Funktionen, die von pushdown-Automaten berechnet werden

Von G. WECHSUNG

In der Literatur über formale Sprachen nehmen die pushdown-Automaten (PDA) als Akzeptoren kontextfreier Mengen einen hervorragenden Platz ein (vgl. z.B. [2]). Was die von PDA mit Ausgabe (pushdown transducer, [1], [3]) berechneten Wortfunktionen betrifft, so sind zwar umfangreiche Ergebnisse über das Verhalten formaler Sprachen verschiedener Typen bei Anwendung solcher Abbildungen bekannt (vgl. z.B. [3]), aber bisher fehlt eine automatenunabhängige Charakterisierung dieser Wortfunktionen. Diese Aufgabe wird in der vorliegenden Arbeit für den Fall deterministischer PDA, die überall definierte Funktionen berechnen, in Angriff genommen. In § 1 werden die nötigen Grundbegriffe und zwei verschiedene Berechnungsbegriffe für PDA eingeführt. § 2 gibt eine Charakterisierung der im ersten Sinne PDA-berechenbaren Wortfunktionen (die Berechnung ist beendet, wenn das Eingabewort abgearbeitet ist). Es ergibt sich eine Klasse sequentieller Funktionen, die in gewisser Weise aus sequentiellen Funktionen endlichen Gewichts stückweise zusammengesetzt werden können. Dabei entstehen auch Funktionen *unendlichen Gewichts*. Die genaue Beschreibung dieser Klasse ist zwar etwas aufwendig, aber begrifflich durchsichtig.

Für die Berechnungen im 2. Sinne (die Berechnung ist beendet, wenn Eingabewort und Speicher abgearbeitet bzw. geleert sind) reichen die sequentiellen Funktionen nicht mehr aus. Es erweist sich aber eine in § 3 beschriebene Verallgemeinerung der in [6] eingeführten quasisequentiellen Funktionen als geeignet. § 4 ist den „*treu ausspeichernden*" PDA gewidmet, wobei der Spezialfall der längentreuen Wortfunktionen eingangs gesondert betrachtet wird. Die Klasse der von treu ausspeichernden PDA im zweiten Sinne berechneten Funktionen ergibt sich als wohldefinierte Klasse quasisequentieller Wortfunktionen (Satz 6′ bzw. Satz 6 für den Spezialfall der längentreuen Funktionen). Diese Funktionen haben die Gestalt $\varphi \circ s$, wobei φ eine sequentielle Funktion einer bestimmten Art und s eine sogenannte quasisequentielle Wortpermutation ist und φ und s eine bestimmte Verträglichkeitsbedingung erfüllen müssen. Die größere Leistungsfähigkeit der PDA gegenüber den endlichen Automaten kommt hier dadurch zum Ausdruck, daß φ auch unendliches Gewicht haben kann, wobei freilich die Klasse der zugelassenen φ verhältnismäßig begrenzt ist. Schließlich wird in § 5 auf die Voraussetzung des treuen Ausspeicherns verzichtet. Die im 2. Sinne berechenbaren Wortfunktionen haben zwar auch hier noch die Gestalt $\varphi \circ s$, jedoch braucht φ nicht mehr sequentiell, sondern nur noch „fastsequentiell" zu sein. Die Klasse der zugelassenen φ wird genau beschrieben (Satz 6″).

§ 1 Pushdown — Automaten mit Ausgabe

Wir interessieren uns hier nur für die folgende deterministische Variante des in [3] definierten pushdown transducers, die durch die Definitionen 1 und 2 präzisiert wird. In jedem Takt führt der Automat in Abhängigkeit vom inneren Zustand, vom gelesenen Eingabebuchstaben und vom letzten Kellerbuchstaben in eindeutig bestimmter Weise folgende Operationen durch: Er ändert den Zustand, ersetzt den gelesenen Eingabebuchstaben durch #, entscheidet, ob das Eingabeband weitergerückt werden soll und führt die Verrückung gegebenenfalls durch, ändert den Speicherinhalt und gibt ein Ausgabewort aus.

Definition 1. $P=[Z, X, Y, K, \mu, z_0]$ heißt deterministischer pushdown-Automat (PDA) oder Kellerautomat $=_{Df}$

1. X ist eine nichtleere endliche Menge (das Eingabealphabet). $\varnothing$, $\# \notin X$. $\varnothing$ ist ein Symbol zur Bezeichnung leerer Zellen auf den betrachteten Bändern.

2. Y ist eine nichtleere endliche Menge (das Ausgabealphabet).

3. Z ist eine endliche Menge (die Menge der Zustände), und $z_0 \in Z$ heißt Anfangszustand von P.

4. K ist eine endliche Menge (das Kelleralphabet). K enthält ein ausgezeichnetes Symbol Δ.

5. μ ist eine eindeutige Abbildung von $Z \times K \times (X \cup \{\#, \varnothing\})$ in $Z \times K^* \times Y^* \times \times \{0, 1\}$. (Für eine beliebige Menge M bezeichnen wir mit M^* die von M erzeugte freie Worthalbgruppe, deren Einselement (leeres Wort) immer e heißt.) μ hat die folgenden Eigenschaften: Ist $\mu(z, k, x)=(z', p, q, \varepsilon)$, so ist $p=\Delta p'$ mit $p' \in (K \setminus \{\Delta\})^*$, falls $k=\Delta$, und es ist $p \in (K \setminus \{\Delta\})^*$, falls $k \neq \Delta$. (Δ ist also ein Markierungszeichen, das den Anfang auf dem Kellerband angibt, nie verändert wird und nie außerhalb des Speicherwortanfangs vorkommen soll.)

Die Arbeitsweise des Automaten wird wie üblich durch Konfigurationen beschrieben.

Definition 2

(a) $[u, v, w]$ heißt Konfiguration von $P=_{Df}$

$$1)\ u \in Z\{\#, e\}X^*,$$

$$2)\ v \in \Delta(K \setminus \{\Delta\})^*,$$

$$3)\ w \in Y^*.$$

(b) $[u, v, w]$ heißt Anfangskonfiguration von $P=_{Df}$ $u \in z_0 X^* \wedge v=\Delta \wedge w=e$.
(c) $[u', v', w']$ heißt unmittelbare Folgekonfiguration von $[u, v, w]$ bezüglich P

$$([u, v, w] \vdash_P [u', v', w'])=_{Df} \quad \text{Falls } u=zx_1 \ldots x_n,$$

wobei

$$x_1 \in \{\#\} \cup X, \quad x_2, \ldots, x_n \in X, \quad n \geqq 0,$$

(ist $n=0$, so setzen wir $u=z$) und $v=\Delta k_1 \dots k_m$, wobei $k_1, \dots, k_m \in K \setminus \{\Delta\}$, $m \geqq 0$, (ist $m=0$, so setzen wir $v=\Delta$ und $k_m=\Delta$) und $\mu(z, k_m, x_1)=(z', p, q, \varepsilon)$, so ist

$$u' = \begin{cases} z' x_2 \dots x_n, & \text{falls } \varepsilon=1 \wedge n>1 \\ z', & \text{falls } (\varepsilon=1 \wedge n=1) \vee n=0 \\ z' \# x_2 \dots x_n, & \text{falls } \varepsilon=0 \wedge n \geqq 1, \end{cases}$$

$v'=\Delta k_1 \dots k_{m-1} p$ (p kann leer sein), $w'=wq$ (q kann leer sein).

(d) $[u', v', w']$ heißt Folgekonfiguration von $[u, v, w]$ bezüglich $P([u, v, w] \models_P [u', v', w'])=_{Df}$ Es existieren Konfigurationen $[u_1, v_1, w_1], \dots, [u_n, v_n, w_n]$ mit

$$[u_1, v_1, w_1]=[u, v, w] \wedge [u_n, v_n, w_n]=[u', v', w'] \wedge$$

$$\wedge [u_1, v_1, w_1] \vdash_P [u_2, v_2, w_2] \vdash_P \dots \vdash_P [u_n, v_n, w_n].$$

Was die Verwendung von PDA als Berechnungsalgorithmen von Wortfunktionen, d. h. von eindeutigen Abbildungen von X^* in Y^*, betrifft, so bieten sich zwei Möglichkeiten an. Erstens kann in Anlehnung an [3] die folgende Berechnungskonzeption gewählt werden.

Definition 3. Die Wortfunktion $f: X^* \to Y^*$ heißt PDA-berechenbar im ersten Sinne $=_{Df}$ Es existiert ein PDA $P=[Z, X, Y, K, \mu, z_0]$ mit

$$\forall w \exists z \exists p (w \in X^* \to z \in Z \wedge p \in K^* \wedge [z_0 w, \Delta, e] \models_P [z, p, f(w)])$$

$$\wedge \sim \exists z' \exists p' \exists v ([z_0 w, \Delta, e] \models_P [z', p', v] \vdash_P [z, p, f(w)]).$$

Hierbei kann der Fall eintreten, daß ein gewisser Anteil an Informationen über w in p und nicht in $f(w)$ eingeht. Daher liegt die Betrachtung einer zweiten Art der Berechnung nahe, die erst dann als beendet angesehen wird, wenn das Wort w gelesen und der Keller vollständig geleert ist.

