4.

Syntaxanalyse und Compiler

Buch Mark Weiss „Data Structures & Problem Solving Using Java“ siehe Seiten: 294-297 (Stacks); 473-502 (Stacks und Compiler); 625-629 und 635-639 (Stack-Implementierung); 681 ff (Bäume); 697-708 (Baumtraversierung)

Aus „Informatik I“ teilweise bekannt:
Taschenrechner, rekursiver Parser, Evaluation eines Ausdrucksbaums

Lernziele Kapitel 4 Syntaxanalyse und Compiler

- Bäume als Konzept (Graph) und als Datenstruktur verstehen
- Baumtraversierung (postorder, inorder etc.) anwenden
- Operatorbäume auswerten
- Syntaxdiagramme (zum Generieren oder Parsen) anwenden
- Rekursiven Abstieg zur Syntaxanalyse anwenden
- Infix-Notation nach Postfix systematisch konvertieren
- Code für eine virtuelle Stackmaschine generieren
- Prinzip des Java-Bytecodes verstehen

Thema / Inhalt

Ein Kapitel, das es in sich hat: Mit der **Syntaxanalyse** besprechen wir einerseits eine der Hauptkomponenten eines **Compilers**, der Programme einer höheren Programmiersprache in bedeutungsgleichen „Instruktionscode" für eine Zielmaschine übersetzt, andererseits ist die automatische Analyse einer syntaktischen Struktur und ihre Zerlegung in Teilkomponenten auch

Thema / Inhalt (2)

sonst oft wichtig – beispielsweise dann, wenn wir in den Dialog mit Computern oder „smarten" Dingen treten.

Die Syntax einer Sprache, insbesondere auch die einer „formalen" Sprache, wird über eine Grammatik definiert. Eine einfache und eingängige Form der Notation einer Grammatik (und damit der syntaktischen Spezifikation einer Sprache) stellen **Syntaxdiagramme** dar. Indem man diese Diagramme in kanonischer Weise traversiert, kommt man an „Terminalsymbolen" vorbei, die man en passant fortwährend notiert – auf diese Weise generiert man schrittweise ein syntaktisch korrektes Sprachfragment, z.B. einen arithmetischen Ausdruck in Infix-Form oder gar ein ganzes Java-Programm. Syntaxdiagramme erlauben einem eine gewisse Freiheit beim Durchlaufen – man möchte ja Unterschiedliches generieren können. Damit mittels einer endlichen Grammatik (also einer kleinen Zahl von zusammengehörigen Syntaxdiagrammen) unendlich viele Sprachfragmente erzeugt werden können, sind Grammatiken im Allgemeinen rekursiv angelegt und enthalten die Syntaxdiagramme Schleifen.

Ein Compiler, oder genereller, ein **Syntaxanalysator**, steht nun vor dem Problem, ein Stück „Text", das etwa in Form einer Zeichenkette (auf Computerdeutsch: als „String" von „Characters") gegeben ist, daraufhin zu analysieren, ob es überhaupt mit der Grammatik gebildet werden kann (also syntaktisch korrekt ist) und wenn ja, aus welchen syntaktischen Teilkomponenten es besteht. Denn tatsächlich reflektiert sich die Bedeutung (d.h., die **Semantik**) im strukturellen Bezugsgefüge der einzelnen Teilkomponenten zueinander. So bedeutet beispielsweise der arithmetische Ausdruck ((8-5)-1) etwas anderes als (8-(5-1)); die drei Operanden stehen in unterschiedlicher Affinität zueinander. So gesehen muss ein Syntaxanalysator das Syntaxdiagramm „mit Spürsinn" so durchlaufen, dass dabei die zu analysierende Vorgabe generiert wird – die konkrete Art und Weise des Durchlaufens reflektiert sodann die Struktur

Thema / Inhalt (3)

des vorgegebenen Textes und damit dessen Semantik. Syntaxanalysatoren nennt man oft auch „**Parser**" – das lateinische Wort „pars"<partis> für „Teil" im Sinne von „aufteilen" ist hier namensgebend (wir erinnern uns: „Gallia est omnis divisa in partes tres...") und nimmt die Rolle der Silbe „lys" bei „Syntaxanalysator" ein, welche vom altgriechischen „lysis" (für „Auflösung") stammt.

Oft ist es so, dass eine zu analysierende Vorgabe syntaktisch **mehrdeutig** ist, dass sie also auf verschiedene Weise aus der Grammatik erzeugt werden kann. Bei zwei verschachtelten if-Anweisungen könnte man einen (im Prinzip optionalen) else-Teil beispielsweise dem inneren oder dem äusseren if zuordnen. Der Satz „Der Affe sagte die Ziege könne nicht Auto fahren" illustriert das Problem syntaktischer Ambivalenzen – um die Mehrdeutigkeit zu beherrschen und im Einzelfall zu kommunizieren, was gemeint ist, setzten wir beim Sprechen gezielt kurze Sprechpausen und eine unterschiedliche Betonung einzelner Wörter ein, im Schriftlichen müssen wir dafür Satzzeichen verwenden: Entweder schreiben wir „Der Affe sagte, die Ziege könne nicht Auto fahren" oder „Der Affe, sagte die Ziege, könne nicht Auto fahren". Ist die Kommasetzung in die Grammatik integriert, dann ist klar, was jeweils der Hauptsatz und dessen Subjekt ist und wer umgekehrt als Objekt der Anschuldigung fungiert.

Um die Syntaxanalyse zu automatisieren, kann man das Prinzip des „**rekursiven Abstiegs**" anwenden. Dabei programmiert man Syntaxdiagramme, und zwar so, dass jedes Teildiagramm systematisch durch eine Java-Methode realisiert wird, welche ein entsprechendes Konstrukt „versteht". Das Akzeptieren bei syntaktischer Korrektheit oder das „Verstanden haben durch geeignetes Durchlaufen des Diagramms" ist allerdings noch nicht unser Endziel; ein Compiler soll aus dem analysierten und verstandenen Programmstück nun auch noch bedeutungsgleichen Befehlscode für eine Zielmaschine generieren. Dazu kann man den Parser, der aus den

Thema / Inhalt (4)

Syntaxdiagrammen gewonnen wurde, so instrumentieren, dass an entscheidenden Stellen, bei denen ein Fragment „verstanden" worden ist, entsprechende Sequenzen des Zielcodes ausgegeben werden. Auf diese Weise gelingt uns ein Java-Programm, das einen beliebig komplexen arithmetischen Infix-Ausdruck in einen äquivalenten Postfixausdruck umwandelt und auf Wunsch sogar gleich „on the fly" auswertet. In analoger Weise kann man Zielcode für die **Java-VM**, die virtuelle Java-Maschine, erzeugen, die sogenannten **Bytecode**, der vom Java-Compiler generiert wird, ausführen kann und einen wesentlichen Teil der Java-Laufzeitumgebung verkörpert. Wir schauen uns in diesem Kapitel einige instruktive Bytecode-Programme an, die aus Java-Programmfragmenten erzeugt wurden.

Syntaktische Strukturen sind typischerweise in sich selbst rekursiv verschachtelt – ein komplexerer arithmetischer Ausdruck besteht beispielsweise aus einfacheren Teilausdrücken, die vermöge eines Operators miteinander verknüpft sind. Die zugrundeliegende Struktur dabei ist ein **Baum**. Bäume sind spezielle (u.a. zyklenfreie) Graphen, die in der Informatik an vielen Stellen eine Rolle spielen; oft drücken sie hierarchische Strukturen aus. Als rekursive Strukturen sind sie prädestiniert dafür, dass rekursive Algorithmen auf sie angesetzt werden – beispielsweise zum systematischen **Traversieren**, um nacheinander alle Knoten zu besuchen und diese dabei auszuwerten, zu manipulieren oder einfach nur auszugeben. Wir besprechen insbesondere die **Inorder**- und die **Postorder**-Traversierung, da diese eine Beziehung zu den Infix- bzw. Postfixausdrücken verkörpern. In späteren Kapiteln werden wir weitere interessante Anwendungen mit Bäumen kennenlernen, z.B. Suchbäume oder Heaps, die auch die Grundlage für effiziente Sortierverfahren darstellen.

Bäume, insbesondere die durch eine eindeutige Wurzel und der damit induzierten Hierarchie charakterisierten **Wurzelbäume**, lassen sich auf recht unterschiedliche Art darstellen bzw.

Thema / Inhalt (5)

visualisieren. Neben der häufig üblichen **Darstellung** als Graph ist manchmal die Repräsentation als verschachteltes Mengendiagramm adäquat, aber auch horizontal linearisierte Darstellungen in Klammernotation oder vertikal verlaufende Listen von Knoten (mit Einrückungen zur Markierung der Verschachtelungstiefe) können zweckmässig sein. Bei Binärbäumen gibt es ausserdem noch eine besonders praktische und effiziente Repräsentation als Array, in welchem die Knoten hintereinanderweg niveauweise abgespeichert werden und die Eltern-Kind-Beziehung durch Verdoppeln oder Halbieren eines Indexwertes verkörpert wird. Ein und derselbe Baum lässt sich also verschieden darstellen, und selbstverständlich können die unterschiedlichen Darstellungen systematisch ineinander umgewandelt werden. Die verschiedenen Darstellungen eines Baumes bezeichnen alle das selbe – den Baum an sich, der ein abstraktes mathematisches Gebilde bleibt. So wie wir ja auch die eine abstrakte Zahl „elf" verschieden notieren können: Als „11" im üblichen Dezimalsystem, als „1011" im Dualsystem oder als „XI" in römischen Ziffern – alles bedeutet das Gleiche. (Vorausgesetzt jedenfalls, man verwendet jeweils die „richtige" Interpretation – denn „XI" könnte, anders interpretiert, ja beispielsweise auch für einen chinesischen Namen stehen!)

In einem Kalkül manipuliert man **Zeichen** nach rein syntaktischen Regeln – und erwartet doch, dass dies auch semantisch korrekt ist und am Ende etwas sinnvolles herauskommt. Die Beziehung zwischen Syntax und Semantik ist daher spannend. Inwiefern Zeichen inhärent Bedeutung tragen und wie Zeichen als Elemente einer syntaktischen Sphäre in die Domäne der Semantik hinweisen, dazu finden sich einige weiterführende Anmerkungen im Bonusteil des Kapitels, darüber hinaus auch dazu, wie in etymologischer Hinsicht „Zeichen" mit „digital" verbunden ist. Aber auch im modernen Zeiten wandeln sich noch Begriffsinhalte, weil sich zum Beispiel die Wichtigkeit dessen, was mit Wörtern und Begriffen bezeichnet wird, durch

Thema / Inhalt (6)

technischen Fortschritt und den damit induzierten sozialen Wandel ändert. Wir hatten dies im ersten Kapitel schon beim Begriff „Algorithmus" gesehen; eine analoge Wandlung und Ausweitung geschieht aktuell mit den Begriffen „digital" bzw. „Digitalisierung" – auch dazu finden sich einige Hinweise auf den Bonus-Slides.

Diese Kapitel bietet auch sonst wieder reichlich Stoff für **historische Anmerkungen**. So stellte sich bald nach der Konstruktion der ersten programmierbaren elektromechanischen und elektronischen Rechenautomaten, also in den ersten Jahren nach dem zweiten Weltkrieg, heraus, dass das Programmieren in maschinennaher Form ausserordentlich fehleranfällig und zeitaufwändig ist, man suchte daher bald nach Möglichkeiten, die in der Wissenschaft weit verbreitete mathematische Notation algebraischer Formeln und die sich eingebürgerte Art der Notation numerischer Algorithmen, zu deren Berechnung ja schliesslich die Maschinen konstruiert wurden, in direkterer Weise als „Sprache" zu verwenden. F.L. Bauer von der TU München leistete hier Pionierarbeit, indem er zeigte, wie mittels Stacks arithmetische Ausdrücke umgewandelt und ausgewertet werden können. **Heinz Rutishauser** verfolgte an der ETH Zürich Anfang der 1950er-Jahre eine besonders interessante Idee: Die Verwendung eines Computers („programmgesteuerter Rechenautomat"), damit dieser konkrete Programme in Maschinenform („Rechenpläne") für sich selbst erzeugt, und zwar aus Problembeschreibungen, die in einer an der Sprache der Mathematik angelehnten algorithmischen Formelsprache verfasst sind – mit anderen Worten, er dachte an das, was später „Compiler" genannt wurde! Rutishauser war in den späten 1950er-Jahren einer der Wegbereiter der Programmiersprache ALGOL, die über nachfolgende Programmiersprachen (wie Simula und C++) als Zwischenglieder zum Urahn von Java wurde.

Thema / Inhalt (7)

Die Forschung an der ETH Zürich von Rutishauser und anderen zu Compilerkonzepten, höheren Programmiersprachen und numerischen Algorithmen bereits in den 1950er-Jahren, lange bevor sich die Informatik als eigene Disziplin etablieren konnte, kam nicht von ungefähr: Ab 1950 besass die ETH Zürich als erste Universität in Kontinentaleuropa einen Computer: Die **Z4** von **Konrad Zuse**, und diese Maschine wollte produktiv genutzt werden und gab daher Ansporn zur Entwicklung geeigneter Programmierkonzepte. Wir stellen daher auch die Z4, ihren Konstrukteur Konrad Zuse, den nachfolgend an der ETH selbst entwickelten elektronischen Rechner **ERMETH**, dessen Konstrukteur **Ambros Speiser** sowie andere Protagonisten vor und berichten von einigen Begebenheiten, die die damalige Pionierzeit der Computernutzung in instruktiver Weise beleuchten.

Aber es gab in der Nachkriegszeit ja nicht nur Rechenautomaten, sondern **Automaten** für vielfältige andere Zwecke – z.B. spielten ab den 1960er-Jahren bis zum Aufkommen von PCs in den 1980ern sogenannte Textautomaten eine wichtige Rolle – heute sind sie fast vergessen. Das generelle Automatenzeitalter wurde als verklärte Zukunftsvision populär und doch entstand das Wort „Automatisierung" überhaupt erst in den 1950er-Jahren und wirkt seit dem (und immer wieder neu) gleichzeitig im Sinne eines Faszinosums als auch eines Schreckgespenstes. Eine kurze Besinnung auf die Begriffe „Automat" und „Automatisierung" wirkt daher erhellend, man sollte dies aber eigentlich in einen grösseren Kontext einbetten, der den Bogen von „Mechanisierung" über „Kybernetisierung" bis „Digitalisierung" spannt – und damit einen relevanten Teil unserer Technik-, Industrie- und sogar Kulturgeschichte ausmacht! An dieser Stelle würde dies aber doch zu weit führen.

Enger zum Vorlesungsthema gehören hingegen Begriff und Tätigkeit des **Programmierens**. Dies ist älter als der Computer oder Rechenautomat, denn programmiert hatte man auch schon

Thema / Inhalt (8)

mechanische Webstühle zur Anfertigung komplexer Muster, und zwar über Streifen von Lochkarten. **Charles Babbage** hat sich von ihnen inspirieren lassen, seine (nie wirklich realisierte) „Analytical Engine" sollte auf diese Weise programmiert werden. Bei Babbage (in dessen Nachlass, aber auch bei Luigi Federico Menabrea und Ada Lovelace, die seine Ideen aufgegriffen und 1842 / 43 publiziert haben) finden sich konkrete Programme für die Analytical Engine, und zwar anwendungsbezogen formuliert und mit symbolischen Variablennamen, beispielsweise zur Auflösung eines linearen Gleichungssystems mit zwei Unbekannten. Nun blieb aber die Analytical Engine ein Papiertiger; die ersten tatsächlich gebauten programmierbaren Computer erschienen rund 100 Jahre später; sie wurden anfangs über Steckbretter, Schalterstellungen, Lochstreifen und Lochkarten programmiert.

Aber noch etwas gibt es zum Programmieren anzumerken: Computer und programmierbare Rechenautomaten entstanden ja nicht aus dem Nichts heraus, sondern weil schon vor dem zweiten Weltkrieg Industrie, Militär und Wissenschaft aufgrund steigender Anforderungen an Qualität und Genauigkeit der Produkte zunehmend Bedarf an immer grösseren und längeren Berechnungen hatte – etwa in Form von Fourier-Transformationen zur Bestimmung von Eigenschwingungen von Maschinen, zur Tabellierung ballistische Flugbahnen unter komplizierten atmosphärischen Bedingungen, zum Ermitteln des Einflusses immer zahlreicherer und kleinerer Himmelskörper aufeinander etc. Auch vor dem Zeitalter programmierbarer Rechenautomaten wurden entsprechende Berechnungen, langwierig und aufwändig, durchgeführt, und zwar manchmal fast fabrikartig organisiert von einigen zig **menschlichen Rechnern und Rechnerinnen**. (Im Englischen hiessen diese Personen tatsächlich „Computer", das Wort wurde später einfach auf die programmierbaren Maschinen übertragen, die die gleiche Tätigkeiten, aber meist schneller und vor allem praktisch fehlerfrei, ausführten.)

Thema / Inhalt (9)

Die menschlichen Rechner arbeiteten meist unter Nutzung mechanischer oder elektromechanischer Tischrechenmaschinen für die vier Grundrechenarten repetitiv an einem kleinen Ausschnitt des Gesamtproblems. Es galt also schon damals, die Gesamtaufgabe algorithmisch in Teilaufgaben herunterzubrechen und den Datenfluss zu organisieren. Man musste den menschlichen Rechnern eindeutig vorschreiben, was sie zu tun hatten – gewissermassen musste man also die menschlichen Rechner und deren Zusammenspiel programmieren. Dies geschah etwa mittels **Rechenschablonen** aus Karton, die über Datenblätter gelegt wurden und schriftliche Anweisungen enthielten, wie mit den Daten umzugehen sei, die über die Fenster einer Schablone sichtbar gemachten wurden: Wie sie rechnerisch zu verknüpfen waren und in welches Fenster der Schablone das Ergebnis auf dem darunterliegenden Datenblatt einzutragen wäre. Dies entsprach einer heutigen Programmanweisung etwa der Art „addiere den Wert von Speicherzelle x zum Wert von Speicherzelle y, multipliziere das Ergebnis mit dem Wert von Speicherzelle z und notiere das Ergebnis in Speicherzelle a. Falls das Ergebnis positiv ist, fahre mit Schablone B17 fort, sonst mit Schablone A5". Insofern sollte es nicht verwundern, wenn die Funktionsweise der ersten programmierbaren Rechenautomaten (also „Computer" im heutigen Sinne) unter Rückgriff auf das seinerzeit Bekannte (nämlich den Ablauf bei menschlichen Rechnern in den Rechenfabriken) erläutert wurde als eine Substitution von Rechner (bzw. meist Rechnerin), Tischrechenmaschine, Rechenschablone und Rechenblatt durch ein einziges automatisch arbeitendes Gerät mit dem kombinierten Leistungsvermögen.

Dass man überhaupt effizient Probleme berechnen kann, die über das hinausgehen, was die Kaufleute bis ins Mittelalter mit dem römischen Zahlsystem, Abakus, Rechenmünzen und Rechentischen erreichen konnte, verdanken wir einem kulturellen Meilenstein: der Einführung der Stellenschreibweise im **indisch-arabischen Ziffernsystem** zusammen mit der Erfindung der Null. Dieses fortschrittliche Rechensystem kam über den persisch-arabischen Raum zu-

Thema / Inhalt (10)

nächst in das südliche Europa, den Sprung über die Alpen schaffte es erst etwas später. Aber Rechenmeister wie Adam Ries Anfang des 16. Jahrhunderts und die Möglichkeit, Rechenbücher in der Sprache des Volkes in grösserer Auflage per Buchdruck herzustellen, verhalfen dem „schriftlichen Rechnen" (dem damals sogenannten „Rechnen auf der Feder") im Dezimalsystem schliesslich überall zum Durchbruch.

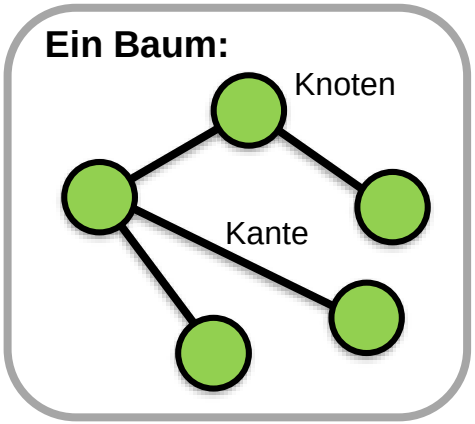
Für die sich entwickelnden Wissenschaften, zunächst für die Astronomie, dann aber auch die Physik sowie weitere Naturwissenschaften und schliesslich für die Ingenieurswissenschaften, war ein solches einfach handhabbares und praktisch anwendbare Rechensystem, das auch für sehr grosse Zahlen anwendbar war, von entscheidender Bedeutung. Als das Rechnen den Menschen zu viel wurde, konnte man es so relativ einfach auf Maschinen auslagern – zunächst, langsam beginnend im 17. Jahrhundert mit Pascal und Leibniz, waren dies handbediente mechanische (später dann elektrisch angetriebene) Maschinen für einzelne Rechenoperationen, Mitte des 20. Jahrhunderts dann programmgesteuerte Rechenautomaten für algorithmisch beschriebene repetitive Rechnungen. Der Computer, ein **automatisierter Rechenknecht**, war erfunden!

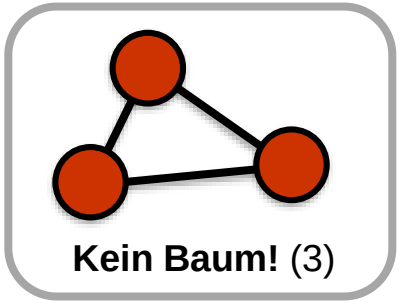
Dass elektronische Computer bald darauf winzig klein, ganz preiswert und irre schnell werden würden; dass man bald nicht nur mit Zahlen an sich rechnen würde, sondern mit allem Denkbaren, das sich in Zahlen ausdrücken lässt, auch mit Messwerten und Eindrücken aus der physischen Welt; und dass Bitfolgen mehr als Zahlen darstellen können und trotzdem ganz analog wie beim eigentlichen Rechnen „digital" verarbeitet werden können – das alles hat man sich vor wenigen Jahrzehnten allerdings noch nicht vorstellen können. Damals, Mitte des letzten Jahrhunderts, diente ein Computer nur zum Rechnen, dafür war er geschaffen und diese Zweckbestimmung drückte sein Name auch aus.

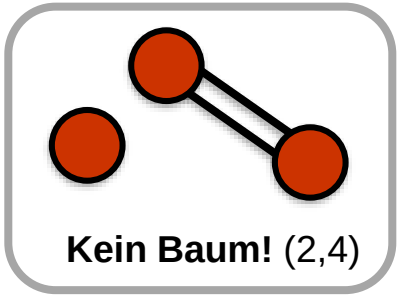
Bäume in der Informatik

Bestehen aus Knoten und Kanten (d.h., sind „Graphen“)

(1) Jede Kante verbindet genau 2 Knoten
(2) Zwischen je 2 Knoten gibt es höchstens eine Kante
(3) Anzahl der Knoten = 1 + Anzahl der Kanten
(4) Sind „zusammenhängend“: von jedem Knoten kann man (evtl. indirekt) jeden anderen (über einen „Weg“) erreichen




Kein Baum! (3)


Kein Baum! (2,4)

Für jeden Baum gilt:

- Zwischen je zwei verschiedenen Knoten gibt es genau einen Weg (Folge von „benachbarten“ Knoten bzw. Kanten)
- Es gibt keine „Zyklen“ (Weg mit Anfangsknoten = Endknoten)

Stichwort Graph – ein kurzer Exkurs

Aus vielen Beispielen sind Graphen zumindest in informeller Hinsicht bereits bekannt. Im mathematischen Sinne besteht ein Graph aus einer Knotenmenge, einer dazu disjunkten Kantenmenge und einer Abbildung, die jeder Kante ein Paar von (durch sie verbundene) Knoten zuordnet – solche Knoten heissen benachbart.

Da (bei $a \neq b$) Knotenpaar (a, b) von (b, a) verschieden ist, sind Kanten grundsätzlich gerichtet, was in der graphischen Darstellung durch einen Pfeil ausgedrückt wird. Für (a, b) fungiert dabei a als Anfangspunkt und b als Endpunkt der Kante. Bei ungerichteten Graphen „identifiziert" man alle jeweiligen Knotenpaare (a, b) und (b, a) und zeichnet Kanten als Linien ohne Pfeil. Ein gerichteter Graph heisst auch Digraph („directed graph").

Kanten, die die gleichen Knotenpaare verbinden, heissen Mehrfachkanten. Wird ein und der selbe Knoten durch eine Kante verbunden, nennt man diese eine Schlinge. Mehrfachkanten und Schlingen werden oft stillschweigend ausgeschlossen. (Hebt man dies hervor, nennt man solche Graphen schlicht.)

In der Informatik sind Knoten- und Kantenmenge i.a. endlich.

Bäume, oben mit Eigenschaften (3), (4) eingeführt, sind spezielle Graphen.

Stichwort Graph (2)

Reiht man Kanten so aneinander, dass der Endpunkt einer Kante als Anfangspunkt einer weiteren Kante fungiert, dann erhält man einen Weg als Knotenfolge. Ein mehrfaches Auftreten von Knoten lässt man dabei i.a. nicht zu; wenn der erste Knoten identisch zum letzten Knoten ist, spricht man allerdings von einem Zyklus (oder Kreis). Gerichtete Wege werden auch als Pfade bezeichnet.

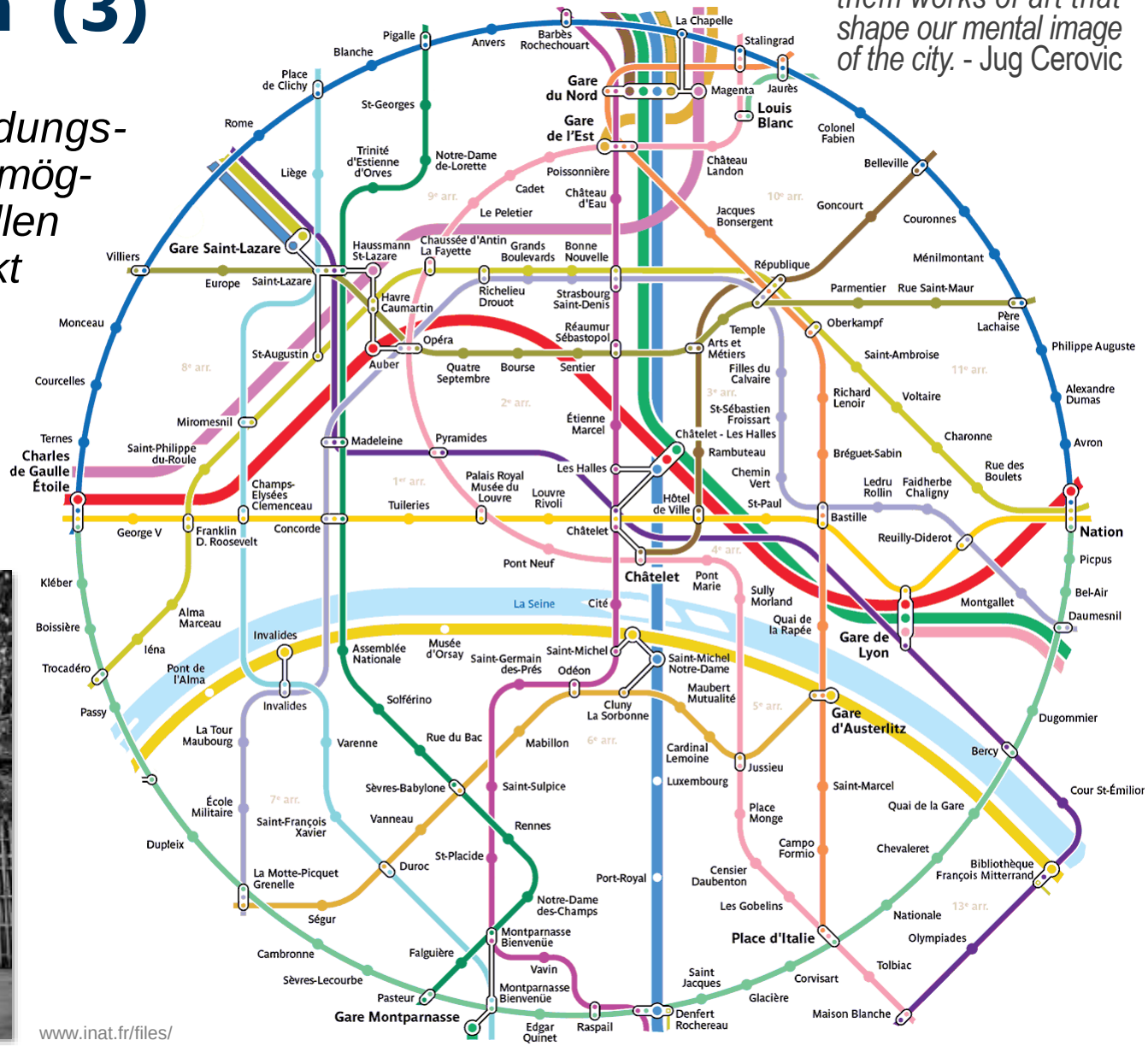
A priori muss es nicht von jedem Knoten zu jedem anderen einen Weg geben – ist das aber der Fall, dann heisst der Graph zusammenhängend. Bäume sind minimal zusammenhängend (und damit als Struktur „verletzlich"): Entfernt man eine beliebige Kante, ist der Graph nicht mehr zusammenhängend. Meist (allerdings nicht bei Bäumen!) gibt es in Graphen mehrere Wege zwischen zwei bestimmten Knoten – dann sucht man im Sinne der dadurch modellierten Anwendung oft den kürzesten Weg (oder den „billigsten", wenn die Kanten mit Kosten markiert sind).

Graphen stellen zweckdienliche Modelle für vielfältige Anwendungen dar: Verkehrsnetze, Kommunikationsnetze (z.B. Routing im Internet), soziale Beziehungen, Firmengeflechte (wer ist an wem zu wieviel Prozent indirekt beteiligt?), Netzpläne (kritischer Pfad von Aktionen), Produktionspläne, biochemische Stoffwechselnetzwerke etc.

Stichwort **Graph (3)**

Ein typisches Anwendungs-beispiel: Suche eines mög-lichst kurzen & schnellen Weges vom Startpunkt zu einem Zielpunkt in einem U-Bahnnetz. Hier: Metro in Paris (map design: Jug Cerovic, Paris).

Metro maps are much more than mere functional diagrams to me, I consider them works of art that shape our mental image of the city. - Jug Cerovic



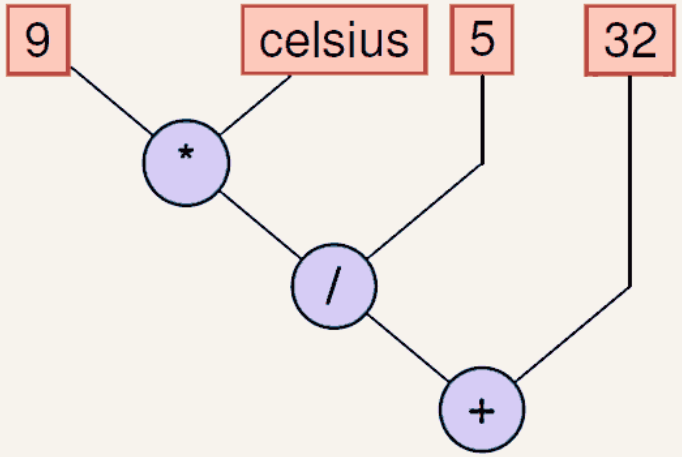
www.inat.fr/files/

Bäume – schon bekannt aus „Informatik I“, z.B.:

Ausdrucksbäume

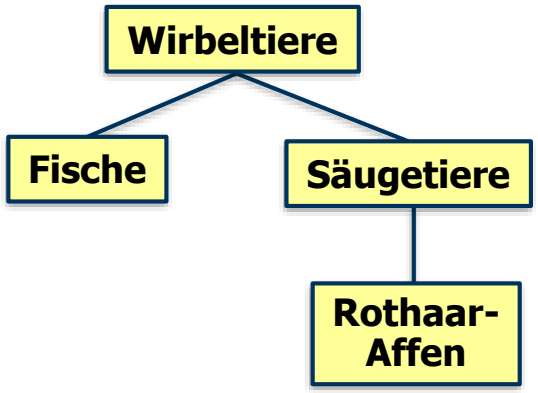
Klammerung ergibt Ausdrucksbaum

```
(((9 * celsius) / 5) + 32)
```

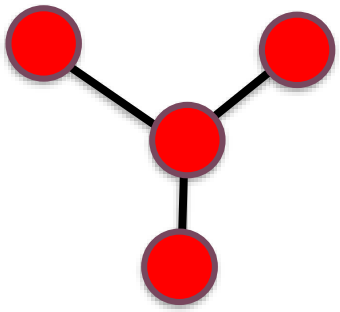


Moin, moin, bitte einen Baum mit 4 Knoten!

Bitteschön!



Dankeschön! Und noch einen anderen, bitte!



Wie wäre es damit?

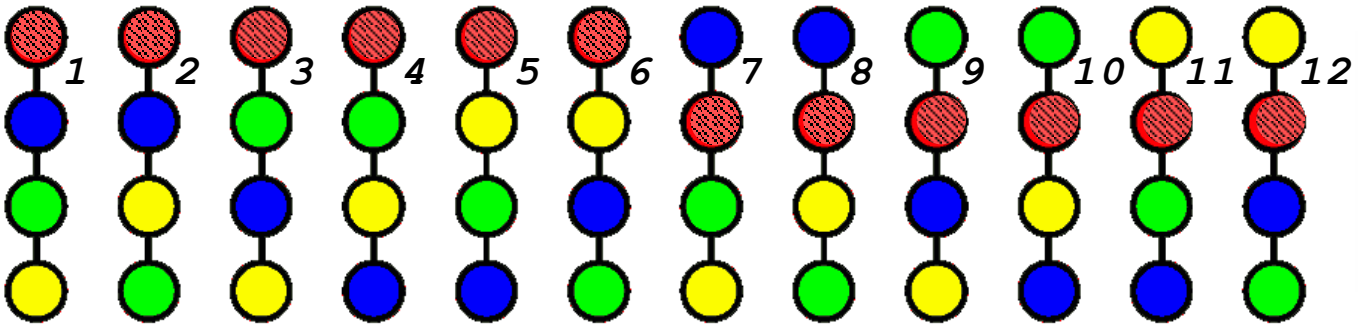
Und noch einen!

→ Ja, aber wie viele ~~wirklich~~ verschiedene 4er-Bäume gibt es?

In welchem Sinne?

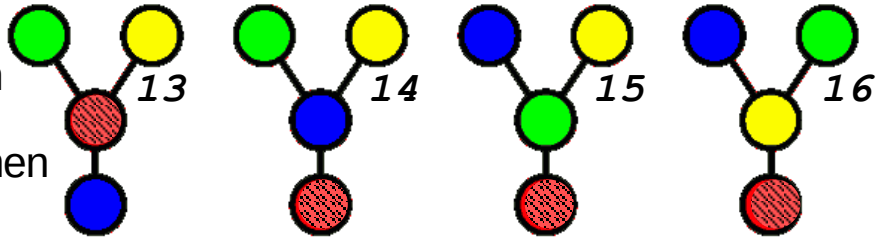
verschiedenen

Beispiel: Alle 16 verschiedenen Bäume mit 4 Knoten



Und was ist damit?

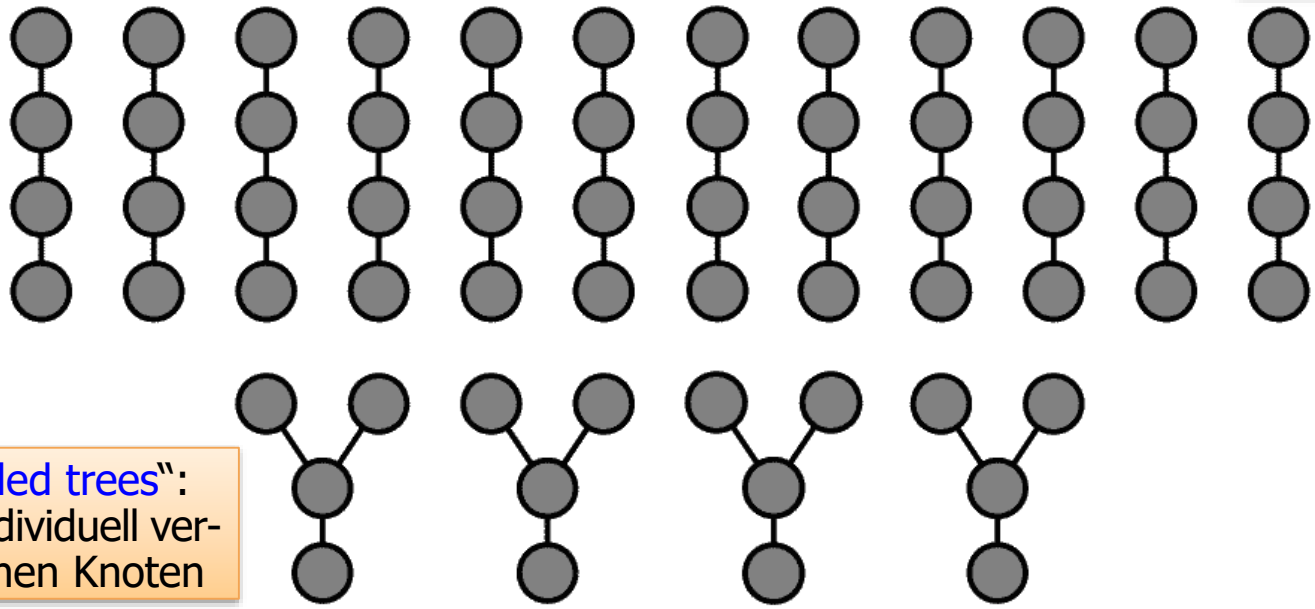
Bei solchen sogen. labeled trees werden alle jew. Knoten als verschieden angesehen



Denkübung: Man überlege sich, wieso es n^{n-2} solcher Bäume gibt (mit n = Zahl der Knoten) → *Formel von Cayley,* 1889

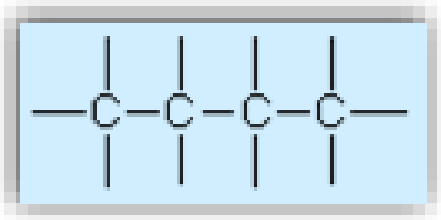
Beispiel: Alle 2 verschiedenen Bäume mit 4 Knoten

ununterscheidbaren

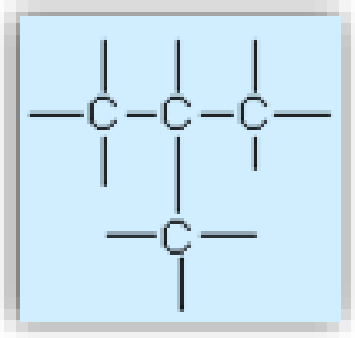


„Unlabeled trees“: Keine individuell verschiedenen Knoten

Analog gibt es auch nur 2 Isomere des Camping-Gases Butan C_4H_{10}:



N-Butan

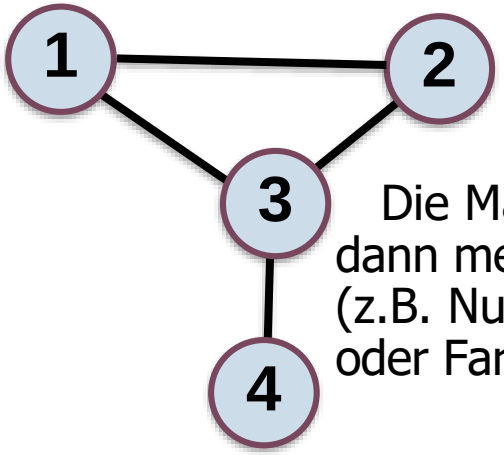


Isobutan

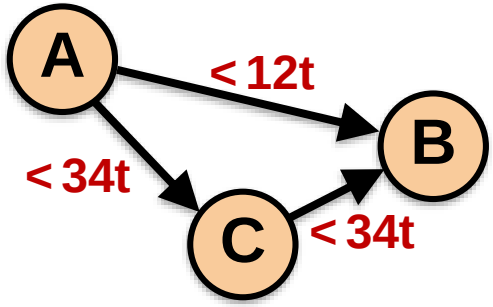
Feuerzeug mit Butan-Füllung

Markierte / unmarkierte Graphen

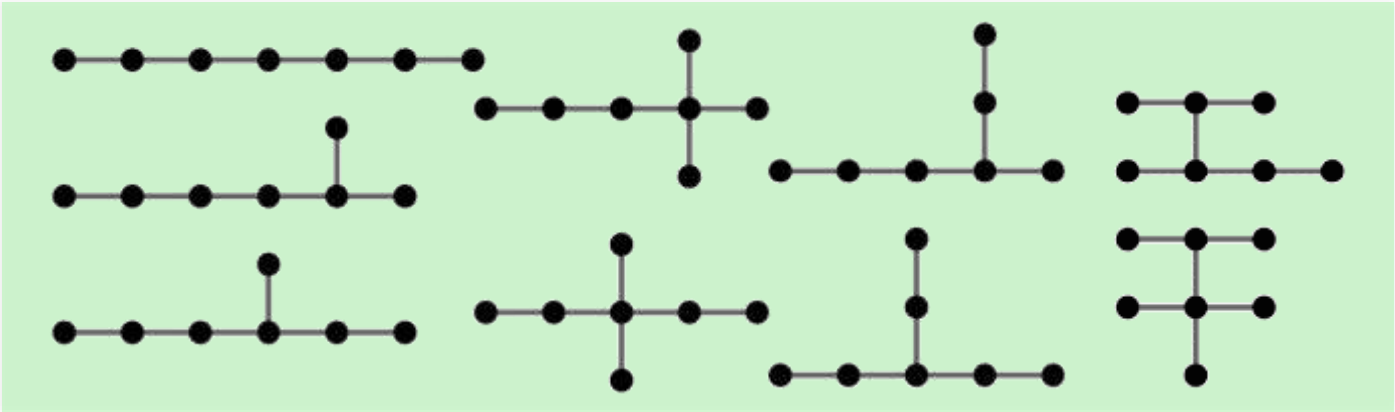
Knoten (oder auch Kanten) eines Graphen können (z.B. zur jeweiligen Unterscheidung) eine Markierung („label") haben



