# Effiziente Speicherstrukturen für Turingmaschinen mit *einem* Kopf

Martin Hühne

Lehrstuhl Informatik II
Universität Dortmund
44221 Dortmund

`huehne@ls2.informatik.uni-dortmund.de`

Die Effizienz von Turingmaschinen hängt sehr stark von der Anzahl und der Struktur ihrer Arbeitsspeicher ab. Wir untersuchen den Einfluß der Speicherstruktur auf die Effizienz von deterministischen Turingmaschinen mit genau einem Schreib/Lesekopf und geben obere Zeitschranken für die Simulation von Mehrband Turingmaschinen auf solchen Ein-Kopf Turingmaschinen an.

Die Struktur des Arbeitsspeichers ist fest und wird durch einen Graphen beschrieben [Re90]. Die abzählbar unendlich vielen Knoten des Graphen repräsentieren die Speicherzellen. Der Graph hat gefärbte, gerichtete Kanten, die die möglichen Bewegungen des Kopfes repräsentieren. Die Anzahl der ausgehenden Kanten pro Knoten ist beschränkt.

Bekannte und vielfach untersuchte Speicherstrukturen sind das Band, das Gitter, und der Baum. Ein-Kopf Turingmaschinen mit der Speicherstruktur "Band" sind sehr ineffizient, da für die Simulation von $t$ Schritten einer Mehrband Turingmaschine Zeit $O(t^2)$ benötigt wird [HS65]. Auf Ein-Kopf Turingmaschinen mit der Speicherstruktur "$d$-dimensionales Gitter", $d \geq 2$, kann die Simulation in Zeit $O(t^{1+1/d} / \log t)$ erfolgen [DH94]. Ein-Kopf Turingmaschinen mit der Speicherstruktur "binärer Baum" können Mehrband Turingmaschinen sogar in Zeit $O(t \log t)$ simulieren [PR81]. Diese Zeitschranken sind scharf. Schnellere deterministische Simulationen von Mehrband Turingmaschinen sind nur auf Modellen bekannt, die mehr als einen Kopf besitzen.

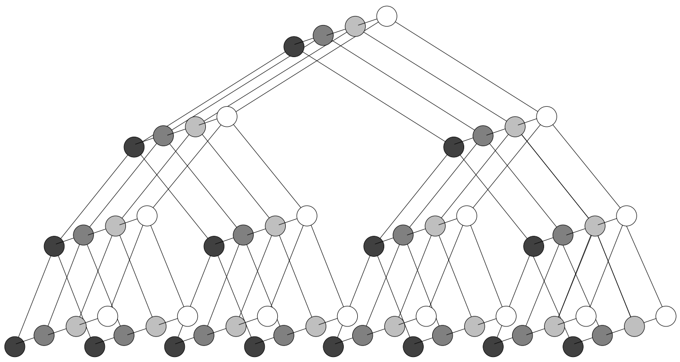
Gibt es Speicherstrukturen für Ein-Kopf Turingmaschinen, sodaß Mehrband Turingmaschinen schneller als in Zeit $t \log t$ simuliert werden können?

Man kann leicht eine Speicherstruktur definieren, in deren Kanten alle möglichen Berechnungen von Turingmaschinen kodiert sind. Sei $\mathcal{C} = \{L_1, L_2, L_3, \ldots\}$ eine abzählbare Menge von Sprachen über dem Alphabet $\{0, 1\}$. Betrachte die baumartige Speicherstruktur mit Knotenmenge $\{0, 1\}^*$, in der neben den Baumkanten für jedes Wort $x_1 \ldots x_j \in L_i$ eine Kante von der Speicherzelle $1^i\, 0\, x_1 \ldots x_j$ zur Wurzel des Baumes vorhanden ist. Eine Ein-Kopf Turingmaschine mit dieser Speicherstruktur kann auf einfache Weise alle Sprachen in $\mathcal{C}$ entscheiden — und zwar selbst dann, wenn einzelne oder gar alle Sprachen in $\mathcal{C}$ nicht rekursiv sind.

