Vorlesung: Komplexitätstheorie **Wintersemester 2006/07**

Lecture 10 — 24. November

Lecturer: Hermann Gruber *Scribe: Hermann Gruber*

10.1 Rolle Rückwärts.

Zur Rekapitulation: Wir haben in der letzten Vorlesung überlegt, welche Berechnungskraft eine (nicht)deterministische polynomialzeitbeschränkte Orakel-Turingmaschine besitzt, deren Orakel "nicht zu schwierig" ist.

Zur Erinnerung: Im Kapitel 2.4 über Relativierung haben wir gesehen, dass es ein Orakel gibt, relativ zu dem $\mathsf{P} = \mathsf{NP}$ gilt (Theorem 5). In der Tat war das Orakel das dort benutzt wurde, PSPACE-vollständig. Wenn das Orakel also "schwierig" genug ist, verschwindet der potentielle Unterschied zwischen P und NP. Intuitiv ist das so erklärbar, dass das Orakel die Berechnungskraft der anfragenden Maschine verschattet.

Wir haben deshalb definiert:

Definition 10.1. *Für eine der Komplexitätsklassen C, die wir aus der Vorlesung kennen, definieren wir* $\mathsf{P}^C = \bigcup_{A \in C} \mathsf{P}^A$, *und analog* NP^C, PSPACE^C

Und uns die Frage gestellt: Was leisten $\mathsf{NP}^{\mathsf{NP}}$, P^{NP}, und ähnliche Klassen?

Wir haben die polynomielle Hierarchie induktiv definiert:

Definition 10.2.

$$\begin{aligned}
\Sigma_0^{\mathsf{P}} &= \{\{0,1\}^*\} \\
\Pi_0^{\mathsf{P}} &= \{\emptyset\} \\
\Sigma_1^{\mathsf{P}} &= \mathsf{NP}, \\
\Pi_1^{\mathsf{P}} &= \mathsf{coNP}, \\
\Delta_1^{\mathsf{P}} &= \mathsf{P}, \\
\Sigma_{i+1}^{\mathsf{P}} &= \mathsf{NP}^{\Sigma_i^{\mathsf{P}}}, \\
\Pi_{i+1}^{\mathsf{P}} &= \mathsf{coNP}^{\Sigma_i^{\mathsf{P}}}, \textit{ und} \\
\Delta_{i+1}^{\mathsf{P}} &= \mathsf{P}^{\Sigma_i^{\mathsf{P}}}.
\end{aligned}$$

Schliesslich ist die polynomielle Hierarchie *definiert als*

$$\mathsf{PH} = \bigcup_{i \geq 0} \Sigma_i^{\mathsf{P}}.$$

Soweit, so gut. Warum sollte z.B. $\Delta_2^{\mathsf{P}} = \mathsf{P}^{\mathsf{NP}}$ wirklich mehr als NP berechnen können?

10.2 Was kann die polynomielle Hierarchie?

Nun, beispielsweise ist sowohl SAT als auch UNSAT in $\mathsf{P}^{\mathrm{SAT}}$:

Um SAT zu entscheiden, reicht die Maschine die SAT-Instanz an das Orakel weiter, und akzeptiert genau dann, wenn das Orakel mit "yes" antwortet. Um UNSAT zu entscheiden, dreht die Maschine die Antwort des Orakels einfach um. In $\Delta_2^{\mathsf{P}} = \mathsf{P}^{\mathsf{NP}}$ können noch komplizierter wirkende Sprachen berechnet werden, nämlich genau alle Sprachen die Turing-reduzierbar auf SAT sind. Erinnern wir uns an eine alte Übungsaufgabe:

Lemma 10.3. *Falls* $\mathsf{P}^{\mathsf{NP}} = \mathsf{NP}$*, so gilt* $\mathsf{NP} = \mathsf{coNP}$*.*

Und die meisten Komplexitätstheoretiker von heute glauben, dass $\mathsf{NP} \neq \mathsf{coNP}$. Nun, weg von Religionsfragen. Was können wir beweisen? Erstens, die polynomielle Hierarchie ist immer noch in PSPACE:

Proposition 10.4. $\mathsf{PH} \subseteq \mathsf{PSPACE}$

Beweis: Durch Induktion zeigen wir, dass für alle $i > 0$ gilt: Σ_i^{P} in PSPACE enthalten ist. Wir wissen ja bereits: $\Sigma_1^{\mathsf{P}} \subseteq \mathsf{PSPACE}$ aus Kapitel 3. Induktionsschritt $(i \to i+1)$:

$$\Sigma_{i+1}^{\mathsf{P}} = \mathsf{NP}^{\Sigma_i^{\mathsf{P}}} \subseteq \mathsf{NP}^{\mathsf{PSPACE}},$$

Da lt. Induktionsannahme $\Sigma_i^{\mathsf{P}} \subseteq \mathsf{PSPACE}$ und da $\mathsf{NP} \subseteq \mathsf{PSPACE}$ auch relativ zu jedem Orakel gilt:

$$\mathsf{NP}^{\mathsf{PSPACE}} \subseteq \mathsf{PSPACE}^{\mathsf{PSPACE}} = \mathsf{PSPACE}$$

