Wilhelm Kämmerers Ideen vom automatischen Gedächtnis

Michael Fothe, Denny Steigmeier

Friedrich-Schiller-Universität Jena
Fakultät für Mathematik und Informatik
Ernst-Abbe-Platz 2
07743 Jena
michael.fothe@uni-jena.de
denny_steigmeier@gmx.de

Kurzfassung: In der Habilitationsschrift von Wilhelm Kämmerer aus dem Jahr 1958 finden sich Ideen zum Keller; er sprach vom automatischen Gedächtnis. Kämmerer setzte diese Struktur zum Aufbrechen von speziellen arithmetischen Formeln ein. Das entwickelte Verfahren eignet sich auch zur Verwaltung von Funktionsaufrufen und anderen Sprachkonstrukten.

Computerpioniere im deutschsprachigen Raum sind diejenigen, die von Konrad Zuse gemalt wurden.

Hartmut Wedekind

1 Zielstellung

Im September 2013 ehrte die Friedrich-Schiller-Universität Jena Wilhelm Kämmerer (1905–1994) mit einer Professorentafel an seinem langjährigen Wohnhaus, nachdem bereits im April 2013 eine „Oprema-Tafel“ am Gebäude der Fakultät für Mathematik und Informatik angebracht wurde [Webc, Webd, Webb]. Damit sollte einem möglichen Vergessen von Pionierleistungen der Informatik entgegengewirkt werden [Ker06]. Es entstand die Idee, sich genauer mit der Habilitationsschrift von Kämmerer zu befassen und diese einem breiteren Publikum bekannt zu machen [Käm58].

Kämmerer geht in seiner Habilitationsschrift einer praktisch bedeutsamen Fragestellung nach und entwickelt dabei Grundlagen für den Compilerbau und damit für den Einsatz höherer Programmiersprachen. Die Untersuchungen beruhen auf Erfahrungen, die er als Entwicklungsleiter im VEB Carl Zeiss Jena mit der Oprema, einem der ersten Computer Deutschlands, und dem Nachfolger ZRA 1 gewonnen hatte [Win10]. Kämmerer schreibt einschränkend: „Diese Abhandlung strebt nicht eine konstruktive Bauvorschrift an, wenn sie auch gelegentlich geschehene Hinweise in dieser Richtung für nicht unangebracht hält“ [Käm58, S. 69]. Im Rahmen einer studentischen Projektarbeit implementierte Denny Steigmeier das von Kämmerer entwickelte Verfahren als Java-Programm (siehe Abschnitt 5

und Quelle [Webe]). Auch hier trat das ein, was Raúl Rojas und andere bei ihren Implementierungsarbeiten zu einer Untermenge des Plankalküls von Konrad Zuse erlebten, dass nämlich mit solchen Projekten „Computergeschichte plötzlich und unerwartet lebendig wird“ [RGF+04, S. 228 f.].

2 Ausgangssituation

Kämmerer führt zu Beginn seiner Habilitationsschrift aus: „[D]ie maschinengerechte Programmierung [erfordert] erheblichen Aufwand an Zeit und routinemäßiger Konzentration und führt zu einer unübersichtlichen, weitschweifigen, schwer kontrollierbaren und derartig individuellen Form, daß Verständigungen über angewandte Verfahren zwischen verschiedenen Rechenzentren nur schwer möglich sind.“ Und weiter: „War man zunächst froh, überhaupt Rechenautomaten zu besitzen, so mußte man bald erkennen, daß man ihre Vorteile mit einer zeitraubenden, Unsicherheit tragenden Programmierungsarbeit bezahlte“ [Käm58, S. 2]. Dieser Standpunkt wird verständlich, wenn man sich Programme aus der damaligen Zeit anschaut, wie zum Beispiel das folgende; es berechnet $z = x^2 + y^2$ [Käm60, S. 214 f.]:

000	0 01 101
001	0 04 101
002	0 07 103
003	0 01 102
004	0 04 102
005	0 02 103
006	0 07 103
007	0 16 103
008	0 00 ...

