

Untere Schranken für das Analyse-
problem contextfreier Sprachen

Günter HOTZ

XI/1975

Abstract :

Let A, B, C (n, n) -matrices and x the usual matrix product. We encode A, B, C in a contextfree language such that the decision problem $AxB=C$ transforms into the word problem for this context-free language. We show that the problem $AxB=C$ has a boolean expression complexity $\geq O(n^3)$. This means, that our contextfree language has the word problem complexity $\geq O(\frac{n}{\log n})^{3/2}$ if we use again the boolean expression complexity as the measure.

1. Die syntaktischen Monoide formaler Sprachen.

Die Bedeutung des syntaktischen Monoides für die Erforschung der formalen Sprachen wurde schon früh bemerkt und ganz besonders durch M.P. Schützenberger (hierzu sehe man etwa [5]) im Zusammenhang mit Codierungsfragen herausgearbeitet. Parallel hierzu entwickelte M.P. Schützenberger eine Theorie formaler Potenzreihen in nicht kommutativen, freien Variablen über Ringen. Wir vereinigen hier beide Ansätze in der Definition der syntaktischen Algebra, die wir über algebraischen und booleschen Ringen aufbauen. Hierdurch wird in natürlicher Weise ein enger Zusammenhang zwischen klassischen mathematischen Theorien und der Theorie der formalen Sprachen hergestellt, der das Wortproblem für contextfreie Sprachen nicht mehr als isoliertes Problem erscheinen lässt, sondern seine Komplexität in enge Beziehung bringt zu der Komplexität der Multiplikation von Zahlen, Polynomen und Matrizen. Wir gehen in dieser Arbeit nur auf den Zusammenhang mit der Matrixmultiplikation ein.

Sei im folgenden R ein algebraischer oder ein boolescher Ring und M ein Monoid. R habe stets ein 1-Element. Wir bilden die Algebra $R[M]$, deren Elemente p formale Summen

$$p = \sum_{m \in M} \alpha_m \cdot m$$

sind, worin

$$\alpha : M \rightarrow R$$

eine Abbildung und $\alpha_m = \alpha(m)$ ist. Weiter sei $\alpha_m \neq 0$ nur für endlich viele $m \in M$.

Wir definieren in naheliegender Weise

$$\sum \alpha_m \cdot m + \sum \alpha'_m \cdot m = \sum (\alpha_m + \alpha'_m) \cdot m$$

und

$$(\sum \alpha_m \cdot m) \cdot (\sum \alpha'_m \cdot m) = \sum_{m=u+v=m} (\alpha_u \cdot \alpha_v) \cdot m.$$

Die Addition ist offensichtlich kommutativ und assoziativ, die Multiplikation assoziativ. Kommutativ ist die Multiplikation nur dann, wenn M kommutativ ist.

Wir identifizieren $u \in M$ mit

$$\sum \alpha_m \cdot m, \quad \alpha(m) = \begin{cases} 1 & \text{für } m=u \\ 0 & \text{sonst.} \end{cases}$$

Weiter identifizieren wir

$$r \in R \text{ mit } \sum \alpha_m \cdot m, \quad \alpha(m) = \begin{cases} r & \text{für } m=\varepsilon \\ 0 & \text{sonst.} \end{cases}$$

Hier ist ε die Einheit in M.

Durch diese Identifikationen fallen die Einheiten von R und M zusammen und wir haben

$$R \subset R \langle M \rangle, \quad M \subset R \langle M \rangle.$$

Besitzt M ein Nullelement, dann gilt für $u \in R \langle M \rangle$

$$u_0 \cdot u = u_0 \cdot \sum \alpha_m \cdot m = \sum \alpha_m \cdot (u_0 \cdot m)$$

$$= u_0 \cdot \sum \alpha_m$$

Also spielt die Klasse $\{\alpha \cdot u_0 \mid \alpha \in R\}$ die Rolle eines Nullelementes in $R\langle M \rangle$.

Wir identifizieren dann

$$o \in R \text{ mit } \alpha \cdot u_0 \quad \text{für } \alpha \in R$$

und haben damit ein Nullelement in $R\langle M \rangle$.

Definition :

Ist $L \subseteq T^*$ eine formale Sprache und $S(L)$ das zu L gehörige syntaktische Monoid, dann heißt $R\langle S(L) \rangle$ die syntaktische Algebra von L über R .

Für das Ziel dieser Arbeit genügt es den Fall zu betrachten, daß R die boolesche Algebra

$$B = B_1 = \{0,1\}$$

ist.

