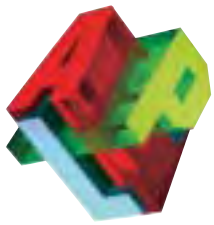
APL-Journal

A Programming Language

2/2004

In dieser Ausgabe:



Wolfgang Everling
APL
Algebraische oder algorithmische Programmiersprache?

Reiner Nußbaum
Erprobte Schnittstelle
Sicherer Transport von Zahlungsverkehrsdaten aus dem Beleghaften Zahlungsverkehr mit gift-ALU

Axel Holzmüller
APL und Java:
vorwärts und rückwärts

Im Blickpunkt

Herbert Voß
LATEX

APL-Germany e.V. Jahrgang 23 Nr. 2 2004 ISSN - 1438-4531 RHOMBOS-VERLAG

Inhaltsverzeichnis

Wolfgang Everling

APL
Algebraische oder algorithmische Programmiersprache? 4

Reiner Nußbaum

Erprobte Schnittstelle
Sicherer Transport von Zahlungsverkehrsdaten aus dem Beleghaften Zahlungsverkehr mit gift-ALU 8

Herbert Voß

LATEX - Flotter Oldie
25 Jahre und kein Ende 12

Axel Güth

Universalgenie
Workstation APL2 Version 2.0 17

Nancy Wheeler

Workstation APL2 Version 2
Service Level 6 21

Axel Holzmüller

APL und Java:
vorwärts und rückwärts 23

Romilly Cocking

SmartRec
A SmartArrays application at BNP Paribas 27

Aufnahmeformular 30

Impressum 2

Editorial 3

Nachruf Ken Iverson 11

Nachrichten 30

www.apl-germany.de

www.rhombos.de

4 Algebraische oder algorithmische Programmiersprache?

Im Herbst 1960 trug Kenneth E. Iverson PhD dem ersten Lehrgang des IBM Systems Research Institute in Manhattan sein Konzept einer Programmiersprache APL ('A Programming Language') vor. Auf mich als Mathematiker übte es eine starke Faszination aus. Im Gegensatz zum FORTRAN (Formula Translator) von John F. Backus wies Iverson zum ersten Mal über die Arithmetik und Arrayverarbeitung hinaus, indem er eine Vielzahl weiterer Operationen ohne Bindung an bestimmte Datentypen einführte. Zugleich wurden wie selbstverständlich Datenstrukturen als weitere Typen zugänglich gemacht und die Operationen auf sie ausgedehnt. Die so entstandene Sprache war verblüffend knapp und flexibel.

8 Erprobte Schnittstelle

Bei der Umstellung des Transfers von Zahlungsverkehrsdaten auf schnelle Intranet Netzwerke muß die Sicherheit des Datentransportes gewährleistet sein. Die Dateien werden mit der Verschlüsselungsmethode von FLAM verschlüsselt. Da FLAM auch auf dem HOST verfügbar ist, können die Daten dort entschlüsselt und weiterverarbeitet werden. Ein Datenaustausch erfolgt in beide Richtungen. Von den Zugangssystemen für den Beleghaften Zahlungsverkehr als Datenlieferanten werden Zahlungsverkehrsdateien angeliefert. Diese Daten werden gebündelt, komprimiert und verschlüsselt und anschließend an geeignete Hostsysteme zur Verarbeitung übertragen. Diese übertragen Quittungsinformationen und Bestandsdaten, die an die Zahlungsverkehrssysteme weiterzuleiten sind. Jeder Verarbeitungsschritt wird protokolliert und zwischengespeichert. Als zeitnahe Arbeitsmethode wird als Automationsverfahren gift-ALU eingesetzt, ein ereignisgesteuertes Automationsverfahren für Unix, Linux und Windows.

12 LATEX - Flotter Oldie

Das Textsatzsystem TeX ist älter als der PC, aber immer noch höchst lebendig. So ist das Erstellen von PDFs in den vergangenen Jahren erheblich leichter geworden, der Anwender hat die Wahl zwischen viel mehr Schriftarten, und sogar interaktive PDF-Formulare lassen sich neuerdings mit pdfTeX erstellen.

21 Workstation APL2 Version 2 Service Level 6

In May 2005, Service Level 6 to Workstation APL2 Version 2 was made available. In addition to the usualfixes for reported problems, this service level contains several interesting enhancments that address re-quirements submitted to IBM by the German customers of APL2.

Impressum

ISSN 1438-4531

Herausgeber: Dipl.-Math. Dieter Lattermann, Walldorf

Redaktion: Dipl.-Volksw. Martin Barghoorn (verantwortlich)
Technische Universität Berlin
Franklinstr. 28
D-10587 Berlin
Tel. (030) 314 24392
Fax (030) 314 25901
eMail: barg@cs.tu-berlin.de

Erscheinungsweise: halbjährlich

Erscheinungsort: Berlin

Verlag: RHOMBOS-VERLAG, Berlin
Kurfürstenstr. 17
D-10785 Berlin
Tel. (030) 261 9461
Fax (030) 261 6300
eMail: verlag@rhombos.de,
Internet: www.rhombos.de

Gestaltung: RHOMBOS-VERLAG, Berlin

Druck: dbusiness.de GmbH, Berlin

Copyright: *APL Germany e.V.*
(für alle Beiträge, die als Erstveröffentlichung erscheinen)

Fotos: M. Barghoorn (Titel, Seite 4), FU-Berlin (S. 22), Pixelquelle (S. 23), Fraunhofer IMK (S. 30), Dieter Klein (S. 31)

Homepage des APL-Germany e.V.: http://www.apl-germany.de
Homepage des APL-Journals beim Verlag: http://www.rhombos.de/shop/a/show/article/?99

Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung von Redaktion und Herausgeber nicht übernommen werden. Sämtliche Veröffentlichungen im APL-Journal erfolgen ohne Berücksichtigung eines eventuellen Patentschutzes. Warennamen werden ohne Gewährleistung einer freien Verwendung benutzt.

Mit Namen gekennzeichnete Artikel geben nicht unbedingt die Meinung des Herausgebers oder der Redaktion wieder. Für unverlangte Einsendungen wird keine Haftung übernommen. Nachdruck ist nur mit Zustimmung des Herausgebers sowie mit Quellenangabe und Einsendung eines Beleges gestattet. Überarbeitungen eingesandter Manuskripte liegen im Ermessen der Redaktion.

Wie immer freuen wir uns auf Ihr Feedback zu diesem Heft. Schreiben Sie uns an barg@cs.tu-berlin.de, oder faxen Sie an: 030 - 314 25901.

Technische Hinweise für Autorinnen und Autoren

Manuskripte werden von der Redaktion entgegengenommen. Sie müssen im Besitz der Nutzungsrechte für die von Ihnen eingereichten Texte, Fotos und Zeichnungen sein. Die Zustimmung zum Abdruck wird vorausgesetzt.

Manuskripte und Beiträge aller Art werden in jeder Form entgegengenommen. Es werden jedoch Beiträge in gespeicherter Form bevorzugt. Bitte speichern Sie Ihren Text im Format Ihrer Textverarbeitung und zusätzlich als RTF-Datei (Rich Text Format). Sie brauchen sich nicht die Mühe machen, Ihren Text aufwendig zu formatieren. Auszeichnungen im Text sollten sich auf „Kursiv“ beschränken. Ihre Datei schicken Sie der Redaktion bitte auf einem Datenträger (Diskette oder CD-ROM). Alternativ hierzu können Sie der Redaktion Ihre Daten auch per Email (barg@cs.tu-berlin.de) schicken.

Wenn Sie Ihr Manuskript einreichen, sollten Sie zusätzlich eine Gliederung und Zusammenfassung Ihrer Arbeit vorlegen.

Wichtig: In der Zusammenfassung (abstract) sollte in wenigen Sätzen die wichtigste Aussage Ihres Textes wiedergegeben sein.

Grafiken

Bitte senden Sie uns einen sauberen Ausdruck der Graphiken auf rein-weißem Papier. Linien (keine Haarlinien!) und Schriften in den Graphiken sollten so groß gewählt sein, dass sie auch bei (stark) verkleinerter Wiedergabe im Heft noch zu erkennen sind; auf der Rückseite vermerken Sie bitte Ihren Namen und die Nummer der Abbildung; Zusätzlich wäre es schön, wenn Sie uns die mit einem gängigen Vektor-Grafikprogramm erstellten Zeichnungen und Grafiken auch als Datei liefern. Grafikdateien (Strichzeichnungen) sollten möglichst im Austauschformat EPS (skalierbar) mit 300 bis 600 dpi Auflösung in schwarz-weiß bzw. Graustufe abgespeichert werden. Nach Rücksprache können sie zusätzlich Ihre Grafiken in Ihrem Original-Dateiformat liefern oder in ihr Textdokument einbetten.

Digitale Fotos benötigen wir im TIF-Format mit einer maximalen Auflösung von 250 dpi. Nicht geeignet sind Abbildungen und Fotos, wenn Sie aus dem Internet stammen beziehungsweise im Original als Bildschirmdarstellung (Auflösung 75 - 150 dpi) vorliegen. Ebenfalls nicht geeignet sind Grafiken, die mit einem pixelorientierten Zeichenprogramm erstellt wurden.

EDITORIAL

Liebe APL-Freundinnen und -Freunde,

seit 1981erscheint das *APL-Journal,* seit 1991 habe ich das Vergnügen, ein paar Geleitworte zu den einzelnen Ausgaben zu schreiben. Dieses ist mein letztes Editorial, ich habe mich entschieden, es mit 14 Jahren Tätigkeit als Vorsitzender von *APL Germany* genug sein zu lassen. Schließlich wissen wir aus der Politik, dass zu langes Verharren auf einem Posten nicht gut tut. Betonen möchte ich aber, dass mir das Amt immer Spaß gemacht hat.

Ganz herzlich bedanke ich mich bei allen meinen Mitstreitern im Vorstand für die Arbeit, die sie geleistet haben. Ohne sie hätten wir keine Tagungen abhalten, keinen Kongress organisieren und kein Journal herausgeben können. Dank gebührt auch allen Referenten auf den Tagungen und den Autoren der Beiträge für das *APL-Journal.*

Dem neuen Vorstand wünsche ich glückliche Hände, dem *Journal* weiteres Gedeihen, willige Autoren und viele geneigte Leser.

Ihr

Dieter Lattermann

Wolfgang Everling

APL

Algebraische oder algorithmische Programmiersprache?

Im Herbst 1960 trug Kenneth E. Iverson PhD dem ersten Lehrgang des IBM Systems Research Institute in Manhattan sein Konzept einer Programmiersprache APL ('A Programming Language') vor. Auf mich als Mathematiker übte es eine starke Faszination aus. Im Gegensatz zum FORTRAN (Formula Translator) von John F. Backus wies Iverson zum ersten Mal über die Arithmetik und Arrayverarbeitung hinaus, indem er eine Vielzahl weiterer Operationen ohne Bindung an bestimmte Datentypen einführte. Zugleich wurden wie selbstverständlich Datenstrukturen als weitere Typen zugänglich gemacht und die Operationen auf sie ausgedehnt. Die so entstandene Sprache war verblüffend knapp und flexibel.

Erinnerungen ...

Wir übten diese neue Sprache damals mit Begeisterung – allerdings ohne die Programme ausführen zu können!

Erst einige Jahre später gab es Interpreter für APL, die die Programmeingabe von Fernschreibtastaturen aufnahmen und die Ergebnisse am selben Gerät unter die Eingabe druckten. Diese Interaktivität war Iversons Hauptanliegen geworden und hatte die Syntax seiner Sprache mitbestimmt: Als Operationszeichen dienten zunächst die im ASCII-6 Bit-Code neben den Großbuchstaben und den Ziffern verfügbaren Zeichen. Weitere Operationszeichen wurden als Folgen der Form Zeichen-Rücktaste-Zeichen eingegeben und erschienen dadurch wie mit einem Tastendruck entstanden. Vom Benutzer selbst definierte Funktionen überwanden die anfängliche Beschränkung auf die Grundoperationen. Klammerung wurde weitgehend durch eine feste Reihenfolge der Interpretation überflüssig. Das machte alle Programmtexte extrem knapp und ihre Eingabe entsprechend schnell.

Diese Eigenschaften blieben heutigen Fassungen der Interpreter und Compiler für APL erhalten oder wurden durch die Byte-Codes noch verbessert. Deshalb habe ich jetzt – nach dem Ende meiner beruflichen Laufbahn und für die wenigen einfachen Rechnungen, die mir noch zu tun bleiben – gern wieder auf die APL2 Software zurückgegriffen.

Einordnung

Mit APL trat Iverson bewußt in Konkurrenz zu FORTRAN. Seine Vertrautheit mit Begriffen und Methoden der mathematischen Logik und der allgemeinen Algebra, die in den 1920-er Jahren entstanden, ging sichtlich tiefer als bei Backus. Dieser hat in seinem ACM Vortrag 1968 'Can Programming be Liberated from the Von-Neumann Style?' selbst den Ansatz einer algebraisch formulierten Sprache mit dem Eingeständnis begleitet, er sei kein Algebraiker. Das erklärt viele Inkonsistenzen seines Vorschlags.

Aber schon Iverson konnte seine Kenntnisse nicht wirklich konsequent in die Definition von APL einbringen. Er hätte mit uralten, uns allen seit der Kindheit eingeprägten Traditionen brechen müssen. Und er hätte die mathematische Algebra bewußt zu einer effizienten Algorithmentheorie erweitern müssen, die es noch nicht gab.

Wie schwer dies Dilemma zu überwinden war, zeigt die Namensgebung der nächsten Sprachentwicklung. Ihr Ergebnis wurde zunächst als ALGEL, nämlich *Algebraic Language*, präsentiert. In ihr war eine algebraische Struktur mit einigen Datentypen und -strukturen (allgemein gesagt: Grundbereichen oder Sorten) vordefiniert. Der Benutzer konnte sie durch eigene Definitionen erweitern. Sie konnte aber nur effizient sein, wenn sie Zwischenergebnisse zu speichern und die Synchronisierung von Operationen zu kontrollieren erlaubte: die typischen Elemente von Algorithmen. Deshalb haben ihre Begründer sie nach mancher Diskussion in ALGOL, also *Algorithmic Language*, umbenannt.

Womöglich hat dabei auch die abschreckende Wirkung des Wortes 'Algebra' mitgespielt: Die Nutzung programmierter Rechenanlagen und ihrer Sprachen war damals noch im kaufmännischen Bereich massiert. Ein Franzose meint mit dem Ausdruck *'C'est de l'Algèbre'* überhaupt etwas Unverständliches!

Mir selbst sind die algebraischen Grundlagen für Aufbau und Programmierung von Rechengeräten erst beim Unterricht in Bonn richtig bewußt geworden, – ja eigentlich erst bei der Überarbeitung zweier Lehrbücher über Organisation und Architekturen von Rechnern. Ein Rezensent fand dazu eine naive Formulierung: „Es ist erstaunlich, wie viel Mathematik der Organisation von Rechengeräten unterlegt werden kann." War ihm entgangen, daß Mathematik den Begriff des Rechnens überhaupt erst formuliert? So bin ich jetzt in der Lage, einige Aspekte der Sprache APL als algebraische

Begriffe zu diskutieren und von den algorithmischen Aspekten abzugrenzen.

Funktionszeichen, Argumente und Stelligkeit

Ein auffallendes Charakteristikum von APL ist die Beschränkung seiner *basic operations* auf *unary* und *binary* oder auch *monadic* und *dyadic use*.

Moderne Darstellungen der mathematischen Logik wie *Ebbinghaus, Flum, Thomas 1986* gehen von der systematischen Syntax mathematischer Terme (oder Formeln im Sinne von FORTRAN) aus. Diese setzt eine Menge *F* von Funktionszeichen voraus. Zu jedem Zeichen ist die Anzahl seiner Argumente festgesetzt, die Null oder eine natürliche Zahl sein kann; diese Anzahl heißt die *Stelligkeit* des Funktionszeichens. *F* zerfällt so in die Mengen F_0, F_1, F_2, ... von Zeichen gleicher Stelligkeit.

APL scheint sich auf die drei genannten Teilmengen zu beschränken. Die Syntax von APL fordert, daß eine Grundoperation entweder nur ein rechts davon geschriebenes Argument oder zwei, links und rechts davon geschriebene Argumente hat. Das ist sichtlich Konzession an die Tradition der Grundschule, die ihren Kindern das Rechnen nur sprachlich (a *und* b *ist* c) plausibel machen kann.

Ohne solche Konzession läßt die systematische Syntax genau die *F*–Terme zu, die aus einem Funktionszeichen bestehen, auf das *rechts* so viele zugelassene Terme folgen, wie seiner Stelligkeit entspricht; bei einstelligen Zeichen ist das gerade der rechte Rest hinter dem Funktionszeichen, der selbst wieder ein Term ist. Das letzte Zeichen jedes Terms ist notwendig nullstellig und heißt auch ein *Konstantenzeichen*. – Tatsächlich ist es auch nur Tradition, Funktionszeichen *links* von ihren Argumenten zu schreiben; ihre Herkunft läßt sich historisch nicht ergründen. Für einen Araber, der von rechts nach links zu schreiben gewohnt ist, wirkt sie völlig anders als für uns!

Für die Grundoperationen ist in APL das rechte Argument leicht zu erkennen: der Rest rechts neben dem Zeichen. Das entspricht der systematischen Syntax. Ein etwa vorhandenes linkes Argument weicht dagegen von ihr ab. Es muß durch Klammerung abgegrenzt werden, wenn es kein Konstantenzeichen ist.

Jeder Term der systematischen Syntax kann bei Kenntnis aller Stelligkeiten eindeutig in sein Funktionszeichen und seine Teilterme zerlegt werden. Dazu genügt eine Durchmusterung von links oder von rechts her. Deswegen wird hier auch von einer klammerfreien Syntax gesprochen. Für die Syntaxprüfung einer APL-Eingabe bot sich die Musterung von links her an, also in der Reihenfolge der Eingabe.

Im folgenden Abschnitt hat *Interpretation* einen anderen Sinn als den, jedes Programm schrittweise auszuführen!

Interpretation von Termen durch algebraische Strukturen

Ihren Sinn erhalten syntaktisch richtige Terme erst im Zusammenhang mit einer zu den Zeichen *F* passenden algebraischen Struktur – oder kurz gesagt: einer *F*–Algebra. Das ist eine Vereinbarung, die 1) eine oder mehrere Mengen (aus Elementen wie Zahlenarten, Wahrheitswerten, Buchstabenfolgen...) als *Grundbereiche* definiert, für jedes Funktionszeichen 2) einen zugehörigen Grundbereich seines Ergebnisses bestimmt und 3) eine *Abbildung* vom cartesischen Produkt der zu den Argumenten gehörigen Bereiche in den Bereich des Funktionsergebnisses zuweist. Für Konstantenzeichen ohne Argument wird ein fester Wert des Ergebnisbereichs festgelegt.

So wird zum Beispiel das zweistellige Pluszeichen + in einem Monoid als dessen einzige Operation, in einer Boole'schen Algebra durch die Abbildung *Disjunktion*: B x B t B, in einer max-plus-Algebra durch die Abbildung *Maximum*, in der Arithmetik durch die Abbildung *Addition* und so weiter interpretiert. Beispiele mit mehr als einem Grundbereich finden sich in Vektoralgebren oder bei der so genannten objektorientierten Bearbeitung von Datenstrukturen. Deshalb sind algebraische Strukturen in der Informatik als 'abstrakte Datentypen' wiedererfunden worden.

Durch die Wahl einer zu ihm passenden *F*-Algebra *A* wird jedem *F*-Term ein Element aus dem Ergebnisbereich seines führenden Funktionszeichens *eindeutig* zugewiesen. Die so bestimmte Abbildung der *F*-Terme in Ergebniselemente wird *Interpretation* durch *A* genannt, ihr Wert für einen bestimmten Term kurz dessen Interpretation. Praktisches Rechnen besteht also überwiegend aus der Ermittlung von Term-Interpretationen.

Auch die *F*-Terme bilden Grundbereiche einer *F*-Algebra *T*. Für jedes Zeichen ist die Abbildung, die einer passenden Folge von Termen als Argumenten den mit dem Zeichen beginnenden Term zuordnet, schon in der Syntax ausgedrückt. Diese Algebra *T* heißt deshalb die *natürliche F*-Algebra. Ihre Interpretation durch eine *F*-Algebra *A* kann sehr knapp definiert werden, weil *T* und *A hinsichtlich der Funktionszeichen aus F gleich-strukturiert* (griechisch kurz: *F*-homo-morph) sind.

Ausführungsreihenfolge der Interpretation

Die Beziehung zwischen Termen und ihren Teiltermen wird oft durch Bäume veranschaulicht. Sie definiert eine Ordnung in *T*. Der intransitive Teil dieser Ordnung, der jeden Term nur mit seinen Argumentermen – oder auch: deren am Anfang bezeichnete Funktionen – in Beziehung setzt, wird in der Informatik *Datenfluß* genannt.

Die Definition der Interpretation legt zwar fest, daß die eines Terms aus denen seiner Argumentterme durch Ausführen einer Abbildung bestimmt werden muß. Sie läßt aber völlig offen, wie oft und in welcher Reihenfolge oder Ordnung die Teilterme interpretiert und die Abbildungen ausgeführt werden.

Zum Beispiel ist es in einer Gatterschaltung – wie etwa einem Addierwerk – den einzelnen Gattern überlassen, wie oft sie aus den zugeführten Argumenten 'ihre' Gatterfunktion bestimmen. Geschieht das an den Ausgängen der Schaltung mehr als einmal, so gilt die Schaltung als unsicher (*hazardous*). Jedenfalls muß vor der weiteren Verarbeitung des Resultats abgewartet werden, bis sich alle einzelnen Gatter auf die eindeutig richtige Interpretation eingestellt haben. Die Dauer dieses Vorgangs bestimmt die Taktfrequenz, mit der die Schaltung benutzt werden kann.

Die wenigsten Abbildungen – und zwar eine für jeden Teilterm und den Term selbst – müssen ausgeführt werden, wenn bei jeder Ausführung ihre Argumente bereits richtig interpretiert sind: Vor jedem Teilterm werden also seine Teilterme interpretiert. Während sich eine solche Ausführungsordnung in einer Gatterschaltung nicht erzwingen läßt, ist ihre Erzwingung in Algorithmen üblich und aus Gründen der Effizienz auch meist notwendig. Die Hilfsmittel, die für diese so genannte *Synchronisierung* der Rechenoperationen *nach dem Datenfluß* vorgeschlagen worden sind, sind vielfältig.