Die Markierung ist dann meist eindeutig (z.B. Nummerierung oder Farbe wie oben)



Beispiel: Kanten mit unterschiedlichen Eigenschaften

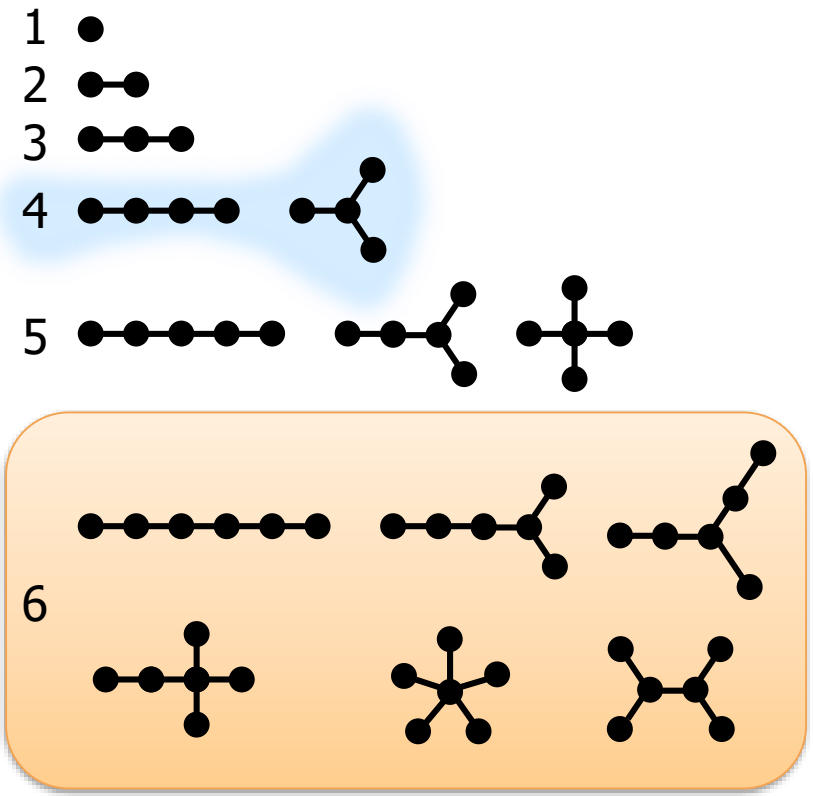


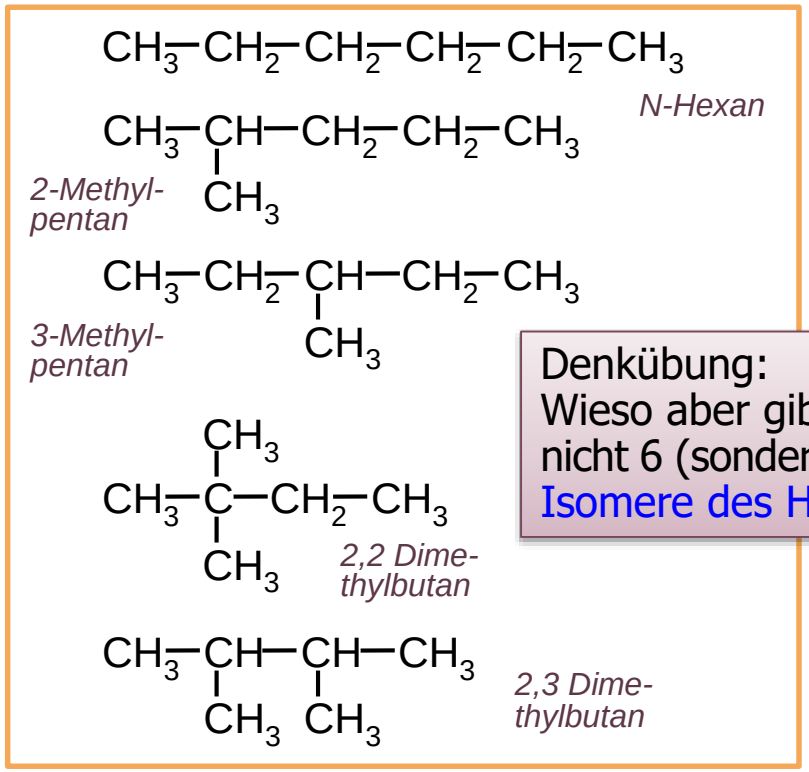
Einige unmarkierte Bäume mit je 7 Knoten

(Denkübung: Einige fehlen – welche?)

unmarkierte

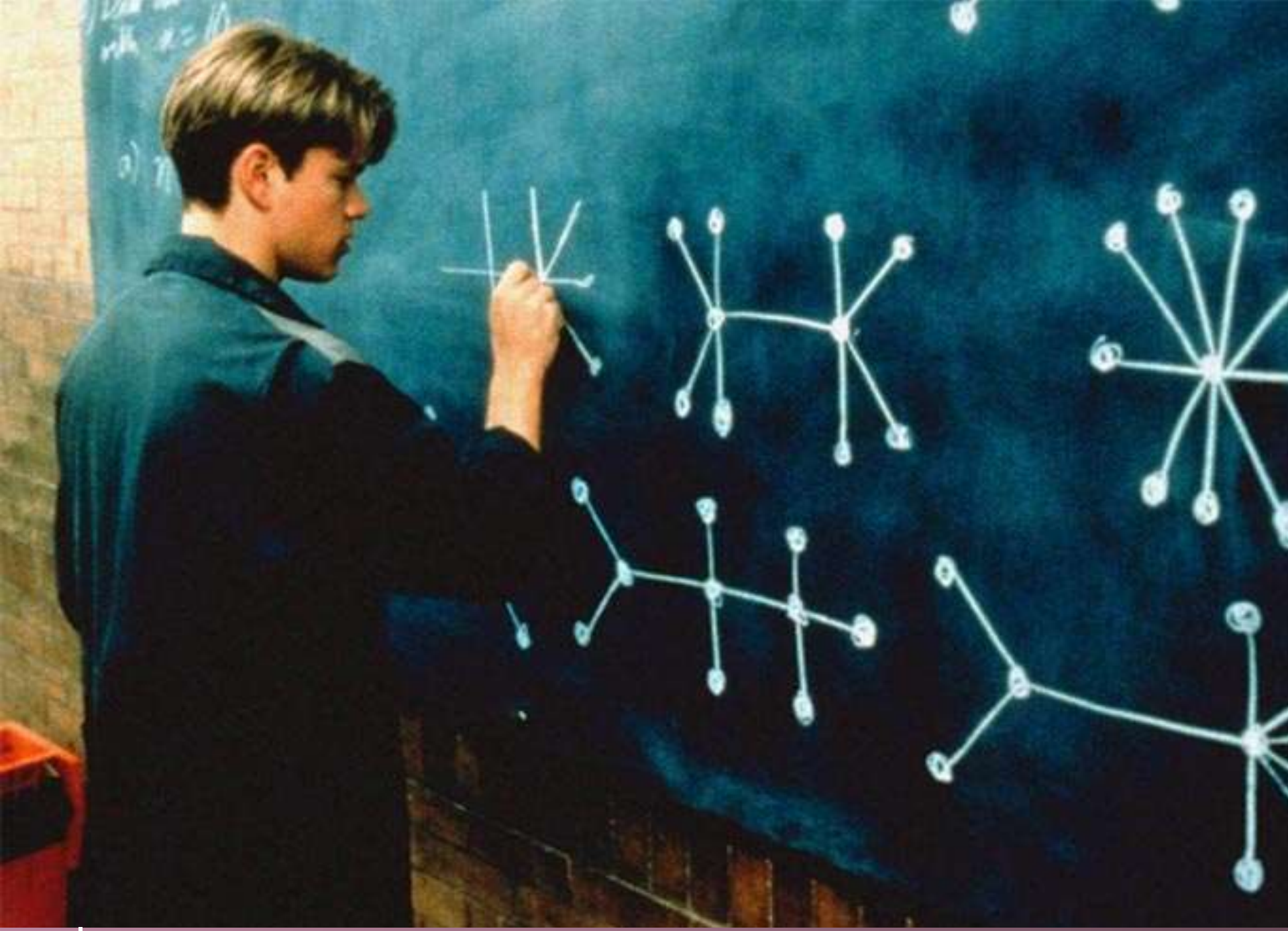
Beispiel: Verschiedene Bäume der Grösse 1 bis 6





Denkübung:
Wieso aber gibt es nicht 6 (sondern nur 5) Isomere des Hexans?

- Die Anzahl der Bäume bei n Knoten entspricht der Folge 1, 1, 1, 2, 3, 6, 11, 23, 47, 106, 235, 551, 1301, 3159, 7741, 19320, 48629, 123867,... (wächst grössenordnungsmässig exponentiell mit n)
- Hingegen die Anzahl der Isomere: 1, 1, 1, 2, 3, 5, 9, 18, 35, 75, 159, 355, 803,...

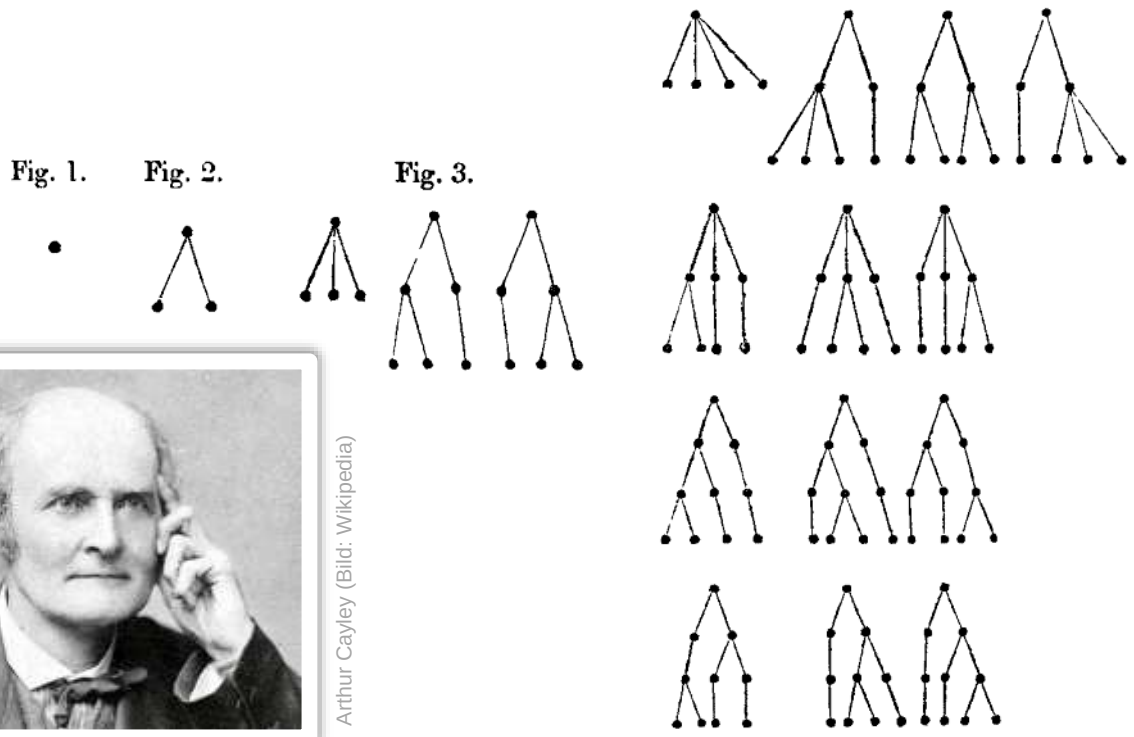


Cayley: Analytical forms called trees

Historische Notiz

THE following class of "trees" presented itself to me in some researches relating to functional symbols; viz., attending only to the terminal knots, the trees with one knot, two knots, three knots, and four knots respectively are shown in the figures 1, 2, 3, and 4:

Fig. 1. Fig. 2. Fig. 3. Fig. 4.



and similarly for any number of knots. The trees with four knots are formed first from those of one knot by attaching thereto in every possible way (one way only) four knotted branches; secondly, from those with two knots by attaching

Arthur Cayley (Bild: Wikipedia)

Bäume als mathematische Strukturen wurden 1857 von Arthur Cayley (1821 – 1895) eingeführt.

Links: Auszug aus: Arthur Cayley: *On the Theory of Analytical Forms called Trees. 2nd Part*. Philosophical Magazine, Vol. 17, 374-378, 1859.

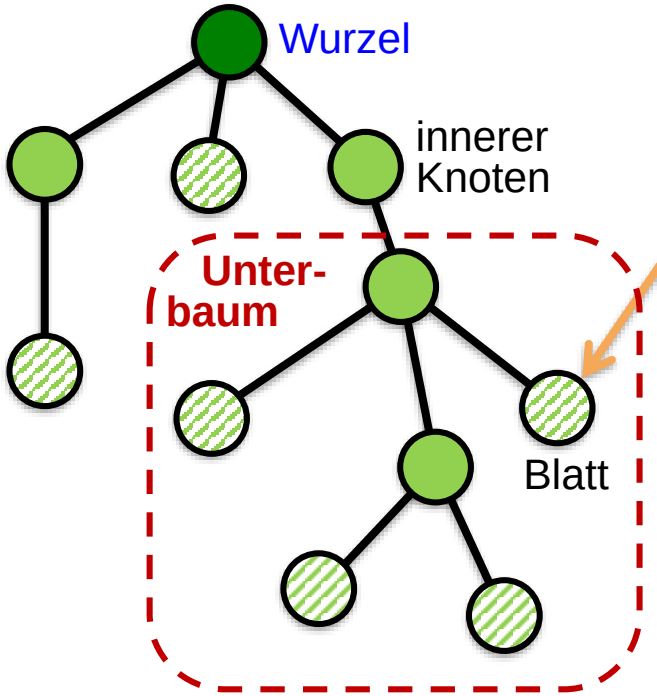
(Mit „knots" sind hier nach heutiger Sprechweise Blätter eines Wurzelbaums gemeint.)

Im Deutschen führte der Mathematiker Wilhelm Ahrens 1901 die Bezeichnung „Baum" (zuvor: „baumförmiger Typus") ein.

Wurzelbäume

Wurzelbäume werden typischerweise zur Darstellung hierarchischer Strukturen verwendet

- Ein bestimmter Knoten wird als „Wurzel“ ausgezeichnet
  - Beachte: Bäume sind in der Informatik meist Wurzelbäume; daher meint man oft „Wurzelbaum“, wenn man einfach „Baum“ sagt
- Werden i.Allg. „umgekehrt“ gezeichnet (Wurzel oben!)



Knoten mit nur einer einzigen „inzidenten“ Kante heissen Blatt

(Eine solche Wurzel sieht man i.a. nicht als Blatt an)

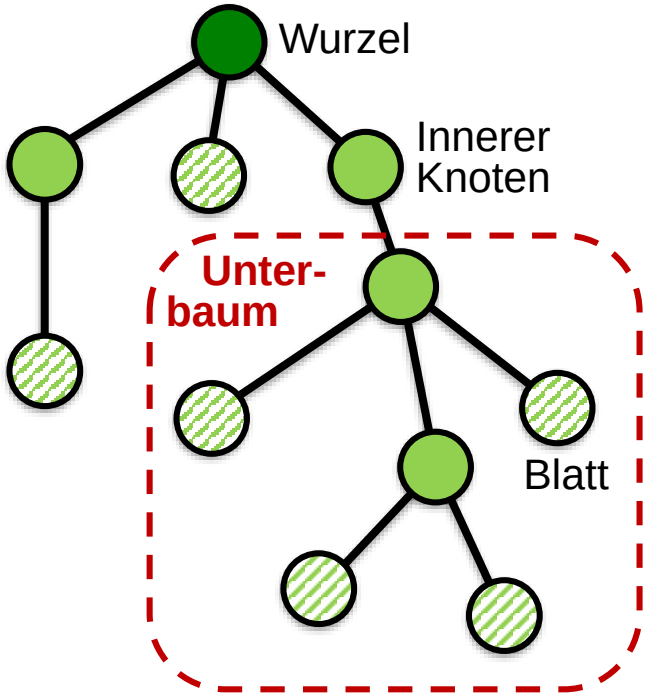
Jeden Knoten eines Baums kann man als Wurzel eines Unterbaums auffassen

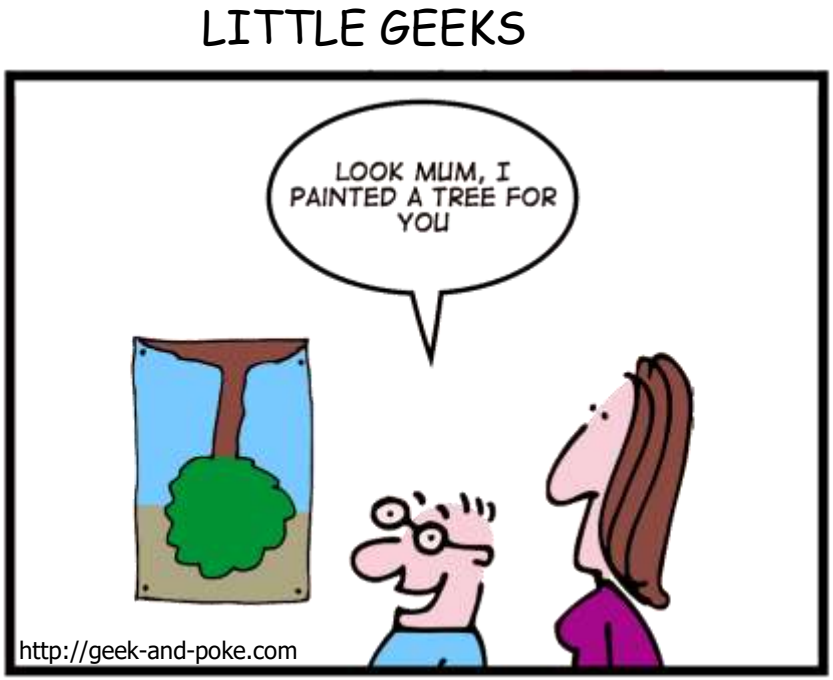
Ausgehend von der Wurzel kann man die Knoten in Ebenen (*Niveau*, *Level*) einteilen → gleiche Entfernung von der Wurzel

Wurzelbäume

Wurzelbäume werden typischerweise zur Darstellung hierarchischer Strukturen verwendet

- Ein bestimmter Knoten wird als „Wurzel“ ausgezeichnet
  - Beachte: Bäume sind in der Informatik meist Wurzelbäume; daher meint man oft „Wurzelbaum“, wenn man einfach „Baum“ sagt
- Werden i.Allg. „umgekehrt“ gezeichnet (Wurzel oben!)

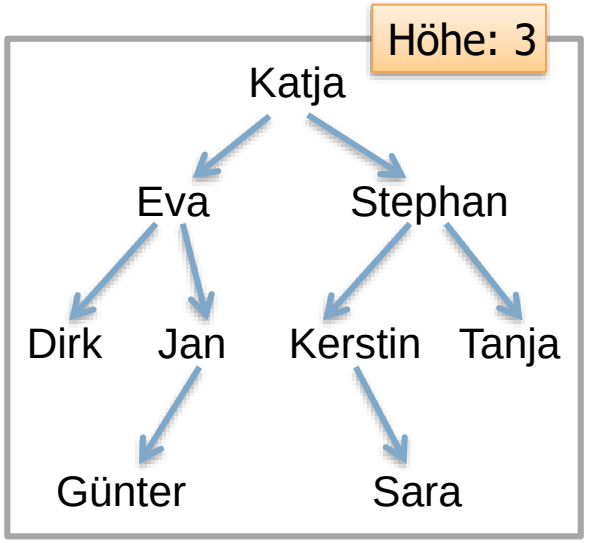




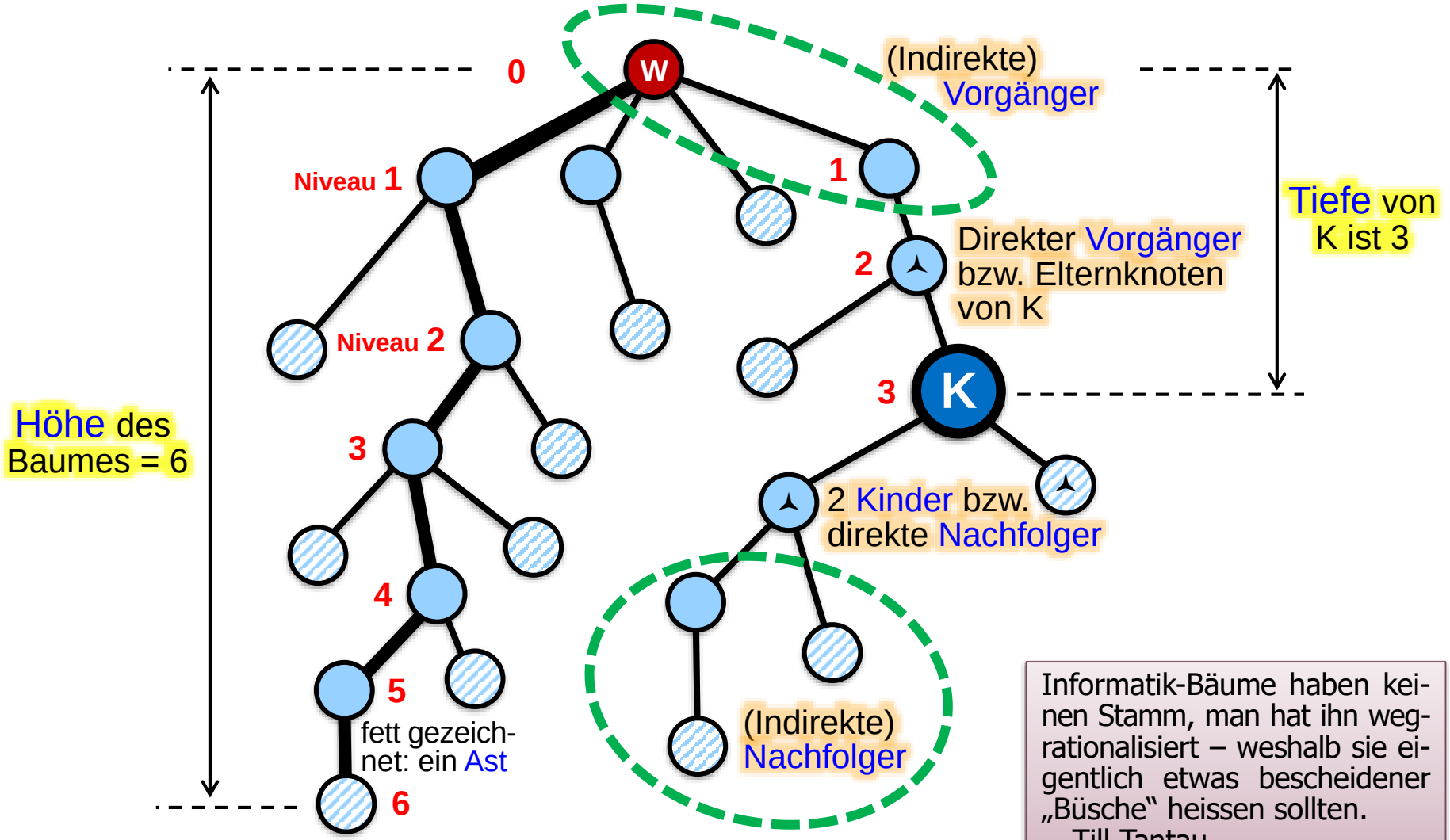
Wurzelbäume : Höhe und Tiefe

- Die Tiefe eines Knotens ist sein Abstand (d.h. die Länge seines Weges) zur Wurzel
  - Die Wurzel hat also die Tiefe 0

- Die Höhe eines Wurzelbaumes ist die maximale Knotentiefe
  - D.h. die Länge eines längsten Weges von der Wurzel zu einem Blatt (es kann mehrere solche – gleich langen – Wege geben!)
  - Mit anderen Worten: Der maximale Abstand zwischen einem Knoten und der Wurzel
  - Ein Baum, der nur aus einer Wurzel besteht, hat also die Höhe 0



Wurzelbäume

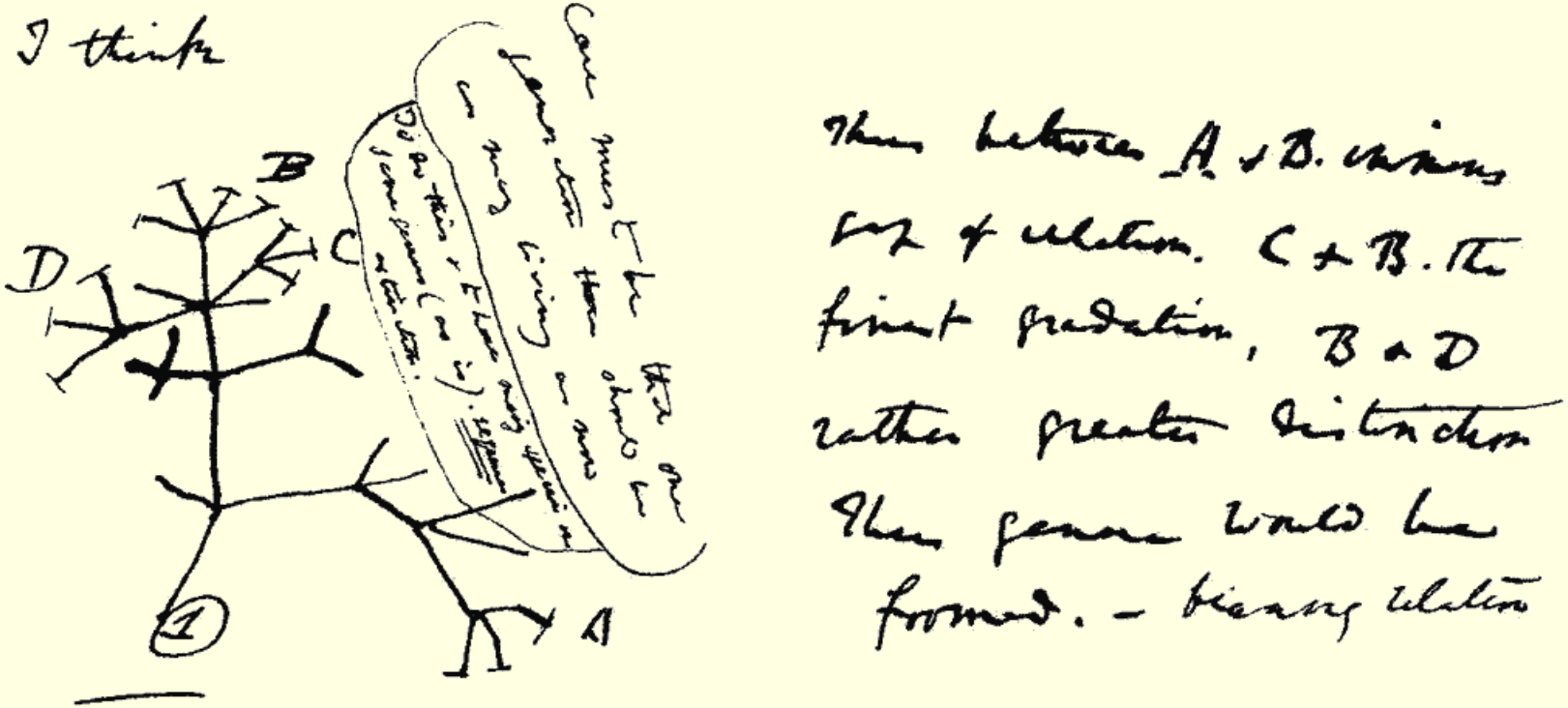


Informatik-Bäume haben keinen Stamm, man hat ihn wegrationalisiert – weshalb sie eigentlich etwas bescheidener „Büsche" heissen sollten.
-- Till Tantau

40+ Beispiele für (Wurzel)bäume...

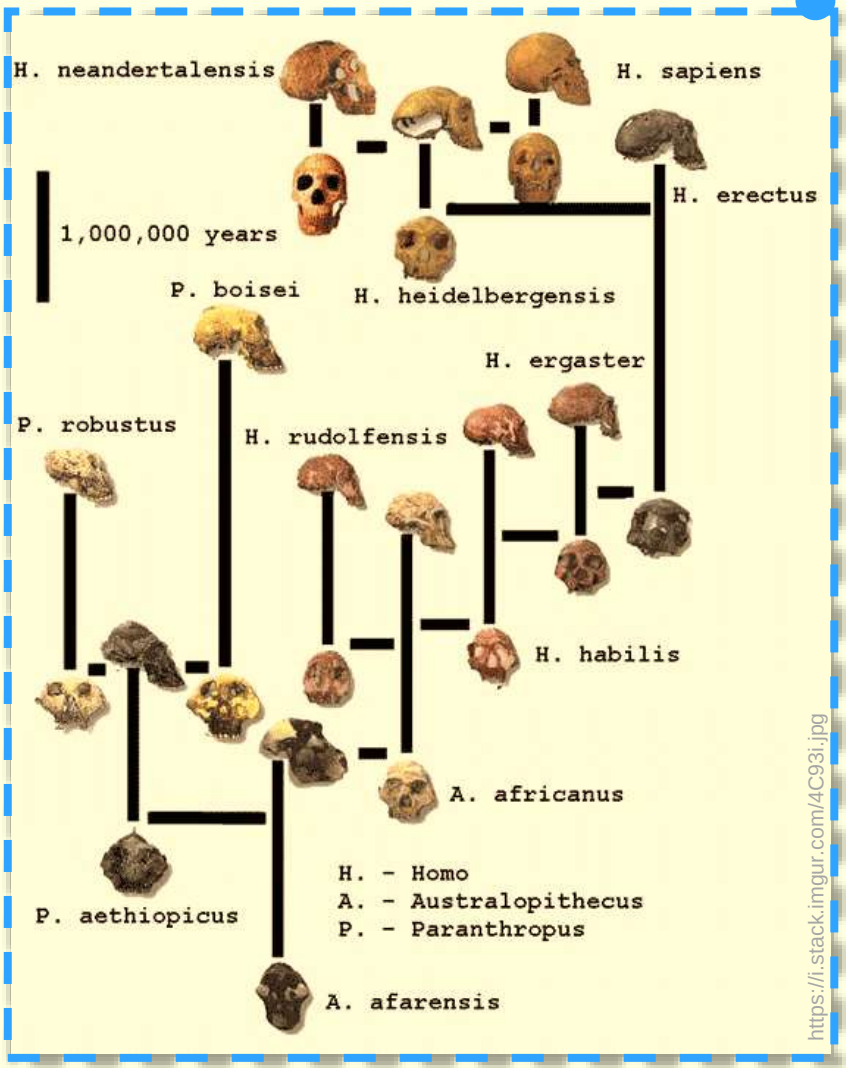
...und solche, die es fast sind

1) Charles Darwin: phylogenetischer Baum

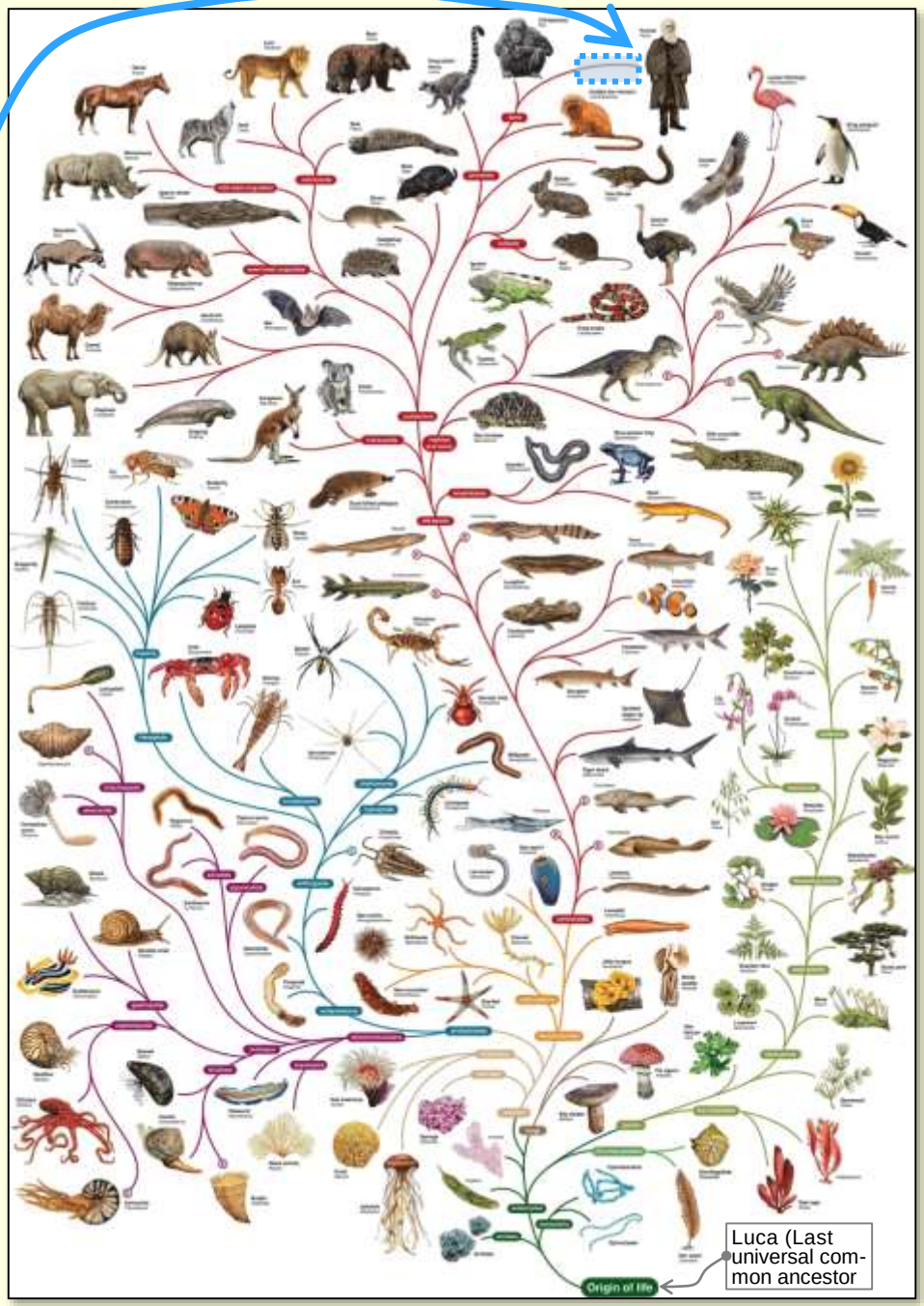


1837, die Geburtsstunde der Evolutionstheorie – der erste Entwurf eines phylogenetischen Baums durch Charles Darwin. “I think case must be that one generation should have as many living as now. To do this and to have as many species in same genus (as is) requires extinction. Thus between A + B the immense gap of relation. C + B the finest gradation. B+D rather greater distinction. Thus genera would be formed…”

2) „Tree of Life“



https://i.stack.imgur.com/4C93i.jpg



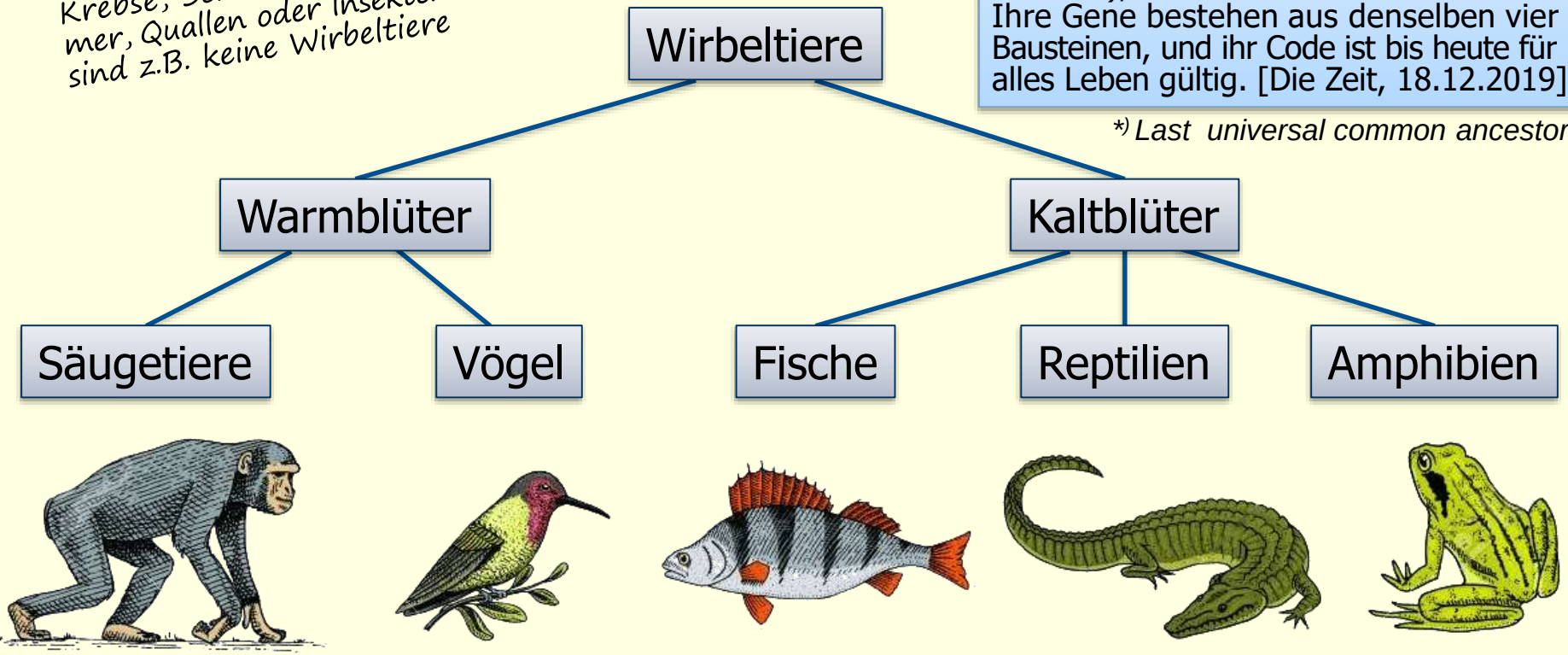
https://nicolasmicheletti.files.wordpress.com/2015/09/treeoflife.jpg

3) Klassifikation im Tierreich

Natürlich hat niemand Luca* je gesehen. Aber es muss dieses Wesen gegeben haben. Das folgern Forschende aus Gemeinsamkeiten aller heutigen Zellen – sie sind durch Zufall nicht zu erklären: Alle Lebewesen benutzen dieselben 20 Aminosäuren (und zwar immer nur deren linksdrehende Variante), um ihre Eiweisse aufzubauen. Ihre Gene bestehen aus denselben vier Bausteinen, und ihr Code ist bis heute für alles Leben gültig. [Die Zeit, 18.12.2019]

*) *Last universal common ancestor*

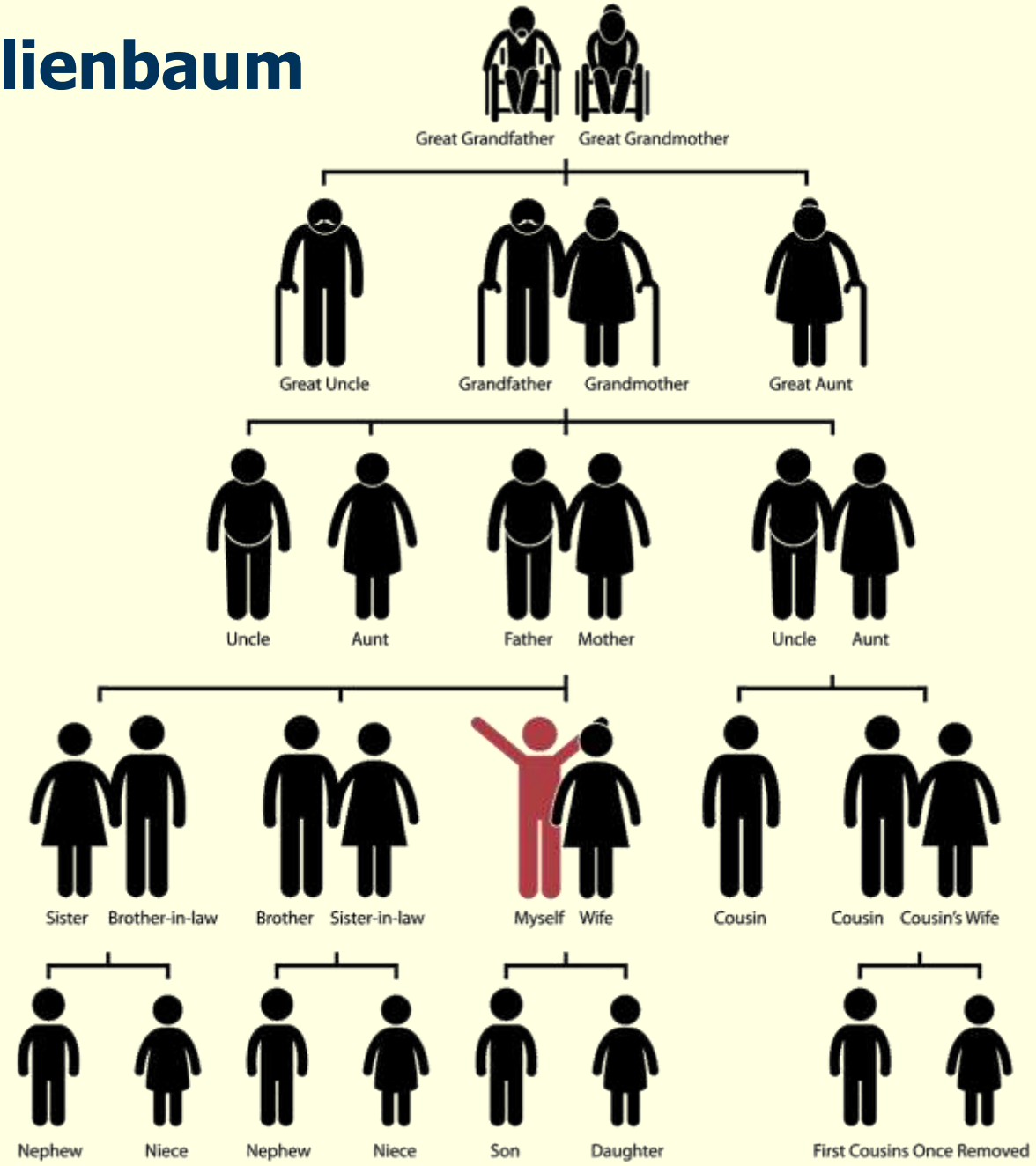
Krebse, Schnecken, Würmer, Quallen oder Insekten sind z.B. keine Wirbeltiere



Aber sind die Vögel und Reptilien nicht näher miteinander „verwandt“ und müssten daher einer gemeinsamen Klasse („Sauropsida“) untergeordnet werden? Und sollte man die Fische nicht von den anderen (Landwirbeltiere bzw. „Tetrapoda“) abtrennen?

