Friedrich L. Bauers und Klaus Samelsons Arbeiten in den 1950er-Jahren zur Einführung der Begriffe Kellerprinzip und Kellerautomat

Hans Langmaack

Christian-Albrechts-Universität zu Kiel
Institut für Informatik
Christian-Albrechts-Platz 4
24118 Kiel
hl@informatik.uni-kiel.de

Kurzfassung: F. L. Bauer und K. Samelson leisteten außerordentliche Pionierbeiträge zur Informatik, insbesondere zur Programmierung elektronischer Rechenanlagen und zur Entwicklung, Definition und Implementierung höherer Programmiersprachen. Über die im Titel genannte Thematik Kellerprinzip und -automat hinaus wird auf einige Vorläufer des Kellergedankens eingegangen.

1 Einleitung

Für die ehrenvolle Einladung, im Jenaer „Kellerkolloquium“ diesen Vortrag zu halten, danke ich sehr. Die beiden Informatikpioniere kenne ich seit 1960. Am 1. Juni trat ich meinen Dienst als (an der Universität Münster frisch Promovierter) wissenschaftlicher Assistent von Herrn Samelson am Mathematischen Institut der Johannes-Gutenberg-Universität in Mainz an. Herr Bauer war Chef des Instituts für Angewandte Mathematik, das seit 1958 den Zuse-Rechner Z22 in Betrieb hatte, wofür Manfred Paul den ALGOL 60-Übersetzer ALCOR MAINZ Z22 konstruierte und programmierte, der Anfang 1961 fertiggestellt war.

Im Herbst 1960 wurde der Rechner SIEMENS 2002 installiert. Weil die Herren Bauer und Samelson über den mitgelieferten ALGOL 60-Übersetzer überhaupt nicht glücklich waren, erhielten Frau Ursula Hill (später Frau Hill-Samelson) und ich den Auftrag, den ALGOL 60-Übersetzer ALCOR MAINZ 2002 zu konstruieren, der Anfang 1962 fertig wurde. Damals lernte ich auch Herrn Wilhelm Kämmerer kennen. Er besuchte uns ca. 1961 in Mainz und interessierte sich für unsere Arbeiten am Übersetzer. Herrn Kämmerers Buch [Kaem60] „Ziffernrechenautomaten“ steht in meinem Bücherregal. Ich kaufte es mir als mein erstes Buch über reale Rechenmaschinen im Kontrast zu den Turing-Maschinen, die ich während meines Nebenfachstudiums „Mathematische Logik“ in Münster kennengelernt hatte [Her61].

Bauer und Samelson leisteten in den 1950/60er-Jahren außerordentliche Pionierbeiträge zur Informatik, insbesondere zur Programmierung elektronischer Rechenanlagen und zur Entwicklung, Definition und Implementierung höherer Programmiersprachen. Die Beiträge

sind heute selbstverständliches informatisches Ideengut; es musste aber erst durch harte Gedankenarbeit und gehöriges Abstrahieren aus dem praktischen Umgang mit elektronischen Rechnern herausdestilliert werden. Der Umgang war damals ein recht handwerkliches Betreiben von Mathematik; die meisten Mathematiker sahen darin überhaupt keine adäquate mathematische Tätigkeit.

Nach Physik- und Mathematikstudium (das Samelson im Dritten Reich verwehrt war) an der Ludwig-Maximilians-Universität München und Promotion in Theoretischer Physik bot sich für die beiden Lehramtskollegen Bauer und Samelson 1952 die große Gelegenheit, wissenschaftliche Mitarbeiter von Robert Sauer am Mathematischen Institut der Technischen Universität (damals Technische Hochschule) München zu werden und am von Hans Piloty und Robert Sauer initiierten Bau und Einsatz (Programmierung) der PERM, der Programmgesteuerten Elektronischen Rechenanlage München, von Anbeginn mitzuwirken. Die ersten Ergebnisse wurden auf der Göttinger Rechenmaschinentagung 1953 vorgestellt und in Journalartikeln 1953, 1954 ausgeweitet [BaSa53, BaSa54]. Es ging zunächst um Bedeutung nichtnormalisierter Gleitkommarechnung und Kompaktifizierung von Elementarbefehlen; für spätere Übersetzertechnik war besonders wegweisend die tiefe Auseinandersetzung mit Unterprogrammtechnik und automatisierter Adressmodifikation in Fortführung der Gedanken von H. Goldstine und J. v. Neumann von 1947 [GovN47].