Die nach John von Neumann benannten Programme geben die Ordnung durch die Aufschreibung einer Befehlsfolge implizit an und ergänzen sie explizit durch besondere Anweisungen wie Sprung, Verzweigung, DO, FOR usw. Um letztere zu vermeiden, haben andere Programme so weit wie möglich Terme mit einer strengen Syntax benutzt und den Datenfluß aus diesen Termen hergeleitet. Bei der systematischen Syntax besteht eine Möglichkeit darin, Terme von dem am weitesten rechts stehenden Zeichen her abzuarbeiten; denn durch die Reihenfolge, in der die Zeichen eines Terms geschrieben sind, sind die mit ihnen beginnenden Teilterme sogar noch stren-

ger geordnet als durch den Datenfluß. Andere Ansätze, vor allem mit mehreren Prozessoren, bemühten sich, mit allen Teiltermen zu beginnen, deren Argumente bereits interpretiert sind. Das nutzten schon *pipeline*- und RISC-Architekturen für Erhöhung der Rechenleistung aus. Wenn die ausführbaren Operationen mit ihren Argumenten als Aufträge frei im Rechensystem zirkulieren und nach der Ausführung weitere Aufträge erzeugen, war von *Datenflußmaschinen* die Rede. Varianten, bei denen alle interpretierten Teilterme durch ihre Interpretation als Konstante ersetzt und so der Term schrittweise vereinfacht wird, wurden als *Reduktionsmaschinen* diskutiert.

APL ist mit dem gelegentlichen Anspruch, seine Programme würden immer von rechts her abgearbeitet, sichtlich ein Konzept aus der Zeit der Einzelrechner. Da die zweistelligen Funktionszeichen in APL zwischen ihre Argumente geschrieben werden und das linke Argument vor der Anwendung des Funktionszeichens interpretiert werden muß, ist der Anspruch nicht ganz berechtigt!

Generische, argumentlos benutzte und überladene Funktionszeichen

In einer Hinsicht kommt die Syntax von APL der systematischen näher als viele andere Programmiersprachen mit Termen: Eine besonders große Zahl von Funktionszeichen ist Zeichen im engeren Sinn, also nicht Buchstabenfolge, Name und ähnliche Zusammensetzung. Das unterstützt ihre Merkbarkeit und ergibt kurze Programtexte.

Allerdings war der Vorrat verfügbarer Zeichen in der Entstehungszeit von APL noch sehr klein. Deshalb hat Iverson manche Konvention eingeführt, um mit diesem Vorrat möglichst viel ausdrücken zu können.

Generic, das heißt eine Gattung bildend, heißen Funktionszeichen, wenn ihre Interpretation vom Datentyp der Argumente abhängig ist. Davon macht APL Gebrauch, indem viele Zeichen sowohl Zahlen wie Datenstrukturen, also Buchstabenfolgen, *vectors* und aus ihnen zusammengesetzte weitere Strukturen als Argumente haben können. Das ist folgerichtig, weil sogar die gebräuchliche Dezimaldarstellung einer Zahl immer eine Datenstruktur ist: Wenn sie nicht gerade aus einer Ziffer besteht, ist sie ein Satz von Polynomkoeffizienten!

Viele Zeichen werden in APL mit und ohne linkes Argument benutzt. In einigen Fällen ist vereinbart, daß beim Fehlen des linken Arguments das Zeichen so interpretiert wird, als sei das linke Argument vorhanden und hätte den Wert eines besonderen Elements der Algebra. Solche *default*-Werte sind: 0 bei *subtract*, 1 bei *divide* und *take*, *e* bei *power* und *logarithm*, die Einheitsmatrix bei *matrix divide*, (2,1) bei *transpose*.

Dies ist noch die plausibelste Art, zweistellige Zeichen auch als einstellige zu definieren. In der mathematischen Logik wird sie syntaktisch korrekt durch die weiter unten diskutierte Einführung von Variablenzeichen vorbereitet. Ein mit ihnen gebildeter Term wird durch die Abbildung aller möglichen Wertbelegungen der Variablen in den Ergebnisbereich interpretiert; Backus hat später auch syntaktisch nicht korrekte Terme mit fortgelassenen Argumenten so aufgefaßt. APL geht noch einen Schritt weiter, indem die Variable auf einen *default*-Wert festgelegt und dann fortgelassen wird.

Weniger plausibel sind die Paarbildungen aus: Negieren des Imaginärteils und Addition, Projektion in den Einheitskreis und Multiplikation, Betrag und Rest modulo, nächst benachbarter ganzer Zahl und Maximum bzw. Minimum, Fakultät und Binomialkoeffizient, Multiplikation mit o (Pi) und Durchnumerierung von Kreis- und verwandten Funktionen, Negation und Entfernen, Linearisieren und Konkatenieren, Strukturerkennung und Strukturveränderung, und so weiter. In diesen Fällen sind die Zeichen mit Bedeutung überladen. Das wird teilweise zur Gedächtnisbelastung für den Benutzer, erhöht aber jedenfalls den Aufwand für Syntaxprüfung, Synchronisierung und Ausführung; denn nun müssen für jedes Zeichen die möglichen Fälle erkannt und unterschieden werden.

Die Unterscheidung der Zeichen für Kreisfunktionen durch die Werte eines zweiten, linken Arguments kann auch als Bildung von zusammengesetzten Funktionszeichen angesehen werden und leitet deshalb zum nächsten Abschnitt über.

Grund- und Definierte Funktionen

Manche der im vorigen Abschnitt erwähnten Inkonsistenzen wurden durch den früher sehr kleinen Zeichenvorrat der benutzten Rechengeräte provoziert. Inzwischen ist seine Erweiterung um frei gewählte Funktionsnamen in den meisten Programmiersprachen selbstverständlich geworden. Im Vortrag 1960 hat Iverson diese Möglichkeit zur Erweiterung seiner Zeichenmenge F noch nicht behandelt; in seinem Buch von 1962 wird sie nur kurz erwähnt – und zwar unter dem Stichwort Subroutine! Heute gehört sie fest zur Sprache APL.

Die Funktionsnamen werden syntaktisch ebenso benutzt wie die Zeichen für Grundfunktionen. Bei ihrer Vereinbarung wird die Stelligkeit mit deklariert, so daß Überladen unmöglich ist. Zugleich werden lokale Namen der Argumente gewählt, die auch in der nachfolgenden Definition der interpretierenden Abbildung durch ein APL-Programm auftreten. Ergebnis der Abbildung ist wieder ein Element eines Grundbereichs.

Datenstrukturen und erzeugende Abbildungen

Von DO- und FOR-Schleifen ist APL dadurch entlastet, daß 1) Datenstrukturen durch Zusammensetzen aus einfacheren Daten sehr einfach erzeugt werden können und 2) viele Funktionen auch Strukturen als Argumente zulassen, auf die sie dann elementweise angewendet werden.

Merkwürdigerweise sind es die strukturbildenden Notationen in APL, die die meisten Präzedenzregeln notwendig machen – Regeln, die Iverson durch seine nahezu systematische Syntax eigentlich vermeiden wollte. Ähnliche Probleme hatte auch Backus durch die Einführung einer Klammerschreibweise [. . .] für lineare Datenstrukturen (in PASCAL: records). Keinem von beiden – und vielleicht auch nicht Nikolaus Wirth, dem Autor von PASCAL – wurde bewußt, daß die Erzeugung solcher Strukturen in der systematischen Syntax die Einführung von Zeichen für strukturbildende Funktionen erfordert. Deren Argumente sind die einfacheren Daten, ihr Ergebnisbereich die Menge der Datenstrukturen.

Diese Notwendigkeit wurde durch die Bequemlichkeit der Aufzählung ohne oder mit Klammerung verdeckt, die den Gedanken an eine Stelligkeit, also eine zum Funktionszeichen passende Argumentzahl erst gar nicht aufkommen läßt! In PASCAL wird die Argumentzahl immerhin durch die explizite Deklaration der Typen von records sichtbar.

Andererseits war die einzige strukturbildende Operation der Kellerstrukturen, kurz mit push bezeichnet, schon vor 1960 bekannt. Mit ihr allein lange records zu erzeugen, hätte vielfache Wiederholung erfordert. Die Einführung von Funktionszeichen (oder [mit verschiedenen Argumentzahlen wäre ebenso unhandlich: Die Zeichenmenge würde unbeschränkt groß. Aber die Alternative konsequent beidseitiger Klammerung aller Datenstrukturen wäre den unerwünschten Präzedenzregeln grundsätzlich vorzuziehen.

Weitere Funktionszeichen sind für den Zugriff zu Elementen innerhalb von Strukturen unentbehrlich. In einer Kelleralgebra stehen dafür top und pop zur Verfügung, die das oberste Element und den Kellerrest ergeben. In FORTRAN ist die Indizierung nicht als Funktionszeichen erkennbar. Auch APL benutzt sie naiv; außerdem stehen auswählende Operationen wie compress zur Verfügung. Backus hat später einfache Indizes als Funktionszeichen eingeführt und selectors genannt.

Es gibt einen naheliegenden Grund, warum die Realisierung beliebiger Datenstrukturen immer auf die Bildung von records zurückgeführt wird, die bei Bedarf durch strukturbeschreibende Information ergänzt sind: die übliche fortlaufende Adressierung von Speicherbereichen! In Programmsystemen für Matrixalgebra habe ich selbst verschiedene Formen der Strukturbeschreibung vor 1960 verwendet. In APL wird diese Art der Realisierung im Zusammenhang mit shape/reshape erkennbar.

Mehrstellige Funktionen oder Funktionen von Datenstrukturen?

Datensätze, also vectors und records, machen die Entscheidung für bestimmte Stelligkeiten der Funktionszeichen eigentlich überflüssig. Statt einer n-stelligen Abbildung kann stets eine einstellige Abbildung entsprechender Strukturen aus n Elementen benutzt werden. In der Praxis wird bei Funktionsaufrufen immer eine solche Struktur als Argument übergeben!

So ließe sich Ausführung von rechts her stets vollständig erzwingen – wenn sie denn wirklich ein Anliegen ist. In fast allen Fällen würde dadurch die Anzahl von Zeichen für strukturbildende Funktionen im Programmtext erhöht.

Auch die Einschränkung, daß Funktionen genau ein Ergebnis erzeugen und im Datenfluß weitergeben, ist nicht mehr wesentlich, wenn das Funktionsergebnis eine Datenstruktur sein kann.

Allgemeinere Datenfluß-Ordnungen

Die mathematische Logik hat im übrigen ein anderes Ziel, als eine Schreibweise für effiziente Algorithmen zu liefern. Mit der Einführung weiterer nullstelliger Zeichen, den Variablen, wird die Diskussion der Gesamtheit von Interpretationen möglich, die durch Wertbelegung dieser Variablen mit allen möglichen Werten aus den zugeordneten Grundbereichen einer interpretierenden Algebra entstehen können.

Mit Forderungen, daß verschiedene Terme bei allen Belegungen gleich interpretiert werden (wie +ab = +ba, ++abc = +a+bc und andere arithmetische Regeln), können dann bestimmte Algebren (BOOLE'sche, max-plus, Zahlen, Keller, Vektoren usw.) spezifiziert werden. Aus ihren Regeln (Axiomen) ergeben sich dann ihre Grundbereiche, die Funktionsabbildungen und Möglichkeiten ihrer Ausführung.

Diese Aufgabenstellung erwähne ich nur, um die Verschiedenheit algebraischer Variablen und der – leider etwas unpassend so genannten – Programmvariablen zu betonen, die in Algorithmen eine Rolle spielen.

Algorithmen, die durch einen Term nicht zweckmäßig, also effizient ausführbar, beschrieben werden, nimmt man wahr, sobald in dem Term ein Teilterm öfter als einmal auftritt. Ein einfaches Beispiel ist ein Term, der das Quadrat eines Teilterms als Produkt zweier gleicher Teilterme ausdrückt; es ist natürlich wenig effizient, wenn der Teilterm nun zweimal interpretiert werden muß.

Derartige Fälle sind so häufig, daß es jahrhundertalte Praxis ist, mehrfach auftretende Teilterme genau einmal auszuführen, das Ergebnis als Interpretation eines Konstantenzeichens festzuhalten und jedes Auftreten des Teilterms durch dieses Konstantenzeichen zu ersetzen.

Das erklärt die Notwendigkeit von Wertzuweisung in den Programmiersprachen – beziehungsweise bei der Programmausführung. Schematische Darstellungen eines Datenflusses in Algorithmen (wie etwa die Petri-Netze) enthalten deshalb eigene Symbole (im Anklang an den algebraischen Begriff 'Stellen' genannt) als Träger von Argumenten und Ergebnissen. In Rechengeräten entsprechen ihnen Register und Speicherelemente.

Außerdem ändert sich der Datenfluß des Terms so, daß nun Ergebnisse mehr als einem Funktionszeichen als Argument zugeführt werden; im Beispiel der Gatterschaltungen wird in solchen Fällen der Ausgang eines Gatter mit mehreren Gattereingängen verbunden und dadurch Material gespart. Der Datenfluß hatkeine Baumstruktur mehr. Aber er bleibt immer noch eine Ordnung, nach der sich die Ausführung richten kann, um die Anzahl ausgeführter Abbildungen so gering wie möglich zu halten.

Eine Theorie der Algorithmen kann deshalb nicht rein algebraisch sein, eine effiziente Programmiersprache sich nicht allein auf die systematische Termsyntax oder eine ihr eng verwandte stützen!

Zyklen und ihre Synchronisierung

Einen weiteren wesentlichen Beitrag zur Effizienz von Rechengeräten und zur Kürze von Programmen bildet die Möglichkeit, Rechnerkomponenten oder Teile des Programmtextes wiederholt einzusetzen. Das nützt die vorhandenen Ressourcenbesser aus, führt aber dazu, daß der Datenfluß von Algorithmen nicht nur die Baumstruktur verliert, sondern nicht einmal mehr eine Ordnung bleibt. Effiziente Ausführung, bei der jedes Funktionszeichen höchstens einmal interpretiert wird, ist nun kein Ziel mehr. An ihre Stelle tritt die Aufgabe, durch hinreichende Synchronisierung *Determiniertheit* zu erreichen, das heißt Eindeutigkeit der *Wertfolgen*, die als Ergebnisse der Funktionsausführungen entstehen.

Wieder ist das wichtigste Hilfsmittel hierfür aus der Schaltungstechnik bekannt. Wird ein Rechenwerk immer wieder benutzt, dann muß die Zuführung von Argumenten so synchronisiert werden, daß die Ergebnisse sich richtig einstellen und weitergeleitet werden können, ehe die nächste Zuführung erfolgt. Petri hatte eine solche Synchronisierungsvorschrift in die Definition seiner Netze aufgenommen.

Der Datenfluß spielt immer noch die entscheidende Rolle: Es ist notwendig und hinreichend für Determiniertheit, solche Synchronisierung für die Kommunikation aller durch Datenfluß in Beziehung stehenden Funktionszeichen oder -einheiten sicherzustellen.

In von-Neumann-Programmen geschieht das durch die Abarbeitung 'Befehl nach Befehl' und die Speicherungen zwischen ihnen. Auch Sprung- und Verzweigungsbefehle dienen so zur Synchronisierung. Sprachen ohne solche Hilfsmittel sind zwar gelegentlich vorgeschlagen, aber nie in ihrer reinen Form verwirklicht worden: Sie wären nicht effizient.

In diesem Sinne ist auch APL nicht algebraisch, sondern richtet sich wie alle in der Praxis bewährten Programmiersprachen nach den Anforderungen für gute Algorithmen. Also war Iversons Bescheidenheit berechtigt, als er den Namen wählte: A P L – *Eine* Programmiersprache.

Kontakt:

Wolfgang Everling, eMail: cweverling.hamburg@T-Online.de

Reiner Nussbaum

Erprobte Schnittstelle

Sicherer Transport von Zahlungsverkehrsdaten aus dem Beleghaften Zahlungsverkehr mit gift-ALU

Bei der Umstellung des Transfers von Zahlungsverkehrsdaten auf schnelle Intranet Netzwerke muß die Sicherheit des Datentransportes gewährleistet sein. Die Dateien werden mit der Verschlüsselungsmethode von FLAM verschlüsselt. Da FLAM auch auf dem HOST verfügbar ist, können die Daten dort entschlüsselt und weiterverarbeitet werden. Ein Datenaustausch erfolgt in beide Richtungen. Von den Zugangssystemen für den Beleghaften Zahlungsverkehr als Datenlieferanten werden Zahlungsverkehrsdateien angeliefert. Diese Daten werden gebündelt, komprimiert und verschlüsselt und anschließend an geeignete Hostsysteme zur Verarbeitung übertragen. Diese übertragen Quittungsinformationen und Bestandsdaten, die an die Zahlungsverkehrssysteme weiterzuleiten sind. Jeder Verarbeitungsschritt wird protokolliert und zwischengespeichert. Als zeitnahe Arbeitsmethode wird als Automationsverfahren gift-ALU eingesetzt, ein ereignisgesteuertes Automationsverfahren für Unix, Linux und Windows.

Einleitung

Ziel des Projektes ist die Umstellung des Transfers von Zahlungsverkehrsdaten auf schnelle Intranet Netzwerke. Dabei soll die Sicherheit des Datentransportes gewährleistet sein. Da die Host Systeme (noch) keine verschlüsselte Datenübertragung zulassen, soll die Sicherheit bei dem Datentransport durch Dateiverschlüsselung mit der Verschlüsselungsmethode von FLAM erfolgen. Da FLAM auch auf dem HOST verfügbar ist, können die Daten dort entschlüsselt und weiterverarbeitet werden. Ein Datenaustausch erfolgt in beide Richtungen.

Von den Zugangssystemen für den Beleghaften Zahlungsverkehr (Zahlungsverkehrssystemen) als Datenlieferanten werden Zahlungsverkehrsdateien in verschiedenen Verzeichnissen, getrennt nach Geschäftsvorfällen (z.B. Schecks, Lastschriften, EZÜ-Daten und Europaüberweisungen) angeliefert. Diese Daten werden, soweit es sich um kleine Dateien handelt, ‚gebündelt', komprimiert und verschlüsselt und anschließend an geeignete Hostsysteme zur Verarbeitung übertragen.

Von den Host Systemen erfolgt eine Übertragung von Quittungsinformationen und Bestandsdaten (z.B. BLZ Daten und Auftraggeber Bestandsdaten), welche an die Zahlungsverkehrssysteme weiterzuleiten sind.

Jeder Verarbeitungsschritt wird protokolliert. Da die Verarbeitungsdaten in jedem Schritt zwischengespeichert werden, kann bei eventuellen Problemen die Verarbeitung an jeder Stelle der Verarbeitungskette wieder aufgesetzt werden.

Damit die Daten zeitnah verarbeitet werden können, wird als Automationsverfahren gift-ALU, ein ereignisgesteuertes Automationsverfahren für Unix, Linux und Windows, eingesetzt.

Nach dem Ende einer Pilotphase, in welcher die Funktionsfähigkeit nachgewiesen wurde, erfolgte im März 2004 der Einsatz im Produktivbetrieb.

Zahlungsverkehrssysteme -> Hostsysteme

Die Zahlungsverkehrssysteme liefern Informationen für die Host Verarbeitung nach Geschäftsvorfällen (GVO) getrennt an. Die Übergabe erfolgt in GVO bezogenen Verzeichnissen. Um die Betriebssicherheit zu erhöhen, werden alle Daten, welche von außen angeliefert werden, auf einer separaten Festplatte abgelegt. In dem lokalen Netz ist eine sichere Verbindung zwischen den Zahlungsverkehrssystemen und dem gift-ALU Rechner vorhanden.

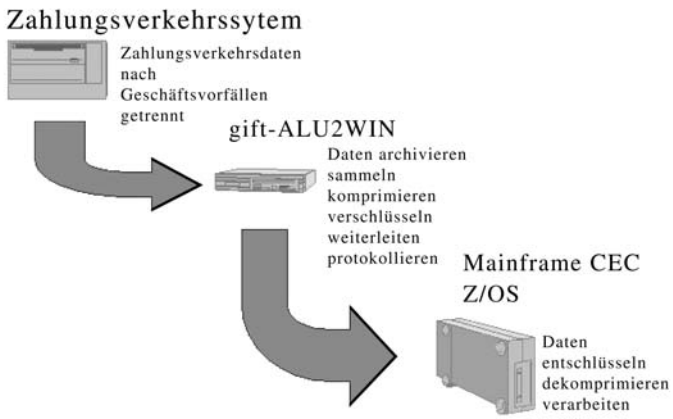


Abb. 1: Übersicht Zahlungsverkehrssysteme -> Hostsysteme

Hostsysteme -> Zahlungsverkehrssysteme

Die Hostsysteme liefern Bestandsdaten und Protokoll-/ Abstimmungsinformationen für die Zahlungsverkehrssysteme. Wie bei den Daten, welche von den Zahlungsverkehrssystemen kommen, erfolgt das Speichern der Host Daten ebenfalls auf einer separaten Festplatte. Nach dem Entschlüsseln und dekomprimieren der Daten, welche vom Host kommen, werden diese für die Zahlungsverkehrssysteme zur Verfügung gestellt. Dies erfolgt in unverschlüsselter Weise, da zwischen dem gift-ALU Rechner und den Zahlungsverkehrssystemen auf lokaler Ebene eine sichere Verbindung realisiert ist.

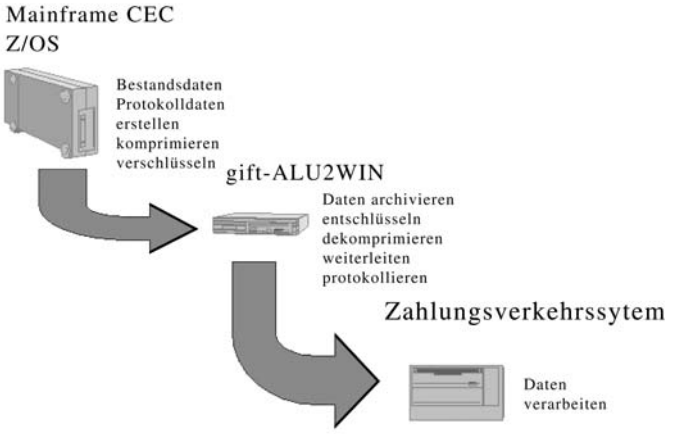


Abb. 2: Übersicht Hostsysteme-> Zahlungsverkehrssysteme

Eingangsverzeichnisse

Die Eingangsverzeichnisse werden grob unterteilt nach den Lieferanten, den Zahlungsverkehrssystemen und den Host Systemen.

