X. Simulation von Turingmaschinen mit logischen Netzen

von Martin Fürer

1. Einleitung

Bevor eine Inhaltsübersicht gegeben werden kann, sind einige Definitionen nötig:

Definition 1 Ein logisches Netz ist ein endlicher gerichteter Graph $G = (P,K)$ (P = Menge der Punkte, K = Menge der gerichteten Kanten, $K \subset P \times P$) ohne gerichtete Zyklen mit folgenden zusätzlichen Strukturelementen und Eigenschaften:

- Die Menge P ist in folgende disjunkte Klassen eingeteilt:

a) Konstanten $\subset$ {null, eins}
b) Eingänge $= \{x_1,\ldots,x_n\}$
c) Ausgänge $= \{y_1,\ldots,y_m\}$
d) Und-Tore
e) Oder-Tore
f) Nicht-Tore (= Inverter)
g) Verzweigungspunkte
(Die Verzweigungspunkte sind eigentlich überflüssig. Sie dienen nur der schöneren Darstellung).

- In die Konstanten und Eingänge führen keine Kanten, in die Ausgänge, Nicht-Tore und Verzweigungspunkte je eine Kante, in die Und-Tore und Oder-Tore je zwei Kanten.
Von den Ausgängen gehen keine Kanten weg, von allen anderen Punkten können beliebig viele Kanten weggehen.

Definition 2 Ein logisches Netz definiert in folgender Weise eine Funktion $f: \{0,1\}^n \rightarrow \{0,1\}^m$ (n = Anzahl Eingänge, m = Anzahl Ausgänge): Einem n-tupel $z = (z_1,\ldots,z_n) \in \{0,1\}^n$ wird zunächst die Funktion $g_z: P \rightarrow \{0,1\}$ zugeordnet, welche folgende Eigenschaften hat:

(1) $g_z(\text{null}) = 0$, $g_z(\text{eins}) = 1$
(2) $g_z(x_i) = z_i \quad i = 1,\ldots,n$

Für alle $a,b,c \in P$:

(3) Falls c ein Und-Tor ist und $(a,c),(b,c) \in K$, so gilt:
$g_z(c) = 1 \iff g_z(a) = 1$ und $g_z(b) = 1$

(4) Falls c ein Oder-Tor ist und $(a,c),(b,c) \in K$, so gilt:
$g_z(c) = 1 \iff g_z(a) = 1$ oder $g_z(b) = 1$

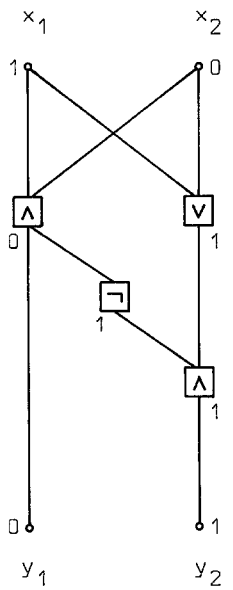
(5) Falls b ein Nicht-Tor ist und $(a,b) \in K$, so ist $g_z(b) = 1-g_z(a)$.

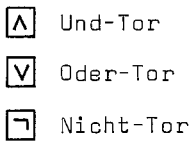
(6) Falls $b \in$ Ausgänge $\cup$ Verzweigungspunkte und $(a,b) \in K$, so ist $g_z(b) = g_z(a)$.

Weil das Netz zyklenfrei ist, gibt es zu jedem $z \in \{0,1\}^n$ genau eine solche Funktion g_z.
Die von einem logischen Netz berechnete Funktion f wird dann definiert durch $f(z) = (g_z(y_1),\ldots,g_z(y_m))$.

Bemerkung Logische Netze können in folgender Weise physikalisch realisiert werden: Die Kanten werden durch elektrische Drähte und die Tore durch entsprechende Schaltelemente dargestellt. Die Funktion g_z (für $z = (z_1,\ldots,z_n)$) gibt dann die Spannung an, die man in den Punkten des Netzes hat, wenn man an den Eingängen $x_1,\ldots,x_n$ die Spannungen $z_1,\ldots,z_n$ anlegt.

Beispiel





Bei jedem Punkt steht der Wert von $g_{(0,1)}$ in diesem Punkt. Alle Kanten sind von oben nach unten gerichtet.

Bezeichnungsweise

B = {0,1}
B* = Menge der endlichen 0-1-Folgen
= Menge der Wörter über dem Alphabet B

lh(X) = Länge der 0-1-Folge X

Definition 3 Eine Funktion f: $B^* \longrightarrow B^*$ heisst längentreu, falls gilt: lh(X) = lh(Y) ⟹ lh(f(X)) = lh(f(Y)).

Beispiel Die charakteristische Funktion irgendeiner Wortmenge ist längentreu, denn es gilt sogar lh(f(X)) = 1 für alle Wörter X.

Die Einschränkung $f|B^n$ einer längentreuen Funktion f auf Wörter der Länge n ist eine Funktion von B^n in B^m für eine natürliche Zahl m. Deshalb kann $f|B^n$ von einem logischen Netz berechnet werden, und die Komplexität von längentreuen Funktionen kann mit Hilfe von logischen Netzen gemessen werden.

Definition 4 Die kombinatorische Netzkomplexität einer längentreuen Funktion f ist die Funktion $C_f : \mathbb{N} \longrightarrow \mathbb{N}$, welche definiert ist durch:
$C_f(n)$ ist die kleinste Zahl k, zu der es ein logisches Netz mit k Toren gibt, welches $f|B^n$ berechnet.

Bemerkung Werden "logische Netze" anstatt mit den Toren $\wedge$, $\vee$, $\neg$ mit solchen definiert, welche zu einer anderen vollständigen Menge von Booleschen Funktionen gehören, so ändert sich die kombinatorische Netzkomplexität höchstens um einen konstanten Faktor.

Zu jedem logischen Netz gibt es zwei Zahlen n,m, so dass das logische Netz eine Funktion f: $B^n \longrightarrow B^m$ berechnet. Um irgendeine längentreue Funktion zu berechnen, wird eine ganze Folge von logischen Netzen benötigt, nämlich für jede Inputlänge n ein logisches Netz.

Definition 5 Eine Folge N von logischen Netzen, bei der das n-te Netz N_n genau n Eingänge hat, berechnet eine Funktion f, welche definiert ist durch:
$f|B^n$ ist die vom n-ten Netz N_n berechnete Funktion.

