FGI-1 – Formale Grundlagen der Informatik I

Logik, Automaten und Formale Sprachen

Aufgabenblatt 13 : Komplexität

Präsenzaufgabe 13.1: Zeige: Wenn $\mathcal{P} \neq \mathcal{NP}$, dann ist $SAT \notin \mathcal{P}$.

Lösung: Die äquivalente Umkehrung lautet: Wenn $SAT \in \mathcal{P}$, dann ist $\mathcal{P} = \mathcal{NP}$. Nach Satz 24.3 ist SAT $\mathcal{NP}$-vollständig. Mit Korollar 24.5 folgt die Behauptung.

Präsenzaufgabe 13.2: Sei DNF – analog zu KNF – die Sprache aller erfüllbarer Formeln in *disjunktiver* Normalform. Liegt DNF in $\mathcal{P}$? Liegt DNF in $\mathcal{NP}$. Ist DNF $\mathcal{NP}$-vollständig?

Lösung: Zunächst das einfache: DNF liegt in $\mathcal{NP}$, denn wie schon für SAT kann man eine Belegung raten und dann deterministisch überprüfen, ob diese erfüllend ist.

Dass DNF $\mathcal{NP}$-vollständig ist, ist dagegen sehr „unwahrscheinlich“, denn DNF liegt in $\mathcal{P}$ (s.u.) – wäre DNF also $\mathcal{NP}$-vollständig, dann hätten wir ja $\mathcal{P} = \mathcal{NP}$.

Warum ist nun DNF in $\mathcal{P}$? Für eine Formel in DNF reicht es aus, wenn eine Klausel erfüllbar ist und eine Klausel ist (da die Literale konjunktiv verknüpft sind) erfüllbar, wenn alle Literale wahr sind.

Wir können eine Klausel daher wahr machen, falls kein Atom sowohl negiert als auch unnegiert vorkommt, indem wir alle Atome, die unnegiert vorkommen, mit 1 belegen und alle Atome, die negiert vorkommen, mit 0 belegen.

Falls ein Atom sowohl negiert als auch unnegiert vorkommt, existiert keine erfüllende Belegung der Klausel.

Wir können also die Klauseln in der Formel von links nach rechts durchlaufen, um zu testen, ob in in mindestens einer Klausel kein Atom sowohl negiert als auch unnegiert vorkommt.

Da die Größe der Formel in $O(n)$ liegt, hat auch jede Klausel eine Größe in $O(n)$. Um jedes Atom auf mehrfaches Vorkommen zu testen, laufen wir für jedes Atom die Klausel einmal ab. Für jedes Atom kann dies in $O(n)$ geschehen. Da es höchstens $O(n)$ Atome gibt, kann der Test pro Klausel in $O(n^2)$ durchgeührt werden. Da es höchstens $O(n)$ Klauseln gibt, können alle Klauseln in $O(n^3)$ überprüft werden. Insgesamt ist Erfüllbarkeit also in kubischer Zeit, also insbesondere in Polynomzeit überprüfbar, liegt also in $\mathcal{P}$.

Präsenzaufgabe 13.3: Angenommen Sie haben gestern beim Nacharbeiten der FGI-Vorlesung eine Methode gefunden, um eine $t(n)$-zeitbeschränkte NTM A mit polynomiellen Aufwand zu simulieren, d.h. es gibt eine DTM B_A, die $L(A)$ akzeptiert und die für jeden Schritt von A höchstens $p(n)$ viele Schritte braucht. Diskutieren Sie die Konsequenzen!

Lösung: Herzlichen Glückwunsch: Sie haben gerade eine Million Dollar gewonnen: (s. `http://www.claymath.org/millennium/P_vs_NP/`).

In diesem Fall liegt nämlich jede Sprache $L \in \mathcal{NP}$ auch automatisch in $\mathcal{P}$. Also gilt $\mathcal{NP} \subseteq \mathcal{P}$ und mit der bekannten Inklusion $\mathcal{P} \subseteq \mathcal{NP}$ folgt direkt $\mathcal{P} = \mathcal{NP}$.

Präsenzaufgabe 13.4: In der Vorlesung haben wir definiert, dass eine TM eine Sprache L mit der Zeitschranke $t(n)$ akzeptiert, wenn die TM auf *allen* Eingaben w nach höchstens $t(|w|)$ Schritten anhält.

Wir definieren nun, dass eine TM eine Sprache L mit der Zeitschranke $t(n)$ **schwach** akzeptiert, wenn die TM auf *allen* ***akzeptierten*** Eingaben $w \in L$ nach höchstens $t(|w|)$ Schritten anhält.

Bezeichne $sDTime(t)$ die Menge aller Sprachen, die mit der Zeitschranke t schwach von einer DTM akzeptiert werden können. Sei $sNTime(t)$ die analoge nichtdeterministische Klasse.