Von den Zahlungsverkehrssystemen werden Informationen getrennt nach Geschäftsvorfällen (GVO) angeliefert. Je nach Verarbeitungszweck sind je GVO mehrere Eingangsverzeichnisse vorgesehen.

▶ Beispiele für Geschäftsvorfälle sind:

- Z1 Aufträge
- Europaüberweisungen
- EZÜ / EZL und Scheckzahlungen

Die Erweitung für neue Geschäftsvorfälle ist stets möglich. Von den Hostsystemen werden in getrennten Verzeichnissen Bestands und Protokollinformationen angeliefert.

▶ Beispiele für Bestandsdaten sind:

- BLZ Bestand
- Auftraggeber Gesamtbestand
- Auftraggeber Update
- Scheck Index Bestand

Protokolldateien werden für jeden GVO zu Abstimmungszwecken angeliefert.

Methoden zur Verarbeitung der Daten

Die Dateien, die von den Zahlungsverkehrssystemen angeliefert werden, sind mehrstufig zu verarbeiten, bevor sie zu den entsprechenden Hostsystemen verschickt werden. Die Verarbeitungsschritte sind im einzelnen:

▶ Eingangsverarbeitung:

- Archivierung der Empfangsdaten
- Protokollieren des Datenempfanges
- Je GVO wird festgelegt:
  - ❑ jede einzelneDatei wird sofort weiterverarbeitet.
  - ❑ die vorhandenen Dateien werden in einer Sammeldatei zusammengefasst.
  - ❑ wenn ‚x'-Dateien vorhanden sind, werden diese in einer Sammeldatei zusammengefasst.
  - ❑ zu bestimmten Zeitpunkten werden alle vorhandenen Dateien zusammengefasst.

▶ Kennzeichnen von Dateien:

- Dateien werden mit einer fortlaufenden Nummer als Datei Suffix gekennzeichnet.

▶ FLAM Komprimierung mit/ohne Verschlüsselung:

➢ Je GVO wird festgelegt:
- Archivieren der FLAM Eingabedaten
- Protokollieren der FLAM Verarbeitung
- Nur komprimieren der Datei im Eingangsverzeichnis
- Komprimieren und verschlüsseln der Datei im GVO bezogenen Eingangsverzeichnis. Je Eingangsverzeichnis sind individuelle FLAM Schlüssel möglich.

▶ FTP Versand:

- Archivieren der FTP Sendedaten
- Protokollieren der FTP Übertragung
- Je GVO kann ein Zielsystem konfiguriert werden
- Je Zielsystem sind individuelle User/Passwort Kombinationen möglich

Etwas einfachere Verfahren sind bei den Daten anzuwenden, welche von den Host Systemen angeliefert werden. Gemäß den Vorgaben, sind dieses Dateien ebenfalls zu verschlüsseln. Wenn man von kleineren Verarbeitungsschritten wie das Wandeln von Zeichensätzen absieht, dann ist für die Host Dateien lediglich das GVO bezogene Entschlüsseln und Dekomprimieren der Dateien anzuwenden. Anschließend können die Dateien in den vorgegebenen Verzeichnissen den Zahlungsverkehrssystemen bereitgestellt werden. Ein Archivieren der Eingangsdateien sowie das Protokollieren der Verarbeitung ist realisiert.

Nachstehend ist ein Beispiel der GVO-bezogenen Verzeichnisse, wie dies von den Zahlungsverkehrssystemen vorgegeben ist:

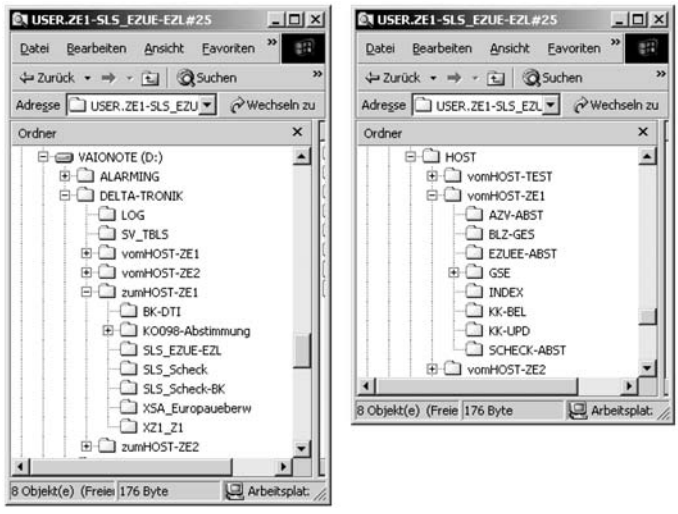


Abb. 3: Beispiel der GVO bezogenen Verzeichnisse

Automatisierter Betrieb der Anwendungen

Für die Realisierung der genannten Anforderungen wurde das Verfahren *gift-ALU* ausgewählt. *gift-ALU* stellt ein ereignisgesteuertes Verfahren für den Betrieb von beliebigen Anwendungen bereit (Patent Az.: DE10260999). Die Umsetzung der Anforderungen wurde vereinfacht, indem zunächst für jede einzelne Anforderung eine eigene automatisierte Anwendung realisiert wurde. Mittels Verzeichnisverweisen, welche die jeweiligen Verarbeitungsergebnisse den folgenden Verarbeitungsschritten bereitgestellt werden, ergeben sich die Jobabläufe in einer intuitiven Art, ohne dass dazu weitere Jobcontrol benötigt wird.

Nachstehend wird beispielhaft gezeigt, wie die Verarbeitungsschritte für den Transport von den Zahlungsverkehrssystemen zum Host in einem normierten Verzeichnisbaum dargestellt werden können:

Die Eingangsverarbeitung greift auf diejenigen Datenbereiche zu, in welche die Zahlungsverkehrssysteme Informationen bereitstellen. Funktional gesehen, wird unterschieden, ob Dateien zusammengefasst werden sollen, oder ob diese direkt (einzeln) zu verarbeiten sind. Damit die Verarbeitungsschritte besser verständlich sind, wurden die realisierten Anwendungen mit einer entsprechenden Nummer als Präfix gekennzeichnet. Somit sind für die Eingangsverarbeitung die Verzeichniszweige 1-SAMMLER, erstellen von Sammeldateien aus dem Eingabebereich und 2-Stamp, verarbeiten einzelner Dateien, welche zur besseren Rückverfolgung mit einer fortlaufenden Zahl als Suffix gekennzeichnet werden, relevant. Unterhalb der Anwendungen, hier Sammler und Stamp, wird die GVO bezogene Verarbeitung implementiert. Bei der Anwendung Sammler kann mit einem numerischen Suffix nach dem GVO Namen angegeben werden, dass eine Mindestanzahl an Dateien vorhanden sein muss, damit eine Verarbeitung stattfindet. Mittels Verzeichnisverweisen auf der GVO Ebene ist es einfach möglich, die Beziehung zwischen dem normierten *gift-ALU* Verzeichnisbaum und den vorgegebenen Datenbereichen der Zahlungsverkehrssysteme herzustellen.

Sind nun auf der Eingangsseite Daten vorhanden, so werden diese von dem Automationssystem erkannt. Anhand der Verzeichnisbaumes wird anschließend die geforderte –erste- Verarbeitung durchgeführt. Wenn beispielsweise für den GVO EZUE_EZL mindestens 25 Dateien vorliegen, wird eine Sammeldatei erstellt. Das Ergebnis der Verarbeitung wird der Anwendung 2-STAMP in einem Unterverzeichnis für den entsprechenden GVO bereitgestellt. Dies erfolgt wie oben bei der Eingangsverarbeitung mit einem Verzeichnisverweis.

Analog wird mit den Aufgaben FLAM komprimieren und FTP Versand verfahren. Die Aufgaben werden einzeln realisiert und die Verarbeitungsergebnisse mit einem Verzeichnisverweis jeweils der nächsten Aufgabe zur Verfügung gestellt.

Obwohl die Automationsverfahren voneinander unabhängig sind, ergeben sich mit dieser Methode mehrstufige Verarbeitungsketten. Dabei werden jedoch keine scripte oder Job Control benötigt. in Abbildung 4 ist der für die Automation verwendete Verzeichnisbaum beispielhaft angegeben.

Die Lösung der anstehenden Aufgaben in kleinen Automationsschritten hat zusätzlich den Vorteil, dass die einzelnen Probleme einfach und überschaubar sind. Ferner ist es später sehr einfach möglich, etwa Verschlüsselungsmethoden oder Datentransportmethoden auszutauschen. In diesen Fällen sind nur die neuen Methoden zu implementieren und anschließend geeignete Verzeichnisverweise zu erstellen. In einer Übergangsphase ist dann auch ein Parallelbetrieb möglich, bei dem etwa ein Teil der GVOs die neuen Methoden verwendet, während andere zunächst unverändert bleiben. Auch ein ‚fall back' ist bei Problemen einfach gegeben, wenn die bisherige Verarbeitung in der Verzeichnisstruktur noch vorhanden ist und mit einem Verzeichnisverweis jederzeit wieder aktiviert werden kann. Die neuen Jobabläufe ergeben sich dann automatisch.

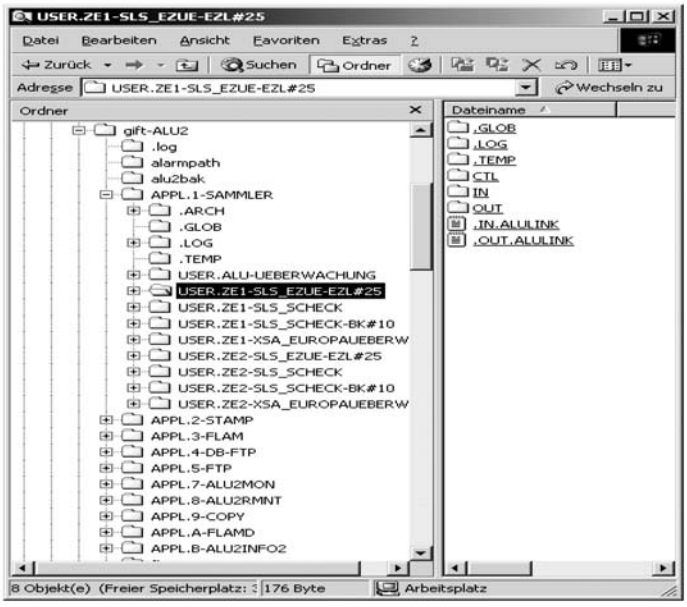


Abb. 4: Für die Automation verwendeter Verzeichnisbaum

Als Beispiel dafür, wie einfach einzelne Aufgaben gelöst werden können, wird die FLAM Komprimierung angeführt. Mit einem sehr kleinen script kann für alle GVOs das GVO bezogene Komprimieren und Verschlüsseln bewerkstelligt werden. Dies kann sogar im parallelbetrieb geschehen.

```
@echo off
@rem FLAM Komprimierung mit gift-ALU
@rem die Komprimierungs- und Verschlüsselungsparameter sind
@rem GVO bezogen und in dem GVO Verzeichnis in der Datei
@rem %alu-user%\.glob\flam.par hinterlegt.
@rem Parallelbetrieb ist möglich.
cd %alu-wrk%

@%alu-sys%\flam\flam parfile=%alu-sys%\appl.%alu-appl%\user.%
alu-user%\.glob\flam.par flamin=%alu-file% flamfile=%alu-ind%.flam
@if not exist %alu-ind%.flam goto err100

@copy "%alu-wrk%\%alu-ind%.flam" %alu-out%\%alu-file%
@if errorlevel 1 goto err102

@echo ok >> %alu-sys%\alu2sig-%alu-appl%-%alu-ind%.ok
@goto ende

:err100
@call %alu-sys%\alu2_sigfailed "Fehler beim FLAM komprimieren"
@goto ende

:err102
@call %alu-sys%\alu2_sigfailed "Fehler102 beim kopieren nach out"
@goto ende

:ende
```

Eventuell erforderliche Archivierungs- und Protokollierungsfunktionen werden von dem Automationssystem durchgeführt. Gleiches gilt für das Bereitstellen von Schnittstellen zu Alarming Systemen im Fehlerfall.

Ausblick / weitere Optionen

Die beschriebene Vorgehensweise hat sich in der Praxis bewährt. Auch im Falle von Fehlersituationen, etwa wenn Daten nicht an den Host weitergeleitet werden können, konnte mit dieser Lösung stets sichergestellt werden, dass der Verzug an Datenlieferung rasch wieder aufgeholt wird.

Als zusätzlicher Nutzen hat sich auch die Offenheit von *gift-ALU* gezeigt. So war es leicht möglich, bei neuen GVOs auch weitere Funktionen wie das Anpassen von Zeichensätzen zwischen den ascii Sätzen von Windows und OS2 herausgestellt. Ferner ist jetzt schon eine Lösung vorhanden, mit der die Daten über einen sicheren Kanal mit den Methoden SCP oder SFTP aus dem Secure Shell Paket (ssh) transportiert werden können. Soweit die Gegenstelle dies unterstützt, kann der (unsichere) FTP Transport durch die sicheren Methoden ersetzt werden. Dies bedeutet, dass in dem Gesamtablauf die Verschlüsselung mit FLAM (sowie das Entschlüsseln auf dem HOST) entfallen kann. Falls schnelle Netzverbindungen vorhanden sind, könnte sogar auf die FLAM Komprimierung (und Dekomprimierung auf dem HOST) entfallen.

Kontakt:

Dr. Nußbaum gift, Buchenerstraße 78, D-68259 Mannheim, eMail: reiner@nussbaum-gift.de

Nachruf

Ken Iverson - ein Pionier der interaktiven Informationsverarbeitung gestorben

Nicht allzu viele IT-Spezialisten wissen heutzutage etwas mit dem Kürzel *APL* anzufangen. Immerhin enthält ein Online-Lexikon der Wochenzeitung „Die Zeit“ folgenden Eintrag:

„*APL* [Abk. für *A P*rogramming *L*anguage; dt.] , eine Programmiersprache, die auf ein 1962 veröffentlichtes Buch des an der Harvard Universität tätigen Kanadiers Kenneth E. Iverson zurückgeht, in dem er die Grundzüge der neuen Sprache vorstellt. Iverson entwickelte in den Folgejahren zusammen mit Adin Falkoff und E.H. Suessenguth die Sprache weiter, bis 1966 der erste Interpreter fertig gestellt werden konnte. APL zeichnet sich durch eine mathematisch orientierte, äußerst knapp gefasste Notation aus (die einzelnen Befehlswörter bestehen nur aus einzelnen Buchstaben). Es ist eine Sprache zur interaktiven Arbeitsweise, bei der die Veränderung von Datenfeldern im Vordergrund steht. Sie wird besonders bei mathematisch-technischen Problemstellungen angewandt. Spezielle Operatoren bewirken, dass Vektoren- und Matrizenrechnungen besonders effizient ablaufen. APL verzichtet auf Funktionshierarchien, wie z.B. »Punktrechnung vor Strichrechnung«, so dass die einzelnen Rechenoperationen immer streng von links nach rechts abgearbeitet werden müssen. Allerdings kann durch das Setzen von Klammern diese strenge Reihenfolge aufgehoben werden. APL existiert heute als Interpreter- und Compiler-Version sowohl für Großrechner als auch für Mikrocomputer.“

Nicht alles ist richtig an dieser Definition. APL unterscheidet zwischen Funktionen, die Daten manipulieren, und Operatoren, die Funktionen modifizieren. APL-Ausdrücke werden von rechts nach links abgearbeitet. Elementare „Befehlswörter“ – Verben - werden durch Sonderzeichen dargestellt. Nicht richtig ist auch, dass Iverson ursprünglich eine Programmiersprache – d.h. eine ausführbare Notation entwicklen wollte. Als Harvard-Dozent lehrte er sein Spezialgebiet – Analysis. Die übliche mathematische Notation erschien ihm wenig konsistent und ungeeignet zur Beschreibung von Algorithmen. So entwickelte er eine eigene Notation, die er in dem erwähnten Buch publizierte. IBM erkannte das Potenzial dieser Ideen und warb Iverson und einige Kollegen (Larry Breed, Roger Moore and Dick Lathwell) von Harvard ab, um diese Notation weiter zu entwickeln. Eine erste größere praktische Anwendung der neuen Sprache war die formale Beschreibung der damals neuen Computergeneration IBM System/360 (A.D. Falkoff, K.E. Iverson, E.H. Sussenguth, *A Formal Description of System/360,* IBM Systems Journal, Vol. 4 No. 4, October 1964).

Es sollte noch bis 1965 dauern, bis eine erste ausführbare Version der Sprache fertig gestellt war, eine im Stapelbetrieb laufende Implementierung für die IBM 7090. Der folgte 1966 ein interaktives System *APL/360,* das Ausgangspunkt für alle späteren APL-Systeme verschiedener Anbieter wurde. Ken Iverson hatte maßgeblichen Anteil an der Fortentwicklung.

Für seine Verdienste ernannte ihn IBM zum „IBM Fellow“, eine Auszeichnung, die nur wenigen IBM-Mitarbeitern zuerkannt wurde.

Auch außerhalb IBM wurde die Arbeit von Iverson anerkannt: Er erhielt 1979 den „Turing Award“, der als Nobelpreis in der Computer-Industrie angesehen wird. Die Begründung lautete: „his pioneering effort in programming languages and mathematical notation resulting in what the computing field now knows as APL, for his contributions to the implementation of interactive systems, to educational uses of APL, and to programming language theory and practice.“

1991 wurde ihm die „ U.S. National Medal of Technology“ verliehen. Außerdem hatte er schon früher den „Harry M. Goode Memorial Award“ gewonnen, in Anerkennung der Konzeption und der Entwicklung von APL. 1981 nominierte ihn die IEEE zum „Computer Pioneer Charter Recipient in the creation and continued viability of the computer industry“

Er verließ IBM 1980, um bei I.P. Sharp in Toronto und später in einer eigenen Firma weiter zu arbeiten.

So genial das Design von APL auch war (und ist), Ken Iverson sah Ungereimtheiten und Verbesserungsmöglichkeiten. Angesichts der im Laufe der Zeit entstandenen Programmbasis verboten sich jedoch unverträgliche Designänderungen. Deshalb entschloss sich Iverson, noch einmal eine neue Sprache zu definieren – J. J kommt ohne die APL-Sonderzeichen aus und löst damit ein Problem, das beim heutigen Stand der Technik gar keines mehr ist.

Bis zum Schluss arbeitete er an der Weiterentwicklung von J, am Computer ereilte ihn ein Schlaganfall. Er starb am 19. Oktober 2004 im Alter von 83 Jahren.

Heute wird APL oft als Nischenprodukt angesehen, dabei ist kaum bekannt, dass große Unternehmen, insbesondere in Deutschland, APL häufig für unternehmenskritische Anwendungen einsetzen. Die APL-Lobby – in Deutschland der Verein *APL Germany e.V.* und die IBM APL Benutzervereinigung GSE – sind Dr. Iverson dankbar für das noch immer herausragende Produktivitätswerkzeug, das er ihnen gab.

Dieter Lattermann
Vorsitzender von *APL Germany e.V.*

Herbert Voß

Flotter Oldie

LaTeX – 25 Jahre und kein Ende

Das Textsatzsystem TeX ist älter als der PC, aber immer noch höchst lebendig. So ist das Erstellen von PDFs in den vergangenen Jahren erheblich leichter geworden, der Anwender hat die Wahl zwischen viel mehr Schriftarten, und sogar interaktive PDF-Formulare lassen sich neuerdings mit pdfTeX erstellen.

Vorgeschichte

Nicht viele Anwendungen können von sich behaupten, seit mehr als einem Vierteljahrhundert zu existieren. Die erste veröffentlichte Version von TeX gab es 1977. Unter der Federführung von Donald Knuth und ausgehend von den Prinzipien des herkömmlichen Drucksatzes entstand ein System für professionellen Textsatz. Knuth nannte es TeX; dieser Name, ursprünglich in griechischen Buchstaben geschrieben, wird mehr oder weniger wie „Tech“ ausgesprochen. Im Jahr 1982 folgte die zweite und 1990 schließlich die dritte und letzte Version.

Eine sicherlich weise Entscheidung traf Knuth, als er anschließend beschloss, TeX nicht weiterzuentwickeln. Von Anfang an hatte er den gesamten, in der Sprache WEB erstellten Quellcode von TeX offen zur Verfügung gestellt, was dazu führte, dass TeX schon sehr früh für jedes System angepasst wurde.

In der Ursprungsversion kann TeX aus heutiger Sicht nur als kryptisch bezeichnet werden, was Leslie Lamport bereits Anfang 1980 veranlasste, unter dem Namen LaTeX eine Sammlung von Makropaketen zu entwickeln, welche die Benutzung von TeX erheblich vereinfachen[1]. LaTeX2e stellt den heute aktuellen Stand dar. Seit einigen Jahren wird unter dem Projektnamen LaTeX3 mehr oder weniger intensiv an einer Fortentwicklung gearbeitet; Lamport stellte seine Arbeit an LaTeX inzwischen allerdings ein.

Was es ist …

Grundsätzlich ist TeX nichts anderes als eine textsatzspezifische Programmiersprache, die einen Datenstrom Token für Token ab abarbeitet. Dabei besteht dieser Datenstrom aus reinem Klartext, der ASCII-Zeichen oder bei Verwendung eines Zusatzpakets auch erweitertes ASCII beziehungsweise Unicode enthält. Dieser wird dann vom TeXCompiler beziehungsweise seit neuestem vom pdfTeX-Compiler bearbeitet und als DVI- oder PDF-Datei ausgegeben. Das klingt nicht sonderlich kompliziert, ist aber für viele Anwender gewöhnungsbedürftig, denn „normale“ Programmiersprachen folgen etwas anderen Prinzipien.