10.3 Charakterisierung durch Quantoren

Proposition 10.5. *Eine Sprache L ist in Σ_i^{P} genau dann, wenn es eine polynomialzeit-berechenbare Sprache $A \in \mathsf{P}$ und ein Polynom $p(n)$ gibt, so dass*

$$x \in L \Leftrightarrow \exists y_1 : \forall y_2 \ldots Q y_k : \langle x, y_1, y_2, \ldots, y_k \rangle \in A$$

wobei die Quantifizierungen auf Wörter der Länge höchstens $q(|x|)$ eingeschränkt ist für ein bestimmtes Polynom q. Die Quantoren alternieren jeweils.

Wir werden im folgenden exzessiv von der Tatsache Gebrauch machen, dass man Tupel von Strings $\langle x_1, \ldots x_n \rangle$ auch als einzigen String (de)kodieren kann.

Beweis (für Σ_2^{P}):
"$\Leftarrow$": Nimm an, es gilt

$$x \in L \Leftrightarrow \exists y_1 \forall y_2 : \langle y_1, y_2, x\rangle \in A \vee y_1 + y_2 > p(n)$$

Wir wollen dazu eine nichtdeterministische, polynomial zeitbeschränkte Orakel-Turingmaschine (= NPOTM) mit Orakel in $\Sigma_1^{\mathsf{P}} = \mathsf{NP}$ konstruieren, die L erkennt.

Wir definieren C als

$$C = \{\langle x, y_1\rangle \mid \forall y_2 \text{ mit } |y_2| \leq p(x) : \langle y_1, y_2, x\rangle \in A\}.$$

Aus Kapitel 2, Proposition 1 wissen wir, dass das Komplement

$$\overline{C} = \{\langle x, y_1\rangle \mid \exists y_2 \text{ mit } |y_2| \leq p(x) : \langle x, y_1, y_2\rangle \in \overline{A}\}$$

in Σ_1^{P} ist, da das Komplement $\overline{A}$ auch in P ist. Wir werden $\overline{C}$ als Orakel für unsere Konstruktion verwenden: Die NPOTM M rät bei Eingabe x einen String y_1, und fragt das Orakel:

Liebes Orakel, falls du grade Zeit hast, bitte sage mir, ist $\langle x, y_1\rangle \in \overline{C}$?

... und akzeptiert genau dann wenn das Orakel mit nein antwortet.

Die Orakelanfrage hängt vom geratenen Wort y_1 ab; die Antwort des Orakels wird nachher invertiert.

"$\Rightarrow$": Nimm an, T ist eine NPOTM mit Orakel $C \in \Sigma_1^{\mathsf{P}}$ und $TIME_T(x) = p(|x|)$.

Wir definieren jede Menge Sprachen, die zusammen die Berechnung der Maschine beschreiben:

1. $\langle x, w\rangle \in$ VALC, falls w einen gültigen Berechnungspfad der Maschine T auf Eingabe x kodiert. Dabei heißt gültig: T darf vom Anfragezustand aus immer in den Zustand q_{yes} oder q_{no} übergehen, egal was das Orakel tatsächlich antwortet.
2. $\langle x, w, u\rangle \in \text{YES} - \text{QUERIES}$, falls $\langle x, w\rangle \in$ VALC und $u = \langle u_1, u_2, \ldots, u_{h_u}$ mit der Folge von Orakelanfragen auf dem Berechnungspfad w übereinstimmt, so dass der jeweilige Folgezustand q_{yes} ist.
3. $\langle x, w, v\rangle \in \text{NO} - \text{QUERIES}$ analog, mit $u = \langle v_1, v_2, \ldots, v_{h_v}$ aber T geht auf dem Pfad w jeweils nachher in den Zustand q_{no}.
4. $u \in \text{YES} - \text{ANSWERS}$, falls u ein Tupel von Wörtern beschreibt, die alle in der Orakelsprache C sind
5. $v \in \text{NO} - \text{ANSWERS}$, falls v ein Tupel von Wörtern beschreibt, von denen keines in C ist.

Wir haben die Sprache VALC so designt, dass sie in P ist. Und in P können wir wahrscheinlich nicht prüfen, ob das Verhalten bezüglich des Orakels konsistent ist— sofern $\mathsf{NP} \neq \mathsf{coNP}$.

