

UNTERSUCHUNGEN ÜBER HALTENDE PROGRAMME FÜR TURING-MASCHINEN
MIT 2 ZEICHEN UND BIS ZU 5 BEFEHLEN

B.WEIMANN, K.CASPER, W.FENZL

Zunächst sollen Definitionen von Begriffen gegeben werden, die in Richtung auf das Ziel dieser Arbeit hin sehr spezieller Natur sind und sich nicht mit den sonst üblichen Definitionen zu decken brauchen.

Def.1: Eine Turing-Maschine besteht im wesentlichen aus einer Zentraleinheit (ZE), einem Band, das nach beiden Seiten unbegrenzt ist und einem Lese- und Schreibkopf (LS). Das Band ist in Felder eingeteilt, die durchnummeriert sind, und zwar vom Feld 0 ausgehend nach rechts die Felder 1,2,3,... und nach links die Felder -1,-2,-3,... Der LS steht über einem Feld des Bandes, dem Arbeitsfeld (AF). Die ZE kann von einem Turingprogramm (TP) gesteuert das Zeichen auf dem AF lesen und in Abhängigkeit davon und dem TP ein Zeichen auf das AF schreiben. Zwei Zeichen stehen zur Verfügung: B und I, wobei B für Leerzeichen steht. Grundsätzlich wird beim Schreibvorgang das AF überschrieben. Mit B überschreiben heißt, das AF löschen. Weiter kann die ZE wieder in Abhängigkeit vom gelesenen Zeichen und vom TP den LS um ein Feld nach rechts oder links verschieben.

Def.2: Ein Turingprogramm besteht aus einer endlichen Anzahl (N) von Befehlen. Jeder Befehl besteht aus zwei Teilbefehlen und hat folgende Form:

$$i B Z_1 V_1 n_1$$
$$i I Z_2 V_2 n_2$$

Dabei bedeuten: i ist die Nummer des Befehls; B und I regeln in Abhängigkeit vom Inhalt der AF's, welcher Teilbefehl durchgeführt wird, d.h. steht auf dem AF ein I, wird der 2. Teilbefehl durchgeführt; $Z_1, Z_2 \in \{B, I\}$ sind die Zeichen, die auf das AF geschrieben werden; $V_1, V_2 \in \{R, L\}$ sind die auszuführenden Verschiebeoperationen, d.h. $V_1=R$, so wird der LS ein Feld nach rechts verschoben; $n_1 \in \{1, \dots, N\}$ ist die Nummer desjenigen Befehls, die die ZE als nächsten ausführt. Ist ein Teilbefehl von der Form iBZ bzw. iIZ , d.h. sind V und n nicht definiert, so handelt es sich um einen Stopbefehl. Dabei wird das Zeichen Z noch auf das AF geschrieben, bevor die ZE die Turingmaschine anhält.

Generalvoraussetzung: TP-e werden nur daraufhin untersucht, was sie bewirken, wenn sie mit dem 1. Befehl gestartet werden und beim Start ein leeres Band und als AF das Feld mit Nummer 0 vorfinden. Wird also aus-

gesagt, daß ein TP stoppt, so gilt dies nur unter obiger Voraussetzung.

Will man alle möglichen TP-e mit n Befehlen unter Berücksichtigung obiger Generalvoraussetzung untersuchen, so wird dieser Versuch bei $(4 \cdot (n+1))^{2n}$ möglichen Programmen sehr schnell zum Scheitern verurteilt sein. Es muß der Versuch gemacht werden, die große Zahl der TP-e durch sinnvolle Identifizierungen in möglichst wenige Äquivalenzklassen einzuteilen. So konnten S. Lin und T. Redo [1] durch Identifizierungen die bei 5 Befehlen möglichen 16 777 216 TP-e auf 82944 Äquivalenzklassen aufteilen. Ihr Verfahren auf TP-e mit 4 Befehlen angewandt, würde zu 100 663 296 Äquivalenzklassen führen. Durch die im folgenden definierten Identifizierungen von TP-en gelang es, die Anzahl der Äquivalenzklassen (bei 4 Befehlen) auf 1 198 690 zu reduzieren.