Wann ist eine Klassifikation „richtiger“ (oder zweckmässiger) als eine andere? Bei der objektorientierten Programmierung ist letzteres jedenfalls eine wichtige Frage!

4) Familienbaum

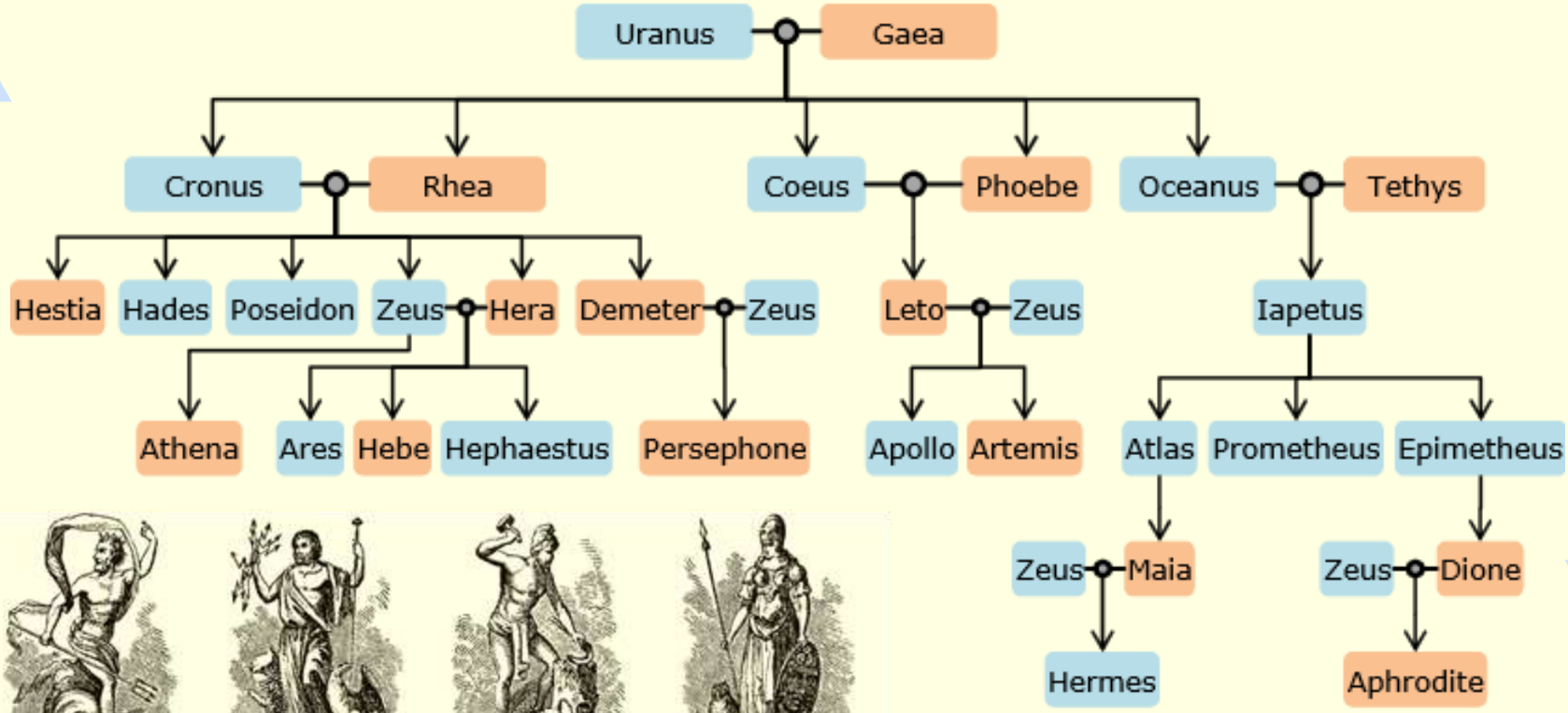


http://clipart-library.com/img1/1471986.jpg

5) Stammbaum griechischer Götter (Ausschnitt)

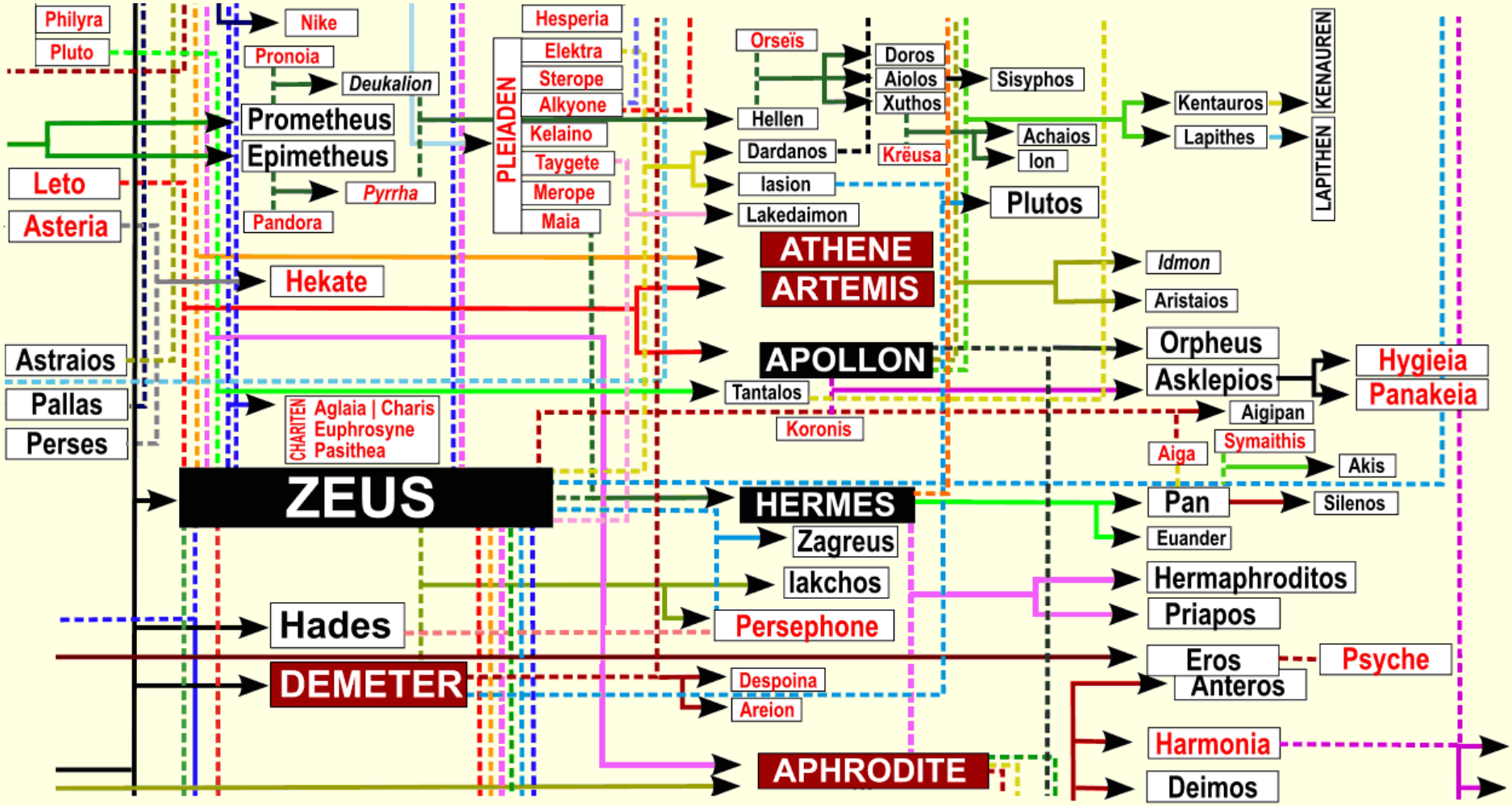
Vorfahren (zeitlich früher)

Nachkommen (zeitlich später)

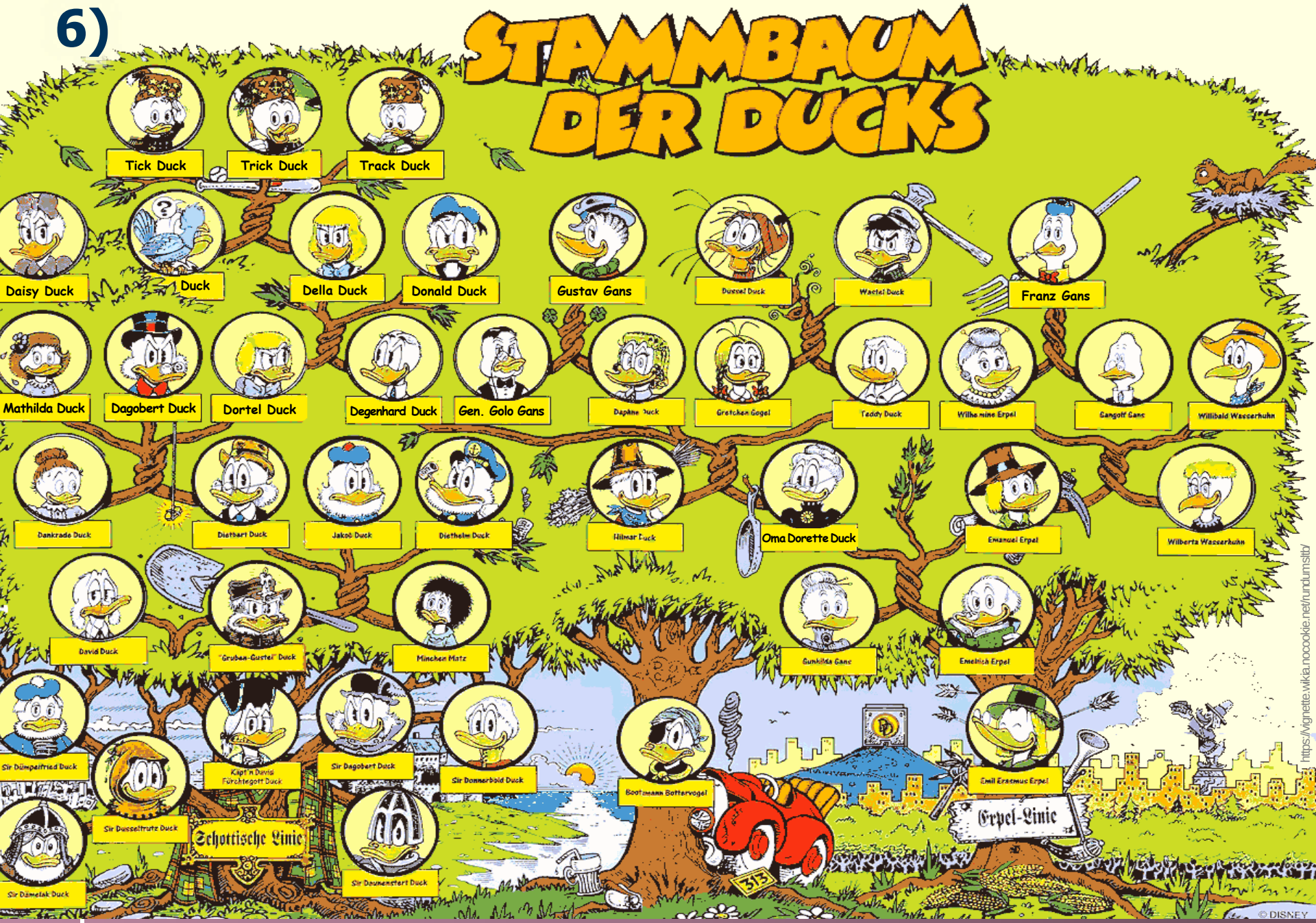


Beachte: Neben den Personen gibt es noch Knoten, die die Verbindung von Eltern symbolisieren. Dadurch kann es zwischen 2 Knoten mehrere Wege geben; im strengen Sinne handelt es sich also nicht um einen Baum! (Und: Womanizer Zeus taucht mehrfach auf; *#MeToo:* Hera, Leto, Maia, Dione, Io,…)

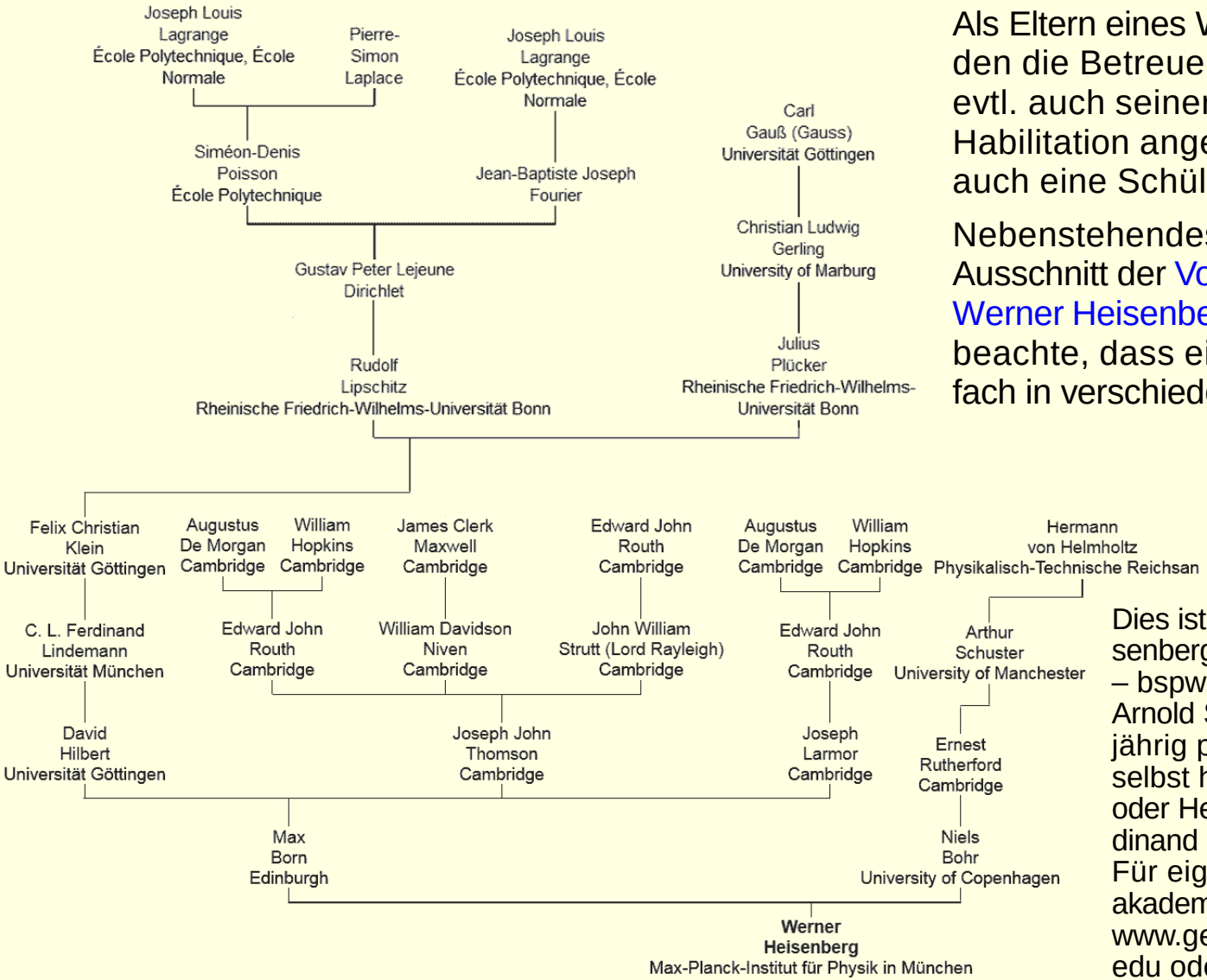
In „Wirklichkeit“ ist die ganze Beziehungskiste bei den Göttern aber viel komplizierter, wie man hier sieht: Wir erkennen einiges wieder, z.B. Hermes als Produkt von Maia und Zeus; Persophone als Frucht von Demeter und Zeus oder Apollo und Artemis als Nachkommen von Zeus und Leto – aber eben auch noch viel mehr: Pluto, Nike, Elektra, Sisyphos, Orpheus, Harmonia, Eros und Psyche haben neben Anderen ihren Auftritt. Dabei ist diese Graphik auch nur ein kleiner Ausschnitt aus dem Gesamtplan der Götterwelt, wie ihn die „Mythographen“ zusammengestellt haben. Die „Gesamtgenealogie der griechisch-mediterranen Mythologie“ von Dieter Macek umfasst über 5000 Göttinnen, Götter und Heroen – der Beziehungsgraph ist 73 m lang und 1.7 m hoch; siehe www.myth-gen.eu.



6)


STAMMBAUM DER DUCKS
Tick Duck
Trick Duck
Track Duck
Daisy Duck
Duck
Della Duck
Donald Duck
Gustav Gans
Dussel Duck
Wachtel Duck
Franz Gans
Mathilda Duck
Dagobert Duck
Dortel Duck
Degenhard Duck
Gen. Golo Gans
Daphne Duck
Gretchen Gogel
Teddy Duck
Wilhelmine Erpel
Gangolf Gans
Willibald Wasserhuhn
Dankrade Duck
Dietbart Duck
Jakob Duck
Diethelm Duck
Hilmar Duck
Oma Dorette Duck
Emanuel Erpel
Wilberta Wasserhuhn
David Duck
"Gruben-Gustel" Duck
München Matz
Gunhilda Gans
Emeinich Erpel
Sir Dümpelfried Duck
Kapt'n Duvisi Fürchtegott Duck
Sir Dagobert Duck
Sir Donnerbold Duck
Bootsmann Bottervogel
Emil Erasmus Erpel
Sir Dusseltrutz Duck
Schottische Linie
Erpel-Linie
Sir Dämelak Duck
Sir Daunenstert Duck
513
https://vignette.wikia.nocookie.net/rundumsitb/
© DISNEY

7) Akademischer Stammbaum



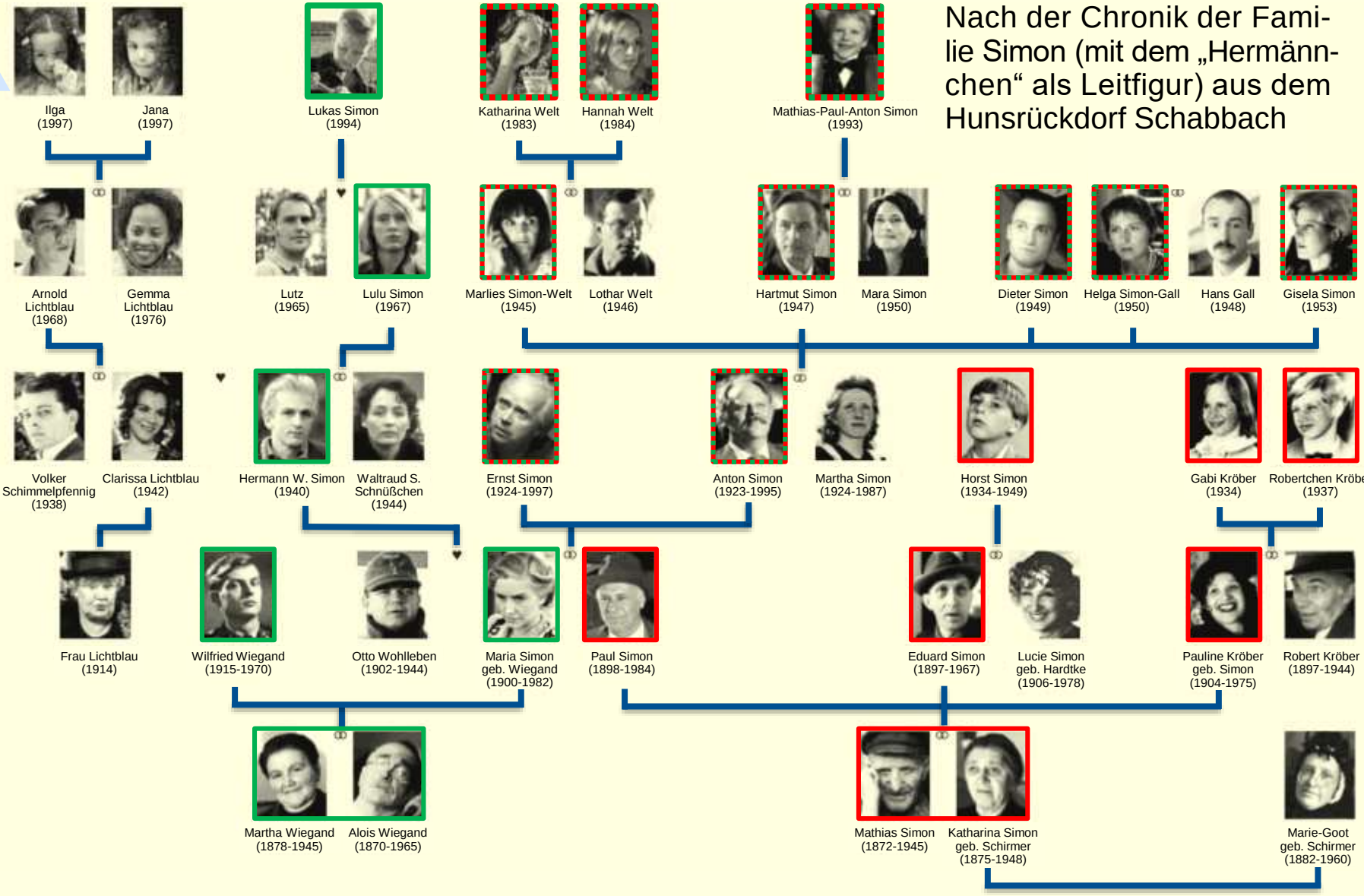
Als Eltern eines Wissenschaftlers werden die Betreuer seiner Dissertation, evtl. auch seiner Postdoktorate und Habilitation angesetzt – allgemeiner auch eine Schüler/Lehrer-Beziehung.

Nebenstehendes Beispiel zeigt einen Ausschnitt der Vorfahren des Physikers Werner Heisenberg (1901 – 1976); man beachte, dass einige Personen mehrfach in verschiedenen Knoten auftreten!

Dies ist nur ein Ausschnitt von Heisenbergs akademischen Vorfahren – bspw. fehlt hier sein Doktorvater Arnold Sommerfeld, bei dem er 23-jährig promovierte; Sommerfeld selbst hat (wie auch David Hilbert oder Hermann Minkowski) bei Ferdinand von Lindemann promoviert. Für eigene Nachforschungen zu akademischer Stammbäume siehe www.genealogy.math.ndsu.nodak.edu oder https://academictree.org

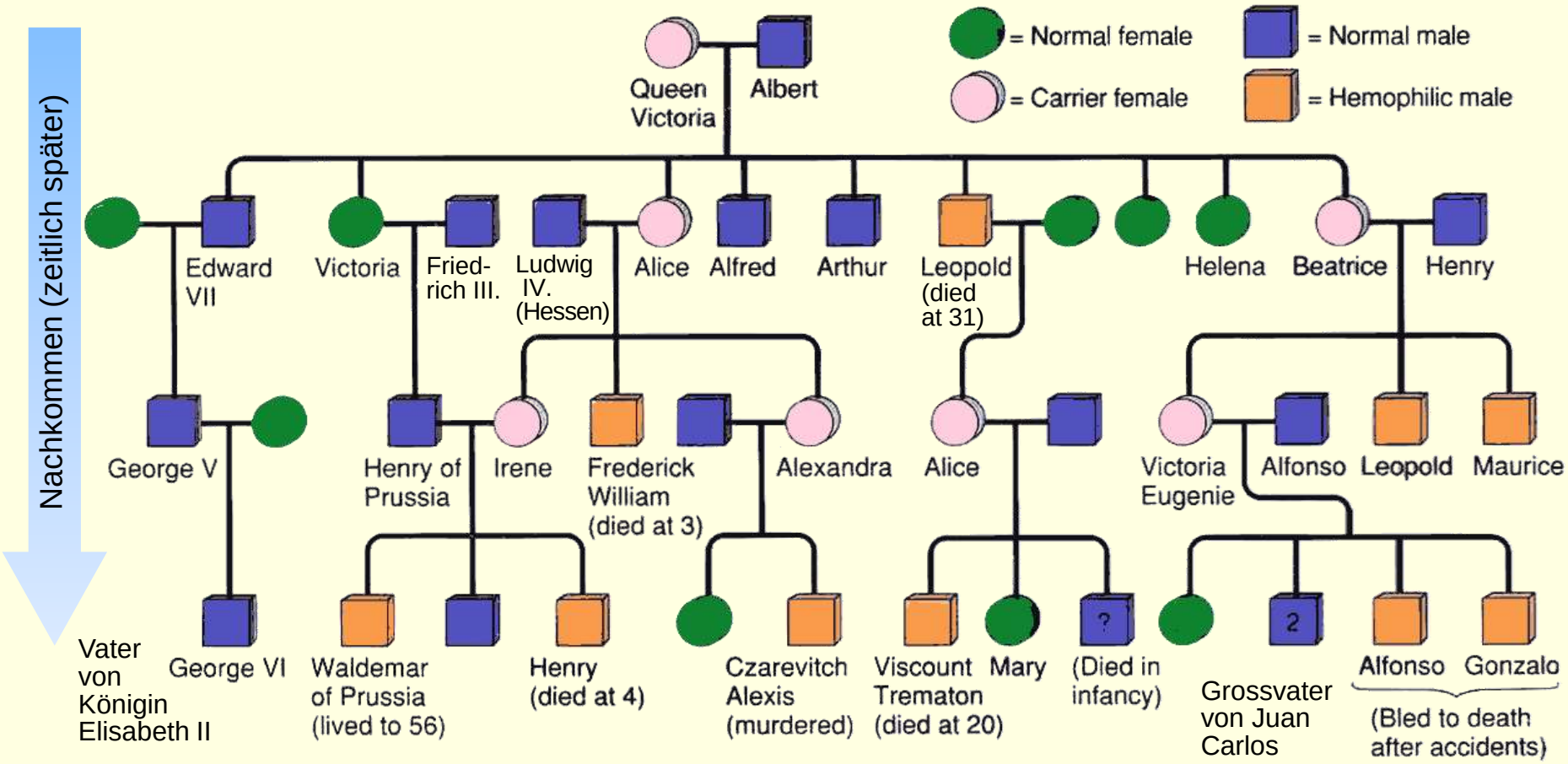
8) Bilden „Blutsverwandte“ einen (Stamm)baum?

Nachkommen (zeitlich später)



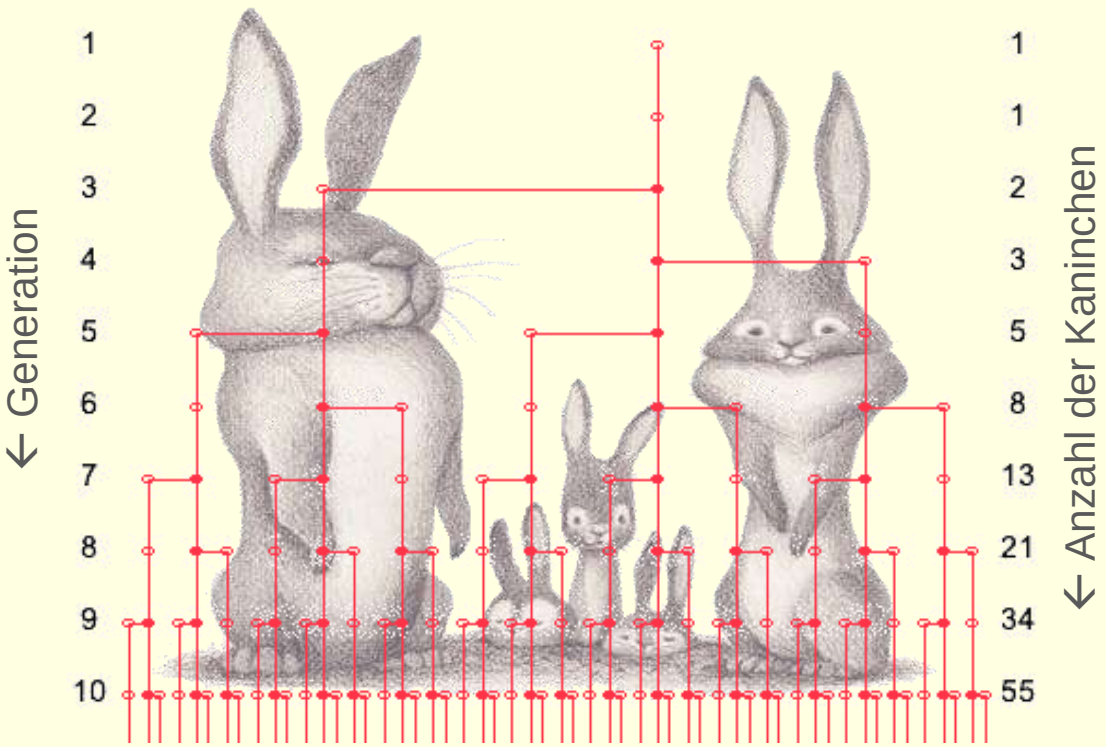
Nach der Chronik der Familie Simon (mit dem „Hermännchen“ als Leitfigur) aus dem Hunsrückdorf Schabbach

9) Vererbung von Hämophilie



Pedigree showing inheritance of hemophilia in the descendants of Queen Victoria (1837–1901). DNA sequencing of samples from the skeletal remains of the murdered Russian royal family reveals that the mutation was a point mutation in the intron preceding exon 4 (of 8) of the factor IX (F9) gene, so the disease was hemophilia B. People with the bleeding disorder hemophilia lack factors that cause the blood to clot. It is a prototype of recessive diseases linked to the X chromosome. The disease affects thousands of people around the world and has even played a part in historic events.

10) Die Kaninchen des Fibonacci



Kaninchen können sich bereits im Alter von einem Monat paaren. Angenommen, am Ende ihres zweiten Lebensmonats produziert jedes Weibchen ein neues Paar (ein Männchen und ein Weibchen). Fibonacci fragte: Wenn keine Kaninchen sterben, wie viele Paare gibt es, ausgehend von einem ersten Paar, nach einem Jahr?

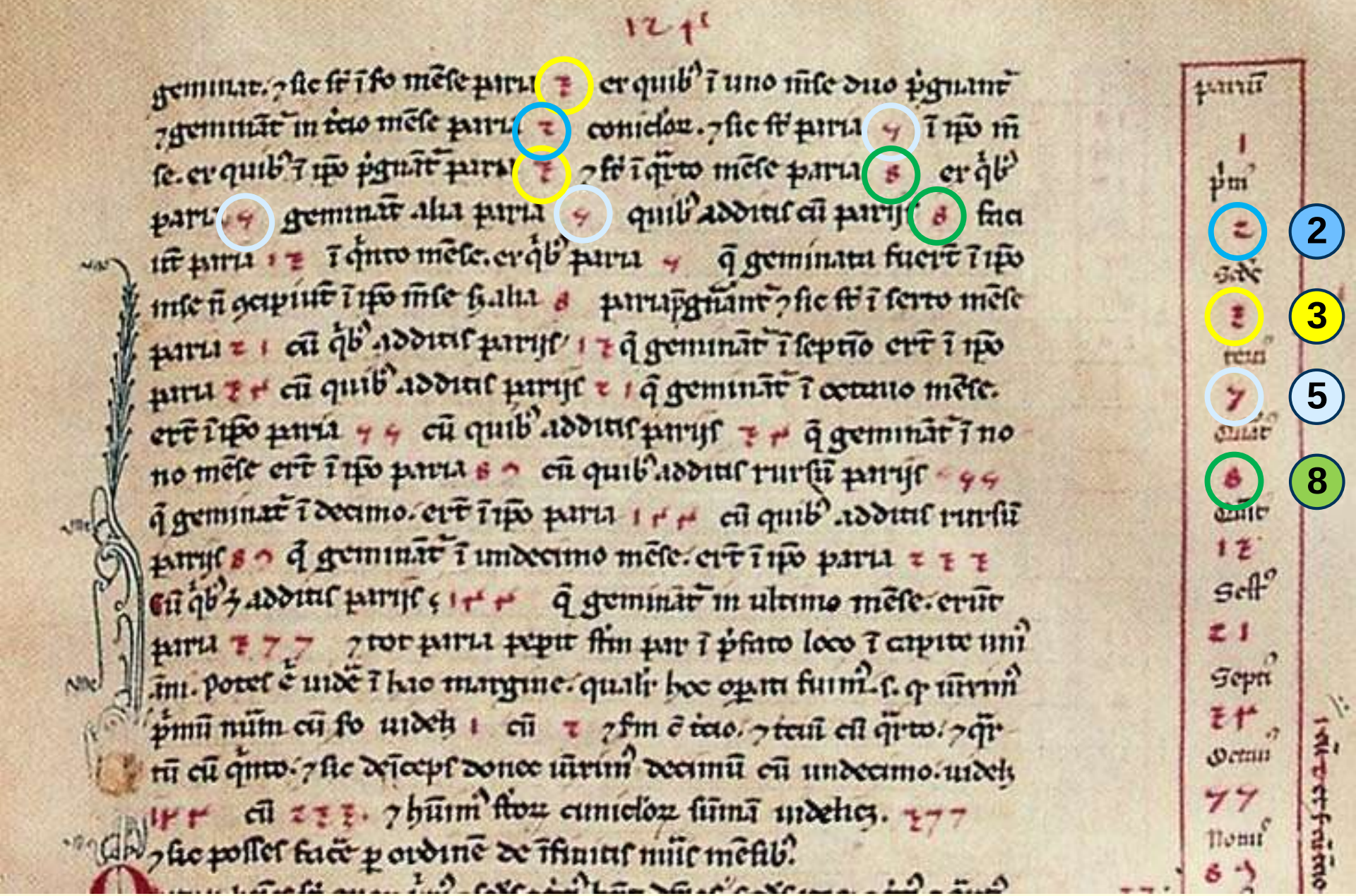
Die Folge 1, 1, 2, 3, 5, 8, 13, 21, ... ($a_{i+2} = a_{i+1} + a_i$) wurde später Fibonacci-Folge genannt. Sie ist nach dem italienischen Mathematiker Leonardo da Pisa (genannt Fibonacci, d.h. „figlio di Bonacci", Sohn des Bonacci), benannt, der damit 1202 das Wachstum einer Kaninchenpopulation beschrieb. Die Folge wächst exponentiell; der Quotient benachbarter Glieder nähert sich schnell dem Goldenen Schnitt (1.618033988…).

Nachdem Fibonacci die Kaninchenzahlen für einige Monate durchexerziert hatte („...*et sic deinceps, donec iunximus decimum cum undecimo, uidelicet 144 cum 233; et habuimus suprascriptorum cuniculorum summam*" – ...und so fort, bis wir zur zehnten die elfte addiert haben, nämlich zu 144 die 233; und wir bekommen die oben erwähnte Summe der Kaninchen), bemerkte er *„et sic posses facere per ordinem de infinitis numeris mensibus"*, man könne dies weiter für eine unendliche Zahl von Monaten durchführen.

Fibonacci lernte als Sohn eines Pisaner Kaufmanns in Nordafrika, Byzanz und Syrien die Mathematik mit den arabischen Ziffern kennen und verfasst später das Rechenbuch *Liber Abbaci*, das das Rechnen mit diesen *novem figurae indorum* in (Süd)europa bekannt machte.

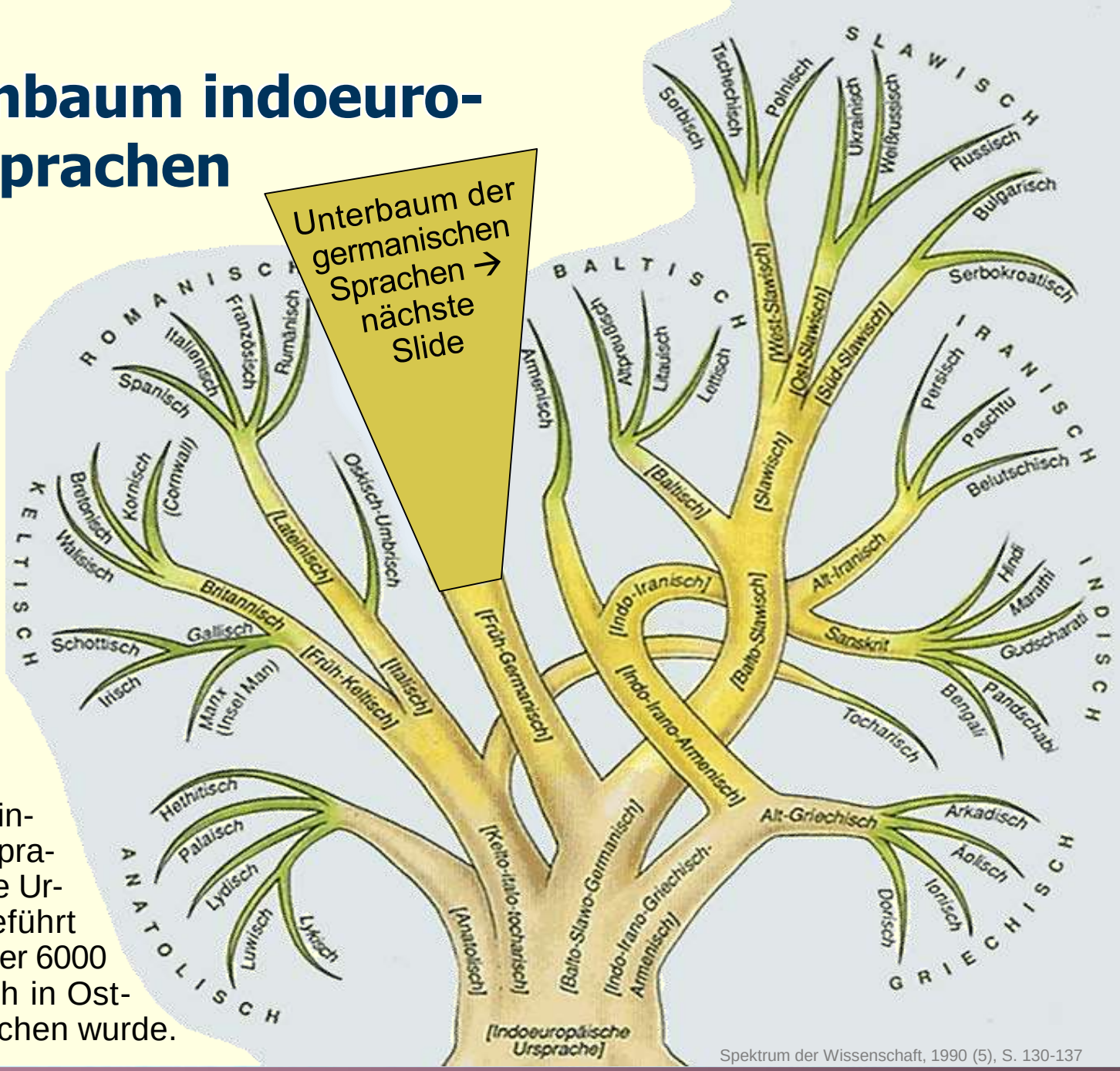
Qvidam posuit unum par cuniculorum in quodam loco, qui erat undique pariete circundatus, ut sciret, quot ex eo paria germinarentur in uno anno: cum natura eorum sit per singulum mensem aliud par germinare; et in secundo mense ab eorum natiuitate germinant. Quia suprascriptum par in primo mense germinat, duplicabis ipsum, erunt paria duo in uno mense. Ex quibus unum, scilicet primum, in secundo mense geminat; et sic sunt in secundo mense paria 3; ex quibus in uno mense duo pregnantur; et geminantur in tercio mense paria 2 coniculorum; et sic sunt paria 5 in ipso mense; ex quibus in ipso pregnantur paria 3; et sunt in quarto mense paria 8; ex quibus paria 5 geminant alia paria 5: quibus additis cum parijs 8, faciunt paria 13 in quinto mense; ex quibus paria 5, que geminata fuerunt in ipso mense, non concipiunt in ipso mense, sed alia 8 paria pregnantur; et sic sunt in sexto mense paria 21; cum quibus additis parijs 13, que geminantur in septimo, erunt in ipso paria 34; cum quibus additis parijs 21, que geminantur in octauo mense, erunt in ipso paria 55; cum quibus additis parjis [*sic*] 34, que geminantur in nono mense, erunt in ipso paria 89; cum quibus additis rursum parijs 55, que geminantur in decimo mense 144; cum quibus additis rursum parijs 89, que geminantur in undecimo mense, erunt in ipso paria 233. Cum quibus etiam additis parijs 144, que geminantur in ultimo mense, erunt paria 377; et tot paria peperit suprascriptum par in prefato loco in capite unius anni. Potes enim uidere in hac margine, qualiter hoc operati fuimus, scilicet quod iunximus primum numerum cum secundo, uidelicet 1 cum 2; et secundum cum tercio; et tercium cum quarto; et quartum cum quinto, **et sic deinceps, donec iunximus decimum cum undecimo, uidelicet 144 cum 233; et habuimus suprascriptorum cuniculorum summam, uidelicet 377; et sic posses facere per ordinem de infinitis numeris mensibus.**

Jemand setzte ein Kaninchenpärchen in einen solchen Ort, der allseits mit Wänden umgrenzt war. Man wünscht zu wissen, wie viele Nachkommen dieses Paares in einem Jahr erzeugt werden. Dabei seien sie so beschaffen, dass sie in jedem Monat ein neues Paar erzeugen; und ab dem zweiten Monat nach ihrer Geburt sind auch die Jungen fruchtbar. Das oben beschriebene Paar wirft im ersten Monat Junge, verdoppelt sich selbst, so dass es zwei Pärchen in einem Monat sind. Von diesen verdoppelt sich eines, nämlich das erste, im zweiten Monat; und so sind im zweiten Monat 3 Pärchen; von diesem werden in einem Monat zwei schwanger; und es entstehen im dritten Monat 2 neue Pärchen; und so sind es 5 Pärchen in diesem Monat; von diesen werden 3 Pärchen schwanger; so dass es im vierten Monat 8 Pärchen sind; von diesen erzeugen 5 Pärchen weitere 5 Pärchen: Diese werden zu den 8 Pärchen hinzugefügt, was 13 Pärchen im fünften Monat ergibt; von diesen werden jene 5, die in diesem Monat geboren wurden, nicht schwanger in diesem Monat, aber die anderen 8 werden es; und so sind es im sechsten Monat 21 Pärchen; dazu kommen 34 Pärchen, die sich im neunten Monat verdoppeln, so dass es in diesem 89 Pärchen werden; zu diesen werden wiederum 55 Pärchen addiert, die sich im zehnten Monat verdoppeln, das sind 144; dazu kommen wieder 89 Pärchen, die sich im elften Monat verdoppeln, das sind in diesem 233 Pärchen. Zu diesen werden noch 144 Pärchen addiert, die im letzten Monat geboren werden, das sind 377 Pärchen; und alle Pärchen stammen von dem oben beschriebenen Pärchen im bereitgestellten Ort während eines Jahres. Ihr könnt am Rand sehen, wie wir dir Rechnung ausgeführt haben, wir haben nämlich die erste Zahl mit der zweiten vereinigt, also 1 mit 2; und die zweite mit der dritten; und die dritte mit der vierten; und die vierte mit der fünften, **und so fort, bis wir zur zehnten die elfte addiert haben, nämlich zu 144 die 233; und wir bekommen die oben erwähnte Summe der Kaninchen, nämlich 377; und so könnt Ihr es nach der Reihe mit einer unendlichen Zahl von Monaten machen.**



Ausschnitt aus dem Liber Abaci (vorherige slide, ab „*…geminat; et sic sunt in secundo mense paria 3…*“ bis zum Abschnittsende „*…ordinem de infinitis numeris mensibus*“).