Um ein 1000 Seiten umfassendes Buch-Manuskript mit hunderten von Abbildungen, Tabellen, Gleichungen, Fußnoten und Literaturzitaten auf einem normalen PC in höchster Qualität zu setzen, benötigt TeX etwa fünf Minuten. Da der gesamte Quelltext keinerlei Zeichen im binären Format enthält, ist die Gefahr, dass das Dokument dabei durch Fehler in der Software oder bei der Bedienung zerstört wird, gleich Null. Die Komplexität des Dokuments ist proportional zur Kompilierzeit – den eigentlichen Satzvorgang beeinflussen weder extrem verschachtelte Formeln noch besonders aufwendige Tabellen sonderlich. Je spezieller die Anforderungen an ein Dokument, desto stärker machen sich also die Vorteile von TeX bemerkbar.

Dazu gehören auch automatisch generierte Literaturverzeichnisse, wie sie in der gesamten Spannbreite von technischen bis zu juristischen Fachtexten Pflicht sind.

Dass ein TeX-Dokument grundsätzlich im Klartext erstellt wird und daher beliebig zwischen verschiedenen Betriebssystemen oder Editoren austauschbar ist, erleichtert insbesondere das Arbeiten in großen Projekten mit mehreren Autoren, die auf unterschiedlichen Plattformen arbeiten.

Ein weiteres Argument für die Anwendung von TeX sind die verfügbaren Schriftarten. Bei der Entwicklung der TeX-Fonts hat man sich mehr Gedanken gemacht als beispielsweise bei der Erstellung der unzähligen TrueType-Schriften. Dieses sorgfältige Design garantiert insbesondere ein perfektes Zusammenspiel von Textfonts und mathematischen Fonts. So lassen sich die Schriften für Fließtext und mathematische Formel gut aufeinander abstimmen.

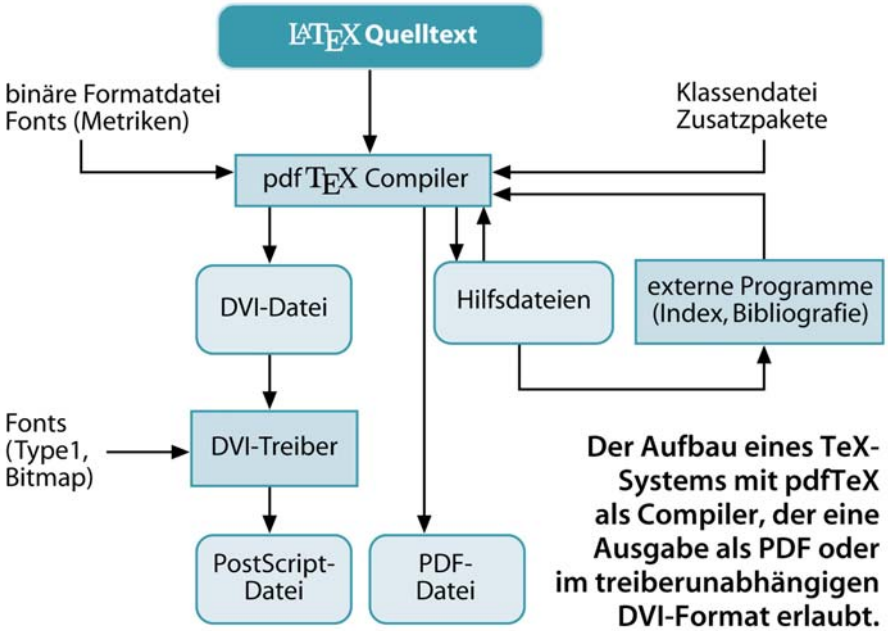
TeX-Einsteiger greifen dazu am besten auf die reichlich mitgelieferten Basisfonts wie Computer Modern oder Latin Modern zurück. Wer den damit verbundenen Arbeitsaufwand nicht scheut, kann sich auch einen neuen Font aus bestehenden zusammenbasteln. Dazu muss man keine neuen Zeichen definieren, sondern kann Anleihen aus mehreren Fonts machen und diese zu einem neuen, virtuellen Font zusammenfassen, der nun ganz normal benutzbar ¡ist. Das bedeutet allerdings einen gewissen Aufwand, der zudem nicht immer sofort von Erfolg gekrönt sein wird.

Nachholbedarf besteht noch bei der Eingabekodierung, etwa für die Arbeit an mehrsprachigen Texten.

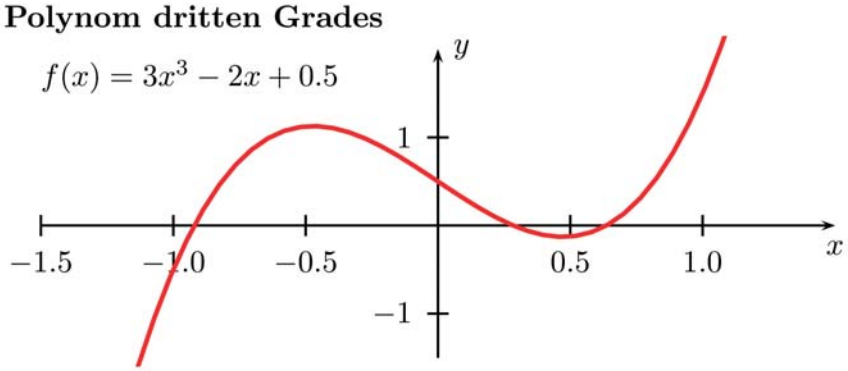
… und was nicht

TeX selbst wird nur für den Übersetzungsvorgang benötigt. Es hat keinerlei Eigenschaften eines Texteditors, womit schon beschrieben wird, was es nicht kann: Es hilft weder bei der reinen Texterfassung noch interessiert es sich dafür, wie und womit der Quelltext erfasst wurde. Dabei ist der Funktionsumfang der Programmiersprache TeX ganz klar auf die Bedürfnisse des Textsatzes ausgerichtet.

Andere Probleme als Textbe- und -verarbeitung lassen sich mit TeX zwar grundsätzlich auch lösen, jedoch benötigt man dazu schon einen latenten Hang zur Selbstkasteiung.



Der Aufbau eines TeX-Systems mit pdfTeX als Compiler, der eine Ausgabe als PDF oder im treiberunabhängigen DVI-Format erlaubt.




```
\documentclass{article}
\usepackage{pst-func}
\begin{document}
\textbf{Polynom dritten Grades}
\psset{xunit=3cm,yunit=1cm}
\begin{pspicture*}(-1.6,-1.6)(1.6,2.15)
  \psaxes[Dx=.5]{->}(0,0)(-1.5,-1.5)(1.5,2)
  \psPolynomial[linewidth=1.5pt,linecolor=red,
     coeff=0.5 -2 0 3]{-1.5}{1.5}
  \uput[-90](1.5,0){$x$}
  \uput[0](0,2){$y$}
  \rput[lb](-1.5,1.5){$f(x)=3x^3-2x+0.5$}
\end{pspicture*}
\end{document}
```

Mit zahllosen Zusatzpaketen kann man TeX je nach eigenen Bedürfnissen um spezielle Funktionen erweitern, etwa zum Zeichnen einer Funktion samt Koordinatensystem.

Für fast alle TeX-Varianten gilt, dass der Compiler aus dem Quelltext unter Einbindung der Fonts und der zusätzlichen Stildateien (Pakete) eine Ausgabe im treiberunabhängigen DVI-Format (Device Independent) erstellt. Lediglich bei Anwendung von pdfTeX entfällt diese letzte Stufe in der Regel, da hier standardmäßig PDF-Dateien erzeugt werden. Die (La)TeX-Philosophie ist damit völlig entgegengesetzt zu der von OpenOffice oder Microsoft Word, denn das aktuelle Layout ist beim Tippen uninteressant – der Schreiber soll sich auf seinen Inhalt konzentrieren, (La)TeX macht danach den Rest. Dass diese ungewohnte Arbeitsweise nicht jedermanns Sache ist, stand der Verbreitung von TeX bisher entgegen.

Mittlerweile gibt es verschiedene grafische Editoren, die einem das Leben erheblich erleichtern, indem sie einzelne Teile des Dokuments im WYSIWYG darstellen. Dazu gehört das Multitalent Emacs mit AUCTeX ebenso wie andere Editoren, wovon nur TeXmaker durch Verwendung der freien Qt-Bibliothek sowohl für Windows als auch Unix zur Verfügung steht.

Dass TeX als ein System, welches einen nicht-veränderbaren Kern aufweist, nach wie vor aktuelle Herausforderungen an den Textsatz erfüllen kann, liegt an der Schnittstelle, die es erlaubt, TeX beliebig zu erweitern. Dies geschieht mit Paketen, die der Anwender nach seinen Bedürfnissen in ein Dokument einbindet.

Die Zahl aller verfügbaren Pakete, welche fast alle durch das Comprehensive TeX Archive Network (CTAN) [2] zur Verfügung gestellt werden, geht in die tausende – die Organisation ist mittlerweile ein echtes Problem.

Begrifflich geht es schon mal ziemlich durcheinander – selbst nach so vielen Jahren wissen nur wenige Anwender, welches Paket was wirklich kann. So ist LaTeX nichts anderes als ein Frontend für TeX, um dessen Anwendung handhabbar zu machen, genauer gesagt eine Makrosammlung für TeX. Daneben gibt es die so genannten Distributionen, deren Ziel es ist, Anwendern ein komplettes Gesamtsystem einschließlich vorhandener Erweiterungen auf ihrem Rechner zu installieren.

TeX-Distributionen gibt es für beliebige Betriebssysteme über das CTAN oder als Bestandteil vieler Linux-Distributionen. Das TeXLive-Projekt entwickelte eine DVD für alle gängigen Betriebssysteme [3]. Alle Distributionen enthalten die wichtigsten Programme, die für eine Ausgabe ¡im Zwischenformat DVI, als PostScript-Datei oder

PDF notwendig sind, außerdem unterschiedliche Zusatzpakete mit erweiterten Möglichkeiten wie Standardlayouts oder Funktionen für Sonderfälle wie den Satz von Musiknoten. Unter Windows stellt MiKTeX den Standard dar. Es ist insbesondere für den Anfänger hilfreich, da es Zusatzpakete halbautomatisch lädt. Die Tabelle auf Seite 175 nennt die bekanntesten TeX-Distributionen. Mittlerweile wird standardmäßig pdfTeX beziehungsweise die erweiterte Version pdfeTeX als Compiler eingesetzt, welche wahlweise DVI- oder PDF-Dateien erzeugen. Dadurch stehen grundsätzlich Dinge wie Hypertext- Dokumente oder Microtyping zur Verfügung. Vor allem Letztgenanntes ergibt noch einmal einen Qualitätssprung, da hier die Formatierung des Textes nicht nur an den Zeichen , sondern auch noch an dem Schwärzungsgrad des Randes ausgerichtet wird, was in der Fachsprache als „character protrusion and font expansion" bezeichnet wird.

Immer häufiger werden PDFDokumente online erstellt, wozu pdfTeX geradezu prädestiniert erscheint, denn mit Hilfe einer beliebigen Skriptsprachen wie PHP, Python oder Perl ist es kein Problem, ein TeX-Dokument zu generieren, dieses dann durch pdfTeX laufen zu lassen und die so erzeugte PDF-Datei als Ergebnis über einen HTTP-Server zum Browser des Anwenders zu schicken. Diese Art der Dokumentenerstellung nimmt ständig zu, da man dabei sämtliche Formatierungsprobleme einfach an den pdfTeX-Compiler übergibt.

So werden beispielweise alle Ausgaben des über ein Intranet unter Apache gesteuerten Gruppenterminkalenders der SPD-Bundestagsfraktion mit mehreren hundert Teilnehmern auf diese Art erzeugt; erst dadurch ließ sich die Ausgabe kompliziert formatierter Dokumente vollständig in den Griff bekommen.

99

Die über die ganze Welt, je nach den auftreffenden Anre|gungen, sich aus- A 198

lers ausgewählte Abhh."[115] Jena 1921[d]. Below rühmt ihn etwas zurückhaltender, deutet aber [a]an, daß Marx wichtigste Erkenntnisse ihm verdanke[a].[116] Neupublikationen sind unterwegs.[117] [b]Scheler erinnert vielfach an ihn[118] [b]. Dagegen hat ihn leidenschaftlich und eindrucksvoll "Schmitt-Doroti_ a. a. O." bekämpft.[119] In Wahrheit sind seine Schriften [d]sehr verschiedenartig (Lehre vom Gegensatz 1804, Vorl. über deutsche Lit. 1807, Elemente der Staatskunst 1809, Vermischte Schriften[2] 1817). Die Elemente[d] ins-

a–a *A*: eine Abhängigkeit Marxens von ihm und ähnlichen Schriftstellern an.
b–b *A*: Schelers Mischung von Rittertum, Literatur und Katholizismus hat bei ihm ihre volle Analogie.
c–c *A*: Schmitt-Doroti_, Die politische Romantik,
d–d *A*: (Lehre vom Gegensatz 1804; Vorlesungen über deutsche Literatur 1807; Elemente der Staatskunst 1809; Vermischte Schriften[2] 1817) sehr verschiedenwertig; die „Elemente"

115 Vgl. Othmar Spann: Vorwort zu: Die Herdflamme. Sammlung der gesellschaftswissenschaftlichen Grundwerke aller Völker und Zeiten. 1. Band. Adam H. Müller. Die Elemente der Staatskunst (1922).
116 Vgl. Georg von Below: Die deutsche Geschichtschreibung von den Befreiungskriegen bis zu unseren Tagen (1916), S. 172–175.
117 Vgl. u. a.: Johannes Hofer: Adam Müller und Metternich (1922); Benno Immendörffer: Adam Müller und die Staatslehre (1922); Friedrich Lenz: Über Adam Müllers Staats- und Gesellschaftslehre (1922); Gustav Seidler-Schmid: Adam Müller. Von der Bedeutung seiner Lehren für unsere Zeit (1922); Otto Weinberger: Adam Müller (1922).
118 Vgl. z. B. Max Scheler: Vom Ewigen im Menschen(1920), S. 161.
119 Vgl. Carl Schmitt-Doroti_: Politische Romantik (1919), z. B. S. 112 f.: „Das Normale im juristischen wie im moralischen Sinne ist ihm inkommensurabel. Einem normativen Begriff gegenüber versagt die Gegensätzlichkeit: die normale Sexualität ist nicht der Indifferenzpunkt von Sadismus und Masochismus, der Mut eines tapfern Mannes nicht die höhere Einheit von Depression und Exaltation, der vernünftig geordnete Staat nicht eine Synthese aus Anarchie und Despotie. Adam Müllers amoralisches Verständnis für alles und sein Gegenteil, seine Sucht, überall zu vermitteln, seine ‚weltumfassende Toleranz', vor der Gentz erschrack, weil ‚dann nichts mehr übrig blieb, was man lieben und rechtschaffen hassen' konnte, seine unmännliche Passivität, zu der er Burkes, de Maistres und Bonalds Abneigung gegen das künstliche ‚Machen' umzubiegen verstand, sein gefühlsmäßig im Grunde immer mit allem einverstandener, alles gutheißender Pantheismus, sind wohl auch individualpsychologisch aus seiner weiblichen, pflanzenhaften Natur zu erklären, für den romantischen Ästhetizismus waren sie aber die geeignete psychische und physische Disposition, weil sie das Subjekt ganz an seinen Affekt und die mit der Verarbeitung des Affekts sich begnügende ästhetische Produktivität wiesen."

Selbst komplizierte Anordnungen von Fußnoten, wie sie bei textkritischen Editionen die Regel sind, können mit dem (La)TeX-Paket bigfoot gesetzt werden.

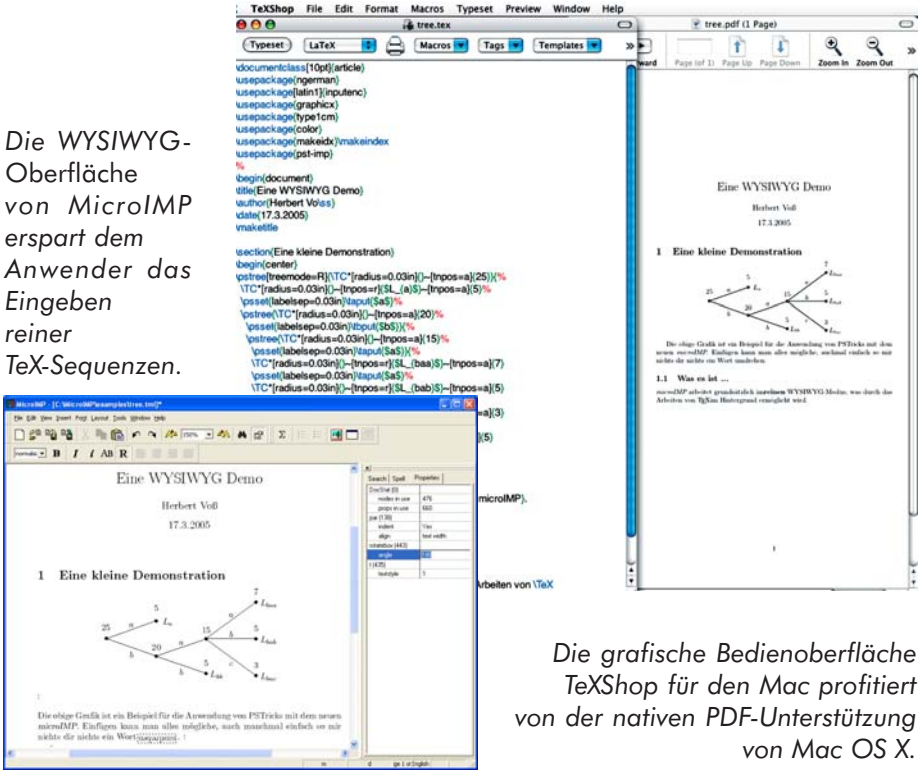
Bild: David Kastrup

Grafische Bedienoberflächen

Eine formvollendete Ausgabe ist eine Sache – die Eingabe des Quellcodes eine völlig andere. Es gibt unterschiedliche Oberflächen, deren gemeinsames Ziel eine (La)TeX-Umgebung ist, die Neulinge auch ohne tiefer gehende TeX- oder LaTeX-Kenntnisse bedienen können. Bis heute gibt es jedoch noch keine voll funktionsfähige WYSIWYGLösung. Das letzte aktive Projekt in dieser Richtung ist MicroIMP für Windows. Es geht wie auch bei früheren Ansätzen darum, möglichst viele Befehle von TeX und LaTeX per Menü verfügbar zu machen – ein Kampf gegen Windmühlen, da TeX ein offenes System ist, welches jeder nach Belieben erweitern kann. Der Entwicklung einer WYSIWYGUmgebung, die stets vorher wissen muss, wie das Ergebnis einer neuen Funktion darzustellen ist, steht das entgegen. Für Anwender, die hauptsächlich Texte ohne exotische Formatierungswünsche erstellen, ist microIMP eine brauchbare Alternative; die Software steht allerdings nur in der Beta-Phase kostenlos zur Verfügung. Anders als beim Vorläufer VTeX wird es keine gesonderte freie Linux-Version geben; die Entwickler setzen hier auf wine, wobei erste Tests allerdings nicht sehr viel versprechend verliefen.

Einen Kompromiss zwischen einem WYSIWYG-Editor und einem reinen Texteditor stellt LyX dar. Es versucht einerseits möglichst viele originalgetreue Darstellungen zu erreichen und dem Benutzer andererseits die Freiheit zu lassen, beliebige Erweiterungen vorzunehmen. Diese werden dann einfach als Codefragmente übernommen und entsprechend im Text markiert. Man spricht in diesem Zusammenhang auch von WYSIWYM (What You See Is What You Mean), was aber letztlich nur das Schönreden eines fehlenden Features ist.

Die WYSIWYG-Oberfläche von MicroIMP erspart dem Anwender das Eingeben reiner TeX-Sequenzen.



Die grafische Bedienoberfläche TeXShop für den Mac profitiert von der nativen PDF-Unterstützung von Mac OS X.

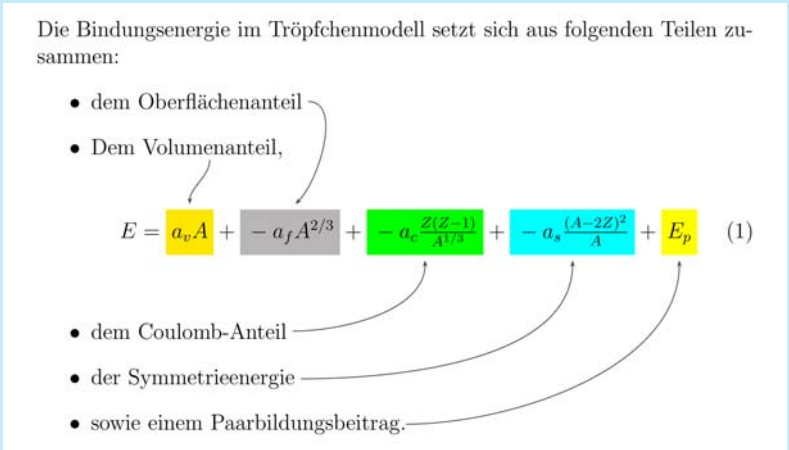
Ähnlich wie Emacs lässt LyX während der Arbeit am Text im Hintergrund bestimmte Teile als eigenständige TeX-Dateien laufen, wandelt die Ausgabe in das PNG-Grafikformat und fügt dieses anstelle des reinen Textes ein. Das funktioniert hervorragend mit allen mathematischen Gleichungen und extern eingebundenen Teildateien beliebigen Inhalts. Insbesondere Neueinsteigern kann LyX die Arbeit erleichtern, denn es bietet Unterstützung für Querverweise und Literaturzitate, wobei ein Rückgriff auf externe Datenbanken möglich ist. Wie fast alle anderen Editoren benötigt auch LyX einen externen Viewer, der die erstellte Ausgabe darstellt. Lediglich Kile aus der KDE-Reihe erlaubt das Benutzen eines internen Viewers, was aber nicht unbedingt empfehlenswert ist, denn man möchte in der Regel Quelltext und Ausgabe nebeneinander betrachten. Die einzige grafische Oberfläche, die sich sowohl für Windows als auch für alle Unix-Varianten eignet, ist der Kile-Ableger TeXmaker. TeXmaker hilft gerade absoluten Anfängern, sich zurechtzufinden, denn das ansprechende Layout der grafischen Oberfläche beschränkt sich auf das Wesentliche. Anders TeXnicCenter oder das kommerzielle WinEdt, deren unzählige Buttons eher etwas für den erfahrenen Anwender sind. Sowohl TeXnicCenter als auch WinEdt laufen nur unter Windows. Sie unterstützen den gesamten Funktionsumfang von reiner Textauszeichnung über mathematische Formeln hin zur Handhabung größerer Projekte mit Teildokumenten, Indexerstellung und Bibliographie.