Folgende TP-e werden miteinander identifiziert:

1. Numeriert man die $2n$ Teilbefehle bei TP-en mit n Befehlen in kanonischer Reihenfolge durch und wird der Ablauf eines TP-s durch eine Folge der Nummern der nacheinander von der ZE ausgeführten Teilbefehle beschrieben, so werden zwei TP-e mit den Folgen i_1, i_2, i_3, \dots bzw. j_1, j_2, j_3, \dots für die Beschreibung des Ablaufs miteinander identifiziert, wenn gilt:

- a) $i_k = j_k \quad k = 1, 2, 3, \dots, s$ für TP-e, die nach Ausführung von genau s Teilbefehlen anhalten.
- $k \in \mathbb{N} \quad$ für nicht-haltende TP-e
- b) Der Teilbefehl mit Nummer i_k ist bei beiden TP-en identisch.

Die nicht benutzten Teilbefehle können als Stopbefehle aufgefaßt werden.

Notation: Ein Teilbefehl, der beim Ablauf eines TP-s nicht benutzt wird, soll durch die Form iB^* bzw. iI^* gekennzeichnet werden.

- 2. Zwei TP-e, die sich nur durch eine Permutation der Reihenfolge der Befehle bei entsprechender Permutation der Nummern der aufzurufenden Befehle unterscheiden.
- 3. Zwei TP-e, die sich nur dadurch unterscheiden, daß sie alle Verschiebeoperationen spiegelbildlich zueinander ausführen, d.h. wo ein R im ersten TP steht, steht im zweiten ein L und umgekehrt.

Mit Hilfe dieser Identifizierungen bleiben für den ersten Teilbefehl des ersten Befehls (dies ist immer der Startbefehl) nur zwei Möglichkeiten (bei TP-en mit mehr als einem Befehl):

- a) 1BIR2
- b) 1BR2

Wird im folgenden von einer Menge von TP-en gesprochen, so ist eine Teilmenge derjenigen Menge gemeint, die aus je einem Vertreter jeder der durch obige Identifizierungen entstehenden Äquivalenzklassen besteht.

Lecture Notes in Economics and Mathematical Systems 78, 72-81

GI. Gesellschaft für Informatik e.V. 2. Jahrestagung (Karlsruhe, 1972)

Die Gesamtheit aller TP-e wird nun in folgende disjunkte Teilmengen aufgespalten:

- $TP \in S(n,m) \iff$
1. TP stoppt
 2. TP besitzt n Befehle
 3. m Teilbefehle von TP werden nach dem Start benutzt
 4. Von jedem Befehl wird mindestens 1 Teilbefehl benutzt ($n = m$)

- $TP \in U(n,m) \iff$
1. TP stoppt nicht
 - 2., 3. und 4. wie bei $S(n,m)$

Wie man sich leicht überlegt, erhält man für $n=1$ folgende Teilmengen:

$$S(1,1) = \left\{ \begin{pmatrix} 1B \\ 1I^* \end{pmatrix}, \begin{pmatrix} 1BB \\ 1I^* \end{pmatrix} \right\} \quad U(1,1) = \left\{ \begin{pmatrix} 1BIR^1 \\ 1I^* \end{pmatrix}, \begin{pmatrix} 1BBR^1 \\ 1I^* \end{pmatrix} \right\}$$

$$S(1,2) = \emptyset \quad U(1,2) = \emptyset$$

Man kann nun, nacheinander von $S(1,1)$ anfangend, alle Teilmengen $S(n,m)$ und $U(n,m)$ aufbauen. Um $S(n,m)$ zu ermitteln, benötigt man alle TP-e, die stoppen und $m-1$ verschiedene Teilbefehle bis zum Stop benutzen. Dies sind

