{ für } n \in \mathbb{N}_{\geq 1}, \text{ so dass } G_0 \not\cong G_1 \}$$

Ein (poly(n), 1)-PCP-Verifizierer V geht bei Eingabe von $\langle G_0, G_1 \rangle$ mit Knotenmenge $\{1,\dots,n\}$ und bei Eingabe eines Zertifikats $\pi \in \{0,1\}^*$ wie folgt vor:

- Sei $H_1, \dots, H_k$ eine Liste aller möglichen Graphen mit Knotenmenge $\{1,\dots,n\}$ (die Binärdarstellung von i der Länge n^2 ist die Adjazenzmatrix von H_i, zeilenweise gelesen)
- Wähle zufällig eine Permutation f von $\{1,\dots,n\}$
- Wähle zufällig ein $b \in \{0,1\}$
- Berechne $H := f(G_b)$ und berechne i s.d. $H = H_i$

- Schau das i-te Bit von π an, d.h. den Wert π_i
- Akzeptiere genau dann, wenn $\pi_i = b$ ist.

Falls $G_0 \cong G_1$ ist, so sieht man leicht, dass

$$\begin{aligned} & Pr\left[V^{\pi}(\langle G_0, G_1\rangle)\right] \\ &= Pr\left[V \text{ akzeptiert bei Eingabe von } \langle G_0, G_1\rangle \text{ und } \pi\right] \\ &= Pr\left[b = \pi_i\right] \\ &= \frac{1}{2}, \end{aligned}$$

denn: $Pr[b=0] = Pr[b=1] = \frac{1}{2}$, und

für jedes $i \in \{1, \dots, k\}$ ist

$$|\{f : f \text{ ist Permutation von } \{1,\dots,n\} \text{ mit } f(G_0) = H_i\}| = |\{f : f \text{ ist Permutation von } \{1,\dots,n\} \text{ mit } f(G_1) = H_i\}|$$

Somit ist Bedingung (3) erfüllt

Falls $G_0 \not\cong G_1$ ist, so sei $\pi \in \{0,1\}^*$ so gewählt, dass f.a. $i \in \{1,\dots,k\}$ gilt:

- Falls $H_i \cong G_0$, so $\pi_i = 0$
- Falls $H_i \cong G_1$, so $\pi_i = 1$.

Für ein solches π gilt für jedes f und jedes b:
Für dasjenige i mit $H_i = f(G_b)$ ist $H_i \cong G_b$ und daher $\pi_i = b$ und V akzeptiert.

Somit ist $Pr\left[V^{\pi}(\langle G_0, G_1\rangle) = 1\right] = 1$ und Bedingung (2) ist erfüllt.
Insbesondere ist V ein $(\text{poly}(n), 1)$-PCP-Verifizierer für GNI.

Definition 8.10 (Die Klasse PCP(r(n), q(n)))

Die Klasse PCP(r(n), q(n)) besteht aus allen Sprachen $L \subseteq \{0,1\}^*$, für die es Zahlen $c, d > 0$ gibt, so dass L einen $(c \cdot r(n), d \cdot q(n))$-Verifizierer besitzt.

Bemerkung 8.11

(a) Bedingung (3) von Definition 8.8 besagt, dass jeder Beweiskandidat im Fall $x \notin L$ mit Wahrscheinlichkeit $\geq \frac{1}{2}$ verworfen werden muss. Dies ist meist schwierig nachzuweisen.

(b) Die Bedingung, dass $|\pi| \leq q(n) \cdot 2^{r(n)}$ ist, stellt keine wesentliche Einschränkung dar, da der Verifizierer mit höchstens $2^{r(n)}$ Ausgängen der $r(n)$ Münzwürfe und mit $q(n)$ Anfragen an π sich insgesamt höchstens $q(n) \cdot 2^{r(n)}$ Bits von π mit WK > 0 ansehen kann.

(c) Bei Bedingung (3) in Definition 8.8 kann an Stelle des Werts $\frac{1}{2}$ auch jede andere Zahl δ mit $0 < \delta < 1$ gewählt werden.

(d) Man sieht leicht, dass Folgendes gilt:

$$PCP(r(n), q(n)) \subseteq NTIME\left(2^{O(r(n))} \cdot O(q(n))\right),$$

denn: Eine NDTM kann sich einen Beweiskandidaten π der Länge $\leq q(n) \cdot 2^{r(n)}$ raten und dann für alle $2^{r(n)}$ möglichen Ausgänge der $r(n)$ Münzwürfe testen, ob x akzeptiert wird und somit $Pr[V^{\pi}(x) = 1]$ ermitteln.

Insbesondere gilt:

$$PCP(\log n, 1) \subseteq NTIME\left(2^{O(\log n)} \cdot O(1)\right) \subseteq NP$$

Der PCP-Satz besagt, dass hier an Stelle von "$\subseteq$" sogar "$=$" steht:

Theorem 8.12 (PCP-Satz: PCP-Sicht)

$$NP = PCP(\log n, 1)$$

(Hier ohne Beweis)

Bemerkung 8.13

Insbesondere ist bekannt, dass 3SAT einen $(O(\log n), 3)$-PCP-Verifizierer besitzt.

Insbesondere heißt das, dass für jede erfüllbare 3CNF-Formel ein Zertifikat π angegeben werden kann, so dass ein Verifizierer, der $O(\log n)$ Zufallsbits benutzt und sich nur 3 (!) Bits des Zertifikats π ansieht, von dessen Korrektheit überzeugt werden kann.

Bemerkung 8.14 ("New shortcut for long math proofs")

Sei $\mathcal{B}$ ein mathematisches Beweissystem, so dass die Korrektheit eines gegebenen Beweises deterministisch in Zeit polynomiell in der Länge des Beweises überprüft werden kann.