Wir stellen einige einfache Eigenschaften von $B\langle M \rangle$ zusammen.

Es gelten die folgenden Lemmata :

$$(L1) \quad u, v \in B\langle M \rangle,$$

$$u = \sum \mu_m m, \quad v = \sum \nu_m \cdot m.$$

a) Aus $u \cdot v = 0$ und M nullteilerfrei folgt $u = 0$ oder $v = 0$.

b) Aus $u \cdot v = 0$ folgt für $m \in \mu^{-1}(1), n \in \nu^{-1}(1)$

$$m \cdot n = 0$$

Wir betrachten nun inverse Paare.

$u, v \in B< M >$ heißen invers \Leftrightarrow

$$u \circ v = 1.$$

$u, v \in B< M >$ heißen quasiinvers \Leftrightarrow

$$u \circ v = 1 + w.$$

$1 + w$ heißt Quasieinheit in $B< M >$.

(L2) a) Die Menge \mathcal{E} der Quasieinheiten bildet eine Unteralgebra von $B< M >$.

b) $1+w$ und v sind genau dann quasiinvers zueinander, wenn dies für w und v gilt.

Wir wenden uns nun einer speziellen syntaktischen Algebra zu.

Die Dyckalgebra.

Sei D_k das syntaktische Monoid der Dycksprache mit k Klammertypen

$$X_k = \{x_1, x_2, \dots, x_k, x_1^{-1}, x_2^{-1}, \dots, x_k^{-1}\}.$$

Das Rechnen in D_k lässt sich leicht durch definierende Relationen vollständig beschreiben.

Sei $R_k = \{x_i x_i^{-1} = 1, x_i x_j^{-1} = 0, 0 \cdot x_i^\varepsilon = 0, x_i^\varepsilon \cdot 0 | i \neq j \text{ und } i, j = 1, 2, \dots, k; \varepsilon = \pm 1\}$.

Es ist $D_k = (X_k \cup \{0\})^*/R_k$

worin / die Bildung des Faktormonoides nach den Relationen R_k bedeutet.

Wir setzen nun

$$\mathcal{G}_k = B < D_k > \quad \text{für } k=1,2,\dots$$

und bezeichnen \mathcal{G}_k als die k-te Dyckalgebra.

Die Bedeutung der Dyckalgebra für die kontextfreien Sprachen ergibt sich aus der engen Verwandtschaft von \mathcal{G}_k mit der von S. Greibach in [1,2] angegebenen schwersten kontextfreien Sprache G. Diese Verwandtschaft drückt sich in dem folgenden Satz aus :

Satz 1 :

Seien $p_1, \dots, p_n \in \mathcal{G}_2$.

Der Ausdruck

$$p_1 \circ p_2 \circ \dots \circ p_n$$

liegt genau dann in G, wenn $p_1 \circ p_2 \circ \dots \circ p_n$ eine Quasieinheit ist.

Beweis :

Wir geben eine informelle Definition der Sprache G.

Seien n große Kästen || gegeben, in die kleine Kästen linear hineingestellt seien, wie dies die Figur

$$|, , | , , , | , |$$

veranschaulicht. Sei w ein Wort aus D_2 und w in $w=w_1, \dots, w_n$ zerlegt.

Jedes w_i wird in eins der kleinen Kästchen gelegt und zwar so, daß w_i in den i-ten großen Kasten zu liegen kommt. Die restlichen kleinen Kästchen werden mit Wörtern aus X_k^* aufgefüllt. Das ganze

Gebilde Kästen mit Füllung ist ein Wort der Greibachsprache und jedes Wort dieser Sprache ist auf diese Weise aufgebaut.

Ersetzt man nun die Komma durch "+" und die inneren Balken | durch *, dann erhalten wir ein Produkt in \mathcal{G}_2 .

Ein Wort ist genau dann in der Greibachsprache, wenn sich beim Ausmultiplizieren mindestens eins der Produkte auf 1 kürzt, d.h. wenn

$$p_1 \cdot p_2 \cdot \dots \cdot p_n = 1+w$$

ist. Dies war gerade zu zeigen.

Die Einbettung der Matrixmultiplikation in \mathcal{G}_2 .

Seien A, B, C quadratische n-reihige Matrizen. Wir beschränken uns nur aus Gründen der Kürze auf diesen Fall und deshalb, weil in ihm alle wesentlichen Ideen zum Ausdruck kommen.

Sei $x_2 = \{x, y, x^{-1}, y^{-1}\}$.