Offenbar gilt: $x \in L$ genau dann wenn es drei Wörter der Länge höchstens $q(|x|)$ gibt, so dass alle folgenden Bedingungen erfüllt sind:

$$\begin{aligned} \langle x, w \rangle &\in \text{VALC} \\ \langle x, w, u \rangle &\in \text{YES} - \text{QUERIES} \\ \langle x, w, v \rangle &\in \text{NO} - \text{QUERIES} \\ u &\in \text{YES} - \text{ANSWERS} \\ v &\in \text{NO} - \text{ANSWERS} \end{aligned}$$

Die ersten drei Konjunktionen sind in P, und wir können sie ersetzen durch eine einzige Sprache $X \in \mathsf{P}$ von Tupeln $\langle u, v, w, x \rangle$. Damit sind wir schon etwas näher am Ziel:

$$x \in L \Leftrightarrow \exists u, v, w : \langle u, v, w, x \rangle \in X \wedge u \in \text{YES} - \text{ANSWERS} \wedge v \in \text{NO} - \text{ANSWERS}$$

... wie immer mit $|u|, |v|, |w| \leq |x|^{O(1)}$. Bleiben noch die Orakel-Antworten. Laut Kap. 2, Prop. 1 gilt:

$$u \in \text{YES} - \text{ANSWERS} \Leftrightarrow \exists y : \langle u, y \rangle \in U$$

mit $U \in \mathsf{P}$, $|y| \in |x|^{O(1)}$ (gähn) und die Negation:

$$v \in \text{NO} - \text{ANSWERS} \Leftrightarrow \forall z : \langle v, z \rangle \notin U$$

mit $|z| \in |x|^{O(1)}$ (gääääähn). Setzt man das oben ein:

$$x \in L \Leftrightarrow \exists u, v, w : \langle u, v, w, x \rangle \in X \wedge \exists y : \langle u, y \rangle \in U \wedge \forall z : \langle v, z \rangle \notin U$$

mit $|\,.\,| \in |x|^{O(1)}$ (echt kein Bock mehr!).

Fingerübung: Stellen Sie die Quantoren um, so die Formel so ausschaut wie es im Theorem steht. Zeigen Sie durch Induktion dass das gewünschte auch für alle Σ_i^{P} gilt.

Theorem 10.6. *Falls* $\Sigma_i^{\mathsf{P}} = \Pi_i^{\mathsf{P}}$, *so gilt* $\mathsf{PH} = \Sigma_i^{\mathsf{P}}$.

Das heißt in der Fachsprache: Die polynomielle Hierarchie kracht zusammen. Oder kurz: PH-Kollaps. Ein ausführlicher Bericht zum PH-Kollaps ist in [1] erschienen.

Beweis: Angenommen, $\Sigma_i^{\mathsf{P}} = \Pi_i^{\mathsf{P}}$. Wir zeigen durch Induktion über k, dass $\Sigma_{i+k}^{\mathsf{P}} = \Pi_{i+k}^{\mathsf{P}} = \Sigma_i^{\mathsf{P}}$.
$i = 0$: Offensichtlich, da $\Sigma_0^{\mathsf{P}} = \{\emptyset\}$ und $\Pi_0^{\mathsf{P}} = \{\{0,1\}^*\}$. Und aus etwas falschem kann man alles folgern.
$(i \to i+1)$: Sei $L \in \Sigma_{i+k+1}^{\mathsf{P}}$. Ernten wir die Früchte des soeben mühevoll hergerechneten Ergebnisses, so schliessen wir: es gibt eine Menge R von Paaren $\langle x, y\rangle$, so dass

$$x \in L \Leftrightarrow \exists y : \langle x, y\rangle \in R$$

mit $|y|$ ...na, duweisstschonwas; und das tolle ist: R ist in Π_{i+k}^{P}.

Übung: Proposition 10.5 sieht aber ein bisschen anders aus als das soeben gesagte, oder?

Dann ist laut Induktionsannahme R auch in Σ_i^{P}, und wir ziehen die beiden Existenzquantoren, die ganz aussen stehen zu einem zusammen, benutzen nochmal Proposition 10.5, und schwupp, $L \in \Sigma_i^{\mathsf{P}}$. Da Σ_i^{P} unter Komplement abgeschlossen ist, gilt auch $\Sigma_{i+k}^{\mathsf{P}} = \Pi_{i+k}^{\mathsf{P}}$

10.4 Danksagungen und Lobhudeleien

Diese Mitschrift basiert auf den Unterlagen von Hermann Gruber, der hauptsächlich wörtlich, aber nicht nur, aus den hervorragenden Notizen von Jan Johannsen abgeschrieben hat, und der wiederum hat es wahrscheinlich von irgendjemand anderem, und verbessert aufgeschrieben. Einiges ist auch dem großartigen Buch [2] entnommen und teilweise vereinfacht worden. Nicht zuletzt kann es auch passiert sein, dass Wissen aus der Akropolis, einem wahren Meilenstein des pädagogischen Rüberbringens [3] hier untergekommen ist.

Bibliography

[1] Aaronson, S. "Polynomial Hierarchy Collapses—Thousands Feared Tractable," http://www.scottaaronson.com/writings/phcollapse.pdf

[2] Bovet, D.P. and Crescenzi, P. "Introduction to the Theory of Complexity" Prentice-Hall, 1998

[3] Papadimitriou, C. "Computational Complexity" Addison-Wesley, 1994