$$S_{n,m} = S(n,m-1) \cup S(n-1,m-1) \quad \text{für } m > n$$

$$S_{n,m} = S(n-1,m-1) \quad \text{für } m = n$$

Zu allen TP-en aus $S(n-1,m-1)$ wird zunächst ein n -ter Befehl hinzugenommen, der von der Form $\begin{pmatrix} 0B \\ nI^* \end{pmatrix}$ ist, so daß alle TP-e aus $S_{n,m}$ n Befehle aufweisen. Folgender Prozeß wird für jedes TP aus $S_{n,m}$ durchgeführt. Sei $TP_1 \in S_{n,m}$ und TP_1 stoppe auf dem Teilbefehl $iB \dots (iI \dots)$. Dieser Teilbefehl wird dann nacheinander ersetzt durch alle Teilbefehle der Form $iBZVn_1, (iIZVn_2)$ mit $Z \in \{B, I\}$, $V \in \{R, L\}$, $n_1 \in \{1, \dots, n\}$ ($n_2 \in \{1, \dots, n\}$). Jedes der so entstandenen TP-e wird daraufhin geprüft, ob es anhält oder nicht. Hält es an, kommt es zur Menge $S(n,m)$. Hält es nicht an, so werden genau wie bei TP_1 $m-1$ Teilbefehle benutzt. (Jeder nicht benutzte Teilbefehl bei TP_1 ist von der Form iB^* bzw. iI^* und damit Stopfbefehl). Dieses Programm kommt dann zur Menge $U(n-1,m-1)$ bzw. $U(n,m-1)$, je nachdem, ob $TP_1 \in S(n-1,m-1)$ oder $TP_1 \in S(n,m-1)$. Der Aufbauprozeß ist in Abb. 1 schematisch wiedergegeben.

Es muß untersucht werden, ob bei diesem Aufbau aus jeder Äquivalenzklasse genau ein Vertreter konstruiert wird. Bei diesen Überlegungen macht nur die Frage Schwierigkeiten, ob zwei TP-e konstruiert werden können, die wegen Identifizierung 2 derselben Äquivalenzklasse angehören. Für ein TP mit n Befehlen sei i_1, i_2, i_3, \dots die Folge der Nummern der nach dem Start nacheinander auszuführenden Befehle (nicht Teilbefehle), also $i_k \in \{1, \dots, n\}$, $k=1, 2, 3, \dots$, so gibt es unter der Voraussetzung, daß in der Folge i_1, i_2, \dots alle Zahlen von 1 bis n vorkommen (diese Voraussetzung ist durch die Definition von $S(n,m)$ und $U(n,m)$ erfüllt), genau eine Permutation π auf $\{1, \dots, n\}$, die folgendes leistet:

$$\pi(i_{j+1}) = \max\{\pi(i_k) \mid k = 1, \dots, j\} + 1$$

$$\pi(i_1) = 1$$

$$\pi(i_{j+1}) = \max\{\pi(i_k) \mid k = 1, \dots, j\} + 1, \text{ wenn } i_{j+1} \text{ mit}$$

keiner der Zahlen i_1, \dots, i_j übereinstimmt.

Wie man leicht sieht, werden beim geschilderten Aufbau nur TP-e zugelassen, die Folgen (wie oben beschrieben) besitzen, bei denen die Identität die eindeutig bestimmte Permutation ist, die dies leistet.

Um die Hauptergebnisse dieser Arbeit formulieren zu können, müssen einige Funktionen definiert werden.

Def. 3: Die Funktionen $\sigma, \varphi, \beta: S(n,m) \rightarrow \mathbb{N}$ für alle $n, m \in \mathbb{N}$ $n \leq m \leq 2n$ und $k, s, f, b: \mathbb{N}^2 \rightarrow \mathbb{N}$ werden definiert durch:

$\sigma(TP)$ = Anzahl der Teilbefehle, die bis zum Stopfbefehl einschließlich durchgeführt werden.