2 K. Samelsons Dresdener Vortrag 1955

Einen ganz entscheidenden Schritt machte Samelson in seinem Vortrag „Probleme der Programmierungstechnik“ 1955 auf dem Internationalen Kolloquium über Probleme der Rechentechnik in Dresden [Sam55/57]. Die PERM war fertiggestellt und lief, für Bauer und Samelson war es das Signal, sich von den konkreten Maschinen zu lösen und abstrakter zu werden. Abstraktion führt nämlich nicht zu Unbrauchbarkeit, im Gegenteil, sie führt zum Eigentlichen, auch in der Informatik. Samelson forderte, „[. . .] die Programmierung soweit wie nur irgend möglich der in jahrhundertelanger Entwicklung entstandenen Symbolik der Mathematik anzugleichen“. Dieser Gedanke, nämlich die Entwicklung und die Programmierung der realen Computer ganz und gar in die große mathematische Tradition stellen zu sollen und das auch zu können, wurde hier wohl zum ersten Mal so deutlich formuliert.

Bauer und Samelson griffen den Rutishauser’schen Gedanken von 1951 [Rut51] des sog. Superplans auf, um diesen Gedanken erheblich zu konkretisieren. Ein Superplan war ein Übersetzerprogramm, ein Compiler, der aus sog. höherem Pseudocode (aus Pseudoprogrammen) Maschinenprogramme erzeugte. Samelson wollte Superpläne mit den folgenden drei Hilfsmitteln realisieren:

- Mit Bibliotheksprogrammen als geschlossenen Unterprogrammen,
- mit Teilprogrammen als offenen Unterprogrammen,
- mit den wohl von Wilkes [Wil53] erstmalig angegebenen algebraischen Adressen, später symbolische Adressen genannt.

Samelson zog den interessanten Vergleich, dass die offenen Unterprogramme gut zur Verdrahtung geeignet seien, während die geschlossenen zu komplex seien, sodass man deren Verdrahtung nicht mehr geschafft habe. Und die Maschinenadressen zu symbolischen Adressen habe man in sog. Telephonbüchern zu suchen.

Bemerkenswert war, dass er die Grundprobleme der Programmierungstechnik zum Bau von programmerzeugenden Superplänen als gelöst ansah und den intellektuellen Mut zum Propagieren universeller höherer Programmiersprachen aufbrachte: „Wir kommen zum Pseudoprogramm und damit zu *dem* Problem der Programmierungstechnik. Der Grund für diese kühn klingende Behauptung ist folgender: Mit der Verwendung von Pseudoprogrammen hat man sich vom speziellen Maschinencode völlig gelöst [...]. [Bisher] nicht gelöst ist das Problem der Verständigung zwischen den verschiedenen um je eine Maschine gruppierten Rechenzentren. Und dieses Problem bedarf unbedingt einer Lösung, die nur in der Einführung eines universellen Pseudocodes bestehen kann.“ Für Bauer und Samelson war völlig klar, woher ein solcher universeller Pseudocode zu nehmen sei. Es gebe den Kern des Codes ja bereits seit 500 Jahren bei den Rechenmeistern des ausgehenden Mittelalters und der Renaissance. Wir alle haben die arithmetische Formelsprache seit der Sexta kennengelernt. Samelson schrieb insbesondere, dass „die Programmierung, also die formale Seite des Maschinenrechnens, die Entwicklung der Mathematik an sich völlig wiederholt [...]“. Wenn der universelle Pseudocode von den Rechenmaschinen gänzlich gelöst werde, dann stehe i. Allg. keine Maschine zur Verfügung, sodass er durch Übersetzen in bekannten Maschinencode interpretiert werden könne. Es müsse aber eine Interpretationsmöglichkeit geschaffen werden. Für Bauer und Samelson war die Semantik der arithmetischen Formeln durch deren Auswertung oder Durchrechnung klar definiert, die alle schulmathematisch gebildeten Menschen im Prinzip in gleicher Weise durchführen. Das reiche aus, um arithmetische Formelsprache und selbst kompletten universellen Pseudocode weltweit zu verstehen. Dazu brauche man keine konkreten Maschinen zu kennen.

Samelson diskutierte dann das systematische Auswerten arithmetischer, teilweise geklammerter Formeln in der wohlbekannten Infixnotation, um daraus Rezepte für richtiges Interpretieren bzw. Übersetzen zu gewinnen. Samelsons Auswerten ist ein Formelabbauen, wobei er immer wieder die linkeste Abbaumöglichkeit sucht und ausnutzt. Dass so ein Procedere funktioniert und semantikgemäß erlaubt ist, ist damals selbst Fachleuten durchaus nicht geläufig gewesen. Denn Backus et al. [Bac$^+$57] mit ihrem 1954–57 gebauten FORTRAN-Compiler propagierten, zuerst fehlende Klammern einzufügen und dann sukzessive innerste Klammerpaare abzubauen, wie es schon Rutishauser 1951 [Rut51] lehrte.