Definition 4. Die Wortfunktion $f: X^* \to Y^*$ heißt PDA-berechenbar im 2 Sinne $=_{Df}$ Es existiert ein PDA $P=[Z, X, Y, K, \mu, z_0]$ mit

$$\forall w \exists z (w \in X^* \to z \in Z \wedge [z_0 w, \Delta, e] \models_P [z, \Delta, f(w)]).$$

Beispiel. Es sei P ein PDA mit $Z=\{z_0\}$, $X=Y=K$ und

$$\mu(z_0, k, x)=(z_0, kx, e, 1)$$

$$\mu(z_0, k, \varphi)=(z_0, e, k, 1), \text{ falls } k \neq \Delta.$$

Die von P im ersten Sinne berechnete Funktion ist die Konstante e, während die im zweiten Sinne berechnete Funktion f_2 die Wortinversion

$$f_2(x_1 \dots x_n)=x_n \dots x_1$$

ist, die wir in Zukunft auch mit $(x_1 \dots x_n)^{-1}$ bezeichnen wollen.

Die Berechnungen im ersten Sinne werden offenbar nur sequentielle Funktionen realisiert, während die Berechnungen im zweiten Sinne über die Klasse der sequentiellen Funktionen hinausführen, was schon durch das vorangehende Beispiel deutlich wird. Dieses Ergebnis steht dem bekannten Sachverhalt gegenüber ([2], Kapitel

2.5), daß die beiden Entscheidungsbegriffe für PDA, die unseren Definitionen 3 und 4 entsprechen, völlig äquivalent sind.

Für die beabsichtigte Analyse des Berechnungsverhaltens von PDA ist es bequem, nur solche PDA zu betrachten, die in jedem Takt den letzten gespeicherten Buchstaben entweder löschen oder ihn als Anfangsbuchstaben des zu speichernden Teilwortes übernehmen. Solche PDA nennen wir *einfach*.

Definition 5. Der PDA $P=[Z, X, Y, K, \mu, z_0]$ heißt einfach $=_{Df}$ $\mu(z, k, x)=$ $=(z', p, q, \varepsilon)\wedge p\neq e\rightarrow k\sqsubset p$. (Hierbei bezeichnet $\sqsubset$ die übliche Anfangswortrelation.)

In den §§ 2, 4 und 5 dieser Arbeit wollen wir grundsätzlich nur noch einfache PDA betrachten, ohne das jedesmal ausdrücklich zu erwähnen. Daß hierin keine wesentliche Beschränkung der Allgemeinheit liegt, zeigen die beiden folgenden Lemmata.

Lemma 1. Zu jedem PDA $P=[Z, X, Y, K, \mu, z_0]$ gibt es einen einfachen PDA $P_1=[Z_1, X, Y, K, \mu_1, z_0^{(1)}]$, der im ersten Sinne die gleiche Wortfunktion berechnet wie P.

Beweis. Wir setzen $Z_1=_{Df}Z\times K$, $z_0^{(1)}=[z_0,\Delta]$ und für jedes $k'\in K$

$$\mu_1([z, k], k', x)=$$

$$=\begin{cases}([z', k''], k', p, q, \varepsilon), & \text{falls } \mu(z, k, x)=(z', pk'', q, \varepsilon)\wedge k''\in K\setminus\{\Delta\}\\ ([z', k'], e, q, \varepsilon), & \text{falls } \mu(z, k, x)=(z', e, q, \varepsilon)\wedge k'\neq\Delta\\ ([z', \Delta], \Delta, q, \varepsilon), & \text{falls } \mu(z, k, x)=(z', e, q, \varepsilon)\wedge k'=\Delta.\end{cases}$$

Wie leicht zu sehen ist, berechnet P_1 im ersten Sinne die gleiche Wortfunktion wie P.

Lemma 2. Berechnet der PDA P im zweiten Sinne die auf ganz X^* definierte Funktion f, so berechnet der PDA P_1, der dem P durch die Konstruktion im Beweis von Lemma 1 zugeordnet werden kann, eine auf ganz X^* definierte Funktion f_1, die mit f in folgender Beziehung steht. Es gibt eine Zerlegung von X^* in endlich viele kontextfreie Mengen $C_1, \ldots, C_s$, und es gibt Wörter $r_1, \ldots, r_s\in Y^*$, so daß

$$f(w)=f_1(w)r_\sigma, \text{ falls } w\in C_\sigma.$$

Beweis. Es sei

$$[z_0w, \Delta, e]\models_P[z_1, p_1k_1, q_1]\vdash_P[z_2, p_2k_2, q_2]\vdash_P\ldots\vdash_P[z_n, \Delta, f(w)].$$

Hierbei sei $[z_1, p_1k_1, q_1]$ die erste Konfiguration der durch $[z_0w, \Delta, e]$ bestimmten Folge, deren erste Komponente nur aus einem Element aus Z besteht. Ferner gelte $k_1, k_2, \ldots, k_{n-1}\in K$. Dieser Konfigurationenfolge entspricht für P_1 die Folge

$$[[z_0, \Delta]w, \Delta, e]\models_{P_1}[[z_1, k_1], p_1, q_1]\vdash_{P_1}[[z_2, k_2], p_2, q_2]\vdash_{P_1}\ldots\vdash_{P_1}[[z_n, \Delta], \Delta, f(w)].$$

Wenn der Fall eintritt, daß für irgendein $v<n$ erstmalig $p_v=\Delta$ ist, so hat die diesem v entsprechende Konfiguration der zweiten Folge die Form $[[z_v, k_v], \Delta, q_v,]$ womit laut Definition 4 die Berechnung von $f_1(w)$ beendet ist. Das Wort r_v, das

$f_1(w)$ noch hinzugefügt werden muß, damit $f(w)$ entsteht, hängt offensichtlich nur von z_ν und k_ν ab. Daher gilt für alle w, die in der Menge

$$C_\nu =_{Df} \{w\colon \exists q([[z_0, \Delta]w, \Delta, e] \models_{P_1} [[z_\nu, k_\nu], \Delta, q])\}$$

liegen, die Beziehung

$$f(w) = f_1(w) r_\nu.$$

Wenn kzM die Kardinalzahl der Menge M bedeutet, gibt es höchstens $kzX \cdot kzK$ verschiedene derartige C_ν, die nach [3] alle kontextfrei sind. Da wegen der Voraussetzung, daß f auf ganz X^* definiert sein soll, jedes $w \in X^*$ in genau einer dieser Mengen C_ν liegen muß, ist das Lemma bewiesen.

§ 2 Funktionen, die im ersten Sinne von PDA berechnet werden

Es kann vorkommen, daß eine von einem PDA im ersten Sinne berechnete Wortfunktion nicht für alle $w \in X^*$ definiert ist. Das ist genau dann der Fall, wenn der PDA während der Abarbeitung von w in eine Situation kommt, in der er # liest und von der aus er nie wieder einen Befehl mit $\varepsilon = 1$ erreicht. Wir wollen hier nur auf ganz X^* definierte Funktionen untersuchen und beweisen dazu

Lemma 3. Zu jedem PDA P, der nach jeder Situation $(z, k, \#)$ wieder einen Befehl mit $\varepsilon = 1$ erreicht, gibt es einen PDA P', der im ersten und zweiten Sinne die gleiche Funktion berechnet wie P und der folgende Eigenschaft hat: Ist

$$\mu'(z, k, \#) = (z', p, q, \varepsilon), \tag{1}$$

so ist $p = e \wedge \varepsilon = 0$ oder $p = k \wedge \varepsilon = 1$.

Beweis. Es sei

$$\mu(z_1, k_1, x) = (z, k_1 p_1 k, q_1, 0), \tag{2}$$

so daß sich P im nächsten Takt in der Situation $(z, k, \#)$ befindet. Wir geben ein System von Reduktionsregeln an, mit denen P' aus P konstruiert werden kann.

1. *Fall.* Es gilt

$$\mu(z, k, \#) = (z', p, q, 0) \tag{3}$$

mit $p \neq e$.

Dann ersetzen wir für jedes Quintupel (z_1, k_1, x, p_1, q_1), das (2) erfüllt, die Befehle (2) und (3) durch

$$\mu(z_1, k_1, x) = (z', k_1 p_1 p, q_1 q, 0).$$

2. *Fall.* Es gilt

$$\mu(z, k, \#) = (z', p, q, 1) \tag{4}$$

mit $p = e$.

In diesem Falle wird ein neuer Zustand z'' eingeführt, und wir ersetzen (2) und (4) durch

$$\mu(z, k, \#) = (z'', e, q, 0) \quad \text{und}$$

$$\mu(z'', l, \#) = (z', l, e, 1) \quad \text{für alle } l \in K.$$

3. *Fall.* Es gilt

$$\mu(z, k, \#)=(z', kp', q, 1) \tag{5}$$

mit $p' \neq e$.

Dann ersetzen wir (2) und (5) durch

$$\mu(z_1, k_1, x)=(z', k_1 p_1 k p', q_1 q, 1)$$

für jedes (2) erfüllende Quintupel (z_1, k_1, x, p_1, q_1).

Nach Voraussetzung über den Definitionsbereich von f treten, ausgehend von $(z, k, \#)$ keine Zyklen auf. Daher können durch endlich viele Anwendungen der vorstehenden Ersetzungsregeln unter eventueller Vergrößerung der Zustandmenge alle Befehle eliminiert werden, die nicht der Bedingung (1) genügen. Den entstehenden PDA nennen wir P'. Es ist klar, daß bei Anwendung dieser Reduktionen das Berechnungsverhalten von P im ersten und zweiten Sinne nicht beeinflußt wird.