Wir kodieren die natürlichen Zahlen $i \in \mathbb{N}_0$ für $0 \leq i < 2^r$ in $\{x, y\}^r$, so daß bei der Interpretation von x als 0 und y als 1 gerade die Darstellung von i im Dualsystem erhalten wird, wenn wir von führenden Nullen absehen.

Wir bezeichnen diese Kodierung mit ϕ .

Es ist also $\phi(i) \in x_2^*$ und $\phi(i)$ hat die Länge r.

ϕ ist auf dem definierten Bereich bijektiv.

Wir setzen nun φ fort zu einer Abbildung

$$\psi: B \times [0:2^r-1] \times [0:2^r-1] \rightarrow \mathbb{Z}_2,$$

indem wir definieren

$$\psi(a, i, k) = a \cdot \varphi(i) \varphi(k).$$

Wir verwenden weiter

$$\psi'(a, i, k) = a \cdot \varphi(i) \cdot (\varphi(k))^{-1}.$$

Nun gehen wir hin und beschreiben die Matrix $A = (a_{ik})_n$ mit $n = 2^r$ durch

$$\psi(A) = \sum_{i,k=0}^{n-1} \psi(a_{ik}, i, k) \in \mathbb{Z}_2.$$

Weiter verwenden wir

$$\psi'(A) = \sum_{i,k=0}^{n-1} \psi'(a_{ik}, i, k) \in \mathbb{Z}_2.$$

Es gilt das

Lemma 1 :

Sind A, B, C quadratische n -reihige Matrizen über B , dann gilt

$A \times B = C$ genau dann, wenn

$$\psi(A) \cdot \psi'(B) = \psi(C)$$

ist.

Beweis :

Es ist

$$\psi(a, i, l) \cdot \psi'(b, m, k) = \begin{cases} (a \circ b) \varphi(i) \varphi(k) & \text{für } l=m \\ 0 & \text{für } l \neq m, \end{cases}$$

da die Worte $\varphi(l)$ und $\varphi(m)$ gleich lang sind.

Damit erhält man

$$\begin{aligned} \psi(A) \cdot \psi'(B) &= \sum_{i,k=0}^{n-1} \sum_{l=0}^{n-1} \psi(a_{il}, i, l) \cdot \psi'(b_{lk}, l, k) \\ &= \sum_{i,k=0}^{n-1} \psi\left(\sum_{l=0}^{n-1} a_{il} b_{lk}, i, k\right) \\ &= \psi(C) \end{aligned}$$

Da ψ bijektiv ist, folgt auch die Umkehrung.

Das Entscheidungsproblem $A \times B = C$

Ist \bar{C} das komponentenweise Komplement von C , dann gilt bekanntlich

Lemma 2 :

Die folgenden drei Aussagen sind äquivalent

$$A \times B = C,$$

$$(A \times B) \cdot \bar{C} + (\bar{A} \times B) \cdot C = 0,$$

$$(A \times B) \cdot \bar{C} = 0 \text{ und } (\bar{A} \times B) \cdot C = 0.$$

Wir wollen nun die Komplexität dieser Aussagen abschätzen. Als Maß verwenden wir die Komplexität der zugehörigen booleschen Funktionen. Sei $C(f)$ die Netzwerkskomplexität und $K(f)$ die Ausdruckskomplexität über $GF(2)$ der booleschen Funktion

$$f : B^n \rightarrow B.$$

In unserem Falle ist f eine Funktion von $3n^2$ Variablen. Wir setzen

1. $f(A, B, C) = 0 \Leftrightarrow A \times B = C,$
2. $g(A, B, C) = 0 \Leftrightarrow (A \times B) \cdot \bar{C} = 0,$
3. $h(A, B, C) = 0 \Leftrightarrow \overline{(A \times B)} \cdot C = 0.$

Es gilt aus Dualitätsgründen

$$C(g) = C(h)$$

und also auch

$$C(f) \leq C(g) + C(h) + 1 = 2 \cdot C(g) + 1.$$

Ist nun $M(n) = C(A \times B)$ die Netzwerkkomplexität der Matrizenmultiplikation, dann gilt offenbar

$$C(g) \leq C(A \times B) + n^2 = M(n) + n^2.$$

Wegen $M(n) \geq n^2$ haben wir also

$$C(g) \leq a \cdot M(n)$$

wo a eine von n nicht abhängige Konstante ist. Da uns $C(g)$ mindestens ebenso gute untere Abschätzungen für $M(n)$ liefert wie $C(f)$, befassen wir uns nun nur noch mit der Funktion g .

