Komplexitätstheorie 3

Montag und Donnerstag
14:15 – 15:45 Uhr
in C-221

Weitere vollständige Sprachen (Probleme)

Definition:
Das Grapherreichbarkeitsproblem (*GAP, Graph Accessibility Problem*):

Eingabe:
Gerichteter Graph $G := (V, E)$ mit Knotenmenge $V = \{v_1, v_2, \ldots, v_n\}$ (*vertices*) und Kantenmenge $E \subseteq V \times V$ (edges) mit zwei ausgewählten Knoten v_a und v_e $(a, e \in \{1, \ldots, n\})$.

Frage:
Gibt es einen gerichteten Pfad von v_a nach v_e, d.h. existiert eine Folge $i_0, i_1, i_2, \ldots, i_k, i_{k+1} (1 \leq i_j \leq n)$ mit $i_0 = a$, $i_{k+1} = e$ und $(e_{i_j}, e_{i_{j+1}}) \in E$ für $0 \leq j \leq k$?

Satz:
Das Grapherreichbarkeitsproblem (GAP) ist vollständig für

$$NL = NSPACE\,(\log(n))$$

bezüglich der Reduktion $\leq_{log}$.

Beweisidee zu "GAP ist NL-vollständig"

Beweis:

a) $GAP \in \mathcal{NL}$: Wenn es einen Weg von v_a zu v_e gibt, so gibt es einen kreisfreien mit höchstens n Kanten. Im Algorithmus wird nun schrittweise eine Folge $i_0, i_1, i_2, \ldots, i_k, i_{k+1} (1 \leq i_j \leq n)$ geraten und jedesmal geprüft, ob $(v_{i_j}, v_{i_{j+1}}) \in E$ ist. Dazu müssen nur drei Zahlen $j, i_j, i_{j+1} \leq n$ binär gespeichert werden. Das kostet nur $O(\log n) = O\big(\log(|G|)\big)$ Platz.

b) $\mathcal{NL} \subseteq LOG(GAP)$: für $L \in \mathcal{NL}$ gibt es NTM A, deren Graph von möglichen Konfigurationen, die aus der Startkonfigurationen erreichbar sind, durchsucht wird. Jede einzelne Konfiguration braucht nur $c \cdot \log n$ Speicherplatz (vergl. frühere Berechnungen). Ein Wort w ist in L, gdw. es Erfolgsrechnung gibt, gdw. es Im Konfigurationspfad von A einen Pfad von Startkonfiguration zu einer Endkonfiguration gibt. Das ist aber eine Frage, die mit GAP entschieden wird.

weitere NL-vollständige Probleme:

Das früher erwähnte Problem *2-SAT* ist ebenfalls NL-vollständig!

Definition:
Das Problem der Nichtleerheit regulärer Sprachen($L(A) \neq \emptyset$?):

Eingabe:
Ein endlicher Automat (*NFA* oder *DFA*) A

Frage:
Gilt $L(A) \neq \emptyset$?

Definition:
Das Universalitätsproblem für *DFA*'s ($L(A) \neq \Sigma^*$?):

Eingabe:
Ein deterministischer endlicher Automat (*DFA*) A

Frage:
Gilt $L(A) \neq \Sigma^*$?

NL-Vollständigkeit durch Sternabschluss:

Definition:

Sei $L_0 := \{v_0\#v_1\#\cdots\#v_n\#v_0^{\text{rev}}\# \mid n \in I\!N, v_i \in \{a,b\}^*\}$.
$L_0 \in DSPACE(\log n)$ ist lineare. kontextfreie Sprache.

Satz:

L_0^+ ist vollständig für $NL = NSPACE(\log n)$ bzgl. $<_{\log}$- Reduktion.

Wäre also $L{=}NSPACE(\ell og\ n)$ gegenüber Kleene Plus abgeschlossen (wie z.B. jede AFL), dann wäre

$$DSPACE(\ell og\ n) = NSPACE(\ell og\ n)$$
und
$$DSPACE(n) = DLBA = Cs = NSPACE(n)$$

(Dieses sog. erste LBA-Problem ist noch immer ungelöst!)

P-vollständige Probleme / Mengen

Satz:

Die Frage, ob $L(G) \neq \emptyset$ für eine $CFG\ G$ gilt ist $P - vollst\ddot{a}ndig$.