Bemerkung Jede längentreue Funktion wird von einer Folge von logischen Netzen berechnet. Dies gilt auch für nicht-rekursive längentreue Funktionen, z.B. für f mit

$$f(X) = \begin{cases} 0 & |X| \in A \\ 1 & |X| \in A \end{cases}$$

für eine nicht-rekursive Menge A.

Uebersicht In diesem Vortrag werden drei Sätze von (M. Fischer und N. Pippenger 1974) bewiesen. Das Hauptresultat ist Satz 3:
Sei M eine Mehr-Band-Turingmaschine, welche eine längentreue Funktion f für alle Inputwörter der Länge n in einer Zeit $\leq T(n)$ berechnet. Dann gilt $C_f(n) = O(T(n) \log T(n))$.

Dieses Resultat wird nicht durch eine direkte Simulation der Turingmaschinen mit Folgen von logischen Netzen bewiesen. Statt dessen wird die TM (Turingmaschine) M zuerst durch eine besonders einfach arbeitende TM (eine stereotype TM, siehe unten) simuliert (Satz 2), und erst diese wird mit logischen Netzen simuliert (Satz 1). Der Satz 2 wurde durch die Arbeit von (Hennie und Stearns 1966) inspiriert.

Mit Hilfe des Satzes 3 kann man untere Schranken für logische Netze sofort übertragen auf untere Schranken für Turingmaschinen. Insbesondere gilt: Falls die kombinatorische Netzkomplexität der charakteristischen Funktion einer vollständigen Sprache exponentiell ist, so ist $P \neq NP$ (vgl. Vortrag II).

2. Simulation von stereotypen Turingmaschinen durch logische Netze

Konventionen für Turingmaschinen

Wir betrachten TM mit mehreren Bändern. Zwei davon sind ausgezeichnet: Das Inputband und das Outputband. Das Alphabet Σ umfasst mindestens $\{0,1,*\}$. Das Input-Output-Alphabet ist $B = \{0,1\}$.
Eine Berechnung beginnt im Zustand s_0 mit folgenden Bandinschriften und Kopfstellungen (mit $\uparrow$ bezeichnet):
Auf dem Inputband $\overset{*}{\uparrow}X$ ($X \in B^*$ ist das Inputwort).
Auf allen übrigen Bändern $\overset{*}{\uparrow}$.
Diese Bezeichnung der Konfiguration bedeutet, dass weiter links und rechts überall das Leerzeichen "*" steht.

Eine TM berechnet eine Funktion $f: B^* \rightarrow B^*$, wenn für jeden Input $X \in B^*$ die Berechnung aufhört mit $f(X)\overset{*}{\uparrow}$ auf dem Outputband. Auf den übrigen Bändern darf dann etwas beliebiges stehen.

Definition 6 Eine IOTM (Input-Output-Turingmaschine) ist eine Mehr-Band-Turingmaschine mit der Eigenschaft, dass der Kopf auf dem Inputband und der Kopf auf dem Outputband nur von links nach rechts verschoben werden können.
Die übrigen Bänder heissen Arbeitsbänder.

Definition 7 Eine IOTM heisst stereotyp (engl.: oblivious), falls die totale Laufzeit und die Positionen der Lese-Schreib-Köpfe zu jeder Zeit t nur von der Länge des Inputs (und nicht vom speziellen Input-Wort) abhängen.

Definition 8 Eine Konfiguration einer k-Band-TM M ist ein Tripel <b,p,s>. Dabei sind
b eine Bandinschrift von M, d.h. ein k-Tupel von Funktionen von $\mathbb{Z}$ in das Alphabet $\sum$,
p ein k-Tupel von ganzen Zahlen (den Positionen der Köpfe) und
s ein Zustand von M.

Jeder IOTM mit k Bändern, Zustandsmenge S und einem Alphabet $\sum$ ordnen wir die Zahl $c_M = (2k+1)\cdot|S|\cdot|\sum|^k$ zu.

Satz 1 Es gibt ein effektives Verfahren F, um
- einer natürlichen Zahl n (Inputlänge) und
- einer stereotypen IOTM M, welche eine längentreue Funktion f in der Zeit T(n) berechnet,

ein logisches Netz F(M,n) zuzuordnen, welches $f|B^n$ mit $c_M\cdot T(n)$ Toren berechnet.
Es gilt also insbesondere $C_f(n) = O(T(n))$.

Beweis Die Konstruktion eines solchen logischen Netzes kann am besten mit Zeichnungen erläutert werden. Die Methode ist nämlich sehr einfach, aber es wäre schwierig, ein so grosses logisches Netz formal zu beschreiben.

Der Einfachheit halber wählen wir $\sum = \{0,1,*\}$, also $|\sum| = 3$. Die Abschätzungen sind für k-Bänder ausgerechnet, es wurden aber nur zwei Bänder gezeichnet. $S = \{s_0,s_1,s_2,\ldots,s_{|S|-1}\}$ ist die Menge der Zustände.

In der Zeit T(n) können höchstens Bandstücke der Länge T(n) benutzt werden.

Für jede Zeit t mit $0 \leq t \leq T(n)$ wählt man

a) für jeden Zustand einen Verzweigungspunkt und

b) für jedes Feld auf den benutzten Bandstücken drei Verzweigungspunkte (für jedes Zeichen $\in \sum$ einen Punkt).

Die Verzweigungspunkte, welche zu einer festen Zeit t gehören, sollen die Konfiguration zur Zeit t darstellen. Sie werden etwa auf gleicher Höhe, die Punkte zur späteren Zeit t+1 werden weiter unten angeordnet. Dazwischen wird ein Stück des logischen Netzes so gebaut, dass die dem Netz zugeordnete Funktion g_Z in einem zur Zeit t gehörigen Verzweigungspunkt P genau dann den Wert 1 annimmt, wenn gilt:

a) P gehört zu dem Zustand, in welchen M mit Input $Z = z_1 z_2 \cdots z_n$ nach t Schritten gelangt, oder

b) P gehört zu demjenigen Zeichen auf einem bestimmten Feld, welches zur Zeit t in diesem Feld steht.

