Strukturelle Komplexitätstheorie

Satz von Mahaney

Dünne Sprachen sind nicht NP**-hart, es sei denn** P = NP

Sebastian Kuhnert

07.12.2004

Inhaltsverzeichnis

Einleitung

Der Satz wurde erstmals 1982 von Stephen R. Mahaney bewiesen. Der hier vorgestellte Beweis lehnt sich eng an die Fassung an, die im Buch *Theory of Computational Complexity* von Du und Ko zu finden ist.[1]

[1] siehe Literaturverzeichnis

1 Begriffsklärungen

1.1 Dünne Sprachen

Dünne Sprachen (engl. *sparse sets*) sind Sprachen, die höchstens polynomiell viele Worte pro Wortlänge haben. Um dies formal definieren zu können, bezeichne für eine Sprache L die Zensus-Funktion $C_L(n) := |L \cap \Sigma^n|$ die Anzahl der Wörter der Länge n.

Definition (Dünne Sprache)
Eine Sprache $S \subseteq \Sigma^*$ heißt dünn, wenn ein Polynom $p_S(n)$ existiert, sodass $\forall n \in \mathbb{N} : C_S(n) \leq p_S(n)$.

Damit ist auch die Anzahl der Wörter mit höchstens n Zeichen durch ein Polynom beschränkt, da für jedes Polynom p eine Konstante c gefunden werden kann, sodass $\Sigma_{k=0}^{n} p(k) \leq np(n) + c$.

Dünnheit ist gegenüber allgemeinen Sprachen eine starke Einschränkung, deren Zensusfunktion nur durch 2^n beschränkt ist. Eine noch weitergehende Einschränkung gilt für Tally-Sprachen, deren Zensusfunktion durch 1 beschränkt ist.

1.2 Selbstreduktionsbäume

Um die Zugehörigkeit eines Wortes w zu einer dünnen Sprache L zu entscheiden, können Wörter $w_1, \ldots, w_k$ erzeugt werden, deren Zugehörigkeit zu L einfacher zu entscheiden ist als die von w. Diese Wörter sind die Kinder von w im Selbstreduktionsbaum. Das Zurückführen auf Teilprobleme wird bis zu einem geeigneten Rekursionsschluss fortgesetzt. Wenn die Kinder eines Wortes w entschieden sind, kann $\chi_L(w)$ durch Auswertung einer logischen Formel berechnet werden, die $\chi_L(w_1) \ldots \chi_L(w_k)$ als Variable enthält. Wird die Disjunktion verwendet, spricht man auch von d-Selbstreduktion.

Die Konstruktion des Selbstreduktionsbaums (engl. *self-reducing-tree*) für w wird in Abbildung 1 veranschaulicht.

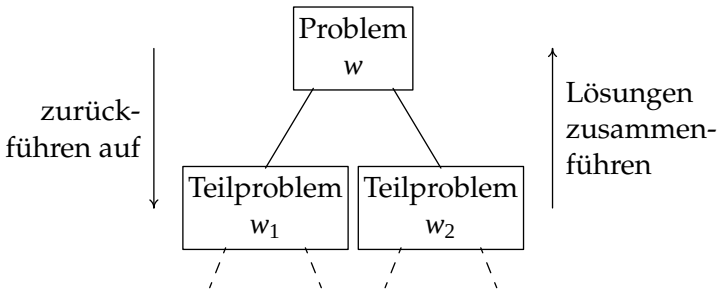

Abbildung 1: Prinzip der Selbstreduktion

2 Der Beweis

2.1 Beweisüberblick

Satz von Mahaney *(Mahaney's Theorem)*
Dünne Sprachen sind nicht NP-hart, es sei denn P = NP.

Als Beweis wird gezeigt, dass wenn eine dünne Sprache S mit SAT $\leq_m^{\mathrm{P}} S$ existiert, P = NP sein muss.

Dazu konstruieren wir eine Sprache L mit $L \leq_m^{\mathrm{P}}$ SAT. Damit gilt auch $L \leq_m^{\mathrm{P}} S$ via einer Reduktionsfunktion $f \in$ FP.