Auch für die Mac-Gemeinde steht eine grafische Oberfläche zur Verfügung: TeXshop kommt im üblichen Mac-Outfit daher und hat einen ähnlichen Funktionsumfang wie die zuvor beschriebenen Oberflächen. Das bereits angesprochene Kile ist Teil des Fink-Projekts und kann deshalb ebenfalls auf einem Mac unter Mac OS X genutzt werden. Alle genannten Frontends kompilieren und konvertieren Dokumente mit nur einem Mausklick und erleichtern die Arbeit mit zahlreichen Vorlagen und Assistenten. Trotz der Vereinfachung für den Anwender setzen sie gewisse Grundkenntnisse in (La)TeX voraus – wie viel, hängt sehr stark von der Komplexität des Quelldokuments ab.

Durch die permanente Seitenansicht, die sich bei Neuübersetzung des Dokuments automatisch aktualisiert, ist der Verlust der echten WYSIWYG-Darstellung nicht mehr das eigentliche Problem

.

```
\documentclass[12pt,a4paper,ngerman]{article}
\usepackage{pstricks,pst-node,babel}
\usepackage[latin1]{inputenc}
\setlength\parindent{0pt}
\pagestyle{empty}
\begin{document}
\definecolor{lila}{rgb}{0.6,0.2,0.5}
\definecolor{darkyellow}{rgb}{1,0.9,0}
Die Bindungsenergie im Tröpfchenmodell setzt sich aus
folgenden Teilen zusammen:
\begin{itemize}
 \item dem \rnode{b}{Oberflächenanteil}
 \item Dem \rnode{a}{Volumenanteil},\\[0.5cm]
  \def\xstrut{\vphantom{\frac{(A)^1}{(B)^1}}}
  \begin{equation}
    E =
    \rnode[t]{ae}{\psframebox*[fillcolor=darkyellow,
      linestyle=none]{\xstrut a_vA}} +
    \rnode[t]{be}{\psframebox*[fillcolor=lightgray,
      linestyle=none]{\xstrut -a_fA^{2/3}}} +
    \rnode[t]{ce}{\psframebox*[fillcolor=green,
      linestyle=none]{\xstrut -a_c\frac{Z(Z-1)}{A^{1/3}}}} +
    \rnode[t]{de}{\psframebox*[fillcolor=cyan,
      linestyle=none]{\xstrut -a_s\frac{(A-2Z)^2}{A}}} +
    \rnode[t]{ee}{\psframebox*[fillcolor=yellow,
      linestyle=none]{\xstrut E_p}}
  \end{equation}\\[-0.2cm]
 \item dem \rnode{c}{Coulomb-Anteil}
 \item der \rnode{d}{Symmetrieenergie}
 \item sowie einem \rnode{e}{Paarbildungsbeitrag}.
\end{itemize}
\psset{linecolor=darkgray,linewidth=0.5pt,nodesep=3pt}
\nccurve[angleA=-90,angleB=90]{->}{a}{ae}
\nccurve[angleB=45]{->}{b}{be}
\nccurve[angleB=-90]{->}{c}{ce}
\nccurve[angleB=-90]{->}{d}{de}
\nccurve[angleB=-90]{->}{e}{ee}
\end{document}
```



Mit dem Paket PSTricks kann der Anwender Text und grafische Elemente beliebig kombinieren.

Die Zukunft

TeX, de facto in seiner Kernentwicklung nur noch auf das Beseitigen von Fehlern beschränkt, hat bislang keinen würdigen Nachfolger gefunden. Hunderte von Softwareentwicklern haben im letzten Jahrzehnt versucht, ein entsprechendes Programm zu entwickeln, welches mindestens alle Nachteile von TeX beseitigt, wozu beispielsweise eine fehlende Gleitkommazahlarithmetik gehört. Irgendwann haben alle ihre Arbeit beendet, sei es mit einem Endprodukt, welches schlechter war als das Original, oder weil sie schlicht nicht fertig wurden.

Weiterentwicklung findet derzeit auf Nebenschauplätzen statt, etwa bei den Paketen, auch Stildateien genannt, welche vom Anwender nach Belieben hinzugeladen werden können. Das inhaltliche Spektrum dieser Pakete auch nur ansatzweise zu beschreiben, würde ein ganzes Buch füllen. Exemplarisch seien hier zwei Beispiele angegeben, zum einen das Paket PSTricks und zum anderen die Möglichkeit, mit hyperref respektive pdfTeX Text und Hypertext-Elemente zu kombinieren. Beide Beispiele zeigen, dass man mit (La)TeX oder pdfTeX mittlerweile die unterschiedlichsten Elemente als Einheit in einen Text einbinden kann. Auf diese Weise können sowohl Text als auch Grafik denselben Font benutzen, und Grafiken lassen sich direkt im Editor bearbeiten. PSTricks stellt PostScript-Funktionen zur Verfügung und bietet mit diversen Zusatzpaketen grafische Elemente von einfachen Linien und Kurven bis hin zu dreidimensionalen Abbildungen [4]. Sicher wird niemand auf die Idee kommen, das Layout der nächsten Prozessorgeneration in Textform zu beschreiben – dennoch gibt es Situationen, in denen eine Beschreibung der Grafik in Textform sinnvoll ist.

Beispielsweise bei einem simplen Ablaufschema kann es viel Zeit sparen, wenn man bei der Texterstellung unmittelbar Änderungen an einer Grafik vornehmen kann, ohne dazu erst eine Grafiksoftware zu starten. Neben den Paketen gibt es weiterhin die verschiedensten eigenständigen Programme, wie makeindex zum Erstellen von Stichwortlisten sowie bibtex für korrekt gesetzte Zitate und Literaturlisten. Allein die Zahl der Bibliographiestile ist beachtlich, wobei eine Anpassung an eigene Notwendigkeiten immer möglich ist.

Dass (La)TeX alles andere als ein alter Hut ist, beweist auch die Zahl der Neuauflagen oder Neuerscheinungen auf dem Buchmarkt eindrucksvoll [5, 6]. Es besteht die begründete Hoffnung, dass es sogar das goldene Jubiläum erreichen wird. Zu tun gibt es noch genug – so kommt etwa das automatische Paketmanagement noch nicht mal in Ansätzen an die Möglichkeiten heran, die man etwa vom CPAN (Comprehensive Perl Archive Network) gewohnt ist. Spannende Ansätze gibt es im exTeX-Projekt, das versucht, mit den Mitteln von Java all das zu realisieren, was für TeX nur mit einigen Klimmzügen möglich ist oder auch bei ant, welches ebenfalls versucht, ein TeX-Ersatz zu werden. (dwi)

```
\documentclass[12pt]{article}
\usepackage[latin1]{inputenc}
\usepackage{palatino}
\usepackage[bookmarks=false]{hyperref}
\setlength\parindent{0pt}
\setlength\parskip{10pt}
\begin{document}
\begin{Form}[action=mailto:voss@perce.de,encoding=html,method=post]
\TextField[width=8cm,name=name,value={},bordercolor=0 0 1]
    {\makebox[2cm][l]{\textbf{Name:}}}
\TextField[width=8cm,name=email,value={@},borderstyle=U,
    backgroundcolor=0.8 0.8 0.8,bordercolor=0 0 0]
    {\makebox[2cm][l]{\textbf{eMail:}}}
Bestätigung: \CheckBox[name=ja,checked]{Ja}
\Submit{Senden} \Reset{Löschen}
\end{Form}
\end{document}
```

Mit pdfTeX lassen sich Textfelder oder Checkboxen in eine Hypertext-Anwendung einbinden.

TeX-Distributionen

emTeX	OS/2 und DOS (veraltet)	www.dante.de/CTAN/systems/msdos/emtex/
teTeX	Linux/Mac (Standard)	www.tug.org/teTeX/
MikTeX	Windows (Standard)	www.miktex.org/
fpTeX	Windows	www.fptex.org/
VTeX	Linux/Windows/OS2	www.micropress-inc.com/
TeXLive	alle Plattformen	www.tug.org/texlive/
proTeXt	Windows	www.tug.org/protext/
OzTeX	Mac OS X	www.trevorrow.com/oztex/

Literatur

[1] Leslie Lamport, Das LaTeX Handbuch, Bonn (1995), ISBN 3-89319- 826-1

[2] Webseite des Comprehensive TeX Archive Network (CTAN), www. ctan.org

[3] Bezugsquellen für die DVD des TeXLive-Projekts: Lehmanns Fachbuchhandlung, www.lob.de oder Deutschsprachige Anwendervereinigung TeX e.V. (DANTE), www. dante.de

[4] Herbert Voß, PSTricks – Grafik für TeX und LaTeX; Berlin (2005), ISBN 3865410731

[5] Frank Mittelbach u.ja., The LaTeX Companion, Boston (2004), ISBN 0-201-36299-6

[6] Elke und Michael Niedermair, LaTeX – das Praxisbuch, Poing (2005), ISBN 3-7723-6434-9

Kontakt:

Herbert Voß, Berlin
eMail: Herbert.Voss@alumni.TU-Berlin.DE

Erstveröffentlichung in c't 2005, 9. Der Nachdruck erfolgte mit freundlicher Genehmigung der Zeitschrift c't (Heise-Verlag, Hannover)

Anzeige

APL Germany e.V.

http://www.apl-germany.de

- Homepage
- Wir über uns
- Was ist APL?
- Wer bietet APL an?
- Veranstaltungen
- Literaturhinweise
- APL-Journal
- APL-Nachrichten
- APL-Links

NEW! Herbst-Tagung

Webmaster: webmaster@hoesle-kienzlen.de

Der *APL Germany e.V.* ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache APL, sowie darüberhinausgehend die Verbreitung des Verständnisses der Mensch - Maschine - Kommunikation zu fördern.

Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet *APL Germany e.V.* den adäquaten organisatorischen Rahmen.

Zu diesem Zweck veranstaltet der *APL Germany e.V.* beispielsweise Vorträge, Diskussionen und APL-Kurse, stellt Kontakte zu Computerfirmen her, die APL anbieten, und gibt das Mitteilungsblatt „APL-Journal“ heraus.

Um die Kommunikation der Mitglieder untereinander zu erleichtern, hat APL Germany einen Nachrichten-Server eingerichtet. Bei diesem Server kann sich jeder, der an APL interessiert ist einschreiben. Eine Mitgliedschaft bei *APL Germany e.V.* ist nicht erforderlich (aber natürlich gerne gesehen).

Kontaktadresse:
APL Germany e.V.
Rheinstr. 23

D-69190 Walldorf

info@apl-germany.de

IBM Workstation APL2 for Multiplatforms Version 2.0 wurde entwickelt vom APL Products and Services Team im IBM Silicon Valley Labor in San José, Kalifornien, und wird laufend erweitert. Das Produkt umfasst universell einsetzbare und untereinander voll kompatible APL2-Umgebungen für die Betriebssysteme

- AIX
- Linux
- Sun Solaris und
- Windows (98, ME, NT, 2000, XP).

Selbstverständlich ist Workstation APL2 Version 2.0 kompatibel mit APL2 Version 2.2 für Großrechner IBM zSeries unter TSO und CMS sowie mit den älteren APL2 Version 1 Produkten für AIX, OS/2, Sun Solaris und Windows. Seit eh und je erlauben alle APL2-Systeme transparentes Im- und Exportieren von Programmen und Daten und sichern damit einen hohen Grad von Anwendungsportabilität. Version 2 baut diese Fähigkeiten noch weiter aus, vor allem durch die volle Portabilität der neuen Namespaces.

Workstation APL2 Version 2.0 ersetzt natürlich die alten APL2 Version 1 Produkte für AIX, Solaris und Windows und kombiniert alle deren Vorteile und Einrichtungen. Darüber hinaus bringt Version 2.0 bis zum Service Level 6 wichtige Neuerungen wie:

- Unterstützung von Linux auf PC-kompatiblen Systemen
- Namespaces zur Einkapselung und Weitergabe von Anwendungen
- Schnittstellen zu vielen anderen Programmiersprachen, besonders zu Java
- Schnittstellen zu Microsoft COM und Visual Basic (nur für die Windows-Plattform)
- Schnittstelle für die TCL/TK-Befehlssprache
- Online-Dokumentation im PDF-Format
- APL2 Runtime Library zur (kostenlosen) Verteilung von APL2-Anwendungen

Durch die vielen Schnittstellen und Prozessoren sowie die 4 Systemplattformen, auf denen es läuft, ist das Produkt ein veritables Universalgenie für die Entwicklung und den Betrieb komplexer rechenintensiver Anwendungen, die ganz nach den Anforderungen auch auf verteilten Systemen laufen können.

So gibt es in Produktion laufende Beispiele komplexer Anwendungen, bei denen die eigentliche APL2-Verarbeitung auf Servern (z.B. APL2/AIX auf Workstations bzw. APL2/TSO auf IBM zSeries Großrechnern) läuft, aber die Benutzer auf dem PC schön grafisch geführt werden mit Workstation APL2.

In anderen Fällen dient der Großrechner als der zentrale Datenbankserver mit APL2 und DB2 und damit in verteilter Verarbeitung synchronisiert verbunden läuft die eigentliche APL2-Anwendung auf den PCs der Benutzer.

Oft werden die APL2-Anwendungen unter Windows entwickelt, teilgetestet und gewartet und dann für Systemtest, Produktionsfreigabe und –betrieb z.B. auf den Großrechner hochgeladen. Aber auch der umgekehrte Weg wird begangen, d.h. vorhandene APL2-Anwendungen werden auf dem Großrechner weiterentwickelt und gewartet, dann runtergeladen und auf Workstation APL2 betrieben mit all den Vorteilen, die die PC-Umgebung für den Benutzer bietet, vor allem aber mit der oft genutzten Möglichkeit, die APL2-Anwendung dort z.B. mit Text- und Tabellenverarbeitung zu verknüpfen.

Die Möglichkeit, vorhandene oder schnell und preiswert neu entwickelte APL2-Anwendungsteile in größere, in anderen Programmiersprachen geschriebene Anwendungen nahtlos (d.h. z.B. auch mit automatischer Konvertierung von Argumenten und Ergebnissen) zu integrieren, wird oft genutzt. Ein Beispiel ist eine komplexe Anwendung mit in Java geschriebener Benutzerführung, bei der die Rechenlogik schnell und effizient in APL2 abläuft, ohne dass der Benutzer dies bemerkt. Mittels der Java-Schnittstelle erschließt Workstation APL2 V2 auch die Welt von IBM WebSphere, dem universellen Entwicklungs- und Managementsystem für Internetportale und Webanwendungen, in das APL2-Anwendungen ebenfalls nahtlos integriert werden können.

Manche Unternehmen nutzen die Möglichkeit, in Namespaces paketierte APL2-Anwendungen kostenlos auf beliebig vielen PCs einsetzen zu können, nicht nur im eigenen Unternehmen, sondern sie geben derartige Anwendungen auch an Kundenunternehmen weiter, u.a. als Marketinginstrumente oder Katalogrechner.

Auch wird hierbei manchmal die Möglichkeit genutzt, dass die paketierte APL2-Anwendung sich verschlüsselt mit dem APL2-Server des Anbieters in Verbindung setzt, um z.B. Angebote abzurufen oder Bestellungen zu platzieren. Denn auch APL2-Anwendungen in Namespaces können die meisten genannten Verknüpfungsmöglichkeiten nutzen.

Die wichtigsten Produkt-Merkmale in Kürze

Entwicklungsumgebung

Interaktives Programmieren wird durch einen GUI-Sitzungsmanager unterstützt. Unter Windows sind die Editoren für Programme, Daten und Dialoge integriert mit dem Sitzungsmanager. Die APL2-Entwicklungsumgebung unter Windows ist entsprechend den Kundenanforderungen bevorzugt und besonders gut ausgestattet für die Entwicklung von Anwendungen, die auf anderen Systemen oder Plattformen laufen sollen.

Grafische Benutzerführung (GUI)

Unter Windows liefert APL2 viele erweiterte Werkzeuge zum Aufbau von GUI-Anwendungen einschließlich einer ausgefeilten Tabellenanzeige.

Schnittstelle zu COM

Unter Windows liefert APL2 mit der externen Funktion COM den Zugriff auf das Microsoft-System „Component Object Model“ (COM) und damit die Möglichkeit, mit zahlreichen Anwendungen mit einer ebensolchen Schnittstelle zu kommunizieren, z.B. das Microsoft Office Paket.

Große Arbeitsbereiche

APL2 nutzt die 32-bit-Architektur, die Anwendungen ermöglicht, in Arbeitsbereichen bis nahe an 2 GB zu arbeiten.

(Englische) On-line Dokumentation

„APL2 Users’ Guide“, „APL2 Language Reference Manual“, „ APL2 Language Summary“, „APL2 Programming: Using SQL“; „APL2 Programming: Developing GUI Applications“, „APL2 Programming: Using APL2 with WebSphere“ und „GRAPHPAK User’s Guide“ werden online in PDF-Format mitgeliefert.

Unterstützung von Landessprachen

APL2 akzeptiert Befehle und gibt Nachrichten in verschiedenen Landessprachen aus. Daten in Unicode werden von der TCL-Schnittstelle und vom HTTP-Server unterstützt.

Namespaces

Mit Namespaces lassen sich Arbeitsbereichen einkapslen und damit Namenskonflikte beim Wiederbenutzen vermindern. Die Unterstützung von Namespaces auf Workstations ermöglicht das Entwickeln und Testen von Anwendungen, die als Namespaces auf Großrechnern laufen sollen.

Runtime Library (Ausführungsbibliothek)

Mit dieser Untermenge des APL2-Systems können in Namespaces paketierte Anwendungen kostenlos weitergegeben und auf Benutzer-Maschinen ausgeführt werden, ohne dass die empfangenden Benutzer APL2-Lizenzen beschaffen müssen.

Verbindung mit Programmen in anderen Sprachen

In anderen Sprachen geschriebene Programme kann man einbinden und in der normalen APL2-Syntax aufrufen. Die neue Schnittstelle APL2PI (APL2 Programming Interface) ermöglicht C-, Visual Basic- und Java-Programmen, APL2-Interpreter zu starten, APL2-Funktionen aufzurufen und auf APL2-Datenstrukturen zuzugreifen. Ein Hin- und Herschalten zwischen dem aufrufenden Programm und der APL2-Anwendung ist über „APL2PI Callback“ möglich.

IBM WebSphere und Internet-Werkzeuge

Besonders gut ausgebildet mit Dokumentation und modifizierbaren Anwendungsbeispielen ist die Möglichkeit, APL2-Anwendungen in IBM WebSphere zu integrieren. Der Arbeitsbereich NETTOOLS und unterstützende Dienstprogramme liefern einen HTTP 1.1 Webserver und Client, die einen Datentransfer und „remote procedure calls“ unabhängig vom APL-Dialekt ermöglichen.

Verteilte Verarbeitung

APL2 bietet die Möglichkeit, Daten und Verarbeitung auf mehrere, über TCP/IP verbundene APL2-Systeme zu verteilen. Die APL2-Systeme unter TSO, CMS, AIX , Linux , Sun Solaris und Windows haben die folgenden Einrichtungen:

- Gemeinsame systemübergreifende Variable zum Datenaustausch und zur Synchronisation in einer Systemumgebung zur verteilten Verarbeitung, auch zwischen entfernten APL2-Systemen, beides auch plattform-übergreifend
- Ein „TCP/IP socket interface“ zur Verbindung mit Nicht-APL-Anwendungen auf entfernten Rechnern
- Programmsteuerung einer APL2-Sitzung durch andere Anwendungen oder andere APL2-Benutzer
- Sitzungsmanager für entfernte APL2-Systeme

Dateien als Variable

Ganze Dateien können wie APL2-Variable mit APL2-Elementarfunktionen und -Operatoren verarbeitet werden.

Objektbibliotheken

APL2-Objekte können über ihre Namen gespeichert werden. Objektbibliotheken sind portabel zwischen allen APL2-Systemen.

Universelle Grafikunterstützung

Der Grafikprozessor erzeugt Grafiken, Texte und Bilder von hoher Qualität mit Hilfe einer einfachen Befehlssyntax. Der Arbeitsbereich GRAPHPAK ist kompatibel mit denen aller anderen APL2-Systeme.

Datenbankzugriff

Der Hilfsprozessor AP127 bildet die Schnittstelle zu IBM DB2, AP227 zu Datenbanksystemen und Anwendungen mit ODBC-Unterstützung. In beiden Fällen wird die SQL-Sprache benutzt.

TCL-Schnittstelle

Die TCL-Befehlssprache liefert eine große Vielfalt von Routinen und Werkzeugen, z.B. zur Untersuchung von Zeichenketten, zur Datei-Ein/-Ausgabe und zum Aufbau grafischer Benutzerführungen (GUI). Diese Einrichtungen können über die TCL-Schnittstelle von APL2 benutzt werden.

Performance-Monitor

Der Arbeitsbereich TIME erlaubt die Analyse von APL2-Programmen zum Auffinden von Performance-Engpässen.

Dienstprogramme zur Kompression

Zwei Paare solcher Dienstprogramme erlauben die Kompression bzw. Dekompression von APL2-Datenstrukturen und APL2-Arbeitsbereichen, um plattformunabhängig den schnelleren Versand solcher großer APL2-Objekte über Netze zu ermöglichen.

Externe Funktionen

Eine Vielzahl mitgelieferter externer Funktionen ermöglicht die rationelle Programmierung und schnelle Ausführung von häufig vorkommenden Anwendungsteilen.

Arbeitsbereiche zur Anwendungsentwicklung

Beispiele und Dienstprogramme in 25 mitgelieferten Arbeitsbereichen können direkt benutzt oder für eigene Zwecke abgewandelt werden. Ein Teil der Arbeitsbereiche wird zusätzlich auch als Namespaces zur Verfügung gestellt.