Beweis:

a) Sei $L \in \mathcal{C}f$ und $G = (V_N, V_T, P, S)$ eine CFG mit $L = L(G)$. Mit dem Standardverfahren muss nur getestet werden, ob das Startsymbol produktiv ist. Dies ist in $DTIME\left(|G|^4\right)$ leicht möglich.
Es reichen $|V_N|^2 \cdot |P| \cdot \max(|w| : (A,w) \in P)$ Schritte aus.

b) Sei $L \in P$, also $L = L(M)$ für eine $p(n)$-zeitbeschränkte DTM M. Die Konfigurationen werden durch Wörter $\alpha \in \{\$\}\Gamma^* Z \Gamma^* \{\$\}$ beschrieben, wobei Z Zustandsmenge und $\$ \notin \Gamma$ ein Randsymbol ist. Sei $\$\alpha_1\cdots\alpha_p\$ \vdash_M \$\beta_1\cdots\beta_p\$$ ein Übergang in M, dann ist β_i o.B.d.A durch $\alpha_{i-1}\alpha_i\alpha_{i+1}$ eindeutig bestimmt (Turing-Tafel als endl. Tabelle).
Wir bilden die Tabelle Δ dieser Tripel $(\alpha_{i-1}, \alpha_i, \alpha_{i+1}, \beta_i)$.

Fortsetzung des Beweises

Bei einer Eingabe $v \in \Sigma^+, v = v_1 \cdots v_n$ beschreibt also $\$z_0 v_1 \cdots v_n \underbrace{\# \cdots \#}_{p(n)-n}\$$ die Startkonfiguration von M. Wir können weiter davon ausgehen, dass $v \in L$ genau dann gilt, wenn M nach $p := p(|v|)$ Schritten im Zustand $\not\in$ hält und der LS-Kopf ganz links steht.

Das bedeutet, dass $v \in L$ genau dann, wenn in folgender Rechnung

$$\$z_0 v_1 \cdots v_n \# \cdots \#\$ \vdash_M^{} \$\alpha_{1,0} \cdots \alpha_{1,p}\$ \vdash_M^{} \$\alpha_{p,0} \cdots \alpha_{p,p}\$,$$

$\alpha_{p,0} = \not\in$ ist.

Zu $v \in \Sigma^+$ wird nun die Grammatik $G(v) := (V_N, \emptyset, P, S)$ mit $P := \left\{ V(a,t,j) \middle| a \in \Gamma \cup Z \cup \{\$\},\ 0 \le t \le p \text{ und } -1 \le j \le p+1 \right\}$ und Startsymbol $S := V(\not\in, p, 0)$ konstruiert.

(Dass diese Konstruktion in logarithmischem Platz möglich ist, ist sehr kompliziert zu beweisen.) Wir geben noch die Produktionen von G bekannt:

Die Produktionen von $G(v)$

$$\begin{aligned}
V(d,t,j) &\longrightarrow V(a,t-1,j-1)V(b,t-1,j)V(c,t-1,j+1) \\
&\qquad \text{für alle } (a,b,c,d) \in \Delta,\ 0 < t \le p,\ 0 \le j < p, \\
V(\$,t,-1) &\longrightarrow \lambda \quad \text{für } 0 \le t \le p, \\
V(\$,t,p+1) &\longrightarrow \lambda \quad \text{für } 0 \le t \le p, \\
V(z_0,0,0) &\longrightarrow \lambda, \\
V(v_j,0,j) &\longrightarrow \lambda \quad \text{für } 1 \le j \le n, \\
V(\#,0,j) &\longrightarrow \lambda \quad \text{für } n \le j \le p.
\end{aligned}$$

Es gilt nun $L(G(v) \neq \emptyset \iff v \in L$, was durch längliche Induktion bewiesen werden kann.

Damit „kennen“ wir das erste vollständige Problem für deterministische polynomzeit Berechnungen.

Auswerteproblem für boolesche Schaltkreise

Definition:

Ein boolescher Schaltkreis C ist ein gerichteter Graph $C := (V, E)$, dessen Knoten Gatter heißen und mit den Zahlen $1, 2, \ldots, |V|$ und mit o bezeichnet werden. Für $(i, j) \in E$ gilt $i < j$, wodurch C azyklisch ist. Gatter haben den Eingangsgrad $0, 1$ oder 2 und den Ausgangsgrad 1.

Jedes Gatter i sei von einer der Sorten

$$s(i) \in \{\neg, \vee, \wedge, \mathit{true}, \mathit{false}\} \cup \{ x_1, x_2, x_3, \ldots\}.$$

Gatter der Sorte x_i sind Eingangsgatter und das Gatter o (*output*) heißt Ausgangsgatter. Sind die Eingangsgatter eine Teilmenge von $\{x_1, \ldots, x_n\}$, so bestimmt ein Bitstring $x \in \{0,1\}^n$ einen Wert des Gatters $C(x) \in \{0,1\}$.

Schaltkreise berechnen genau die boolesche Funktionen $\{0,1\}^n \to \{0,1\}$.

Komplexität boolescher Schaltkreise

Satz:

1. Die Frage, ob es für die Eingangsgatter eines booleschen Schaltkreises eine Variablenbelegung gibt, mit der am Ausgangsgatter der Wert 1 erzeugt wird, ist NP-vollständig.
2. Das Auswerteproblem, kurz CV (*circuit value*) d.h. die Berechnung des Ausgangswertes eines booleschen Schaltkreises bei einer Eingabe ist P-vollständig.