$\varphi(TP)$ = Anzahl der verschiedenen Felder, die als Arbeitsfelder auftreten (Feldweite von TP).

$\beta(TP)$ = Anzahl der I's, die nach dem Stop auf dem Band stehen.

$k(n,m) = |S(n,m)|$ = Anzahl der TP-e in $S(n,m)$

$s(n,m) = \max\{\sigma(TP) \mid TP \in S(n,m)\}$

$f(n,m) = \max\{\varphi(TP) \mid TP \in S(n,m)\}$

$b(n,m) = \max\{\beta(TP) \mid TP \in S(n,m)\}$ (Busy-Beaver-Spiel)

Um einen noch genaueren Überblick zu bekommen, wurden die Teilmengen $S(n,m)$ und $U(n,m)$ in disjunkte Teilmengen zerlegt:

$$S(n,m) = S_{BR}(n,m) \cup S_{IR}(n,m) \cup S_{BB}(n,m) \cup S_{IB}(n,m)$$

$$U(n,m) = U_{BR}(n,m) \cup U_{IR}(n,m) \cup U_{BB}(n,m) \cup U_{IB}(n,m)$$

Die einzelnen Teilmengen werden definiert durch:

$TP \in S_{BR}(n,m) \iff$

1. TP $S(n,m)$
2. Der erste Befehl lautet 1BBR2
3. Alle benötigten AP's haben eine Nummer ≥ 0
(Nur die "rechte Seite" des Bandes wird benutzt)

$TP \in S_{IR}(n,m) \iff$

1. und 3. wie bei $S_{BR}(n,m)$
2. Der erste Befehl lautet 1BIR2

$TP \in S_{BB}(n,m) \iff$

1. und 2. wie bei $S_{BR}(n,m)$
3. "Beide Seiten" des Bandes werden benutzt.

$TP \in S_{IB}(n,m) \iff$

1. und 3. wie bei $S_{BB}(n,m)$, 2. wie bei $S_{IR}(n,m)$

Entsprechend werden die Teilmengen von $U(n,m)$ definiert. Durch Indizierung der Funktionen k, s, f, b (z.B. k_{BR}, k_{IR}, \dots) sollen die Funktionen gekennzeichnet werden, die dadurch entstehen, daß in den Definitionen statt der Menge $S(n,m)$ die der Indizierung entsprechende Teilmenge zugrunde gelegt wird.

n, m	k _{BR}	S _{BR}	f _{BR}	i _{BR}	k _{IR}	S _{IR}	f _{IR}	i _{IR}	k _{BB}	S _{BB}	f _{BB}	i _{BB}	k _{IB}	S _{IB}	f _{IB}	i _{IB}
2,2	2	2	2	1	2	2	2	2	--	--	--	--	--	--	--	--
2,3	2	4	2	1	8	3	2	2	--	--	--	--	--	--	--	--
2,4	2	6	3	2	8	4	2	2	--	--	--	--	12	6	4	4
3,3	8	3	3	2	8	3	3	3	--	--	--	--	--	--	--	--
3,4	38	5	3	2	54	5	3	3	4	5	3	2	36	5	4	4
3,5	138	9	4	4	266	10	4	4	32	9	4	4	282	17	5	5
3,6	378	14	5	4	590	13	5	4	180	17	6	5	1492	21	7	6
4,4	24	4	4	3	24	4	4	4	8	4	3	3	8	4	3	3
4,5	320	7	4	4	424	7	4	4	96	8	4	4	366	6	5	5
4,6	2376	18	5	5	3362	12	5	5	1210	18	6	5	4704	20	6	6
4,7	14148	24	7	6	20446	23	7	5	10864	30	8	7	45548	38	10	9
4,8	45504	43	11	7	59664	58	10	7	56082	71	12	10	230762	107	16	13