11) Stammbaum indoeuropäischer Sprachen



Der Stammbaum indoeuropäischer Sprachen kann auf eine Ursprache zurückgeführt werden, die vor über 6000 Jahren (vermutlich in Ostanatolien) gesprochen wurde.

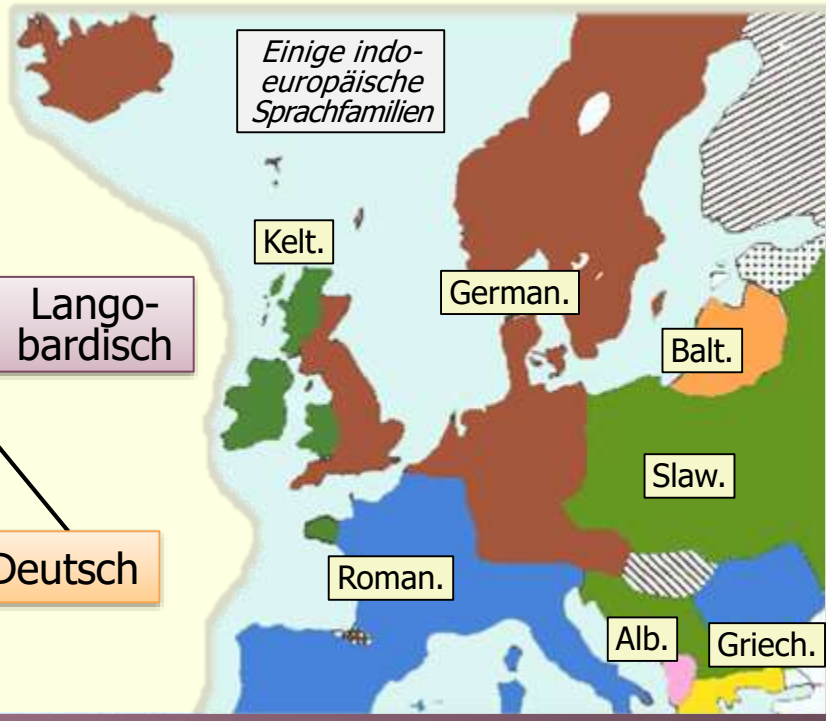
Spektrum der Wissenschaft, 1990 (5), S. 130-137

Unterbaum der germanischen Sprachentwicklung

ausgestorben

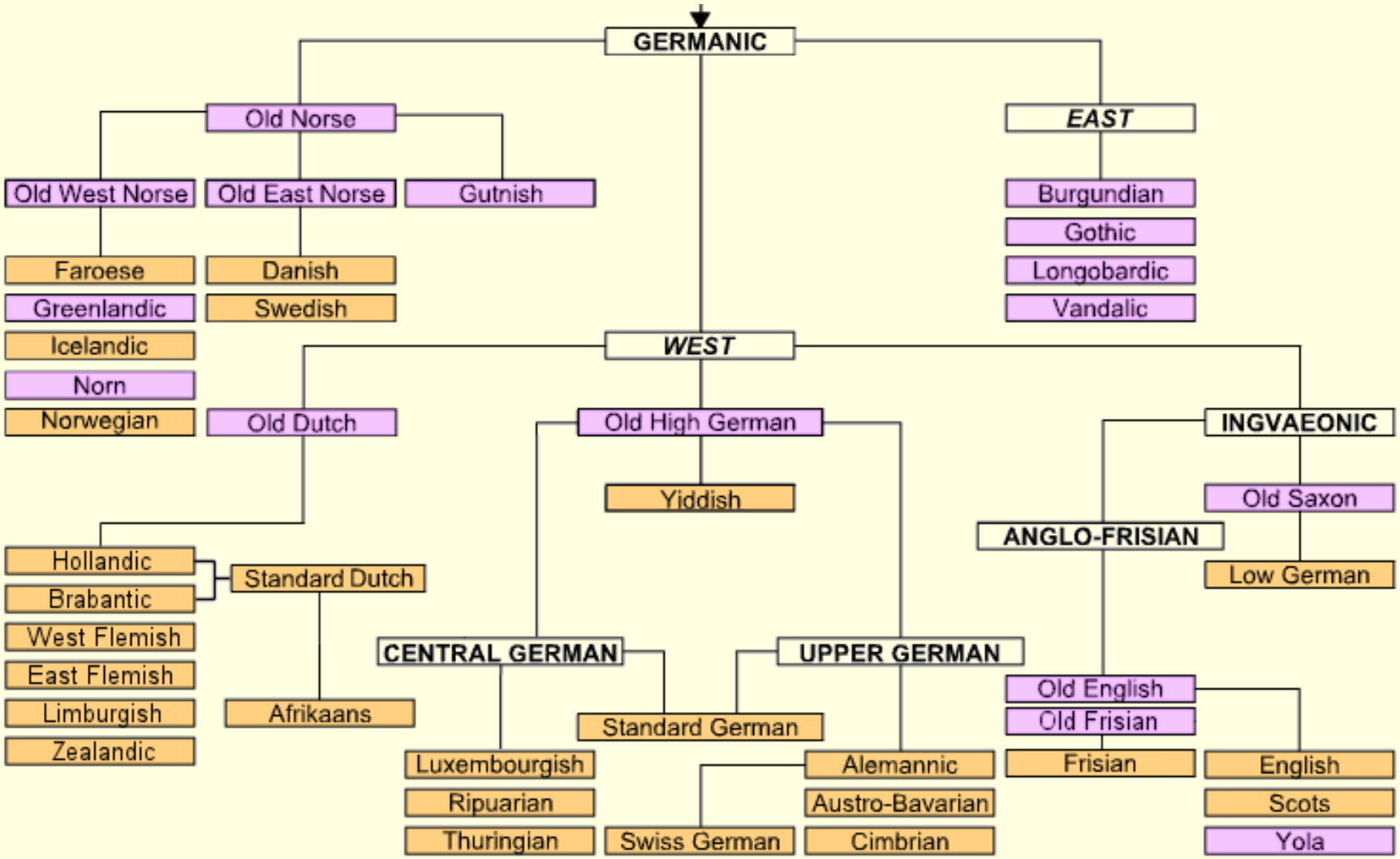
lebende Sprache

- **Indoeuropäisch**
  - **Germanisch**
    - **Nord-Germanisch**
      - West-Nordisch
        - Isländisch
        - Norwegisch
      - Ost-Nordisch
        - Dänisch
        - Schwedisch
    - **West-Germanisch**
      - Niederländisch (Flämisch); Afrikaans
      - Angelsächsisch
        - Englisch
      - Friesisch
      - Deutsch
      - Langobardisch
    - **Ost-Germanisch**
      - Burgundisch
      - Vandalisch
      - Gotisch



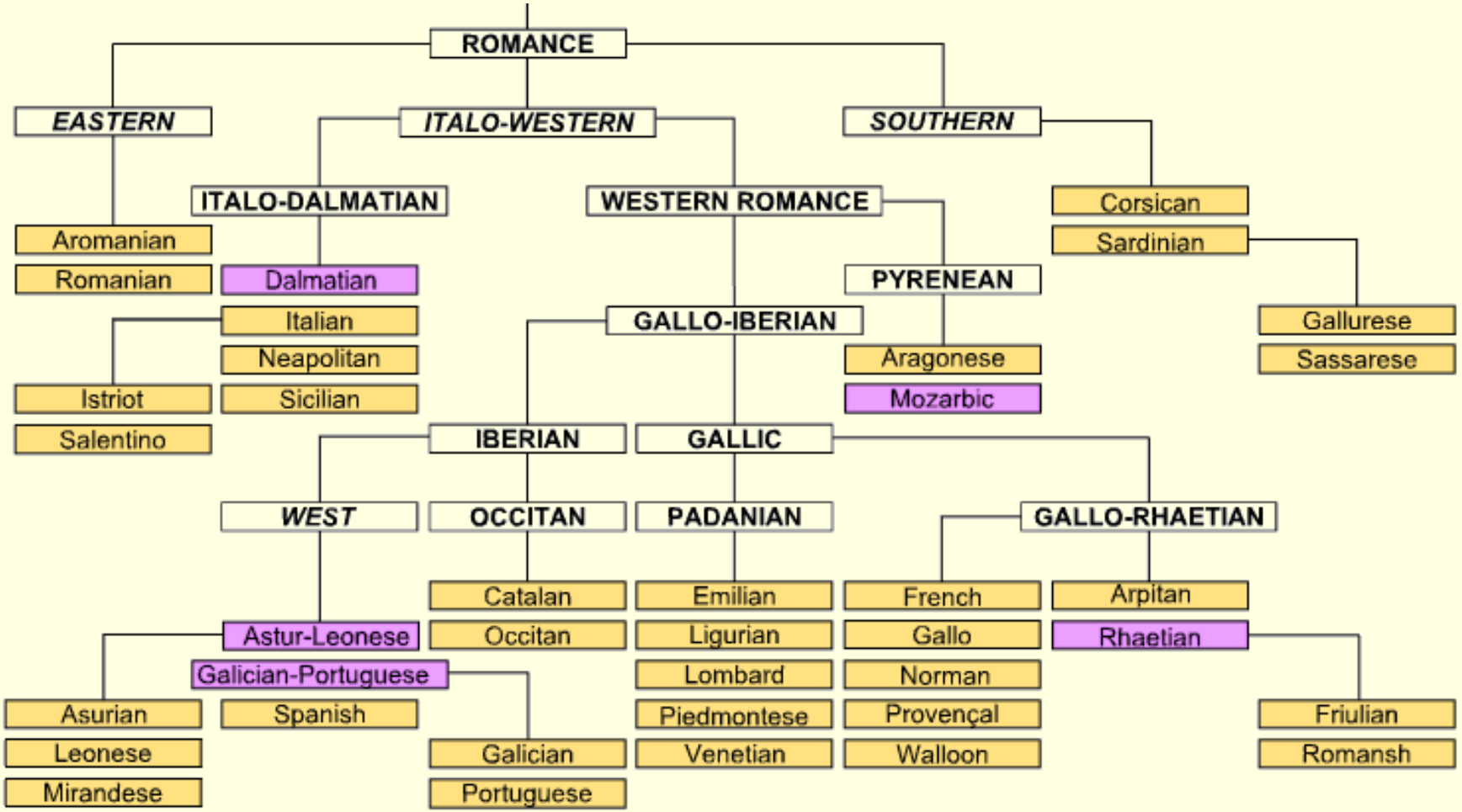
- *Wo ist Schweizerdeutsch?*
- *Und fehlen nicht Jiddisch und Niederdeutsch?*

Eine etwas andere / detailliertere Klassifikation



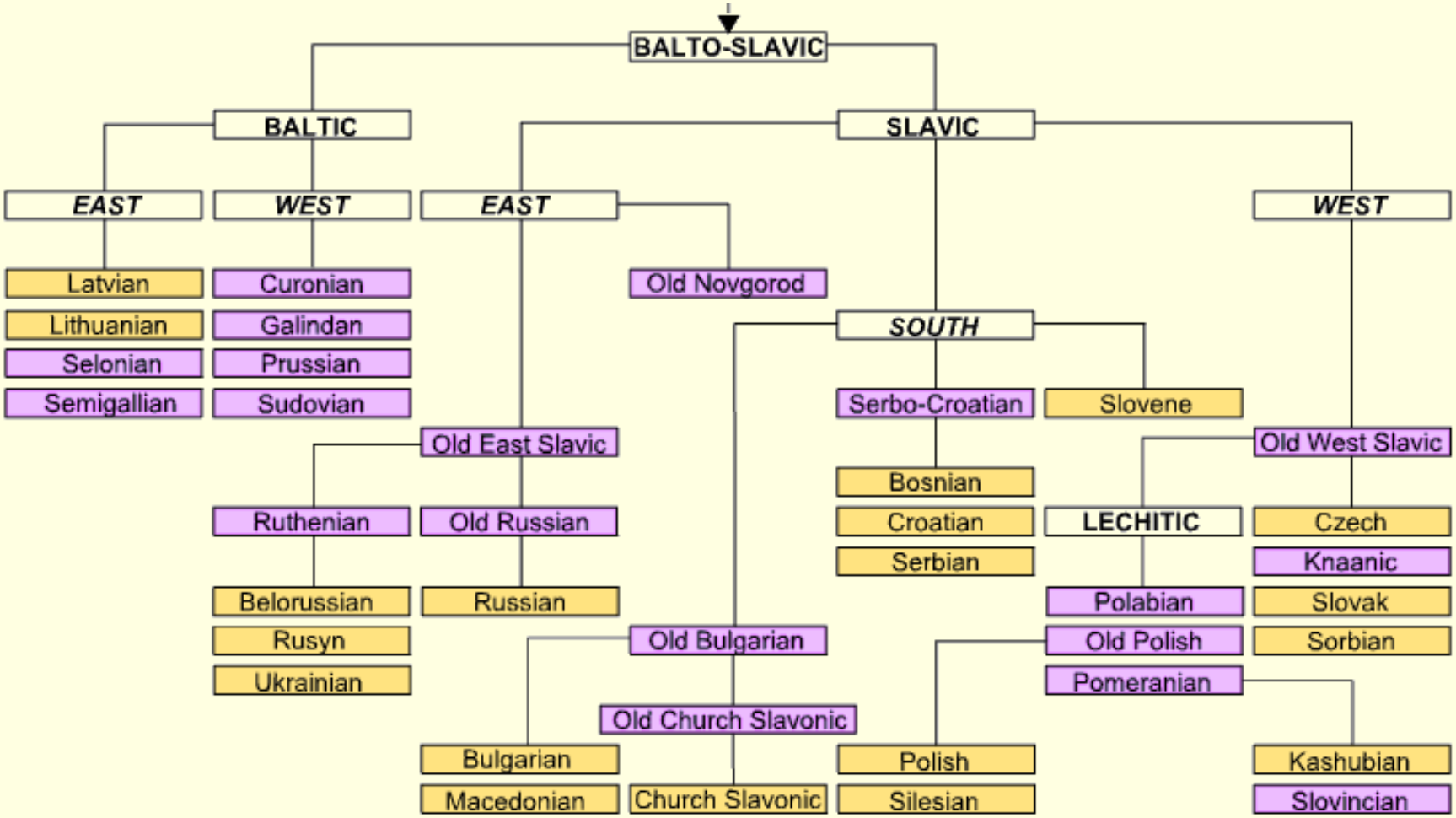
https://elimperiodedes.files.wordpress.com/2013/03/indoeuropeantree.png

Unterbaum der romanischen Sprachen



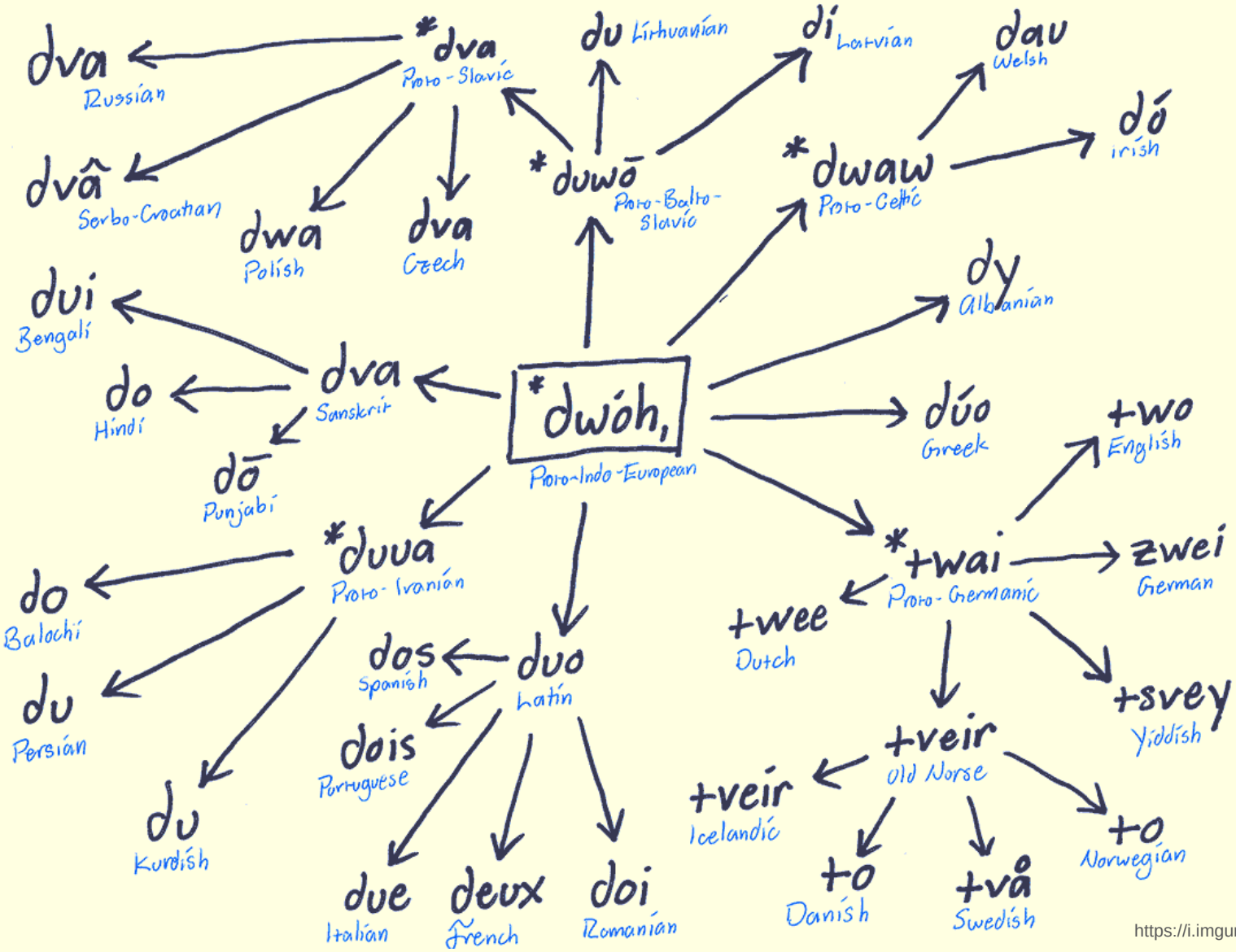
https://elimperiodedes.files.wordpress.com/2013/03/indoeuropeantree.png

Unterbaum der balto-slawischen Sprachen



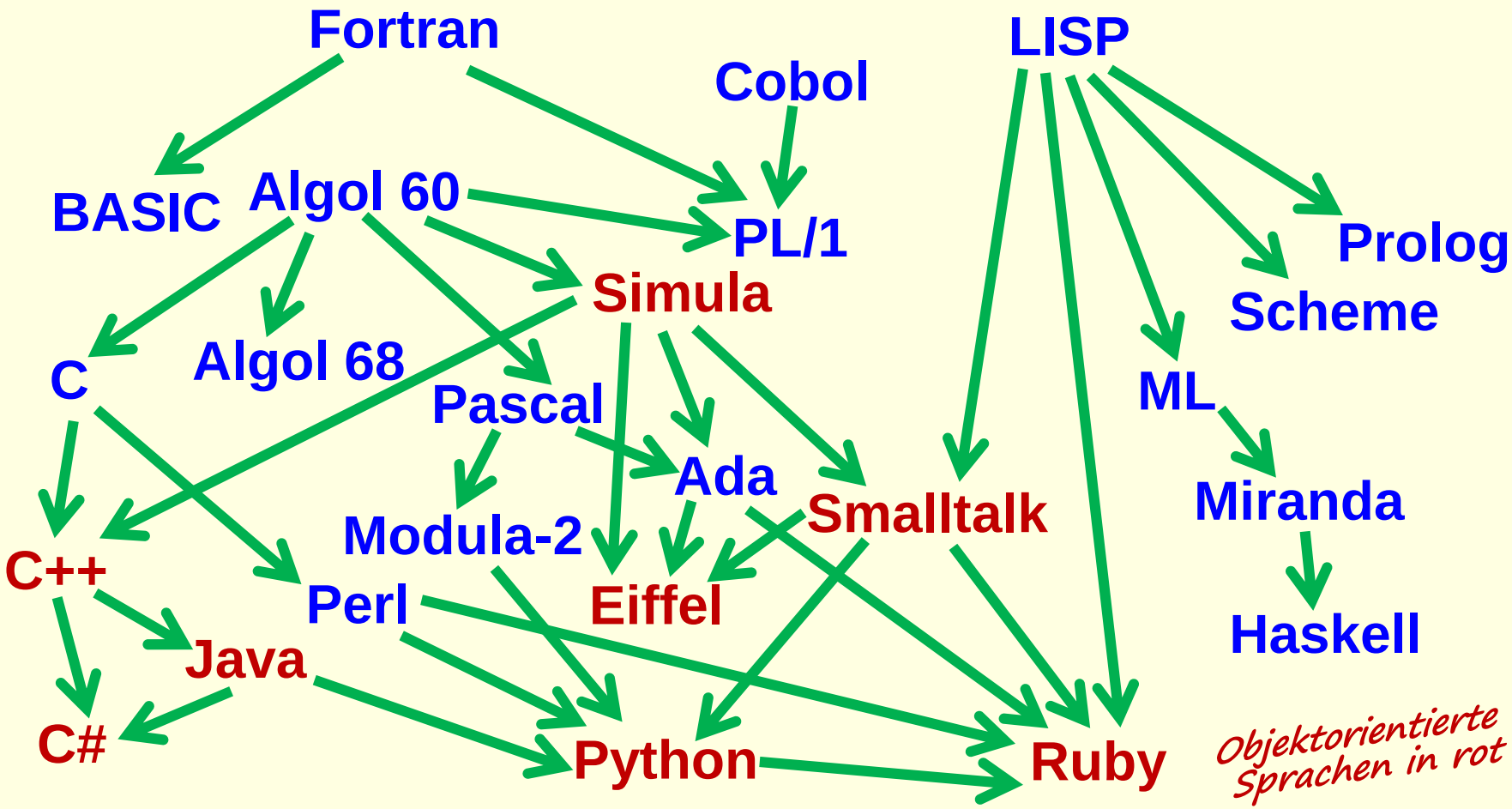
https://elimperiodedes.files.wordpress.com/2013/03/indoeuropeantree.png

12) Etymologischer Entwicklungsbaum von „zwei“



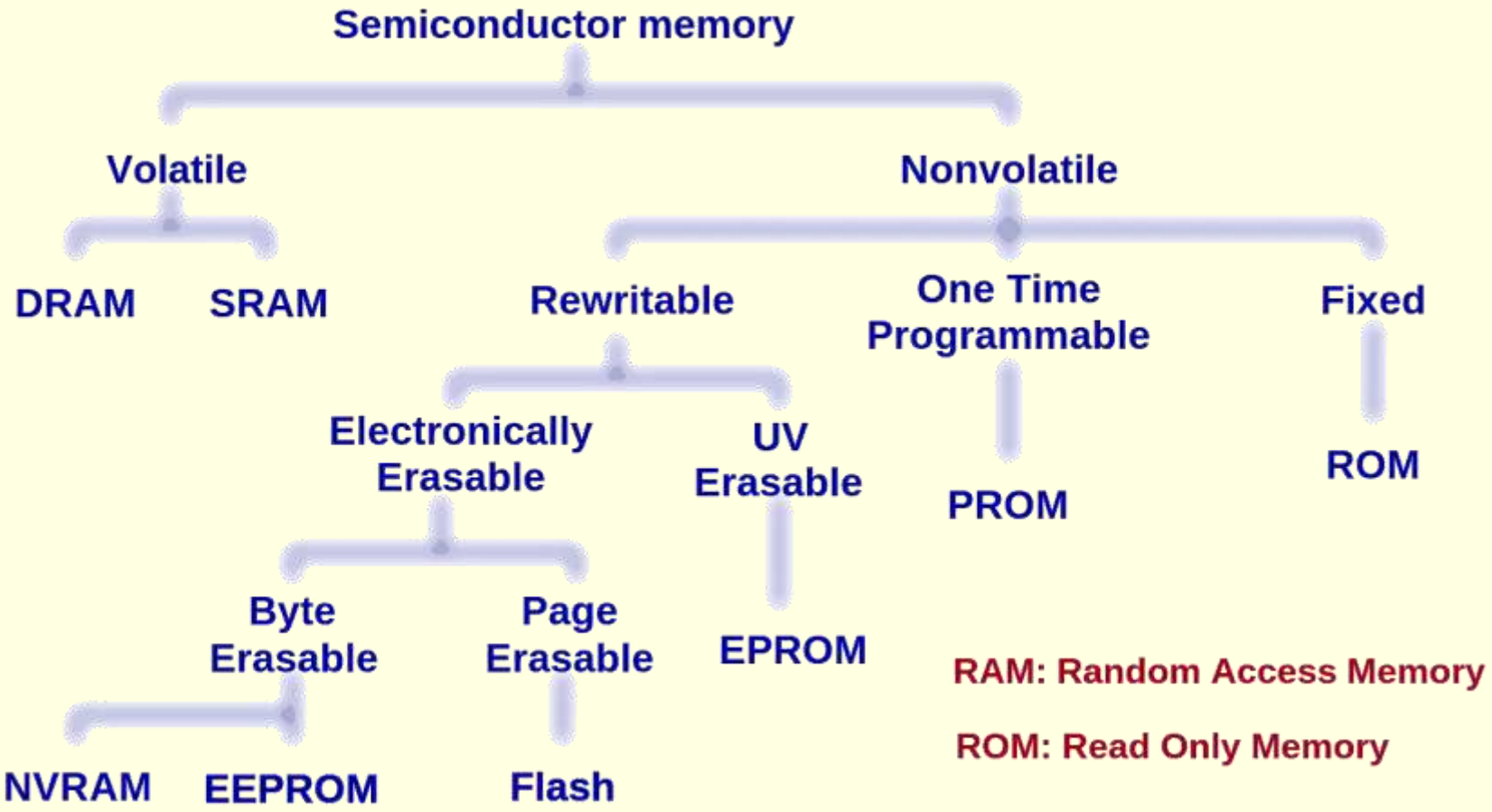
https://i.imgur.com/7AeYcfK.jpg

13) Ein Programmiersprachen-Stammbaum

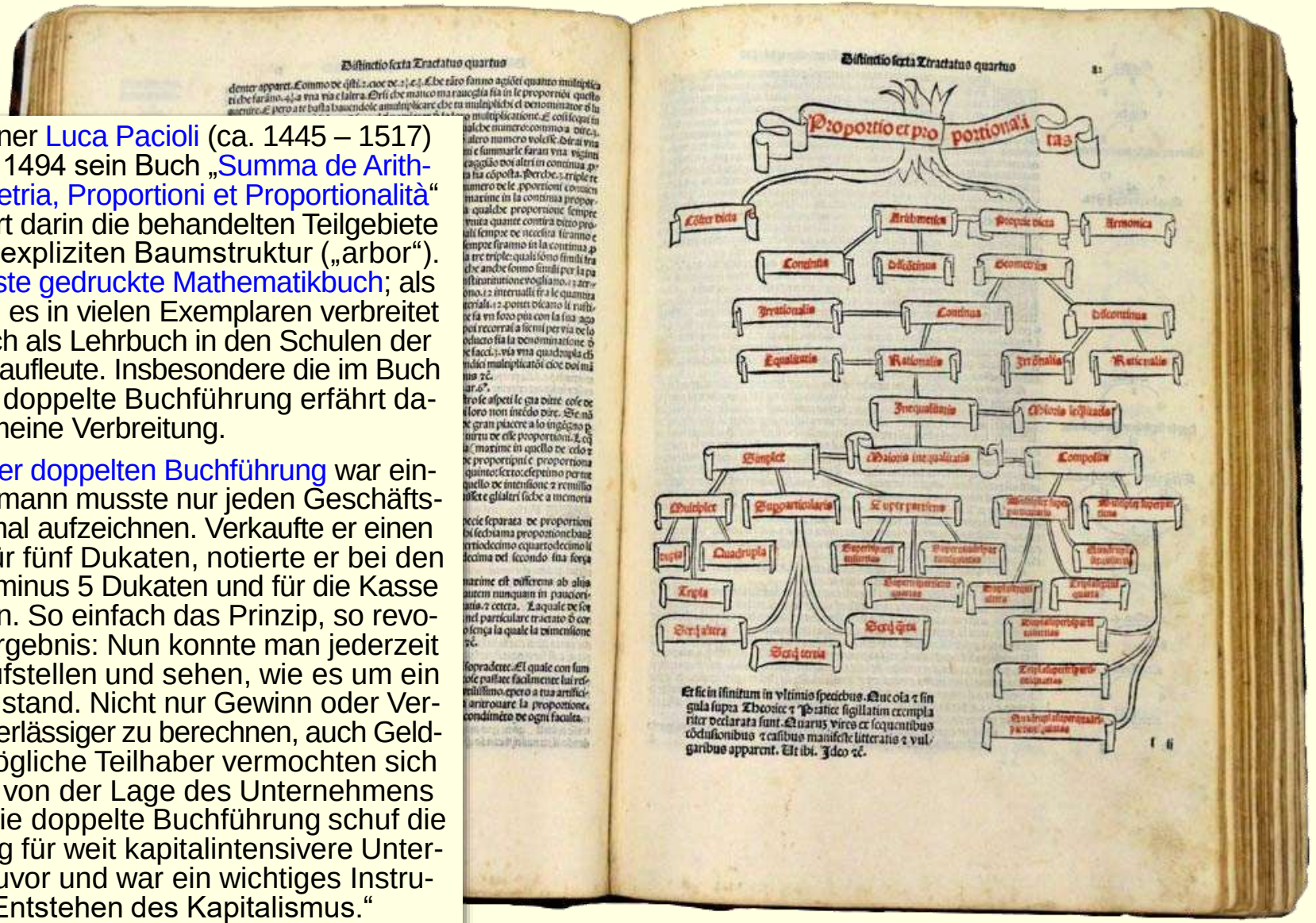


Dies ist allerdings nur eine sehr grobe Darstellung der Verwandtschafts- und Abstammungsgeschichte. Algol 60 wird oft als das „Latein der Programmiersprachen“ bezeichnet, weil viele Eigenschaften von Algol in die heutigen Programmiersprachen vererbt worden sind. Entsprechend stellt Simula das „objektorientierte Latein“ dar.

14) Klassifikation von Halbleiterspeichern



15) Arbor Proportio et Proportionalitas



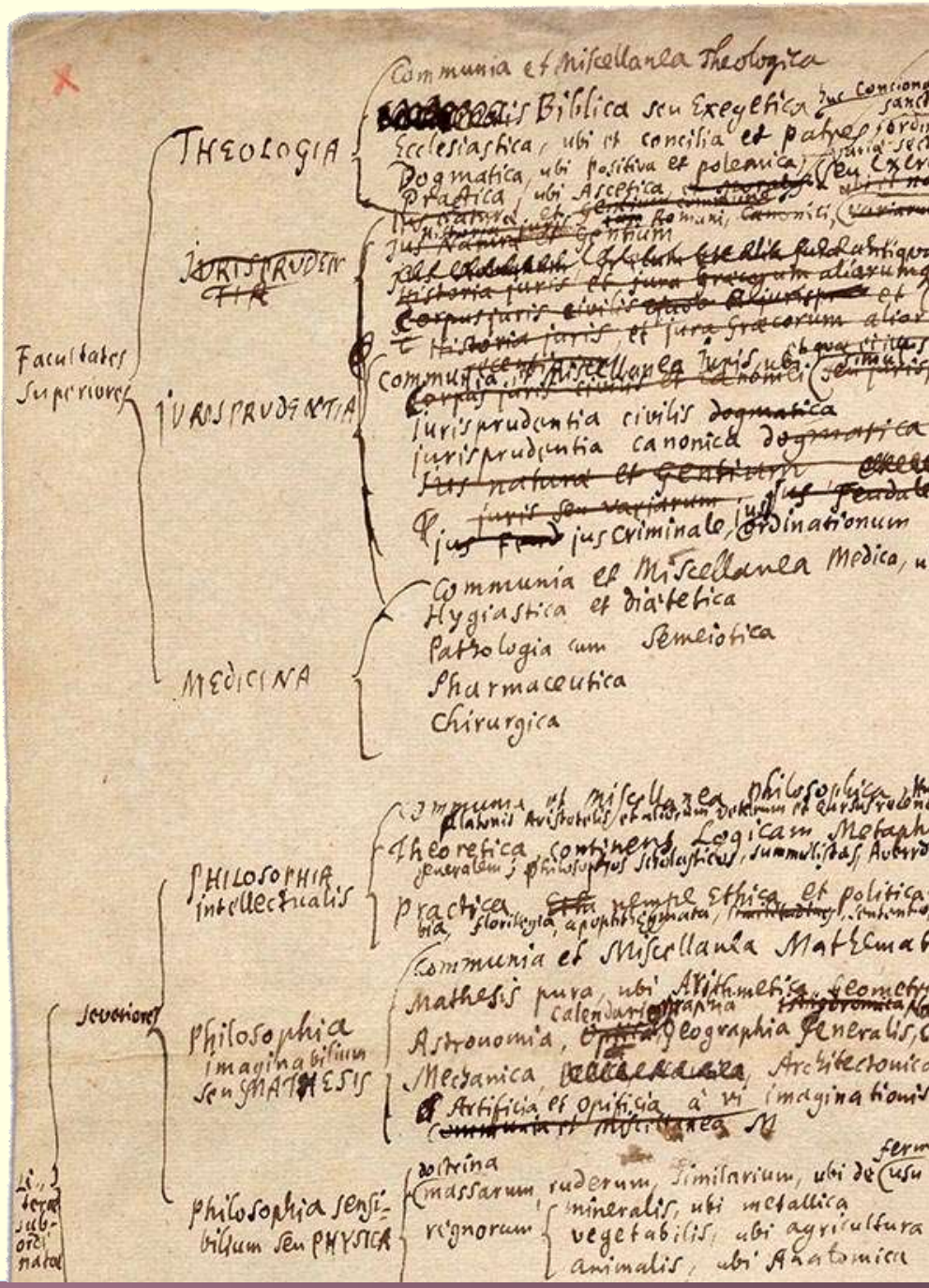
Der Franziskaner Luca Pacioli (ca. 1445 – 1517) veröffentlichte 1494 sein Buch „Summa de Arithmetica, Geometria, Proportioni et Proportionalità“ und klassifiziert darin die behandelten Teilgebiete anhand einer expliziten Baumstruktur („arbor“). Es war das erste gedruckte Mathematikbuch; als solches wurde es in vielen Exemplaren verbreitet und diente auch als Lehrbuch in den Schulen der italienischen Kaufleute. Insbesondere die im Buch beschriebene doppelte Buchführung erfährt damit eine allgemeine Verbreitung.

„Das Prinzip der doppelten Buchführung war einfach: Ein Kaufmann musste nur jeden Geschäftsvorgang zweimal aufzeichnen. Verkaufte er einen Ballen Tuch für fünf Dukaten, notierte er bei den Tuchvorräten minus 5 Dukaten und für die Kasse plus 5 Dukaten. So einfach das Prinzip, so revolutionär das Ergebnis: Nun konnte man jederzeit eine Bilanz aufstellen und sehen, wie es um ein Unternehmen stand. Nicht nur Gewinn oder Verlust waren zuverlässiger zu berechnen, auch Geldgeber oder mögliche Teilhaber vermochten sich rasch ein Bild von der Lage des Unternehmens zu machen. Die doppelte Buchführung schuf die Voraussetzung für weit kapitalintensivere Unternehmen als zuvor und war ein wichtiges Instrument für das Entstehen des Kapitalismus.“
[Die Zeit, 24/2006]

16) Leibniz' Wissens-systematik

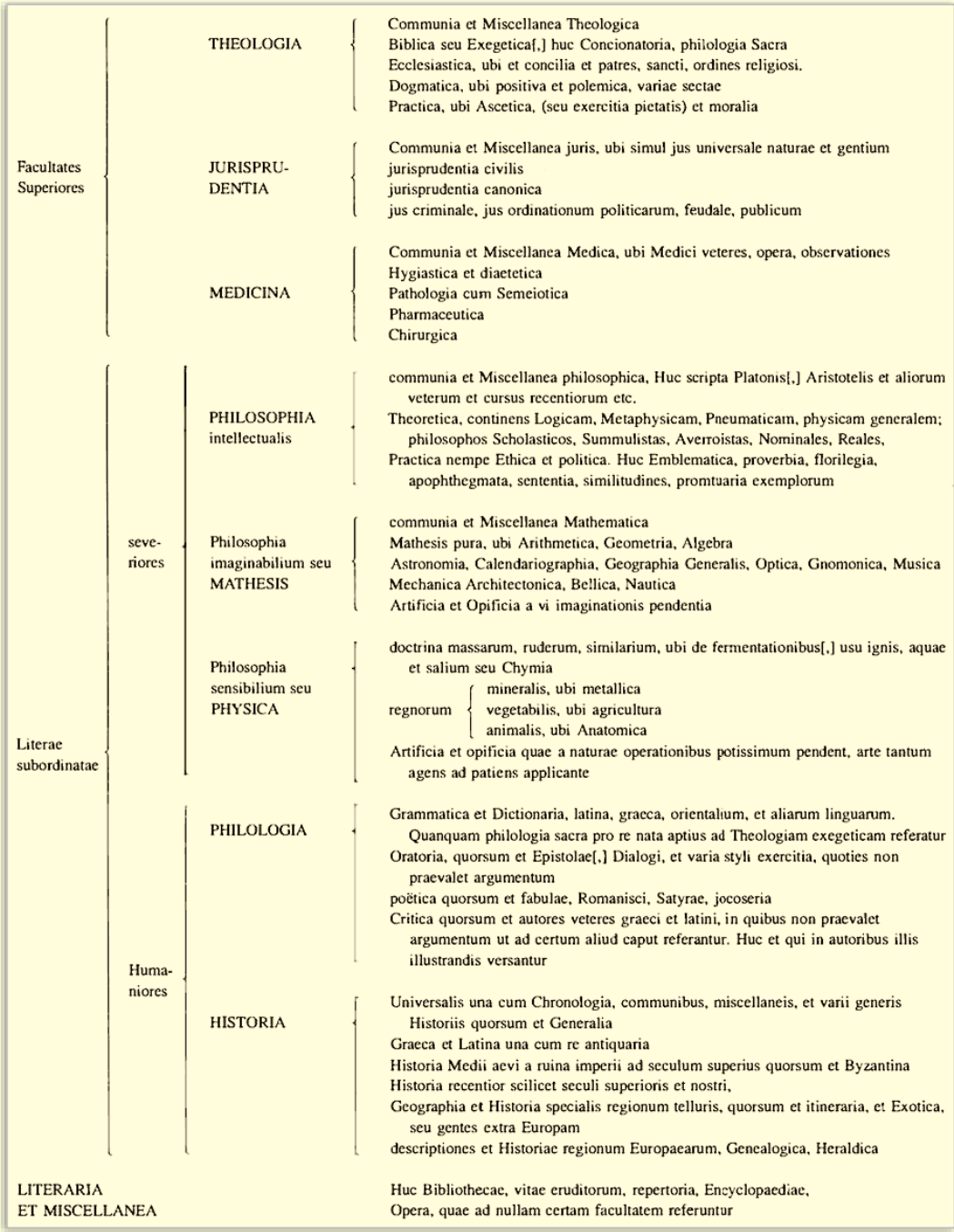
Bevor Texte in digitaler Form vorlagen, vor Datenbanken, dem Internet und Suchmaschinen, war das Wissen der Welt und das Wissen über die Welt in Bibliotheken gesammelt, dort aber nur schwer aufzufinden und zu erschliessen, da die Bücher oft nicht systematisch angeordnet waren und es praktisch keine systematischen Kataloge gab.

Gottfried Wilhelm Leibniz übernahm ab 1691 die Verantwortung als oberster Bibliothekar für die berühmte Büchersammlung von Wolfenbüttel, mit rund 135 000 Titeln seinerzeit eine der grössten wissenschaftlichen Bibliotheken der Welt. Für diese gab es bis dahin im Wesentlichen nur einen chronologische geführten Erwerbskatalog (im Durchschnitt kamen ca. 10 Titel pro Tag hinzu), in dem die Metadaten der neuen Bücher (inklusive einer Klassifikation nach 20 Sachkategorien) verzeichnet waren.