Der folgende Hilfssatz zeigt, daß die Arbeitsweise der PDA in gewisser Weise normiert werden kann. Für die geplante Analyse ist es angenehm, diese Normiertheit voraussetzen zu können.

Lemma 4. Zu jedem PDA $P=[Z, X, Y, K, \mu, z_0]$ existiert ein PDA $P'=$ $=[Z', X, Y, K, \mu', z_0]$, der im ersten und zweiten Sinne die gleiche Funktion berechnet wie P und folgende Bedingung erfüllt

$$\mu(z, k, x)=(z', p, q, \varepsilon) \wedge x \in X \rightarrow p \neq e.$$

Beweis. Man setzt $\mu'=\mu$ für alle Tripel (z, k, x), die der Bedingung des Lemmas genügen. Ist $\mu(z, k, x)=(z', e, q, 0)$ für $x \in X$, so führt man einen neuen Zustand z_1 ein und ersetzt diesen Befehl durch

$$\mu'(z, k, x)=(z_1, k, q, 0),$$

$$\mu'(z_1, k, \#)=(z', e, e, 0).$$

Ist $\mu(z, k, x)=(z', e, q, 1)$, so führt man zwei neue Zustände z_2 und z_3 ein und ersetzt diesen Befehl durch

$$\mu'(z, k, x)=(z_2, k, q, 0),$$

$$\mu'(z_2, k, \#)=(z_3, e, e, 0),$$

$$\mu'(z_3, k', \#)=(z', k', e, 1),$$

für alle $k' \in K$. Der Zwischenzustand z_3 mußte eingeführt werden, damit der neue Automat der Bedingung von Lemma 3 genügt. Z' unterscheidet sich von Z um die hierbei neu hinzugefügten Zustände. Daß P' bezüglich beider Berechnungsarten zu P äquivalent ist, ist trivial.

Von jetzt an betrachten wir grundsätzlich nur solche PDA, die der Bedingung (1) von Lemma 3 genügen und gemäß Lemma 4 normiert sind.

Gegeben sei der PDA $P=[Z, X, Y, K, \mu, z_0]$, und es sei o.B.d.A. $Y \cap K=\varnothing$. Um sein Berechnungsverhalten zu untersuchen, ordnen wir P zwei miteinander gekoppelte endliche partielle Automaten $Q_1=[U, X, Y \cup K, f_1, g]$ und $Q_2=$ $=[V, K, Y, f_2, h]$ zu. In diesen Quintupeln bedeuten die Komponenten der Reihe nach die Zustandsmenge, Eingabemenge, Ausgabemenge, Überführungsfunktion und Ausgabefunktion ([4]).

Das Automatenpaar Q_1, Q_2 soll die Arbeit von P beschreiben. Solange P einen Teil des Eingabewortes w abarbeitet (d.h. Befehle mit $\varepsilon=1$ anwendet), simuliert Q_1 die Tätigkeit von P. Sobald aber P die Abarbeitung von w zum Zwecke der Ausspeicherung unterbricht (d.h. Befehle mit $\varepsilon=0$ auftreten), erreicht Q_1 einen Zustand aus V, und der Ausspeicherungsprozeß wird nun von Q_2 simuliert, wobei als Eingabe das bis dahin von P gespeicherte invertierte Wort dient. Ist die Ausspeicherungsperiode vorüber, so wird ein Zustand aus U erreicht, und Q_1 tritt wieder in Aktion.

Im einzelnen sind Q_1 und Q_2 folgendermaßen definiert.

$$U = U' \cup U'' \text{ mit}$$

$$U' =_{Df} \{[z_0, \Delta]\} \cup \{[z, k] : \exists z' \exists k' \exists x \exists p \exists q \, (\mu(z', k', x) = (z, pk, q, 1))\},$$

$$U'' =_{Df} \{z : \exists z' \exists k \exists x \exists p \exists q \, ([z', k] \in U' \wedge \mu(z', k, x) = (z, kp, q, 0))\}.$$

Für $[z, k] \in U'$ sind die Überführungsfunktion f_1 und die Ausgabefunktion g definiert durch

$$f_1([z, k], x) =_{\mathrm{Df}} \begin{cases} [z', k'], & \text{falls } \mu(z, k, x) = (z', pk', q, 1) \\ z', & \text{falls } \mu(z, k, x) = (z', kp, q, 0) \end{cases}$$

$$g([z, k], x) =_{Df} pq, \quad \text{falls} \quad \mu(z, k, x) = (z', kp, q, \varepsilon).$$

Auf U'' sind f_1 und g nicht definiert.

$$V_h = V' \cup V'' \text{ mit}$$

$$V' =_{Df} \{z : \exists z' \exists k \exists q \, (\mu(z', k, \#) = (z, e, q, 0))\} \cup U'',$$

$$V'' =_{Df} \{[z, k] : \exists z' \exists q \, (z' \in V' \wedge \mu(z', k, \#) = (z, k, q, 1))\}.$$

Für $z \in V'$ sind Überführungs- und Ausgabefunktion von Q_2 folgendermaßen definiert:

$$f_2(z, k) =_{\mathrm{Df}} \begin{cases} z', & \text{falls } \mu(z, k, \#) = (z', e, q, 0) \\ [z', k], & \text{falls } \mu(z, k, \#) = (z', k, q, 1). \end{cases}$$

$h(z, k) =_{\mathrm{Df}} q$, falls $\mu(z, k, \#) = (z', p, q, \varepsilon)$. Auf V'' werden f_2 und h nicht definiert. Wir vereinbaren noch, die auf X^* bzw. K^* erweiterten Ausgabefunktionen von Q_1 bzw. Q_2 ([4]) ebenfalls mit g bzw. h zu bezeichnen. Ferner nehmen wir zur Erleichterung der folgenden Untersuchungen die Mengen U'' und V'' in der Form $V'' = \{u_1, \ldots, u_n\}$, $U'' = \{v_1, \ldots, v_m\}$ an und setzen $u_0 =_{\mathrm{Df}} [z_0, \Delta]$.

Mit g_i (bzw. h_j) bezeichnen wir die von Q_1 im Zustand u_i $(i=0, \ldots, n)$ bzw. von Q_2 im Zustand v_j $(j=1, \ldots, m)$ berechneten Funktionen. Es seien δ_Y und δ_K die durch

$$\delta_Y(w) = \begin{cases} w, & \text{falls } w \in Y \\ e, & \text{falls } w \in K \end{cases}$$

$$\delta_K(w) = \begin{cases} e, & \text{falls } w \in Y \\ w, & \text{falls } w \in K \end{cases}$$

definierten Homomorphismen von $(Y \cup K)^*$ auf Y^* bzw. K^*. Damit gewinnen wir die für das weitere wichtigen Funktionen

$$d_i =_{\mathrm{Df}} g_i \circ \delta_Y \quad \text{und} \quad s_i =_{\mathrm{Df}} g_i \circ \delta_K.$$

(Wir bezeichnen mit fog die Funktion, die durch $\mathrm{fog}\,(x) = g(f(x))$ aus f und g entsteht.) Ferner setzen wir

$$M_i^j =_{\mathrm{Df}} \{p : p \in X^* \wedge f_1(u_i, p) = v_j\} \quad \text{für } i = 0, \ldots, n;\ j = 1, \ldots, m,$$
$$M_i =_{\mathrm{Df}} \{p : p \in X^* \wedge \forall q (q \sqsupset p \to f_1(u_i, q) \notin U'')\} \quad \text{für } i = 0, \ldots, n,$$
$$N_j^i =_{\mathrm{Df}} \{w : w \in K^* \wedge f_2(v_j, w) = u_i\} \quad \text{für } i = 0, \ldots, n;\ j = 1, \ldots, m.$$

Wir beschaffen uns jetzt eine Funktion s, die das Verhalten des Speichers von P in dem Sinne beschreibt, daß $s(p)$ den Inhalt des Speicherbandes nach Abarbeitung von p (einschließlich sich eventuell anschließender Ausspeicherungsoperationen) angibt. Der expliziten Darstellung von s stellen wir eine anschauliche Erläuterung voran. Bei Abarbeitung von p wächst s zunächst so lange an, bis ein erstes $v_{j_1} \in U''$ erreicht wird. Dies möge für das Anfangswort p_1 von p der Fall sein. Nun setzt eine Ausspeicherungsphase ein, in deren Verlauf der bisherige Speicherinhalt $s_0(p_1)$ um ein eindeutig bestimmtes Endwort q_1 verringert wird. Das dabei übrigbleibende Anfangsstück von $s_0(p_1)$ ist $s(p_1)$. Die Eingabe von q_1^{-1} auf v_{j_1} führt in Q_2 zu einem eindeutig bestimmten $u_{i_1} \in V''$, mit dem der Prozeß fortgesetzt wird, wenn p noch nicht abgearbeitet ist.