Beweis:

a) $CV \in P$ ist wegen der Zyklenfreiheit einfach zu zeigen. (Wie machen *Sie* das?)

Circuit Value

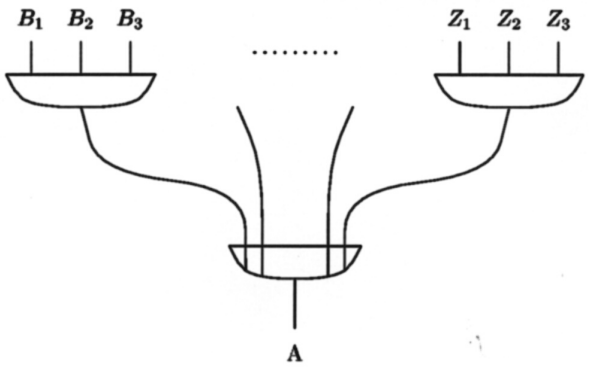
b) Sei $L \in P$. wie im Beweis für das (nicht)Leerheitsproblem von *CFG*'s konstruiert man zunächst eine Grammatik. Dort waren die Produktionen nur von zwei Arten:

$$A \longrightarrow B_1B_2B_3 \mid \cdots \mid Z_1Z_2Z_3,$$

oder

$$A \longrightarrow \lambda.$$

Jede Varable wird als Gatter interpretiert:



Ende des Beweises

Da Gatter maximal 2 Eingänge haben dürfen, muss der Schaltkreis noch umgebaut werden, was mit einem konstanten Faktor zur Vergrößerung der Anzahl der Gatter gelingt. Die Anzahl der Produktionen für eine Variable A ist nur von der *DTM* M abhängig. Der Eingangsgrad des Gatters zur Produktion für ein Nonterminal ist also konstant und unabhängig von v.

Die Variablen mit Regeln $A \rightarrow \lambda$ werden zu Gattern mit dem Wert *true* und alle, die auf keiner linken Seite auftauchen bekommen den Wert *false*.

Der so konstruierte Schaltkreis $G(v)$ liefert den Wert 1 genau dann, wenn $v \in L$ ist.
Damit ist die Reduktion gezeigt.

weitere P-vollständige Probleme (ohne Beweis)

Definition:
Context-free Infinity

Eingabe:
eine kontextfrei Grammatik G.

Frage:
Ist $L(G)$ unendlich?

Definition:
Generation Problem

Eingabe:
Eine endliche Menge A mit einer Verknüpfung $\circ$ (nicht notwendig assoziativ!), eine Teilmenge $T \subseteq A$ und ein $x \in A$.

Frage:
Lässt sich x als Produkt von Elementen aus T darstellen?

noch ein P-vollständiges Problem (ohne Beweis)

Definition:
Linear Programming in $\mathbb{Q}$:

Ein Polynomzeitalgorithmus wurde 1979 von Kachian gefunden. Bessere von Karmakar 1984!

Eingabe:
Eine Matrix $V \in \mathbb{Z}^{n \times m}$, Vektoren $C \in \mathbb{Z}^n, D \in \mathbb{Z}^m$ und ein $B \in \mathbb{Z}$.

Frage:
Gibt es Vektor $X \in \mathbb{Q}^n$ mit $V^\top \cdot X \leq D$ und $C \cdot X \geq B$?

Es muss darauf hingewiesen werden, dass ein sehr ähnliches Problem NP-vollständig ist:

Definition:
Linear Programming in $\mathbb{Z}$:

Gute u. schwierige Literatur: Schrijver: Theory of Linear and Integer Programming, Wiley (1994)

Eingabe:
Eine Matrix $V \in \mathbb{Z}^{n \times m}$, ein Vektor $B \in \mathbb{Z}^m$.

Frage:
Gibt es Vektor $X \in \mathbb{Z}^n$ mit $V \cdot X \geq B$?

PRIMES

Lange Zeit war das Problem, zu testen, ob eine beliebige, binär dargestellte Zahl eine Primzahl sei nur als Element von $NP \cap co{-}NP$ bekannt.
Jedoch wurde 2002 von Manindra Agrawal, Nitin Saxena und Neeraj Kayal gezeigt, das sogar $PRIMES \in P$ gilt!

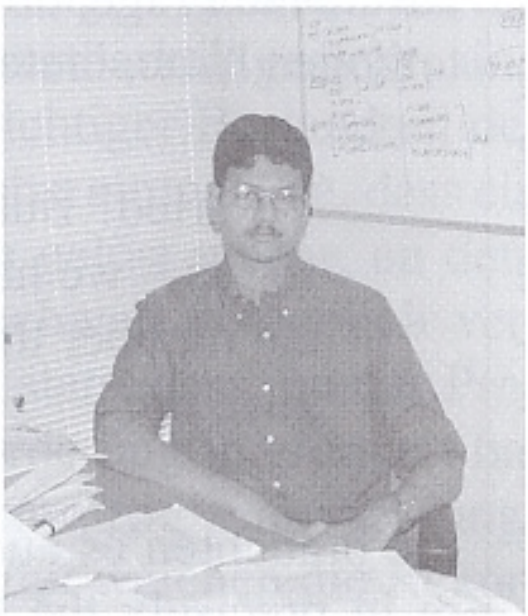
4. August 2002

Manindra Agrawal

Professor at the Indian Institute of Technology

Gruppenbild & Manindra Agrawal

4. August 2002



Nitin Saxena

Frühjahr 2003: Bachelor in Computer Science at the Indian Institute of Technology

"New Method said to solve Key Problem in Math"

Dies war der Titel eines Beitrags in der New York Times am 8. August 2002.

