EIN AUTOMATENTHEORETISCHES MODELL EINER SPEICHERHIERARCHIE *

H.-G. STORK

0.Einleitung.Es wird ein einfaches Modell eines Rechnersystems mit Vielfachprogrammierung beschrieben. Die kleinsten, zwischen Speicher und Hilfsspeicher bewegten Einheiten seien die Seiten der Programme. Die Algorithmen, welche das Rangieren von Seiten regeln, werden - etwa wie in |3| - als Automaten formalisiert. Es wird eine Aussage über "Seitenverkehr" und "Speicherauslastung" bewiesen. Ein weiterer Zweck dieser Note ist die Illustration einer im Zusammenhang mit endlichen Automaten bisher wenig beachteten Fragestellung durch ein konkretes Beispiel.

1.Beschreibung des Modells. N sei eine endliche Menge, und es sei $\not\in \notin N$.

DEFINITION 1:Ein Referenzengenerator über N ist ein endlicher Automat
$\Pi = (R, r^0, I, N \cup \{\not\in\}, \vartheta, \lambda)$ mit
- R endliche Zustandsmenge
- $r^0 \in R$ Anfangszustand
- $I = \{0,1\}$ Eingabemenge
- $N \cup \{\not\in\}$ Ausgabemenge

Die Transitionsfunktion $\vartheta: R \times I \to R$ und die Ausgabefunktion $\lambda: R \times I \to N \cup \{\not\in\}$ genügen folgenden Bedingungen:$\forall r \in R: \vartheta(r,0) = r$, $\lambda(r,0) = \not\in$, $\lambda(r,1) \neq \not\in$.

Bemerkung: Π ist ein autonomer Automat, der ein- und ausgeschaltet werden kann: wir deuten Π als ein Programm, das, solange es nicht unterbrochen wird, "Referenzen" auf die ihm zugewiesenen "Seiten" erzeugt.

DEFINITION 2:Eine Paging-Strategie über N ist ein endlicher Automat
$P = (Q, q^0, N \cup \{\not\in\}, \delta, \tau)$ mit:
- Q endliche Zustandsmenge
- $q^0 \in Q$ Anfangszustand
- $N \cup \{\not\in\}$ Eingabemenge

Die Transitionsfunktion $\delta: Q \times (N \cup \{\not\in\}) \to Q$ und die Funktion $\tau: Q \to 2^N$ genügen folgenden Bedingungen:

i) $\forall q \in Q: \delta(q,\not\in) = q$

ii) $\forall q \in Q, \forall x \in N: x \in \tau\delta(q,x)$

iii) $\tau q^0 = \emptyset$

Wie üblich kann δ auf $(N \cup \{\not\in\})^*$ fortgesetzt werden.

Bemerkung: $\tau q \subseteq N$ ist die Menge der dem "Paging-Zustand" q zugeordneten Seiten.

* 1. GI-Fachtagung AFS, LNCS 2, 1973

Bezeichnung: Einem Zustandsübergang $(q,x) \to \delta(q,x)$ des Automaten P sind die Kosten $k_p = |\tau\delta(q,x) - \tau q| + |\tau q - \tau\delta(q,x)|$, $k_p = k_p(q,x)$, zugeordnet. (Für $A \in 2^N$ ist $|A| = \text{card } A$!)
(k_p ist die Anzahl der bei einem Zustandsübergang rangierten Seiten)

$N_1, \ldots, N_k$ seien nun paarweise disjunkte, endliche Mengen; $\Pi_1, \ldots, \Pi_k$ seien Referenzengeneratoren (d. h. "Programme" !) über $N_1, \ldots, N_k$ resp., welche sich einen Speicher der Kapazität $m \in \mathbb{N}$ (d. h. der Speicher kann höchstens m verschiedene Seiten aufnehmen) teilen und jeweils die Paging-Strategien $P_1, \ldots, P_k$ benutzen. ($\Pi_i = (R_i, \ldots)$, $P_i = (Q_i, \ldots)$) Für letztere fordern wir außerdem, daß $|\tau_i q_i| \leq 1$, $q_i \in Q$, $1 \leq i \leq k$, wobei 1 ein Teiler von m ist. Zur Beschreibung dieses Systems zu einem gegebenen Zeitpunkt benötigen wir die Information darüber,

1. welche Programme im Speicher vertreten sind,
2. welches die Zustände der $P_1, \ldots, P_k$ und
3. welches die Zustände der $\Pi_1, \ldots, \Pi_k$ sind.