Leider haben wir noch keine brauchbare untere Abschätzung für $C(g)$, aber wir sind doch in der Lage $K(g)$ scharf abzuschätzen. Hierzu verwenden wir den Satz von Neziporuk [3], der das folgende besagt :

Sei f eine Schaltfunktion deren Variablenmenge sich in k paarweise fremde Blöcke von je m Variablen zerlegen lässt.

Also sei etwa

$$X = \{x_{11}, \dots, x_{1m}, x_{21}, \dots, x_{2m}, \dots, x_{k1}, \dots, x_{km}\}$$

die Menge der Variablen von f . Weiter sei

$$\varphi : X \rightarrow \{0,1\} \cup X$$

eine Einsetzung.

φ heißt eine zulässige Einsetzung, wenn es ein i gibt mit

$$\varphi(x_{lr}) = \begin{cases} x_{ir} & \text{für } l=i \\ \alpha \in \{0,1\} & \text{für } l \neq i. \end{cases}$$

Also eine zulässige Einsetzung ersetzt alle Variablen bis auf die genau eines Blockes durch Konstanten.

Die durch diese Einsetzung φ der Funktion f zugeordnete Schaltfunktion von n -Variablen bezeichnen wir mit f_φ .

Satz von Neciporuk :

Ist $F = \#\{f_\varphi \mid \varphi \text{ zulässige Einsetzung für } f\}$

($\# M$ ist die Anzahl der Elemente der Menge M),

dann gilt :

$$K(f) \geq a \cdot k \cdot \log F,$$

worin a eine nicht von X abhängige Konstante ist.

Wir verwenden diesen Satz, um $K(g)$ abzuschätzen. Wir partitionieren die Variablen von A, B, \bar{C} wie folgt:

Die i -te Klasse der Partition enthält

die i -te Spalte von A ,

die i -te Zeile von B ,

die i -te Zeile von C .

Wir haben also $k=n$. Aus Symmetriegründen können wir uns bei der Berechnung von F auf die zulässigen Einsetzungen ϕ beschränken, die nur die erste Klasse von Variablen wieder in Variablen überführen. Aus Gründen der Übersichtlichkeit spezialisieren wir noch weiter, indem wir alle Variablen von A und B , die nicht der ersten Klasse angehören durch 0 ersetzen. Damit haben wir die folgende Situation, wenn wir

$$x_i = a_{il}, y_i = b_{li}, z_i = c_{li}$$

setzen

$$\begin{pmatrix} x_1 \\ \vdots \\ 0 \\ \vdots \\ x_n \end{pmatrix} \times \begin{pmatrix} y_1 \dots y_n \\ \vdots \\ 0 \\ \vdots \\ y_n \end{pmatrix} \cdot \begin{pmatrix} z_1 \dots z_n \\ c_{21} \dots c_{2n} \\ \vdots \\ \vdots \\ c_{n1} \dots c_{nn} \end{pmatrix}$$

oder für unsere nun noch zulässigen ψ

$$f_\psi = \sum_{L=2, k=1}^n x_L y_k c_{ik} \oplus \sum_{L=1}^n x_L y_i z_i$$

worin \oplus die Addition in $GF(2)$ bezeichnet. Nun definieren verschiedene

Polynome über GF(2) verschiedene boolesche Funktionen, also für jede Wahl der n^2-n Variablen $\{c_{ik} | L=2, \dots, n; k=1, \dots, n\}$ erhalten wir eine andere Funktion f_φ . Also gilt

$$F \geq 2^{n^2-n}.$$

Setzen wir in die Formel von Neciporuk ein, dann erhalten wir

$$K(g) \geq a \cdot (n^3 - n^2).$$

Nun kann man g aber auch mit n^3 Bauelementen als Ausdruck realisieren. Also gilt der

Satz 2 :

$$K(g) = o(n^3),$$

d.h. die Ausdruckskomplexität von $(AxB) \cdot C$ über GF(2) ist $o(n^3)$.

Dieses Resultat macht eine genaue Untersuchung von $(AxB) \cdot C$ interessant, denn verhielte sich g wie fast alle booleschen Funktionen, dann würde gelten

$$C(g) = o(n^2 \log n).$$

Dies würde eine nicht triviale untere Schranke für die Matrixmultiplikation ergeben. Ebenso mag natürlich der Unterschied zwischen $C(g)$ und $K(g)$ den Faktor $\frac{n^2}{\log n}$ erreichen, wie in dem Beispiel von W. Paul [4], was uns keine brauchbare Schranke liefern würde.

Matrixmultiplikation und kontextfreie Sprachen

Wir betten nun das Produkt $(A \cdot B) \cdot C$ mittels unseren Abbildungen ψ, ψ' und einer weiteren Abbildung in \mathcal{G}_2 ein.