Kämmerer geht in seiner Arbeit auf die Rechenplanberechnung nach Rutishauser näher ein und schreibt: „Das Charakteristische dieses Verfahrens ist ein fortgesetztes Durchmustern mit einem immer wieder von links nach rechts wechselnden Augenspiel“ [Käm58, S. 27]. Das Verfahren besitzt einen schlechten (quadratischen) Zeitaufwand mit der Eingabelänge. Friedrich L. Bauer bezeichnet es wohl in Anspielung an die Prozession in Echternach (Luxemburg) als „Springprozession Rutishausers“ [Bau04, S. 238]. Kämmerer führt aus: „In der vorliegenden Abhandlung werden in konsequenter Weiterführung der abgelaufenen Entwicklungstendenz auf einem neuen Weg Prinzipien für die Struktur eines Automaten erarbeitet, der in enger Anlehnung an das mathematische Formelbild eine bequeme, übersichtliche und adressenfreie Programmierung gestattet“ [Käm58, S. 3]. Und auch: „[D]as nach mathematischem Formelbild eingegebene Programm [ist] schon der Plan [...], den der Automat direkt verarbeiten kann“ [Käm58, S. 21]. Kämmerer nutzt als Gymnasiallehrer für Mathematik und Physik (promoviert in Mathematik) die Notation, mit der er bestens vertraut ist. Er stellt auch fest: „Das Programmieren würde mit der Aufstellung des algorithmischen Plans und der strukturellen Klärung der auftretenden Größen sein Ende finden“ [Käm58, S. 21].

3 Nutzung des automatischen Gedächtnisses

Das automatische Gedächtnis ist bei dem von Kämmerer entwickelten Verfahren zum Aufbrechen von Formeln von zentraler Bedeutung. Er schreibt: „[W]esentlich an der Methode ist, daß dem Automaten ein Gedächtnis eigen ist, über das er selbst verfügt“ [Käm58, S. 39]. Die Begriffe Gedächtnis und Speicher verwendet er synonym: „Eine nach diesen Prinzipien entwickelte Maschine wird über einen großen Teil des maschineneigenen Speicherraums selbst verfügen“ [Käm58, S. 25]. Die Arbeitsweise des Automaten kennzeichnet Kämmerer wie folgt: „[A]usführen wird er [...] jeweils nur, was sich als ausführungsmöglich bis dahin geklärt hat, das restliche wird er in seinem Gedächtnis zurückstellen [...]“ [Käm58, S. 67].

Wesentlich ist die Zelle vom Dienst (ZvD): „Der Schwerpunkt, die Stelle höchsten Bewußtseins, liegt dabei in der ZvD“ [Käm58, S. 39]. Kämmerer beschreibt die Erhöhung des Spannungsgrades: „Ein Aufsteigen in der Lage des ZvD [sic] ist mit Niederschrift von z. Z. hinsichtlich der Ausführungsbestimmungen unklar bleibenden Plananordnungen verbunden.“ Für eine Verringerung des Spannungsgrades gilt: „Ein Absteigen in der Lage des ZvD [sic] ist mit Lesen von Rückständen und entsprechenden Ausführungsmaßnahmen verbunden“ [Käm58, S. 30]. „Dabei sind jeweils alle Zellen mit Nummern über der der ZvD noch leer, oder ihr Inhalt ist schon wieder uninteressant geworden. Alle Zellen, deren Nummer unter der der ZvD liegen, haben dagegen wesentliche Informationen [...]“ [Käm58, S. 30]. „Während des Anstehens eines Befehls im Befehlsregister verbleibt der größte Teil dieser Gedächtniszellen im ‚Unterbewußtsein‘ des Automaten“ [Käm58, S. 39]. Wie Kämmerer auf diese psychoanalytische Begrifflichkeit gekommen ist, konnte nicht geklärt werden.

In der Habilitationsschrift sind zwei Beispiele beschrieben. In der Beschreibung des ersten Beispiels (siehe Abbildung 1) kann man das typische „Gebirge“ erkennen, das beim Aufbrechen von Formeln entsteht.