Woher rührt die Unentschiedenheit über bestmögliches Vorgehen? Die schon in der Schule gelernten Klammereinsparungs- und Vorrangregeln wie

- Potenzieren $\uparrow$ geht vor Multiplizieren $\times$ und Dividieren $/$,
- diese gehen vor Addieren $+$ und Subtrahieren $-$,
- bei gleichem Rang von Operatoren geht (meist) links vor rechts,
- und geklammerte Operanden haben Vorrang vor zugehörigem Operator

erlauben nämlich durchaus nichtdeterministisches Auswerten. Z. B. hat die Formel

$$a - 5 + c \times (12 - e) + f \uparrow 2$$

15 verschiedene korrekte Auswertungsreihenfolgen mit gleichem Endergebnis. Wenn die Variablen a, c, e, f die Werte $12, 2, 3, 3$ haben, dann ist stets 34 das Endergebnis. Bei vollständiger Klammerung, etwa in

$$(\ (\underset{7}{\underline{(a-5)}} + (c \times \underset{9}{\underline{(12-e)}}\ )\ ) + \underset{9}{\underline{(f \uparrow 2)}}\ ),$$

ist die Auswertungskonfluenz (in der Theorie der λ-Kalküle spricht man von Church-Rosser-Eigenschaft) völlig evident, weil gefährliche Überlappungen, die zu Mehrdeutigkeiten führen können, ausgeschlossen sind. Gleichgültig in welcher Reihenfolge die lokalen Gipfel des Rutishauser'schen Klammergebirges abgetragen werden, gelangt man stets zu gleicher Normalform, zu gleichem Endergebnis. Bauer und Samelson haben den großen Hin-und-her-Leseaufwand aber nicht mitmachen wollen, weil an den eingefügten Klammern niemand interessiert ist. Die wegweisende Einsicht ist gewesen, dass hinter den Klammereinsparungs- und Vorrangregeln Auswertungsregeln, d. h. echt verkürzende Ersetzungs- oder Reduktionsregeln, stehen, die genau die gleichen Auswertungsschritte wie bei vorheriger vollständiger Klammerung ohne gefährliche Überlappungen zulassen.

$$\underset{\substack{< \quad \geq \\ 7}}{\underline{a-5}} + c \times (\ \underset{\substack{< \quad \geq \\ 9}}{\underline{12-e}}\ ) + \underset{\substack{< \quad \geq \\ 9}}{\underline{f \uparrow 2}}$$

$$\begin{array}{c} \text{- - - -} \\ \geq \quad < \end{array}$$

So z.B. kommt die überlappende Addition $5 + c$ nicht zum Zuge, sogar aus zwei Gründen, wobei einer genügte: Der linke Nachbaroperator $-$ hat gleiche oder höhere Priorität, der rechte $\times$ hat echt höhere Priorität als der Operator $+$.

1955 spricht Samelson noch nicht vom Kellern, aber das Auftreten von Zwischenresultaten $H_i, H_{i+1}, \ldots$ (abgelegt im linearen Hilfsspeicher H), deren letztgewonnenes Resultat als erstes seinen Speicherplatz wieder freigibt (*last-in-first-out*-pulsierendes Verhalten), ist bereits klare Beschreibung des Begriffs Zahlkeller, der zuerst in der Patentschrift [BaSa57] „Verfahren zur automatischen Verarbeitung von kodierten Daten und Rechenmaschine zur Ausübung des Verfahrens“ 1957 von Bauer und Samelson auftritt.

3 F. L. Bauers und K. Samelsons Patentschrift 1957

Die o. g. Formel mit den erwähnten Variablenwerten führt zu folgenden sukzessive gelesenen und abgebauten Formelvorderteilen mit zuletzt gelesenen Symbolen, wobei es aus technischen Gründen zweckmäßig sein kann, ein(en) Formelvorderteil(-kellerinhalt) in Operations- und Zahlkellerinhalt aufzuspalten, wie es die Patentschrift empfiehlt:

bisher gelesenes und abgebautes Formelvorderteil	zuletzt gelesenes Symbol	Operationskellerinhalt	Zahlkellerinhalt
$a - 5$	$+$	$-$	$a\,5$
7	$+$	$\emptyset$	7
$7 + c \times (12 - e$	$)$	$+ \times (-$	$7\,c\,12\,e$
$7 + c \times (9$	$)$	$+ \times ($	$7\,c\,9$
$7 + c \times 9$	$+$	$+ \times$	$7\,c\,9$
$7 + 18$	$+$	$+$	$7\,18$
25	$+$	$\emptyset$	25
$25 + 3 \uparrow 2$	$\emptyset$	$+ \uparrow$	$25\,3\,2$
$25 + 9$	$\emptyset$	$+$	$25\,9$
34	$\emptyset$	$\emptyset$	34