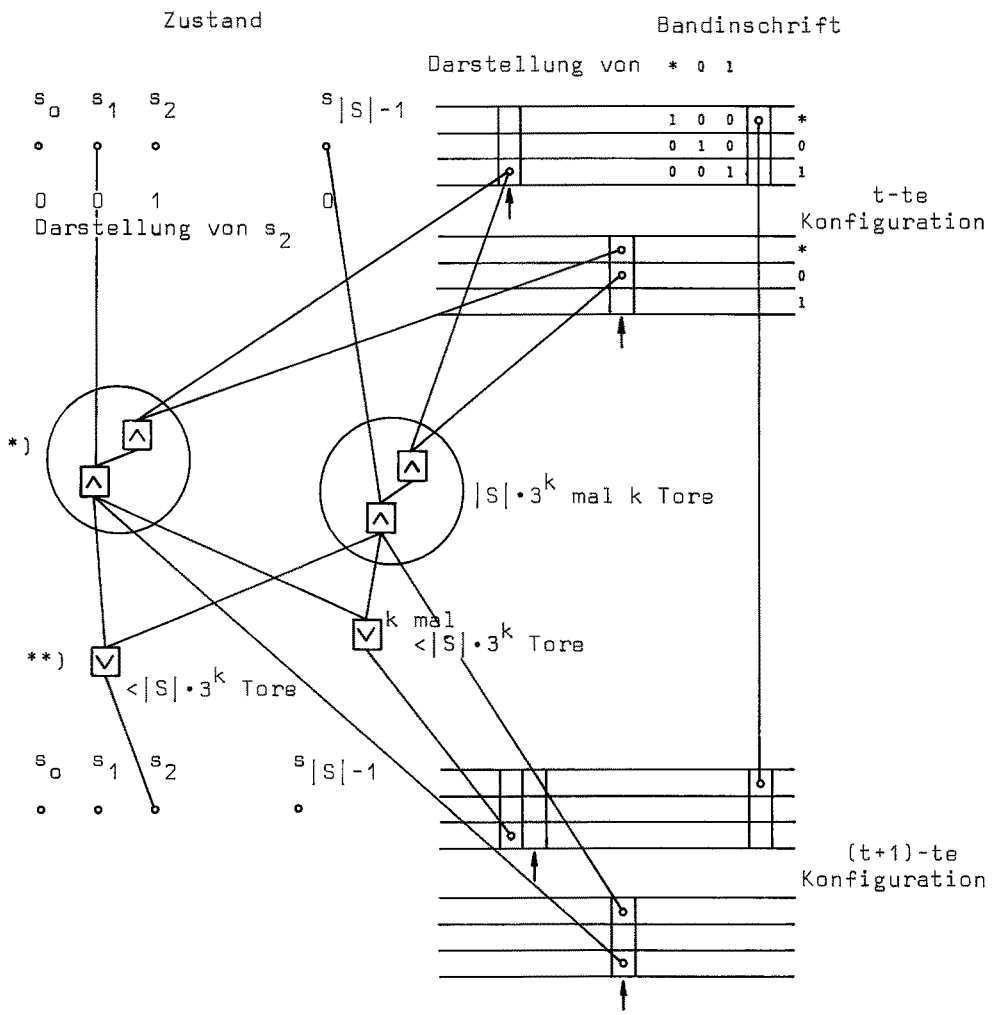
Weil die Maschine M stereotyp ist, befinden sich die Köpfe zur Zeit t für alle Inputwörter der Länge n in den gleichen Positionen. Zur Zeit t+1 kann daher fast die ganze Bandinschrift unverändert übernommen werden. Nur in den Positionen, wo sich die Köpfe befinden, ändert sich die Bandinschrift. Dabei liefert jede Kombination von Zustand und Bandinschrift an den beobachteten Stellen einen neuen Zustand und eine neue Bandinschrift an den gleichen Stellen. Dazu gehört das Netz-Stück, das mit der folgenden Zeichnung angedeutet ist.

Alle Kanten sind von oben nach unten gerichtet. Die Positionen der Köpfe (mit ↑ bezeichnet) sind bekannt, weil die IOTM stereotyp ist.

Zu *) (siehe Zeichnung nächste Seite):

Zum Uebergang von der t-ten zur (t+1)-ten Konfiguration gehört ein Netz-Stück, das aus $|S| \cdot |\sum|^k = |S| \cdot 3^k$ Kopien der eingekreisten Schaltung aufgebaut ist. $|S| \cdot 3^k$ ist nämlich die Anzahl Möglichkeiten, einen Zustand mit einer beobachteten Bandinschrift zu kombinieren. Hier wird speziell der Zustand s_1 mit einer 1 auf dem ersten Band und dem Leerzeichen * auf dem beobachteten Feld des zweiten Bandes kombiniert.

Die bezeichnete eingekreiste Schaltung drückt aus: Falls die IOTM zur Zeit t im Zustand s_1 auf dem ersten Band eine 1 und auf dem zweiten Band das Leerzeichen * beobachtet, so geht die IOTM zur Zeit t+1 über in den Zustand s_2, lässt die 1 auf dem ersten Band stehen und schreibt auch ins beobachtete Feld auf dem zweiten Band eine 1.



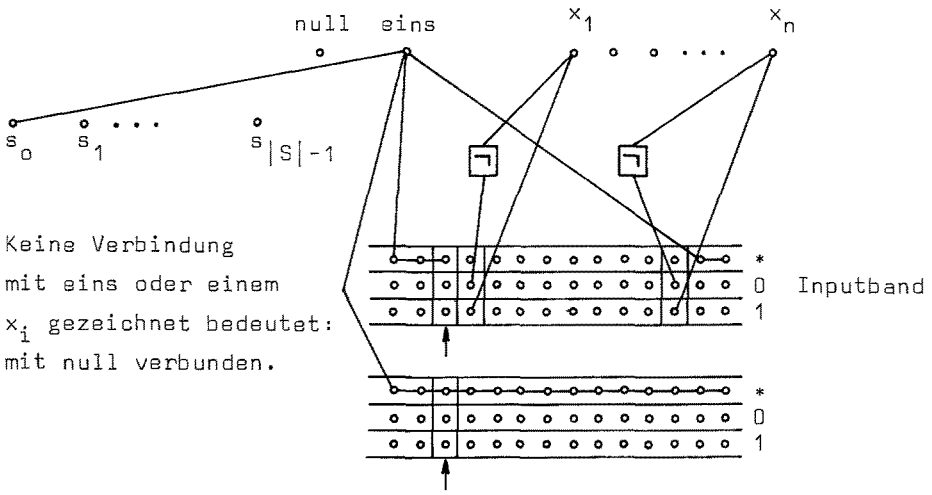
Zu **):

Falls mehrere Konfigurationen die gleiche Aktion (z.B. Uebergang in den Zustand s_2) bewirken, so werden die entsprechenden Befehle mit Oder-Toren gesammelt.