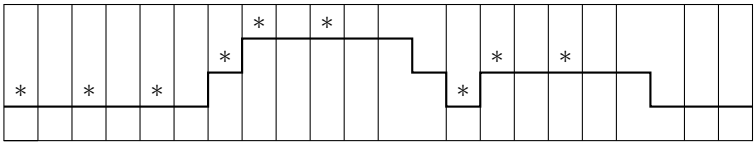

Abbildung 1: Lage der Zelle vom Dienst (ZvD) bei der Bearbeitung des Beispiels $a - b + c * (d + e) - (f + g) \Rightarrow h$. Der Doppelpfeil ist der Zuweisungsoperator. Der Stern bedeutet: Neubeschreibung der ZvD [Käm58, S. 49].

Nachfolgend sind drei Beispiele mit steigender Komplexität angegeben; diese sind nicht in der Habilitationsschrift enthalten. In der Spalte 1 sind unter der Überschrift „Rechenplan“ die eingelesenen Zeichen aufgeführt. Der Spalte 2 ist der aktuelle Wert des Resultatsregisters (RR) zu entnehmen. Die ZvD kann einen Wert sowie Anmerkungen (Zeichen) aufnehmen. In der Spalte 3 steht der Wert, in der Spalte 4 stehen die Anmerkungen Q_1 und Q_2, wenn sie gesetzt werden. Spalte 5 gibt die Lage der ZvD an. Zu Beginn und am Ende der Bearbeitung ist die Lage stets 1. In den Spalten 6 und 7 sind die Abfragen, ob die Anmerkungen Q_1 bzw. Q_2 gesetzt sind, protokolliert. In diesem Beitrag wird der Stern als Multiplikationsoperator verwendet (Kämmerer verwendete den Punkt).

	1 Rechenplan	2 RR	3 ZvD	4 Q_i	5 Lage ZvD	6 Q_1?	7 Q_2?
1	Anfang	0	0		1.		
2	a	a			1.		
3	$*$				1.		
4	b	$a * b$			1.		
5	$-$		$0 + a * b$	Q_1	1.	nein	
6	c	c			1.		
7	$*$				1.		
8	d	$c * d$			1.		
9	$\Rightarrow$	$a * b - c * d$			1.	ja	
10	r	$\langle \text{RR} \rangle \Rightarrow r$			1.		

Tabelle 1: Beispiel $a * b - c * d \Rightarrow r$

Schauen wir uns nun das Beispiel aus der Tabelle 1 genauer an. Am Anfang werden das Resultatsregister und die Zelle vom Dienst gelöscht. Ein Wert wird nach dem Einlesen in das Resultatsregister gebracht (Zeilen 2 und 6). Es gibt eine Besonderheit: Wird nach dem Einlesen eines Multiplikationsoperators ein Wert eingelesen, so wird der Inhalt des Resultatsregisters sofort mit diesem Wert multipliziert und das Produkt wird in das Resultatsregister gebracht (Zeilen 4 und 8). Nach dem Einlesen des Subtraktionsoperators (Zeile 5) wird geschaut, ob die Anmerkung Q_1 gesetzt ist (Spalte 6). Da sie in unserem Beispiel nicht gesetzt ist, wird nicht subtrahiert, sondern der Wert der ZvD und der Inhalt des Resultatsregisters werden addiert; die Summe wird in der ZvD gespeichert. Zusätzlich wird die Anmerkung Q_1 gesetzt. Die nachhängende Subtraktion wird in der Zeile 9 nach dem Einlesen des Zuweisungsoperators ausgeführt. Vom Wert der ZvD wird der Inhalt des Resultatsregisters subtrahiert; das Ergebnis wird in das Resultatsregister gebracht. In der Zeile 10 wird das Berechnungsergebnis der Variablen r zugewiesen. In dem Beispiel wird nur eine ZvD benötigt (siehe Spalte 5).