Schließlich darf die Gesamtzahl der Seiten von im Speicher enthaltenen Programmen m nicht überschreiten.

Bezeichnungen: Es sei $\not\S \notin \bigcup Q_i$.

1. $Q = (Q_1 \cup \{\not\S\}) \times \ldots \times (Q_k \cup \{\not\S\})$
2. $R = R_1 \times \ldots \times R_k$
3. $S: Q \to 2^{[k]}$, $[k] = \{1, \ldots, k\}$, ist gegeben durch:
   $S(q) = \{i \mid i \in [k], p_i(q) \neq \not\S\}$
4. $Z = \{(z_1,z_2) \mid z_1,z_2 \in Q,\ p_i(z_1) \neq \not\S \iff p_i(z_2) = \not\S, \sum_{i \in S(z_1)} |\tau_i p_i(z_1)| \leq m\}$

(Dabei ist p_i die Projektion auf die i-te Komponente.)

DEFINITION 3: Eine Paging-Maschine (über $\Pi_1, \ldots, \Pi_k$ und $P_1, \ldots, P_k$) ist ein endlicher Automat $M = (Z \times R, (z^o, r^o), \Delta, \Lambda)$ mit:

- $Z \times R$, Z und R wie oben, Zustandsmenge
- $(z^o, r^o) = |(q_1^o, \ldots, q_k^o), (\not\S, \ldots, \not\S), (r_1^o, \ldots, r_k^o)|$ Anfangszustand

$\Delta: Z \times R \to Z \times R$ und $\Lambda: Z \times R \to \{0,1\}^k$ sind Abbildungen, die folgenden Bedingungen genügen:

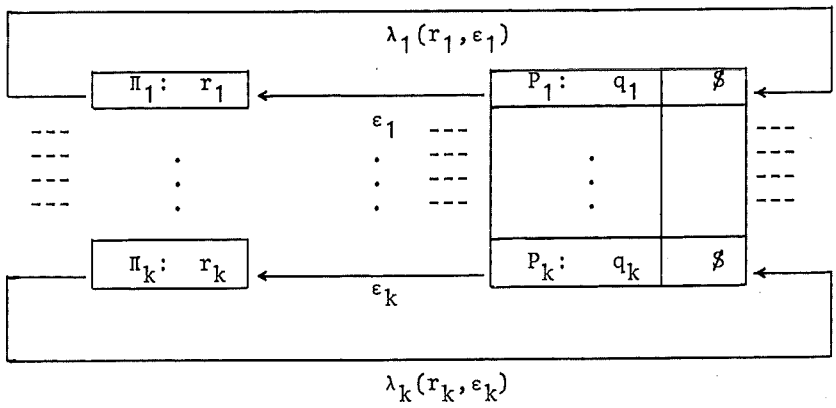
i) $\Lambda(z_1,z_2,r) = (\varepsilon_1, \ldots, \varepsilon_k)$, $\varepsilon_i = 1 \iff i \in S(z_1)$, $1 \leq i \leq k$

ii) Sei $\Delta(z_1,z_2,r) = (z_1', z_2', r')$. Für $i = 1, \ldots, k$ gilt:

   a) $p_i(z_j') = \delta_i(q_i, \lambda_i(r_i, \varepsilon_i))$ für genau ein $j = 1,2$, wobei:
      $\not\S \neq q_i = p_i(z_j)$ für genau ein $j = 1,2$
   b) $p_i(r') = \vartheta_i(r_i, \varepsilon_i)$
   c) $r \neq \bar{r}$, $\delta_i(q_i, \lambda_i(r_i, \varepsilon_i)) = \delta_i(q_i, \lambda_i(\bar{r}_i, \varepsilon_i))$, $1 \leq i \leq k \implies$
      $(p_1\Delta(z_1,z_2,r), p_2\Delta(z_1,z_2,r)) = (p_1\Delta(z_1,z_2,\bar{r}), p_2\Delta(z_1,z_2,\bar{r}))$

Bemerkung: Die Maschine M arbeitet autonom, beginnend im Zustand (z^o, r^o).