Am Sonntag, 4. August 2002 verschickten die drei Autoren einen 9-seitigen preprint.

Am Montag, 5.8., befand Carl Pomerance das Ergebnis für korrekt, informierte die New York Times und organisierte spontan ein Seminar.

"This algorithm is beautiful!" (Carl Pomerance).

"It's the best result I've heard in over 10 years!" (Shafi Goldwasser).

Am Dienstag wurde der preprint vom 6.8.2002 im Internet frei zugänglich gemacht:

http://www.cse.iitk.ac.in/news/\\primality.html

Eine Auszeichnung

Innerhalb der ersten 10 Tage registrierte Agrawal auf der extra eingerichteten web-Seite über zwei Millionen Zugriffe:
"I never imagined that our result will be of much interest to traditional mathematicians!"

Nun jedoch:
October 2002: Manindra Agrawal gets the Clay Research Award!

Prof. Manindra Agarwal has been awarded the Clay Research Award given by the Clay Mathematical Institute for his work on primality testing.

Previous recepients of this prestigious award include
Andrew Wiles (1999, *proved Fermat' s conjecture*),
Stanislav Smirnow (2001, *Stochastiker*)
Oded Schramm (2002, *Stochastiker*)
Laurent Lafforgue (*2000 Fields medalist*),
Edward Witten (2001 Clay research Award, *1990 Fields medalist*), and
Alain Connes (2000 Clay research Award, *1982 Fields medalist*).

Was gab es bisher? 1967: Miller & Rabin

Der **Schultest**, alle Zahlen k, $2 \leq k \leq \sqrt{n}$ auf $n/k \in \mathbb{Z}$ zu überprüfen, ist exponentiell in $\ell og(n)$.
Aus $n = 2^{\ell og(n)}$ folgt doch:

$$\sqrt{n} = n^{\frac{1}{2}} = (2^{\log n})^{\frac{1}{2}} = (\sqrt{2})^{\log n}$$

Der kleine **Fermatsche Satz** sagt, dass für eine Primzahl n und dazu teilerfremdes a stets gilt:

$$a^n \equiv a \mod n$$

Die Umkehrung gilt jedoch nicht: Es gibt faktorisierbares n, und die Kongruenz gilt für alle $a \in \mathbb{N}$!

Der probabilistische Primzahltest von **Miller und Rabin** (1976) stellt nach k Läufen entweder fest, dass die Eingabe mit Sicherheit zusammengesetzt, oder mit einer Irrtun$O(k \log^2 n)$nlichkeit von weniger als 4^{-k} eine Primzahl ist. Der Algorithmus arbeitet in der Zeit .
Das heißt in der Komplexitätstheorie kurz:

$$\text{PRIMES} \in co - \mathcal{RP}$$

Dieser Primzahltest arbeitet sehr schnell und wird in der Kryptographie zur Erzeugung von "industrial-grade primes" (Henri Cohen) verwendet.

Miller zeigte unter der Annahme, dass die erweiterte Riemannsche Hypothese gilt, dass dann:

$$\text{PRIMES} \in \mathcal{P}$$

Die erweiterte Riemannsche Hypothese ist jedoch seit über 100 Jahren unbewiesen!

1977: Solovay & Strassen

Solovay und **Strassen** zeigten mit probabilistischem Verfahren, dass die Wahrscheinlichkeit eines Fehlers beliebig klein gemacht werden kann, und daher galt:

und $\text{PRIMES} \in \mathcal{BPP}$ $\text{PRIMES} \in co - \mathcal{RP}$

Das bedeutet: bei einer faktorisierbaren Zahl sagt der Algorithmus mit beliebig kleiner Wahrscheinlichkeit sie sei Primzahl, bei Primzahlen irrt er aber niemals.

$L \in RP$, falls eine polynomial zeitbeschränkte NTM existiert mit:

- $x \in L$ impliziert mindestens die Hälfte der Rechnungen sind Erfolgsrechnungen
- $x \notin L$ impliziert keine Rechnungen ist eine Erfolgsrechnung!

$L \in co{-}RP$ genau dann, wenn $\Sigma^* \backslash L \in RP$.