Das Beispiel aus der Tabelle 2 enthält ein Klammerpaar. Nach dem Einlesen des Multiplikationsoperators (Zeile 5) wird, da eine öffnende Klammer folgt, der Spannungsgrad um 1 erhöht. Die Anmerkung Q_2 wird gesetzt; sie steht für eine nachhängende Multiplikation. (Das Setzen der Anmerkungen Q_2 und Q_1 würde für eine nachhängende Division stehen.) Des Weiteren wird der Inhalt des Resultatsregisters in die ZvD kopiert. Nach dem Einlesen der öffnenden Klammer (Zeile 6) wird der Spannungsgrad um 1 erhöht und das Resultatsregister und die ZvD werden gelöscht. Nach dem Einlesen der schließenden Klammer (Zeile 10) folgen zwei Schritte: Zuerst wird eine nachhängende Subtraktion ausgeführt. Vom Wert der ZvD wird dazu der Inhalt des Resultatsregisters subtrahiert; das Ergebnis wird in das Resultatsregister gebracht. Der Spannungsgrad wird um 1 verringert. Gleich anschließend wird eine nachhängende Multiplikation ausgeführt (Zeile 11), und zwar werden der Wert der ZvD auf Stufe 2 (Wert von b) und der Inhalt des Resultatsregisters multipliziert; das Produkt kommt in das Resultatsregister. Der Spannungsgrad wird noch einmal um 1 verringert. Nach dem Einlesen des Subtraktionsoperators (Zeile 12) wird die nachhängende Addition ausgeführt und Q_1 wird gesetzt. Die nachhängende Subtraktion

wird nach dem Einlesen des Zuweisungsoperators ausgeführt (Zeile 14). Abschließend erfolgt die Zuweisung des Ergebnisses an die Variable s.

	1 Rechenplan	2 RR	3 ZvD	4 Q_i	5 Lage ZvD	6 Q_1?	7 Q_2?
1	Anfang	0	0		1.		
2	a	a			1.		
3	$+$		$0+a$		1.	nein	
4	b	b			1.		
5	$*$		b	Q_2	2.		
6	$($	0	0		3.		
7	c	c			3.		
8	$-$		$0+c$	Q_1	3.	nein	
9	d	d			3.		
10	$)$	$c-d$			3.	ja	
11		$b*(c-d)$			2.	nein	ja
12	$-$		$a+b*$ $(c-d)$	Q_1	1.	nein	
13	e	e			1.		
14	$\Rightarrow$	$a+b*(c-d)-e$			1.	ja	
15	s	$\langle \text{RR} \rangle \Rightarrow s$			1.		

Tabelle 2: Beispiel $a+b*(c-d)-e \Rightarrow s$

Das Beispiel aus der Tabelle 3 ist extrem, denn die erste Operation, die beim Durchlaufen der Zeichenfolge von links nach rechts ausgeführt wird, ist die am weitesten rechts stehende. Es wird also als erstes $i+k$ gerechnet. Bis dahin werden die Operationen, die verzögert ausgeführt werden sollen, und die benötigten Operanden im automatischen Gedächtnis gespeichert. Ab Zeile 25 werden die Operationen nachträglich ausgeführt, und zwar abwechselnd eine Addition und eine Multiplikation.

4 Funktionsaufrufe

Das in der Habilitationsschrift entwickelte Verfahren ermöglicht auch Funktionsaufrufe aus dem automatischen Gedächtnis heraus. Zulässig sind Funktionen von einem oder mehreren Parametern und in Verschachtelung [Käm58, S. 68]. Parameter können zusammengesetzte Ausdrücke sein [Käm58, S. 37 f.]. Das Vorgehen entspricht der Übergabeart *call by value* für Parameter. Auch wenn rekursive Aufrufe von Funktionen nicht vorgesehen sind, handelt es sich um einen insgesamt weitgehenden Leistungsumfang.