Folgendes Schema veranschaulicht ihre Arbeitsweise:



Ein "Arbeitstakt" umfaßt die Ausgabe der $\varepsilon_1,\ldots,\varepsilon_k$, das Einlesen von $\lambda_1(r_1,\varepsilon_1),\ldots,\lambda_k(r_k,\varepsilon_k)$, die Zustandsübergänge der $P_1,\ldots,P_k$ und das Umsortieren von Programmen im Speicher. Dem Automaten M ist damit die Menge IN der natürlichen Zahlen als "Arbeitszeitskala" zugeordnet.

Für eine Paging-Maschine M definieren wir die Relation $S_M \subseteq 2^{[k]} \times 2^{[k]}$ wie folgt:

$$S_M = \{(A,A') \mid A,A' \in 2^{[k]}, \exists (z_1,z_2,r) \text{ mit } \Delta(z_1,z_2,r) = (z_1',z_2',r') \text{ und } A = S(z_1), A' = S(z_1')\}$$

S_M gibt die möglichen, auf eine Belegung folgenden Belegungen des Speichers mit Programmen (nicht Seiten!) wider. (S_M heißt Programmstrategie)

LEMMA 1: Es sei M eine Paging-Maschine. Unter Beibehaltung von Δ werde M so abgeändert, daß jedes Π_i, $1 \le i \le k$, ersetzt wird durch ein Π_i' über der selben Seitenmenge. Man erhält auf diese Weise wieder eine Paging-Maschine M'.

Der Beweis ergibt sich aus Bedingung ii)c) der Definition 3.

KOROLLAR: Für die Maschine M' aus Lemma 1 gilt: $S_M = S_{M'}$.

Mit Hilfe der Speicherbelegungsfunktion $\tau: Z \times R \to 2^{\bigcup_{i=1}^{k} N_i}$,

$\tau(z_1,z_2,r) = \bigcup_{i \in S(z_1)} p_i(z_1)$, definieren wir außerdem noch zwei "Kostenfunktionen" für M. ($\tau_i(\not\S) = \emptyset$!)

Bezeichnungen: 1. $v_M: Z \times R \to IN$, $v_M(s) = |\tau\Delta(s) - \tau(s)| + |\tau(s) - \tau\Delta(s)|$ heißt Seitenverkehr von M beim Zustand s.

2. $d_M: Z \times R \to IN$, $d_M(s) = m - |\tau(s)|$ heißt Speicherdefekt von M beim Zustand s ($s = (z_1,z_2,r)$!)

LEMMA 2: Es sei M eine Paging-Maschine. Dann gilt für $s = (z_1, z_2, r)$:

$$v_M(s) = \sum_{i \in S(z_1)} k_i(q_i, \lambda_i(r_i, \varepsilon_i)) + \sum_{i \in S(z_1') - S(z_1)} |\tau_i \delta_i(q_i, \lambda_i(r_i, \varepsilon_i))| + \sum_{i \in S(z_1') - S(z_1)} |\tau_i q_i| \quad , (\Delta(s) = (z_1', z_2', r')!)$$

Der Beweis ergibt sich durch Nachrechnen unter Beachtung der Definitionen.

Ist insbesondere $s = s^t = \Delta^t(s^o)$, so schreiben wir auch: $v_M(t) = v_M(s^t)$ und setzen: $V_M(t) = \sum_{i=0}^{t} v_M(i)$ (Seitenverkehr bis t)

2. Eine Aussage über Demand-Maschinen

DEFINITION 4: Eine Paging-Strategie $P = (Q, \ldots)$ heißt Demand-Strategie, falls: $\forall q \in Q, \forall x \in N: \tau\delta(q,x) \subseteq \tau q \cup \{x\}$.
Eine Paging-Maschine M heißt Demand-Maschine, falls sämtliche P_i, $1 \leq i \leq k$, Demand-Strategien sind.