Tabelle 1

n, m	k _{BR}	S _{BR}	f _{BR}	i _{BR}	k _{IR}	S _{IR}	f _{IR}	i _{IR}
5,5	96	5	5	4	96	5	5	5
5,6	2036	9	5	5	2320	9	5	5
5,7	27804	19	6	6	36180	19	6	6
5,8	276610	33	9	8	357192	39	9	8
5,9	?	178	16	12	?	84	14	13
5,10	?	237	21	18	?	189	22	19

Tabelle 2

TP	1BBR2 1IBL2 2BBR3 2IIL4 3BIL1 3IIR3 4BBL1 4IB	1BBR2 1IBL4 2BBR3 2IIR2 3BIR4 3IBR4 4BIL1 4II	1BBR2 1IIL3 2BIR3 2IIR2 3BIR4 3IIL1 4BIL1 4II	1BIR2 1IBL3 2BIR3 2IB 3BIR4 3IBR2 4BIL4 4IBL1	1BIR2 1IIL3 2BBR3 2IIR3 3BIL1 3IIR4 4BIR2	1BIR2 1IBL2 2BBR3 2IIR4 3BIL1 3IBR4 4BIR3 4II	1BBR2 1IIL3 2BIL1 2IIR2 3BIL2 3IBL4 4BI 4IBL1	1BBR2 1IIR2 2BIL3 2IIR3 3BIR1 3IIL4 4BI 4IBL3	1BBR2 1IIL3 2BIR3 2IIR1 3BIL1 3IBL4 4BIR1 4II
¹ Index	BR	BR	BR	IR	IR	IR	BB	BB	BB
σ(TP)	43	35	20	58	42	19	71	45	64
ψ(TP)	6	11	7	8	10	7	12	12	11
β(TP)	2	5	7	4	7	7	8	9	10

TP	1BIR2 1IIL2 2BIL1 2IBL3 3BI 3IIL4 4BIR4 4IBR1	1BIR2 1IBR3 2BIL1 2IIR1 3BI 3IIR4 4BII4 4IBI2	1BIR2 1IBR1 2BIL3 2IBR4 3BBL4 3IBL2 4BIR1 4II	TP	1BBR2 1IBL1 2BBR3 2IBL5 3BIR4 3IBL4 4BIR1 4IBL4 4IIR5 5BIR3 5IB	1BBR2 1IIR5 2BBR3 2IIL2 3BIL4 3IIR2 4BIR1 4IBL4 5BIR3 5IIR2	1BIR2 1IBL1 2BBR3 2IIL2 3BIR4 3IIR3 4BIR5 4IBR5 5BIL1 5II	1BIR2 1IBL1 2BBR3 2IIR5 3BIR4 3IIL3 4BIL1 4IIR3 5BIR4 5IIR3
¹ Index	IB	IB	IB	TP	BR	BR	IR	IR
σ(TP)	107	96	61		237	169	189	170
ψ(TP)	14	14	16		15	21	19	22
β(TP)	13	13	9		5	18	18	19

Tabelle 3

¹ z.B. Index=IB ↔ TP ∈ S_{IB}(n, m)

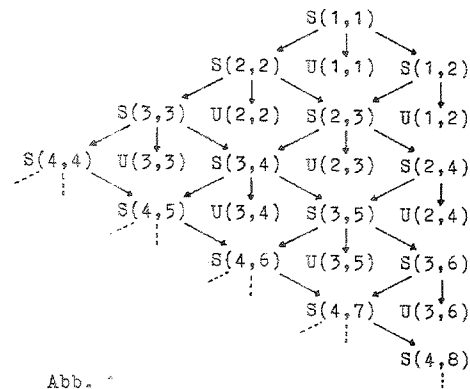


Abb. 1

Die Menge S^n aller haltenden TP-e mit n Befehlen unter Beachtung der Identifikationen 1, 2 und 3 ergibt sich zu:

$$S^n = \sum_{i=1}^n \sum_{j=1}^{2n} S(i, j)$$

Entsprechend ergibt sich die Menge U^n aller nicht-haltenden TP-e mit n Befehlen zu:

$$U^n = \sum_{i=1}^n \sum_{j=1}^{2n} U(i, j)$$

Um die Teilmengen $S(n, m)$ und $U(n, m)$ zu ermitteln, wurde ein Computerprogramm geschrieben, das den in Abbildung 1 schematisch dargestellten Aufbau leistet. Die Entscheidung, ob ein TP bei Variation des Teilbefehls, auf dem es stoppt, noch anhält, wurde zunächst danach getroffen, ob es hält, bevor eine vorgegebene Anzahl von Teilbefehlen ausgeführt ist, d.h. ein TP könnte zunächst fälschlicherweise als nicht-haltend gelten, wenn $G(\text{TP})$ größer als ein vorgegebener Wert ist. Die zunächst als nicht-haltend angenommenen TP-e wurden dann, wie noch geschildert wird, mit Hilfe weiterer Computerprogramme exakt daraufhin untersucht, ob sie halten oder nicht. Alle als nicht-haltend angenommenen TP-e wurden dann auch als solche erkannt, so daß die in Tabelle 1 angegebenen Werte für $n=4$ als bewiesen anzusehen sind. Damit ist unter anderem das Busy-Beaver-Spiel für $n=4$ gelöst (Lösung: 13).

In Tabelle 2 sind noch einige Werte angegeben, die erhalten wurden bei dem Versuch, auch die Teilmengen $S(5, 1)$, $U(5, \dots, 10)$ und die Werte der entsprechenden Funktionen zu bestimmen.

Auf Grund des enormen Anwachsens der Rechenzeiten der Computerprogramme konnte dies nur teilweise gelöst werden. Die angegebenen Werte gelten unter der Voraussetzung:

$$\text{TP} \in S_{\text{BR}}(5, 10) \cup S_{\text{IR}}(5, 10) \implies G(\text{TP}) \leq 500$$

d.h., die zunächst wie bei $n=4$ als nicht-haltend angenommenen TP-e wurden nicht weiter untersucht.

In der Tabelle 3 sind Programme angegeben, die einige Maximalwerte der Tabellen 1 und 2 realisieren.

Zur Untersuchung der zunächst als nicht-haltend angenommenen TP-e wurden die verschiedenen Teilmengen wie folgt zusammengefaßt:

$$\begin{aligned} U_{\text{BR}}^2 &= U_{\text{BR}}(2, 2) \cup U_{\text{BR}}(1, 2) & U_{\text{BR}}^5 &= U_{\text{BR}}(4, 5) \cup U_{\text{ER}}(3, 5) \\ U_{\text{BR}}^3 &= U_{\text{BR}}(3, 3) \cup U_{\text{BR}}(2, 3) & U_{\text{BR}}^6 &= U_{\text{BR}}(4, 6) \cup U_{\text{ER}}(3, 6) \\ U_{\text{BR}}^4 &= U_{\text{BR}}(4, 4) \cup U_{\text{BR}}(3, 4) \cup U_{\text{BR}}(2, 4) & U_{\text{BR}}^7 &= U_{\text{BR}}(4, 7) \end{aligned}$$

$U_{\text{BR}}^8 = U_{\text{BR}}(4, 8)$ braucht nicht gebildet zu werden, da TP-e aus U_{BR}^8 keinen Stopfbefehl besitzen und somit trivialerweise nicht halten. Analog zu

U_{BR}^i werden die Mengen U_{IR}^i , U_{BB}^i und U_{IB}^i gebildet.

Vier Algorithmen wurden aufgestellt, mit deren Hilfe bei bestimmten Bedingungen das Nicht-halten eines TP-s exakt nachgewiesen wurde. Sie sollen hier nur kurz skizziert werden.

1. Algorithmus zur Erkennung einfacher Schleifen (SCH)

Der Algorithmus versucht, einen Teilbefehl zu finden, der beim Ablauf des Programms immer wieder aufgerufen wird und bei dem die Bandbelegungen, von dem AF aus gesehen, das dieser Teilbefehl benutzt, in allen Feldern, die beim weiteren Ablauf noch als AF-er benutzt werden, identisch ist.