Mit der Schreibweise

$$w \setminus v =_{\mathrm{D}} \begin{cases} u, & \text{falls } w = uv \\ \text{nicht definiert} & \text{sonst} \end{cases}$$

können wir s explizit angeben:

$$\left.\begin{array}{l} s(p) = (((\ldots((s_0(p_1) \setminus q_1) s_{i_1}(p_2)) \setminus q_2 \ldots) s_{i_{t-1}}(p_t)) \setminus q_t) s_{i_t}(p_{t+1}), \\ \text{falls } p = p_1 \ldots p_{t+1} \text{ mit} \\ \exists j_1 \exists q_1 \exists i_1 (p_1 \in M_0^{j_1} \wedge q_1^{-1} \in N_{j_1}^{i_1}) \wedge \exists j_2 \exists q_2 \exists i_2 (p_2 \in M_{i_1}^{j_2} \wedge q_2^{-1} \in N_{j_2}^{i_2}) \wedge \ldots \\ \ldots \wedge p_{t+1} \in M_{i_t} \wedge \text{ die } q_i \text{ sind Endwörter derjenigen Wörter, von} \\ \text{denen sie abgezogen werden.} \\ \text{Für } t = 0 \text{ wollen wir die Darstellung so verstehen: } s(p) = s_0(p) \text{ und } p \in M_0. \end{array}\right\} \quad (6)$$

Man beachte daß alle vorkommenden Größen p_ν, q_ν, i_ν und j_ν durch p und P vollkommen eindeutig festgelegt sind. Wir führen zwei abkürzende Sprechweisen ein.

Definition 6. $w \alpha N_j^i =_{\mathrm{Df}} \exists w_1 (w_1 \sqsubseteq w \wedge w_1 \in N_j^i)$. Weil N_j^i total ungeordnet ist bezüglich $\sqsubseteq$, gibt es für $w \alpha N_j^i$ genau ein solches w_1. Deshalb hat folgende Definition einen Sinn.

Definition 7. Ist $w \alpha N_j^i$, so setzen wir $h_j(w) = h_j(w_1)$ mit

$$w_1 \sqsubseteq w \wedge w_1 \in N_j^i.$$

Damit sind wir in der Lage, die von P im ersten Sinne berechnete Funktion f folgendermaßen zu beschreiben.

$$\left.\begin{array}{l} f(p) = d_0(p_1)h_{j_1}(s(p_1)^{-1})d_{i_1}(p_2)h_{j_2}(s(p_1p_2)^{-1})\dots \\ \dots d_{i_{k-1}}(p_k)h_{j_k}(s(p_1\dots p_k)^{-1})d_{i_k}(p_{k+1}),\ \text{falls}\ p=p_1\dots p_{k+1}\wedge \\ \exists j_1\exists i_1(p_1\in M_0^{j_1}\wedge s(p_1)^{-1}\alpha N_{j_1}^{i_1})\wedge\exists j_2\exists i_2(p_2\in M_{i_1}^{j_2}\wedge s(p_1p_2)^{-1}\alpha N_{j_2}^{i_2})\wedge\dots\wedge p_{k+1}\in M_{i_k}. \\ \text{Für } k=0 \text{ wollen wir diese Darstellung so verstehen: } f(p)=d_0(p) \text{ und } p\in M_0. \end{array}\right\} (7)$$

Wir definieren nun automatenunabhängig eine Klasse von sequentiellen Wortfunktionen von X^* in Y^*, die sich genau als die Klasse derjenigen (auf ganz X^* definierten) Wortfunktionen erweisen wird, die von PDA im ersten Sinne berechnet werden.

Definition 8. $f\in P_1(X, Y)=_{\mathrm{Df}}$

1. Es gibt eine endliche Menge K, und es existieren reguläre Mengen M_i, $M_i^j\subseteq X^*$ $(i=0, \dots, n; j=1, \dots, m)$ und reguläre Mengen $N_j^i\subseteq K^*$ $(i=1, \dots, n;\ j=1, \dots, m)$ mit den Eigenschaften

(a) $\forall i\forall j\,(M_i\cap M_i^j=\varnothing)$

(b) $\forall i\forall j\forall j'\,(j\neq j'\to M_i^j\cap M_i^{j'}=\varnothing)$

(c) $\forall i\forall p\,(p\notin M_i\leftrightarrow\exists j\exists q(q\sqsubseteq p\wedge q\in M_i^j))$

(d) $\forall i\forall j$ ($\bigcup\limits_{\nu} M_i^{\nu}$ und $\bigcup\limits_{\mu} N_j^{\mu}$ sind bezüglich $\sqsubseteq$ total ungeordnet)

(e) $\forall j\forall i\forall i'\,(i\neq i'\to N_j^i\cap N_j^{i'}=\varnothing)$

(f) $\exists\Delta(\Delta\in K\wedge\forall i\forall w\forall j(w\Delta\alpha N_i^j))$.

2. Es gibt sequentielle Funktionen endlichen Gewichts

$$s_0, \dots, s_n : X^*\to K^*,$$

$$d_0, \dots, d_n : X^*\to Y^*,$$

$$h_1, \dots, h_m : K^*\to Y^*,$$

so daß f durch (7) festgelegt ist, wobei die dazu benötigte Funktion s sich nach (6) ergibt. Wir wollen kurz sagen, f sei durch $\{M_i, M_i^j, N_j^i, s_i, h_j\}_{n,m}$ bestimmt.

Es gilt der

Satz 1. Eine auf ganz X^* definierte Wortfunktion mit Werten in Y^* ist genau dann PDA-berechenbar im ersten Sinne, wenn sie zu $P_1(X, Y)$ gehört.

Beweis

1. Daß eine von einem PDA berechnete überall auf X^* definierte Funktion zu $P_1(X, Y)$ gehört, (wobei Y das Ausgabealphabet das berechnenden PDA ist), geht aus der bisherigen Analyse hervor.

2. Es sei $f\in P_1(X, Y)$.

Wir geben einen PDA P an, der f im ersten Sinne berechnet. Durch elementare automatentheoretische Konstruktionen kann man zunächst zwei endliche partielle Mealy-Automaten $Q_1=[U, X, Y, K, f_1, g]$ und $Q_2=[V, K, Y, f_2, h]$ (mit $Y\cap K=\varnothing$)

konstruieren, die folgende Eigenschaft haben. Es gibt eine Teilmenge $U''=\{v_1, \ldots, v_m\}\subseteq U\cap V$ und eine Teilmenge $V''=\{u_1, \ldots, u_n\}\subseteq U\cap V$ und ein $u_0\in U$, so daß gilt: Versieht man Q_1 mit dem Anfangszustand u_i $(i=0, \ldots, n)$, so akzeptiert dieser initiale Automat durch v_j $(j=1, \ldots, m)$ die Menge M_i^j, und er berechnet eine Funktion g_i, die aus den Funktionen d_i und s_i durch

$$g_i(x_1, \ldots, x_n)=_{\mathrm{Df}}\sigma_1\omega_1\sigma_2\omega_2\ldots\sigma_n\omega_n, \quad \text{falls} \quad d_i(x_1\ldots x_\nu)=d_i(x_1\ldots x_{\nu-1})\omega_\nu$$

und

$$s_i(x_1\ldots x_\nu)=s_i(x_1\ldots x_{\nu-1})\sigma_\nu$$

hervorgeht. Versieht man Q_2 mit dem Anfangszustand v_j $(j=1, \ldots, m)$, so akzeptiert dieser initiale Automat durch den Finalzustand u_i $(i=1, \ldots, n)$ die Menge N_j^i, und er berechnet die Funktion h_j.

Ausgehend von diesen beiden gekoppelten Automaten geben wir jetzt einen PDA $P=[Z, X, Y, K, \mu, u_0]$ an:

$$Z=_{\mathrm{Df}}U\cup V.$$

μ wird folgendermaßen definiert:

a) $v\in V\setminus V''$

Gilt $f_2(v, k)=v'$ und $h(v, k)=q$, so setzen wir

$$\mu(v, k, \#)=\begin{cases}(v', e, q, 0), & \text{falls } v'\notin V''\\ (v', k, q, 1), & \text{falls } v'\in V''.\end{cases}$$

b) $u\in U\setminus U''$

Gilt $f_1(u, x)=u'$ und $g(u, x)=pq$ mit $p\in K^*\wedge q\in Y^*$, so setzen wir für alle $k\in K$

$$\mu(u, k, x)=\begin{cases}(u', k, q, 0), & \text{falls } u'\in U''\\ (u', kp, q, 1), & \text{falls } u'\notin U''.\end{cases}$$

Der so entstandene PDA berechnet ersichtlich genau die gegebene Funktion f im ersten Sinne. Damit ist Satz 1 bewiesen.

Wir bemerken noch, daß $P_1(X, Y)$ sequentielle Funktionen unendlichen Gewichts enthält. Dazu geben wir folgendes Beispiel an:

$$f(a^{n_1}b^{m_1}\ldots a^{n_k}b^{m_k})=\begin{cases}ba^{n_1}b^{m_1}\ldots a^{n_k}b^{m_k-1}, & \text{falls } m_k\geqq 1\\ ba^{n_1}b^{m_1}\ldots a^{n_{k-1}}b^{m_{k-1}-1}, & \text{falls } m_k=0.\end{cases}$$

Diese Funktion wird durch den PDA $P=[\{z_0, z_1\}, \{a, b\}, \{a, b\}, \{\Delta, a'\}, \mu, z_0]$ berechnet, dessen μ durch die Tabelle

Z	K	X	Z	K	Y	ε
z_0	Δ	a	z_0	$\Delta a'$	e	1
z_0	Δ	b	z_0	Δ	b	1
z_0	a'	a	z_0	$a'a'$	e	1
z_0	a'	b	z_1	a'	b	0
z_1	a'	#	z_1	e	a	0
z_1	Δ	#	z_0	Δ	e	1

gegeben ist. f ist demnach eine Funktion aus $P_1(X, X)$ mit $X=\{a, b\}$. Wegen $f(a^n)=e$ und $f(a^nb)=ba^n$ folgt für den durch a^n bestimmten Zustand f_an von f die Beziehung $f_an(b)=ba^n$. Demnach gilt $f_an\neq f_am$ für $n\neq m$, womit gezeigt ist, daß f unendliches Gewicht hat.