Das zweite Beispiel, das in der Habilitationsschrift beschrieben ist, lautet in der maschinengerechten Schreibweise der Plangleichung [Käm58, S. 50]:

$$\exp(-\text{pot}(x;2)) * \sin(c*x) \Rightarrow z$$

	1 Rechenplan	2 RR	3 ZvD	4 Q_i	5 Lage ZvD	6 Q_1?	7 Q_2?
1	Anfang	0	0		1.		
2	a	a			1.		
3	$+$		$0+a$		1.	nein	
4	b	b			1.		
5	$*$		b	Q_2	2.		
6	$($	0	0		3.		
7	c	c			3.		
8	$+$		$0+c$		3.	nein	
9	d	d			3.		
10	$*$		d	Q_2	4.		
11	$($	0	0		5.		
12	e	e			5.		
13	$+$		$0+e$		5.	nein	
14	f	f			5.		
15	$*$		f	Q_2	6.		
16	$($	0	0		7.		
17	g	g			7.		
18	$+$		$0+g$		7.	nein	
19	h	h			7.		
20	$*$		h	Q_2	8.		
21	$($	0	0		9.		
22	i	i			9.		
23	$+$		$0+i$		9.	nein	
24	k	k			9.		
25	$)$	$i+k$			9.	nein	
26		$h*(i+k)$			8.	nein	ja
27	$)$	$g+h*(i+k)$			7.	nein	
28		$f*(g+h*(i+k))$			6.	nein	ja
29	$)$	$e+f*(g+h*(i+k))$			5.	nein	
30		$d*(e+f*(g+h*(i+k)))$			4.	nein	ja
31	$)$	$c+d*(e+f*(g+h*(i+k)))$			3.	nein	
32		$b*(c+d*(e+f*(g+h*(i+k))))$			2.	nein	ja
33	$\Rightarrow$	$a+b*(c+d*(e+f*(g+h*(i+k))))$			1.	nein	
34	t	$\langle \text{RR} \rangle \Rightarrow t$			1.		

Tabelle 3: Beispiel $a+b*(c+d*(e+f*(g+h*(i+k)))) \Rightarrow t$

Nachfolgend wird der Funktionsaufruf $\mathrm{pot}(x; n)$ zum Berechnen einer Potenz erläutert. Der Wert des ersten Parameters wird (ggf. nach dessen Berechnung) der Speicherzelle am Anfang des Unterprogramms pot zugewiesen. Der Wert des zweiten Parameters wird (ggf. nach dessen Berechnung) der nachfolgenden Speicherzelle $\mathrm{pot} + 1$ zugewiesen. Für die Parameterübergabe gilt in unserem Beispiel: $x \Rightarrow \langle \mathrm{pot} \rangle, 2 \Rightarrow \langle \mathrm{pot} + 1 \rangle$ [Käm58, S. 51]. Danach erfolgt ein Sprung nach $\mathrm{pot} + 2$ und das Unterprogramm wird abgearbeitet. Anschließend enthält das Resultatsregister den Funktionswert. Bei Kämmerer heißt es: „Greift hinsichtlich der ZvD um eine Stufe zurück und erhält dort die Rückkehradresse“ [Käm58, S. 38]. In unserem Beispiel liefert das automatische Gedächtnis die Adresse $\mathrm{A}+10$. Ein Sprung dorthin im Hauptprogramm liefert den nächsten Befehl, der abzuarbeiten ist.

5 Implementierung des Verfahrens

Die von Kämmerer angegebene Bauvorschrift wurde als Java-Programm implementiert. Das Programm besitzt einen objektorientierten Aufbau; es besteht aus acht Klassen (siehe Abbildung 2).