Dann gehört die folgende Sprache $L_{\mathcal{B}}$ zu NP:

$$L_{\mathcal{B}} := \{ \langle x, 1^n \rangle : \begin{array}{l} x \text{ ist eine mathematische Aussage,} \\ n \in \mathbb{N}, \text{ es gibt einen Beweis} \\ \text{in } \mathcal{B} \text{ der Länge} \leq n \text{ von } x \end{array} \}$$

Aus dem PCP-Satz (Theorem 8.11) folgt, dass $L_{\mathcal{B}} \in PCP(\log n, 1)$ liegt. D.h. es gibt Zahlen $c, d > 0$ s.d. $L_{\mathcal{B}}$ einen $(c \cdot \log n, d)$-PCP-Verifizierer besitzt, der $c \cdot \log n$ Zufallsbits benutzt und sich nur d Bits eines Zertifikats ansieht, um sich von dessen Korrektheit zu überzeugen.

Man kann nachweisen, dass die beiden Varianten des PCP-Satzes (Theorem 8.12 und Theorem 8.5) äquivalent zueinander sind. D.h.:
Gegeben Theorem 8.12, kann man relativ leicht Theorem 8.5 beweisen.
Und umgekehrt kann man auch Theorem 8.12 relativ leicht beweisen, wenn Theorem 8.5 gegeben ist.

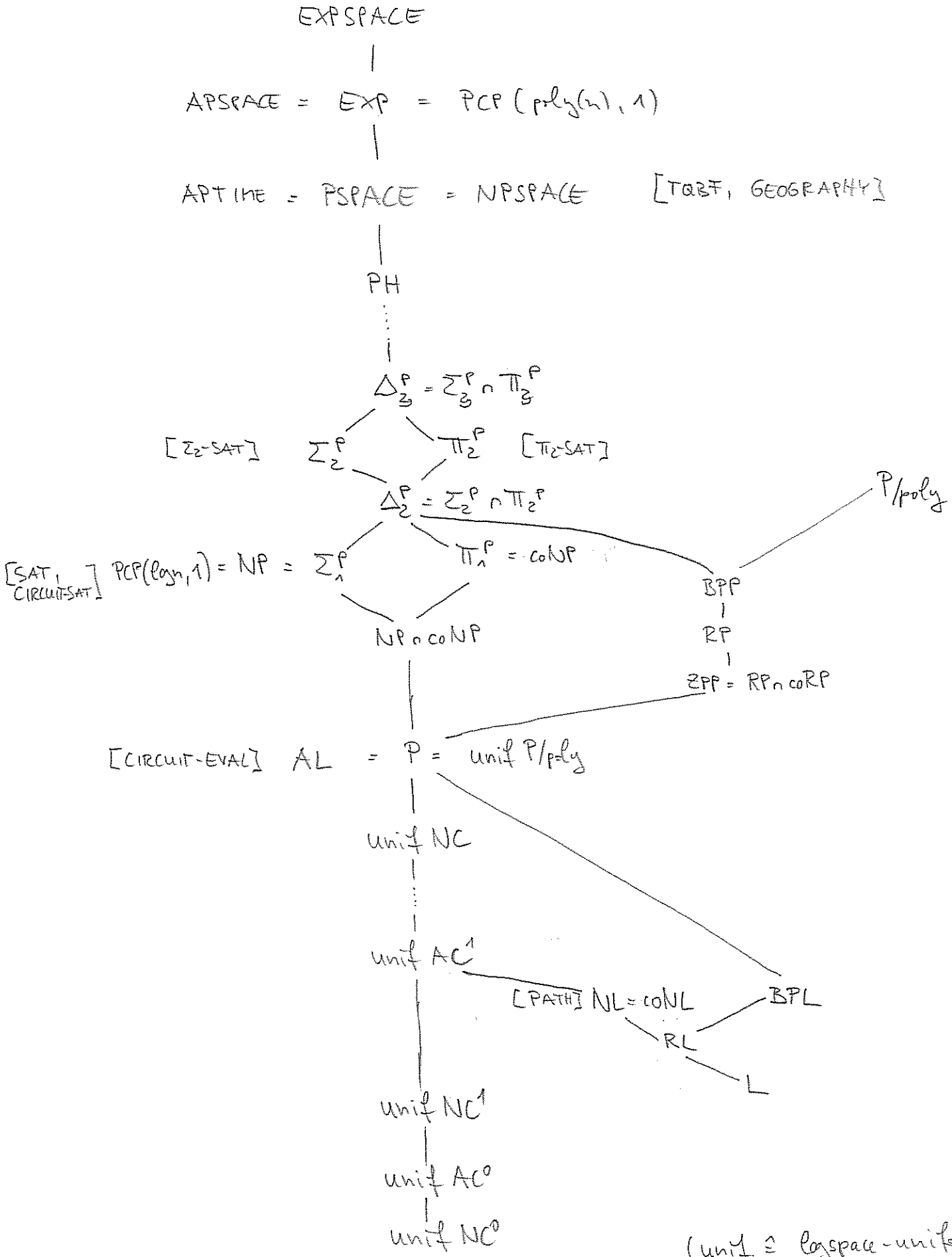
Es sind auch Varianten des PCP-Satzes für größere Komplexitätsklassen bekannt, etwa:

Theorem 8.15

$$PCP(poly(n), 1) = NEXP$$

(Hier ohne Beweis)

Überblick über die in dieser Veranstaltung betrachteten Komplexitätsklassen [und einige vollständige Probleme]



(unif ≙ logspace-uniform bzw. polynomial-uniform