Im Anschluss zeigen wir, dass SAT unter Kenntnis von f in deterministischer Polynomialzeit entscheidbar ist. Wegen der NP-Vollständigkeit von SAT würde auch P = NP gelten.

2.2 Eine selbstreduzierbare Variante von SAT

2.2.1 Selbstreduktion für die Erfüllbarkeit von Formeln

Wir betrachten zunächst anhand von Abbildung 2, wie die Erfüllbarkeit der Formel $\varphi = x_1 \wedge \neg x_2$ mit einem Selbstreduktionsbaum entschieden werden kann. Die für die Selbstreduktion notwendige Zerlegung wird dadurch realisiert, dass die einzelnen Variablen sukzessive belegt werden. Dabei wird das Problem mit jedem Schritt einfacher, weil eine Variable weniger belegt werden muss. Wenn alle Variablen belegt sind (also ein Blatt erreicht ist), wird φ mit dieser Belegung ausgewertet. Wenn nun als logische Funktion zur Zusammenführung der Teilergebnisse die Disjunktion gewählt wird, wird genau die Erfüllbarkeit entschieden.

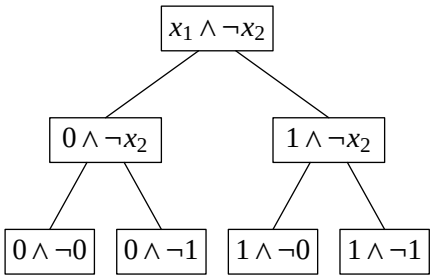

Abbildung 2: Selbstreduktionsbaum für eine Formel

Um diese Vorgehensweise auf beliebige Formeln zu verallgemeinern, führen wir die Notation $\varphi|_w$ für die Belegung der ersten m Variablen von φ mit dem Bitstring $w = b_1 b_2 \cdots b_m \in \{0,1\}^m$ ein. Das Ergebnis ist der Selbstreduktionsbaum in Abbildung 3.

Für den Beweis ist es zusätzlich notwendig, dass die Paare $\varphi|_w$ aus Formel φ und (partieller) Variablenbelegung w auf eine einheitliche Form gebracht werden können. Um dies zu erreichen, belegen wir alle noch nicht durch w belegten Variablen

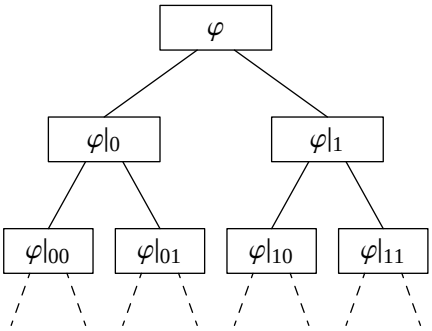

Abbildung 3: Selbstreduktionsbaum für beliebige Formeln

von φ mit 0. Die so entstandene Variablenbelegung nennen wir w'. Auf syntaktischer Ebene kann w' aus w durch Padding mit 0 erzeugt werden.

2.2.2 Die Sprache L

Zunächst führen wir noch zwei Notationen ein: $|\mathrm{Var}(\varphi)|$ für die Anzahl der Variablen in φ sowie $\preceq$ für die lexikographische Ordnung auf Bitstrings.

L soll nun Tupel $\langle \varphi | w \rangle$ aus einer Formel φ und einer Belegung w enthalten, die φ wahr macht. Dabei stellt sich jedoch folgendes Problem: Um allen Knoten im Selbstreduktionsbaum von φ ein Tupel dieser Form zuzuordnen, müssen die partiellen Belegungen gepaddet werden. Dabei kann es aber nun passieren, dass $\varphi|_{01}$ wahr sein kann, ohne dass $\varphi|_{00}$ wahr ist. Nun ist aber $\varphi|_{01}$ ein Kind von $\varphi|_0$, das zu $\varphi|_{00}$ gepaddet wird. Damit der Baum mit den sukzessiven Belegungen seine Reduktionsbaumeigenschaft behält, müssen die gepaddeten Varianten aller Tupel auf dem Weg zu einem Blatt in L ebenfalls in L aufgenommen werden. Da dies nur schwer zu prüfen ist, nehmen wir einfach alle Tupel $\langle \varphi | w \rangle$ auf, deren Belegung w kleinergleich einer erfüllenden Belegung u ist:

$$L = \left\{ \langle \varphi | w \rangle \;\middle|\; |w| = |\mathrm{Var}(\varphi)|, \left(\exists u \in \{0,1\}^{|\mathrm{Var}(\varphi)|} : w \preceq u, \varphi|_u = 1 \right) \right\}$$

L liegt in NP, weil für jede Eingabe $\langle \varphi, w \rangle$ ein $u \succeq w$ geraten werden kann, für das dann nur noch $\varphi|_u = 1$ geprüft werden muss. Damit gilt $L \leq_m^{\mathrm{P}} \mathrm{SAT}$.

Mit unserer Voraussetzung, dass eine NP-harte dünne Sprache S existiert, gilt wegen der Transitivität der $\leq_m^{\mathrm{P}}$-Reduktion auch $L \leq_m^{\mathrm{P}} S$ via einer Funktion $f \in \mathrm{FP}$. Polynomialzeit ist in diesem Beweis ausreichend, es gilt natürlich auch $L \leq_m^{\mathrm{L}} \mathrm{SAT}$ und $L \leq_m^{\mathrm{L}} S$.

Um im weiteren Verlauf auf die Schranken für f und S zurückgreifen zu können, definieren wir folgende Bezeichner: Sei $p_f(|\varphi|)$ die polynomielle Laufzeitschranke für $f(\langle \varphi | w \rangle)$ und $p_S(n)$ die polynomielle Schranke für $\Sigma_{k=0}^{n} C_S(k)$. Die Länge von w entspricht der Anzahl der Variablen in φ, sodass p nur von φ abhängt.

2.3 SAT – hier in Polynomialzeit

Um SAT mithilfe von f in Polynomialzeit zu entscheiden, bauen wir für Eingabe φ ebenenweise den Selbstreduktionsbaum auf, was einer Breitensuche über mögliche Belegungen entspricht. Dabei werfen wir nachdem eine Ebene aufgebaut ist so viele Knoten weg, dass nur polynomiell viele übrigbleiben. Dabei müssen wir sicherstellen, dass das Akzeptanzverhalten nicht verändert wird. Hierzu stellen wir zunächst einige Betrachtungen am Selbstreduktionsbaum an.

2.3.1 Eigenschaften des Selbstreduktionsbaumes

Im Selbstreduktionsbaum seien die Nachfolger aller Knoten jeweils nach der Größe ihrer Variablenbelegung bezüglich $\preceq$ geordnet.

Wenn φ erfüllbar ist, existiert eine bezüglich $\preceq$ größte gültigmachende Belegung w_0. Genau alle Knoten auf und links von dem Pfad von der Wurzel zu $\langle\varphi|w_0\rangle$ sind mit gepaddeter Belegung in L enthalten. In Abbildung 4 sind diese Knoten hervorgehoben.

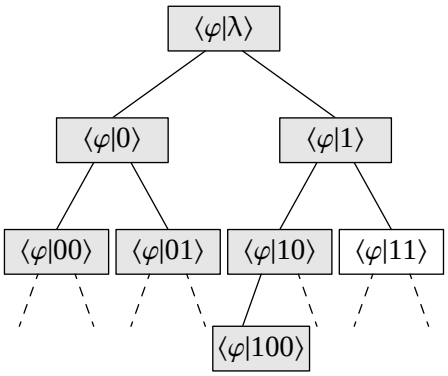

Abbildung 4: Selbstreduktionsbaum mit größtem erfüllenden Blatt

Nun können wir jedem Knoten $\langle\varphi|w\rangle$ ein Label $f(\langle\varphi|w'\rangle)$ zuweisen. Weil f die Reduktionsfunktion von L auf S ist, sind genau die Label in S, deren Knoten in L sind. Dieser Sachverhalt ist in Abbildung 5 angedeutet.