$L \in BPP$, falls eine polynomial zeitbeschränkte NTM existiert mit:

- $x \in L$ impliziert mindestens $3/4$ der Rechnungen sind Erfolgsrechnungen
- $x \notin L$ impliziert mindestens $3/4$ der Rechnungen sind keine Erfolgsrechnungen!

$L \in ZPP$, falls $L \in RP \cap co{-}RP$

1983: Adleman, Pomerance und Rumley

Adleman, Pomerance und Rumley fanden 1983 ein deterministisches Verfahren - sog. *cyclotomic method* -, welches den kleinen Fermatschen Satz so auf ganze Zahlen des Kreisteilungskörpers verallgemeinerte, dass damit alle Primzahlen charakterisiert werden konnten. Die Laufzeit dieses Algorithmus liegt bei:

$$(\log n)^{O(\log\log\log n)}$$

Wobei $n \in \mathbb{N}$ wieder die zu testende Zahl und nicht deren Binärdarstellung ist. Für Zahlen von mehreren tausend Dezimalstellen ist dies bisher in der Praxis noch ungeschlagen, obwohl dies immer noch kein polynomielles Verfahren ist!

Dieses Verfahren macht keine Fehler und ist “beinahe” polynomial. Leider sehr schwer zu programmieren, und ***daher*** fehlerträchtig!

1986: Goldwasser & Killian + 1992: Adleman & Huang

Goldwasser und Killian nutzten 1986 elliptische Kurven für ein relativ ineffizientes probabilistisches Verfahren.

Atkin fand einen ähnlichen, aber viel effizienteren Algorithmus, Elliptic Curve Primality Proving (ECPP), der Zahlen mit mehr als 1000 Ziffern testete.

Adleman und Huang modifizierten 1992 das Verfahren von Goldwasser und Killian um mit einem probabilistischen Verfahren folgendes zu erhalten:

Nach k Durchläufen hat man eine definitiv korrekte Antwort oder mit einer Wahrscheinlichkeit von 2^{-k} gar keine. In der Sprache der Komplexitätstheorie schreibt man dies als:

. $\text{PRIMES} \in \mathcal{RP}$ $\text{PRIMES} \in co-\mathcal{RP}$

erhielt man also:

$$\text{PRIMES} \in \mathcal{ZPP}$$

Dies bedeutet, dass der Algorithmus immer korrekt arbeitet, aber ***bisweilen sehr lange*** braucht!

Manindra Agrawal

1991 promoviert am Dept. of Computer Science & Electrical Engimeering des Institute of Technology in Kanpur (IITK), und war 1995/96 als Humboldt Stipendiat an der Univ. Ulm.

Ein neuer probabilistischer Primzahltest findet sich in seiner Arbeit
"Primality and Identity Testing via Chinese Remaindering",
in der folgende Verallgemeinerung des Fermatschen Satzes auf Polynome vorkam:

Sind $a,n\in\mathbb{N}$ teilerfremd, so ist n genau dann ***prim***, wenn in $\mathbb{Z}[x]$ gilt:

$$(x-a)^n \equiv x^n - a \mod (x^r-1, n)$$

Hierbei meint $mod(x^r-1,n)$ gleiche Reste der Polynomdivision durch x^r-1 und gleiche Reste der Division der Koeffizienten durch n.

Leider hilft das nicht weit, denn die Berechnung von $(x-a)^n$ ist sehr aufwändig!
Sein eigenes Verfahren (s.o.) war zudem weitaus schlechter als das von Miller und Rabin.

Mit Studierenden, die alle ein strenges Zulassungsverfahren durchlaufen hatten, um am IITK studieren zu dürfen, arbeitete Manindra Agrawal weiter am Primalitätstest.

Die erste Bachelor Arbeit

Das Zulassungsverfahren zu einem der sieben Standorte der IITK läuft so ab:
Etwa 150.000 Bewerber schreiben eine 3-stündige Klausur in Mathematik, Physik und Chemie.
Von diesen bestehen ungefähr 15.000 und werden zu einer zweiten Prüfung eingeladen, in der sie in jedem Fach eine 2-stündige Klausur schreiben müssen. Danach werden die insgesamt 2.900 Studienplätze vergeben!

Den Bachelor Studenten Rajat Bhattacharjee und Prashant Pandey, kam die Idee, statt der riesigen Potenzen $(x-a)^n$ nur die Reste nach Division durch x^r-1 zu betrachten.
Für r logarihmisch in n lassen sich die kleineren Reste in polynomieller Laufzeit berechnen!

Ist n Primzahl, so gilt ja für alle r und zu n teilerfremde a:

$$(x-a)^n \equiv x^n - a \mod (x^r-1, n)$$

Für welche r und a folgt aber, dass n Primzahl ist?

In ihrer gemeinsamen Bachelor-Arbeit untersuchten die Studenten viele $r\leq 100$ und $n\leq 10^{10}$ bei festem $a=1$, und vermuteten nach ihren Ergebnissen, dass für teilerfremde r und n aus

$$(x-1)^n \equiv x^n - 1 \mod (x^r-1, n)$$

entweder folgt, dass n Primzahl ist oder dass $n^2\equiv 1 \ mod \ r$ gilt.