§ 3 Quasisequentielle Funktionen

Für die Analyse des Berechnungsverhaltens von PDA im zweiten Sinne, die über die sequentiellen Funktionen hinausführt, stellen wir hier eine geeignete Funktionenklasse bereit, die durch Verallgemeinerung eines Ansatzes aus [6] gewonnen wird.

X sei ein beliebiges endliches Alphabet. Mit $|w|$ bezeichnen wir die Länge des Wortes $w\in X^*$.

Definition 9

(a) Es sei $\mathfrak{T}$ die Menge aller allgemein rekursiven Funktionen t von X^* in Nz mit der Eigenschaft

$$\forall w(w\in X^*\to 1\leqq t(w)\leqq|w|).$$

(b) Es sei $t\in\mathfrak{T}$. Wir setzen für beliebige Wörter p mit $|p|=|w|$ über beliebigem Alphabet

$$\pi_{t(w)}(p)=\xi_1\ldots\xi_{t(w)-1}\xi_{t(w)+1}\ldots\xi_n,\quad\text{falls}\quad p=\xi_1\ldots\xi_n.$$

(Die Definition ist so zu verstehen, daß im Falle $t(w)=1$ oder $=n$ der erste bzw. letzte Buchstabe von p gestrichen wird.)

Definition 10

(a) Eine längentreue Funktion f von X^* in Y^* (d.h. eine solche, für die gilt $\forall w(w\in X^*\to|f(w)|=|w|)$ heißt $\langle t, t'\rangle$-sequentiell $=_{Df}$

$$\forall w\big(w\in X^*\to f(\pi_{t(w)}(w))=\pi_{t'(w)}(f(w))\big).$$

(b) f heißt quasisequentiell $=_{Df}$ Es gibt $t, t'\in\mathfrak{T}$, so daß f $\langle t, t'\rangle$-sequentiell ist.

(c) $S_{tt'}$ bezeichne die Menge aller $\langle t, t'\rangle$-sequentiellen Funktionen (von X^* in Y^*).

Die Verallgemeinerung gegenüber [6] besteht im Verzicht auf die zusätzliche Eigenschaft an die $t\in\mathfrak{T}$, daß $(t(w)$ nur von der Länge von w abhängt. Obwohl durch Definition 10 eine größere Funktionenklasse festgelegt wird als durch die entsprechende Definition in [6], wollen wir die dort eingeführte Bezeichnung „quasisequentielle Funktionen" beibehalten und künftig im Sinne von Definition 10 verstehen.

Die sequentiellen Funktionen ([5], [4]) ergeben sich genau als $\langle l, l\rangle$-sequentielle Funktionen, wobei l die durch $l(w)=_{Df}|w|$ definierte Funktion ist. Künftig wird l immer in dieser Bedeutung gebraucht.

Für unsere Zwecke sind folgende Sätze aus [6] wichtig, deren Beweise unabhängig von der hier vorgenommenen Verallgemeinerung von dort übernommen werden können.

Satz 2. Für $t\neq l$ ist S_{lt} von der Klasse der sequentiellen Funktionen verschieden. Die einzigen sequentiellen Funktionen in S_{lt} sind die Homomorphismen.

Unter einer Wortpermutation verstehen wir eine Funktion s mit der Eigenschaft $s(x_1 \ldots x_n) = x_{\sigma_w(1)} \ldots x_{\sigma_w(n)}$ für jedes $w = x_1 \ldots x_n \in X^*$, wobei σ_w eine Permutation der Indexmenge $\{1, \ldots, |w|\}$ ist.

Satz 3. In jedem S_{lt} gibt es genau eine Wortpermutation s_{lt}. s_{lt} ist die identische Abbildung genau dann, wenn $l = t$ ist.

Zur Illustration geben wir für s_{lt} die Folge der erzeugenden Indexpermutationen an:

$$\sigma_x(1) =_{\mathrm{Df}} 1$$

$$\sigma_{wx}(i) =_{\mathrm{Df}} \begin{cases} \sigma_w(i) & \text{für } i < t(wx) \\ \sigma_w(i-1) & i > t(wx) \\ n+1 & i = t(wx). \end{cases}$$

Definition 10′

(a) Ist $\{\sigma_w : w \in X^*\}$ die Schar der erzeugenden Indexpermutationen für s_{lt}, so setzen wir für beliebige Wörter $p = y_1 \ldots y_n$ mit $n = |w|$ über beliebigem Alphabet Y

$$s_{lt}^w(p) =_{\mathrm{Df}} y_{\sigma_w(1)} \cdots y_{\sigma_w(n)}.$$

(b)
$$(\varphi * s_{lt})(w) =_{\mathrm{Df}} s_{lt}^w(\varphi(w)).$$

Satz 4. $f \in S_{lt}$ genau dann, wenn es eine sequentielle Funktion φ gibt mit

$$f = \varphi * s_{lt},$$

wobei s_{lt} die in S_{lt} vorhandene Wortpermutation ist.

Auch nichtlängentreue quasisequentielle Funktionen werden benötigt. Wir wollen sie in Anlehnung an Satz 4 erklären. Um eine geeignete Wortpermutation zu erklären, die an die Stelle von s_{lt} aus Satz 4 tritt, führen wir eine Funktion r ein, die jedem $w \in X^*$ eine natürliche Zahl $\zeta(w)$ und eine Permutation ϱ_w der Zahlen $1, \ldots, \zeta(w)$ zuordnet. Dabei ist $\zeta(wx)$ für die zu definierenden Funktionen der Längenzuwachs des Bildwortes von wx gegenüber dem Bildwort von w. Daher hat das Bildwort von w, wenn (was wir immer fordern wollen) das Bildwort des leeren Wortes wieder das leere Wort ist, die Länge $\lambda_r(w) = \sum_{w_1 \sqsubseteq w} \zeta(w_1)$. Wir definieren die Wortpermutation $s_{l[t,r]}$ durch Angabe der Folge $\{\sigma_w : w \in X^*\}$ der erzeugenden Indexpermutationen:

$$\sigma_x(1) = 1$$

$$\sigma_{wx}(i) = \begin{cases} \sigma_w(i) & \text{für } 1 \leqq i < t(wx) \\ \sigma_w(i - \zeta(wx)) & \text{für } t(wx) + \zeta(wx) \leqq i \leqq \lambda(wx) \\ \lambda_r(w) + \varrho_w(i - t(wx) + 1) & \text{für } t(wx) \leqq i < t(wx) + \zeta(wx). \end{cases}$$

Damit erklären wir nichtlängentreue quasisequentielle Funktionen.

Definition 11. Die Wortfunktion $f : X^* \to Y^*$ heißt $\langle l, [t, r]\rangle$-sequentiell $=_{\mathrm{Df}}$ Es existiert eine sequentielle Funktion $\varphi : X^* \to Y^*$ mit $|\varphi(w)| = \lambda_r(w)$ und

$$f = \varphi * s_{l[t,r]}.$$

Für $\zeta(w) \equiv 1$ ergeben sich als Spezialfall die längentreuen quasisequentiellen Funktionen.

Die Verallgemeinerung, daß der Zuwachs von $f(wx)$ gegenüber $f(w)$ nicht in Form eines einzigen Teilwortes an einer Stelle in $f(w)$ eingeschoben wird, sondern sich auf mehrere Stellen verteilt, wird hier nicht gebraucht.

§ 4 Funktionen, die im zweiten Sinne von treu ausspeichernden PDA berechnet werden

Die sogenannten treu ausspeichernden PDA verdienen deswegen besondere Beachtung, weil sie quasisequentielle Funktionen berechnen.

Definition 12. Der PDA $[Z, X, Y, K, \mu, z_0]$ heißt treu ausspeichernd $=_{\mathrm{Df}}$ Es existiert eine eindeutige Abbildung $\alpha: K \to Y^*$ mit

$$\forall y \forall z \forall z' \forall k \forall \varepsilon \left(\mu(z, k, \#) = (z', e, y, \varepsilon) \to y = \alpha(k)\right)$$

$$\forall y \forall z \forall z' \forall k \left(\mu(z, k, \varrho) = (z', e, y, 0) \to y = \alpha(k)\right).$$

Wir behandeln zu Beginn den Spezialfall der synchronen PDA.