2.3.2 Abschneiden von Teilbäumen

Durch diese Label wird es möglich, die Dünnheit von S zum Abschneiden (engl. *prune*) von Teilbäumen zu nutzen. Die Funktion f kann nur polynomiell viele unterschiedliche in S liegende Label erzeugen, weil sie durch eine polynomielle Zeit- und damit auch Platzschranke begrenzt ist.

Die erste Möglichkeit, Teilbäume abzuschneiden, bietet sich dadurch an, dass unterschiedliche Knoten einer gerade erzeugten Ebene das gleiche Label erhalten können. In Abbildung 6 wird beispielhaft angenommen, dass die Label von $\langle\varphi|01\rangle$

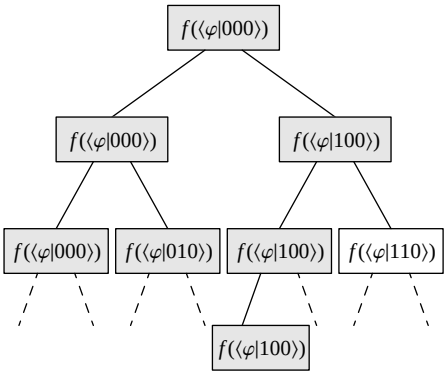

Abbildung 5: Label am Selbstreduktionsbaum

und $\langle\varphi|10\rangle$ gleich sind. Weil f eine Reduktionsfunktion ist, gilt

$$f(\langle\varphi|w_1'\rangle) = f(\langle\varphi|w_2'\rangle) \Rightarrow \Big(\langle\varphi|w_1'\rangle \in L \Leftrightarrow \langle\varphi|w_2'\rangle \in L\Big)$$

Damit können wir einen der beiden Knoten weglassen, ohne etwas daran zu ändern, ob ein Knoten der aktuellen Ebene in L liegt. Da wir später über das am weitesten rechts liegende Blatt in L argumentieren werden und deshalb keinen Knoten auf dem Pfad von der Wurzel zu diesem Knoten weglassen können, entscheiden wir uns für das Weglassen des kleineren der beiden Knoten. An diesem kann das größte Blatt in L nicht hängen.

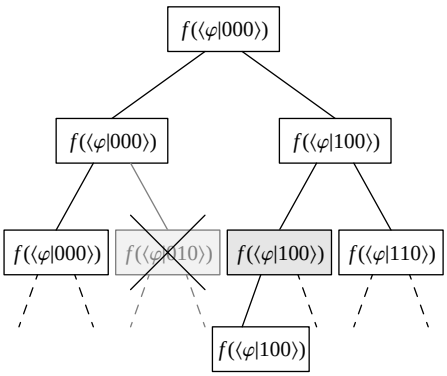

Abbildung 6: Abschneiden durch gleiche Label

Nach diesem ersten Schritt können immer noch mehr als $p_S(p_f(|\varphi|))$ Knoten in der aktuellen Ebene übrig bleiben. Es bietet sich aber noch eine zweite Möglichkeit, weitere Knoten wegzulassen: Nur die ersten $p_S(p_f(|\varphi|))$ Label können in S liegen, weil die in S liegenden Label am Anfang stehen, nur polynomiell viele der erzeugten Label in S liegen können und nach dem ersten Schritt keine Duplikate mehr vorkommen. Diese Tatsache ermöglicht uns, alle weiteren Knoten wegzulas-

sen, wie in Abbildung 7 angedeutet wird (markiert sind die Knoten, die nach dem ersten Schritt übrigbleiben).

Nach diesem zweiten Schritt bleiben nur polynomiell viele Knoten in der aktuellen Ebene übrig. Unter diesen ist auch der Vorgänger des größten in L liegenden Blattes, da dieser ebenfalls in L liegt und damit das Label in S liegt.