In der Patentschrift geht es um die Erfindung eines Rechenautomaten, der gegenüber den bisherigen maschinenprogrammgesteuerten Rechenanlagen sog. formelgesteuert ist, d. h. gesteuert von Programmen einer höheren Programmiersprache mit Ergibtanweisungen, aufgebaut aus mathematischen Formeln in üblicher Infixnotation, Zahlen, Wahrheitswerten, einfachen und indizierten Variablen. Abbildung 1 zeigt den Schaltplan eines formelgesteuerten Rechenautomaten für eine einfachste Programmiersprache, deren Programme nur arithmetische Formeln mit Zahlenwerten als Operanden sind. Ich zitiere aus Spalte 2 von [BaSa57]: „Die Erfindung beruht i. w. auf dem Gedanken, den Komponenten einer Rechenmaschine einen Analysator beizuordnen, dem die mathematischen Formeln in üblicher Schreibweise zugeführt werden. [...]. [D]ie den einzelnen Zeichen entsprechenden Signale [werden] in der Reihenfolge der Aufschreibung dem Analysator zugeführt und in diesem [...] entsprechend der Reihenfolge des Eingangs geprüft, ob die Operationen sofort ausführbar sind oder ob der Eingang weiterer Signale abgewartet werden muß; in diesem letzteren Falle werden die noch nicht verarbeitbaren Zeichen in einen Speicher (Keller) eingeführt; beim Eintreffen neuer Zeichen im Analysator, die die Ausführung einer Operation mit gespeicherten Zeichen ermöglichen, werden diese gespeicherten Zeichen in der durch die Art der Einführung festgelegten, umgekehrten Reihenfolge entnommen und verarbeitet“.

Das oberste Geschoss des Operationskellers nennt die Patentschrift Diskriminatorregister D. Am Ausgang des Vorentschlüsslers, im Entschlüsselungsregister E findet sich das gerade neu gelesene Symbol ein. Hier haben wir das Kennzeichnende des Bauer-Samelson'schen Kellerprinzips und Kellerautomaten: Jedes Paar von Inhalten von D und E entscheidet über die nächstfolgende Handlung des Operationsumsetzers (ausgebildet als Matrixschaltung Abbildung 2), d. h. ob und wie oberste Kellereinträge modifiziert werden, ob und welche Maschinenoperationen ausgeführt werden, ob weitergelesen wird oder ob der Auswertungsprozess als erfolgreich beendet angesehen werden kann.

Technikhistorisch interessant ist die folgende Beobachtung: Obwohl in den Jahren 1955–57 die Idee des Compilierens von höheren Programmiersprachen nach Maschinencode bekannt ist, halten die Autoren Bauer und Samelson wie auch die kooperierenden Kollegen auf der elektrotechnischen Seite, Hans und Robert Piloty (letzterer später an der TH Darm-