Definition 13. Der PDA $P = [Z, X, Y, K, \mu, z_0]$ heißt synchron $=_{\mathrm{Df}}$ Ist $\mu(z, k, x) = (z', p, q, \varepsilon)$, so ist

(a) falls $x \in X$ ist, $p = k \wedge q \in Y$ (P druckt) oder $p = kk' \wedge q = e$ mit $k' \in K \setminus \{\Delta\}$ (P speichert),

(b) falls $x = \# \wedge k \neq \Delta$ ist, $p = e \wedge q \in Y$ (P speichert aus),

(c) falls $x = \# \wedge k = \Delta$ ist, $p = \Delta \wedge q = e \wedge \varepsilon = 1$,

(d) falls $x = \varrho \wedge k \neq \Delta$ ist, $p = e \wedge q \in Y \wedge \varepsilon = 1$.

Eine im zweiten Sinne von einem synchronen PDA berechnete Funktion ist offenbar längentreu.

Die Analyse des Berechnungsverhaltens im zweiten Sinne eines PDA $P = [Z, X, Y, K, \mu, z_0]$ stützt sich auf eine sequentielle Funktion τ_P, die die Arbeitsweise von P beschreibt. Ausgehend von P führen wir die gleichen Überlegungen durch wie im § 2 und verfügen damit über die Mengen M_i, M_i^j, $N_j^{i'}$ und die Funktionen g_i, d_i, s_i und h_i ($i = 0, \ldots, n$; $j = 1, \ldots, m$; $i' = 1, \ldots, n$). Wenn γ der durch

$$\gamma(u) =_{\mathrm{Df}} \begin{cases} D & \text{für } u \in Y, \\ S & \text{für } u \in K, \end{cases}$$

festgelegte Homomorphismus von $(Y \cup K)^*$ in $\{D, S\}^*$ ist, während η der Homomorphismus von Y^* in $\{A\}^*$ ist, der durch $\eta(y) = A$ für jedes $y \in Y$ bestimmt ist, so setzen wir

$$\gamma_i =_{\mathrm{Df}} g_i \circ \gamma, \quad \eta_j =_{\mathrm{Df}} h_j \circ \eta$$

und definieren τ_P durch $\{M_i, M_i^j, N_j^i, s_i, \gamma_i, \eta_j\}_{n,m}$ gemäß Definition 8.

Die so definierte Funktion τ_P beschreibt insofern die Arbeitsweise von P, als $\tau_P(w)$ genau die Folge der vorzunehmenden Operationen (D = Drucken, S = Speichern, A = Ausspeichern) angibt, die bei Abarbeitung von w im Sinne der Definition 3 auszuführen sind.

Definition 14. θ sei der Homomorphismus von $\{A, D, S\}^*$ in die Halbgruppe $[Nz, +]$ der natürlichen Zahlen, der durch $\theta(A)=\theta(D)=1$, $\theta(S)=0$ definiert wird. w' entstehe aus dem Wort w durch Streichen des letzten Buchstaben. Dann setzen wir $t_P(w) =_{Df} 1+\theta(\tau_P(w')) = 1+$Anzahl der in $\tau_P(w')$ vorkommenden A und D.

Die Bedeutung dieses t_P wird deutlich durch den

Satz 5a. Wird die Wortfunktion f durch einen synchronen treu ausspeichernden PDA P im zweiten Sinne berechnet, so ist $f \in S_{lt_P}$ (vgl. Def. 10c, wobei wie oben $l(w)=|w|$ gesetzt ist.)

Beweis. Nach Definition 10 ist für jedes $w \in X^*$ zu zeigen

$$\pi_{t_P(w)}(f(w))=f(\pi_{l(w)}(w)).$$

Es sei $f(x_1 \ldots x_n)=y_1 \ldots y_n$. Wir betrachten den Zeitpunkt, in dem die Abarbeitung von $x_1 \ldots x_n$ im ersten Sinne gerade beendet ist. Bis zu diesem Moment seien bereits die Buchstaben $y_1 \ldots y_k$ ausgegeben worden. Hieraus folgt:

1. Im Speicher von P befindet sich noch ein Wort der Länge $n-k$, das in den folgenden Takten ausgegeben wird, wobei das Restwort $y_{k+1} \ldots y_n$ entsteht.

2. Nach Definition von t_P ist $t_P(x_1 \ldots x_n x)= k+1$ für jedes $x \in X$.

Es sei $w=x_1 \ldots x_n x_{n+1}$. Wird $f(x_1, \ldots, x_{n+1})$ berechnet und ist $\tau_P(w)= =\tau_P(x_1 \ldots x_n)\sigma_{n+1}$, so sind zwei Fälle zu unterscheiden.

1. *Fall.* $\sigma_{n+1}=SA^c$ $(c \geqq 0)$.

Dann wird x_{n+1} als irgendein k_{n+1} gespeichert und anschließend werden c Buchstaben des Speichers, als erster der zuletzt gespeicherte Buchstabe k_{n+1}, ausgespeichert. Damit ist die Abarbeitung von w im ersten Sinne beendet. Der Rest von $f(w)$ entsteht durch nachfolgende Speicherleerung. Es ergibt sich

$$f(w)=y_1 \ldots y_k y_{n+1} y_{k+1} \ldots y_n \quad \text{mit} \quad y_{n+1}=\alpha(k_{n+1}).$$

2. *Fall.* $\sigma_{n+1}=DA^c$ $(c \geqq 0)$.

Dann wird x_{n+1} als ein y'_{n+1} gedruckt. Alles weitere verläuft wie im ersten Fall. Es entsteht

$$f(w)=y_1 \ldots y_k y'_{n+1} y_{k+1} \ldots y_n.$$

In beiden Fällen gilt

$$\pi_{t_P(w)}(f(w))=y_1 \ldots y_k y_{k+1} \ldots y_n=f(x_1 \ldots x_n)=f(\pi_{l(w)}(w)),$$

was wir beweisen wollten.

Nach Satz 4 gibt es für diese Funktion f eine Darstellung der Form $\varphi * s_{lt_P}$ mit sequentiellem φ, über das der nächste Satz eine Aussage macht.

Satz 5b. Die eben erwähnte Funktion φ ist gemäß Definition 8 durch $\{M_i, M_i^j, N_j^i, s_i, \bar{g}_i, \bar{h}_j\}_{n,m}$ gegeben, wobei alle $\bar{h}_j$ die konstante Funktion e bedeuten, die $\bar{g}_i$ aus g_i (vgl. § 2) dadurch entstehen, daß alle $k \in K$ durch $\alpha(k)$ ersetzt werden (vgl. Def. 12), während alle anderen Mengen bzw. Funktionen genau diejenigen sind, die sich bei der Analyse von P in § 2 ergeben haben.

Beweis. Durch Induktion über die Wortlänge von w zeigen wir

$$f(w)=s^{w}_{lt_p}(\varphi(w)).$$

Der Induktionsbeginn ist klar.

Induktionsannahme. Es gilt $f(w)=s^{w}_{lt_p}(\varphi(w))$ für $w=x_1\dots x_n$.
Induktionsschluß. Sei

$$f(x_1\dots x_n)=y_1\dots y_n,$$

$$\varphi(x_1\dots x_n)=y_{i_1}\dots y_{i_n}$$

und

$$\varphi(x_1\dots x_nx_{n+1})=y_{i_1}\dots y_{i_n}y_{i_{n+1}}.$$

Nach Definition von φ ist $y_{i_{n+1}}$ entweder $\alpha(k_{n+1})$, wobei k_{n+1} der beim Lesen von x_{n+1} gespeicherte Buchstabe ist, oder der beim Lesen von x_{n+1} gedruckte Buchstabe. Aus dem Beweis von Satz 5a wissen wir, daß $y_{i_{n+1}}$ in das Wort $f(x_1\dots x_n)$ an der Stelle $t_P(x_1\dots x_{n+1})$ eingefügt wird

$$f(x_1\dots x_{n+1})=y_1\dots y_k y_{i_{n+1}}y_{k+1}\dots y_n.$$

(Hierbei haben wir $t_P(x_1\dots x_{n+1}) = k+1$ angenommen.) Andererseits wirkt aber $s^{wx}_{lt_p}$, angewendet auf $\varphi(x_1\dots x_{n+1})$ nach Definition (§ 3) folgendermaßen. Der letzte Buchstabe $y_{i_{n+1}}$ wird an der Stelle $t_P(x_1\dots x_{n+1})$ in das Wort $s^{w}_{lt_p}(\varphi(x_1\dots x_n))$, das nach Induktionsannahme mit $y_1\dots y_n$ übereinstimmt, eingefügt. Also gilt

$$s^{wx}_{lt_P}(\varphi(x_1\dots x_{n+1})) = y_1\dots y_k y_{i_{n+1}}y_{k+1}\dots y_n = f(x_1\dots x_{n+1}),$$

womit der Satz bewiesen ist.

Die Funktionen φ aus Satz 5b und τ_p sind in folgendem Sinne miteinander verträglich: Sind φ und τ_p nach Definition 8 durch

$$\{M_i, M^j_i, N^i_j, s_i, \varphi_i, h_j\}_{n,m} \text{ bzw. } \{R_i, R^j_i, S^i_j, r_i, \tau_i, k_j\}_{n',m'}$$

gegeben, so gilt $n=n'$, $m=m'$ und für alle i und j

$$M_i=R_i,\quad M^j_i=R^j_i,\quad N^i_j=S^i_j \quad\text{und}\quad s_i=r_i.$$

Für die Umkehrung der bisherigen Analyseergebnisse benötigen wir folgende Definitionen.