2. Der 2. Algorithmus (ST1) sortiert die TP-e aus, die nicht halten können, weil der (die) Stopfbefehl(e) bzw. der (die) nicht benutzte(n) Befehl(e) von keinem Teilbefehl aufgerufen wird (werden).

3. Der 3. Algorithmus (ST2) prüft, welche Möglichkeiten es bei entsprechender Bandbelegung gibt, daß das Programm zu einem Stopfbefehl (nicht benutzter Befehl) kommt. Dabei wird versucht, vom Stop her rückwärts den Ablauf der letzten 5 Teilbefehle zu konstruieren. Gibt es keine Bandbelegung, die zum Stop führen könnte, wird das Programm als nicht-haltend aussortiert.

4. Algorithmus zur Erkennung von sich vergrößernden Schleifen (VS).

Zur Demonstrierung, was dieser Algorithmus leistet, soll ein Beispiel dienen: Der LS ändert beim Ablauf des Programms unter anderem seine Richtung bei den Feldern 2, 3, 4, 5, ... (in chronologischer Reihenfolge), und zwar von rechts nach links. Zwischen den aufgeführten "Wendepunkten" ändert der LS seine Richtung von rechts nach links bei den Feldern 7, 9, 11, 13, ... Es wird nun untersucht, ob sich der Zyklus $2 \rightarrow 7 \rightarrow 3$ (Ablauf des Programms vom AF mit Nummer 2 über das AF mit Nummer 7 zum AF mit Nummer 3) im anschließenden Zyklus $3 \rightarrow 9 \rightarrow 4$ in "gewisser Weise" wiederholt. Läßt sich nachweisen, daß sich diese Art von Wiederholung beliebig fortsetzt, wird das Programm als nicht-haltend aussortiert.

In den Tabellen 4a, b, c, d sind die Zahlen der als nicht-haltend angenommenen TP-e sowie die Zahlen der nach Anwendung der einzelnen Algorithmen übrigbleibenden Programme aufgeführt. Die nach Anwendung aller 4 Algorithmen noch übrigbleibenden 396 Programme, mußten dann "per Hand" als nicht-haltend aussortiert werden. Bei diesen TP-en handelt es sich um solche, die mit abgewandelten Algorithmen der 4. Art aussortiert werden können.

i	Anzahl	U ⁱ _{BR} Anzahl nach Reduktion durch			
		SCH	ST1	ST2	VS
2	5	-	-	-	-
3	24	-	-	-	-
4	189	2	-	-	-
5	1409	11	7	5	-
6	9922	164	62	47	-
7	60800	1278	1007	466	40
Σ	72340	1455	1076	518	40

Tabelle 4a

i	Anzahl	U ⁱ _{BB} Anzahl nach Reduktion durch			
		SCH	ST1	ST2	VS
2	2	-	-	-	-
3	14	-	-	-	-
4	121	-	-	-	-
5	1223	16	8	5	-
6	10724	373	179	60	-
7	88504	4783	3361	1958	92
Σ	100588	5172	3548	2023	92

Tabelle 4c

i	Anzahl	U ⁱ _{IR} Anzahl nach Reduktion durch			
		SCH	ST1	ST2	VS
2	4	-	-	-	-
3	27	-	-	-	-
4	229	-	-	-	-
5	1890	8	-	-	-
6	13797	96	50	33	-
7	84653	809	372	181	8
Σ	100600	913	422	214	8

Tabelle 4b

i	Anzahl	U ⁱ _{IB} Anzahl nach Reduktion durch			
		SCH	ST1	ST2	VS
2	-	-	-	-	-
3	14	-	-	-	-
4	248	4	-	-	-
5	2940	51	40	27	-
6	34390	1033	391	186	2
7	298086	12044	9822	5334	254
Σ	335678	13132	10253	5547	256