Sei

$$\tilde{\psi}(a, i, k) = a \cdot (\varphi(i) \circ \varphi(k))^{-1}$$

und

$$\tilde{\psi}(A) = \sum_{i,k=0}^{n-1} \tilde{\psi}(a_{ik}, i, k).$$

Nun gilt das

Lemma 3 :

Für quadratische n -reihige Matrizen A und B gilt

$$\sum_{i,k=0}^{n-1} a_{ik} b_{ik} = \psi(A) \cdot \tilde{\psi}(B).$$

Beweis :

Es ist

$$\psi(a_{ik}, i, k) \cdot \tilde{\psi}(b_{rs}, r, s) = \begin{cases} a_{ik} b_{ik} & \text{für } i=r, k=s \\ 0 & \text{für } (i, k) \neq (r, s) \end{cases}$$

Hieraus folgt die Aussage des Lemmas.

Nun folgt aus Lemma 1 und Lemma 3

$$g(A, B, C) = \psi(A) \cdot \psi'(B) \cdot \tilde{\psi}(C).$$

Hieraus folgt weiter

$$g(a, B, C) = 0 \Leftrightarrow \psi(A) \cdot \psi'(B) \cdot \tilde{\psi}(C)$$

ist nicht Quasieinheit.

Also gilt

Lemma 4 :

$g(A, B, C) = 0 \Leftrightarrow$ Der Ausdruck $\psi(A) \cdot \psi'(B) \cdot \tilde{\psi}(C)$ liegt nicht
in der kontextfreien Sprache G.

Aus Satz 2 und Lemma 4 erhalten wir eine untere Schranke für die
Ausdruckskomplexität von G :

Satz 3 :

Ist $w \in G$ und Länge $(w) = m$, dann ist

$$K(m) \geq \left(\frac{m}{\log m}\right)^{3/2}.$$

Beweis :

Ist $w = \text{Ausdruck } \psi(A) \cdot \psi'(B) \cdot \tilde{\psi}(C)$,

dann gilt

$$\text{Länge } (w) = a \cdot n^2 \cdot \log n = m$$

worin a eine Konstante ≥ 1 ist.

Nun gilt nach Satz 2

$$K(w) = a' n^3.$$

Setzen wir

$$n = \sqrt{\frac{m}{\log m}},$$

dann erhalten wir

$$K(w) \geq a'' \left(\frac{m}{\log m}\right)^{3/2},$$

womit unser Satz bewiesen ist.

Es stellt sich nun das

Problem :

Nicht jedes Wort aus G läßt sich als Bild von $(A \times B) \cdot C^{\neq 0}$ erhalten.

Hat die Teilsprache von G , die als Bild auftritt, die gleiche Analysekomplexität, wie G selbst? In diesem Fall liefert die Komplexität von G untere Schranken für die Matrixmultiplikation.

Schlußbemerkung :

Da Valiant [6] gezeigt hat, daß man die Analyse kontextfreier Sprachen auf die Matrixmultiplikation zurückführen kann, so daß die Komplexität der Matrixmultiplikation auch diese Schranken für die Analyse kontextfreier Sprachen liefert, erscheint das Wortproblem für kontextfreie Sprachen nun nicht mehr als isoliertes Problem, sondern seine Verbindung mit grundlegenden mathematischen Algorithmen ist offensichtlich geworden. Wir werden an anderer Stelle zeigen, daß eine ebenso enge Beziehung zwischen G_1 und der Polynommultiplikation besteht.

Literatur :

- [1] Greibach, S.A.: Remarks on Context-Free Languages and Polynomial Time Recognition. Reprint.
- [2] Greibach, S.A.: Jump Pda's, Deterministic Context-Free Languages, Principal AFDLs and Polynomial Time Recognition. Proceedings of Fifth Ann. ACM Symposium on Theory of Computing, Austin / Texas, May 1973.
- [3] Neciporuk, E.I.: A Boolean Function. Soviet.Math.Probl.7, 1966, pp. 999-1000.
- [4] Paul, W.J.: A $2.5 N$ Lower Bound on the Combinational Complexity of Boolean Functions. Cornell TR 74-222, 1974
- [5] Perrot, J.-F.: Monoides Syntactiques des Langages Algébriques. TR Nr. I P, 74-22, Université de Paris, Inst. de Programmation
- [6] Valiant, L.G.: General Context-Free Recognition in less than cubic Time. TR Carnegie-Mellon University, January 1974