Definition 15

(a) Wir bezeichnen mit T die Menge aller Funktionen aus $P_1(X, \{A, D, S\})$, die nach Definition 8 dargestellt werden können, wobei folgende zusätzliche Bedingungen erfüllt sind

1. $\forall i\forall w\,(|d_i(w)|=|w|)$
2. $d_i:X^*\to\{D, S\}^*$
3. $h_j(w)=A^{|w|}$ für jedes j
4. $\forall i\forall w\forall x(d_i(wx)=d_i(w)D\leftrightarrow s_i(w)=s_i(wx))$
5. $\forall i\forall w\forall x(|s_i(wx)|-|s_i(w)|\leqq 1)$.

(b) Mit $\mathfrak{T}_1$ bezeichnen wir die Menge aller $t\in\mathfrak{T}$, die aus den Funktionen $\tau\in T$ nach Definition 14 hervorgehen.

Bemerkungen

1. Die Funktionen aus $\mathfrak{T}_1$ sind alle mit wachsender Wortlänge monoton nicht fallend.

2. Die Beziehung zwischen den $t \in \mathfrak{T}_1$ und den $\tau \in T$ ist nicht eineindeutig. Gehört nämlich t nach Definition 14 zu τ, so auch zu jedem τ', das aus τ dadurch hervorgeht, daß gewisse D durch SA ersetzt werden.

Definition 16. Wir bezeichnen mit $\Phi(X, Y)$ die Menge aller Funktionen aus $P_1(X, Y)$, bei deren Darstellung nach Definition 8 alle h_j konstant auf das leere Wort abbilden und alle d_i längentreu sind. Ferner soll gelten $\forall i \forall w \forall x(|s_i(wx)| - |s_i(w)| \leqq 1)$.

$\Phi(X, Y)$ enthält nur sequentielle Funktionen, jedoch auch solche mit unendlichem Gewicht.

Beispiel. Wir wählen $X=\{a, b\}$, $K=\{0, 1\}$, $Y=\{a, b\}$,

$$M_0^1 = \{ba^n b : n \geqq 1\}, \quad M_1^1 = \{a\}, \quad N_1^1 = \{0\}, \quad N_1^2 = \{1\},$$

$d_0(ba^n b) = ba^n b$ für jedes n, $d_1(a) = a$, $d_2(a) = b$, $s_0(ba^n) = s_0(ba^n b) = 10^n$, $s_1(a) = e$, und die übrigen Bestimmungsstücke denken wir uns beliebig, aber im Einklang mit Definition 8 fixiert. Die hierdurch definierte Funktion heiße φ. Dann gilt

$$\varphi(ba^n ba^{n+1}) = ba^n ba^n b = ba^n b\varphi_{ba^n b}(a^{n+1})$$

und für $m > n$

$$\varphi(ba^m ba^{n+1}) = ba^m ba^{n+1} = ba^m b\varphi_{ba^m b}(a^{n+1}).$$

Hieraus folgt $\varphi_{ba^n b} \neq \varphi_{ba^m b}$ für $n \neq m$. Also hat φ kein endliches Gewicht.

Definition 17. Ist $t \in \mathfrak{T}_1$ und $\varphi \in \Phi(X, Y)$, so heißen t und φ verträglich $=_{\mathrm{Df}}$ Es gibt ein mit φ verträgliches (im Sinne der Ausführungen im Anschluß an Satz 5b) $\tau \in T$, aus dem t nach Definition 14 hervorgeht.

Satz 6. Eine längentreue Wortfunktion f von X^* in Y^* ist genau dann im zweiten Sinne von einem treu ausspeichernden synchronen einfachen PDA berechenbar, wenn es miteinander verträgliche Funktionen $\varphi \in \Phi(X, Y)$ und $t \in \mathfrak{T}_1$ gibt mit

$$f = \varphi * s_{lt}.$$

Beweis. Die eine Hälfte der Behauptung folgt ganz leicht aus den Sätzen 5a und 5b unter Berücksichtigung der Definitionen 15, 16 und 17. Es ist noch zu zeigen: Wenn $f = \varphi * s_{lt}$ und φ und t sind verträgliche Funktionen aus $\Phi(X, Y)$ bzw. $\mathfrak{T}_1$, so ist f PDA-berechenbar.

Nach Voraussetzung gibt es ein zu t gehöriges τ, das mit φ verträglich ist. τ und φ seien nach Definition 8 gegeben durch $\{M_i, M_i^j, N_j^i, s_i, \tau_i, h_j\}_{n,m}$ bzw. $\{M_i, M_i^j, N_j^i, s_i, \varphi_i, h_j'\}_{n,m}$. Wir definieren die Funktionen ψ_i durch

$$\psi_i(x_1 \ldots x_N) =_{\mathrm{Df}} H_1 y_1 H_2 y_2 \ldots H_N y_N, \text{ falls } \varphi_i(x_1 \ldots x_\nu) = \varphi_i(x_1 \ldots x_{\nu-1}) y_\nu$$

und

$$\tau_i(x_1 \ldots x_\nu) = \tau_i(x_1 \ldots x_{\nu-1}) H_\nu \quad \text{für} \quad \nu = 1, \ldots, N.$$

Wegen der vorausgesetzten Verträglichkeit ist es möglich, gemäß Definition 8 eine Funktion ψ durch $\{M_i, M_i^j, N_j^i, s_i, \psi_i, h_j\}_{n,m}$ zu definieren.

Wie beim Beweis von Satz 1 verschaffen wir uns zu ψ die gekoppelten Automaten Q_1 und Q_2. Mit den gleichen Bezeichnungen wie dort berechnet Q_1 im Zustand u_i $(i=0, \ldots, n)$ die Funktion ψ_i. Die Konstruktion des PDA $P=[Z, X, Y, K, \mu, z_0]$, der f berechnet, geschieht folgendermaßen. Wir setzen

a) für $v \in V \setminus V''$, $f_2(v, k)=v'$

$$\mu(v, k, \#) = \begin{cases} (v', e, k, 0), & \text{falls } v' \notin V'' \\ (v', k, e, 1), & \text{falls } v' \in V'', \end{cases}$$

b) für $u \in U \setminus U''$, $f_1(u, x)=u'$

$$\mu(u, k, x) = \begin{cases} (u', k, y, 0), & \text{falls } u' \in U'' \wedge g(u, k) = Dy, \\ (u', k, y, 1), & \text{falls } u' \notin U'' \wedge g(u, k) = Dy, \\ (u', kk', e, 1), & \text{falls } u' \notin U'' \wedge g(u, k) = Sk', \\ (u', kk', e, 0), & \text{falls } u' \in U'' \wedge g(u, k) = Sk'. \end{cases}$$

Wie leicht zu sehen ist, berechnet P das gegebene f, womit der Satz bewiesen ist.

Satz 6 läßt sich leicht auf nichtsynchrone einfache treu ausspeichernde PDA verallgemeinern. Wie oben bilden wir die Funktion τ_P. Aus ihr ist zu ersehen, welche Stellen in $f(wx)$ gestrichen werden müssen, um zu $f(w)$ zu gelangen. Ist

$$\tau_P(wx) = \tau_P(w) S^{a(wx)} D^{b(wx)} A^{c(wx)},$$

so sind gerade $a(wx)+b(wx)$ Stellen zu streichen, wobei die erste durch $t_P(wx)$ angegeben wird. Weiter ist zu sehen, daß die Länge von $f(w)$ durch $\lambda(w) = \sum_{u \sqsubseteq w} (a(u)+b(u))$ gegeben ist.

Wir definieren nun eine Folge von Permutationen ϱ_w durch

$$\varrho_e(1) = 1$$

$$\varrho_w(i) = \begin{cases} a+i & \text{für } 1 \leqq i \leqq b(w), \\ a+b+1-i & b(w) < i \leqq a(w)+b(w). \end{cases}$$

Da a, b und λ von P abhängen, hängen auch die ϱ von P ab. Wir setzen $r_P(w) =_{\mathrm{Df}} \varrho_w$ und nennen $[t_P, r_P]$ den zu τ_P gehörigen Typ.

Mit $\tilde{\Phi}(X, Y)$ bezeichnen wir die Klasse aller Funktionen aus $P_1(X, Y)$, bei deren Darstellung gemäß Definition 8 alle Funktionen h_j konstant auf das leere Wort abbilden.

Mit $\tilde{T}$ bezeichnen wir die Klasse aller Funktionen $\tau \in P_1(X, \{A, D, S\})$, bei deren Darstellung gemäß Definition 8 folgende Bedingungen erfüllt sind.

1. $d_i: X^* \to \{D, S\}^*$,
2. $\forall i \forall w \forall x \forall a \big(\exists b (d_i(wx) = d_i(w) S^a D^b \leftrightarrow |s_i(wx)| = a + |s_i(w)|\big)$,
3. $\forall j \forall w \big(h_j(w) = A^{|w|}\big)$.

Mit $\tilde{\mathfrak{T}}_1$ wird die Klasse aller Paare $[t, r]$ bezeichnet, die zu den $\tau \in \tilde{T}$ gehören.

Damit können wir die folgende Verallgemeinerung von Satz 6 formulieren, deren Beweis analog zum Beweis von Satz 6 verläuft.