Daher ist es offensichtlich notwendig, die Menge der betrachteten Speicherstrukturen geeignet einzuschränken. Reischuk nennt eine Speicherstruktur *konstruierbar*, falls jede Speicherzelle eine kurze, eindeutige Adresse hat und falls es eine linear zeitbeschränkte Zwei-Band Turingmaschine gibt, die bei Eingabe der Adresse einer Speicherzelle die Adressen aller Nachbarzellen sowie die Färbung der zugehörigen Kanten berechnet [Re90].

Im Vortrag werden verschiedene konstruierbare Speicherstrukturen vorgestellt, auf denen Mehrband Turingmaschinen schneller als auf dem Baum simuliert werden können.

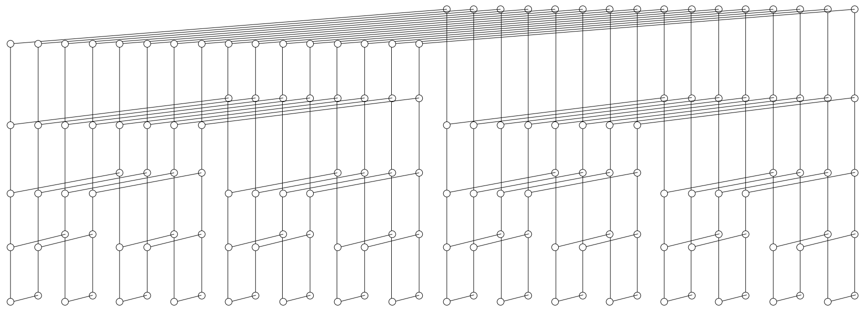
Die erste solche Speicherstruktur ist die "Hecke". Die Hecke läßt sich als das kartesische Produkt von Baum und Band beschreiben. Auf einer Ein-Kopf Turingmaschine mit Speicherstruktur "Hecke" können $t$ Schritte von Mehrband Turingmaschinen in Zeit $O(t\ \log t \,/\, \log\log t)$ simuliert werden. Grundlage für die Simulation ist ein schneller Kopieralgorithmus. Üblicherweise benötigt eine Ein-Kopf Turingmaschine Zeit $\ell{\cdot}\Delta$, um einen String der Länge $\ell$ an eine andere Stelle im Speicher zu kopieren, die $\Delta$ Speicherzellen entfernt ist. Wir benutzen eine zuerst auf dem Gitter entwickelte Technik [Ti87, DH94], mit der diese Kopieraufgabe mit einem Kopf in Zeit $O(\ell \cdot \Delta \,/\, \log \Delta)$ gelöst werden kann. Allerdings ist vor dem Kopieren eine einmalige Vorbereitungszeit erforderlich ($O(\Delta^{3/2})$). Diese Technik kann nicht auf dem Baum oder auf dem Band angewendet werden.



Ein Ausschnitt aus der Speicherstruktur "Hecke".

Die Speicherstruktur "Cube Connected Cycles" (CCC) ist von der gleichnamigen Parallelrechnerarchitektur abgeleitet, mit der verteilt arbeitende Prozessoren effizient gekoppelt werden können [PV81, MadH83]. Ein-Kopf Turingmaschinen mit der Speicherstruktur "CCC" sind besonders effizient. Sie können Mehrband Turingmaschinen in Zeit $O(t\,\sqrt{\log t})$ simulieren. Auch diese Simulation basiert auf einer schnellen Kopiertechnik. (Kopierzeit: $O(\Delta + \ell^2)$, Vorbereitungszeit: $O(\Delta^2\, 2^{\sqrt{\Delta}})$.)

Auf der Hecke und auf dem CCC können auch Turingmaschinen mit mehreren anderen Speicherstrukturen effizient simuliert werden. So können $t$ Schritte von Turingmaschinen mit mehreren $d$-dimensionalen Gittern auf einer Turingmaschine mit



Ein Ausschnitt aus der Speicherstruktur "Cube Connected Cycles".

einer einzelnen Hecke in Zeit $O(t \log t / \sqrt[d]{\log \log t})$ und auf einer Turingmaschine mit einem CCC in Zeit $O(t \log t / \sqrt[2d]{\log t})$ simuliert werden. Eine Hecke simuliert Turingmaschinen mit mehreren Bäumen oder Hecken in Zeit $O(t \log t / \log \log \log t)$.