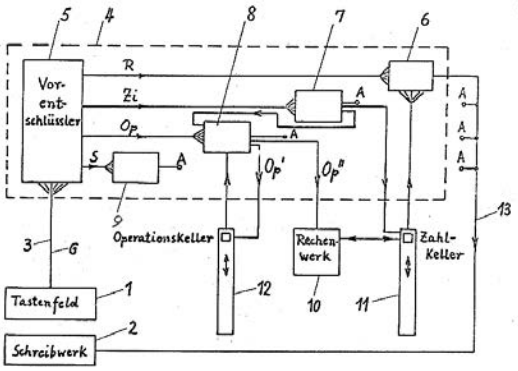


Abbildung 1: Schaltplan eines formelgesteuerten Rechenautomaten

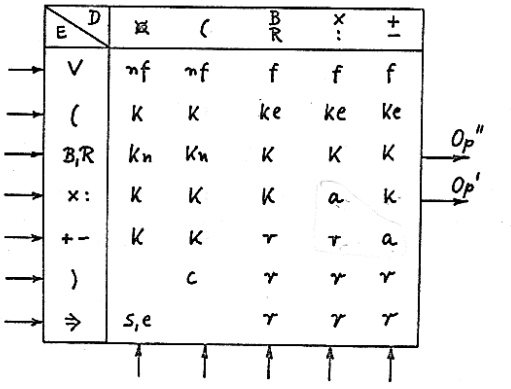


Abbildung 2: Matrixschaltung

stadt), die damals mögliche maschinelle Unterstützung durch Speicherkapazitäten und Rechengeschwindigkeit kaum für ausreichend, um real einsatzfähige Compiler herzustellen. Das Vertrauen in das Verstehen komplexer Schaltnetze für einen formelgesteuerten Rechenautomaten und in die materielle Umsetzungsmöglichkeit ist bei den Elektro- und Nachrichtentechnikern sehr hoch gewesen. Die Pläne für den entsprechenden Ausbau der PERM wurden dann aber doch nicht umgesetzt. In Hardware realisiert wurde der von Angstl und Bauer in den 1950er-Jahren konzipierte Relaisrechner STANISLAUS [Bau60, Bau77], der aussagenlogische, in polnischer Notation geschriebene Formeln auswertet (1950 entworfen als Mechanik, 1957 als Relaisrechner vorgeführt).

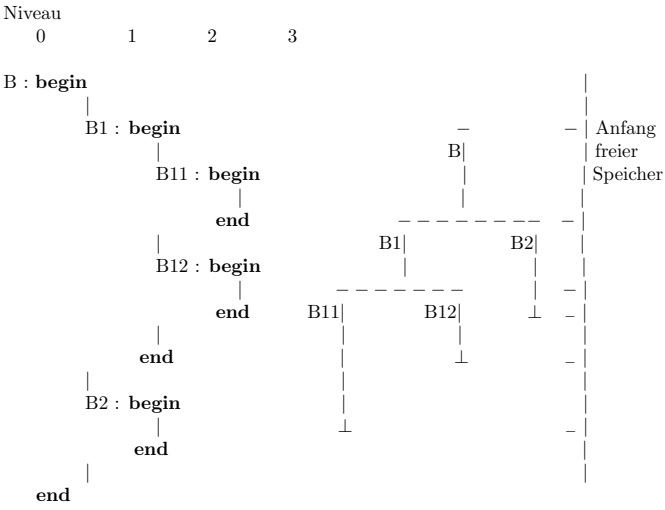
4 K. Samelsons und F. L. Bauers Artikel „Sequentielle Formelübersetzung“ 1959

Bauers und Samelsons Kellerprinzip und Kellerautomat sind 1957 noch nicht recht publik geworden, weil der Begriff „Keller“ eben nur in der Patentschrift stand und die Patentrichtlinien verlangten, dass über den Inhalt nur nach Auslegung der Schrift weiter publiziert werden dürfe. Daher ist das Bauer-Samelson'sche Kellerverfahren erst in dem 1959er Artikel [SaBa59] „Sequentielle Formelübersetzung“ in den „Elektronischen Rechenanlagen“ voll wahrgenommen worden. Wegen der Wichtigkeit für die Automatentheorie zitiere ich wörtlich: „In der Sprechweise der Theorie der Automaten kann das Prinzip so formuliert werden: Durch die gesamte Besetzung des Kellers wird ein Zustand (des Übersetzungsvorgangs) definiert, der effektiv in jedem Augenblick nur von dem obersten Kellerzeichen abhängt, und neu gelesene Information plus Zustand bestimmen die Aktionen [...]“. Die Autoren setzen nun ganz und gar auf das Übersetzen statt das Interpretieren. Vor ihrem geistigen Auge stand die international konzipierte höhere Programmiersprache ALGOL 58, 1958 definiert durch ein ACM/GAMM-Komitee, wofür Samelson zusammen mit A. J. Perlis den Bericht schrieb und herausgab [PeSa58/59].

In Samelsons Dresdener Vortrag hat es nicht bloß den einen deutlichen Hinweis auf kellerartiges Verhalten gegeben, nämlich das erwähnte Zahlkellerverhalten (ohne Erwähnung des Wortes „Keller“) beim Auswerten und Durchrechnen arithmetischer Formeln. Darüberhinaus lässt sich das Pulsieren des Zahlkellers zur Laufzeit auf den Datenspeicher für Teil- und Unterprogramme ausdehnen. Das tat auch Samelson und schrieb 1955: „Was in jedem Moment noch an für Zwischenrechnungen freien Speicherzellen zur Verfügung steht, hängt von der ganzen Vorgeschichte des Programms bis zum betrachteten Moment ab und läßt sich kaum vernünftig vom Compiler am Anfang festlegen. Es bleibt hier wohl nichts anderes übrig, als jeweils beim Übergang in ein neues abgeschlossenes Teilprogramm festzustellen, von wo ab der Speicher für Rechnungen frei ist, und diese Angabe dem Teilprogramm etwa in der Weise mit auf den Weg zu geben, daß man sie unter einem festen Variablennamen im ‚Telephonbuch‘ eintragen läßt, so daß also die zu der Variablen ‚Anfang freier Speicher‘ gehörige Rufnummer = Speicheradresse sich während der Rechnung laufend ändert“. Rechnungen eines neu aktivierten abgeschlossenen Unterprogramms brauchen insbesondere Speicherplätze für seine unterprogrammeigenen Variablen (einfache Variablen und Felder (arrays) wechselnder Größe), für Parameter (Argumente), Zwischenergebnisse, Unterprogrammresultat und Rückkehradresse.