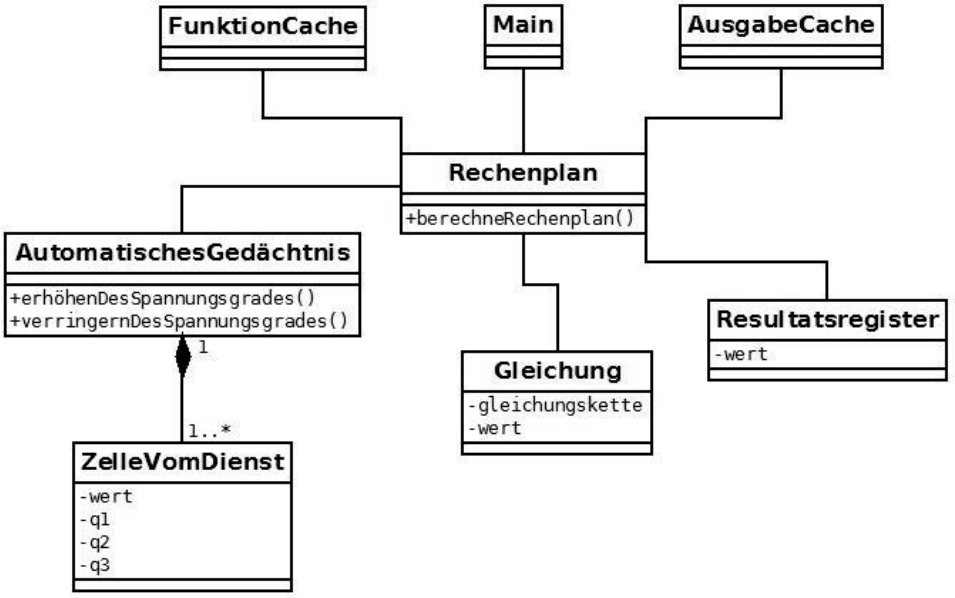

Abbildung 2: Aufbau der Implementierung

Die Klassen Main, FunktionCache und AusgabeCache stellen technische Klassen dar. Die Main-Klasse dient zum Initialisieren der benötigten Objekte und zum Starten des Programms. Im AusgabeCache werden die jeweiligen Ausgaben des Rechenplans zwischengespeichert und nach der Berechnung auf der Java-Oberfläche ausgegeben. Der

FunktionCache ist ein Zwischenspeicher zum Abarbeiten von Bibliotheksprogrammen bzw. Funktionen. Der Rechenplan ist die zentrale Klasse. Diese Klasse besitzt eine Gleichung, ein Resultatsregister und ein automatisches Gedächtnis. In der Gleichung ist die Gleichungskette hinterlegt und das Resultatsregister dient zum Abspeichern des Inhalts gemäß Rechenplan. Das automatische Gedächtnis ist als Stack implementiert und besitzt die Methoden `push()` (Erhöhung des Spannungsgrades) und `pop()` (Verringerung des Spannungsgrades). Die Elemente dieses Speichers sind die ZvD. Jede ZvD besitzt einen Zeiger auf ihren Vorgänger bzw. einen Null-Zeiger im Falle der untersten ZvD.

Nachdem der Programmbenutzer die Formel eingegeben hat, wird diese zunächst dem Rechenplan übergeben und in der Klasse Gleichung hinterlegt. Im nächsten Schritt wird die Methode `berechneRechenplan()` in der Klasse Rechenplan aufgerufen. Diese Methode liest die Zeichen der Formel schrittweise aus. Die Zeichen werden einer Switch-Struktur übergeben, in der sie erkannt und die entsprechenden Methoden aufgerufen werden. Mögliche Aufrufe sind:

- `symbol()`
- `addition()`, `subtraktion()`
- `multiplikationMitSymbol()`, `multiplikationOhneSymbol()`
- `divisionMitSymbol()`, `divisionOhneSymbol()`
- `klammerAuf()`, `klammerZu()`
- `semikolon()`
- `ergibt()`

Eine Ausnahme zur schrittweisen Abarbeitung der Zeichen besteht bei einem Multiplikationsoperator. Hierbei wird zusätzlich auf das folgende Zeichen der Gleichung zugegriffen. Nachdem der Rechenplan abgearbeitet wurde, kann das Ergebnis der Benutzungsoberfläche übergeben und ausgedruckt werden.

Die Darlegungen in der Habilitationsschrift erwiesen sich als konsistent und als geeignete Grundlage für die Implementierung des entwickelten Verfahrens.