Satz 6′. Eine Wortfunktion f von X^* in Y^* ist genau dann im zweiten Sinne von einem einfachen treu ausspeichernden PDA berechenbar, wenn es miteinander verträgliche Funktionen $\varphi \in \tilde{\Phi}(X, Y)$ und $\tau \in \tilde{T}$ gibt, so daß für den zu τ gehörigen Typ $[t, r]$ gilt

$$f = \varphi * s_{t,[t,r]}.$$

Bemerkung. Um nichtlängentreue Funktionen berechnen zu können, bei denen $f(wx)$ aus $f(w)$ dadurch hervorgeht, daß an mehreren Stellen des Wortes $f(w)$ Teilwörter eingefügt werden, muß man nichtsynchrone PDA betrachten, die über Befehle folgender Form verfügen: Werden im Zustand z die Eingabe x und das Speichersymbol k gelesen, so wird der Zustand z' angenommen, das Eingabeband um ε verschoben, und es werden

$$\begin{cases} k_{11}\ldots k_{1a_1} \text{ gespeichert,} \\ y_{11}\ldots y_{1b_1} \text{ gedruckt,} \\ c_1 \text{ Buchstaben ausgespeichert} \end{cases} \quad \ldots \quad \begin{cases} k_{n1}\ldots k_{na_n} \text{ gespeichert,} \\ y_{n1}\ldots y_{nb_n} \text{ gedruckt,} \\ c_n \text{ Buchstaben ausgespeichert.} \end{cases}$$

Diese Verallgemeinerung bleibt hier außer Betracht.

§ 5 Funktionen, die im zweiten Sinne von beliebigen PDA berechnet werden

Die Sätze 6 und 6′ zeigen, daß die im zweiten Sinne von treu ausspeichernden PDA berechneten Funktionen sequentielle Funktionen mit anschließender Permutation der Buchstaben der Bildwörter sind. Diese Permutationen hängen dabei von der Art und Weise der Speicherbenutzung ab. Die von beliebigen PDA berechneten Funktionen kann man sich so vorstellen, daß zunächst eine sequentielle Funktion berechnet wird, dann setzt eine Permutation s ein, und danach werden eventuell diejenigen Buchstaben geändert, die ausgespeichert worden sind. Denkt man sich diese letzten Änderungen *vor* der Permutation durchgeführt, so kann auch hier eine Gleichung der Form

$$f = \varphi * s \tag{8}$$

aufgeschrieben werden, wobei allerdings φ i.a. nicht mehr sequentiell ist.

Wir wollen das an einem Beispiel verfolgen und betrachten dazu einen PDA, für dessen τ-Funktion gilt

$$\tau(x_1x_2x_3x_4) = DSSD$$
$$\tau(x_1x_2x_3x_4x_5) = DSSDDA.$$

Hieraus folgt für das zugehörige s_{lt}

$$s_{lt}(x_1x_2x_3x_4) = x_1x_4x_3x_2 \tag{9}$$
$$s_{lt}(x_1x_2x_3x_4x_5) = x_1x_4x_5x_3x_2.$$

f sei die berechnete Funktion, und es sei

$$f(x_1x_2x_3x_4) = y_1y_4y_3y_2 \tag{10}$$
$$f(x_1x_2x_3x_4x_5) = y_1y_4y_5\bar{y}_3\bar{y}_2$$

mit $y_3 \neq \bar{y}_3$, $y_2 \neq \bar{y}_2$. Für φ ergibt sich aus (8), (9) und (10)

$$\varphi(x_1x_2x_3x_4) = y_1y_2y_3y_4$$

$$\varphi(x_1x_2x_3x_4x_5) = y_1\bar{y}_2\bar{y}_3y_4y_5.$$

Dieses φ ist nicht sequentiell, weil die zweite und dritte Stelle stören. Blendet man sie aus, so ist der Rest sequentiell. Solche Funktionen könnte man fastsequentiell nennen.

Wir wollen eine genaue Charakterisierung der möglichen fastsequentiellen φ angeben. Dazu gehen wir bei gegebenem (einfachen) PDA P wie im § 2 von den Automaten Q_1 und Q_2 aus und verfügen somit über die regulären Mengen M_i, M_i^j, N_j^i und die in den einzelnen Zuständen aus V'' bzw. U'' berechneten Funktionen g_i bzw. h_j. Nun erweitern wir den Definitionsbereich der h_j unter Beibehaltung der gleichen Bezeichnung auf $(K \cup Y)^*$: Ist $q_0, \ldots, q_r \in Y^*$ und $p_1, \ldots, p_r \in K^*$ und bezeichnet $h_{j,p}$ den von p bestimmten Zustand von h_j, so setzen wir

$$h_j(q_0p_1q_1p_2q_2 \ldots p_rq_r) =_{\mathrm{Df}} q_0h_j(p_1)q_1h_{j,p_1}(p_2)q_2 \ldots h_{j,p_1 \ldots p_{r-1}}(p_r)q_r. \tag{11}$$

Die Funktion φ aus (8) ergibt sich damit so

$$\left.\begin{array}{l} \varphi(p) = [h_{j_k}(\ldots[h_{j_2}([h_{j_1}(g_0(p_1))^{-1}]^{-1}g_i(p_2))^{-1}]^{-1} \ldots g_{i_{k-1}}(p_k))^{-1}]^{-1}g_{i_k}(p_{k+1}), \\ \text{wobei } p = p_1 \ldots p_{k+1} \text{ und} \\ p_1 \in M_0^{j_1} \wedge s(p_1)^{-1}\alpha N_{j_1}^{i_1} \wedge p_2 \in M_{i_1}^{j_2} \wedge s(p_1p_2)^{-1}\alpha N_{j_2}^{i_2} \wedge \ldots \wedge p_{k+1} \in M_{i_k}. \\ \text{Für } k=0 \text{ ist die Gleichung } \varphi(p) = g_0(p) \text{ gemeint.} \end{array}\right\} \tag{12}$$

Die Ergebnisse des vorigen Paragraphen sind hierin als Spezialfall enthalten. Dazu braucht man nur alle h_j als identische Abbildungen zu wählen, wodurch ein $\varphi \in \Phi(X, Y)$ entsteht.

Um das Ergebnis der Analyse formulieren und umkehren zu können, definieren wir eine Klasse $P_2(X, Y)$ fastsequentieller Wortfunktionen von X^* in Y^*.

Definition 18. $f \in P_2(X, Y) =_{\mathrm{Df}}$.

1. Es gibt eine endliche Menge K, und es existieren reguläre Mengen $M_i, M_i^j \subseteq X^*$ $(i=0, \ldots, n;\ j=1, \ldots, m)$ und reguläre Mengen $N_j^i \subseteq K^*$ $(i=1, \ldots, n;\ j=1, \ldots, m)$ mit den Eigenschaften (a)—(f) aus Definition 8.

2. Es gibt sequentielle Funktionen endlichen Gewichts

$$s_0, \ldots, s_n : X^* \to K^*,$$

$$g_0, \ldots, g_n : X^* \to (K \cup Y)^*,$$

$$h_1, \ldots, h_m : K^* \to Y^*,$$

so daß f durch (12) gegeben ist, wobei s durch (6) definiert ist und alle h_j durch (11) auf $(K \cup Y)^*$ fortgesetzt sind.

Wir wissen bereits aus dem eingangs dieses Paragraphen angegebenen Beispiel, daß $P_2(X, Y)$ auch nichtsequentielle Funktionen enthält.

Der folgende Satz ist eine Verallgemeinerung von Satz 6′. Der Beweis verläuft analog zum Beweis von Satz 6.

Satz 6″. Eine Wortfunktion f von X^* in Y^* ist genau dann im zweiten Sinne von einem einfachen PDA berechenbar, wenn es miteinander verträgliche Funktionen $\varphi \in P_2(X, Y)$ und $\tau \in \tilde{T}$ gibt, so daß für den zu τ gehörigen Typ $[t, r]$ gilt

$$f = \varphi * s_{l,[t,r]}.$$

Функции вычисляемые автоматами с магазинной памятью

Рассматриваются два вида вычисления функций автоматами с магазинной памятью. Описываются классы вычисляемых таким образом функций независимо от понятия автомата. Вычисляемые в первом смысле функции являются последовательностными функциями состоящими в некотором смысле но кускам из последовательностных функций конечного веса. Вычисляемые во втором смысле функции являются квази-последовательностными функциями точно описанного вида.

SEKTION MATHEMATIK DER
FRIEDRICH SCHILLER UNIVERSITÄT
69 JENA, DDR
UNIVERSITÄTSHOCHHAUS

Literatur

[1] EVEY, R. J., The theory and application of pushdown store maschines, mathematical linguistics and automatic translation, *Comput. Lab. Rept.*, Harvard University, v. NSF-10, May, 1963.
[2] GINSBURG, S., *The mathematical theory of contextfree languages,* Mc Graw-Hill Book Comp., New York, 1966.
[3] GINSBURG, S., G. F. ROSE, Preservation of languages by transducers, *Information and Control.*, v. 9, 1966, pp. 153—176.
[4] GLUSCHKOW, W. M., *Theorie der abstrakten Automaten,* VEB Verlag der Wissenschaften, Berlin 1963.
[5] RANEY, G. N., Sequential functions, *J. Assoc. Comput. Mach.*, v. 5, 1958, pp. 177—180.
[6] WECHSUNG, G., Quasisequentielle Funktionen, *Acta Cybernet.*, c. 2, 1973, pp. 23—33.

(Eingegangen am 27. Juli 1972)