Tabelle 4d

Programmfeld				
Leitwerk		Befehlspeicher		
Adresse	Index	Operationsteil	AdrTeil	
A	B	C	D	E
1				
	I			
2	B			
	I			
3	B			
	I			

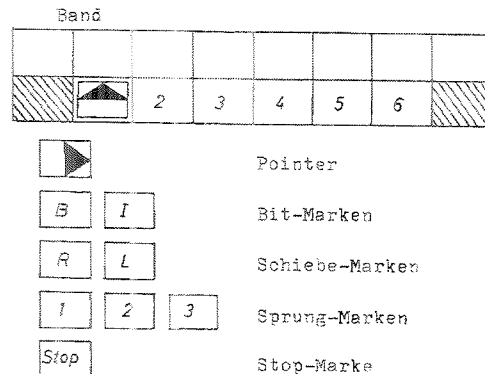


Abb. 2

Es ist naheliegend, eine leicht modifizierte Turingmaschine als Ausbildungsobjekt im Bereich der Informatik nutzbar zu machen. Die Eigenschaft als Lehrmodell ergibt sich aus der Eigenschaft der Maschine:

- a) Sie enthält alle grundsätzlichen Funktions- und Strukturelemente, die ein Computer besitzt, b) alle Funktionen und Strukturen sind unmittelbar anschaulich, c) der Gesamt Ablauf und die Wirkungsweise der Maschine bleiben während des logischen Ablaufs übersichtlich und direkt einsichtig, ohne daß das erzielte Ergebnis trivial ist.

Wir haben als Lehrmodell eine Turingmaschine mit drei Befehlen und einem endlichen Band mit 6 Zellen benutzt. Die Felder werden im Spielablauf durch verschiedene Marken besetzt. Feldabschnitte tragen Namen, die Funktionen im Computer entsprechen. Die Phasen des Spielablaufs werden durch einen "Pointer" im Leitwerk und im unteren Teil des Bandes markiert (s. Abb.2). Die Spielregeln entsprechen denen der Turingmaschine, die Selbsthalteforderung wird aber immer explizit angegeben.

Das Modell kann in verschiedenen Stufen der Ausbildung eingesetzt werden, wir haben es zur Grundausbildung für Programmierer benutzt. Dabei liegt im 1. Teil der Ausbildung, der Spielphase, das Schwergewicht auf der Beschäftigung mit dem Modell. Seine Ausnutzung erfolgt unter 4 Gesichtspunkten:

- a) Einsichten in logische Abläufe (durch Nachvollzug gegebener Programme) und Entwicklung konstruktiven Denkens (durch Erstellung von Bandstrukturen), b) Darstellung der Information, die sehr leicht aus den Spielmarken und deren Bedeutung erarbeitet werden kann, c) Veranschaulichung der wichtigsten Computerbegriffe und Verständnis der Rechnerorganisation, d) Grundschemata des Programmierens, wie Sprung, Verzweigung und Schleife. Die Modellaufgaben, der Spielanweisung beigefügt, werden dem Stand der Ausbildung inhaltlich und graduell angepaßt, wobei auch Anregungen zu rein spielerischen Übungen (z.B. maximale Bandbesetzung) eingefügt sind.

Von dieser Spielphase aus aufbauend, erfolgt die Einführung in eine Programmiersprache, wobei jetzt der Hintergrund für den Lernenden wesentlich transparenter ist und viele Begriffs- und Vorstellungsschwierigkeiten bereits überwunden sind. Nach unserer Erfahrung läßt sich der Erfolg in der Spielphase und der Programmierphase korrelieren, so daß sich hieraus ggf. didaktische Möglichkeiten ergeben. Ein systematischer und zahlenmäßiger Zusammenhang wurde jedoch nicht ermittelt, da unsere Bestrebungen mehr auf Nutzung im Unterricht als auf Testeignung gerichtet waren.

[1] S. Liu, T. Rado; Journal of the ACM 12 (1965)