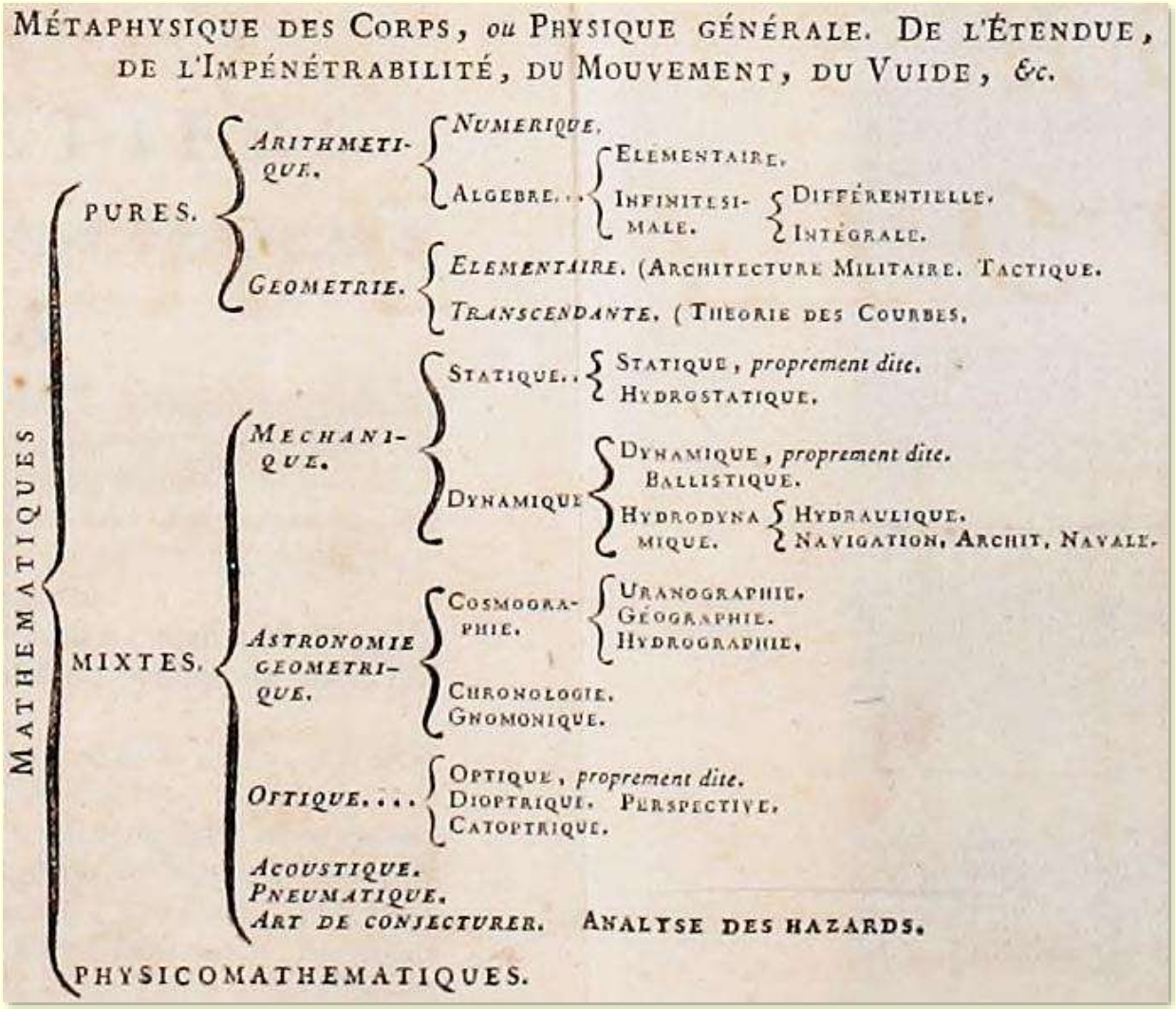
Zu den ersten Massnahmen von Leibniz gehörte es, die einzelnen Einträge des in Buchform vorhandenen Katalogs komplett auf einzelne Zettel abschreiben zu lassen, diese dann neu nach Autoren zu ordnen und zu einem neuen Katalog zusammenzukleben. Der so entstandene alphabetische Verfasserkatalog wurde anschliessend nochmals in systematisierter Weise abgeschrieben. Das ganze Vorhaben dauerte acht Jahre.

Leibniz ging es aber in Prinzip um viel mehr: Er wollte das Wissen der Welt auch systematisch klassifizieren und befasste sich immer wieder mit der Frage nach einem geeigneten hierarchischen Begriffssystem. Der Ausgangspunkt für die Ordnung des Wissens waren die vier traditionellen Fakultäten der klassischen Universitäten – Theologie, Jurisprudenz, Medizin und Philosophie. Mühe hatte Leibniz bei der „vernünftigen“ Anordnung der tieferen Knoten seines Baumes, den er so darstellte, dass rekursiv den jeweils übergeordneten Begriffen mittels Schweifklammern untergeordnete Begriffe zugeteilt werden.



- Facultates Superiores
  - THEOLOGIA
    - Communia et Miscellanea Theologica
    - Biblica seu Exegetica[,] huc Concionatoria, philologia Sacra
    - Ecclesiastica, ubi et concilia et patres, sancti, ordines religiosi.
    - Dogmatica, ubi positiva et polemica, variae sectae
    - Practica, ubi Ascetica, (seu exercitia pietatis) et moralia
  - JURISPRUDENTIA
    - Communia et Miscellanea juris, ubi simul jus universale naturae et gentium
    - jurisprudentia civilis
    - jurisprudentia canonica
    - jus criminale, jus ordinationum politicarum, feudale, publicum
  - MEDICINA
    - Communia et Miscellanea Medica, ubi Medici veteres, opera, observationes
    - Hygiastica et diaetetica
    - Pathologia cum Semeiotica
    - Pharmaceutica
    - Chirurgica
- Literae subordinatae
  - severiores
    - PHILOSOPHIA intellectualis
      - communia et Miscellanea philosophica, Huc scripta Platonis[,] Aristotelis et aliorum veterum et cursus recentiorum etc.
      - Theoretica, continens Logicam, Metaphysicam, Pneumaticam, physicam generalem; philosophos Scholasticos, Summulistas, Averroistas, Nominales, Reales,
      - Practica nempe Ethica et politica. Huc Emblematica, proverbia, florilegia, apophthegmata, sententia, similitudines, promtuaria exemplorum
    - Philosophia imaginabilium seu MATHESIS
      - communia et Miscellanea Mathematica
      - Mathesis pura, ubi Arithmetica, Geometria, Algebra
      - Astronomia, Calendariographia, Geographia Generalis, Optica, Gnomonica, Musica
      - Mechanica Architectonica, Bellica, Nautica
      - Artificia et Opificia a vi imaginationis pendentia
    - Philosophia sensibilium seu PHYSICA
      - doctrina massarum, ruderum, similarium, ubi de fermentationibus[,] usu ignis, aquae et salium seu Chymia
      - regnorum
        - mineralis, ubi metallica
        - vegetabilis, ubi agricultura
        - animalis, ubi Anatomica
      - Artificia et opificia quae a naturae operationibus potissimum pendent, arte tantum agens ad patiens applicante
  - Humaniores
    - PHILOLOGIA
      - Grammatica et Dictionaria, latina, graeca, orientalium, et aliarum linguarum. Quanquam philologia sacra pro re nata aptius ad Theologiam exegeticam referatur
      - Oratoria, quorsum et Epistolae[,] Dialogi, et varia styli exercitia, quoties non praevalet argumentum
      - poëtica quorsum et fabulae, Romanisci, Satyrae, jocoseria
      - Critica quorsum et autores veteres graeci et latini, in quibus non praevalet argumentum ut ad certum aliud caput referantur. Huc et qui in autoribus illis illustrandis versantur
    - HISTORIA
      - Universalis una cum Chronologia, communibus, miscellaneis, et varii generis Historiis quorsum et Generalia
      - Graeca et Latina una cum re antiquaria
      - Historia Medii aevi a ruina imperii ad seculum superius quorsum et Byzantina
      - Historia recentior scilicet seculi superioris et nostri,
      - Geographia et Historia specialis regionum telluris, quorsum et itineraria, et Exotica, seu gentes extra Europam
      - descriptiones et Historiae regionum Europaearum, Genealogica, Heraldica
- LITERARIA ET MISCELLANEA
  - Huc Bibliothecae, vitae eruditorum, repertoria, Encyclopaediae,
  - Opera, quae ad nullam certam facultatem referuntur

17) Système figuré des connaissances humaines

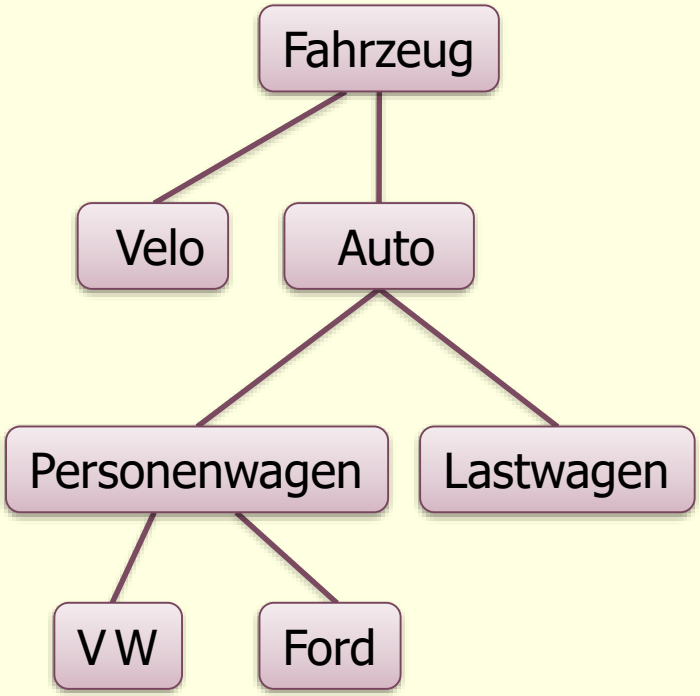


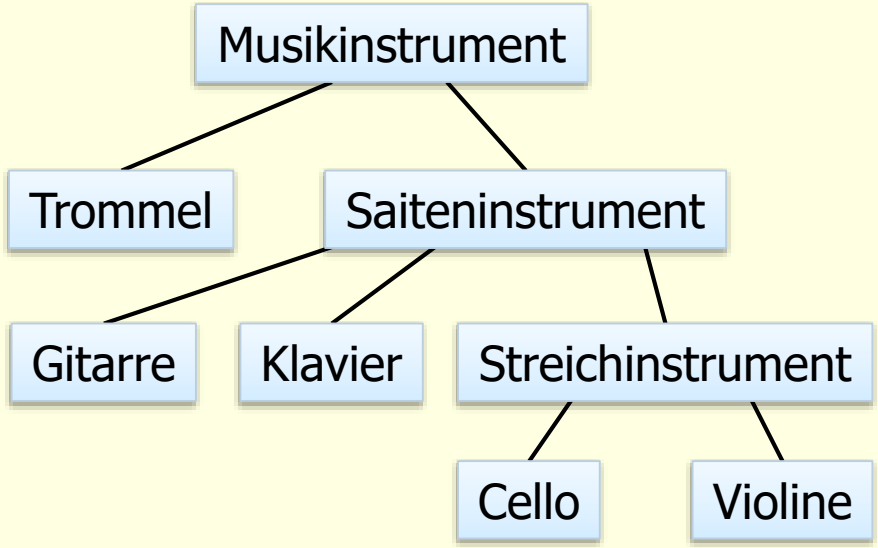
Le discours préliminaire des Éditeurs (1751) est le premier livre de *l'Encyclopédie ou Dictionnaire raisonné des sciences, des arts et des métiers* éditée de 1751 à 1772 sous la direction de Diderot et D'Alembert. Le discours préliminaire a été écrit par d'Alembert pour décrire la structure des articles de l'Encyclopédie et leur philosophie. À la fin du livre, d'Alembert inclut une explication détaillée du système des connaissances humaines. Cela comprend *un tableau qui établit une complexe généalogie des connaissances et la façon dont l'homme les a réparties dans les domaines spécifiques.* Le graphique montre aussi une progression de la connaissance à travers les âges. [Wikipedia]

(Ausschnitt)

18) Konzeptbaum

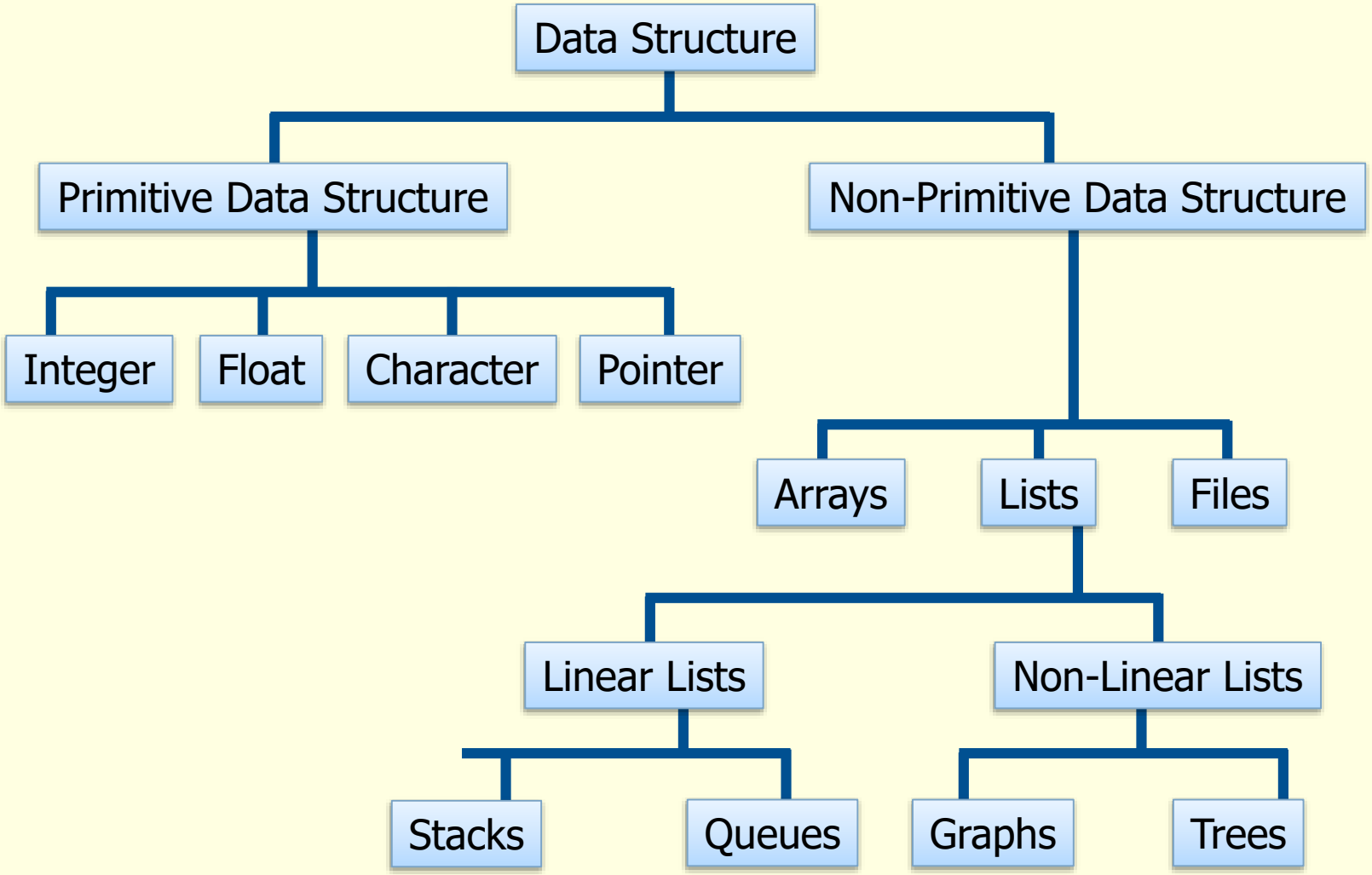
Ein Grundkonzept der objekt-orientierten Programmierung!





- Ein VW *ist ein* Personenwagen *ist ein* Auto *ist ein* Fahrzeug
- Eine Violine *ist ein* Streichinstrument *ist ein* Musikinstrument
- Ein Auto hat Räder ⇒ ein VW hat Räder

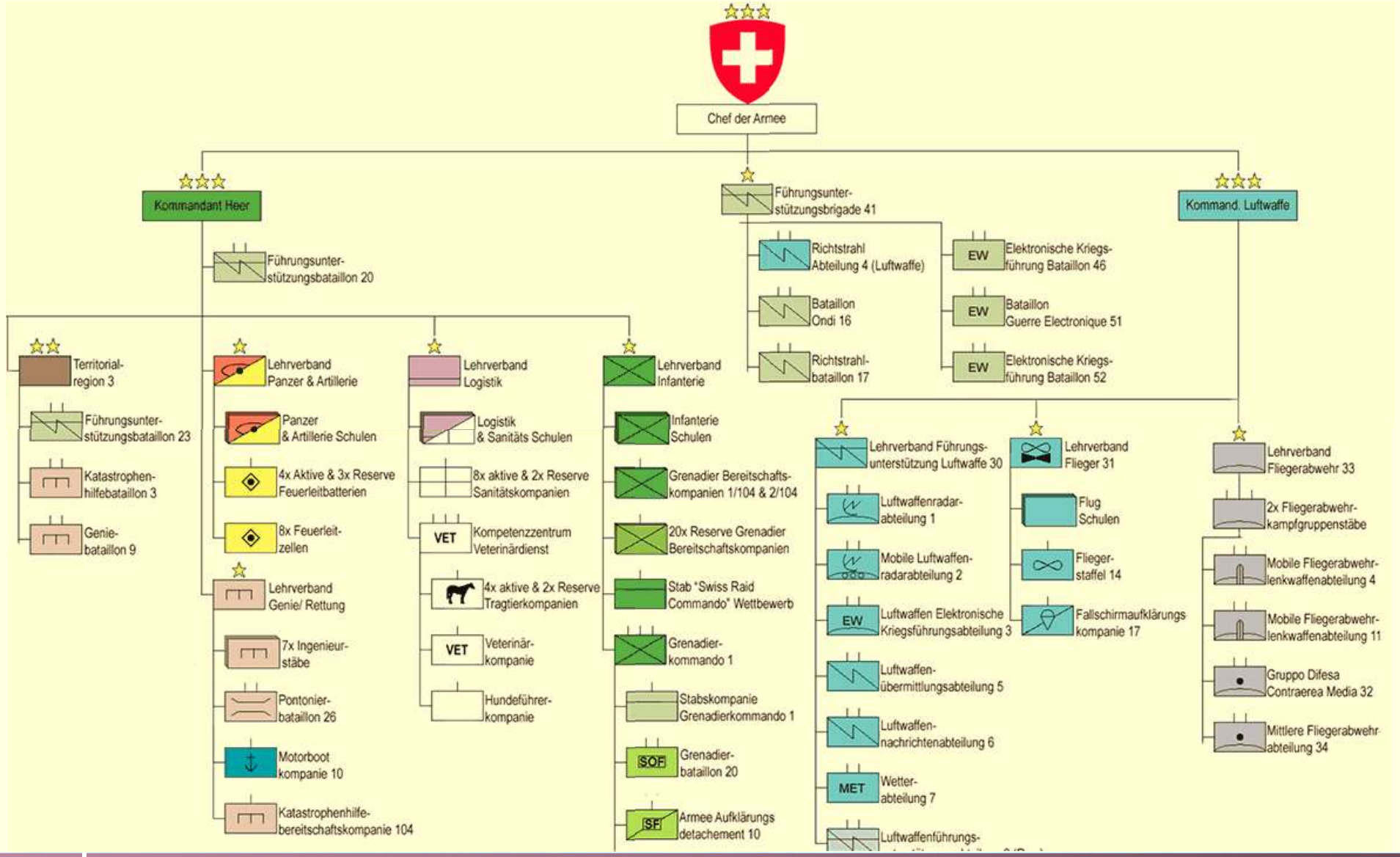
19) Klassifikation von Datenstrukturen



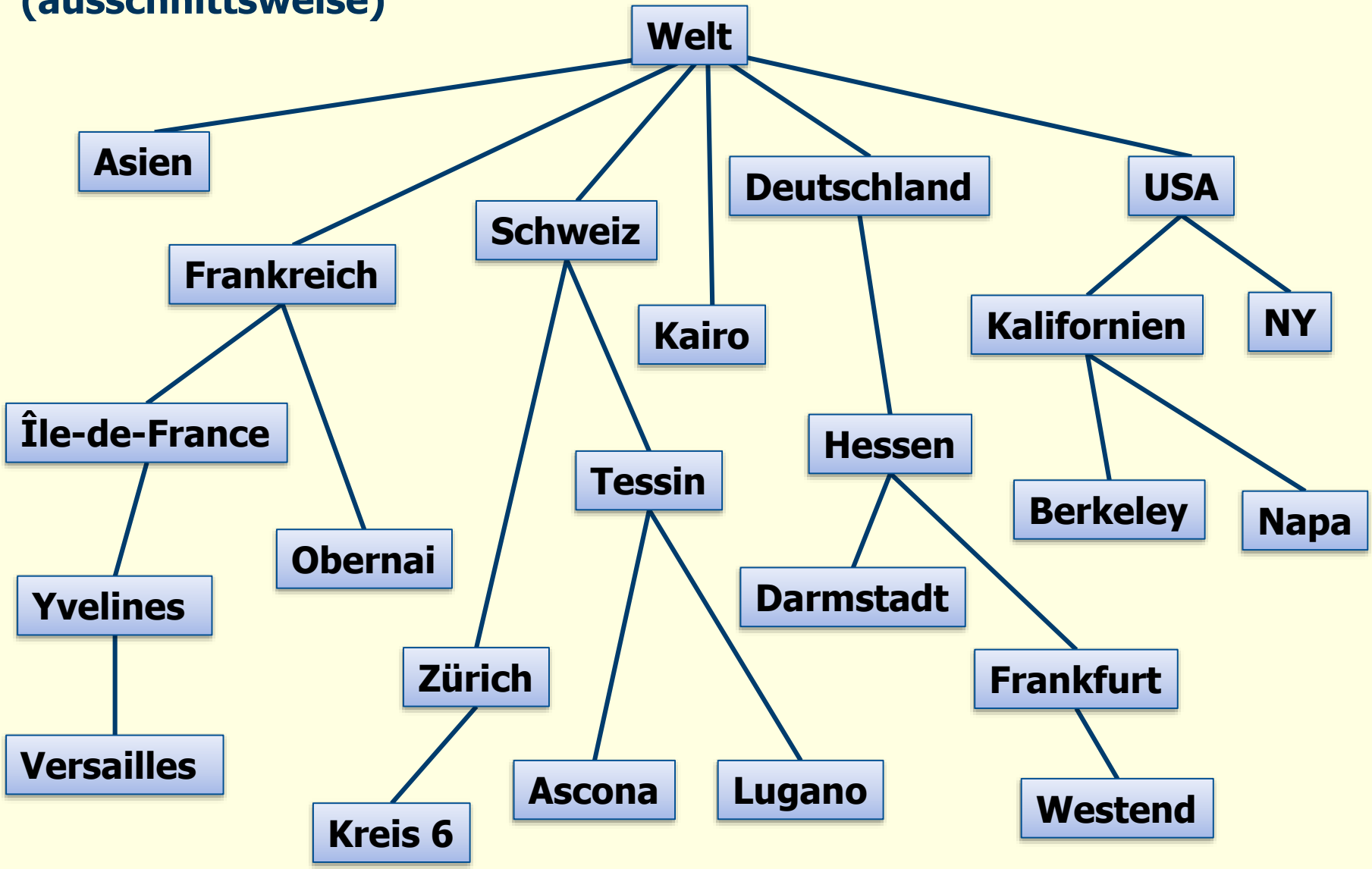
20) Dezimalklassifikation

Ein Bild aus dem Buch „Die Welt-Registratur“ von Karl Wilhelm Bührer und Adolf Saager (1912) als ein historisches Beispiel für ein hierarchisches Dezimalklassifikationssystem für Begriffe und Wissensgebiete. Der amerikanische Bibliothekar Melvil Dewey entwickelte das Prinzip zur Dewey Decimal Classification (DDC) weiter, mit der viele Bibliotheksbestände, vor allem in den USA, klassifiziert werden. Dabei steht 004 z.B. für die gesamte Kategorie „Informatik“; Backtracking findet man dann unter 004.0151; 004.015113 steht für mathematische Logik in der Informatik; 004.082 für Computer und Frauen; 004.652 für Peer-to-Peer-Computing; 700 ist die Oberkategorie für Künste und Unterhaltung; 735.22 steht für die Bildhauerkunst des 19. Jahrhunderts; 799.313 für Schiessen auf bewegliche Ziele und 799.3132092 speziell für Wurftaubenschützen.

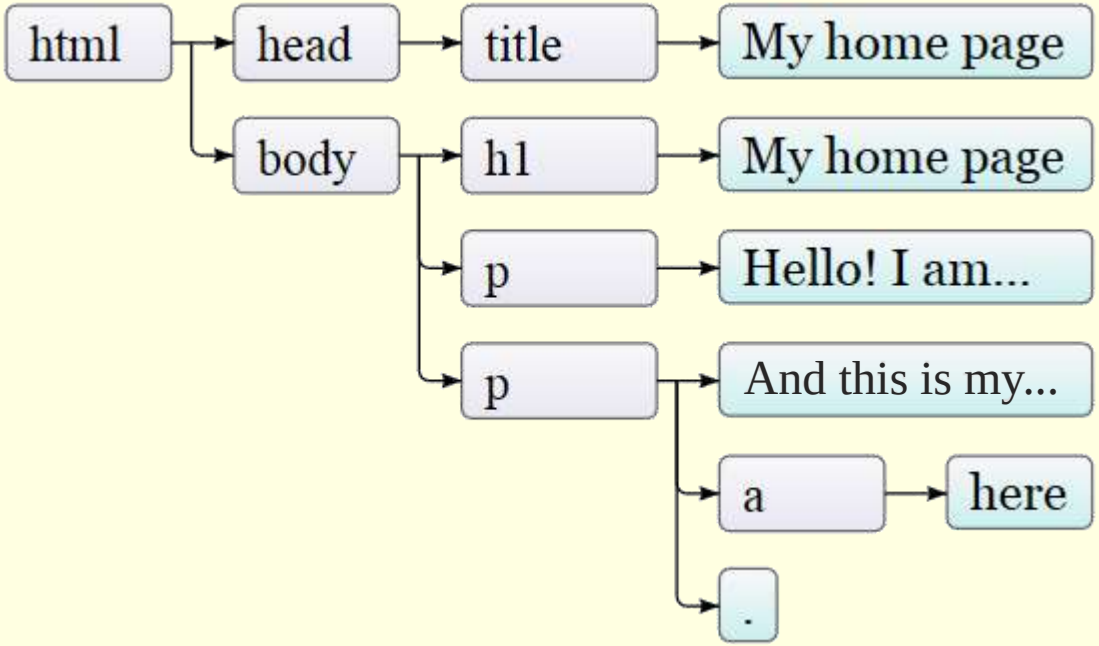
21) Gliederung der Schweizer Armee (Ausschnitt)



22) Die Welt
(ausschnittsweise)



23) HTML-Dokumentenstruktur




```
<!doctype html>
<html>
 <head>
  <title>My home page</title>
 </head>
 <body>
  <h1>My home page</h1>
  <p>Hello! I am … .</p>
  <p>And this is my new book:
   <a href="http://....net">
    here</a>.</p>
 </body>
</html>
```

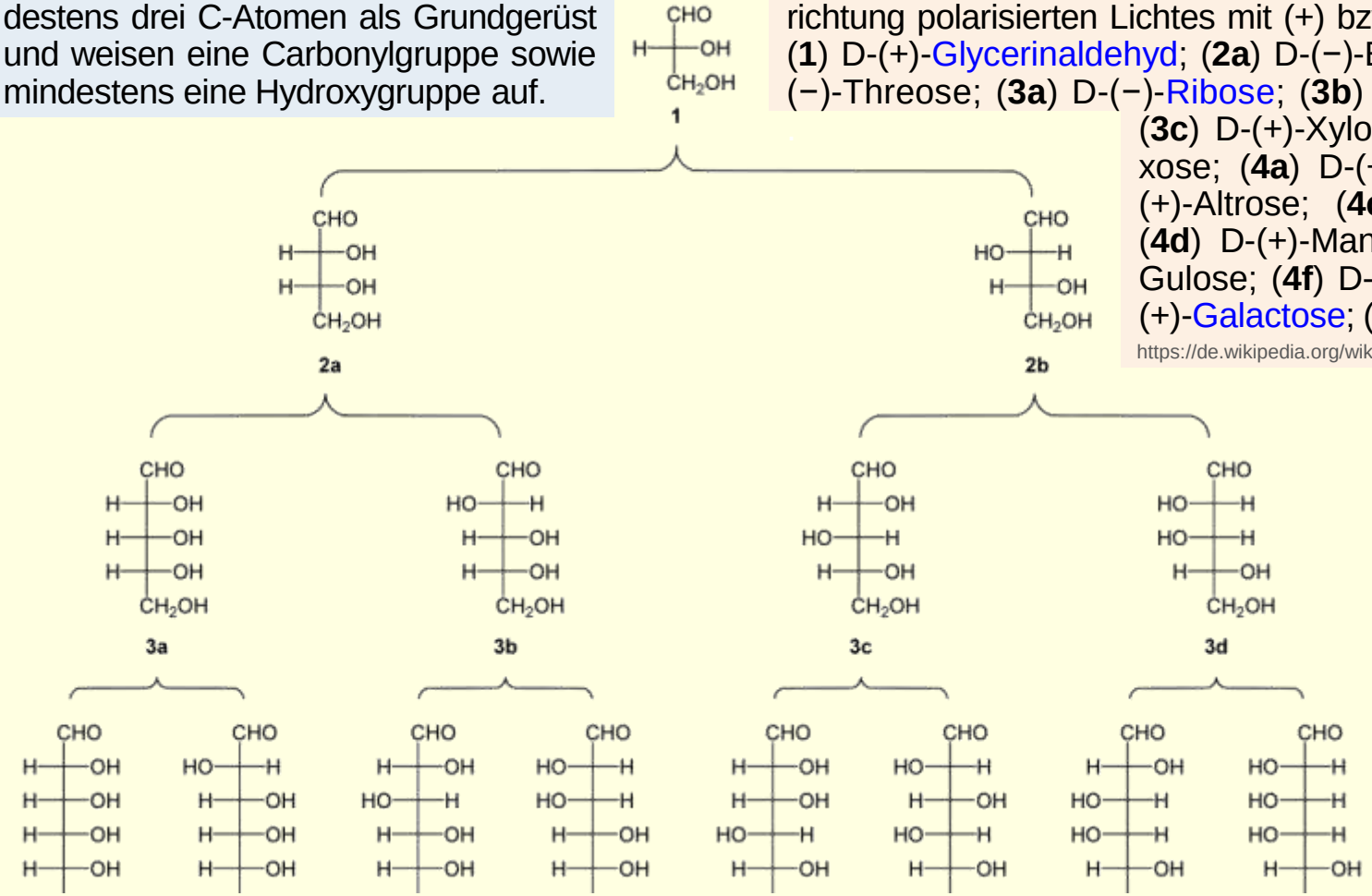
Die Hypertext Markup Language (HTML) ist eine textbasierte Auszeichnungssprache zur Strukturierung digitaler Dokumente. HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt. HTML dient dazu, einen Text semantisch zu strukturieren, nicht aber zu formatieren. (Die visuelle Darstellung ist nicht Teil der HTML-Spezifikationen, sondern wird durch den Webbrowser sowie Gestaltungsvorlagen wie CSS bestimmt.)

24) Monosaccharide

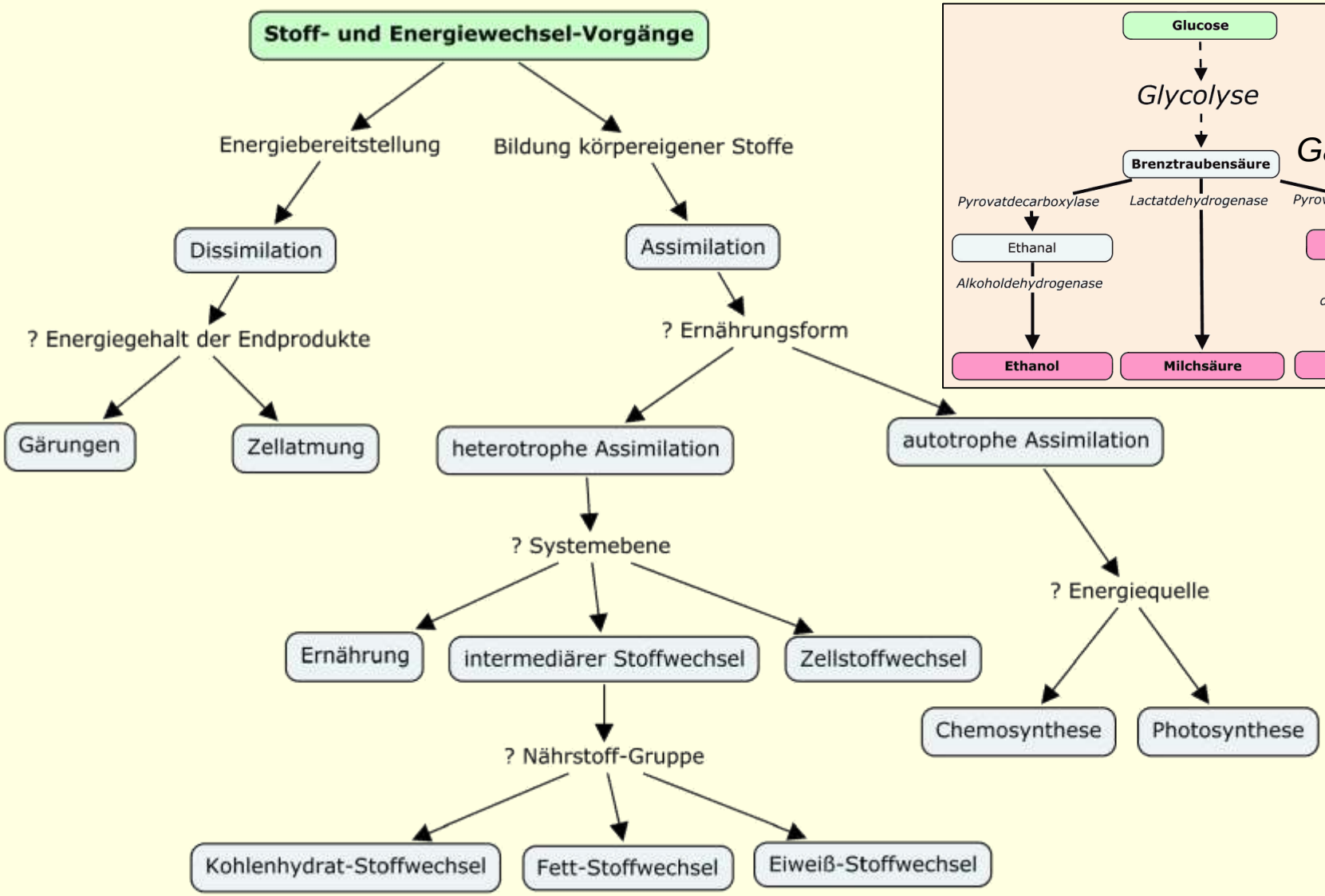
Monosaccharide sind die Produkte der partiellen Oxidation von Alkoholen; sie stellen die Bausteine aller Kohlenhydrate dar. Sie besitzen eine Kette aus mindestens drei C-Atomen als Grundgerüst und weisen eine Carbonylgruppe sowie mindestens eine Hydroxygruppe auf.

Stammbaum der D-Aldosen. Durch Anhängen von CH–OH-Gruppen verlängert man das Grundgerüst, so dass sich weitere Zucker ableiten lassen (von Triosen mit drei C- bis Hexosen mit sechs C-Atomen). Dabei ist die Drehrichtung polarisierten Lichtes mit (+) bzw. (−) angegeben. (**1**) D-(+)-Glycerinaldehyd; (**2a**) D-(−)-Erythrose; (**2b**) D-(−)-Threose; (**3a**) D-(−)-Ribose; (**3b**) D-(−)-Arabinose; (**3c**) D-(+)-Xylose; (**3d**) D-(−)-Lyxose; (**4a**) D-(+)-Allose; (**4b**) D-(+)-Altrose; (**4c**) D-(+)-Glucose; (**4d**) D-(+)-Mannose; (**4e**) D-(−)-Gulose; (**4f**) D-(−)-Idose; (**4g**) D-(+)-Galactose; (**4h**) D-(+)-Talose.

https://de.wikipedia.org/wiki/Monosaccharide

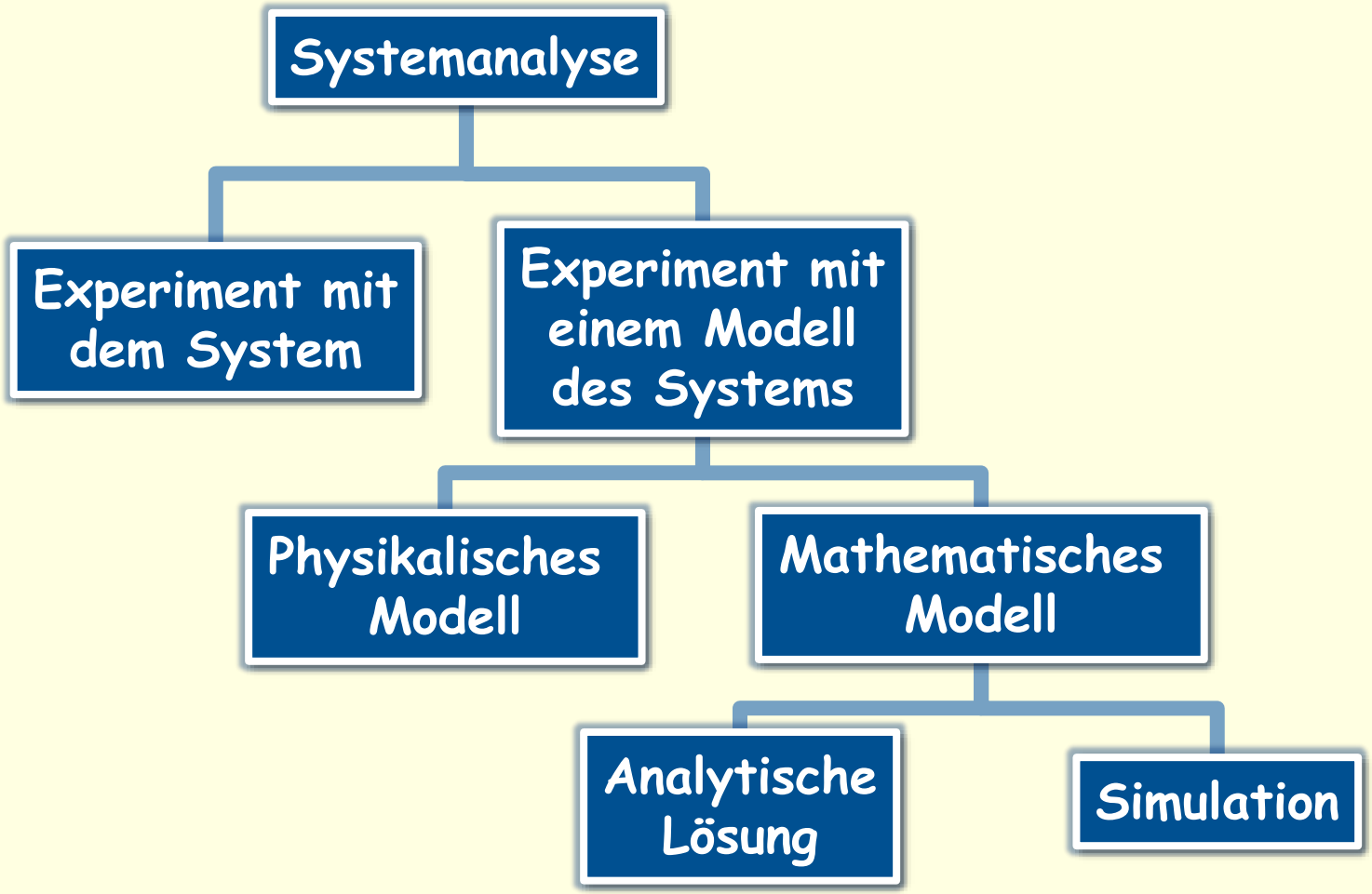


25) Klassifikation der Stoffwechselvorgänge



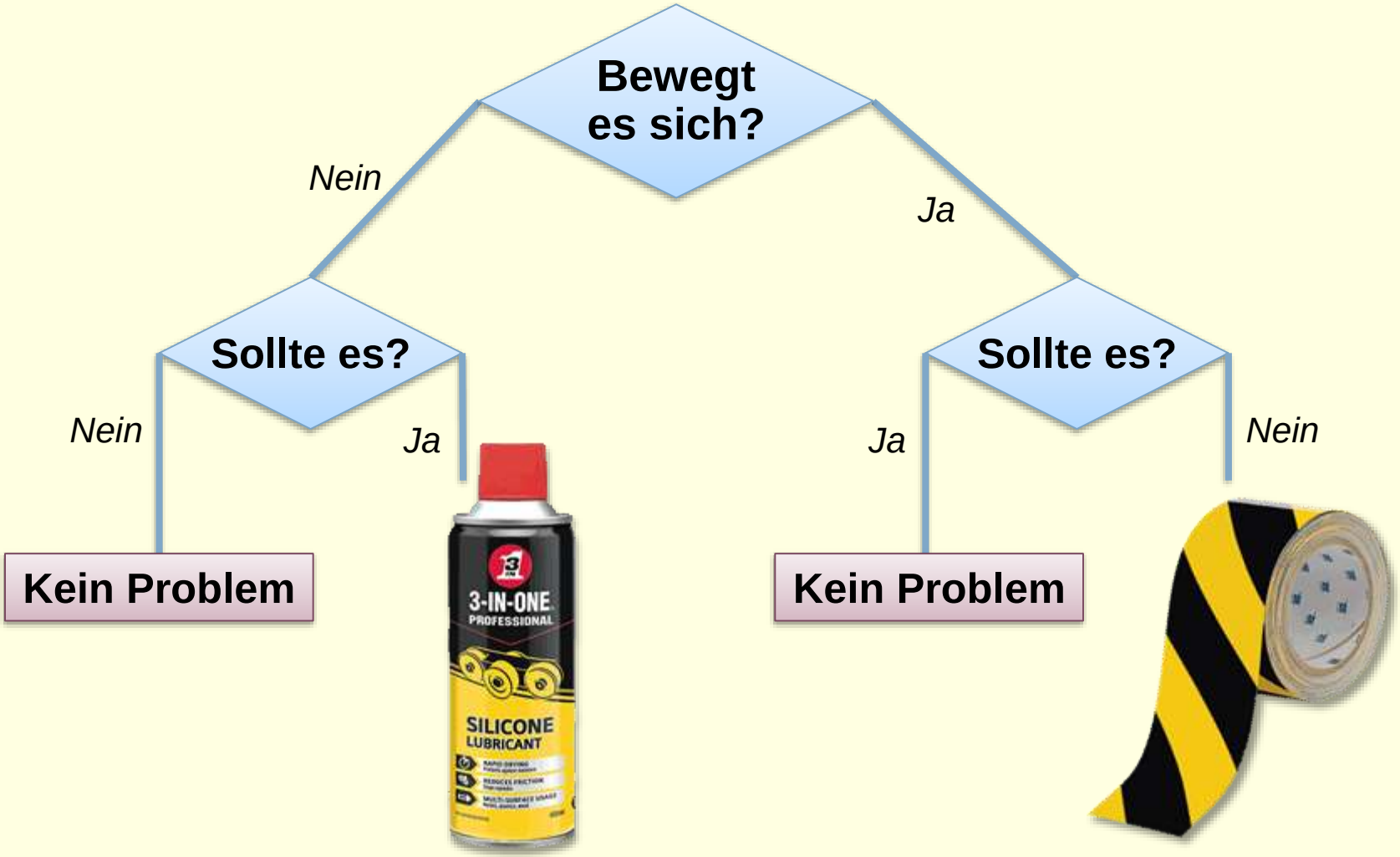
www.lern-soft-projekt.de/Medien/NaWi/BK_EL_Sek.II_SEW_2010_.pdf