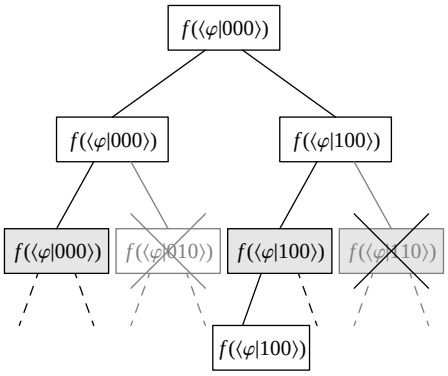

Abbildung 7: Abschneiden durch Dünnheit

2.3.3 Der Algorithmus

Nach diesen Vorüberlegungen können wir den folgenden Algorithmus formulieren:

Algorithmus für SAT **unter Kenntnis von** f

Eingabe φ.

1. Setze $W_0 \coloneqq \{\langle\varphi|\lambda\rangle\}$
2. Für $i = 1$ bis $|Var(\varphi)| - 1$:
   a) Setze $W_i \coloneqq \left\{\langle\varphi|u0\rangle, \langle\varphi|u1\rangle \;\middle|\; \langle\varphi|u\rangle \in W_{i-1}\right\}$
   b) Für jede Kombination $\langle\varphi|u\rangle, \langle\varphi|v\rangle \in W_i$ mit gleichem Label lösche das kleinere Element aus W_i
   c) Solange $|W_i| > p_S(p_f(|\varphi|))$, lösche das größte Element aus W_i
3. Setze $W_{|Var(\varphi)|} \coloneqq \left\{\langle\varphi|u0\rangle, \langle\varphi|u1\rangle \;\middle|\; \langle\varphi|u\rangle \in W_{Var-1}\right\}$
4. Wenn es ein $\langle\varphi|u\rangle \in W_{|Var(\varphi)|}$ mit $\varphi|_u = 1$ gibt, akzeptiere, sonst sonst verwerfe.

2.3.4 Laufzeit und Korrektheit

Die Laufzeit des Algorithmus ist durch ein Polynom beschränkt, weil die einzelnen Schritte nur polynomiell lange dauern und die Anzahl der Elemente in den einzelnen W_i durch $p_S(p_f(|\varphi|))$ beschränkt ist. Dadurch werden die einzelnen Schritte auch nur polynomiell häufig ausgeführt.

Wenn der Algorithmus akzeptiert, hat er eine gültigmachende Belegung gefunden und φ liegt in SAT. Es wird kein $\varphi \notin$ SAT akzeptiert.

Als Letztes bleibt zu zeigen, dass alle $\varphi \in$ SAT akzeptiert werden: In diesem Fall existiert ein größtes Blatt $\langle \varphi | w_0 \rangle$ mit $\varphi|_{w_0} = 1$. Wir haben bereits gezeigt, dass die Knoten auf dem Weg zu diesem Blatt nicht aus den W_i entfernt werden können. Also gibt es mit $\langle \varphi | u \rangle$ ein Element in $W_{|\mathrm{Var}(\varphi)}$, das zu 1 ausgewertet wird. Damit akzeptiert der Algorithmus.

Zusammenfassung

Wenn zu einem Problem eine Reduktion auf eine dünne Sprache bekannt ist, kann die Suche nach einer akzeptierenden Berechnung auf polynomiellen Aufwand reduziert werden.

Analog lässt sich auch zeigen, dass für beliebig große $k \in \mathbb{N}$ dünne Sprachen nicht $\leq^{\mathrm{P}}_{k\text{-}tt}$-hart für NP sein können, es sei denn P = NP.

Literatur

- Erstveröffentlichung:
  Stephen R. Mahaney: *Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis.* Journal of Computer and System Sciences, 25(2), 130-142, 1982

- Quelle für den hier vorgestellten Beweis:
  Ding-Zhu Du und Ker-I Ko: *Theory of Computational Complexity*,
  Wiley & Sons, 2000, pp. 261-262

- Alternativer Beweis von Ogihara und Watanabe
  aus dem Skript *Introduction to to Complexity Theory* der University of Wisconsin, Madison `www.cs.wisc.edu/~jyc/810notes/lecture11.pdf`