Wir beschreiben eine konstruierbare Speicherstruktur, auf der $t$ Schritte von Mehrband Turingmaschinen in Zeit $O(t)$ simuliert werden können. Die simulierende Maschine nutzt ihre Speicherstruktur in einer ungewohnten Weise. Anstatt die Beschriftung der Bänder in den Zellen ihres Speichers abzulegen, merkt sie sich die Beschriftung der Bänder allein durch die *Position* ihres Kopfes. Die Linearzeit-Simulation funktioniert auch dann, wenn die simulierende Maschine ihren Speicher gar nicht beschreiben darf, sondern nur einen Pebble im Speicher verschieben kann.

Wir beschreiben eine weitere konstruierbare Speicherstruktur, auf der $t$ Schritte von Zwei-Band Turingmaschinen, die Platz $s$ benutzen, in Zeit $O(\sqrt{t \cdot s} + n^2)$ simuliert werden können. (Hierbei bezeichnet $n$ die Eingabelänge.) Auch hier werden Bandbeschriftungen vollständig durch die Position des Kopfes dargestellt. Mit einer einzelnen Bewegung des Kopfes in dieser Speicherstruktur werden gleich mehrere Schritte der Zwei-Band Turingmaschine simuliert. Nach einem Resultat von Hartmanis benutzen Ein-Band Turingmaschinen in Zeit $t$ maximal Platz $O(n + t / \log t)$ [Ha68]. Damit können $t$ Schritte von Ein-Band Turingmaschinen, $t = \Omega(n^2 \log n)$, in sublinearer Zeit $O(t / \sqrt{\log t})$ auf dieser konstruierbaren Speicherstruktur simuliert werden. Dies ist möglicherweise ein Hinweis darauf, daß Reischuks Definition konstruierbarer Speicher zu weit gefaßt ist.

Ich danke Martin Dietzfelbinger für wertvolle Hinweise und für die Gelegenheit zu ausführlichen Diskussionen. Diese Arbeit wurde durch die Deutsche Forschungsgemeinschaft unterstützt (DFG Projekt Di 412/2-2). Die Ergebnisse über die Speicherstruktur "Hecke" wurden auf der letzten MFCS vorgestellt [Hü95].

## Literatur

[DH94] M. Dietzfelbinger und M. Hühne, Matching upper and lower bounds for simulations of several tapes on one multidimensional tape. In *Proc. 14th FST & TCS*, LNCS 880, 1994, 24–35.

[Ha68] J. Hartmanis, Computational complexity of one-tape Turing computations. *Journal of the ACM* **15** (1968), 325–339.

[HS65] J. Hartmanis und R.E. Stearns, On the computational complexity of algorithms. *Transactions of the American Mathematical Society* **117** (1965), 285–306.

[Hü95] M. Hühne, The hedge: An efficient storage device for Turing machines with one head. In *Proc. 20th MFCS*, LNCS 969, 1995, 247–256.

[MadH83] F. Meyer auf der Heide, Infinite cube-connected cycles. *Information Processing Letters* **16** (1983), 1–2.

[PR81] W.J. Paul und K.R. Reischuk, On time versus space II. *Journal of Computer and Systems Sciences* **22** (1981), 312–327.

[PV81] F. Preparata und J. Vuillemin, The cube-connected cycles: A versatile network for parallel computation. *Communications of the ACM* **24** (1981), 300–309.

[Re90] K.R. Reischuk, *Einführung in die Komplexitätstheorie*. B.G. Teubner, Stuttgart, 1990.

[Ti87] P. Tiwari, Lower bounds on communication complexity in distributed computer networks. *Journal of the ACM* **34** (1987), 921–938.