26) Methoden der Systemanalyse

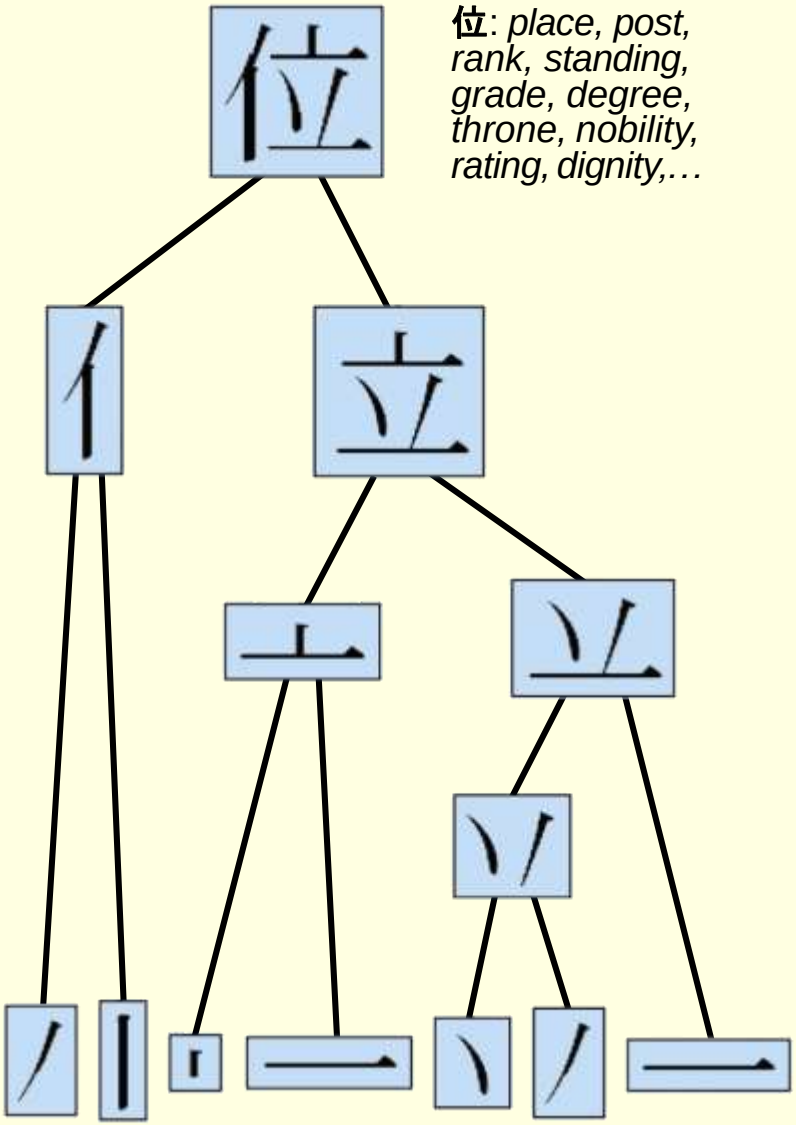


27) Entscheidungsbaum

Entscheidungsbaum für Techniker



28) Struktur von Kanji-Zeichen



位: *place, post, rank, standing, grade, degree, throne, nobility, rating, dignity,…*

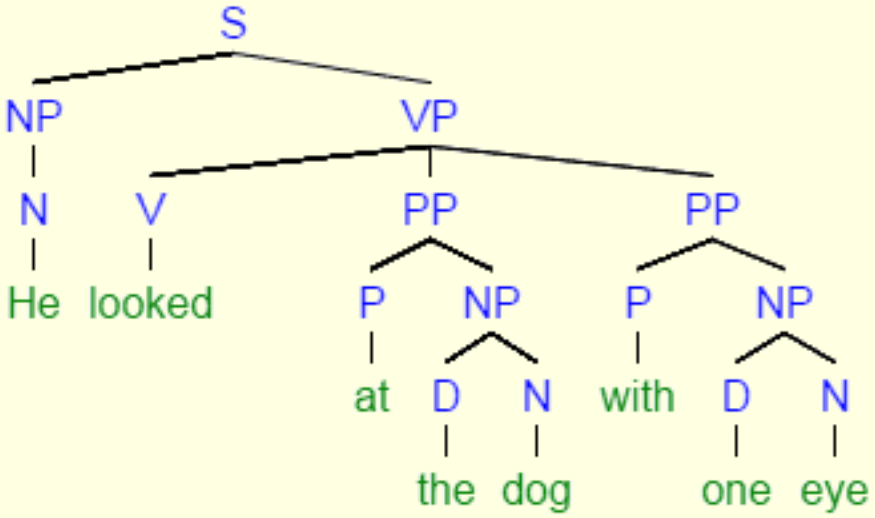
A new approach to online recognition of handwritten Kanji characters focusing on their hierarchical structure: A stochastic context-free grammar is introduced to represent the Kanji character generating process in combination with Hidden Markov Models (HMM) representing Kanji sub-strokes and to improve the recognition accuracy of important and frequently used Kanji characters in which inter-stroke relative positions play important roles. Combining the stroke likelihood and the relative-position likelihood between character-parts in the parsing process is expected to compensate their ambiguities.

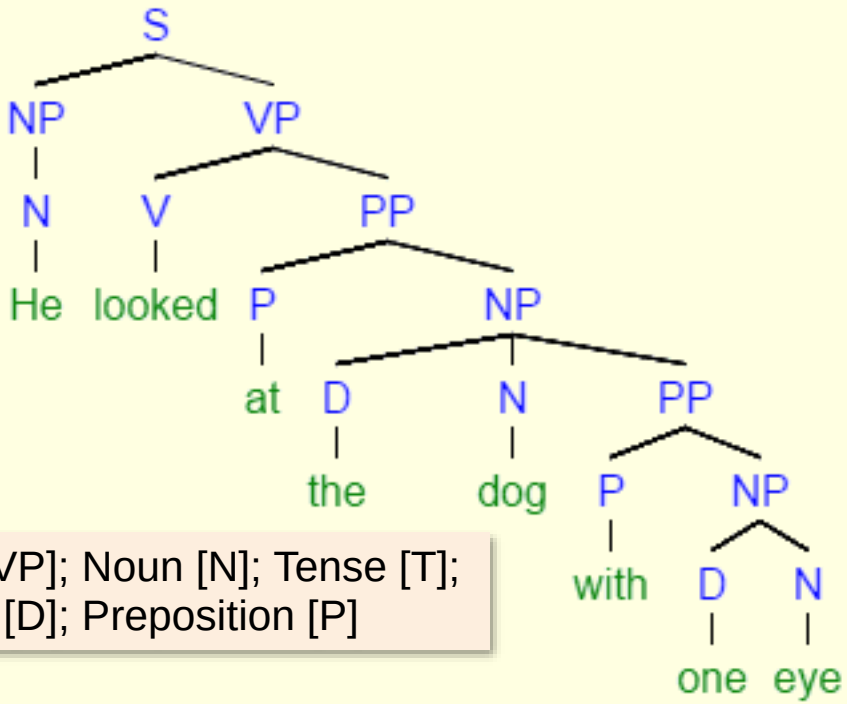
Kanji are composed of some character-parts, i.e., combination of two or more strokes. For example, the Kanji “位” consists of character-parts “イ” and “立”, themselves composed of smaller character-parts which can finally be divided into strokes.

[Ota, I., Yamamoto, R., Sako, S., Sagayama, S. (2007): Online handwritten kanji recognition based on inter-stroke grammar. ICDAR 2007, V. 2,1188-1192]

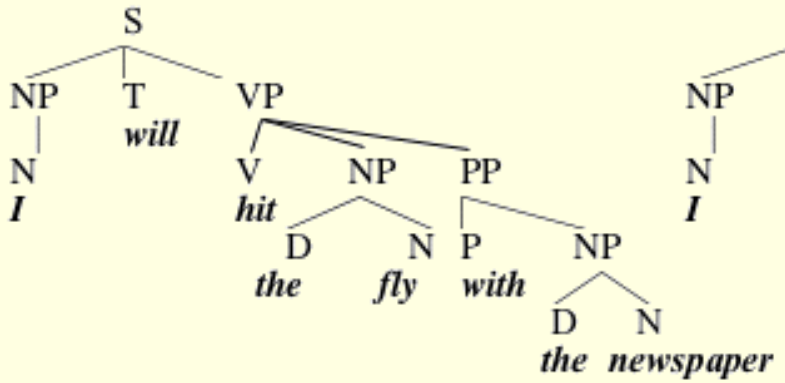
29) Syntaxbäume

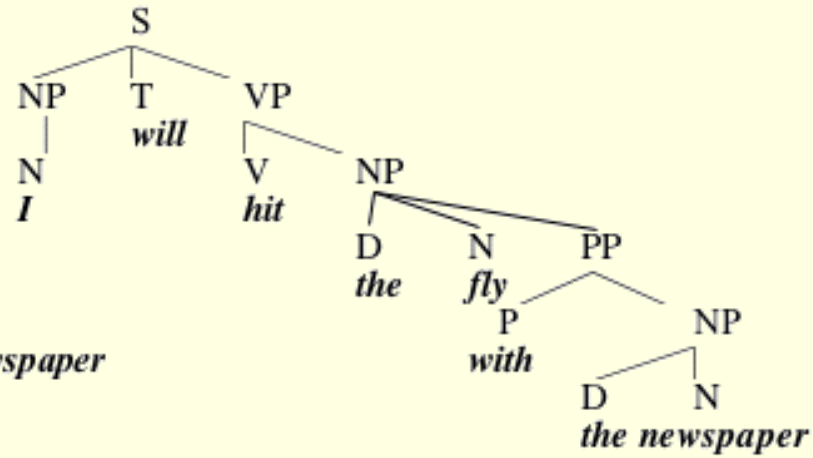
definieren die bedeutungsbestimmende Satzstruktur





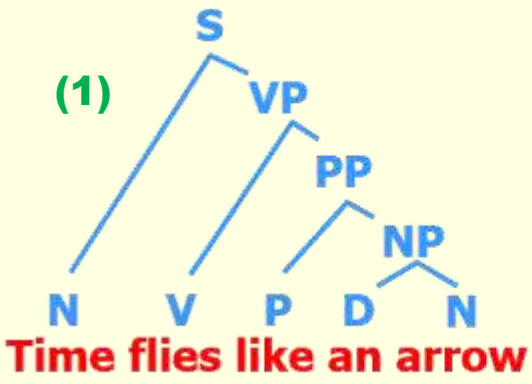
Sentence [S]; Noun Phrase [NP]; Verb Phrase [VP]; Noun [N]; Tense [T]; Verb [V]; Prepositional Phrase [PP]; Determiner [D]; Preposition [P]

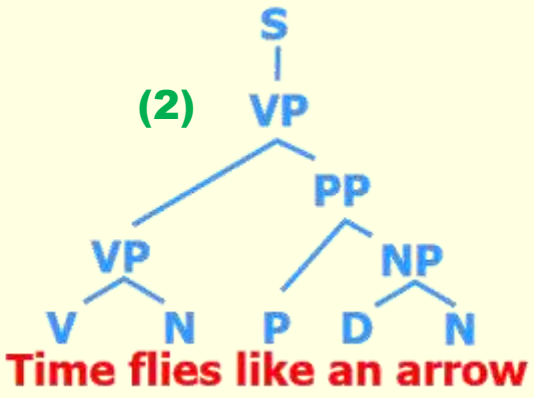


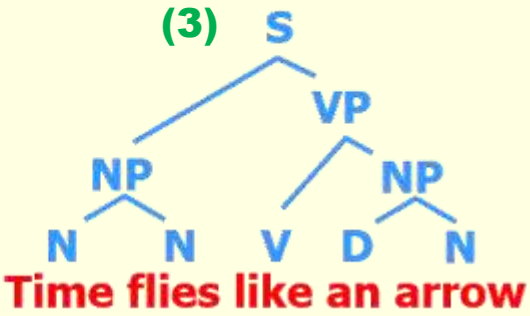


Stichwort „Mehrdeutigkeit“

- I read a book about evolution in 10 minutes.
- I read a book about evolution in the last million years.







Der bekannte Satz „Times flies like an arrow“ ist ein besonders interessantes Beispiel, da sowohl „time“ als auch „flies“ Substantiv oder aber Verb sein kann und „like“ entweder als Verb oder als Präposition fungieren kann. Wenn der Satz im Sinne von „Tempus fugit“ gemeint ist, dann trifft die Phrasenstruktur (1) zu. Wäre er im Sinne von „Fruit flies like a banana“ gemeint, dann wäre (3) die korrekte Struktur; bei „Time races with a stopwatch!“ hingegen (2). Im Prinzip könnte „like“ auch ein Substantiv sein („She has many *likes* and dislikes about her job where she is helping Facebook users to get more *likes* on pages“); im vorliegenden Kontext erscheint das allerdings nicht sinnvoll. Immerhin fanden Tüftler vereint mit Experten elf verschiedene Interpretationen des Satzes!

Stichwort „Mehrdeutigkeit“

Solche syntaktischen Mehrdeutigkeiten gibt es natürlich auch im Deutschen:

„Er trifft Männer mit Pistolen.“

Ist „trifft mit Pistolen“ oder „Männer mit Pistolen“ gemeint? Speziell im Deutschen gibt es manchmal auch Mehrdeutigkeiten bzgl. Subjekt und Objekt eines Satzes, die einem oft nicht bewusst werden, weil man aufgrund der unmittelbar mitschwingenden Satz- und Wortbedeutung alternative syntaktische Gliederungen meist gar nicht bewusst erwägt:

„Saftige Steaks machen auch viele Studentinnen an.“

Eine andere Form von Mehrdeutigkeit, die z.B. automatischen Übersetzungssystemen sehr zu schaffen machen, sind semantischer Art, zu deren Disambiguierung man mehr Kontext und manchmal auch „Weltwissen“ benötigt. Wie ist das Relativpronomen „sie“ hier zuzuordnen bzw. ins Französische zu übersetzen – mit „ils“ oder „elles“?:

„Die Buben wollten die Mädchen ins Kino einladen, aber sie hatten kein {Geld / Interesse}.“

„Google Translate“ wählt in beiden Fällen „ils“. Bei „we gave the monkeys the bananas – they were {ripe / hungry}“ wird die erste Alternative korrekt übersetzt – “nous avons donné aux singes les bananes – elles étaient mûres”; bei der zweiten werden aber als Seiteneffekt Bananen zu Affen gewandelt – „nous avons donné les singes aux singes – ils avaient faim“.

Gelegentlich ist sogar der situative Kontext relevant – muss man hier bei der Übersetzung ins Englische „lunch“ oder „dinner“ verwenden?:

„Wir machen eine Pause – wollen wir zusammen Essen gehen?“

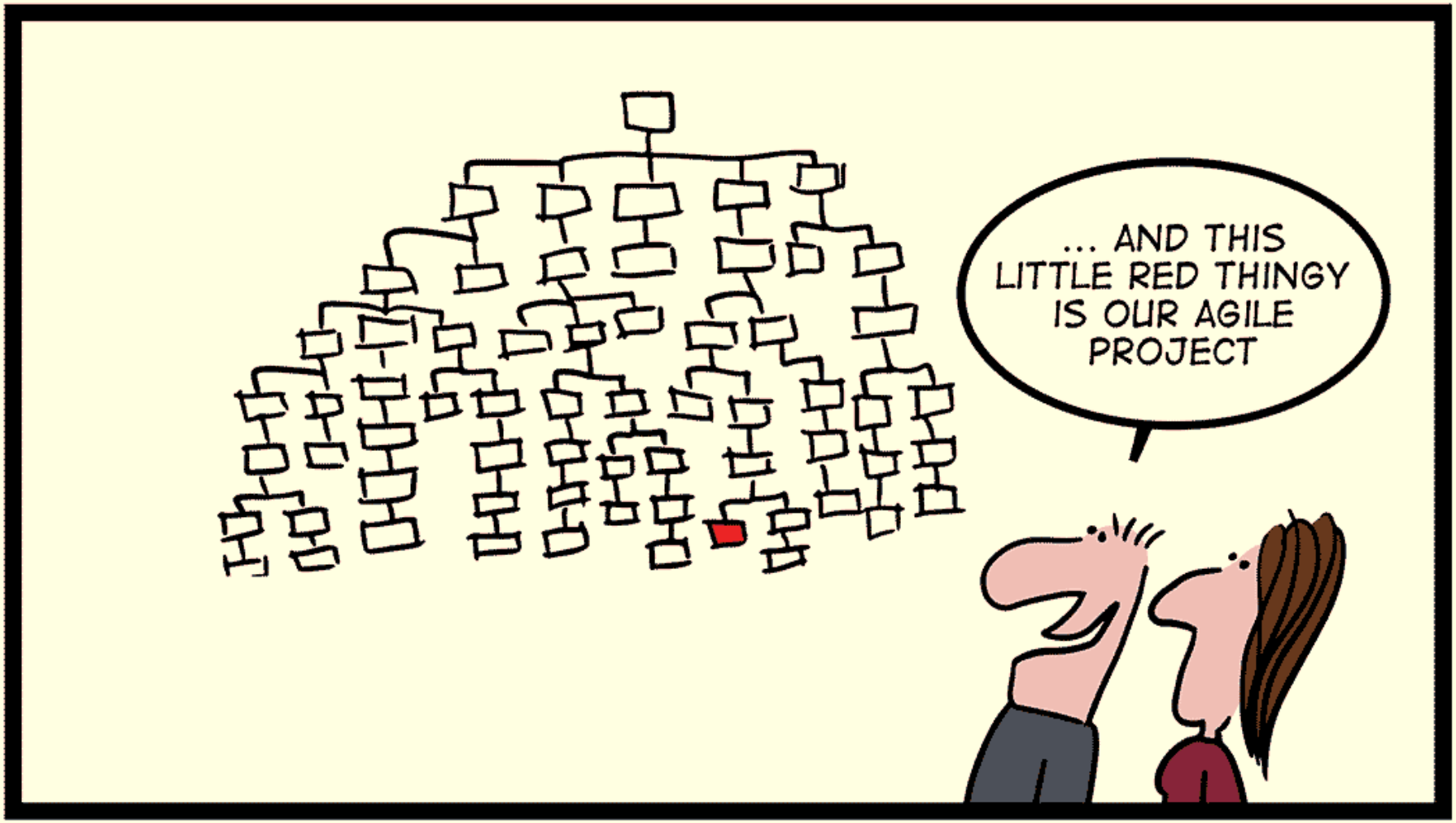
Und bei Sätzen mit Redewendungen und potentiellen Metaphern wie

„Wollen Sie mich auf den Arm nehmen?“

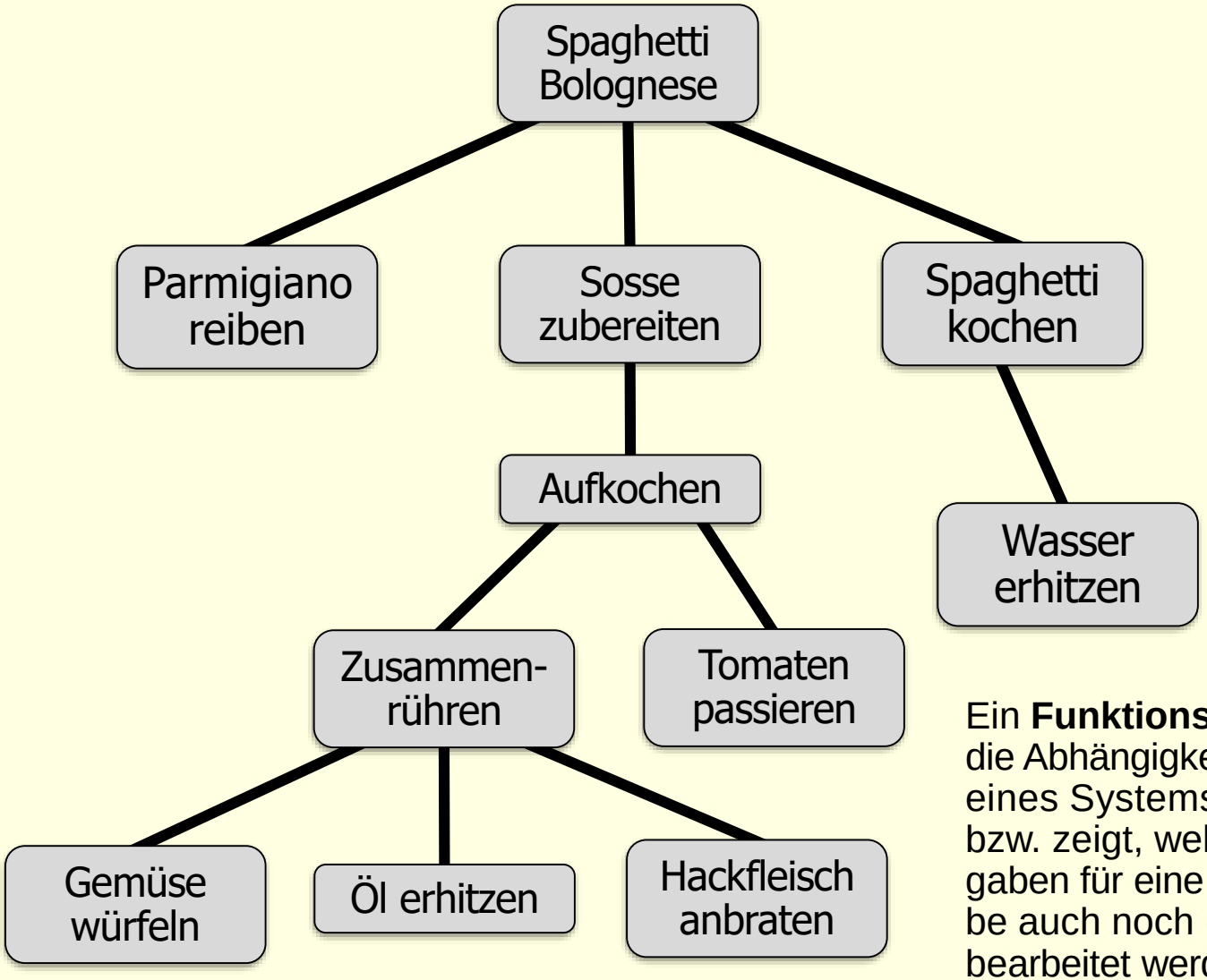
ist hinsichtlich der pragmatischen Ambivalenz besondere Vorsicht angebracht! Zu guter Letzt: Lehrer in Deutschland warnen vor Abschaffung des „ß“ und Ersatz durch „ss“ mit dem Beispiel

„Trinke Bier in Maßen!“

30) Projekthierarchie



31) Funktionsbaum



Ein **Funktionsbaum** beschreibt die Abhängigkeit von Funktionen eines Systems untereinander bzw. zeigt, welche anderen Aufgaben für eine bestimmte Aufgabe auch noch (evtl. vorgängig) bearbeitet werden müssen.

32) Ziele und Teilziele

„Wenn eine Aufgabe zu gross ist, macht man mehrere kleine Aufgaben daraus. Alles, was unangenehm oder undurchschaubar ist, klären wir durch Aufteilen und Gliedern. Teilziele und Aufgaben haben eine Struktur wie ein Baum.“

www.meineziele.info

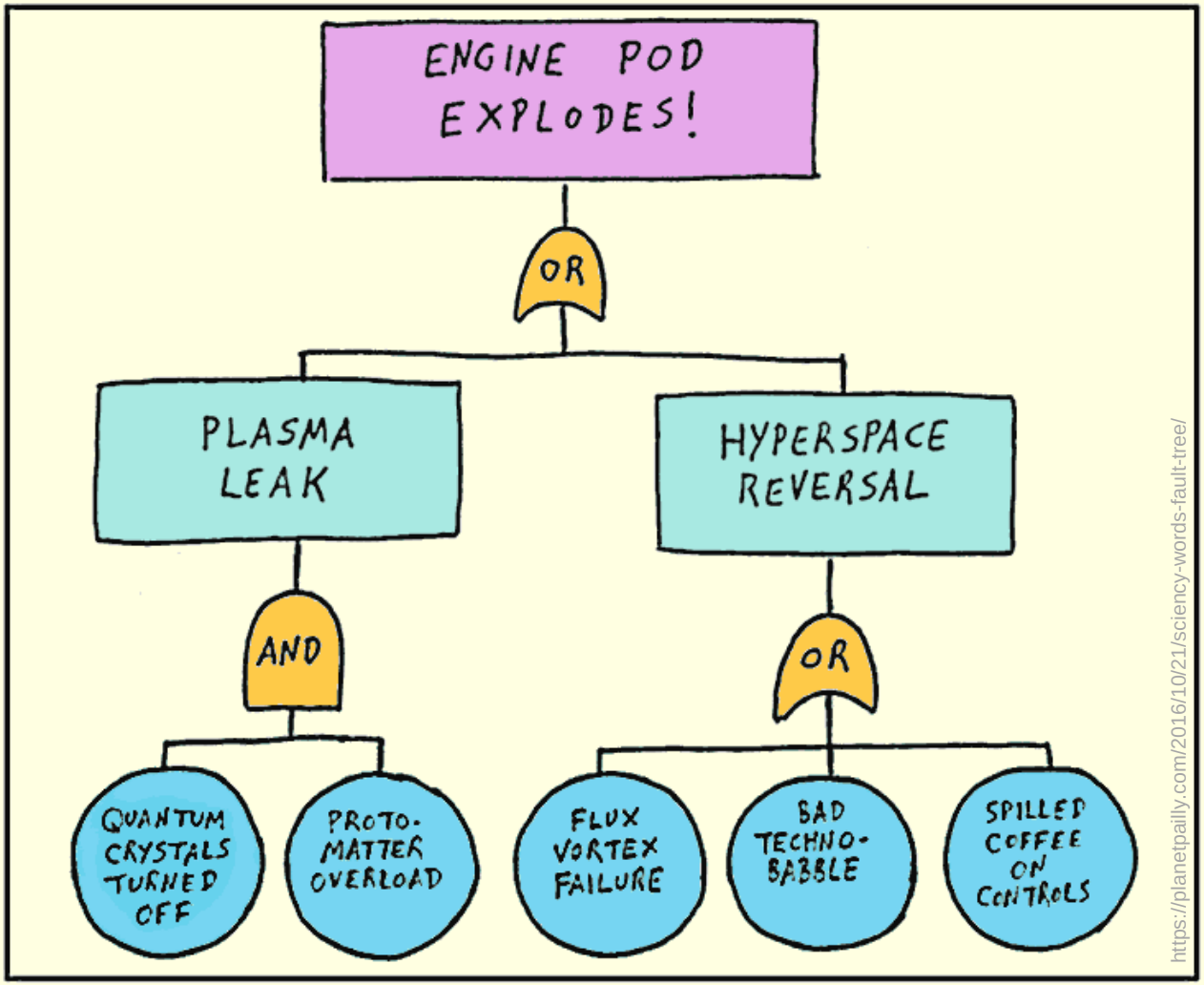
33) Spielbaum: Game over

Analyse einer Spielsituation in Bezug auf die Folgezüge durch Schachweltmeister Michail Botvinnik (1911 – 1995). Der Spielbaum hat 145 Knoten; Weiss ist am Zug und gewinnt beispielsweise so:

1. g4+ Kh4
2. Lh6 Dxh6
3. Dh2+ Kg5
4. Dd2+ Sf4
5. Dd8 matt.

Botvinnik veröffentlichte seine Studie 1925 in der Schachzeitschrift Schachmatny Listok (Шахматный листок). Er berichtet, dass dies eine aufbereitete Version einer Situation sei, in die er im Alter von 14 Jahren kurz zuvor in einem Spiel gegen N.M. Liutov in Leningrad geriet: „The end was so unexpected that for a moment my opponent did not notice that the game was over.“

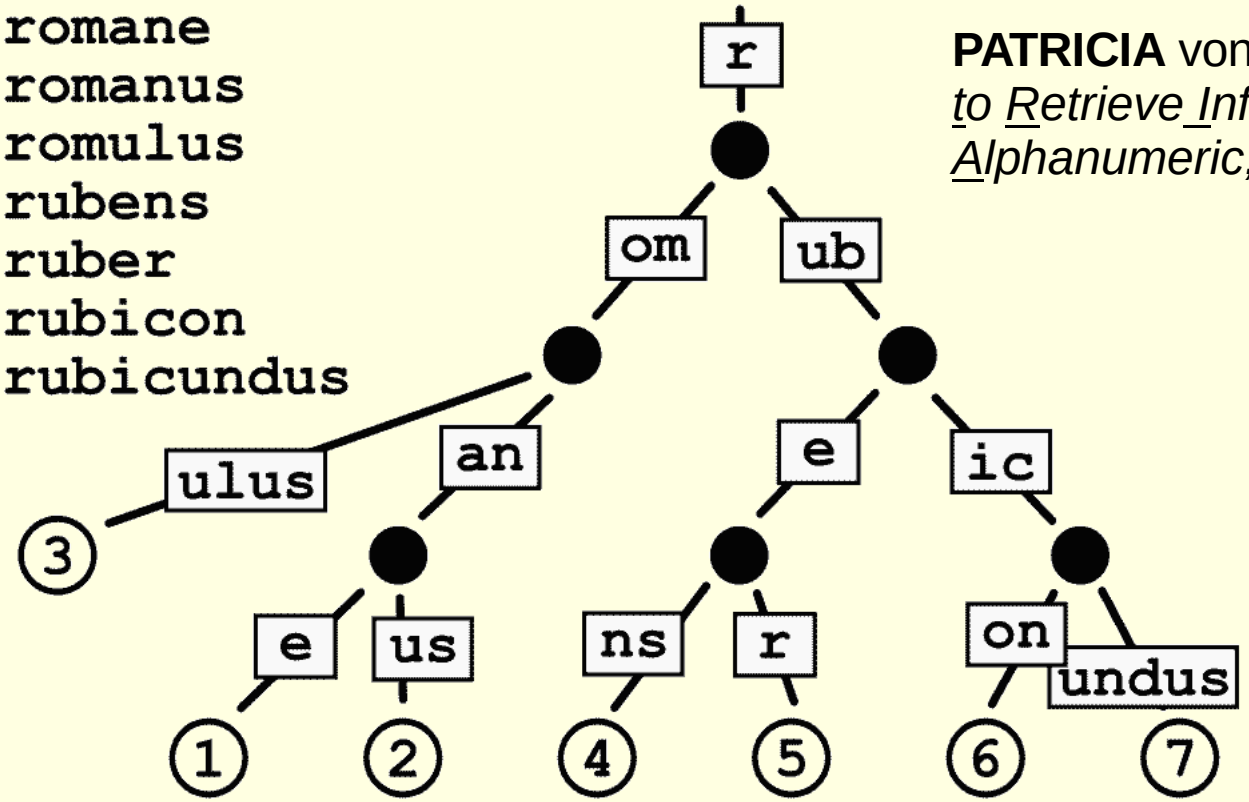
34) Fehlerbaum



Ein Und-Oder-Baum!

35) Patricia-Trie

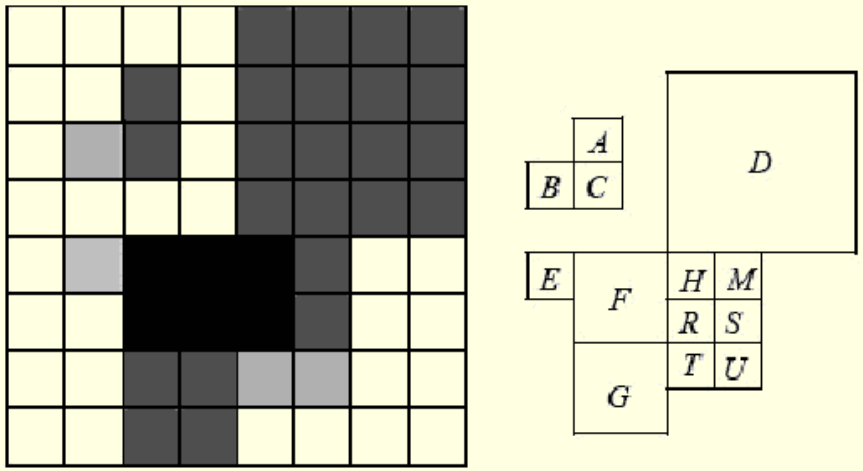
```
1 romane
2 romanus
3 romulus
4 rubens
5 ruber
6 rubicon
7 rubicundus
```



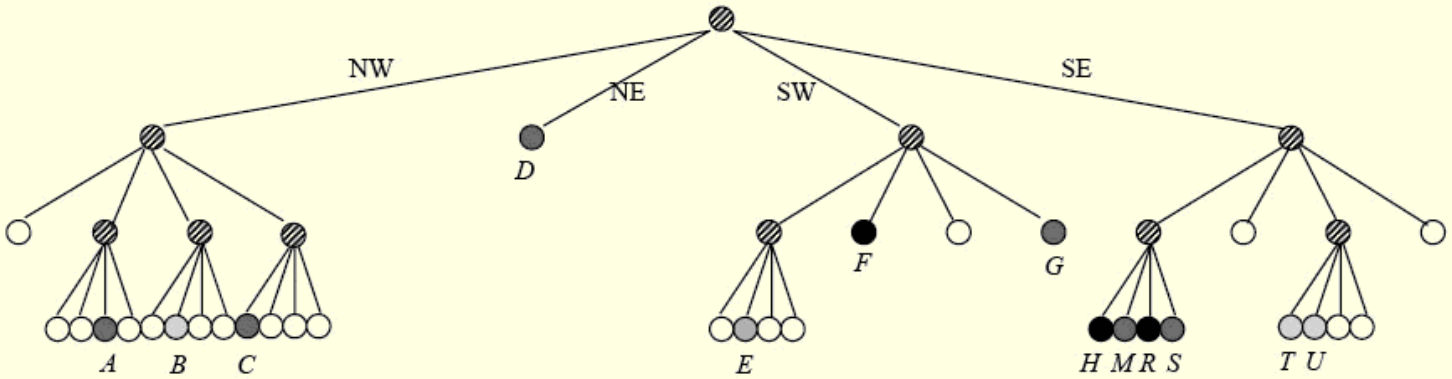
PATRICIA von *Practical Algorithm to Retrieve Information Coded in Alphanumeric;* **trie** von *Retrieval.*

Eine Datenstruktur zur gleichzeitigen Speicherung von mehreren Zeichenketten – eine Anwendung stellt z.B. die effiziente Repräsentation von IP-Adressen bei Internet-Routern dar.

36) Quadtrees

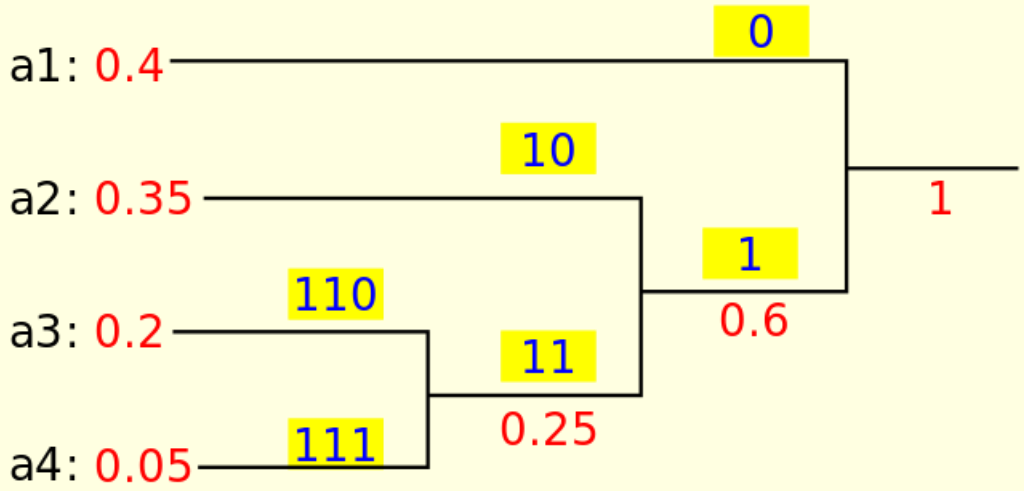


Jeder innere Knoten hat vier direkte Nachfolger. In rekursiver Weise wird damit ein zweidimensionaler Raum bis zu einer gewissen Tiefe in je vier Quadranten unterteilt. Es können so z.B. variabel aufgelöste Bilder repräsentiert werden oder Daten zu Teilgebieten gespeichert werden (z.B. die Durchschnittstemperatur auf einer Karte), wo grössere zusammenhängenden Teilgebieten der gleiche Wert zukommt.



37) Huffman-Codebaum

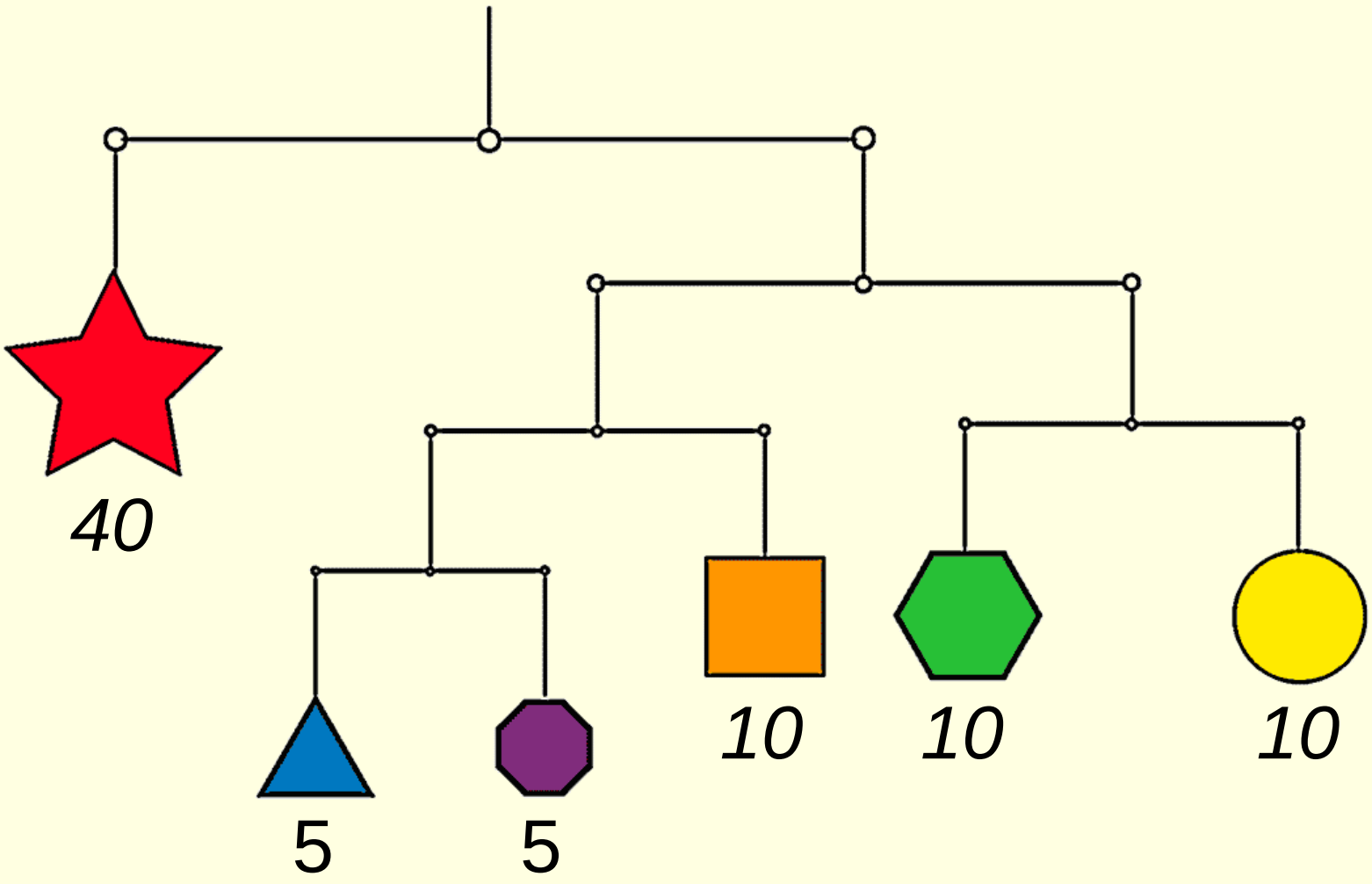
Die Huffman-Kodierung ordnet einer festen Anzahl an Quellsymbolen jeweils Codewörter mit variabler Länge zu. Häufiger auftauchende Zeichen werden dabei mit weniger Bits repräsentiert als seltener auftauchende, wobei kein Codewort der Beginn eines anderen ist (Präfixfreiheit).



