# Strukturelle Komplexitätstheorie

Dr. Gengler

Aufgabenblatt 11

Bearbeitung bis zum 30.06.2014

**Definition [sparse]:** Eine Sprache $L$ über einem Alphabet $\Sigma$ nennen wir *sparse (dünn)* gdw. es ein Polynom gibt, so dass:

$$\forall n \in N : |\{x \in L \mid |x| \leq n\}| \leq p(n)$$

**Aufgabe 1**

Zu einer Sprache $L$ definieren wir das Pseudo-Komplement $\widetilde{L}$ durch

$$\widetilde{L} := \{(s, \$^k, \$^n) \mid \exists\, x_1, \ldots, x_k \in L : \text{ paarweise verschieden } \wedge\ |x_i| \leq n \wedge x_i \neq s\}$$

1. Zeigen Sie, dass mit $L$ auch $\widetilde{L}$ in NP liegt.

2. Wir definieren: $c_L(n) := |\{w \in L \mid |w| \leq n\}|$. Zeigen Sie, dass für $n \geq |s|$ gilt::
   $$\begin{array}{lll} k < c_L(n) & \Longrightarrow & (s, \$^k, \$^n) \in \widetilde{L} \\ k = c_L(n) & \Longrightarrow & (\ (s, \$^k, \$^n) \in \widetilde{L} \Longleftrightarrow s \notin L\ ) \\ k > c_L(n) & \Longrightarrow & (s, \$^k, \$^n) \notin \widetilde{L} \end{array}$$
3. Zeigen Sie, dass, falls $L \in$ NP liegt und $c_L(n)$ polynomial-zeit-berechenbar und polynomial beschränkt ist, so auch $\overline{L}$ in NP liegt.

**Aufgabe 2**

Zeigen Sie: Ist $\overline{\text{SAT}}$ polyzeit-reduzierbar auf eine sparse Sprache $L$, so gilt P = NP.

**Hinweis:** Zu einer booleschen Formel $F$ betrachten wir die beiden Formeln $F_0$ und $F_1$, die aus $F$ entstehen in dem man die erste in $F$ auftauchende Variable auf FALSE bzw. TRUE setzt und so weit wie möglich vereinfacht. Beachten Sie, dass die so entstehenden Formeln bei geeigneter Codierung kürzer werden. Zu einer Formel $G$ betrachten wir dann den folgenden endlichen, geordneten, binären, knotenbewerteten Baum: die Wurzel ist mit der Formel $G$ bewertet; ein Knoten, der mit einer Formel $F$ mit Variabeln bewertet ist, hat zwei Söhne, die mit $F_0$ bzw. $F_1$ bewertet sind; Knoten, die FALSE oder TRUE bewertet sind, sind Blätter. Durch eine Tiefensuche nach einem mit TRUE bewertetem Blatt in diesem Baum zu $G$ können wir entscheiden, ob die Formel $G$ erfüllbar ist.

Sei $g$ die Reduktionsfunktion von $\overline{\text{SAT}}$ auf $L$. Wir können die Tiefensuche geschickt abkürzen: Wir führen eine Liste mit, in dem wir Worte aus $L$ aufsammeln. (Zu Beginn wissen wir, dass $g$(FALSE) in der Liste ist). Bei jedem neu besuchten Knoten mit Bewertung $F$ berechnen wir zunächst $g(F)$: Ist $g(F)$ schon in der Liste, so brauchen wir den Unterbaum nicht mehr zu untersuchen, wir wissen ja schon dass die Formel nicht erfüllbar ist. Zu untersuchen brauchen wir also nur Teilbäume von Knoten $F'$, bei denen $g(F')$ noch nicht auf der Liste steht. Gehen wir in der Tiefensuche zurück, so müssen wir für eine Formel $F''$ neu festgestellt haben, dass sie nicht erfüllbar ist (Wieso?), somit konnten wir $g(F'')$ in die Liste hinzufügen. Diese Liste ist aber polynomial beschränkt.

Wieviel Zeit braucht dieses Verfahren?

**Aufgabe 3**
Zeigen Sie: Gibt es eine NP-vollständige Sprache $S$, die sparse ist, so gilt P = NP.

**Hinweis:** Wir beschreiben ein Entscheidungsverfahren für SAT. Wir benutzen das Pseudo-Komplement $\widetilde{S}$, so wie es in Aufgabe 1 definiert ist. Sei $f$ eine poly-Zeit-Reduktion von SAT auf $S$ (mit Zeitschranke $r$) und $g$ eine poly-Zeit-Reduktion von $\widetilde{S}$ auf $S$ (mit Zeitschranke $q$). Wir betrachten Funktionen $h_i$, die einer booleschen Formel $F$ jeweils den Wert $g(\,f(F)\,,\ \$^i\,,\ \$^{r(n)}\,)$ zuweisen. Für $k = c_S(r(n)) := |\{w \in S \mid |w| \leq r(n)\}|$ können wir das Verfahren aus Aufgabe 2 mit $h_k$ durchführen: Zeigen Sie hierzu, dass:

$$F \in \overline{\text{SAT}} \Longleftrightarrow g(\,f(F)\,,\, \$^{c_S(r(n))}\,,\, \$^{r(n)}\,) \in S.$$

Leider wissen wir nicht, wie wir $k$ effizient bestimmen können. Deshalb führen wir nacheinander $h_i$ für $i = 1, 2, \ldots, p(r(n))$ aus (da $S$ sparse ist, gibt es ein Polynom $p$ mit $|\{x \in S \mid |x| \leq n\}| \leq p(n)$ ): findet ein $h_i$ ein Blatt TRUE, so ist die Formel erfüllbar und wir können akzeptierend stoppen; verbraucht ein $h_i$ zuviel Zeit, so ist es nicht das gewünschte $h_k$ und wir brechen $h_i$ ab und machen weiter; hat am Ende kein $h_i$ ein TRUE gefunden, so hat insbesondere auch $h_k$ kein TRUE gefunden, und die Formel ist nicht erfüllbar (Wieso?).