Pro Berechnungsschritt braucht es höchstens $(2k+1)\cdot|S|\cdot|\Sigma|^k = c_M$ viele Tore. Also gilt $C_f(n) \leq c_M \cdot T(n)$.

Vom logischen Netz müssen noch Anfang und Schluss definiert werden. Am Anfang ist die IOTM im Zustand s_o, und das Inputwort $X = x_1x_2\cdots x_n$ steht auf dem Inputband:

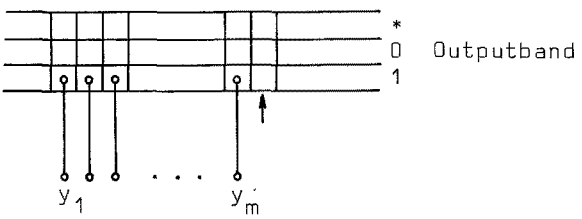
Anfang:



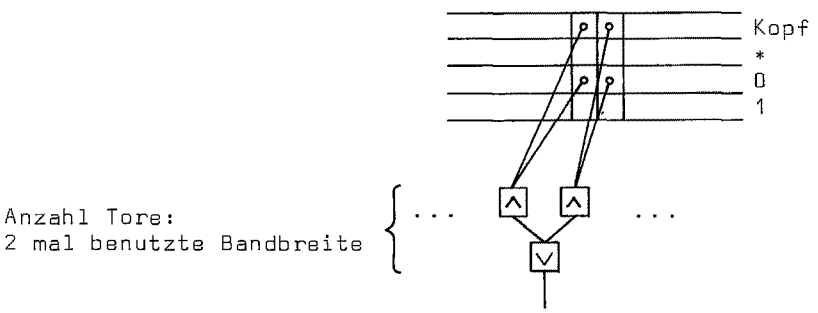
Schluss:

m ist die Outputlänge, welche zur Inputlänge n gehört.

Damit ist Satz 1 bewiesen.



Bemerkung Auf analoge Art können gewöhnliche (nicht stereotype) Mehr-Band-Turingmaschinen durch logische Netze direkt simuliert werden. Dies kann dadurch geschehen, dass in jedem Feld auf den Bändern noch ein Punkt eingeführt wird, der angibt, ob der Kopf dieses Feld beobachtet.
Folgende Schaltung stellt dann z.B. fest, ob eine 0 im beobachteten Feld steht:



Man bekommt so ein logisches Netz mit einer Grösse
< C•T(n)• (|S| + maximal benutzte Bandbreite), wobei die Konstante C nur

von der Anzahl Bänder und von der Grösse des Alphabets abhängt. Die benutzte Bandbreite ist aber im allgemeinen von der Grössenordnung T(n). Das logische Netz hat dann die Grössenordnung $(T(n))^2$. (Savage, Schnorr).

Wir werden zeigen, dass wir ein wesentlich kleineres Netz (der Grösse O(T(n) logT(n))) bekommen, wenn wir die gewöhnliche Mehr-Band-TM zuerst mit einer stereotypen TM und erst diese mit einem logischen Netz simulieren. (Fischer und Pippenger 1974).

(Schnorr 1975) hat bemerkt, dass das Produkt "Zeit mal Logarithmus der Zeit" ersetzt werden kann durch "Zeit mal Logarithmus der benutzten Bandbreite". Schnorr zeigt, dass auch die Komplexität von endlichen Booleschen Funktionen $f: B^n \rightarrow B$ mit Turingmaschinen sinnvoll gemessen werden kann, wenn die Programm-Grösse ins Komplexitätsmass einbezogen wird.

3. Simulation von Turingmaschinen durch stereotype Turingmaschinen

Satz 2 Sei M eine IOTM, welche eine längentreue Funktion f so berechnet, dass die Positionen der Köpfe auf dem Input- und dem Output-Band zu jeder Zeit t nur von der Länge n des Inputwortes und nicht von den speziellen Inputsymbolen abhängen. Ebenso soll auch die totale Laufzeit T(n) nur von n abhängen.
Dann gibt es eine stereotype IOTM M' (d.h. auch die Positionen der anderen Köpfe hängen zu jeder Zeit t nur von n ab) mit zwei Arbeitsbändern, welche f in der Zeit O(T(n)logT(n)) berechnet.

Bemerkung Die im folgenden beschriebene Simulationsmethode ist verwandt mit der Methode von (Hennie und Stearns 1966) zur Simulation von Mehr-Band-Turingmaschinen mit Zwei-Band-Turingmaschinen.

Beweisidee Aus dem Inhalt der Arbeitsbänder von M' soll der Inhalt der Arbeitsbänder der zu simulierenden Maschine M jederzeit ersichtlich sein. Diese Information wird auf den Arbeitsbändern von M' aber nicht kompakt gespeichert, sondern in Blöcke aufgeteilt, welche mit einer gewissen Regelmässigkeit so angeordnet sind, dass dazwischen ebenso grosse Lücken entstehen.

Auf einem Band kann die IOTM M in einem Schritt je nach dem beobachte-

ten Zeichen: Nur die Bandinschrift ändern (1.) oder den Kopf nach links (2.) oder rechts (3.) bewegen.
Um einen solchen Schritt von M zu simulieren, muss die stereotype IOTM M' (unabhängig vom beobachteten Zeichen) nach rechts und links bis zur nächsten Lücke gehen und wieder zurückkommen. Je nach dem zuerst beobachteten Zeichen macht M' in den entsprechenden Fällen folgendes:

1. Zuerst: Bandinschrift ändern. Auf der Wanderung: nichts.
2. Das beobachtete Zeichen wird nach rechts in die 1. Lücke verfrachtet. Vom 1. Block links wird ein Zeichen geholt.
3. Vom 1. Block rechts wird ein Zeichen geholt. Das beobachtete Zeichen wird in die nächste Lücke links gebracht.