Werkzeuge zum Schreiben von Hilfsprozessoren

Musterprozessoren, C header files und eine Serverschnittstelle erlauben eine leichte Implementierung von Spezial- und Client-/Server-Prozessoren.

System Services

APL2-Schnittstellen zu System-Services werden durch *synchron arbeitende assoziierte Prozessoren* mit Namensassoziierung und durch *asynchrone Hilfsprozessoren* bereitgestellt. Diese Hilfsprozessoren laufen in eigenen Adressräumen und kommunizieren mit zugehörigen Anwendungen über den Shared Variable Processor (SVP) und gemeinsame Variable.

Als synchrone assoziierte Prozessoren werden mitgeliefert:

P10 *REXX-Prozessor:* erlaubt die Benutzung von REXX-Funktionen wie APL2-Funktionen, Manipulation von REXX-Programmen wie APL2-Strukturen und leichten Zugriff auf Betriebssystemdateien.

P11 *Aufruf externer Routinen:* ein APL2-Programm kann damit Routinen außerhalb des APL2-Arbeitsbereiches aufrufen, die sich in einem APL2-Namespace oder in der Bibliothek einer anderen Programmiersprache befinden. zahlreiche solcher externer P11-Routinen und Dokumentationen dazu werden mitgeliefert, sowie auch Musterprogramme als Basis für Entwicklung eigener externer Routinen.

P12 *Dateien als Variable:* Dateien (Textdateien des Betriebssystems und solche im AP210-Format) können wie APL2-Variable mit APL2-Elementarfunktionen und -Operatoren verarbeitet werden.

P14 Ermöglicht den Zugriff auf Java-Felder und –Methoden. Das APL2 Programming Interface (APL2PI) arbeitet mit P14 und bildet die Brücke zu IBM WebSphere.

Mitgelieferte asynchrone Hilfsprozessoren:

AP100 *für Host-System-Befehle:* übermittelt Host-System-Befehle aus APL2-Anwendungen

AP101 *für Alternativ-Eingabe (Stack):* zur Erstellung von programmierten Alternativ-Eingaben an APL2.

AP119 *TCP/IP Socket API:* leitet Aufrufe an TCP/IP weiter, das die Netz-Kommunikation sicherstellt.

AP120 *APL2-Sitzungsmanager:* er ermöglicht dem Benutzer die Kontrolle über sämtliche von ihm gestarteten APL2-Prozesse, selbst auf entfernten APL2-Systemen, auch wenn sie auf einer anderen Plattform laufen.

AP127 *DB2-Prozessor:* zum Zugriff auf IBM DB2 (auch auf verteilte DB2-Systeme) über die SQL-Sprache.

AP144 *X Window API (nur für APL2 auf Unix-Plattformen):* ist die Schnittstelle zu einer großen Zahl von Xwindow-Systemaufrufen (Xlib). Arbeitsbereiche für Cover-Funktionen und Demos sind beigefügt.

AP145 *GUI Serviceprocessor (nur für APL2 unter Windows):* ermöglicht die einfache Verwendung zahlreicher Objekte der Windows Services zur grafischen Benutzerführung. Arbeitsbereiche für Cover- und Beispiel-Funktionen sowie Demos sind beigefügt.

AP207 *Universeller Grafikprozessor:* erzeugt Grafiken, Texte und Bilder von hoher Qualität mit Hilfe einer einfachen Befehlssyntax. Arbeitsbereiche für Demos und GRAPHPAK sind beigefügt.

AP210 *Dateiprozessor:* zum Schreiben und Lesen von Dateien des Betriebssystems. Diese können Texte, binäre Daten und APL2-formatierte Objekte enthalten.

AP211 *APL2 Objektbibliotheks-Prozessor:* zum Schreiben und Lesen von APL2-Strukturen über ihre Namen. Die damit erzeugten Dateien sind portabel auf alle APL2-Plattformen (die von APL2/DOS sind lesbar).

AP227 *ODBC-Prozessor:* zum Zugriff über SQL auf alle Datenbank-Systeme und Anwendungen, die das ODBC-Protokoll unterstützen.

AP488 *GPIB-Prozessor:* zum Zugriff auf externe Geräte, die über den General Purpose Interface Bus (GPIB) und National Instruments GPIB Treiber angeschlossen sind. (nur für Windows Systeme).

Technische Informationen

Hardware-Voraussetzungen:

- Mindestens 25 MB freier Plattenplatz
- Mindestens 15 MB Hauptspeicher über den Bedarf des Betriebssystems hinaus

Software-Voraussetzungen:

Workstation APL2 Version 2.0 hat die folgenden Software-Mindestvoraussetzungen:

- AIX 5.0 mit Motif 1.2 und X Window System X11R5
- Eine Linux-Distribution für PC-kompatible Maschinen mit Motif 2.1 und X Window System X11R6
- Solaris 2.5 mit Motif 1.3 und X Window System X11R5
- Windows 98, ME, NT, 2000 oder XP

Einige Einrichtungen von Workstation APL2 haben zusätzliche Mindestvoraussetzungen:

- Adobe Acrobat Reader ist erforderlich zur Darstellung der Online-Dokumentationen.
- IBM DB2 Universal Database, Version 5.0 für AP 127
- Ein ODBC Driver Manager, der ODBC 3.0 Application Programming Interfaces unterstützt, für AP 227.
- TCP/IP für AP 119 und systemübergreifende gemeinsame Variable.
- IBM Object REXX zum Aufruf von REXX-Routinen mit Prozessor 10
- TCL und TK 8.1 für die externe TCL-Funktion.
- Eine lauffähige Java-Version für Kompression/Dekompression

Englische Dokumentationen:

Workstation APL2 Version 2.06 enthält die folgenden englischen Dokumentationen im PDF-Format:

➢ APL2 User's Guide	SC18-7021
➢ APL2 Programming: Language Reference	SH21-1061
➢ APL2 Programming: Using SQL	SH21-1057
➢ APL2 Programming: Using APL2 with WebSphere	SH18-9442
➢ APL2 Programming: Developing GUI Applications (nur für Windows)	SH18-7383

- APL2 GRAPHPAK: User's Guide and Reference SH21-1074
- APL2 Language Summary SX26-3851

Kompatibilität:

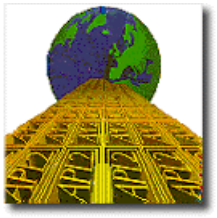
Workstation APL2 V2.0 ist kompatibel mit den APL2V1-Produkten für AIX, OS/2, Sun/Solaris und Windows. Mit diesen Produkten geschriebene Anwendungen laufen auch auf Workstation APL2 V2.0, das aber ein neues Format für die Arbeitsbereiche benutzt. D.h., mit V2.0 gespeicherte Arbeitsbereiche können nicht in V1.0-Systemen geladen, sondern nur durch)OUT und)IN zurück migriert werden .

Hinweis:

Diese Veröffentlichung von DPC ist eine auf deutsche bzw. europäische Verhältnisse angepasste Übertragung des IBM Dokuments „Workstation APL2 for Multiplatforms Version 2.0 Technical Brief" vom Dezember 2004 und weiterer IBM Veröffentlichungen vom Mai 2005. Sie kann nur allgemeine Hinweise geben. IBM, APL2, OS/2, AIX, und DB2 sind Warenzeichen der IBM Corporation. Sun und Solaris sind Warenzeichen von Sun Microsystems, Inc. Windows, Windows 98, Windows ME, Windows NT, Windows 2000, Windows XP und COM sind Warenzeichen von Microsoft Corporation. Linux ist ein Warenzeichen von Linus Torvalds. Adobe und Acrobat sind Warenzeichen von Adobe Systems, Inc. Motif ist ein Warenzeichen von The Open Group.

Workstation APL2 Version 2.0 kann bei DPC nicht nur neu bestellt, sondern auch von einem der Vorgängerprodukte APL2 Version 1.0 für AIX, OS/2, Solaris oder Windows aufgerüstet werden, beides über das IBM-Vertriebsprogramm „Passport Advantage".Weitere Informationen findet man auf den APL-Webseiten von IBM oder von DPC:

- http://www.ibm.com/software/admtools/apl
- http://www.dpc.de



Kontakt:

Dittrich & Partner Consulting GmbH, Kieler Str.17, D-42697 Solingen, Tel. 0212-26066-0, Fax: -26066-66, eMail: info@dpc.de

IBM „APL Products and Services": eMail: apl2@vnet.ibm.com, Fax: 001-408-463-4488, Tel. 001-408-463-APL2 (001-408-463-2752)

Der Buchtip

GUI-Utilities für Dyalog APL/W

Ein Buch von Kai Jäger, erschienen im Selbstverlag (BoD), erhältlich im Buchhandel und bei Internet-Anbietern wie Libri und Amazon mit der ISBN-Nr. 3-8311-1831-0.

Das Buch beschreibt frei verfügbare Software, mit deren Hilfe die ohnehin hohe Produktivität eines APL-Programmierers noch beträchtlich erhöht werden kann. Diese Software reduziert den Aufwand für die Erstellung eines GUIs erheblich, wenn die Aufgabenstellungen aus dem kaufmännischen Bereich stammen.

Das Buch wendet sich an den gestandenen APL-Programmierer, der Anwendungen realisiert. Für den APL-Neuling ist es nicht geeignet. Es ist insbesondere keine Einführung in die GUI-Programmierung mit APL. Es ist nur für Dyalog APL/W nutzbar.

Wer GUIs erstellen muss, der wird in der verfügbaren, außerordentlich mächtigen Software eine große Hilfe finden. Mit Hilfe dieses Buches ist auch die Einarbeitung in die grundlegenden Konzepte kein Problem mehr.

Wer APL als mathematisches Werkzeug für statistische Untersuchungen oder zur Datenanalyse benutzt, wird die hier vorgestellte Software nur dann sinnvoll nutzen können, wenn er zumindest gelegentlich auch GUIs erstellen muss.

Nebenbei lernt man noch frei verfügbare Software kennen, welche die Verwendung von Software-Modulen anderer Autoren in eigenen Applikationen stark vereinfacht. Auch Programmier-Teams werden durch diese Software unterstützt.

Weitere Informationen finden Sie unter: http://www.kai-jaeger.de

Nancy Wheeler

Workstation APL2 Version 2 Service Level 6

In May 2005, Service Level 6 to Workstation APL2 Version 2 was made available. In addition to the usual fixes for reported problems, this service level contains several interesting enhancments that address requirements submitted to IBM by the German customers of APL2.

Compression Utilities

A new set of external functions provides the ability to compress APL2 objects. The resulting compressed objects take up much less space than the objects in their original form, and thus are more economical when sent across networks. There are two pairs of compression functions.

- ZIP and UNZIP operate on individual arrays. The array is passed as an argument to ZIP and the compressed object is returned as its result. To uncompress, pass the compressed object to UNZIP and the original array is returned as the result.
- ZIPWS and UNZIPWS operate on the entire active workspace. ZIPWS compresses each object in the workspace and catenates them into a single compressed object. Once the compressed object is created, the original objects are erased, and)SAVE will produce a much smaller workspace file. After)LOAD, UNZIPWS is used to restore the original objects and erase the compressed object. The argument to both ZIPWS and UNZIPWS is the name to be used for the compressed object.

The compression utility functions are made possible by the use of built-in facilities of Java and the APL2 Java interfaces.

COM Interface for Windows

A new external function provides access to the Microsoft *Component Obect Model* (COM) from APL2.

The Microsoft Component Object Model is a system for creating software components that can interact. COM is the foundation technology for Microsoft's OLE (compound documents), ActiveX (Internet-enabled components), and others.

Many applications, such as Microsoft's Office suite of products and elements of Windows, are distributed as COM components. Through the COM external function, these applications can be used from APL2.

COM is an object-oriented system. This means that COM components are distributed as *classes* and operate as *objects*. An application, such as APL2, creates an *instance* of a COM class to create a COM *object*. Once an object is created, the application can use *methods* and *properties* within the object. When the application has finished using the object, it releases the object to free its resources.

The following log demonstrates using COM to control Microsoft Word:

```
      3 11 ⎕NA 'COM'
1
      ⍝ Start Microsoft Word
      WORD←COM 'CREATE' 'Word.Application'

      ⍝ Open a document
      FILE←COM 'METHOD' WORD 'Documents.Open' 'c:\sample.doc'

      ⍝ Get the builtin style constants
      CONS←COM 'QUERY' 'CONSTANTS' WORD 'WdBuiltinStyle'

      ⍝ Get the value of the constant for level 1 heading style
      INDEX←CONS[CONS[;1]⍳⊂'wdStyleHeading1';2]

      ⍝ Get the font used for level 1 headings
      COM 'PROPERTY' FILE 'Styles.Item().Font.Name' INDEX
Verdana
      ⍝ Set the font used for level 1 headings
      COM 'PROPERTY' FILE 'Styles.Item().Font.Name' INDEX 'Arial'

      ⍝ Save the changes
      COM 'METHOD' FILE 'Save'

      ⍝ Close the file
      COM 'METHOD' FILE 'Close'

      ⍝ Release the file
      COM 'RELEASE' FILE
      ⍝ Close Word
      COM 'METHOD' WORD 'Quit'

      ⍝ Release Word
      COM 'RELEASE' WORD
```

The COM function opens up many new possibilities for combining APL2 with other Windows programs.

Calls to APL2 from Visual Basic

Visual Basic has been added to the list of languages and programs that are supported by the APL2 Program Interface (Calls to APL2). A set of subroutines is provided for use as Visual Basic external procedures. These routines allow you to manage an APL2 session from within your Visual Basic program. For example, Excel macros written in Visual Basic can call APL2 to perform calculations.

The Visual Basic to APL2 interface is similar to that already provided for C and Java. Routines are provided to start and end an APL2 session, manage APL2 objects in the workspace, convert APL2 objects to and from Visual Basic *variant* form, and execute APL2 functions and expressions. A complete sample Visual Basic program is shown in the *APL2 User's Guide* along with examples of how to call each routine.

Sample C Program for Calls to APL2

While the *APL2 User's Guide* does include program segments that show how to use the APL2 Programming Interface to request the various services supported by the C interface, it does not show a complete, working C program that demonstrates the entire process. Such a program is now being shipped with APL2. On Windows, find it in the APL2 \samples directory. On Unix systems, it is located in the /examples/qna directory.

Windows Session Manager

A new option has been added to the Session Manager's *Object Popup Menu* on Windows. If the mouse pointer is positioned over the name of a variable, a right click and selection of the *Display* option will open a window containing the output of the APL2 DISPLAY function for that variable.

AP 145 Extensions

Several extensions have been made to AP 145 in this service level:

- The Treeview control is added to the list of Windows Common Controls supported by APL2.
- The new NAME property allows controls within a dialog to be identified by character names rather than numbers.
- The FOCUS property for Listview controls returns the index of the item that currently has focus.
- New events supported are Kill focus and Set focus for Listview controls, Start Menu for dialogs, and Context Menu.
- The Resizable style for Grid controls allows new rows and columns to be added under program control.

Kontakt:

Nancy Wheeler , eMail: nwheeler@us.ibm.com

Die längste elektronische Schultafel Berlins

FU-Informatiker entwickeln intelligenten Hörsaal

Berlin. Der erste elektronische Hörsaal ist an der Freien Universität Berlin eingeweiht worden. Kernstück des Vorlesungssaals sind vier berührungssensitive Projektionsflächen mit einer Gesamtlänge von sechs Metern. Berlinweit handelt es sich hierbei um den größten, berührungsempfindlichen Computerbildschirm. Der Dozent kann an den elektronischen Bildschirmen direkt arbeiten, Worte oder Diagramme wie an einer herkömmlichen Kreidetafel anbringen. Doch die Tinte ist nur noch digital: Statt Kreide benutzt der Dozent einen Laserstift, mit dessen Hilfe das Tafelbild von einem Computer erzeugt wird. Der Vorteil für die Studierenden liegt auf der Hand: Ist die Vorlesung zu Ende, steht die Transkription bereits im Netz. Die Studierenden können sich auf die Vorlesung konzentrieren und müssen nicht mehr mitschreiben. Das Institut für Informatik hat den "intelligenten Hörsaal" vor wenigen Wochen eingeweiht, doch das E-Kreide-Projekt läuft bereits seit fast fünf Jahren und hat europaweit mehrere Preise gewonnen.

Neben der Benutzung als herkömmliche Tafel sind Bilder und interaktive Internet Dienste während des Vortrags in das Tafelbild integrierbar. Diese können direkt von der Festplatte oder aus dem Internet geladen werden. E-Chalk bietet ausserdem Formelauswertung und Funktionsplot über eine Handschrifterkennung. Der Zuhörer kann den Vortrag im Hörsaal oder zu Hause mitverfolgen. Hierzu wird nur ein normaler Browser benötigt. E-Kreide überträgt und speichert gleichzeitig Tafelbild, Audio und Video. Vorlesungen werden so gespeichert, daß sie jederzeit aus dem Internet abspielbar sind. Von dem Tafelbild wird außerdem ein PDF erzeugt, so daß die Zuhörer in der Veranstaltung nicht mehr mitschreiben müssen.

Moderne E-Tafel: Statt Kreide benutzt der Dozent einen Laserstift (Foto: FU Berlin)

Die intelligente Kreide-Tafel kann die Dozenten bei verschiedenen Aufgaben unterstützen: Die Tafel übersetzt Sprachen, kann mathematische Funktionen als Bilder wiedergeben oder logische Schaltungen erkennen und zum Laufen bringen. Auch Computerprogramme können handschriftlich eingegeben werden und sind sofort lauffähig.

Rojas schreibt in grüner Farbe die mathematische Formel f(x,y) = xy an die E-Tafel und während er noch erklärt, baut sich auf der Plasmaoberfläche langsam der zugehörige dreidimensionale Graph auf. "Hierzu benutzen wir einen kommerziellen algebraischen Server", sagt der Informatik-Professor. Wer wissen möchte, was das Wort "Kreide" auf Englisch heißt, braucht es ebenfalls nur auf die Tafel schreiben und schon erscheint der englische Begriff "chalk" - allerdings ebenfalls nur, wenn über einen Script eine Verbindung zum Internet besteht und der Dozent zuvor ein "bookmark" zu der entsprechenden Seite gesetzt hat.

Kontakt: Prof. Dr. Raúl Rojas, Institut für Informatik der Freien Universität Berlin, Projekt E-Chalk, Takustr. 9, D-14195 Berlin, Tel. 030.838-75130, eMail: rojas@inf.fu-berlin.de, Internet: www.e-kreide.de, www.inf.fu-berlin.de/.

Axel Holzmüller

APL2 und Java

vorwärts und rückwärts

Rückblende: Java

- objektorientierte Progammiersprache
- plattformunabhängig
- Klasse – Objekt
  Attribute – Instanzvariablen
  Verhaltensweisen – Methoden
- Java erlaubt Multithreading
- Zahlreiche Klassenbibliotheken
- Entwicklungsumgebung
  - Java SDK (incl. JRE)
  - Eclipse

Processor 14 - Calls to Java

▶ Name Association Syntax:

```
(class signature) 14 NA 'surrogate member'
```

```
class ←! 'javax/sound/midi/MidiSystemé
signature ←! '()Ljavax/sound/midi/Synthesizer;'
member ←! 'getSynthesizer'
```

(getSynthesizer ist danach eine Funktion)

P14 Hilfsfunktionen

… aus 2 DEMOJAVA

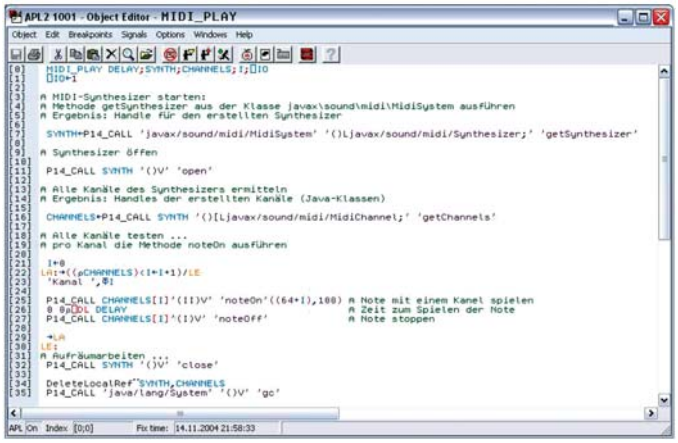
➢ P14_ASSOC
P14_ASSOC class signature member
statt „14 NA“

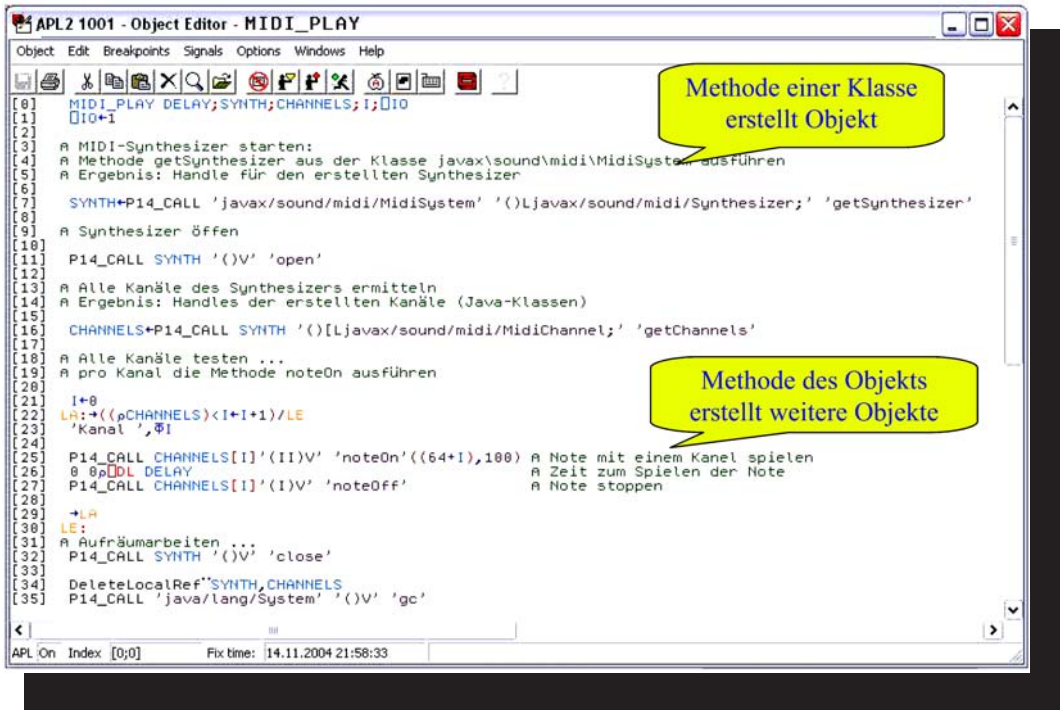
➢ P14_CALL
P14_CALL class signature member parameter
führt Java-Methode aus

➢ P14_MAKE
P14_MAKE class signature parameter
erstellt eine Instanz einer Klasse

APL2 macht Musik

▶ „Calls to Java“ – Beispiel: MIDI_PLAY





Tor zur Java Welt

- hilfreiche Java-Kassen und -Applikationen
  - Graphik, z.B. VTK
  - WebServices (SOAP, XMP-RPC)
  - XML (SAX und DOM-Parser)
  - … u.v.m.
- Links
  - java.sun.com
  - www.apache.org
  - www.ibm.com

Calling APL2 from Java

- Klassen in com.ibm.apl2 (apl2.jar)
  - Apl2interp - führt APL2-Anweisungen aus
  - Apl2object - konvertiert Daten zwischen Java und APL2
  - Apl2cdr - konvertiert Daten zwischen Workspace- und CDR-Formaten (zum Übertragen von Variablen zwischen APL2-Interpretern)
  - Apl2exception - behandelt Fehler während der Ausführung von APL2-Anweisungen

Ein ganz einfaches Beispiel (siehe APL2 User's Guide, S. 501)

```
Apl2interp Slave = new Apl2interp(new String[] {„-ws“,“1m“});
Apl2object Time = Slave.Execute(Apl2interp.QTS);
int[] TimeArray = Time.intarrayValue();
Time.Free();
Slave.Stop();
System.out.println(„The time is „ + TimeArray[4] + ‘:’ +
TimeArray[5] + ‘.’) ;
```

… was zu beachten ist

Definierte APL2-Funktionen ausführen:

- ➡ nur mit Funktionen eines Namespaces
  - Apl2interp Objekt instantiieren
  - Funktion aus Namespace assoziieren
  - Funktion (evtl. mit Argumenten) ausführen
  - evtl.: Ergebnis weiterverarbeiten

➢ Konvertierung von Argumenten und Ergebnis mit Apl2object

Wozu das alles?