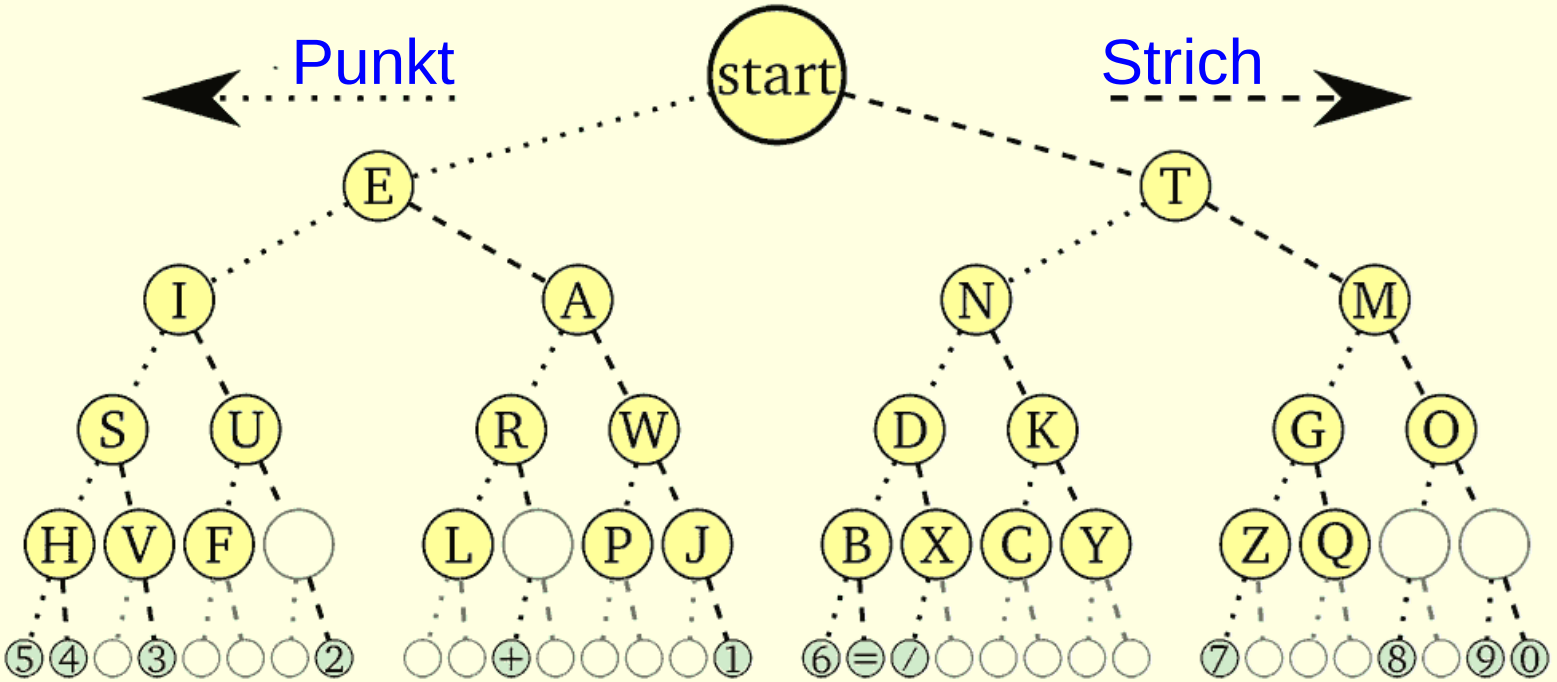
Symbol	Code
a1	0
a2	10
a3	110
a4	111

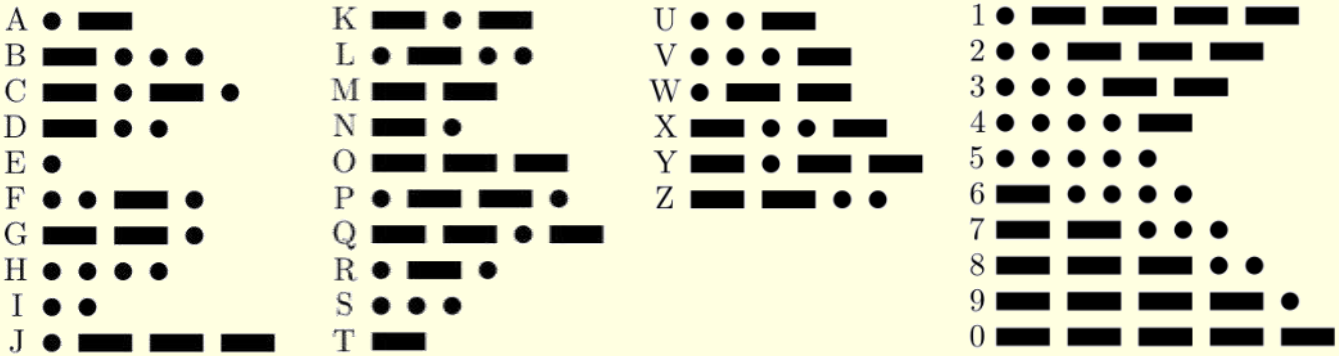
Beispiel: Eine Quelle generiert Symbole a1, a2, a3, a4 mit den jeweiligen Wahrscheinlichkeiten 0.4, 0.35, 0.2, 0.05. Die hier nicht näher beschriebene Konstruktion des Codebaums liefert die zugehörigen Codewörter 0, 10, 110, 111. Die Länge der Codewörter entspricht idealerweise ihrem Informationsgehalt. Im Beispiel beträgt die Entropie der Quelle 1.74 Bits/Symbol; die durchschnittliche Länge eines Codeworts beträgt hier 1.85 Bits/Symbol (statt 2 Bits/Symbol bei einer naiven Codierung jedes Symbols durch 2 Bits)

38) Mobile oder Huffmann-Codebaum?



39) Morsealphabet

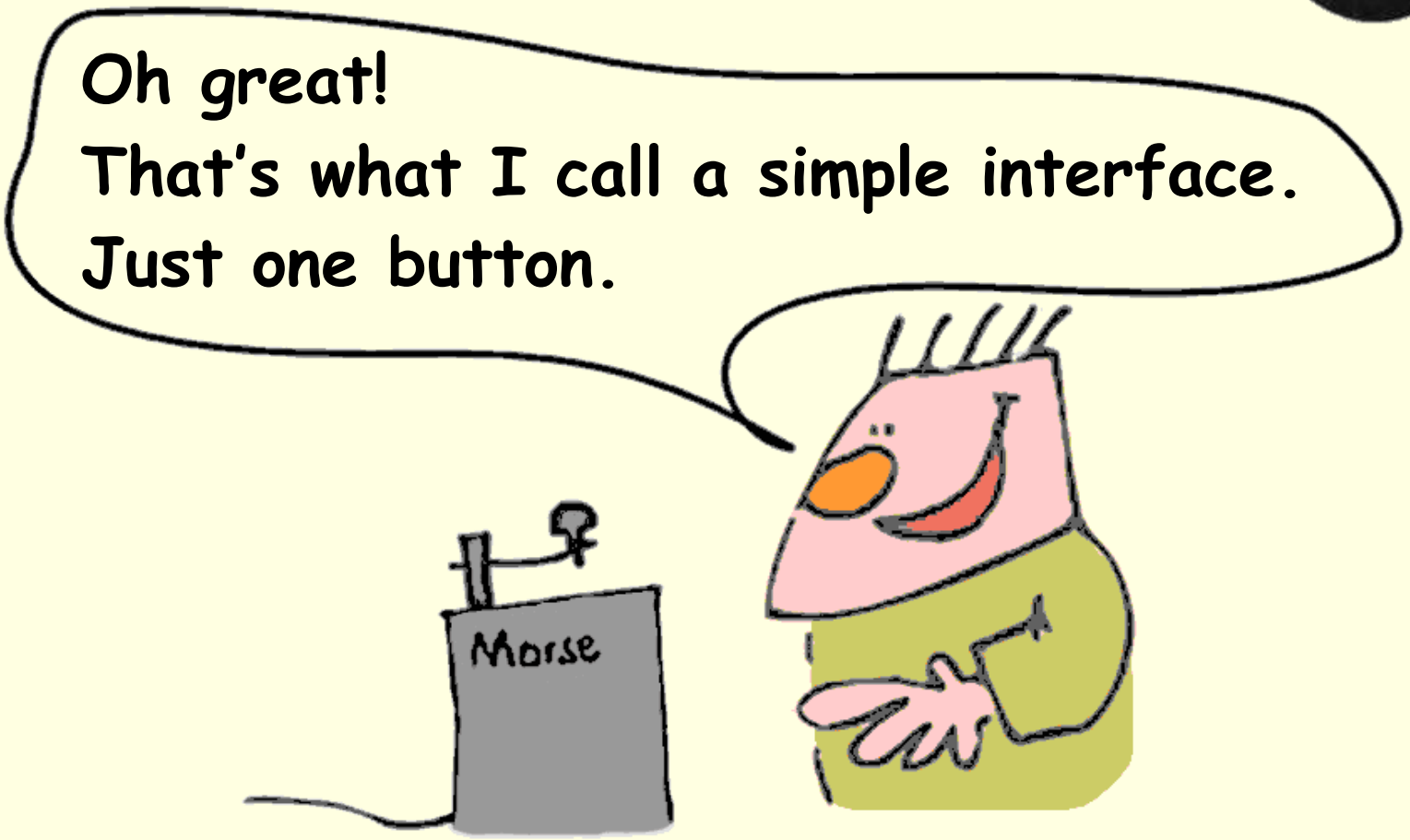




Pausen zwischen Buchstaben (und längere Pausen zwischen Wörtern) sind notwendig!

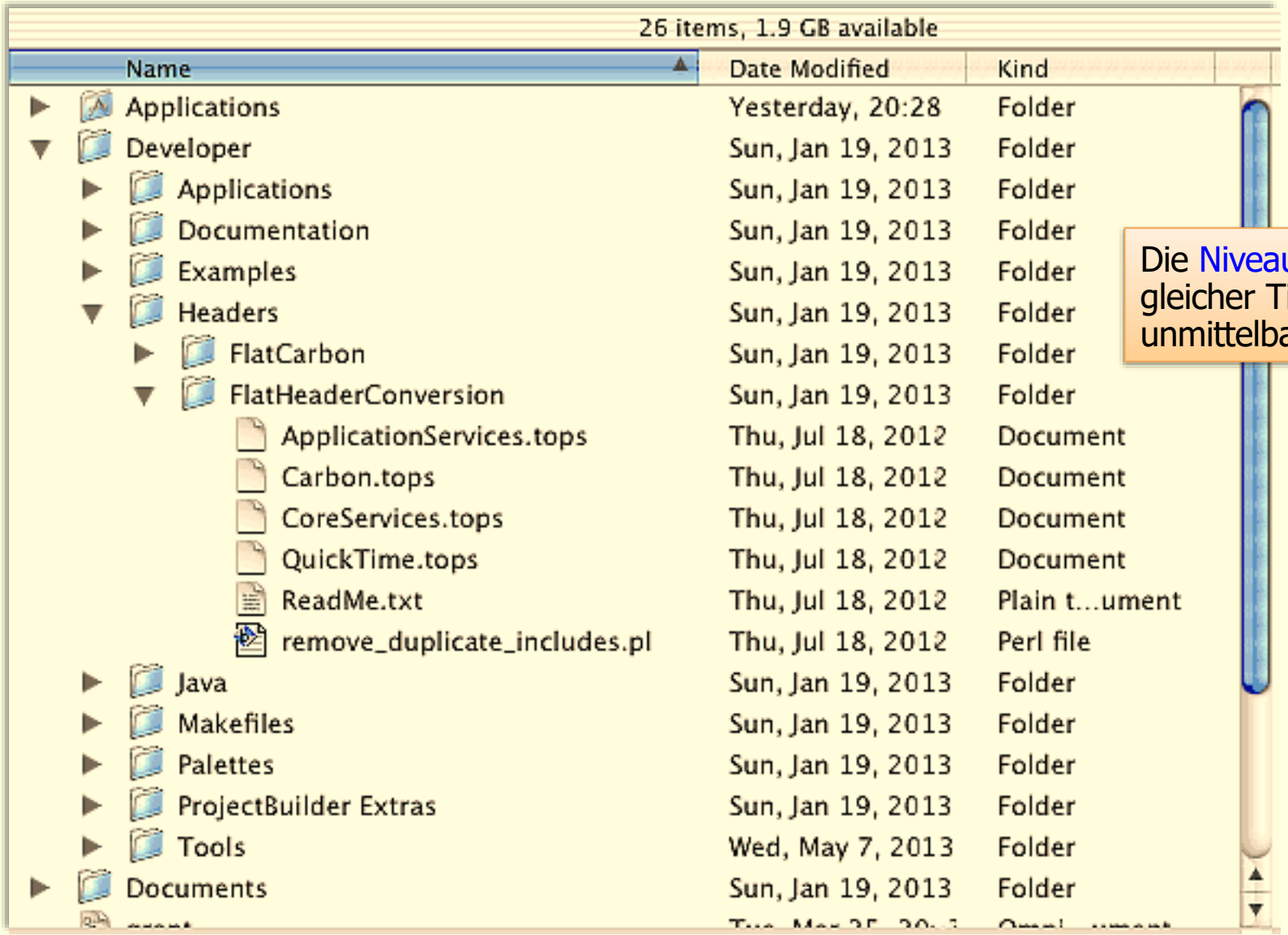
Stichwort „Morse“

Samuel Morse baute 1836 aus Drahtresten, Blechabfällen, einer Staffelei (Morse war Professor für Malerei) und seiner Wanduhr einen elektrischen Schreibtelegrafen. Die Nachrichtenübermittlung erfolgte drahtgebunden („Telegrafenleitungen“), dazu definierte er einen Code aus langen und kurzen Stromstössen. (Erst ab 1898 entwickelte G. Marconi die Funktelegrafie.)



http://geekandpoke.typepad.com/geekandpoke/images/morse_3.jpg

40) Dateihierarchie (Darstellung in eingerückter Form)



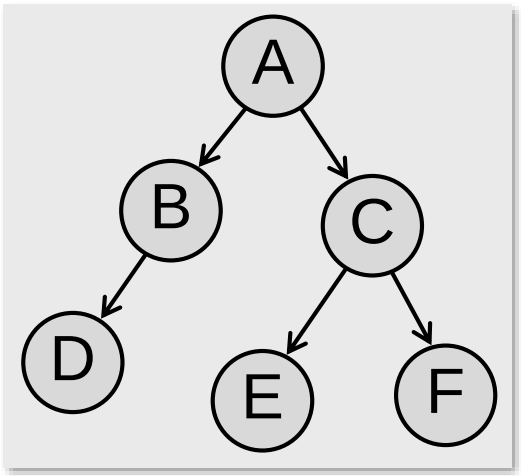
Die Niveaus (Knoten gleicher Tiefe) sind unmittelbar sichtbar

41) Textgliederung (Darstellung in eingerückter Form)

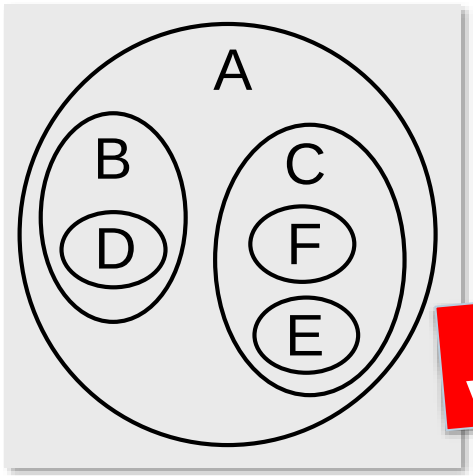
Noch mehr Bäume

- Einige der hier angerissenen Aspekte werden später vertieft, z.B.:
  - Syntaxbäume spielen eine Rolle bei der Analyse von Programmstrukturen und formelmässigen Konstrukten
  - Konzeptbäume dienen der Festlegung von begrifflichen Hierarchien im Rahmen der Objektorientierung
  - Spielbäume dienen Analyse strategischer und kombinatorischer Spiele
- Wir werden später auch noch andere Anwendungsbereiche sowie spezialisiertere Baumtypen kennenlernen, z.B.:
  - Operatorbäume für arithmetische Ausdrücke
  - Suchbäume zum effizienten Zugriff auf Daten mit Schlüsselwerten
  - Backtrack-Bäume zum systematischen Durchforsten grosser kombinatorischer Zustands- und Informationsräume

Darstellung von Wurzelbäumen



Als (gerichteter) Graph



Als Mengendiagramm
(zeigt „Verschachtelung“)

Alles sind nur unterschiedliche Darstellungen des gleichen „abstrakten“ Baums!

Man wähle die für den jeweiligen Zweck geeignetste!

In eingerückter Form („indentation“)

```
A
  B
    D
  C
    E
    F
```

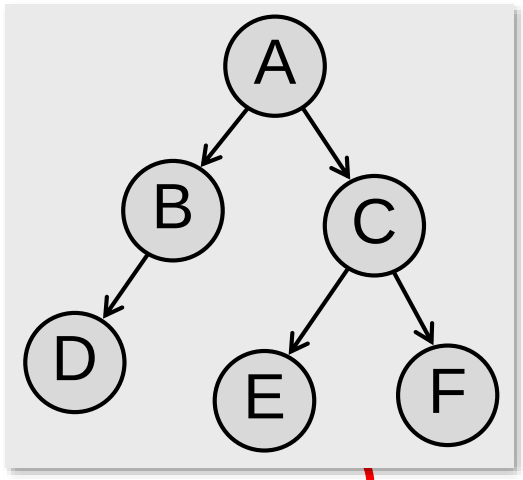
```
A ( B ( D ) , C ( E , F ) )
```

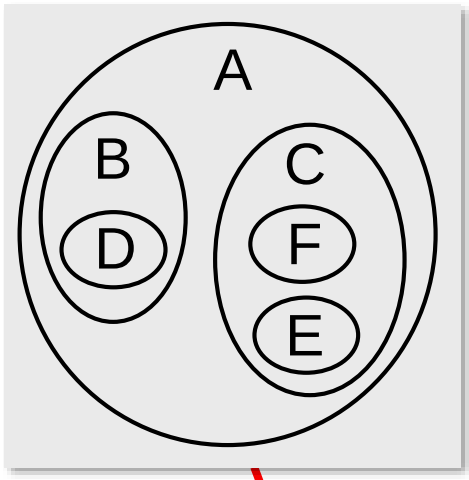
In Klammerdarstellung

Name der Wurzel, danach in Klammern die Unterbäume, jew. durch Komma getrennt

Lineare Darstellung ist sehr kompakt; geeignet z.B. zur Ein- / Ausgabe

Darstellung von Wurzelbäumen





Alles sind nur unterschiedliche Darstellungen des gleichen „abstrakten" Baums!

Mit jeweils genau der selben Bedeutung

Alle Darstellungen sind gleichwertig (→ „äquivalent")

Mengendiagramme können allerdings (im Unterschied zu den anderen Darstellungen) keine Ordnung auf Unterbäumen ausdrücken

Sind verschiedene Zeichen für den selben (abstrakten) Baum

Zum einzigen & wahren Wert

```
A
  B
    D
  C
    E
    F
```

```
A ( B ( D ) , C ( E , F ) )
```

Darstellung von ~~Wurzelbäumen~~

Alles sind nur unterschiedliche Darstellungen des gleichen

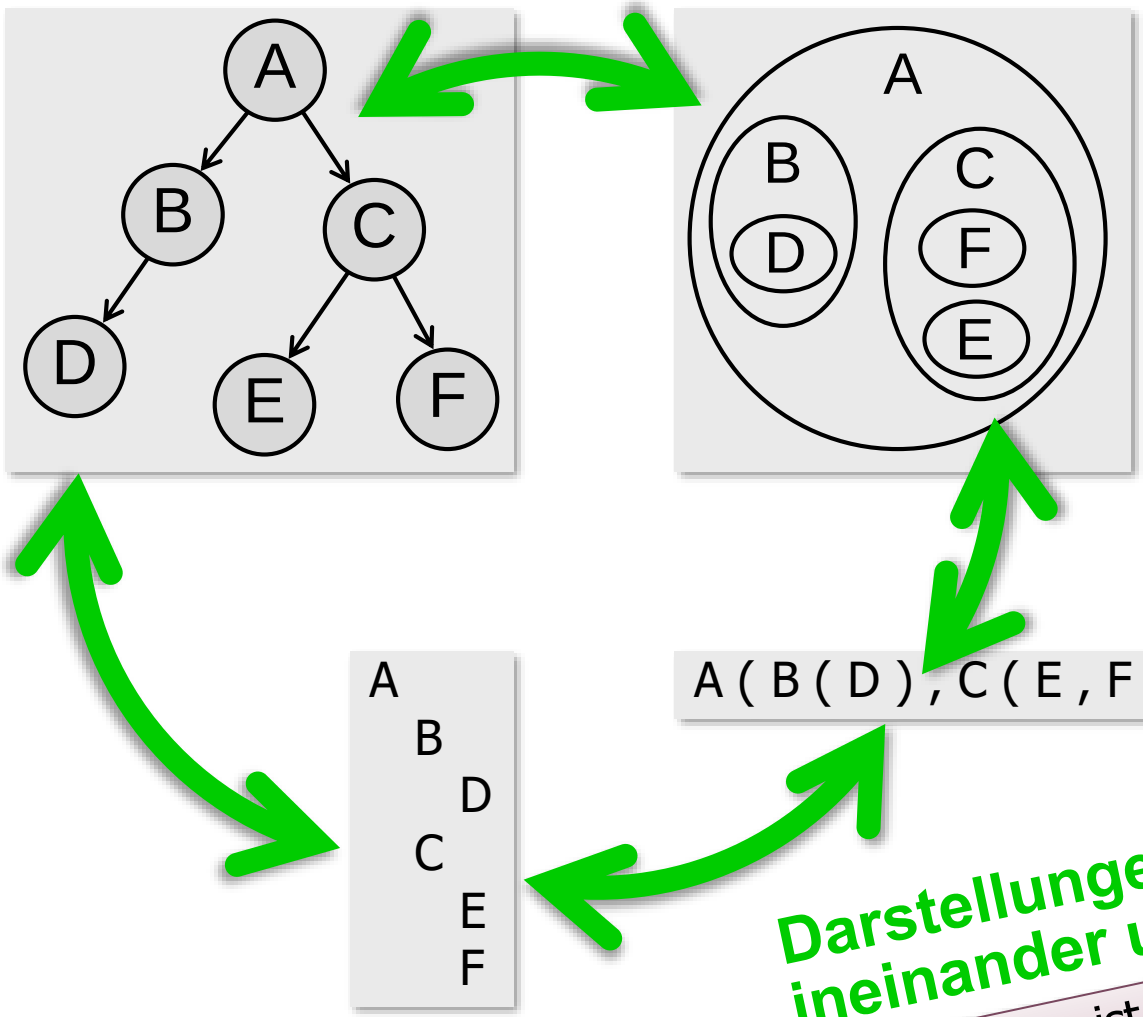
Mit jeweils genau der selben Bedeutung

Alle Darstellungen sind gleichwertig (→ „äquivalent“)

Sind ~~verschiedene Zeichen~~ für ~~den selben~~ Affen „Chacha“

Zum einzigen & wahren Affen

Darstellung von Wurzelbäumen



Alles sind nur unterschiedliche Darstellungen des gleichen „abstrakten" Baums!

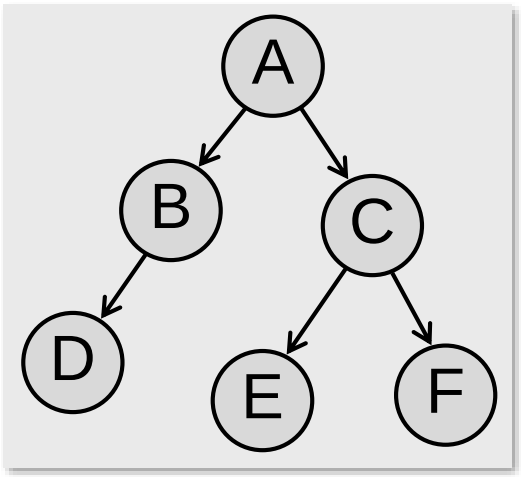
Mit jeweils genau der selben Bedeutung

Alle Darstellungen sind gleichwertig (→ „äquivalent")

Darstellungen systematisch ineinander umwandeln?

Ja, das ist keine allzu schwere Übung → am besten mit rekursivem Ansatz

Darstellung vs. „Ding an sich“



„Das ist der Baum!“

Streng genommen falsch: Es handelt sich nur um eine bestimmte (manchmal gut geeignete) Darstellung eines Baums

„Das ist nur eine Darstellung des Baums“

```
A(B(D),C(E,F))
```

Das „nur“ ist hier abwertend und schon fast beleidigend – weglassen!

„Kann man den Baum aus der Klammerdarstellung eindeutig rekonstruieren?“

Der Frage liegt eine falsche Auffassung zugrunde – gemeint ist offenbar: „Kann man die Klammerdarstellung eindeutig (und algorithmisch) in eine Darstellung (des gleichen Baums) als gerichteten Graphen umwandeln?“

„Ah ja!“

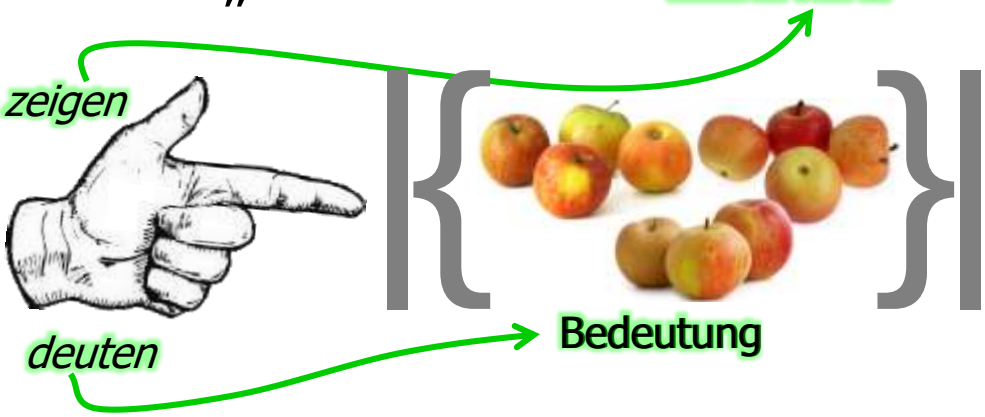
Alles klar? Dann doch noch eine Denkübung: Handelt es sich bei `A(B(D),C(E,F))` und `A(C(E,F),B(D))` um den selben Baum?

„Meinst du nicht eher «...verschiedene Darstellungen des selben...»“?

Ertappt!

Darstellung und Bedeutung

- Generell lässt sich eine Information auf verschiedene Weise darstellen bzw. repräsentieren
  - Kann jeweils abhängig vom Zweck mehr oder weniger adäquat sein
- Wir kennen das von Zahlen – z.B. der „elf“ in diversen Zeichen:
  - Dezimal: 11
  - Dual: 1011
  - Im Gefängnis: 𝍸 I
  - Römisch: XI
  - Vor 1860: eilf
  - Englisch: eleven



- Dies alles soll, „richtig“ interpretiert, immer das selbe bedeuten
  - → Semantik (von griech. σημαντικός = bezeichnend bzw. σῆμα = Zeichen)
- Darstellungen, Zeichen etc. lassen sich oft verschieden deuten
  - Z.B. „XI“ auch als Buchstabenfolge oder Wort einer Sprache
  - Ohne konkrete Interpretation sind sie mehrdeutig bzw. bedeutungslos

Bedeutungslose Zeichenmanipulation?

- Ein Zeichen wie 01000001 kann vieles bedeuten – abhängig vom Kontext oder Bezugssystem bzw. je nach Interpretation etwa:
  - Einemillioneins (Zahl zur Basis 10)
  - Fünfundsechzig (Dualsystem)
  - Grossbuchstabe 'A' (ASCII-Code)
  - Istore_2 (Java Bytecode)
  - Null – Eins – Null – Null – Null – Null – Null – Eins (Bitfolge von links nach rechts)
- Computer speichern und verarbeiten Zeichen uninterpretiert
  - Sie haben quasi keine Ahnung, was eine Bitsequenz (für uns) bedeutet
- Daher kann es zu folgenschweren Fehlinterpretationen kommen
  - Wenn etwa Daten als Befehlscodes interpretiert werden
- Dennoch können (quasi ohne Sinn und Verstand) Zeichen mit Digitalschaltungen oder Software „sinnvoll" manipuliert werden
  - Man baut und programmiert informationsverarbeitende Systeme gerade so, dass die Transformation von der Eingabe zur Ausgabe bei einer bestimmten festgelegten Interpretation zur beabsichtigten Informationsverarbeitung passt
  - Z.B.: „Bitfolgen A auf Eingang 1 und B auf Eingang 2 erzeugen auf dem Ausgang eine Bitfolge, die der Summe von A und B entspricht, wenn man alles an den Ein- und Ausgängen als Zahlen im Dualsystem interpretiert"

Zeichen – Gekritzel auf einem Stück Papier

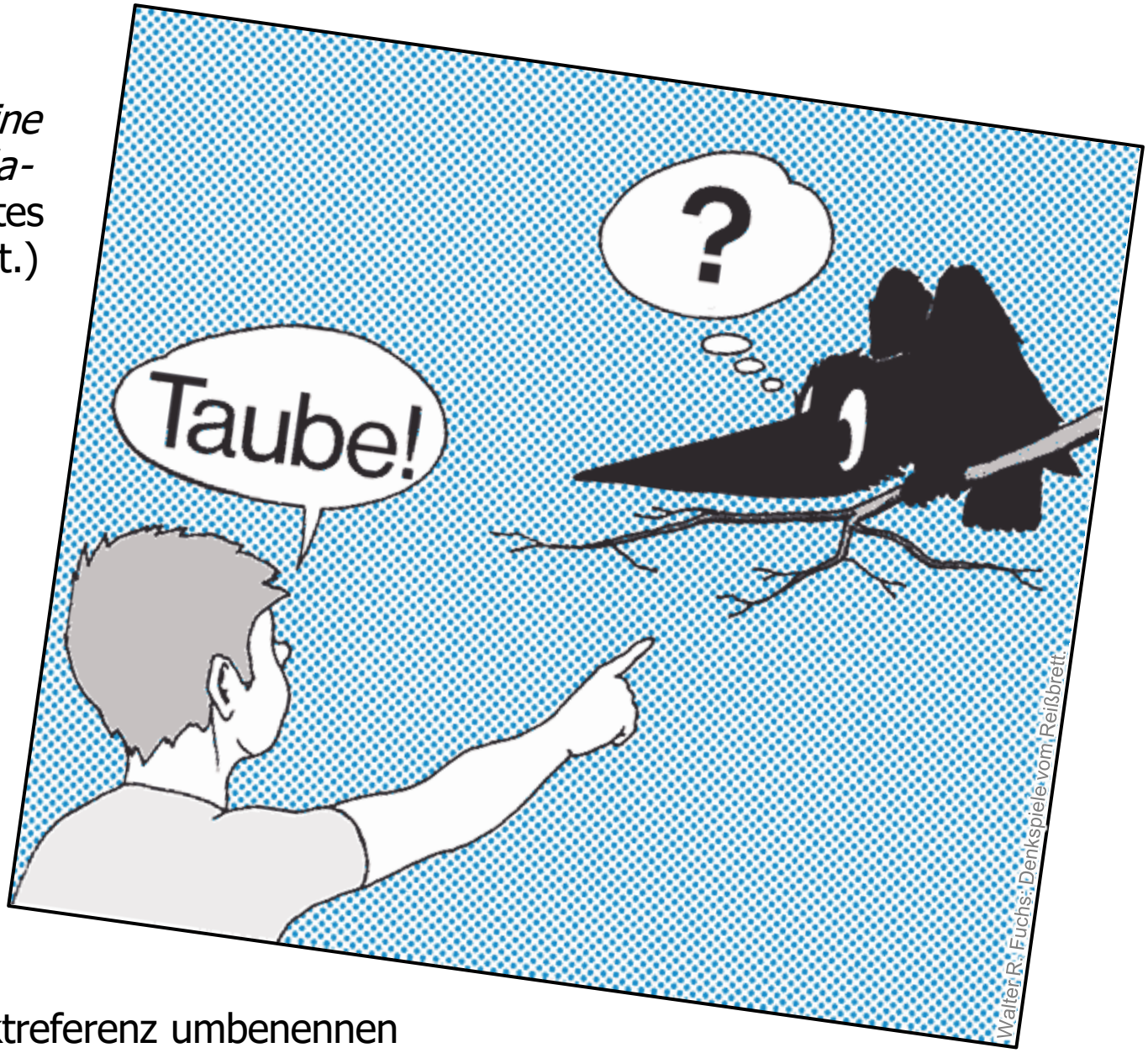
„...Zeichen. Gekritzel auf einem Stück Papier, das erst von einem Menschen durch die Bezüge, die er daran knüpft, zum Leben erweckt werden muss.

Durch ihre Bezüge bekommen die Zeichen Bedeutung. Aber auch losgelöst von ihrer Bedeutung kann man mit Zeichen etwas Sinnvolles anstellen. Man kann sie umformen. Wenn man sie nach bestimmten Regeln umformt, dann kommt dabei etwas heraus. Jeder kennt das, wenn man etwas schriftlich ausrechnet: Zuerst steht eine Folge von Zeichen da. Alle Information liegt vor, aber die Frage ist offen. Dann formt man die Zeichen nach bestimmten Regeln um und kommt zum Ergebnis. *Manchmal sogar ganz ohne die Bedeutung der Zeichen.* Man muss noch nicht einmal wissen, dass »5« Fünf bedeutet, solange man sich an die Regeln hält, wie aus »5+3« eine »8« zu machen ist.

Zeichen sind wie der Brennpunkt in meinem Auge: Ein kleiner Punkt, an dem auf der einen Seite der unterschiedlich grosse Bezugskegel der Bedeutungen abgeht. Auf der anderen Seite des Brennpunkts öffnet sich auch etwas: der Fächer formalen Schliessens. Die Möglichkeiten, die Zeichenfolge nach Regeln umzuformen. Dieser Kegel der Möglichkeiten des Umformens sind die Algorithmen."

[Sebastian Stiller: Planet der Algorithmen]

Ein Rabe, den man eine Taube nennt, wird dadurch nicht weiss. (Altes slawisches Sprichwort.)



(Wenn wir eine Objektreferenz umbenennen oder löschen, ändert das nichts am Objekt.)

Zeichen

Semiotik: Wissenschaft, die sich mit Zeichensystemen befasst

- Zeichen sind Objekte, deren Funktion darin besteht, auf andere Objekte hinzuweisen oder etwas über sie auszusagen
  - Ihre primäre Funktion ist die der Referenz, nicht die der Präsenz
  - Ein Zeichen liefert also nicht „das Ding" selbst, sondern Information zum bezeichneten Ding
- Beispiel: Verkehrszeichen
  - Kein an sich wesentliches Objekt, sondern Information zu einem anderen Objekt

- Damit die transportierte Information „ankommt":
  - Wiedererkennbarkeit für relevante Adressaten (hier: Autofahrer)
  - Wenn Zeichen nicht (z.B. als Piktogramme oder Ikonen) direkt verständlich sind, ist für deren Bedeutung eine explizite Übereinkunft bzw. ein „Code" nötig

- In der Informatik: Berechnung = Zeichenmanipulation
  - Geht im Computer ohne Referenz auf Bezugsobjekt (d.h. rein syntaktisch)!

Zeichen


CAUTION
WATER ON ROAD
DURING
RAIN


PHYSICALLY
IMPOSSIBLE
ENTRY
1,5 m


SIGN NOT
IN USE

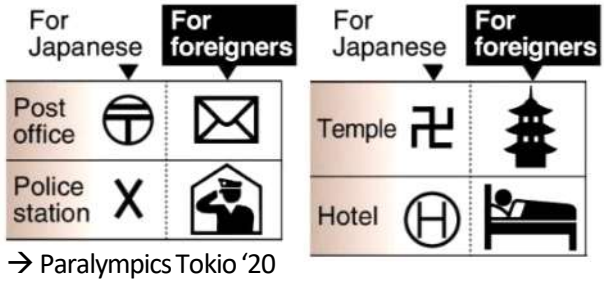

NORD
EST
OUEST


Hauptwegweiser

Zeichen

Nothing is a sign unless it is interpreted as a sign. -- Charles Peirce

- In der Semiotik-Theorie von Charles Peirce (1839 – 1914) betrachtet man unterschiedliche Arten von Zeichen:
- Ein icon (vom Griech. εἰκών = Bild) ist ein Zeichen, das eine wahrnehmbare Ähnlichkeit zum referenzierten Objekt hat



→ Paralympics Tokio '20

- Ein Symbol ist durch Arbitrarität und Konventionalität bezüglich des Bezeichneten charakterisiert
  - Ohne einen „Code", z.B. den Morse-Code, bleibt uns folgende Symbolfolge unverständlich: -·-· --- -·· ·
  - Der Schweizer Sprachwissenschaftler Ferdinand de Saussure hatte allerdings eine andere Auffassung von „Symbol"; es gibt hier keine objektiv richtige Theorie!

Das ist nicht das berühmte Bild von René Magritte; es handelt sich um eine digitale Kopie einer Fotografie von Magrittes Bild

Alice trifft Goggelmoggel...

..., eine eiförmige Kreatur, der mit ihr über Semiotik diskutiert. (Zum Nachforschen: Gab es diese oder ähnliche Sprachtheorien damals tatsächlich schon?) Unten 2 Textpassagen aus „Alice hinter den Spiegeln" von Lewis Carroll, übersetzt von Christian Enzensberger. „Through the Looking Glass" (1871) ist die Fortsetzung zu Alice im Wunderland. Goggelmoggel heisst im Original „Humpty Dumpty" und ist eine Figur aus einem populären britischen Kinderreim; in anderen Sprachen wird er Lille Trille, Unto Dunto, Zanco Panco oder Rondu-Pondu genannt.

„Steh nicht herum und gackere vor dich hin", sagte Goggelmoggel und blickte sie zum ersten Mal an, „sondern sag, wie du heisst und was du willst."

„Ich heisse Alice, aber –"

„Albern genug für einen Namen!", unterbrach sie Goggelmoggel unwirsch. „Was soll der denn bedeuten?"

„Muss denn ein Name etwas bedeuten?", fragte Alice zweifelnd.

„Das ist doch klar", sagte Goggelmoggel, kurz auflachend.

Alice trifft Goggelmoggel…

[…] „Wenn das keine Glocke ist!“

„Ich verstehe nicht, was Sie mit ‚Glocke‘ meinen“, sagte Alice.

Goggelmoggel lächelte verächtlich. „Wie solltest du auch – ich muss es dir doch zuerst sagen. Ich meinte: ‚Wenn das kein einmaliger schlagender Beweis ist!‘“

„Aber ‚Glocke‘ heisst doch gar nicht ein ‚einmalig schlagender Beweis‘“, wandte Alice ein.

„Wenn ich ein Wort gebrauche“, sagte Goggelmoggel in recht hochmütigem Ton, „dann heisst es genau, was ich für richtig halte – nicht mehr und nicht weniger.“

„Es fragt sich nur“, sagte Alice, „ob man Wörter einfach etwas anderes heissen lassen kann.“

„Es fragt sich nur“, sagte Goggelmoggel, „wer der Stärkere ist, weiter nichts“.

Der bekannte französische Psychoanalytiker Jacques Lacan (1901 – 1981) war von Lewis Carrolls Büchern begeistert und bezeichnete Goggelmoggel als „maître du signifiant“. (Wir aber möchten mit diesem eiförmigen Besserwisser lieber kein Scrabble spielen, oder?)

Jabberwocky [Lewis Carroll: Behind the Looking Glass, 1872]

Jabberwocky	Der Zipperlake
'Twas brillig, and the slithy toves	Verdaustig wars, und glasse Wieben
Did gyre and gimble in the wabe;	Rotterten gorkicht im Gemank;
All mimsy were the borogoves,	Gar elump war der Pluckerwank,
And the mome raths outgrabe.	Und die gabben Schweisel frieben.

„Das reicht fürs erste" unterbrach sie Goggelmoggel; „da kommen schon recht viele schwere Wörter vor. Verdaustig heißt vier Uhr nachmittags – wenn man nämlich noch verdaut, aber doch schon wieder durstig ist." „Das paßt sehr gut", sagte Alice; „und glaß?" „Nun glaß heißt glatt und naß. Das ist wie eine Schachtel, verstehst Du: zwei Bedeutungen werden dabei zu einem Wort zusammengesteckt." „Jetzt versteh ich's schon", sagte Alice nachdenklich. „Und was sind Wieben?" „Also, Wieben sind so etwas Ähnliches wie Dachse – und wie Eidechsen – und so etwas Ähnliches wie Korkenzieher." „Das müssen aber merkwürdige Geschöpfe sein." „Das wohl", sagte Goggelmoggel; „Sie bauen außerdem ihre Nester unter Sonnenuhren – und außerdem fressen sie nur Käse." „Und was ist rottern und gorkicht?" „Rottern ist das gleiche wie rotieren; das heißt: sich schnell drehen. Gorkicht heißt alles, was sich in Holz einbohrt." „Und ein Gemank ist dann wohl der freie Platz um eine Sonnenuhr von der Art, wie sie oft in einem Park stehen?" fragte Alice über ihre eigene Scharfsinnigkeit verwundert.

Lewis Carroll (1832 – 1898), mit bürgerlichem Namen Charles Lutwidge Dodgson, lehrte Mathematik in Oxford und verfasste neben seinen Hauptwerken *Alice im Wunderland* und *Alice hinter den Spiegeln* Erzählungen, Gedichte, aber auch wissenschaftliche Werke.

Die indogermanische Wurzel von *sign*

Das englische Wort für „Zeichen“ im oben verwendeten Sinn ist „sign“ (oder alternativ, wie auch im Deutschen, „symbol“). Es stammt (vermittels des Französischen „signe“) aus dem Lateinischen „signum“ und beruht auf der indogermanischen Wurzel **sekw-* bzw. **sek-*, was die Wortfamilie zu „schneiden“ ausmacht.