In |1| wurde gezeigt, daß es zu jeder Paging-Strategie eine Demand-Strategie gibt, welche bis zu jedem Zeitpunkt einen geringeren Seitenverkehr liefert. Wir geben dieses Resultat als Lemma 4 unten in leicht verschärfter Form wieder. Im Rahmen des hier betrachteten Modells entspricht es dem Fall $k = 1$. Für $k > 1$ gilt es im allgemeinen jedoch nicht, wenn man die Programmstrategie beibehält. (Es ist nicht schwer, etwa im Fall $k = 2$ ein Gegenbeispiel anzugeben.) Im folgenden soll gezeigt werden, daß unter einer stark einschränkenden Bedingung für die Programmstrategie der Übergang zu Demand-Strategien auch im vorliegenden Modell einen geringeren Seitenverkehr ergibt, was dann allerdings zu einer schlechteren Speicherauslastung führt.

Ist P eine Paging-Strategie, so gibt es Abbildungen $I_p, O_p: Q \times (N \cup \{\not\in\}) \to 2^N$ mit folgenden Eigenschaften:

i) $\tau\delta(q,x) = [\tau q - O_p(q,x)] \cup I_p(q,x)$
ii) $O_p(q,x) \subseteq \tau q - \{x\}$
iii) $I_p(q,x) \cap \tau q = \emptyset$
iv) $x \in \tau q \cup I_p(q,x)$

LEMMA 3: Eine Paging-Strategie P ist genau dann Demand-Strategie, Falls für alle $(q,x) \in Q \times (N \cup \{\not\in\})$ gilt: $I_p(q,x) = \{x\} - \tau q$.

LEMMA 4: Es sei P eine Paging-Strategie über N. Dann gibt es eine Demand-Strategie P' über N mit folgenden Eigenschaften:
$\forall w = x_1 x_2 \ldots \in (N \cup \{\not\in\})^\infty, T \in \mathbb{N}$:
i) $\sum_{0}^{T} k_{p'}(q'^t, x_{t+1}) \leq \sum_{0}^{T} k_p(q^t, x_{t+1})$ und

ii) $|\tau q'^t| \leq |\tau q^t|$, für alle $t \in IN$. ($q^t = \delta(q^o, x_1 \ldots x_t)$!)

BEWEIS: P' ist gegeben durch: $Q' = Q \times 2^N$, $q'^o = (q^o, \emptyset)$,
$\delta'[(q,E),x] = [\delta(q,x), E \cup (I_p(q,x) - \{x\})]$,
$\tau'(q,E) = \tau q - E$. Es gilt: $O_{p'}[(q,E),x] = O_p(q,x) - E$ und
$I_{p'}[(q,E),x] = \{x\} - \tau'(q,E)$. Hieraus und aus $k_p(q,x) = |I_p(q,x)| + |O_p(q,x)|$ folgen sofort die Behauptungen.

Zur Formulierung einer entsprechenden Aussage über Paging-Maschinen benötigen wir eine letzte Bezeichnung.

Bezeichnung: Eine Paging-Maschine M heißt extern bestimmt, falls ihre Programmstrategie S_M eine Abbildung ist.

Letzteres kann zum Beispiel dann der Fall sein, wenn die Reihenfolge, in der die Programme bearbeitet werden, durch eine im voraus festgelegte Prioritätenregel gegeben ist.

SATZ: Es sei M eine extern bestimmte Paging-Maschine über $\Pi_1, \ldots, \Pi_k$ und $P_1, \ldots, P_k$. Dann gibt es eine Demand-Maschine M' über $\Pi_1, \ldots, \Pi_k$ und $P'_1, \ldots, P'_k$ mit $S_{M'} = S_M$, sodaß für alle $t \in IN$ gilt:
i) $V_{M'}(t) \leq V_M(t)$ und ii) $d_{M'}(t) \geq d_M(t)$

Der Beweis ergibt sich aus Lemma 2 in Verbindung mit Lemma 4. Dabei ist zu beachten, daß zu jedem Zeitpunkt t gilt: $S(z_1(t)) = S(z'_1(t))$, wobei $(z_1(t), \ldots)$ (bzw. $(z'_1(t), \ldots)$) der Zustand der Maschine M (bzw. M') zur Zeit t ist. Es ist klar, daß M' schneller arbeitet als M, wenn man jedem Arbeitstakt einer Paging-Maschine eine zeitliche Länge zuordnet, die proportional ist der Anzahl der während des Arbeitstaktes rangierten Seiten.