Zwischen diesen Simulationsschritten werden in regelmässigen Abständen die Blöcke und Lücken wieder richtig verteilt. Damit auch grosse Blöcke rasch verschoben werden können, braucht man zwei Köpfe. Diese beiden Köpfe müssen aber nicht auf zwei Bänder verteilt sein. Die Beschreibung von M' wird einfacher, wenn sich beide Köpfe auf dem gleichen Arbeitsband befinden. Am Schluss des Beweises werden wir zeigen, dass das zweiköpfige Arbeitsband ohne weiteres durch zwei einköpfige Arbeitsbänder ersetzt werden kann.

Bemerkung Um Turingmaschinen mit mehrköpfigen Bändern genau zu definieren, müssen wir z.B. festlegen: Falls mehrere Köpfe gleichzeitig auf dasselbe Feld schreiben wollen, so hat der Kopf mit der grössten Nummer Priorität. Hier wird aber dieser Ausnahmefall nicht auftreten.

Beweis (Satz 2)

Das zweiköpfige Arbeitsband von M' hat mehrere Spuren. (Ein k-spuriges Band einer TM mit Alphabet Σ ist nichts anderes als ein gewöhnliches Band einer TM mit Alphabet Σ^k). Zur Darstellung eines halben Arbeitsbandes von M werden 3 Spuren (= 1 Kanal) auf dem Arbeitsband von M' verwendet. (Die Bänder von M denke man sich z.B. rechts vom Kopf entzwei geschnitten. Die rechte Bandhälfte wird herübergeklappt und durch einen anderen Kanal dargestellt). Eine Illustration folgt weiter unten.

Falls M mit dem Alphabet Σ arbeitet, so benutzt M' das Alphabet $\sum \cup \{+\}$. Dabei wird + als Leerzeichen verwendet, denn * wird gebraucht, um das Leerzeichen * von M darzustellen.
Das Arbeitsband von M' ist in Segmente verschiedener Länge unterteilt. Das i-te Segment ($i \geq 0$) umfasst die Felder 2^i bis $2^{i+1}-1$. Nur die Band-

hälfte mit positiven Feldnummern wird benutzt. Das Wort, welches auf einer Spur in einem Segment steht, heisst Block. Jeder Block ist entweder ganz voll (ohne +) oder ganz leer (+++...+).
Durchgeht man alle nicht leeren Segmente in absteigender Folge und setzt alle vollen Blöcke eines Kanals (innerhalb der Segmente nach Spurnummern geordnet) zusammen, so erhält man den Inhalt eines simulierten Halb-Bandes von M. Wir sagen: Der Kanal beschreibt dieses Halb-Band.

Eine mögliche Darstellung eines Bandes von M mit der Inschrift ...**abcdefghik&m**... (a,...,m$\in \sum$) auf dem Arbeitsband der stereotypen TM M' ist:

9	8	7	6	5	4	3	2	1	
...	*	1	*	*	*	0	*	0	1	Spur 0
...	*	*	a	b	c	d	e	f	g	Spur 1.1
...	+	+	+	+	+	+	+	+	h	Spur 1.2 } Kanal 1
...	+	+	+	+	+	+	+	+	+	Spur 1.3
			...		*	*	*	m	i	Spur 2.1
			...		+	+	&	k	+	Kanal 2
			...		+	+	+	+	+	
					:					voller Block: * m
										leerer Block: + +
3				2			1		0	Segmentnummer

Die Bedeutung von Spur 0 wird später erklärt.

Definition 9 Eine korrekte Konfiguration von M' ist eine Konfiguration, bei der alle Blöcke entweder ganz voll oder ganz leer sind und bei der sich in keinem Kanal ein leerer Block über einem vollen Block befindet.

Die Zustandsmenge S' von M' sei S×$\overline{S}$ (S = Zustandsmenge von M, $\overline{S}$ beliebige Menge).

Definition 10 Eine korrekte Konfiguration K' = (b',p',s') von M' stellt eine Konfiguration K = (b,p,s) von M dar, falls gilt:

1. Werden die Bänder von M rechts von den Kopfpositionen entzwei geschnitten, so werden die entstehenden Halb-Bänder durch die entsprechenden Kanäle des Bandes von M' beschrieben.

2. p' beliebig
3. s' = <s,$\bar{s}$> für irgendein $\bar{s}$.

Bemerkung Eine Konfiguration von M kann durch mehrere Konfigurationen von M' dargestellt werden.
Werden aber zwei Konfigurationen K_1, K_2 von M durch die gleiche Konfiguration K' von M' dargestellt, so unterscheiden sich K_1 und K_2 nur durch eine Verschiebung der Feld-Numerierung auf den Bändern von M.

Bei einer korrekten Konfiguration definieren wir:

Definition 11 Für einen gegebenen Kanal heisst ein Segment sauber, falls 1 oder 2 Blöcke voll sind. Sind 0 oder 3 Blöcke voll, so heisst es schmutzig. (Die vollen Blöcke befinden sich immer in den oberen Spuren eines Kanals).

Wir beschreiben nun eine Prozedur Sim(n), welche angibt, wie M' 2^n Schritte von M simuliert.

Eigenschaften von Sim(n) Gegeben sei eine korrekte Konfiguration K'_1 von M' mit Kopf 1 bei Feld 1 und Kopf 2 bei Feld 2. Die ersten n Segmente seien für alle Kanäle sauber. Dann leistet Sim(n) folgendes:
1. Die Segmente 0,1,...,n-1 sind auch nachher sauber.
2. In Segment n ändert sich die Anzahl volle Blöcke pro Kanal höchstens um ± 1.
3. Die Segmente n+1, n+2,... bleiben unverändert.
4. Die Konfiguration K'_2 von M' nach der Ausführung von Sim(n) ist wieder korrekt. Falls K'_1 die Konfiguration K_1 von M darstellt und K_1 nach 2^n Schritten von M übergeht in K_2, so stellt K'_2 die Konfiguration K_2 dar.

Definition von Sim(n) (mit rekursivem Aufruf)

Sim(0) = Simuliere 1 Schritt der IOTM M.
Die Köpfe auf dem Arbeitsband von M' verschieben sich dabei nicht.