Die wesentlichen Züge der Behandlung von Funktionen und Prozeduren werden in dem Artikel „Sequentielle Formelübersetzung“ [SaBa59] in Abschnitt „6. Vereinbarungen (declarations)“ diskutiert, wenn auch verhältnismäßig knapp; die Autoren sehen sich dazu gerechtfertigt, weil sie auf den Samelson'schen Dresdener Vortrag 1955/57 verweisen können. Sie schreiben 1959: „[. . .] Prozedur-Vereinbarungen [. . .] [führen] zu dynamischen [Unterprogrammen]. Dem aus der Auswertung der Vereinbarung resultierenden Unterprogramm ist also wie bei allen Unter- bzw. Bibliotheksprogrammen ein Anschlußteil voranzustellen, der die Übernahme der Rückkehradresse und der Programmparameter durchführt, im dynamischen Fall ist ein Adaptieren zur Berechnung des benötigten Hilfsspeichers und zur Adressierung der auf den Hilfsspeicher bezüglichen Befehle (mit Hilfe eines speziellen Parameters, der den Beginn des freien Speichers angibt) hinzuzufügen. Auch hier handelt es sich um bekannte Tatsachen, auf die nicht näher eingegangen zu werden braucht.“

Die Niveaustruktur von Blöcken und die damit einhergehende pulsierende Speicherverteilung gehören zu den wichtigsten Beiträgen Samelsons zur Programmiertechnik. Er brachte die Blockstruktur in ALGOL 60 [Nau+60] ein. Im ALGOL 58-Bericht [PeSa58/59] war noch keine Rede von Bindungs- und Gültigkeits-(Sichtbarkeits-)Bereichen von Identifikatoren, Phänomenen, die in Prädikatenlogik und λ-Kalkülen seit den 1930er-Jahren bekannt waren [Her61]. In einem ALGOL 58-Programm kann jeder Identifikator in nur einer Bedeutung auftreten. Daraus folgt für ALGOL 60, dass in parallelen Blöcken verschiedene einfache Variablen, Felder und Hilfsvariablen für Zwischenwerte gleiche Speicherplätze belegen dürfen. Das folgende Bild über Niveaustruktur von Blöcken und pulsierende Speicherverteilung stammt aus dem Buch von Bauer, Heinhold, Samelson, Sauer „Moderne Rechenanlagen“ 1965 [BHSS65, Lan02]:



Samelsons Vision über das Blockkonzept brachte die Mehrheit des ALGOL 60-Komitees dazu, dass sie sich von Samelson und Mitstreitern wie J. McCarthy, E. W. Dijkstra vom statischen Binden von Identifikatoren überzeugen ließen. Es wird dadurch zum Ausdruck

gebracht, dass bei Prozedurrumpf- und Parameterersetzungen (operationelle Kopierregelsemantik) Bindungsverfälschungen durch gebundene Identifikatorumbenennungen zu vermeiden waren. Sog. dynamisches Binden macht solche impliziten Umbenennungen nicht. Diese Überlegungen haben zur Folge, dass Programme mit Prozeduren sich als Programme mit i. Allg. unendlich tief geschachtelten generierten (modifiziert kopierten) Blöcken ohne Prozeduren und ohne Prozeduranweisungen auffassen lassen. Diese Sicht bietet eine Rechtfertigung für pulsierend arbeitende Laufzeitkeller. Rückblickend wären explizite Hinweise im ALGOL 60-Bericht auf α- und β-Reduktion in λ-Kalkülen für Programmierer und Übersetzerkonstrukteure sehr nützlich gewesen.