- APL2 Anwendungen können in Java-Projekten genutzt werden
- Stärken von APL2 stehen Java Entwicklern zur Verfügung
- Java als „Connector“ zwischen APL2- und anderen Anwendungen
- Nutzung von APL2 innerhalb Java-Anwendungen (z.B. SAX XML)
- APL2 unter Java Server Pages
- „Multithreaded APL2“

APL2Env (Rückblende)

… zeigt aktuelle APL2-Umgebung an

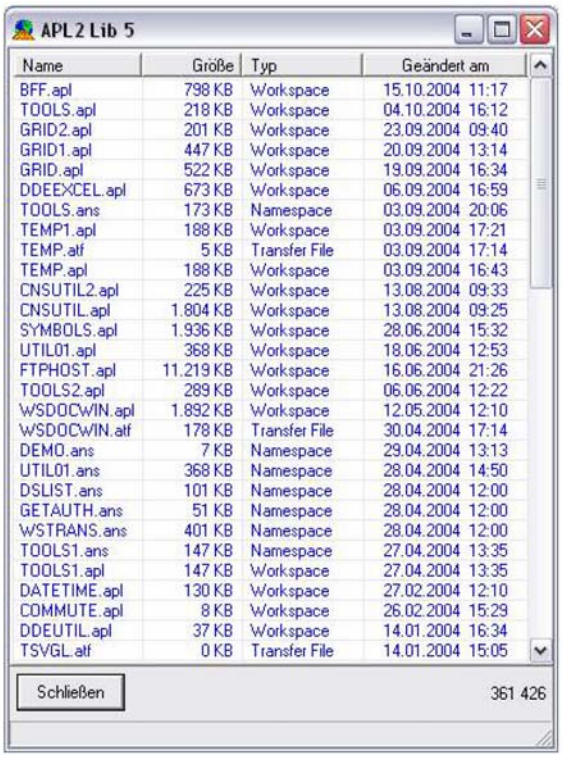
- Bibliotheksdefinitionen
- Weitere APL2-Umgebungsvariablen
- APL2 Aufrufsoptionen
- Prozessor 11 Namen

Aber: Session Manager ist während der Anzeige gesperrt

➡ Lösung:

Multithreaded APL2

Lösung: Ausführung in eigenem Thread



aber: kein Multithreading unter APL2

Java unterstützt Multithreading

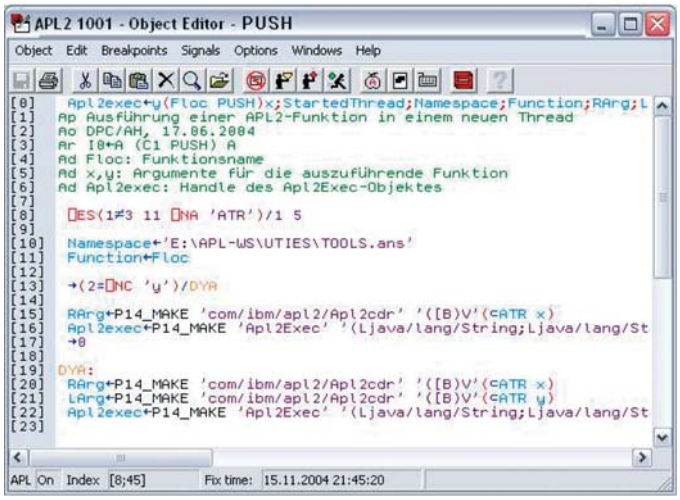
➡ APL2 – Java – APL2:

Aus 1 mach 2 …

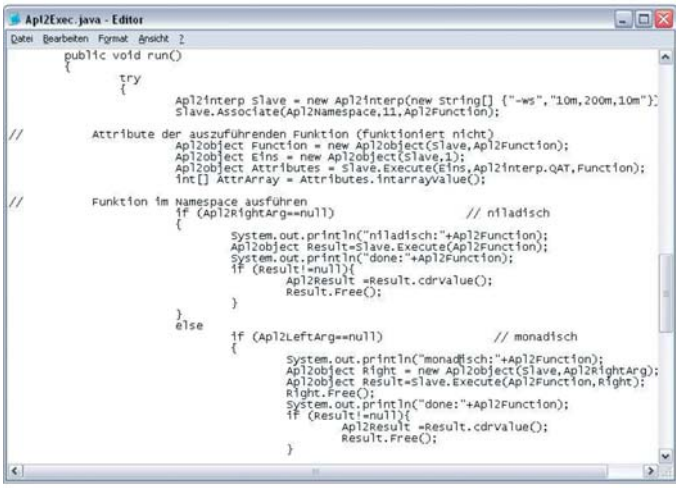
Vorwärts und rückwärts

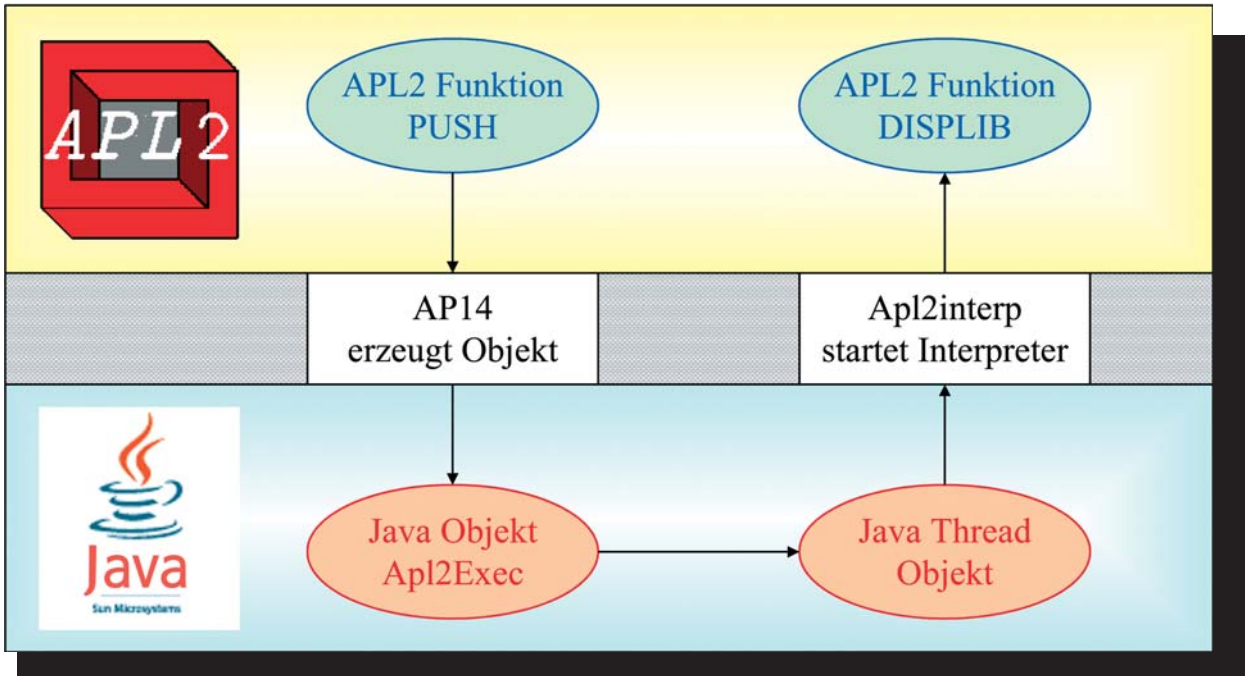
Die Dienstleister

PUSH



Apl2Exec

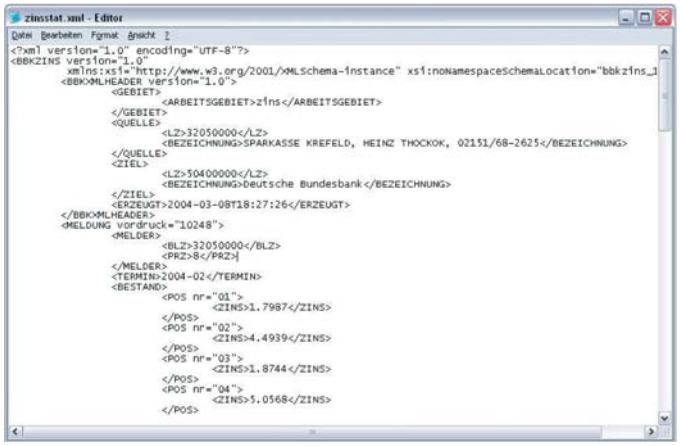




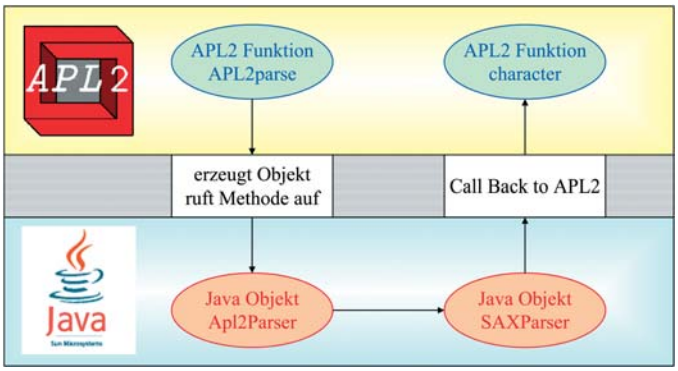
… und mehr

- weitere Tools: DISPLIB, LastFns, DISPVAR
- Parallele Ausführung von Funktionen
- APL2-Server

... und XML-Parser unter APL2



APL2 und SAX



Tor zur Java Welt

- Bemerkenswertes:
  - Session ID: unter Java gestartete Interpreter erhalten normale Session ID
  - Classpath
- Java erlaubt Multithreading:
  „multithreaded APL2“, parallele Ausführung

```
(⊂'COUNT')PUSH¨10 20 15 5
```

Tor zur Java Welt

„Multithreading“ für APL2 (nah und fern)

Java als „Connector“ zu anderen Anwendungen

- Anschluß an aktuelle Trends: XML, WebServices u.a.

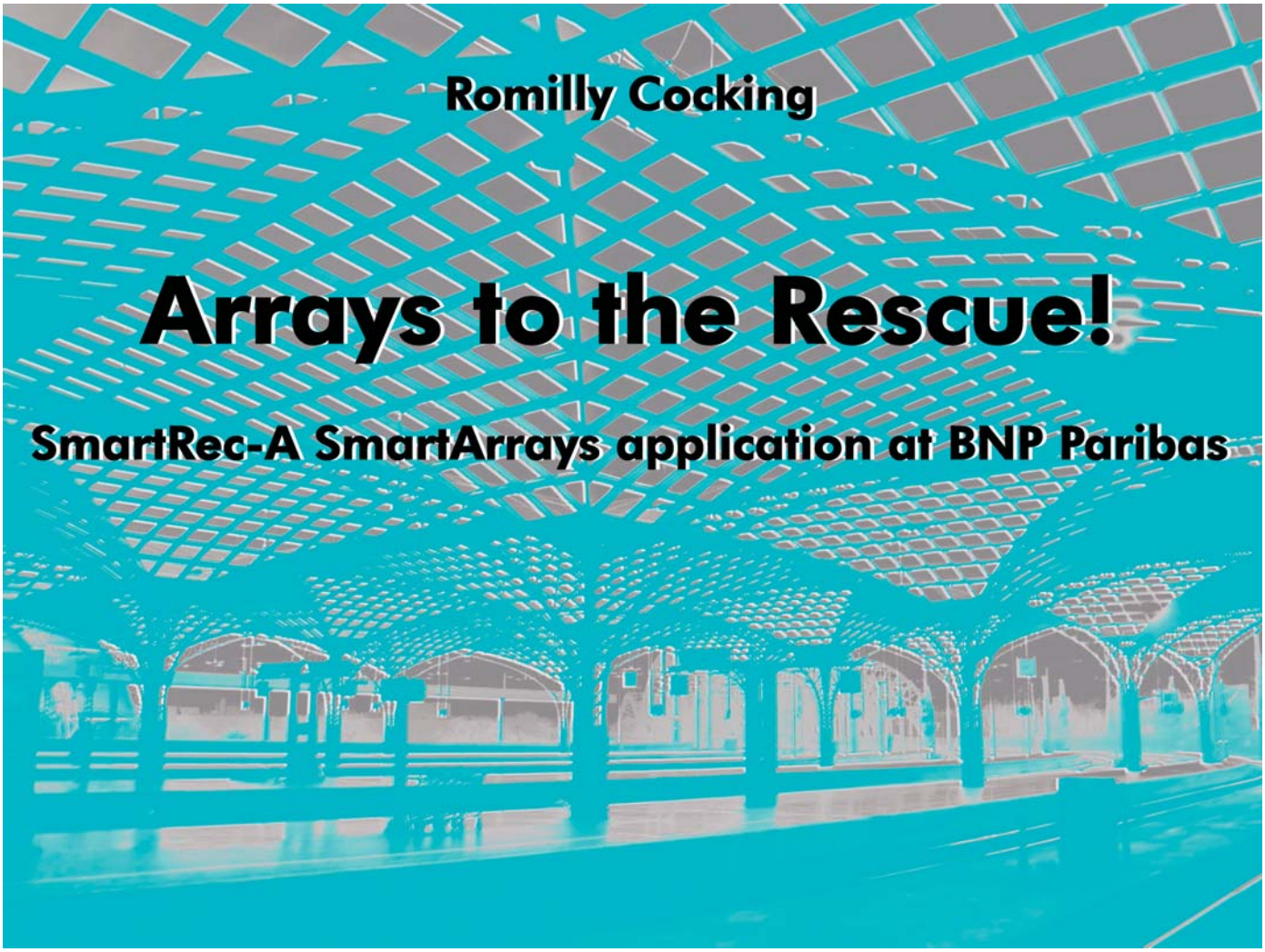
Kontakt:

Axel Holzmüller,
eMail: axel.holzmueller@dpc.de

Dittrich & Partner Consulting GmbH, Kieler Str.17, D-42697 Solingen, Tel. 0212-26066-0, Fax: -26066-66, eMail: info@dpc.de

APL - Germany e.V.

http://www.apl-germany.de



Romilly Cocking

Arrays to the Rescue!

SmartRec-A SmartArrays application at BNP Paribas

SmartRec

- A Configurable Reconciliation Engine
- Implemented in Java
- Based on SmartArrays

SmartArrays

- Very fast array engine
- Inspired by APL
- Developed by James Wheeler
  - ❑ APL*Plus
- And Jim Brown
  - ❑ APL2
- For C++, Java, .Net developers

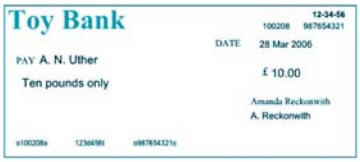
What is Reconciliation?

- A Business Process
- Matches and Compares
- Two or more sets of records
  - ❑ that should tell the same story

example

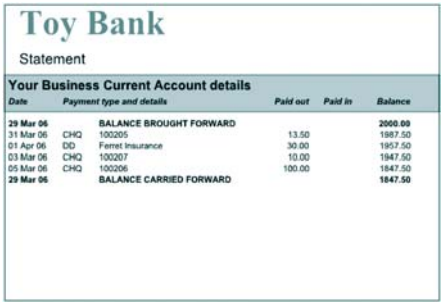
- Personal banking
- Comparing chequeswith a bank statement

A Cheque



Account	Ch. Number	Date	Amount	Payee
987654321	100208	28/03/06	£10.00	A.N.Uther

A Bank Statement



What banks want to reconcile

- Trade data
  - ❑ Internally (Front and Back office)
  - ❑ Externally (Buyer and seller)
- User data
- Accounting data

Key concerns

- Flexibility
- Reliability and accuracy
- Performance
  - ❑ Most banks use RDBMS
  - ❑ Volumes of data very large
  - ■ Example - 2.000.000 records vs 2.000.000 records
  - ■ Currently takes *4 hours* on RS6000

Where data comes from

- Flat files
- Spreadsheets
- Databases
- XML

First results

- Complex reconciliation
- User data
- 14937 vs. 37314 records
- Old system:

- ~ 90 *minutes*on RS6000
- SmartRec: ~90 *seconds*on this laptop
  - Now ~ 30 seconds using SmartArrays v3 + toolkit

SmartRecstatus

- Will replace existing solution (RAID) throughout BNPP
- Longer term, will replace 3rd party products
- Plan to make SmartReca product (this year)

ne technology

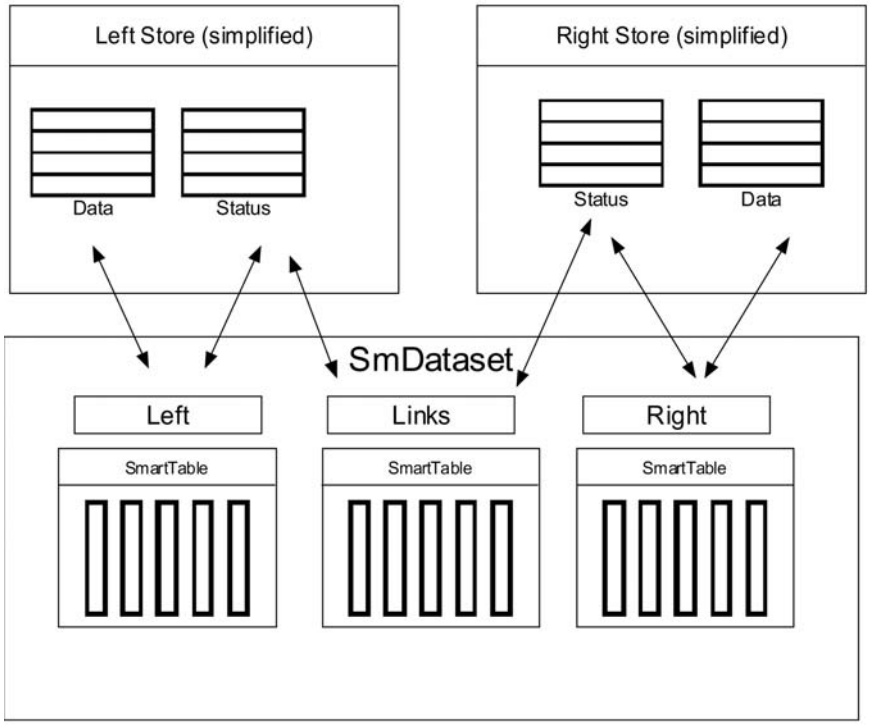
Requirements

- Linking and comparing
- Transformations
- Filtering

Architecture

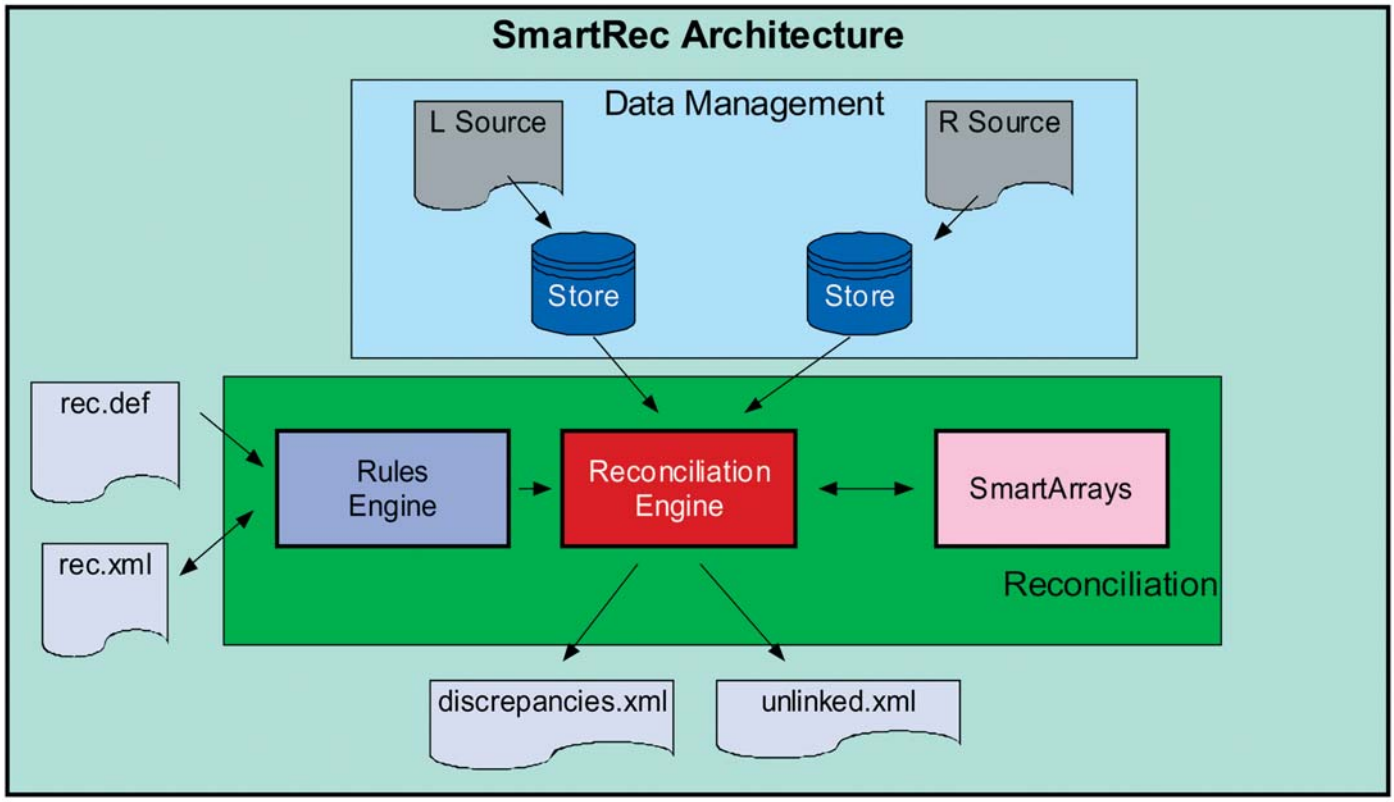
- Compiler
  - Turns specification into objects that define what is to be done
- Data Manager
  - Loads data
  - Remembers unlinked records from previous runs
- Reconciliation Engine
  - Interprets and executes the reconciliation
    - Filters and Transforms data
    - Links and compares data
    - Creates outputs