Da letzteres für eine der ersten $\ell og(n)$ Primzahlen nicht gilt, bekäme man einen Primzahltest in polynomialer Laufzeit $O(\ell og^{3+\varepsilon} n) = O([\ell og(n)]^{3+\varepsilon})$.

Die zweite Bachelor-Arbeit

Neeraj Kayal und Nitin Saxena waren 1997 im indischen Team der internationalen Mathematik Olympiade. Sie wandelten in ihrer gemeinsamen Bachelor-Arbeit den bisherigen Test dahingehend ab, dass sie anstelle von $a=1$ den Wert von r fixierten und dafür a variierten.
Am Morgen des 10. Juli 2002 gelang der Durchbruch!
Das Ergebnis wurde von Dan Bernstein eine knappe Woche nach der ersten Publikation geglättet und lautet nun:

Für $n \in \mathbb{N}$ seien q, r prim und $s \leq n$ so gewählt, dass

$$q|(r-1), n^{(r-1)/q} \not\equiv 0,1 \mod r \textbf{ und } \binom{q+s-1}{s} \geq n^{2\lfloor\sqrt{r}\rfloor}$$

Unter diesen Bedingungen ist n eine Primzahlpotenz, wenn für alle a, $1 \leq a < s$, folgendes gilt:

(i) a und n sind teilerfremd

(ii) im Polynomring $\mathbb{Z}[x]$ gilt: $(x-a)^n \equiv x^n - a \mod (x^r - 1, n)$

Der AKS-Algorithmus

Der Satz führt zu folgendem AKS-Algorithmus:

1. Entscheide, ob n echte Potenz einer natürlichen Zahl ist. Wenn ja, weiter bei 5.
2. Wähle (q, r, s) gemäß den Voraussetzungen des Satzes.
3. Für $a = 1, \ldots, s-1$ tue jeweils folgendes:
   (i) Ist a Teiler von n, gehe zu 5.
   (ii) Ist $(x-a)^n \not\equiv x^n - a \textbf{ mod} (x^r - 1, n)$, gehe zu 5.
4. n ist Primzahl
5. n ist zusammengesetzt.

Wieso ist dies nun polynomiell?
1. lässt sich mit Variation der Newton-Iteration in polynomialer Laufzeit erledigen
3. Dieser Schritt kann bei schneller FFT-basierter Arithmetik (Fast-Fourier-Transformation) in $\tilde{O}(s \cdot r \cdot \log^2 n)$ Zeit durchgeführt werden. (Die Tilde steht für weitere logarithmische Faktoren)

Ziel von 2. ist das Wachsen von s und r polynomiell in $\ell og\ n$.
Für $s = \theta q$ mit festem Faktor θ liefert die Stirlingsche Formel: $\log \binom{q+s-1}{s} \equiv c_\theta^{-1} q$.

Die Bedingungen des Satzes erfordern demnach asymptotisch $q \gtrsim 2c_\theta \lfloor\sqrt{r}\rfloor \log n$.

Details zur Analyse

Für große n kann die asymtotische Abschätzung nur funktionieren, wenn es unendlich viele Primzahlen r gibt, so dass $r-1$ einen Faktor$q \geq r^{\frac{1}{2}+\delta}$ besitzt.
Das bestmögliche Verhältnis q/r erhält man mit den ungeraden Primzahlen q, für die auch $2q+1$ prim ist. Sophie Germain hatte 1823 gezeigt, dass der erste Fall der Fermatschen Vermutung für diese Primzahlen gilt:

$x^q + y^q = z^q$ besitzt keine ganzzahlige Lösung mit $q \nmid xyz$.

Fast 10 Jahre vor dem Beweis der Fermatschen Vermutung durch Andrew Wiles zeigten 1985 Adleman, Fouvry und Heath-Brown, dass der erste Fall der Fermatschen Vermutung für unendlich viele Primzahlen richtig ist! Dazu wird gefordert, dass die Abschätzung

$$\left|\{r \leq x \mid q, r \text{ prim }, q \text{ teilt } r-1, q \geq x^{\frac{1}{2}+\delta}\}\right| \geq c_\delta \frac{x}{\ln x}$$

einen zulässigen Exponenten $\delta > 1/6$ besitzt. Étienne Fouvry zeigte 1985: $\delta = 0,1687 > \frac{1}{6}$ machte aber einen Fehler, der erst 1996 zu $\delta = 0,1683 > \frac{1}{6}$ korrigiert worden war!
Mit $r = O(log^{\frac{1}{3}} n)$ und $q, s = O(log^{1+\frac{1}{2\delta}} n)$ bekommt der AKS-Algorithmus eine garantierte Laufzeit von $\tilde{O}(\log^{11,913} n)$, oder ohne die logarithmischen Faktoren von

$$O(\log^{12} n) = O(m^{12})$$

Wie praktisch ist das nun?