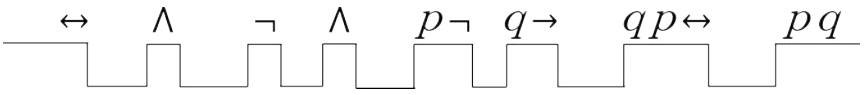
5 Vorläufer des Bauer-Samelson'schen Kellerprinzips

Wie steht es um Vorläufer des Bauer-Samelson'schen Kellerprinzips? 1950 entwickelt H. Angstl [Bau60] eine Mechanik, um klammerfreie aussagenlogische Formeln mit Symbolen der Stelligkeit ≥ 0 in polnischer Notation [Luk29] auf Wohlgeformtsein zu prüfen (vorgetragen 1950 im Logistikseminar von Prof. Britzelmayr an der LMU München). Angstl setzt die Symbole einer zu prüfenden Formel wie Spielsteine in eine Reihe, wobei je zwei aufeinanderfolgende Steine einen Zwischenraum der Größe der Stelligkeit des vorangehenden Symbols lassen. Der Boden jeden Zwischenraums wird abgesenkt um die Höhe der Spielsteine, sodass jeder Zwischenraum zur Fallgrube wird.

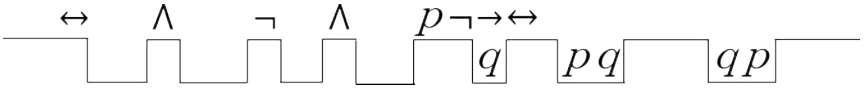
Beispiel: Die Tautologie

$$((\neg (p \wedge \neg q) \wedge (q \rightarrow p)) \leftrightarrow (p \leftrightarrow q))$$

in polnischer Notation



Die Spielsteine werden von hinten nach vorn geschoben. Trifft der vorderste geschobene Stein auf eine Fallgrube der Größe n, fallen die vorderen n geschobenen Steine hinein, sodass die übrigen darüber hinwegrutschen. Dabei schiebt man eine „kellerartig" pulsierende Serie von Spielsteinen vor sich her.



Wenn am Schluss alle Fallgruben vollgefüllt sind und ganz vorne nur noch exakt ein Stein im „Keller" übrig ist, genau dann ist die vorgelegte Formel wohlgeformt.

Dies ist im Grunde genau die Burks-Warren-Wright'sche mathematischer formulierte nichtrekursive Definition einer wohlgeformten Formel [BWW54] von 1954: Jedes Symbol bekommt ein

$$\textit{Gewicht} = 1 - \textit{Stelligkeit},$$

und eine Symbolreihe bekommt als *Gewicht* die Summe der Einzelgewichte. Eine nichtleere Symbolreihe heißt dann *wohlgeformte Formel*, wenn alle nichtleeren Postfixe positives Gewicht haben und das *Gesamtgewicht* 1 ist. Das positive Gewicht eines Postfix ist gerade die Länge des zugehörigen Angstl'schen Kellerinhalts, z. B. 4 ist die Länge von $p \neg \rightarrow \leftrightarrow$. Liest man eine Formel in polnischer Notation von hinten, dann reduziert sich der Bauer-Samelson'sche Operationskeller auf ein einziges Operationsregister.

Die rekursive Definition des Wohlgeformtseins einer Formel ist allerdings deutlich verständlicher und natürlicher als die obige nichtrekursive Definition. Die Äquivalenz beider Definitionen bedarf eigentlich eines Beweises, worauf weder Angstl noch Burks, Warren und Wright eingehen. Die rekursive Definition bietet aber kein unmittelbar deterministisches Entscheidungsverfahren.

1952 beschreibt W. L. van der Poel [vdP52] einen einfachen Rechner, dessen Befehle auch eine last-in-first-out-Organisation von Rückkehradressen bei wiederholt aufgerufenen Unterprogrammen erlauben. Das Charakterisierende eines Kellerautomaten ist aber nicht ganz einfach zu erkennen und wird nicht explizit formuliert.

S. Ginsburg unterscheidet in seinem Buch „The Mathematical Theory of Context-free Languages" 1966 [Gin66] zwischen bloßer Verwendung eines Kellerspeichers (pushdown tape) und dem Begriff Kellerautomat (pushdown automaton oder acceptor). Ein Kellerspeicher trete zuerst bei Burks, Warren und Wright 1954 auf und komme 1959/60 auch bei Bauer und Samelson vor. Der Kellerautomat sei dagegen zuerst von N.Chomsky [Cho62] formalisiert worden. Unser Darmstädter Kollege und anerkannter Automatentheoretiker Hermann Walter bestand aber in einer Diskussion der 1970er-Jahre [Wal70] darauf, dass F. L. Bauer und K. Samelson den Kellerautomaten 1957 erfunden hätten.