6 Charakterisierung des Verfahrens

An der Verarbeitung von Ausdrücken kann man vieles studieren, was für kontextfreie Sprachen insgesamt relevant ist. Kämmerer entwickelt ein Verfahren speziell zum Aufbrechen von Formeln, die Additionsoperatoren $+$ und $-$, Multiplikationsoperatoren $*$ und $/$, Klammern sowie Funktionsaufrufe enthalten können. Eine Erweiterungsmöglichkeit auf weitere Stufen von Operatoren wird in der Habilitationsschrift nicht thematisiert; sie ist für das angestrebte Ziel nicht erforderlich. Eine Erweiterungsmöglichkeit ist auch nicht offensichtlich, da Additions- und Multiplikationsoperatoren uneinheitlich behandelt werden.

Man vergleiche zum Beispiel die Begrenzerpaarmethode mit Gewichtsvektor und einem Keller nach Edsger W. Dijkstra (1961), bei der alle Operatoren in gleicher Weise bearbeitet werden [Ker73, S. 455 ff.]. Das von Kämmerer entwickelte Verfahren kennt keine Unterscheidung von Zahlen- und Operatorenkeller. Sowohl Operatoren (kodiert über so genannte Anmerkungen) als auch Variablen sind stattdessen im automatischen Gedächtnis enthalten (in den Zellen vom Dienst).

Die gegebene Formel wird einmal von links nach rechts durchlaufen. Das aktuell eingelesene Zeichen bestimmt den Fortgang der Arbeiten. Mitunter ist eine Vorausschau um ein weiteres Zeichen erforderlich. Das Ausführen von Operationen wird, falls notwendig, verzögert. Die dafür erforderlichen Angaben werden in das automatische Gedächtnis gebracht und ihm später wieder entnommen. Dieser Speicher flexibler Größe wird ökonomisch verwaltet. Das Verfahren besitzt einen (wünschenswerten) linearen Zeitaufwand mit der Eingabelänge.

Friedrich L. Bauer geht auf Kämmerers Arbeit wie folgt ein: „Nachzutragen wäre, daß auch in der am 21.4.1958 eingereichten Habilitationsschrift von Wilhelm Kämmerer [...] gewisse Parallelen zum Kellerprinzip zu finden sind. Es ist zu vermuten, daß Kämmerer der Dresdner Vortrag von Samelson (1955) entgangen war. Von der am 30. März 1957 eingereichten Patentanmeldung von Bauer und Samelson für eine ‚formelgesteuerte Maschine' konnte Kämmerer natürlich noch nicht wissen" [Bau04, S. 238]. N. Joachim Lehmann sieht es folgendermaßen: „Erste Überlegungen dazu wurden von K. Samelson 1955 in Dresden angedeutet. Die endgültige Fassung des Kellerprinzips haben dann W. Kämmerer (1958, 1959) sowie K. Samelson und F. L. Bauer (1959) unabhängig voneinander entwickelt und publiziert" [Leh04, S. 257].

7 Weiterer Ausbau und Ausblick

Kämmerer beschreibt im §9 seiner Habilitationsschrift Ideen zu Sprachelementen höherer Programmiersprachen. Konkret geht er auf die folgenden Konstrukte ein:

- unbedingter Sprung,
- Alternative, Wahrheitswerte,
- Schleifen, Index-gebunden,
- Schleifen, ohne Bindung an einen Index,
- öffnendes und schließendes Symbol,
- Summenzeichen,
- Matrizen.

Er stellt fest, dass „für derartige strukturelle Fragen in der Schreibweise noch kein Analogon zu dem mathematischen Formelbild erwachsen ist" [Käm58, S. 55]. Für einige dieser

Konstrukte schlägt er eine Realisierung mithilfe des automatischen Gedächtnisses vor. Ein bedeutender Meilenstein in der Programmiersprachenentwicklung war dann ALGOL 60. An der Entwicklung dieser Programmiersprache war Kämmerer nicht beteiligt [Bau04, Leh04].

Kämmerer nahm eine Zusammenfassung seiner Habilitationsschrift in ein Lehrbuch auf [Käm60, S. 289 f.]. Im Berichtsband „Informatik“ der Jahresversammlung 1971 der Leopoldina gibt Kämmerer (als deren Mitglied) einen breit angelegten Überblick über das Thema „Algorithmische Sprachen“, in dem er u. a. den Zusammenhang zwischen Kellerautomaten und kontextfreien Sprachen thematisiert [Käm72]. Dabei erwähnt Kämmerer seine eigenen Untersuchungen von 1958 nicht einmal in einer Fußnote. Diese Form der Zurückhaltung ist beeindruckend.