Sim(n+1) = Führe folgende Prozeduren aus:
1. Sim(n)
2. Reinige(n)
3. Zurück(n) (Köpfe zurück zu den Feldern 1 und 2).
4. Sim(n)
5. Reinige(n)

Mit Hilfe dieser Prozeduren, die noch genauer beschrieben werden, kann die Arbeitsweise von M' am einfachsten mitgeteilt werden. Diese Prozeduren sind aber keine Turingmaschinenprogramme, denn es kommen darin rekursive Aufrufe vor. Erst später wird gezeigt, wie man M' als TM definiert.

Zuerst überlegen wir uns, dass die Köpfe immer am richtigen Ort sind: Vor der Ausführung von Sim(n) befinden sich die Köpfe 1 und 2 über den Feldern 1 und 2. Nachher befinden sie sich über den Feldern 2^n und 2^{n+1}, gerade richtig, um mit Reinige(n) das Segment n zu reinigen. Dabei werden Daten zwischen den Segmenten n und n+1 verschoben.

Der zeitliche Ablauf sieht jetzt so aus: Nach der Simulation des $2^x(2y+1)$-ten Schrittes werden die Segmente 0,1,2,...,x der Reihe nach gereinigt. Dann gehen die Köpfe zurück auf die Felder 1 und 2 für die Simulation des nächsten Schrittes.

Beschreibung von Sim(0) (Segment 0 sei sauber)
Die linke Hälfte (inklusive beobachtetes Zeichen) eines Bandes werde durch Kanal 1, die rechte Hälfte (ohne beobachtetes Zeichen) werde gespiegelt durch Kanal 2 dargestellt.

Die IOTM M kann auf diesem Band einen der folgenden Schritte ausführen:
1. Bandinschrift ändern
2. Kopf um 1 nach links bewegen
3. Kopf um 1 nach rechts bewegen

Diese Schritte werden auf M' durch Sim(0) folgendermassen simuliert:
1. Der unterste volle Block im Segment 0 von Kanal 1 wird geändert. Kanal 2 bleibt unverändert.
2. Der unterste volle Block im Segment 0 von Kanal 1 wird vertauscht mit dem ersten leeren Block im Segment 0 von Kanal 2.
3. Umgekehrt.

Gleichzeitig werden mit den Kanälen 3, 4, 5, ... die anderen Bänder simuliert. Den neuen Zustand von M merkt sich M' mit ihrem eigenen Zustand.
Im folgenden beschreiben wir nur, was mit einem Kanal geschieht. Mit den anderen Kanälen wird gleichzeitig dasselbe gemacht.

Beschreibung von Reinige(n) (Es seien weder alle Blöcke eines Kanals in den Segmenten n und n+1 voll, noch alle leer).

Das Unterprogramm Reinige(n) reinigt das Segment n, indem es Daten zwischen den Segmenten n und n+1 verschiebt.

Falls alle Spuren (eines Kanals) in Segment n leer sind, so wird der unterste volle Block aus Segment n+1 geholt. Durch Halbierung werden daraus 2 obere Blöcke von Segment n gemacht.
Falls alle Spuren in Segment n voll sind, so werden die oberen 2 Blöcke von Segment n zu einem Block zusammengesetzt und nach Segment n+1 gebracht. Gleichzeitig wird der verbleibende Block in Segment n von der untersten in die oberste Spur verschoben.
Falls Segment n schon sauber ist, so werden keine Blöcke verschoben.
In jedem dieser 3 Fälle werden aber die gleichen Bewegungen der Köpfe ausgeführt, denn die IOTM M' soll stereotyp sein.

Beweis der Eigenschaften von Sim(n) (mit Induktion nach n):

n = 0: klar.

n + 1: Wir setzen voraus, dass zu Beginn die Segmente 0,...,n+1 sauber sind. Sim(n+1) ruft zweimal Sim(n) auf. Nach Induktionsvoraussetzung wird dabei höchstens das n-te Segment schmutzig. Anschliessend wird jeweilen das n-te Segment gereinigt. Dabei wird höchstens das (n+1)-te Segment schmutzig. Es ist aber nicht möglich, dass beim Reinigen beide Male Information in die gleiche Richtung verschoben wird, denn bei zweimaligem Aufruf von Sim(n) ändert sich die Anzahl volle Blöcke im n-ten Segment höchstens um ± 2. Das ergibt höchstens ± 1 Block für das (n+1)-te Segment. Alle grösseren Segmente bleiben natürlich unverändert.
Weil mit beiden Aufrufen von Sim(n) je 2^n Schritte von M simuliert werden, so werden mit Sim(n+1) 2^{n+1} Schritte simuliert.

Rechenzeit R

Reinige(n) gehe folgendermassen vor sich: (Der Einfachheit halber gehen die Köpfe immer 1 Feld zu weit, d.h. bis zum ersten Feld des nächsten Segmentes).

1. Kopf 1 befindet sich am Anfang des n-ten, Kopf 2 am Anfang des (n+1)-ten Segmentes.
2. Kopf 2 geht bis zur Mitte, Kopf 1 bis ans Ende des Segmentes.

   Dabei werden Daten in beiden Richtungen übertragen. Kopf 1 sieht an jeder Stelle, ob Daten übertragen werden müssen, denn ein leerer Block besteht nur aus +, ein voller Block enthält nirgends +:

$2 \cdot 2^n$ Schritte (2^n Bewegungsschritte, dazwischen je eine Aenderung in Bandinschrift).

3. Kopf 1 geht zurück an den Anfang des n-ten Segmentes: 2^n Schritte.
4. Kopf 2 geht über die 2. Hälfte des (n+1)-ten Segmentes. Gleichzeitig geht Kopf 1 über das n-te Segment. Dabei werden wieder in beiden Richtungen Daten übertragen: $2 \cdot 2^n$ Schritte.
5. Kopf 1 befindet sich jetzt am Anfang des (n+1)-ten, Kopf 2 am Anfang des (n+2)-ten Segmentes.