Zur Zeit ist der AKS-Algorithmus noch zu langsam:
Der Rabin & Miller Test kann auf handelsüblichem 2 GHz PC industrial-grade primes (unterer Qualitätsstufe) mit 512 Binärstellen in Bruchteilen von Sekunden erzeugen.
Mit dem ECPP-Verfahren kann man dann deren Primzahleigenschaft in wenigen Sekunden verifizieren.

Der AKS-Algorithmus braucht dafür bisher noch ein paar Tage!

Dank Dan Bernstein, Hendrik Lenstra, Felipe Voloch, Björn Poonen und Jeff Vaaler wurde seit der Urfassung schon ein Zeitgewinn mit Faktor $0{,}5 \cdot 10^{-6}$ erzielt (Stand 25.01.2003)

Aber wie beim ECPP Verfahren sind auch hier in Zukunft sicher noch einige Verbesserungen zu erwarten!

PSPACE-vollständige Probleme / Mengen

Definition:
Die Menge M der quantifizierten booleschen Formeln ist die kleinste Menge, die alle Variablen $x_1, x_2, \ldots$, die Konstanten $0, 1$ (für die Wahrheitswerte) und mit $E, F \in M$ auch die Formeln

$$(E), \neg E, (E \wedge F), (E \vee F), \exists x E, \forall x E,$$

für alle Variablen x enthält.

Die Menge M ist eine kontextfreie Sprache, und eine Belegung b ist eine Funktion $b : \{x_1, x_2, \ldots, \} \rightarrow \{ 0, 1\}$. Wir sagen, dass b die Formel F erfüllt, wenn $b(F) = 1$ ist. F heißt gültig, wenn F von jeder Belegung erfüllt wird. Geschlossene Formeln sind solche, in denen keine Variable frei vorkommt. Die Variable x_i kommt in der Formel F frei vor, wenn

1. $F = x_i$ für eine Variable x_i, oder
2. $F = (E)$, $F = \neg E$, $F = \exists x_j E$, oder $F = \forall x_j E$ mit $i \neq j$ und x_i kommt in E frei vor, oder
3. $F = (G \vee H)$ oder $F = (G \wedge H)$ und x_i kommt in G oder H frei vor.

QBF

Definition:
QBF bezeichne die Menge der geschlossenen, quantifizierten gültigen Formeln.

> **Satz:**
>
> Die Menge *QBF* ist vollständig für *PSPACE*.

Beweis:

a) Für eine geratene Belegung b ist die Auswertung $b(F)$ mit polynomiellem (genauer quadratischem) Platz möglich. (wie geht's genau?)
Wegen $NPSPACE = PSPACE$ reicht das dann aus.

b) Für die Härte der Sprache *QBF* erinnern wir uns an die Möglichkeit der erneuten Berechnung, wie Sie im Beweis von Savitch benutzt wurde, und werden die Erreichbarkeit im Konfigurationsgraphen einer beliebigen polynomiell platzbeschränkten *DTM* überprüfen.

QBF Beweis(-Idee, Fortsetzung)

Sei $L = L(M)$ für $p(n)$-platzbeschränkte 1-Band DTM ($p \in POLY$), dann hat der Konfigurationsgraph höchstens $2^{p(n)}$ viele Knoten, deren Binärkodierung höchstens $p(n)$ Symbole haben.

Mit dem Ansatz von Savitch ergibt sich:

1. $PFAD(x, y, i) \iff \exists z \in KONF : PFAD(x, z, \lceil \frac{i}{2} \rceil) \wedge PFAD(z, y, \lfloor \frac{i}{2} \rfloor)$, wobei das Prädikat $PFAD(x, y, i)$ erfüllt ist, wenn in der NTM M von Konfiguration x zur Konfiguration y ein Pfad mit Länge i existiert. (Die Fälle $i = 0$ mit $x = y$ und $i = 1$ sind einfach.)
2. Wir definieren für jedes i eine quantifizierte boolesche Formel $\psi_i(A_x, B_y)$ mit allen und weiteren freien Variablen, die eine erfüllende Belegung genau dann hat, wenn die anderen Variablenbelegungen in A_x und B_y Konfigurationen kodieren, für die $PFAD(x, y, i)$ wahr ist. Wenn $\psi_{p(n)}(A_{\text{start}}, B_{\text{end}})$ wahr ist, bedeutet das $PFAD(x_{\text{start}}, x_{\text{end}}, p(n))$ und also $w \in L$.