From database to array and back



SmartArrays code

```
SmArraycp = keys.eq(SmArray.vector(1).
  rotateAxis(keys, 0)).reduceAxis(Sm.and, 1);
SmArraybv= SmArray.scalar(0).catenate(cp)
  .dropBy(-1);
bv= bv.lt(bv.rotateBy(1));
```



Internals

- 1-1 reconciliation
- 1-m reconciliation
- m-nreconciliation
- Grouping and Joining
- XML reconciliation

Reconciliation specification

- Declarative language
  - (John Scholes would say Izzy)
- Orginally created to help me write tests
- When users saw it, they wanted it

Rec.def

```
reconciliation {
   stores {
     leftdocs {
       source delimited separator = ";".
      fields {
              id(),
              sale_price(name="sale price", number),
              cost(name="cost", number),
              cp(name="counterparty")
      }
      working {
              profit(name="we made", number)
            }
            transformations {
              profit = sales - cost.
            }
       }
       rightdocs {
         source delimited separator = ";".
           fields {
             id(),
             profits(name="profit", number),
             cp(name="counterparty")
     }
   }
  model model1 {
  link id to id.
  compare cp to cp.
  compare profit to profits.
  }
}
```

1-1 Reconciliation

Model 1		
Object A1		**Object B1**
Portfolio = 105	Matching Criterion	Portfolio = 105
Amount = 10,000	Matching Criterion	Amount = 10,000
Currency = GBP	Discrepancy Criterion	Currency = GBP
Location = London	Discrepancy Criterion	Location = Paris

1-m reconciliation (complex)

Model 4		
Object A5		**Object B5**
		Object B5 ID = ZOZ
Object A5 ID = YOY	Matching Criterion	Reference A5 ID = YOY
Portfolio = 109	Discrepancy Criterion	Portfolio = 109
Amount = 10,000	Discrepancy Criterion	Amount = 12,000
Currency = GBP	Discrepancy Criterion	Currency = GBP
Location = London	Discrepancy Criterion	Location = London
Object A5		**Object B6**
		Object B6 ID = BOB
		Reference B5 ID = ZOZ
Object A5 ID = YOY		no reference to A5 ID
Portfolio = 109	Discrepancy Criterion	Portfolio = 110
Amount = 10,000	Discrepancy Criterion	Amount = 10,000
Currency = GBP	Discrepancy Criterion	Currency = GBP
Location = London	Discrepancy Criterion	Location = London

Evolution

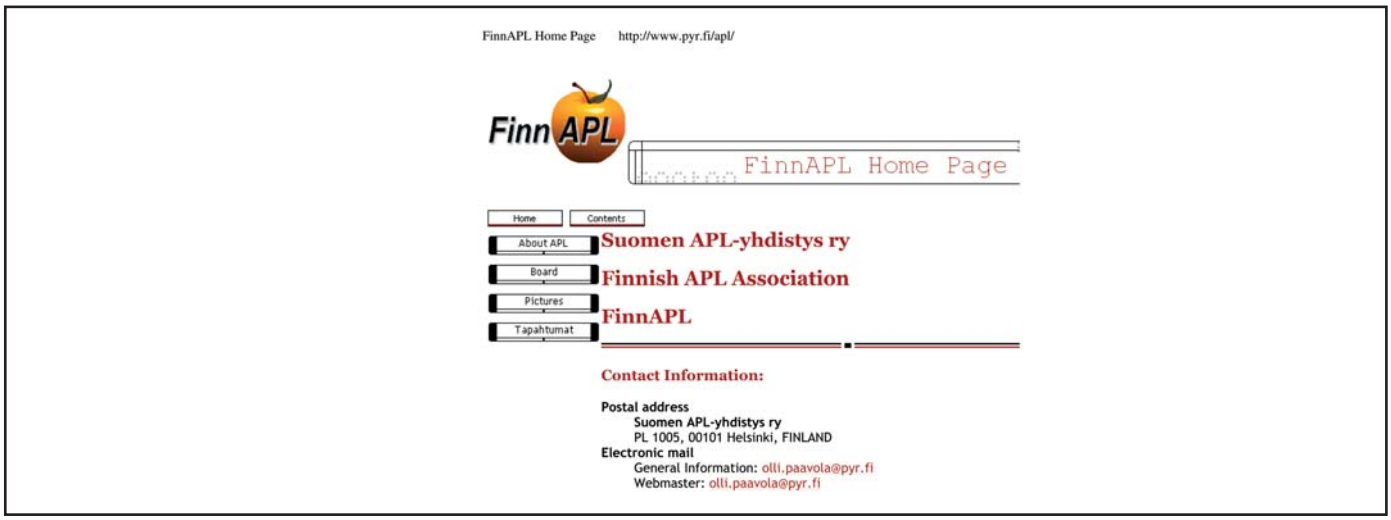
- Server based version
- Production version
- Product family
- GUIs
  - for reconciliation definition
  - for results

Conclusions

- BNP Paribas delighted
  - Outstanding Performance
  - Astounding Progress
  - Sees SmartArrays as key to success
- Arrays worked wonders!
  - Now spreading within BNP Paribas
  - Interest from other UK banks

Kontakt:

Romilly Cocking,
http://www.cocking.co.uk



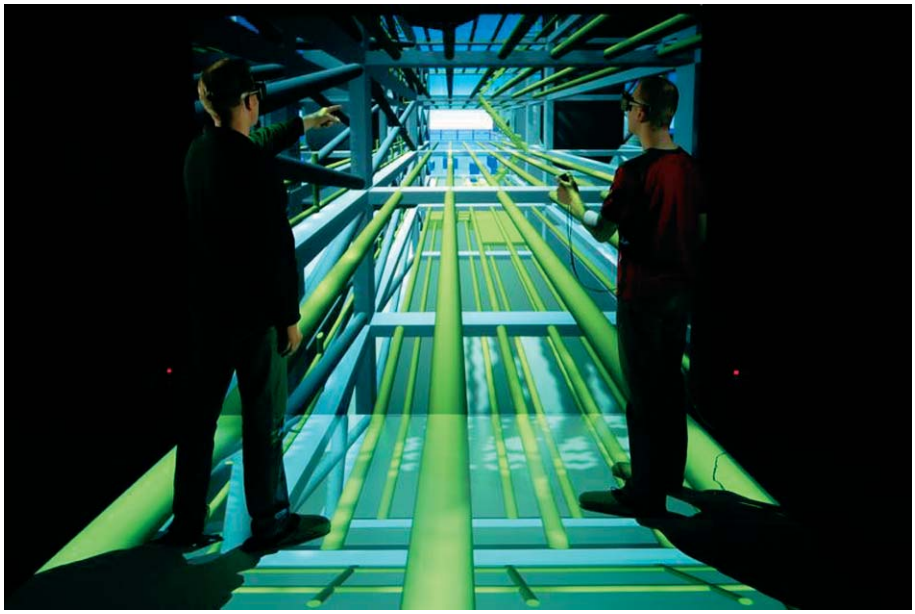
TwoView: Neues Display-System für Arbeiten im virtuellen Raum

Neue Technik ermöglicht räumliche Präsenz virtueller Objekte und kann in der Industrie eingesetzt werden

Sankt Augustin/Düsseldorf. Das Fraunhofer-Institut für Medienkommunikation IMK in Sankt Augustin und die Firma Barco haben ein neuartiges, projektorbasiertes Display-System entwickelt, das eine exakte Zusammenarbeit und gemeinsame Interaktion von zwei Benutzern in einer virtuellen Umgebung ermöglicht. Mit dem Display-System, das TwoView, werden die gemeinsame Analyse von komplexen Daten und das kollaborative Engineering im virtuellen Raum entscheidend besser als mit den bisher üblichen Systemen unterstützt, berichtet das IMK.

Im Gegensatz zu konventionellen Display-Systemen wie der CAVE, die nur ein einziges Bild für mehrere Betrachter darstellen, wird beim TwoView-Display für jeden der beiden Benutzer ein eigenes stereoskopisches 3D-Bild erzeugt. Dies ermöglicht es, virtuelle Objekte für jeden Betrachter räumlich absolut korrekt und deckungsgleich darzustellen. "Wenn zwei Benutzer eine Darstellung exakt am selben Punkt im Raum wahrnehmen, wird eine völlig neue Präsenz virtueller Objekte erzielt", erläutert Andreas Simon, der das System am Fraunhofer IMK als Display-Spezialist konzipiert hat. Die "Echtheit" eines virtuellen Objekts lässt sich ganz anders akzeptieren, wenn man beobachten kann, dass eine andere Person diesen Gegenstand genau so sieht wie man selbst. Erst so wird eine wirkliche Zusammenarbeit im virtuellen Raum ermöglicht.

Vergleichbare Displays wurden bisher nur als Forschungsprototypen aufgebaut. Mit dem TwoView-Display verfügt das Fraunhofer IMK erstmals über ein System, das den produktiven Einsatz dieser Technik im industriellen Umfeld ermöglicht. Das TwoView wird zum Beispiel zukünftig im VRGeo-Projekt (http://www.vrgeo.org) genutzt, in dem Anwendern aus der Öl- und Gasindustrie die Zusammenarbeit beim Entwurf aufwendiger Installationen, etwa bei der Planung von Bohrlöchern, ermöglicht wird und in dem die Zusammenarbeit von Experten bei der Analyse komplexer geowissenschaftlicher Datensätze mit Hilfe von Virtual-Reality-Technologie verbessert wird. Weitere unmittelbare Anwendungsfelder des TwoView-Displays sind das kollaborative Engineering in der Automobilindustrie und auch die Diagnose in der Medizintechnik.



L-Shape: Zusammenarbeit im virtuellen Raum (Foto: Fraunhofer IMK)

Kontakt: Fraunhofer-Institut für Medienkommunikation IMK, Schloss Birlinghoven, D-53754 Sankt Augustin, Thorsten Holtkämper, eMail: thorsten.holtkaemper @imk.fraunhofer.de, Tel: 02241/14-3419, Dr. Manfred Bogen, eMail: manfred. bogen@imk.fraunhofer.de, Tel: 02241/14-2367, Internet: www.imk.fraunhofer.de/de/twoview
Barco GmbH, Tiefenbroicher Weg 35/A2, D-40472 Düsseldorf, Internet: www.barco.com.

Quantencomputer aus gewöhnlichen Transistoren

Kleiner Elektronen-Spin soll Rechentechnik revolutionieren

Los Angeles. Forschern der Universität von Kalifornien (UCLA) ist es gelungen, den Spin eines Elektrons in einem gewöhnlichen Transistor-Chip zu verändern. Dabei entdeckten sie, dass damit auch die Ladung wechselte. Ihre Versuche gelten als wichtiger Schritt für die Konstruktion von Quantencomputern.

Um den Elektronen-Spin zu ändern nutzten die Wissenschaftler eine Radio-Frequenz im Mikrowellenbereich. „Unser Versuch hat gezeigt, dass ein ganz einfacher Transistor, wie er in einem gewöhnlichen PC oder Mobiltelefon verwendet wird, für den Bau von Quantencomputern geeignet ist“, erklärte UCLA-Professor Hong Wen Jiang.Durch das gelungene Experiment soll der Bau von Quantencomputern weit realistischer werden, da der Spin eines einfachen Elektrons schließlich ein Q-Bit, den Grundbaustein eines Quantencomputers repräsentiert.

Bevor sich der Erfolg einstellte, hatten die Forscher um UCLA-Professor Hong Wen Jiang und den Doktoranden Ming Xiang mit Mio. von Elektronen experimentiert.Für den gelungenen Versuch arbeiteten Jiang und Xiang mit gewöhnlichen Transistoren bei minus 400 Grad Fahrenheit. In Zukunft soll die Technik aber auch bei normalen Temperaturen funktionieren. Dadurch soll die kommerzielle Nutzung vereinfacht werden. (pte)
Ausführliche Informationen: http://www.newsroom.ucla.edu/page. asp?RelNum=5356

Kontakt: UCLA, 405 Hilgard Avenue, Box 951361, Los Angeles, CA 90095-1361, Mr. Stuart Wolpert, eMail: stuartw@college.ucla.edu, Internet: http://www.ucla.edu.

Roboter mit Köpfchen

Intelligente Systeme von morgen sollen wie Menschen lernen und Probleme lösen können

Sankt Augustin. Roboter haben ein schlechtes Image: Sie gelten als eher beschränkt und unkreativ. Doch künftig sollen sie intelligent und flexibel werden. Im EU-Projekt MACS bringt ein internationales Forscherteam ihnen funktionsorientiertes Denken bei. Die Roboter von morgen sollen wie Menschen lernen und Probleme lösen können.

Wann ist eine Maschine intelligent? Wenn sie genau das tut, wofür sie gebaut wurde - beispielsweise Putzen oder Rasen mähen? Oder beginnt Intelligenz erst jenseits der vorprogrammierten Handlanger-Tätigkeiten? „Ein wichtiges Kriterium ist Flexibilität", sagt Dr. Erich Rome, Abteilungsleiter am Fraunhofer-Institut für Autonome Intelligente Systeme AIS und Leiter des EU-Projekts „Multi-sensory Autonomous Cognitive Systems Interacting with Dynamic Environments for Perceiving and Using Affordances", kurz MACS. „Für einen Menschen ist es selbstverständlich, flexibel auf äußere Umstände zu reagieren. Sie denken immer funktional, also am Ergebnis orientiert", erklärt Rome. „Roboter hingegen arbeiten bisher objektorientiert." Wenn eine Maschine darauf programmiert ist, einen Stuhl zu bringen, dann sucht sie ausschließlich nach einer Sitzgelegenheit mit vier Füßen und einer Lehne. Auf die Idee, dass auch ein leerer Getränkekasten den Anforderungen genügen könnte, kommt sie nicht.

"Um Maschinen funktionales Denken beizubringen, muss man ihnen die Gelegenheit geben, die Dinge zu begreifen wie kleine Kinder", ergänzt der Informatiker Rome. „Sie müssen ausprobieren und lernen können, was man mit einem Gegenstand alles machen kann." Der amerikanische Wahrnehmungspsychologe James Gibson hat dafür den Begriff „Affordanz" eingeführt: Er ist abgeleitet vom englischen Wort „afford" - zu deutsch „anbieten" oder „gewähren". Eine Affordanz ist eine Handlungsmöglichkeit, die die Umwelt einem Menschen bietet, und die er erkennen und nutzen kann. „Im EU-Projekt MACS bauen wir auf Gibsons Theorie auf, um neue Ansätze für intelligente und lernfähige Roboter zu entwickeln", so Rome. Der Fraunhofer-Forscher und sein Team arbeiten dabei eng mit Wissenschaftlern der schwedischen Linköpings Universitet, der Forschungsgesellschaft Joanneum Research und der Studiengesellschaft für Kybernetik in Österreich sowie der türkischen Middle East Technical University zusammen. Die EU unterstützt das Projekt mit 1,9 Millionen Euro.Drei Jahre haben die Forscher Zeit, um KURT 3D, einem flexibel einsetzbaren Roboter vom Fraunhofer-Institut in St. Augustin, das funktionale Denken beizubringen. Derzeit wird ein Übungsraum konzipiert - eine Art Spielzimmer, in dem KURT 3D mit seinem magnetischen Greifarm nach Herzenslust experimentieren kann. Er wird voraussichtlich ein ganzes Sortiment von Metallbausteinen bekommen, die sich anheben, stapeln oder verschieben lassen. Beispielsweise große und kleine Dosen, Rampen und Schwellen. Wenn der Roboter mit diesen Objekten „spielt", lernt seine Steuerungssoftware ständig dazu. Nach einiger Zeit soll KURT 3D seinen Aktionsradius selbstständig vergrößern. Dazu muss er allerdings erst einmal einiges über seine Umwelt herausfinden: beispielsweise dass sich eine Tür nur öffnet, wenn man mehrere schwere Gegenstände auf einen Schalter stellt; oder dass sich Schwellen mit Hilfe von Rampen überwinden lassen. „Im EU-Projekt werden wir noch an den Grundlagen arbeiten", erklärt Rome. „Unser Ziel ist es dabei, zu zeigen, dass funktionale Roboter wie Menschen lernen und Probleme lösen können." Langfristig werden die neuen, flexiblen Denkstrukturen nach Ansicht des Projektleiters die Roboterwelt verändern: „Die Grundlagenforschung von heute wird in die Entwicklung der nächsten und übernächsten Roboter-Generation mit einfließen. Bisher sind Objekt- und Funktionsorientierung separate Ansätze. Tatsächlich könnten sich die beiden Ansätze jedoch ergänzen. Je nach Situation kann es für einen Roboter sinnvoll sein, nach Handlungsalternativen zu suchen oder stur nach Plan vorzugehen."

Im EU-Projekt MACS soll der Roboter KURT 3D spielerisch seine Umgebung kennen lernen. (Foto: Dieter Klein)

Kontakt: Dr. Erich Rome, Tel. 02241-1426 83, eMail: erich.rome@ais.fraunhofer.de. Fraunhofer-Institut für Autonome Intelligente Systeme (AIS), Schloss Birlinghoven, D-53754 Sankt Augustin, Tel. 02241-14-27 56, -26 78, Fax: -14-23 84, Internet: www.ais.fraunhofer.de.

Allgemeine Informationen

(Stand Herbst 2004)

APL-Germany e.V. ist ein gemeinnütziger Verein mit Sitz in Düsseldorf. Sein Zweck ist es, die Programmiersprache APL, sowie die Verbreitung des Verständnisses der Mensch-Maschine Kommunikation zu fördern. Für Interessenten, die zum Gedankenaustausch den Kontakt zu anderen APL-Benutzern suchen, sowie für solche, die sich aktiv an der Weiterverbreitung der Sprache APL beteiligen wollen, bietet APL-Germany den adäquaten organisatorischen Rahmen.

Auf Antrag, über den der Vorstand entscheidet, kann jede natürliche oder juristische Person Mitglied werden. Organe des Vereins sind die mindestens einmal jährlich stattfindende Mitgliederversammlung sowie der jeweils auf zwei Jahre gewählte Vorstand.

Vorstand

Vorsitzender

Dieter Lattermann
Rheinstraße 23
69190 Walldorf
Tel. (06227) 63469
eMail: Dieter_Lattermann@t-online.de

Schriftführer

Conrad Hoesle-Kienzlen
CHK IT-Consulting
Augustenstr. 57
70178 Stuttgart
Tel. 0172 722 04 48
Fax 0711 674 38 53
eMail: conrad@hoesle.net

2. Vorsitzender

Martin Barghoorn
(Technische Universität Berlin)
Lückhoffstr. 8
14129 Berlin
Tel. (030) 314 24392
Fax: (030) 314 25901
eMail: martin@barghoorn.de
http://stat.cs.tu-berlin.de/~barg/

Schatzmeister

Jürgen Beckmann
Im Freudenheimer Grün 10
68259 Mannheim
eMail: jbecki@onlinehome.de

Beitragssätze

Ordentliche Mitglieder:

Natürliche Personen 31,00 Euro p.a.
Studenten / Schüler 10,00 Euro p.a.

Außerordentliche Mitglieder:

Juristische /
natürliche Personen 511,00 Euro p.a.

Bankverbindung:

BVB Volksbank eG Bad Vilbel
BLZ 518 613 25, Konto-Nr. 523 2694

Hinweis:

Wir bitten alle Mitglieder, uns *Adressenänderungen* und *neue Bankverbindungen* immer sofort mitzuteilen.

Geben Sie bei *Überweisungen* den *Namen* und/oder die *Mitgliedsnummer* an.

Einzugsermächtigung

Mitglieds-Nr.: _____

Ich erkläre mich hiermit widerruflich damit einverstanden, daß APL Germany e.V.

den jeweils gültigen Jahres-Mitgliedsbeitrag von meinem unten angegebenen Konto abbucht.

Einen eventuell bestehenden Dauerauftrag habe ich bei meiner Bank gelöscht.

Bankbezeichnung: _____

BLZ: _____ Konto-Nr.: _____

Datum: _____ Unterschrift: _____