Im folgenden geben wir noch eine notwendige Bedingung dafür, daß eine Maschine M extern bestimmt ist. Hierzu setzen wir o.B.d.A. voraus, daß es für alle $i = 1, \ldots, k$ Zustände $q_i \in Q_i$ gibt mit $|\tau_i q_i| = 1$ (1: s.o.!)

LEMMA 5: M sei eine extern bestimmte Paging-Maschine, und es sei $k > \frac{m}{I}$. Ist dann $(A,A') \in S_M$ und $|A| > \frac{m}{I}$, so gilt: $|A'| < |A|$.

Beweisskizze: Wir nehmen an: $|A'| \geq |A|$, für ein $(A,A') \in S_M$. Mann kann dann - wie in Lemma 1 - eine Maschine M' so konstruieren, daß zu einem Zeitpunkt t gilt: $\sum_{i \in A} |\tau_i p_i(z'_1(t))| \leq m$, aber $\sum_{i \in A'} |\tau_i p_i(z'_1(t+1))| > m$,
im Widerspruch zu Lemma 1.

KOROLLAR: Mit den Voraussetzungen des obigen Lemmas gilt: $|S(z_1(t))| \leq \frac{m}{I}$ für fast alle $t \in IN$.

Das heißt: eine extern bestimmte Maschine bearbeitet während fast jeden Taktes höchstens $\frac{m}{l}$ Programme.

3. Eine allgemeinere Fragestellung. Lemma 4 und der nachfolgende Satz besagen inhaltlich, daß die bezüglich des Seitenverkehrs optimalen Paging-Strategien in der Klasse der Demand-Strategien zu suchen sind. Wir wollen diesen Typ von Resultat in einen etwas allgemeineren Rahmen einordnen:

Es sei $\mathcal{A}$ eine Klasse von endlichen Automaten mit Eingabemenge I und Zustandsmengen Q(a), a $\varepsilon \mathcal{A}$. Gegeben seien ferner Funktionen $k_a: Q(a) \times I \to IN$ (z.B.), die einem Zustandsübergang $(q,x) \to \delta_a(q,x)$ die "Kosten" $k_a(q,x)$ zuordnen. k_a, a $\varepsilon \mathcal{A}$, kann ohne Mühe auf $Q(a) \times I^*$ erweitert werden (s.o.!).

1. Bezüglich einer gegebenen Klasse $\mathcal{E}$ von Eingabefolgen charakterisiere man diejenigen a' $\varepsilon \mathcal{A}$, die eine beliebige Folge aus $\mathcal{E}$ zu geringeren Kosten verarbeiten als ein gegebener Automat a $\varepsilon \mathcal{A}$.

Im Zusammenhang mit Paging-Strategien gibt es zu dieser Fragestellung einige Lösungsansätze: In |1| und |2| werden Strategien angegeben, die optimal sind bezüglich jeder Folge aus I^*; in |3| wird die Klasse $\mathcal{E}$ statistisch definiert und die Optimalität einer Strategie für $\mathcal{E}$ bewiesen. Diese Strategien sind jedoch nicht durch endliche Automaten im Sinne von Definition 2 darstellbar.

Umgekehrt kann man nun folgende Fragestellung formulieren:

2. Gegeben sei ein Automat a $\varepsilon \mathcal{A}$ und eine Äquivalenzrelation auf der Menge der Eingabefolgen. Man charakterisiere diejenigen Eingabefolgen w' (bzw. die sie "erzeugenden" Programme) aus einer Äquivalenzklasse [w], die den Automaten a bei geringeren Kosten durchlaufen als eine gegebene Folge w.

Im Zusammenhang mit Paging-Strategien liegen hierzu - soweit dem Verfasser dieser Note bekannt - bisher nur experimentelle Studien vor. (|4|)

Literatur

|1| Mattson R.L., u.a.: Evaluation Techniques for Storage Hierarchies, IBM Systems Journal, Vol. 9.2, 1970

|2| Belady L.A.: A Study of Replacement Algorithms for a Virtual Storage Computer, IBM Systems Journal, Vol. 5.2, 1966

|3| Aho A.V., Denning P.J., Ullman J.D.: Principles of Optimal Page Replacement, JACM, Vol. 18.1, 1971

|4| Hatfield D.J., Gerald J.: Program Restructuring for Virtual Memory, IBM Systems Journal, Vol. 10,3, 1971