$R(\text{Reinige}(n)) = 5 \cdot 2^n$

$R(\text{Zurück}(n)) = 2^{n+2} - 2 < 4 \cdot 2^n$

$R(\text{Zurück}(0)) = 2$

$R(\text{Sim}(0)) = 1$

Behauptung $R(\text{Sim}(n)) \leq 7 \cdot 2^n \cdot n$ für $n>0$

Beweis

$$\begin{aligned} R(\text{Sim}(1)) &= 2 \cdot R(\text{Sim}(0)) + 2 \cdot R(\text{Reinige}(0)) + R(\text{Zurück}(0)) \\ &= 2 + 2 \cdot 5 + 2 \\ &= 7 \cdot 2^1 \cdot 1 \end{aligned}$$

$$\begin{aligned} R(\text{Sim}(n+1)) &= 2 \cdot R(\text{Sim}(n)) + 2 \cdot R(\text{Reinige}(n)) + R(\text{Zurück}(n)) \\ (n>0) \quad &< 2 \cdot 7 \cdot 2^n \cdot n + 2 \cdot 5 \cdot 2^n + 4 \cdot 2^n \\ &= 7 \cdot 2^{n+1} \cdot (n+1) \end{aligned}$$

Sei $2^{\ell-1} < T(n) \leq 2^\ell$

Dann vermag Sim(ℓ) die ganze Berechnung zu simulieren.

Rechenzeit von M' = $R(\text{Sim}(\ell)) \leq 7 \cdot 2^\ell \cdot \ell < 7 \cdot 2T(n) \cdot (\log T(n)+1)$

Also gilt für $2 \leq T(n)$:

Rechenzeit von M' $\leq C \cdot T(n) \cdot \log T(n)$ für eine Konstante C.

Spur 0 Wir haben die stereotype IOTM M' bis jetzt noch nicht vollständig definiert. Zwei Fragen sind noch offen:

1. Wie stellt M' die Segmentgrenzen fest?
2. Wie merkt sich M', wo sie in der Ausführung der Prozedur Sim(n) steht?

Dazu wird die Spur 0 verwendet:

In Spur 0 wird der Anfang jedes Segmentes durch 0 oder 1 markiert, während sonst * steht. Von diesen Markierungen kann die Maschine nicht nur die Segmentgrenzen, sondern auch die Schrittzahl ablesen: Die Nullen und Einsen geben als Binärzahl an, wieviele Schritte schon simuliert

wurden.
Die Zerlegung der Schrittzahl $z = 2^x(2y+1)$ gibt an, dass nach dem z-ten Simulationsschritt die Segmente 0, 1, 2, ..., x gereinigt werden müssen. Die Schrittzahl wird während dem Reinigen geändert. Die Maschine muss sich dabei immer nur einen Uebertrag merken. Es müssen genau diejenigen Segmente gereinigt werden, bei welchen sich die Binärziffer beim Zählen ändert.

Initialisierung Zu Beginn der Berechnung ist das ganze Arbeitsband von M' leer, d.h. in jedem Feld steht das Leerzeichen +. Bis jetzt haben wir aber immer vorausgesetzt, dass zu Beginn der Berechnung alle Segmente sauber seien. Dazu müsste z.B. in jedem Kanal die erste Spur mit * gefüllt sein, während die beiden anderen Spuren leer wären. Damit würden leere Arbeitsbänder von M dargestellt (* ist das Leerzeichen von M). Ebenso sollten zu Beginn der Berechnung in Spur 0 gewisse Marken vorhanden sein. Die Maschine M' muss aber nicht ihr Arbeitsband zuerst so initialisieren, denn sie kann einfach jedes Segment dann initialisieren, wenn es zum ersten mal (von Kopf 2) überstrichen wird:

- In Spur 0 wird der Segmentanfang mit 0 markiert.
- Die Maschine verhält sich so, als wäre die erste Spur jedes Kanals mit * statt mit dem Leerzeichen + gefüllt.
- Kopf 1 gibt die Grösse des Segmentes an, indem er gleichzeitig zweimal das halb so grosse Segment überstreicht.

Weiter wird beim ersten Simulationsschritt eine 1 in Spur 0 von Segment 0 gesetzt.

Die IOTM M' soll anhalten, sobald alle Schritte von M simuliert sind.

M' ist dann eine stereotype IOTM mit einem zweiköpfigen Arbeitsband und allen in Satz 2 behaupteten Eigenschaften.

Durch geringfügige Aenderungen kann eine IOTM konstruiert werden, welche mit zwei einköpfigen Arbeitsbändern dasselbe leistet.
Dazu bieten sich etwa folgende drei Methoden an:

1. Das zweite Arbeitsband wird nur als Hilfsband verwendet. Jeder Block, der auf dem Hauptband verschoben werden muss, wird zuerst auf das Hilfsband übertragen.
2. Besonders schnell geht die Simulation, wenn alle geraden Segmente auf dem einen und alle ungeraden Segmente auf dem anderen Band angeordnet werden.

3. Wenn das Alphabet vergrössert wird, so kann nach (Stoss 1970) jede TM mit einem k-köpfigen Band in der gleichen Zeit durch eine TM mit k einköpfigen Bändern simuliert werden. Falls die erste TM stereotyp ist, so ist es auch die zweite.

Damit ist Satz 2 bewiesen.

Bemerkung 1. Zwei Köpfe (auf ein oder zwei Bänder verteilt) sind nötig, um die Verschiebung der Blöcke in linearer Zeit durchzuführen. Bei dieser Methode wäre sonst die totale Rechenzeit von der Grössenordnung $(T(n))^2$.

2. Auch die IOTM M, welche simuliert wird, darf mehrere Köpfe auf dem gleichen Band haben. Sie kann zuerst nach (P.C. Fischer, Meyer und Rosenberg 1972) in der gleichen Zeit durch eine IOTM mit einem Kopf pro Band so simuliert werden, dass die Input- (Output-) Symbole zu den genau gleichen Zeiten gelesen (geschrieben) werden.
Eine einfachere Simulation von Mehr-Kopf-TM durch stereotype Zwei-Kopf-TM wird in (Fischer und Pippenger 197?) beschrieben.

Aus den Sätzen 1 und 2 folgt unmittelbar, dass jede IOTM, welche die Voraussetzungen von Satz 2 erfüllt, durch eine Folge von logischen Netzen der Grösse $O(T(n) \log T(n))$ simuliert werden kann. Dies gilt aber sogar für jede TM, welche eine längentreue Funktion berechnet.

Satz 3 Sei T eine Funktion von $\mathbb{N}$ in $\mathbb{N}$ und sei M eine Mehr-Band-Turingmaschine, welche eine längentreue Funktion f für alle Inputwörter der Länge n in einer Zeit $\leq T(n)$ berechnet.
Dann gilt $C_f(n) = O(T(n) \log T(n))$.