Dem Computerpionier Wilhelm Kämmerer (gemalt von Konrad Zuse!) gelangen nicht nur bei der Computerentwicklung, sondern auch auf dem Gebiet der Programmiersprachen und zu einer wichtigen Struktur der Informatik bemerkenswerte Leistungen [JG95, Weba].

Literatur

[Bau04] F. L. Bauer. Die Algol-Verschwörung. In H. D. Hellige, Hrsg., *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive*, Seiten 237–254. Springer-Verlag, Berlin, Heidelberg, New York, 2004.

[JG95] J. Jänike und F. Genser. *Die Vergangenheit der Zukunft. Deutsche Computerpioniere*. Düsseldorf, 2. Auflage, 1995.

[Käm58] W. Kämmerer. Ziffern-Rechenautomat mit Programmierung nach mathematischem Formelbild. Habilitationsschrift, Friedrich-Schiller-Universität Jena, http://www.db-thueringen.de/servlets/DocumentServlet?id=22616, 1958.

[Käm60] W. Kämmerer. *Ziffernrechenautomaten*. Band 1 von *Elektronisches Rechnen und Regeln*. Akademie-Verlag Berlin, 1960.

[Käm72] W. Kämmerer. Algorithmische Sprachen. In J.-H. Scharf, Hrsg., *Informatik*, Nova Acta Leopoldina 206, Bd. 37/1, Seiten 403–417. Johann Ambrosius Barth, Leipzig, 1972.

[Ker73] I. O. Kerner. *Numerische Mathematik und Rechentechnik. Teil 2/2*. Mathematisch-Naturwissenschaftliche Bibliothek, Bd. 47/2. Teubner, Leipzig, 1973.

[Ker06] I. O. Kerner. OPREMA und ZRA 1 – die Rechenmaschinen der Firma Carl Zeiss Jena. In F. Naumann und G. Schade, Hrsg., *Informatik in der DDR – eine Bilanz*, Lecture Notes in Informatics – Thematics, Seiten 147–177. Bonn, 2006.

[Leh04] N. J. Lehmann. ALGOL im Ostblock und der Weg zu Systemen von Programmiersprachen. In H. D. Hellige, Hrsg., *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive*, Seiten 255–273. Springer-Verlag, Berlin, Heidelberg, New York, 2004.

[RGF+04] R. Rojas, C. Göktekin, G. Friedland, M. Krüger, L. Scharf, D. Kuniß und O. Langmack. Konrad Zuses Plankalkül. Seine Genese und eine moderne Implementierung. In H. D. Hellige, Hrsg., *Geschichten der Informatik. Visionen, Paradigmen, Leitmotive*, Seiten 215–235. Springer-Verlag, Berlin, Heidelberg, New York, 2004.

[Weba] Computerpioniere. http://www.superbrain.eu/Pioniere.htm.

[Webb] Der erste Informatiker in der Tafelrunde.
http://www.uni-jena.de/Mitteilungen/PM130913_Kämmerer_Tafel.html.

[Webc] Erster DDR-Computer hatte 500 Kilometer lange Leitungen.
http://www.heise.de/ix/meldung/Erster-DDR-Computer-hatte-500-Kilometer-lange-Leitungen-1834998.html.

[Webd] Erster DDR-Computer. Mit diesem Monstrum konnte man rechnen.
http://www.spiegel.de/einestages/erster-ddr-computer-oprema-a-951147.html.

[Webe] Java-Implementierung. https://cms.rz.uni-jena.de/minet_multimedia/autoGed.zip.

[Win10] J. F. H. Winkler. Konrad Zuse und die Optik-Rechenmaschine in Jena. *LOG IN*, Heft Nr. 166/167, Seiten 132–136, 2010.

Die angegebenen Internetquellen wurden zuletzt am 1. März 2015 geprüft.