Wir definieren die schwachen Varianten von $\mathcal{P}$ und $\mathcal{NP}$:

$$\mathcal{P}_s := \bigcup_{i \geq 1} sDTime(n^i) \qquad \mathcal{NP}_s := \bigcup_{i \geq 1} sNTime(n^i)$$

1. Zeigen Sie: $\mathcal{P}_s \subseteq \mathcal{NP}_s$
2. Zeigen Sie: $\mathcal{P} \subseteq \mathcal{P}_s$
3. Zeigen Sie: $\mathcal{NP} \subseteq \mathcal{NP}_s$
4. Zeigen Sie: $\mathcal{P}_s = \mathcal{P}$
5. Zeigen Sie: $\mathcal{NP}_s = \mathcal{NP}$

Lösung:

1. $\mathcal{P}_s \subseteq \mathcal{NP}_s$ gilt, da jede DTM eine NTM ist.
2. $\mathcal{P} \subseteq \mathcal{P}_s$ gilt, da die normale Akzeptierungsbedingung ein Spezialfall der schwachen Varianten ist.
3. $\mathcal{NP} \subseteq \mathcal{NP}_s$: s. Punkt 2)
4. $\mathcal{P}_s = \mathcal{P}$ gilt. Dazu ist nur noch $\mathcal{P} \supseteq \mathcal{P}_s$ zu zeigen.

   Wir wissen, dass $L \in \mathcal{P}_s$ genau dann gilt, wenn es eine DTM M gibt, die L in $t(n) = n^i$ schwach erkennt, d.h. alle Worte $w \in L$ in $t(n)$ akzeptiert werden. Auf den anderen Worte darf M beliebig lange rechnen.

   Da wir aber wissen, dass eine Eingabe nach mehr als $t(n) = n^i$ Schritten nicht mehr akzeptiert werden kann (denn sonst wäre ja schon zuvor akzeptiert worden), können wir die Berechnung nach $t(n) = n^i$ Schritten auch einfach abbrechen.

   Dazu müssen wir die Anzahl der Rechenschritte mitzählen. Wir geht dies im Detail?

   Wir bilden eine zweite Spur und berechnen zuerst $|w|^i$ in unärer Kodierung. Dies geht in Polynomzeit $p(|w|^i)$. (Man kann zeigen, dass es sogar in der Zeit $|w|^i$ geht.)

   Nach jedem Schritt dekrementieren wir den Zähler um Eins. Dies geht auch in Polynomzeit. (Genauer: Wenn $t(n) = c^i$ die Laufzeit ist, dann hat der unär kodierte Zähler für $|w|^i$ die Länge $|w|^i$. Ein Dekrementierschritt dauert dann höchstens $O(t(n))$ Schritte, da die TM dazu den ganzen Zähler, – ggf. mehrfach – entlanglaufen muss, um ein Symbol wegzustreichen. Also dauert jeder Schritt in der Simulation $ct(n)$ Schritte und die gesamte Laufzeit ist dann $t(n)(ct(n)) + p(|w|^i) = ct^2(n) + t(n) = O(t^2(n))$.)

   Wenn der Zähler bei Null ist, wird die Rechnung abgebrochen.

   Die so konstruierte TM M' akzeptiert jetzt L in Polynomzeit nach der „normalen“ Definition von Zeitbeschränkung.
5. $\mathcal{NP}_s = \mathcal{NP}$. Siehe Punkt 4)

Präsenzaufgabe 13.5: Finden Sie den Fehler in dem folgenden „Gottes-Beweis“.

1. Sei die Instanz w des Problems SAT gegeben.
2. Definiere $EVAL = \{\langle F, \mathcal{A}_1, \ldots, \mathcal{A}_k\rangle \mid \mathcal{A}_1, \ldots, \mathcal{A}_k \text{ sind alle Belegungen von } F\}$.
3. Wir wissen, dass wir die Formel F bei einer gegebenen Belegung $\mathcal{A}$ in Polynomzeit auswerten können.

   Sei die Laufzeit in n^l bei $n = |F| + |\mathcal{A}|$ für geeigneten Exponenten l.

   Die Laufzeit für die k-fache Evaluation ist dann: $k \cdot n^l$.
4. Da die Eingabe die Länge

   $$|F, \mathcal{A}_1, \ldots, \mathcal{A}_k| \simeq n + k \cdot n = (k+1)n$$

   besitzt, ist die Laufzeit der k-fachen Evaluation, gemessen in dieser Eingabelänge $N := (k+1)n$, wiederum höchstens von der Potenz l, d.h. höchstens N^l, denn es gilt:

   $$N^l = ((k+1)n)^l \geq (k+1)^l n^l \geq kn^l$$

   Also ist $EVAL \in \mathcal{P}$
5. Sei also B die DTM, die $EVAL$ in Polynom-Zeit akzeptieren kann.
6. Wir geben jetzt eine Reduktion von SAT auf $EVAL$ an. Wir generieren dazu alle möglichen Belegungen $\mathcal{A}_1, \ldots, \mathcal{A}_n$ und geben sie zur Eingabe hinzu:

   $$f(\langle F\rangle) := \langle F, \mathcal{A}_1, \ldots, \mathcal{A}_n\rangle$$

   Die neue Eingabe $f(w)$ wird dann von der DTM B geprüft.
7. Die Reduktion ist korrekt, denn es gilt:

   $$\langle F\rangle \in SAT \iff f(\langle F\rangle) \in EVAL$$

8. Also lässt sich das $\mathcal{NP}$-vollständiges Problem SAT auf das Problem $EVAL$ reduzieren, das wiederum in $\mathcal{P}$ liegt.
9. Also folgt aus Korollar 24.5 die Gleichheit: $\mathcal{P} = \mathcal{NP}$.

Lösung: Leider haben wir keine Polynomzeit-Reduktion angegeben, denn wir haben ja 2^n viele Belegungen, wenn n die Anzahl der Atome in F ist. Also haben wir keine Polynomzeit-Reduktion angegeben. Pech.