Beweis Falls für irgendein n_0 gilt $T(n_0) \leq n_0$, so kann M für alle $n \geq n_0$ nicht einmal den ganzen Input lesen, und die Behauptung von Satz 3 gilt trivialerweise. Im folgenden können wir daher voraussetzen $n<T(n)$.

Die TM M kann nicht wie im Satz 2 direkt durch eine stereotype IOTM simuliert werden, weil M für gleich lange Wörter ganz verschiedene Rechenzeiten haben kann. Die stereotype IOTM müsste sich dann für ein Wort der Länge n nach der maximalen Rechenzeit von M für gleich lange Wörter richten. Zur Berechnung der maximalen Rechenzeit zu gegebener Wortlänge gibt es aber vermutlich kein schnelles Verfahren.

Wir konstruieren deshalb eine IOTM M", welche ausser dem Input X von M

auch noch eine geschätzte Rechenzeit t als Input bekommt. Die IOTM M" mit dem Input $1^{\ell}0X$ der Länge $lh(1^{\ell}0X) = t$ simuliert dann t Schritte von M mit Input X. Falls die Schätzung genügend gross war, so wird also richtig simuliert.
M" soll linear beschränkt sein, d.h. falls die Schätzung t richtig ist, so soll sich die Rechenzeit nur um einen konstanten Faktor vervielfachen. M" wird selber noch nicht stereotyp sein, sondern nur die Voraussetzungen von Satz 2 erfüllen. Mit Hilfe der Sätze 1 und 2 erhalten wir dann eine Folge von logischen Netzen, welche M" simuliert. Schliesslich werden die Netze so abgeändert, dass sie f berechnen.

Konstruktion von M" Zusätzlich zu den Bändern von M habe M" ein weiteres gewöhnliches Band (Zählband), sowie ein Input- und ein Output-Band, deren Köpfe sich nur nach rechts bewegen können. Dem Inputband von M entspricht ein Pseudo-Inputband von M", dem Outputband von M ein Pseudo-Outputband von M". Einem Input $X \in \{0,1\}^*$ von M und einem beliebig vorgegebenen $t > lh(X)$ entspricht ein Input $Y \in \{0,1\}^*$ von M" der Form $1^{\ell}0X$ mit $\ell \geq 0$ und $t = lh(Y)$.

M" arbeitet folgendermassen:
1. Der Input Y wird vom Inputband gelesen. Dabei wird auf dem Zählband die Länge von Y vermerkt, während X auf das Pseudo-Inputband kopiert wird. (Es wird erst etwas kopiert, nachdem die Maschine eine 0 überlesen hat).
1^t ist der einzige Input, der nicht die Form $1^{\ell}0X$ hat. Auch bei diesem Input muss M" die Voraussetzungen von Satz 2 erfüllen. Wir können die IOTM M" aber ohne weiteres so konstruieren, dass sie beim Input 1^t genau das gleiche macht wie beim Input $1^{t-1}0$. (Sie simuliert dann M mit leerem Input).
Während der nächsten t Schritte bewegt sich der Kopf auf dem Pseudo-Inputband wieder an den Anfang von X und wartet dort.

2. M" führt t Schritte von M aus. Die Schritte werden auf dem Zählband gezählt. Der Input kommt vom Pseudo-Inputband, der Output geht auf das Pseudo-Outputband. Falls M" früher fertig wird, so werden leere Befehle ausgeführt, d.h. es wird nur noch gezählt.

3. Die ersten t Zeichen, welche links vom Kopf auf dem Pseudo-Outputband stehen, werden auf das Outputband geschrieben. (Falls bei 2. die ganze Berechnung simuliert worden ist, so steht nach unserer Konvention der ganze Output links vom Kopf). Auf dem Zählband werden die Schritte gezählt. Anstatt das Leerzeichen * wird z.B. 0 auf das Outputband über-

tragen. Dadurch wird die von M" berechnete Funktion g längentreu, denn es gilt jetzt lh(g(Y)) = lh(Y) = t.

Nach den Sätzen 1 und 2 gibt es eine Folge N von logischen Netzen, welche M" simuliert. Weil M" linear beschränkt ist, so hat das Netz N_t zur Berechnung von $g|B^t$ die Grössenordnung O(t log t). Für t = T(n) erhalten wir das Netz zur Berechnung von $f|B^n$ aus dem Netz N_t, indem wir die ersten t-lh(f(0^n)) Ausgänge (t=T(n) $\geq$ lh(f(0^n))) weglassen und die ersten t-n Eingänge (t=T(n)>n) durch Verzweigungspunkte ersetzen, von denen der letzte mit "null" und alle anderen mit "eins" verbunden sind. Weil sich die Anzahl Tore dabei nicht vergrössert, gilt für die logische Netzkomplexität: $C_f(n) = O(T(n) \log T(n))$.

Damit ist Satz 3 bewiesen.

Bemerkung Falls wir uns nur für logische Netze und nicht für stereotype IOTM interessieren, so sind noch einige Verbesserungen um konstante Faktoren möglich. Die IOTM M" kann dann durch eine TM-ähnliche Maschine ersetzt werden, welche ohne zu zählen die richtige Anzahl Schritte ausführt (Zählband und "$1^{\ell}0$" überflüssig) und bei der die Kopfbewegungen nicht durch die Zustände gesteuert werden. Dann wird auch die Spur 0 im Beweis von Satz 2 nicht gebraucht.

Literatur

Fischer, M.J. und Pippenger, N.J.(197?), in Vorbereitung

Fischer, M.J. und Pippenger, N.J., private Mitteilungen.

Fischer, P.C. and Meyer, A.R. and Rosenberg, A.L., Real-Time Simulation of Multihead Tape Units, J. ACM 19 (1972) 590-607.

Hennie, F.C. and Stearns, R.E., Two-Tape Simulation of Multitape Turing Machines, J. ACM 13 (1966) 533-546.

Schnorr, C.P., The Network Complexity and the Turing Machine Complexity of Finite Functions, 1975 Annual Meeting of GAMM, Göttingen, April 2-5 (1975).

Stoss, H.-J., k-Band-Simulation von k-Kopf-Turing-Maschinen, Computing 6 (1970) 309-317