Dank: Ich danke Herrn Kollegen Friedrich L. Bauer für seinen Vorschlag, ich möge diesen Vortrag im Jenaer „Kellerkolloquium" halten. Ferner danke ich Frau Annemarie Langmaack für das Setzen dieses Artikels in LaTeX.

Literatur

[Bac+57] Backus, J. W., et al.: The FORTRAN Automatic Coding System. Proc. Western Joint Computing Conf. 11, 188-198 (1957)

[Bau60] Bauer, F. L.: The Formula-Controlled Logical Computer "Stanislaus". Math. Tab. a. Oth. Aids to Comp., Vol. XIV, 69, January (1960)

[Bau77] Bauer, F. L.: Angstl's Mechanism for Checking Wellformedness of Parenthesis-Free Formulae. Math. of Computation 31, 318–320 (1977)

[BaSa53] Bauer, F. L., Samelson, K.: Optimale Rechengenauigkeit bei Rechenanlagen mit gleitendem Komma. Z. Angew. Math. Phys. 4, 312–316 (1953)

[BaSa54] Bauer, F. L., Samelson, K.: Maßnahmen zur Erzielung kurzer und übersichtlicher Programme für Rechenautomaten. Z. Angew. Math. Mech. 34, 262–272 (1954)

[BaSa57] Bauer, F. L., Samelson, K.: Verfahren zur automatischen Verarbeitung von kodierten Daten und Rechenmaschinen zur Ausübung des Verfahrens. Patentanmeldung Deutsches Patentamt (1957)

[BHSS65] Bauer, F. L., Heinhold, J., Samelson, K., Sauer, R.: Moderne Rechenanlagen. Stuttgart: Teubner (1965)

[BWW54] Burks, A. W., Warren, D. W., Wright, J. B.: An Analysis of a logical machine using parenthesis-free notation. Mathematical Tables and Other Aids to Computation, vol. 8, pp. 53–57 (1954)

[Cho62] Chomsky, N.: Context-free grammars and pushdown storage. M.I.T. Res. Lab. Electron. Quart. Proc. Rept. 65 (1962)

[Gin66] Ginsburg, S.: The Mathematical Theory of Context Free Languages. McGraw-Hill (1966)

[GovN47] Goldstine, H., von Neumann, J.: Planning and Coding for an Elektronic Computing Instrument. Institute for Advanced Study: Princeton (1947)

[Her61] Hermes, H.: Aufzählbarkeit, Entscheidbarkeit, Berechenbarkeit. Berlin Göttingen Heidelberg: Springer (1961)

[Kaem60] Kämmerer, W.: Ziffernrechenautomaten. Akademie Verlag Berlin (1960)

[Lan02] Langmaack, H.: Klaus Samelsons frühe Beiträge zur Informatikentwicklung. Informatik Spektrum 18 (April 2002), 132–137 (2002)

[Luk29] Lukasiewicz, J.: Elementy Logiki Mathematycznej, Warszawa (1929)

[Nau$^+$60] Naur, P., (ed.) et al.: Report on the Algorithmic Language ALGOL 60. Num. Math. 2, 106–136 (1960)

[PeSa58/59] Perlis, A. J., Samelson, K. (eds.): ACM Committee on Programming Languages and GAMM Committee on Programming. Report on the Algorithmic Language ALGOL. Num. Math. 1, 41–60 (1959)

[Rut51] Rutishauser, H.: Über automatische Rechenplanfertigung bei programmgesteuerten Rechenanlagen. Z. Angw. Math. Mech. 31, 255 (1951)

[Sam55/57] Samelson, K.: Probleme der Programmierungstechnik. Intern. Koll. über Probleme der Rechentechnik, Dresden 1955. Berlin: VEB Deutscher Verlag der Wissenschaften, S. 61–68 (1957)

[SaBa59] Samelson, K., Bauer, F. L.: Sequentielle Formelübersetzung. Elektr. Rechenanl. 1 (4), 176–182 (1959)

[SaBa60] Samelson, K., Bauer, F. L.: Sequential Formula Translation. CACM, vol. 3, 76–83 (1960)

[vdP52] van der Poel, W. L.: Dead Programmes for a Magnetic Drum Automatic Computer. Appl. Sci. Res. B. 3: 190–198 (1952)

[Wal70] Walter, H.: Persönliche Mitteilung. Saarbrücken 1970er-Jahre

[Wil53] Wilkes, M. V.: The Use of a Floating Address System for Orders in an Automatic Digital Computer. Proc. Cambridge Philos. Soc. 49, 84–89 (1953)