QBF Beweis(-Idee, Fortsetzung)

3. $\psi_0(A, B)$ bedeutet $\forall i : a_i = b_i$ oder die von B beschriebene Konfiguration folgt auf die von A beschriebene in einem Schritt.
4. Da der Versuch, $\psi_{i+1}(A, B)$ durch $\exists Z : \psi_i(A, Z) \wedge \psi_i(Z, B)$ zu ersetzen, leider zu exponentiellen Formeln führt, also keine log-space Reduktion wäre, werden wir die Formel ψ_i wiederverwenden, d.h. mehrfach nutzen! Dazu wird $\psi_{i+1}(A, B)$ ersetzt durch:
5. $\exists Z \forall X \forall Y : \Big[\big((X = A) \wedge (Y = Z)\big) \vee \big((X = Z) \wedge (Y = B)\big) \Longrightarrow \psi_i(X, Y)\Big]$
   Dadurch werden die Formeln wie Makros benutzbar.

Aus $X = A$ wird die Formel $\bigwedge_{1 \le i \le p(n)} x_i = a_i$

und aus $\forall X$ wird $\forall x_1 \forall x_2 \cdots \forall x_{p(n)}$

und entsprechend bei $\exists X$.

Die Anfangsformeln $\psi_0(A, B)$ werden wie beim Beweis des Satzes von Cook notiert.

Mehr Details z.B. bei Papadimitriou....

QBF in Pränex-Normalform mit Matrix in $3-KNF$

Definition:

a) F_{QSAT} sei die Menge aller quantifizierten booleschen Formeln in Pränex-Normalform der Art:

$$\exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n \ \Phi,$$

wobei $Q_n := \begin{cases} \forall, & \text{falls } n \text{ gerade,} \\ \exists, & \text{falls } n \text{ ungerade} \end{cases}$ und die sogenannte Matrix Φ eine boolesche Formel über $\{0, 1, x_1, x_2, \ldots, x_n\}$ in 3-KNF ist.

b) Die Menge $QSAT$ ist definiert als
$QSAT := \{w \in F_{QSAT} \mid w \text{ besitzt erfüllende Belegung}\}$.

Satz:

Die Menge $QSAT$ ist vollständig für $PSPACE$.

$PSPACE$-vollständige Sprachen

Definition:

Das Universalitätsproblem für *NFA*'s ($L(A) \neq \Sigma^*$?):

Eingabe:
Ein nichtdeterministischer endlicher Automat (*NFA*) A

Frage:
Gilt $L(A) \neq \Sigma^*$?

Satz:
Das Universalitätsproblem für *NFA*'s ist vollständig für *PSPACE*!

weitere *PSPACE*-vollständige Sprachen

Definition:
Function Generation Problem

Eingabe:
Eine endliche Menge A, eine Menge F von Funktionen $g : A \to A$ und eine Funktion $f : A \to A$.

Frage:
Lässt sich f als Hintereinanderausführung von Funktionen aus F erzeugen?

Satz:
Das Function Generation Problem ist *PSPACE*-vollständig.

Das Spiel "Geographie"

Das Spiel Geographie beginnt damit, dass ein Spieler den Namen einer Stadt sagt, z.B. "Hamburg". Der jeweils nächste Spieler muss ebenfalls den Namen einer Stadt nennen, die mit dem Buchstaben beginnt, mit dem die zuletzt genannte aufhört, hier z.B. "Graz", und weiter "Zaragossa"- "Aachen" - "Nizza" - "Athen" - "Nürnberg". (jede Stadt darf nur einmal genannt werden!)

Verlierer ist derjenige, der keine Stadt mehr nennen kann.

Mathematisch verallgemeinert wird das zu einem beliebigen, endlichen Graphen $G{:=}(V,E)$, dessen Knoten mit beliebigen Zeichenketten über beliebigen Alphabeten beschriftet sind, und dessen Kanten die oben beschriebenen Relation: "beginnt-mit-dem-letzten-Buchstaben-von" beachten. Das Spiel besteht nun aus abwechselndem Markieren von Knoten, wobei jeweils ein Nachbar des zuletzt Markierten zu kennzeichnen ist.

Das Problem, ob es für den ersten Spieler eine Gewinnstrategie gibt, ist *PSPACE*-vollständig!

noch mehr *PSPACE*-vollständige Probleme:

Zu gegebener kontextsensitiven (oder monotonen) Grammatik G und einem Wort w ist die Frage

$$\text{“}w \in L(G)?\text{”}$$

PSPACE-vollständig!

Gegeben eine DTM A und ein Wort w, dann sind folgende Fragen *PSPACE*-vollständig:

a) akzeptiert A das Wort w, ohne die ersten $|w|+1$ Bandfelder zu verlassen?

b) hat A eine nichthaltende Rechnung mit Benutzung von höchstens $|\langle A \rangle|$ Symbolen?

Ein oft erwähntes Problem:

Definition:
Erreichbarkeit bei 1-sicheren Petrinetzen

Eingabe:
Ein 1-beschränktes Petrinetz $N = (P, T, F)$ und zwei Markierungen $m_0, m_1 \in \{0,1\}^{|P|}$.

Frage:
Gibt es eine Schalfolge in N, mit der m_1 von m_0 aus erreicht wird?

Satz:
Das Erreichbarkeitsproblem für 1-beschränkte Petrinetze ist vollständig für *PSPACE*!