Die Bedeutung von „signum“ im Sinne von „Zeichen“, „Kennzeichen“, „Vorzeichen“ etc. geht insofern auf in Holzstäbe eingeschnittene Marken bzw. Zeichen zurück.

Von der Sprachwurzel **sek-* sind beispielsweise auch die deutschen Wörter „Säge“, „Sense“, „Segel“ (abgeschnittenes Stück Stoff) oder „Sichel“ abgeleitet, ferner „Scheide“ und indirekt sogar „Skandal“. Im Lateinischen hat sich „secare“ mit der direkten Bedeutung „schneiden“ erhalten, was zu deutschen Lehn- und Fremdwörtern wie „sezieren“, „Sektion“, „Segment“, „Sektor“, aber auch „Insekt“ führte. Auch das lateinische „sexus“ (getrennte Geschlechter) stammt von dieser Sprachwurzel, interessanterweise aber auch Wortformen aus „scio“ (Wissen) bzw. „scientia“ (Kenntnis, Wissenschaft – im Sinne von „eine Sache von einer anderen trennen können“) und damit auch das englische „science“ und das deutsche „gescheit“. „Schranke“, „Schirm“ und „screen“ scheinen sich ebenfalls daraus abzuleiten.

Indogermanische Wurzeln von *Zeichen* und *digital*

Ein Blick in einige Wörterbücher verrät, wie „Zeichen" mit „digital" und anderen Wörtern zusammenhängt

digital *(Adjektiv)*

- Signale und Daten dargestellt in Ziffern (meist binär als Folge von Nullen und Einsen).
- Das Adjektiv wurde im Dt. zunächst im med. Bereich im Sinne von »mithilfe des Fingers« verwendet. In dieser Bedeutung handelt es sich um eine Entlehnung aus lat. *digitalis*, einer adj. Ableitung des Substantivs *digitus* »Finger«. In der Technik und in der Datenverarbeitung steht »digital« für »zahlenmäßig, ziffernmäßig; in Stufen erfolgend« und ist in der 2. Hälfte des 20. Jh.s aus engl. *digital* übernommen worden, einer adj. Ableitung des Substantivs *digit* in seiner Bedeutung »Ziffer«.

„Digital" in Meyers Conversationslexikon von 1846: 1) was die Finger oder Zehen betrifft; 2) fingerförmig.

Digit *das; -[s], -s*

- Vom Engl. *digit*, eigtl. = (zum Zählen benutzter) Finger (von lat. *digitus* = Finger; Zehe).
- Ziffer (= schriftliches Zahlzeichen), Stelle (in der Anzeige eines elektronischen Geräts).

Als im Italienischen das Wort *nulla* „Nichts" an die Stelle von *cifra* „Null" (aus arabisch *sifr*) trat, übernahm *cifra* die Aufgabe von *figura*, das bisher „Zahlzeichen" bedeutet hatte.

digitus [lat.]

- = dt. *Finger.*
- Von der indogermanischen Wurzel ***deyk̂-*** »zeigen, aufzeigen, weisen«; vgl. Sanskrit *dic-, diśáti* »aufweisen, sehen lassen, hinweisen, zeigen«; griechisch δείκνῡμι »zeigen, nachweisen«; althochdeutsch *zeigon* → dt. *zeigen* (sowie *bezeichnen*, *anzeigen, Zeichen, [ver]zeihen, zeichnen* [ahd. *zeihhonon*] etc.); angelsächsisch *tæcan* → engl. *to teach* (zeigen im Sinne von erklären); engl. *token* (entspr. *Zeichen* im Sinne von Wunderzeichen).
- Vgl. auch lat. *dicere* (sagen = mit Worten auf etwas hinweisen), *indicare* (anzeigen), *index* (Anzeiger), *iudex* (der das Recht Weisende); franz. *dire*; dt. *Diktion*; dt. *Zehe,* engl. *toe.*

Indogermanische Wurzeln von *Zeichen* und *digital*

Englische Wörterbücher

digital *(adjective)*

- representing data as a series of numerical values;
- available in electronic form; readable and manipulable by computer;
- pertaining to, noting, or making use of computers and computerized technologies;
- of or relating to a digit or finger; performed with the fingers.
- *digital data:* A description of data which is stored or transmitted as a sequence of discrete symbols from a finite set, most commonly this means binary data represented using electronic or electromagnetic signals.

Word history: Meaning "using numerical digits" is from 1938.

Digit *(noun)*

- any of the Arabic figures of 1 through 9 and 0 (also called *figure*);
- any of the symbols of other number systems, as 0 or 1 in the binary;
- a finger or toe (this sense in English is attested from 1640s).

Word origin: 1350 – 1400 (Middle English); from Latin *digitus* finger.

Early 13cent. from Old French *figure* (10cent.) "shape, body, form, figure; symbol, allegory," from Lat. *figura* "shape, form, figure," originally in English with meaning "numeral," but sense of "form, likeness" is almost as old.

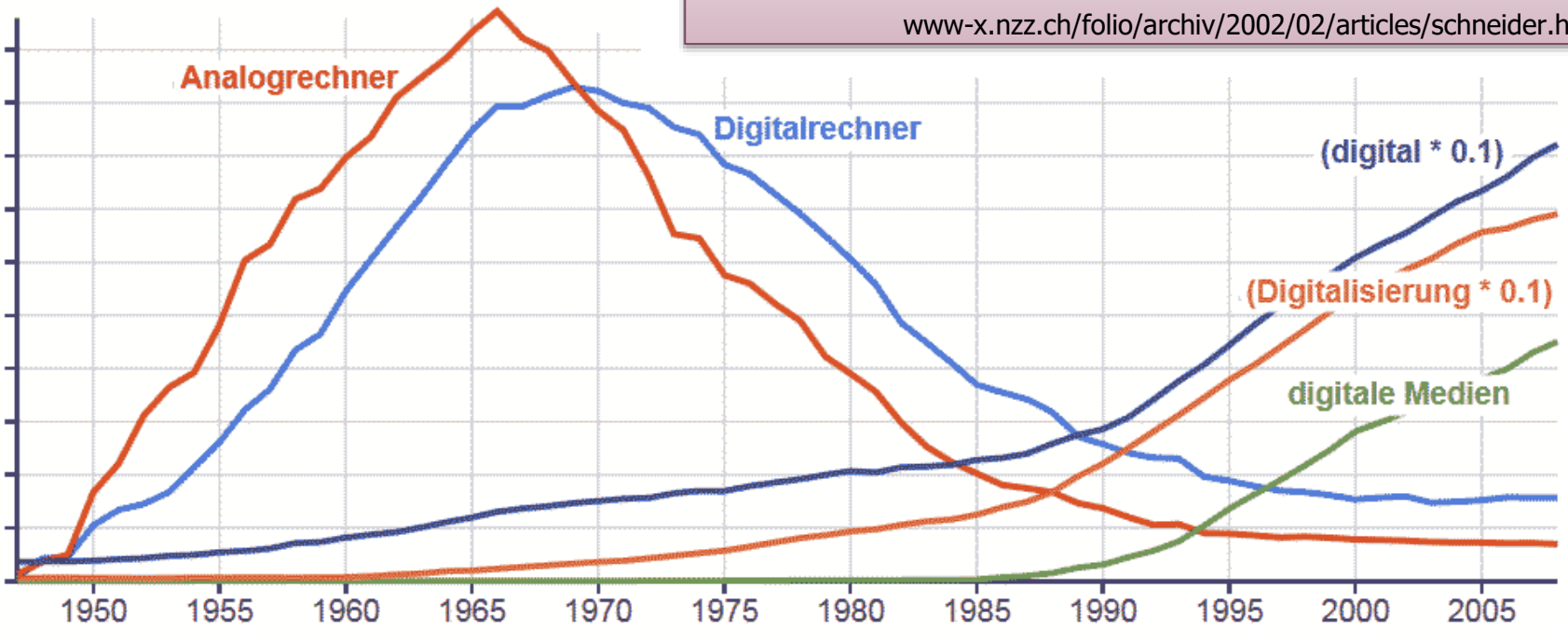
Der angelsächsische Benediktiner Beda nannte im 8. Jh. das seinerzeit übliche Fingerrechnen „computus digitorum"; im 10. Jh. bezeichnete dann Gerbert von Aurillac bei seinem speziellen Abacus die zugehörigen (auf arabisch mit 1 – 9 „bezifferten"!) Rechensteine, die in den Dezimalspalten verschoben wurden, ebenfalls als digiti, es waren gewissermassen symbolische Finger.

„Digital" und „Zeichen" (und auch „token" im Englischen) haben also die gleiche Sprachwurzel *deyḱ-; trotzdem stört es uns nicht, wenn wir Begriffe wie *digitales [Wasser]zeichen* („*deyḱdeyḱ"?) oder *digitaler Finger[abdruck]* (*empreinte digitale digitale/numérique* im Französischen!) verwenden.

Alles digital!

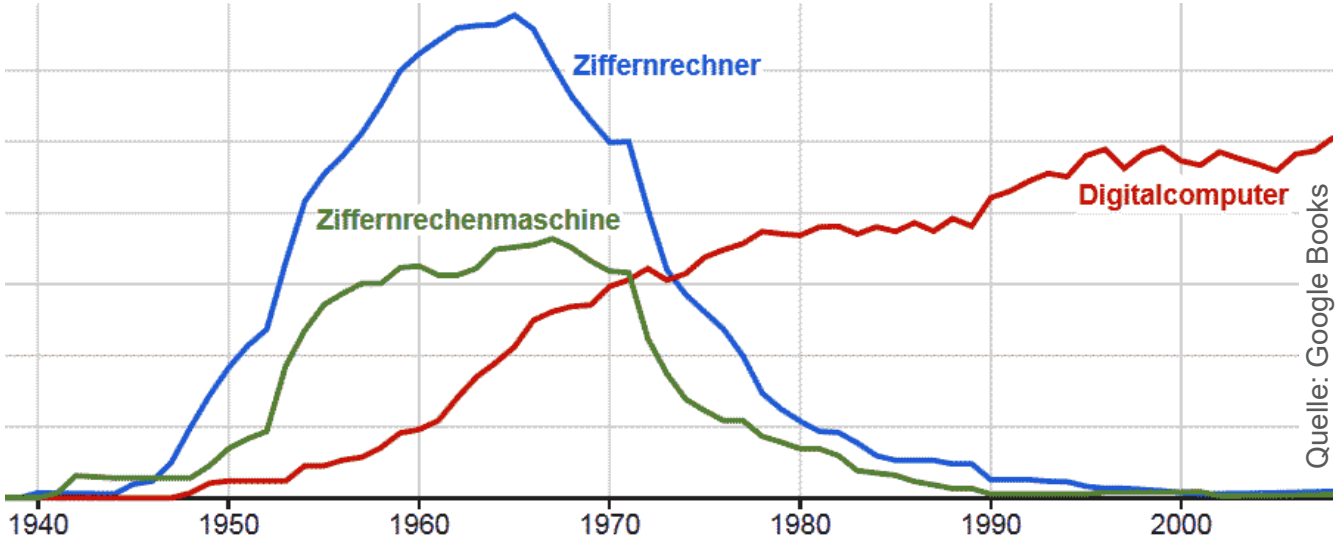
Was bedeutet digital? Erklärung von U. S. aus B., Besitzerin eines Radioweckers mit Digitalanzeige, eines Handys und zweier Stereoboxen, auf denen «digital» steht: «Also wenn eine Uhr nicht rund ist, dann ist sie digital.»

www-x.nzz.ch/folio/archiv/2002/02/articles/schneider.html



Der Begriff „digital“ tritt (in seiner nicht-medizinischen Bedeutung) erst ab den 1950er-Jahren in der deutschen Sprache auf, entlehnt aus dem Englischen (digital = using numerical digits). Anfangs musste man bei Computern noch extra das „Digitale“ hervorheben, um sich von den Analogcomputern abzuheben; generell wurden Computer seinerzeit auch noch als Rechner oder „Rechenautomaten“ bezeichnet. Die Begriffe „digital“ und „Digitalisierung“ erhielten, zusammen mit der Popularisierung des Internet, in den späten 1980er-Jahren einen gewaltigen Schub, auch wenn „Digitalisierung“ seinerzeit (die Graphik reicht nur bis 2008) noch gar nicht in der umfassenden Bedeutung wie heute gebraucht wurde: *In Erweiterung der ursprünglichen Bedeutung, der Umwandlung von analogen in digitale Signale, bezieht sich „Digitalisierung“ heute im Geschäftskontext auf den generellen Einsatz informationstechnischer Innovationen.*

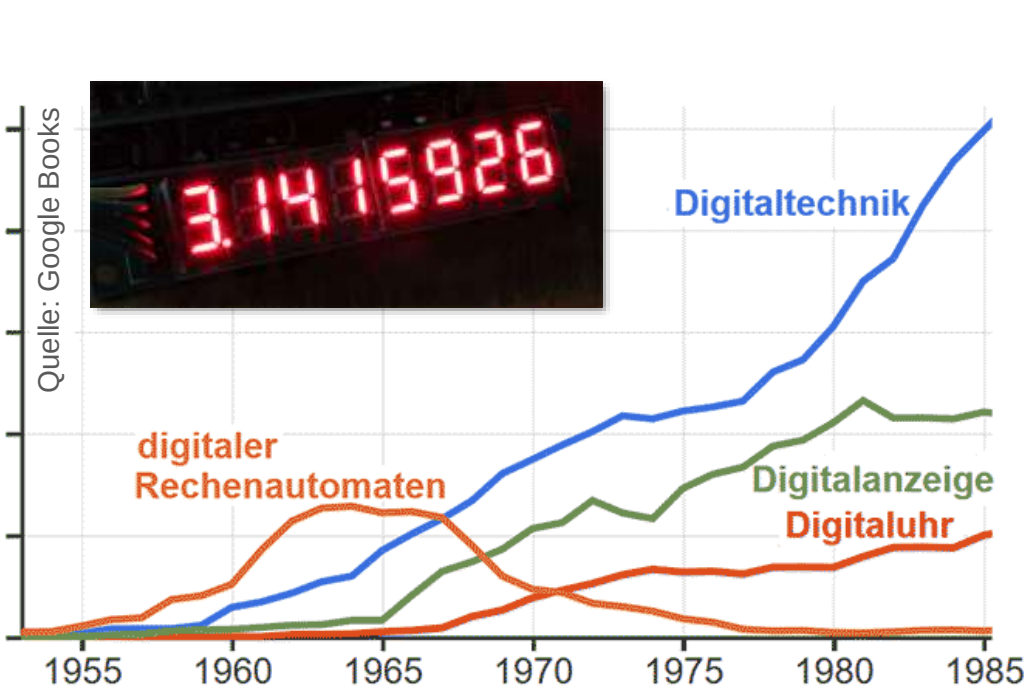
Alles digital!

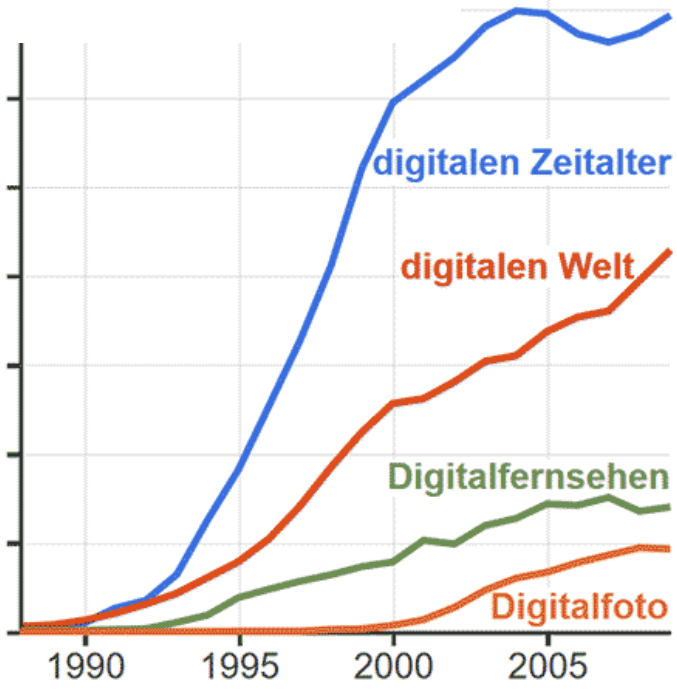


Schon seit dem Mittelalter wird für das engl. Wort „digit“ im Deutschen „Ziffer“ (und im Französischen „chiffre“) verwendet. Daher ist der „digital computer“ anfangs auch konsequenterweise „Ziffernrechner“ (oder auch „Ziffernrechenmaschine“) genannt worden; erst später wurde dies durch den Anglizismus „Digitalcomputer“ ersetzt. Das Wort „Ziffer“ entstammt übrigens dem Arabischen „sifr“ (صِفْر), wo es „nichts“ bedeutet und auch die Ziffer „Null“ bezeichnete; „zero“ ist davon abgeleitet – und auch im Deutschen wurde die Null anfangs „cifra“ genannt (und eine Ziffer dafür „Figur“, vgl. engl. „figure“). Das indisch-arabische Zahlensystem gelangte schon um 750 nach Persien; Leonardo von Pisa („Fibonacci“) lernte es auf seinen Reisen im Mittelmeergebiet kennen, er schrieb darüber 1202 ein Buch „Liber Abaci“ und machte es so im Europa südlich der Alpen bekannt. In den italienischen Handelsstädten ersetzte es zu Beginn des 15. Jahrhunderts das römische Zahlensystem, im deutschen Sprachraum dauerte es bis zum Ende des 15. Jahrhunderts. Insbesondere die Null hatte es schwer, akzeptiert zu werden – bedeutet sie selbst doch nichts, verzehnfacht aber den Wert einer anderen Ziffer. „Wie die Puppe ein Adler sein wollte, der Esel ein Löwe, die Äffin eine Königin – so wollte die cifra eine Figur sein“ wurde sie beispielsweise verspottet.

Alles digital!

Seit einigen Jahren hat sich der Begriff „Digital" über fast alle Lebens- und Arbeitsbereichen gelegt. Eine Definition ist dabei völlig unwichtig geworden, „Digital" funktioniert als universeller Qualitätsbegriff, als Marke, als Synonym für Fortschritt. -- Norbert Nowotsch





Hier sieht man, wie die „Digitalisierung" phasenweise Bedeutung erlangt und mit den entsprechenden Erscheinungen und Produkten in den allgemeinen Wortschatz eindringt. Anfangs war dies etwa die Digitalanzeige, zunächst in der klassischen 7-Segment-Darstellung mit den „eckigen" Ziffern, die sich z.B. bei Digitaluhren und Radioweckern manifestierten, dann aber auch bei vielen Haushaltsgeräten und Messinstrumenten. Digitalfernsehen und Digitalkamera bzw. Digitalfoto als „voll digitalisierte" Produkte, bei denen die Signalverarbeitung in digitalisierter Weise erfolgt, tauchten erst später auf. Teilweise wird „digital" dann auch im Wechselspiel mit „elektronisch" verwendet (z.B. digitale / elektronische Signatur). Begriffe wie „digitale Revolution" (im Bild nicht dargestellt), „digitales Zeitalter" oder „digitale Welt" erscheinen ab ca. 1990.

Digitalisierung als Modebegriff

„In der Bevölkerung teilt die Digitalisierung das Los vieler anderer politischer Schlagworte, die so oft genannt und gehört werden, dass sich der Normalbürger nicht viel darunter vorstellen kann. [...] Im schlimmsten Fall weiss er nichts über Digitalisierung, bestenfalls noch, dass sie ‚irgendwas mit Computern zu tun hat'.“

[Heike Schmoll: Die Digitalisierung als Schlagwort und Schlüsselbegriff in politischen Reden, Frankfurter Allgemeine Zeitung, 27.10.2016, S. 8]

„Seit ein paar Jahren wird die Welt digital. Oder zumindest scheint es so. Alles digitalisiert sich, Bücher, Fernsehen, Arbeit, Autos, Strom, Telefon, Politik, sogar Radio. [...] Auch wenn man jetzt ständig davon hört, ist das Phänomen Digitalisierung nicht neu. Computer haben das analoge Stadium ab den 1940er Jahren allmählich verlassen. Banken, Versicherungen und zahlenintensive Verwaltungsbereiche digitalisierten ihre Rechenvorgänge ab den 1960er Jahren.

→

*Suchergebnisse zu „digital" bei Google im Jahr **2006***

Digitalisierung als Modebegriff (2)

1976 heißt es im Spiegel: ‚Die elektronische Revolution hat die Zeitungsverlage erreicht', gemeint waren Lichtsatz, Bildschirmterminals und Speicherung der Texte auf Magnetbändern. Im Lauf der 80er und 90er wurden textlastige Verwaltungstätigkeiten digital. Die Umstellung von Schallplatte auf CD fand in den 1980er Jahren statt. Ende 1997 war das deutsche Telefonnetz vollständig digitalisiert. Fotografie und Film folgten.

Während der ersten siebzig Jahre dieser Vorgänge spielte der Begriff der Digitalisierung keine große Rolle. Wo er ab den 1990er Jahren vereinzelt auftauchte, bezeichnete er noch die konkrete Digitalisierung von etwas: von Telefonnetzen, Wörterbüchern, Landkarten. Erst ab 2010 wird der Begriff häufiger und in seiner heutigen vagen Bedeutung verwendet.“

[Kathrin Passig, Aleks Scholz: Schlamm und Brei und Bits – warum es die Digitalisierung nicht gibt. Merkur 69 (798), S. 75-81, 2015]

*Suchergebnisse zu „digital" bei Google im Jahr **2016***

„Digitalisierung“ bei Wikipedia

Digitalisierung steht für

- die Auswirkung der verstärkten Nutzung von Computern und Digitaltechnik in Wirtschaft, Kultur, und Politik
- die Überführung analoger Größen in diskrete Werte
- die Einführung von Digitaltechnik
- Verabreichung von Digitalis-Präparaten in der Medizin

Die Begriffserklärung wurde und wird immer wieder verändert; hier die Version Ende 2017

Hingegen legen z.B. Michael Barot und Daniel Baumgartner in ihrem Artikel „Informatik am Gymnasium“ [Informatik-Spektrum 42 (2), 2019] den Begriff strikt (und eindimensional) aus:

Digitalisierung. Darunter versteht man den Prozess, wenn Informationen von ihrer analogen Form in eine digitale Form übersetzt werden. Zum Beispiel findet eine Digitalisierung statt, wenn die in analoger Form codierte Musik einer Schallplatte in die digitale Form einer CD oder MP4-Datei übersetzt wird. Und genau dies ist die Bedeutung des Begriffs der Digitalisierung: die Speicherung von Daten in digitaler Form. Digitalisiert wird also eine alte Fotografie oder die Aufzeichnung einer Sonate. Und nicht etwa die Gesellschaft, wie man häufig da und dort liest oder hört.

„Digitalisierung“ im Lexikon

Aus der Enzyklopädie der Wirtschaftsinformatik, Stichwort „Digitalisierung“
(Auszug; Version vom 27.02.2019, Autor: Thomas Hess, LMU München)

Der Begriff Digitalisierung kann auf unterschiedliche Art und Weise interpretiert werden. Traditionell ist die technische Interpretation. Danach bezeichnet Digitalisierung einerseits die Überführung von Informationen von einer analogen in eine digitale Speicherform und andererseits thematisiert sie die Übertragung von Aufgaben, die bisher vom Menschen übernommen wurden, auf den Computer. Heute wird Digitalisierung häufig – etwas breiter – mit der Einführung digitaler Technologien in Unternehmen und als Treiber der digitalen Transformation gleichgesetzt.

Digitale Transformation lässt sich mittlerweile in allen gesellschaftlichen Bereichen erkennen. So verändern sich durch digitale Transformation z.B. Angebot und Nachfrage auf Arbeitsmärkten, die politische Willensbildung oder auch die rechtlichen Rahmenbedingungen. Eine besondere Bedeutung hat die digitale Transformation für Unternehmen. Durch die digitale Transformation agieren Unternehmen in veränderten Märkten und in modifizierten Wertschöpfungsstrukturen. Sie haben sich im Rahmen der digitalen Transformation in den letzten Jahren insbesondere mit der Veränderung ihrer Kernprozesse (sei es im Hinblick auf Effizienz oder auch Kundenorientierung), ihrer Schnittstellen zum Kunden, ihrer Produkte und Services und übergreifend ihrer Geschäftsmodelle beschäftigt.

Digitalisierung und die darauf aufbauende digitale Transformation ist keineswegs ein neues Phänomen – schon vor 20 Jahren war die IT-basierte Verbesserung von Geschäftsprozessen ein Thema vieler Unternehmen. Bedingt durch umfangreiche Fortschritte in vielen technologischen Feldern hat der Druck zur Transformation in den letzten Jahren aber stark zugenommen.

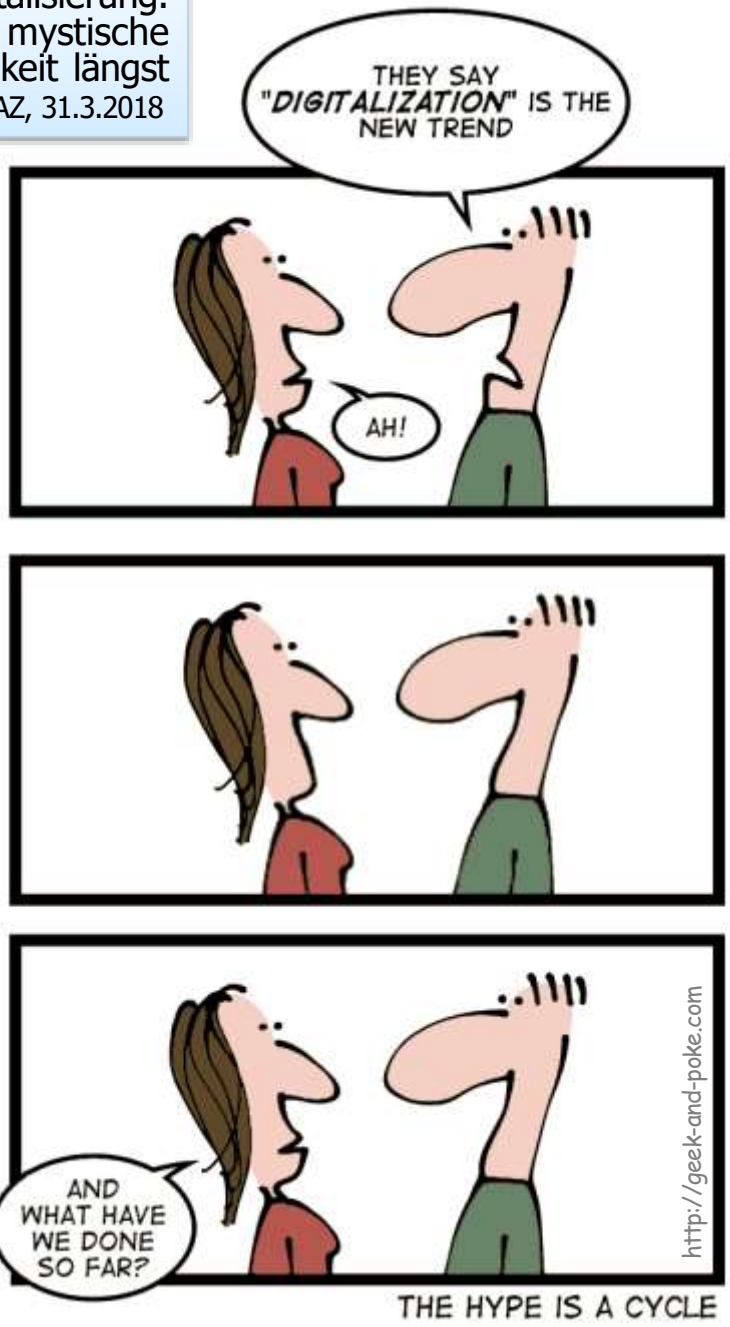
Das D-Wort

Jeder liebt das Zauberwort Digitalisierung. Hohle Reden beschwören eine mystische Kraft, vor deren Unausweichlichkeit längst kapituliert wurde. -- Harald Staun, FAZ, 31.3.2018

Was heisst „Digitalisierung“? Niemand weiss es. Doch alle reden davon. An der Generalversammlung des Wirtschaftsverbands Swico im Mai trug das Eröffnungsreferat den Titel: „Nicht schon wieder das D-Wort“. Auch bei den Digitalisierungsprofis sind in Sachen Digitalisierung Ermüdungserscheinungen festzustellen. Worauf verweist dieses Wort? Digitaltechnik und Informatisierung können nicht gemeint sein, denn die Kommerzialisierung der Computertechnik begann in den 1950er Jahren, und es dürfte sich herumgesprochen haben, dass nicht jede Informatikinvestition gewinnbringend ist. Die Ökonomen, die in den 1980er und 1990er Jahren vergeblich versuchten, die durch die Informatik verursachten Produktivitätsgewinne zu erfassen, sprachen von einem „Produktivitätsparadox“. Wenn also Digitalisierung nicht gemeint ist, was heisst „Digitalisierung“? Was ist das Ziel der Digitalisierung? Digitalisierung?

Das D-Wort ist eine zeitgemässe Art zu sagen: Wandel, Veränderung, Restrukturierung. Das D-Wort hat noch eine weitere Bedeutungsebene, auf der sich Elemente der Technikeuphorie vergangener Zeiten erhalten haben. Es klingt beim D-Wort die Vorstellung mit, dass Technik eine Urkraft sei, der sich der Mensch nicht in den Weg stellen dürfe.

– Stefan Betschon, NZZ vom 16.7.2019



Digitalisierung als kulturelle & soziale Innovation?

„Nur eine kleine Minderheit hat bisher verstanden, dass die digitale Automation vor allen Dingen eine kulturelle und soziale Innovation ist. Es geht um neue Arbeit, um Grundversorgung, um Entwicklung, die abseits der Lohnarbeit stattfindet. Es geht um Zugänge zu Wissen und Bildung, und es geht um eine massive Erleichterung des Alltags für viele, die in der Diskussion gar nicht gesehen werden – die, die heute noch die schmutzigen, schlechtbezahlten Routinejobs machen. Es geht um die Befreiung des Einzelnen von Zwängen, das ist der Zweck, es geht um mehr Freiheit, überall. Um was denn sonst, ihr Komiker?“

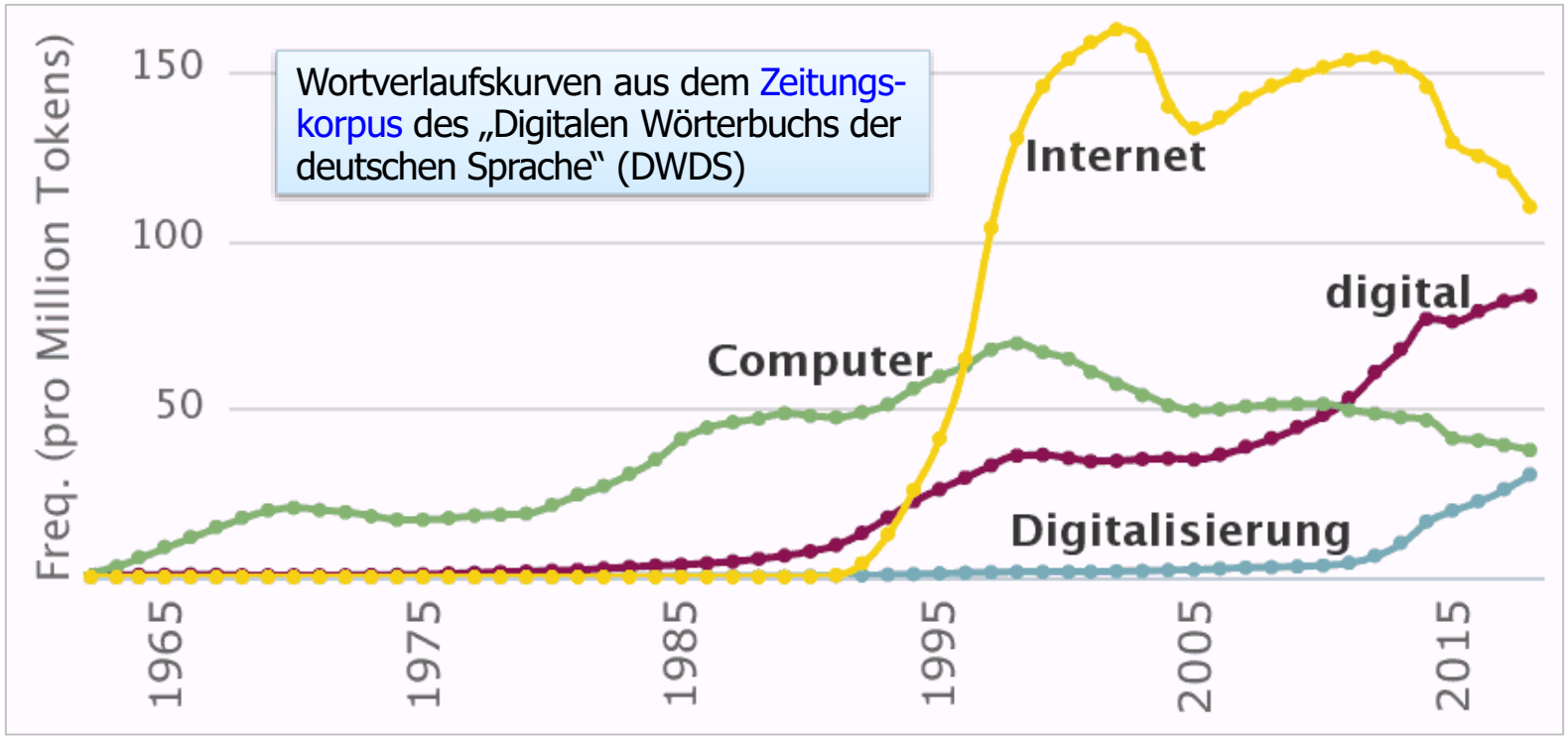
Wolf Lotter, in taz FUTURZWEI

www.dwds.de/wp?q=Digitalisierung&compmethod=diff&comp=&pos=2&minstat=0&minfreq=5&by=logDice&limit=20&view=cloud

Adjektivattribute zu „Digitalisierung“ im DWDS-Wortprofil

Google autocomplete schlägt bei “Digitalization as a…” vor: business issue, can-opener, catalyst and facilitator, chance, defining strategic principle, differentiator, disruptive technological change, driver of growth, driver of innovation, driver of mission, factor for success, foundation for innovation, golden opportunity, growth enabler, guiding light, holistic driver, job killer, key factor, key issue per se, key objective, lever to improve efficiency, man-made afterlife, matter of sustainability, means of economic and social empowerment, means of governing, means to grow, mediator, mindset, pathway to overcome some fundamental future problems, personal fitness program, phenomenon, process, radical and disruptive innovation, resource, risk, route to economic inclusion, scientific term, service, social phenomenon, sociotechnical process, solution, strategic priority, strategy, substitute religion, technical exercise, technology, threat, tool, toolbox, tsunami, turning point for the future, way to reach much bigger goals.

Digitalisierung – digital – Internet – Computer

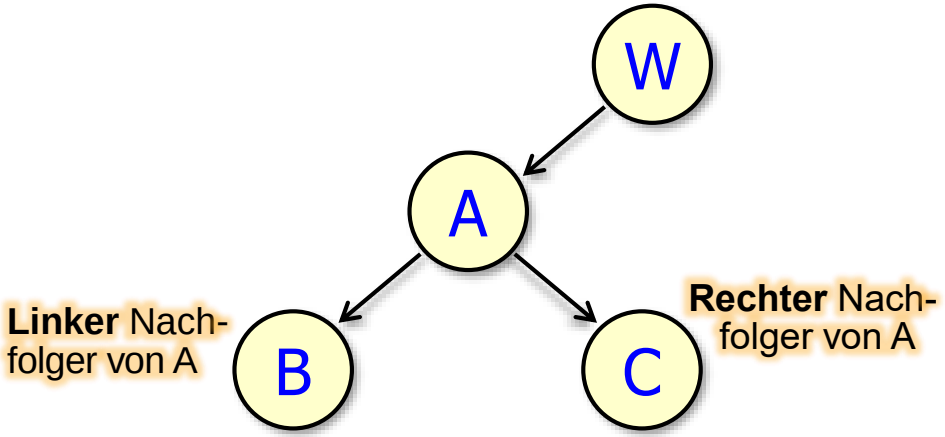


www.dwds.de/r/plot?view=1&corpus=zeitungen&norm=date%2Bclass&smooth=spline&genres=0&grand=1&slice=1&prune=0&window=3&wbase=0&logavg=0&logscale=0&xrange=1945%3A2018&q1=Digitalisierung&q2=digital&q3=Computer&q4=Internet, abgerufen am 2/28/2019.

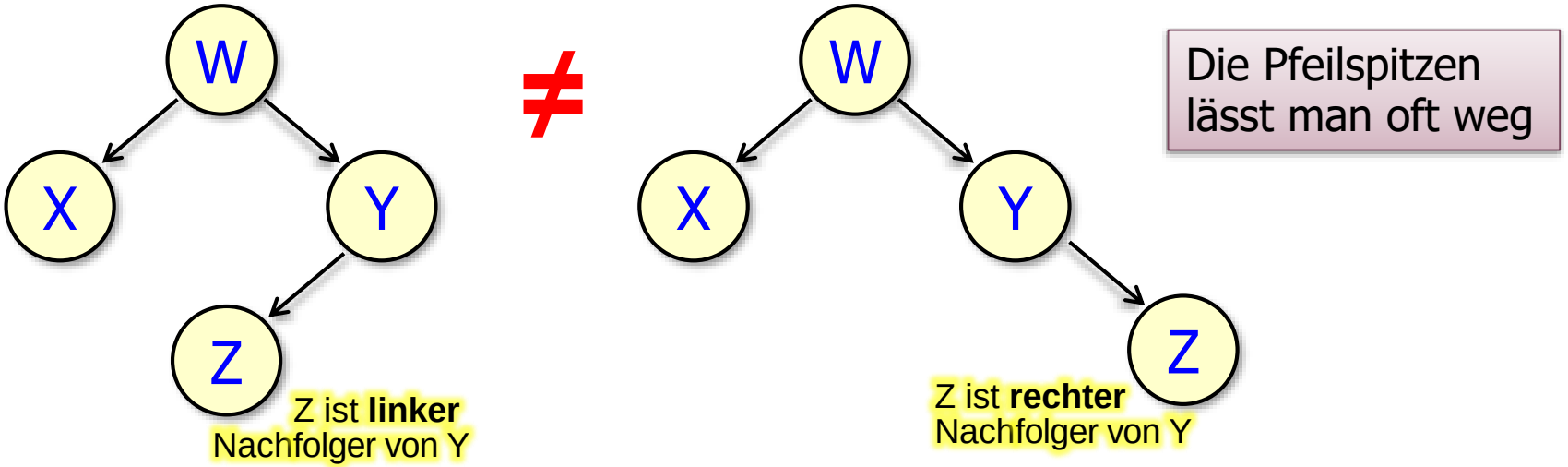
„Spätestens in den Koalitionsverhandlungen haben auch CDU, CSU und SPD das Zauberwort ‚Digitalisierung‘ entdeckt und es derart lieb gewonnen, dass sie es gar nicht oft genug in den Koalitionsvertrag schreiben konnten: 13 Seiten des Vertrags beschäftigen sich explizit mit dem Thema ‚Digitalisierung‘, 93 Mal wurde das Wort ‚Digitalisierung‘ per Copy-and-paste im Dokument verteilt, 148 Mal das Adjektiv ‚digital‘ – wie eine Konjunktion, die selbst die größten politischen Differenzen zusammenhalten kann. [...] Alles soll digital werden: der europäische Binnenmarkt, die öffentliche Verwaltung, der Mittelstand, die Polizei, die Ausbildungsstrategie, die Geschichte der deutschen Frauenbewegung, das Zahnbonusheft,...“ -- Harald Staun, FAZ vom 31.3.2018

Binärbaum

- Wurzelbaum, wo jeder Knoten *höchstens zwei* direkte Nachfolger hat



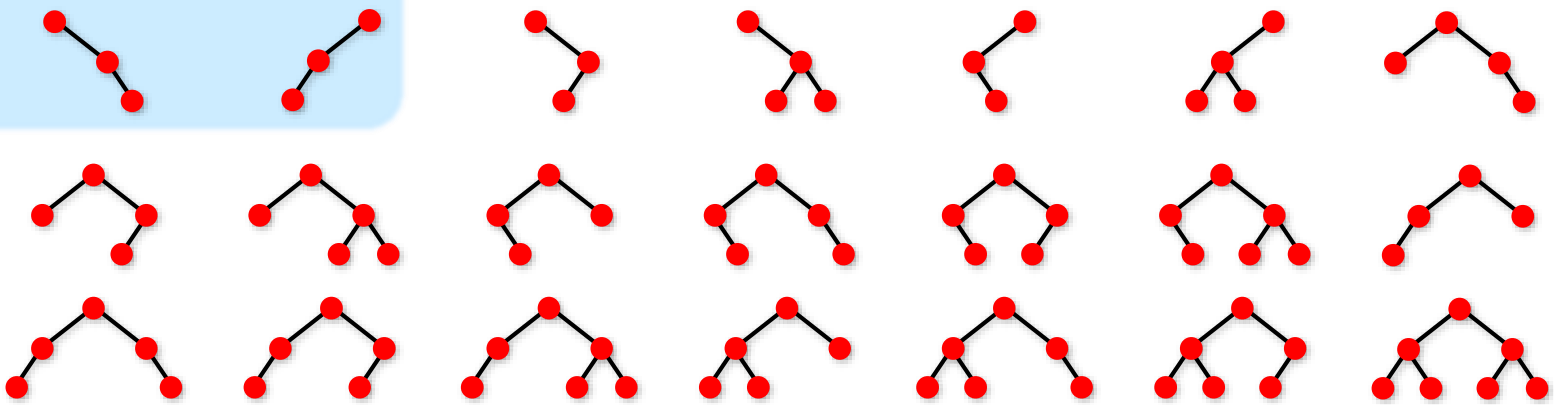
- Meist differenziert man dann zwischen dem linken und rechten Nachfolger (bzw. Unterbaum), so dass z.B. die folgenden Bäume als verschieden angesehen werden:



unmarkierten

Beispiel: Alle 21 Binärbäume der Höhe 2

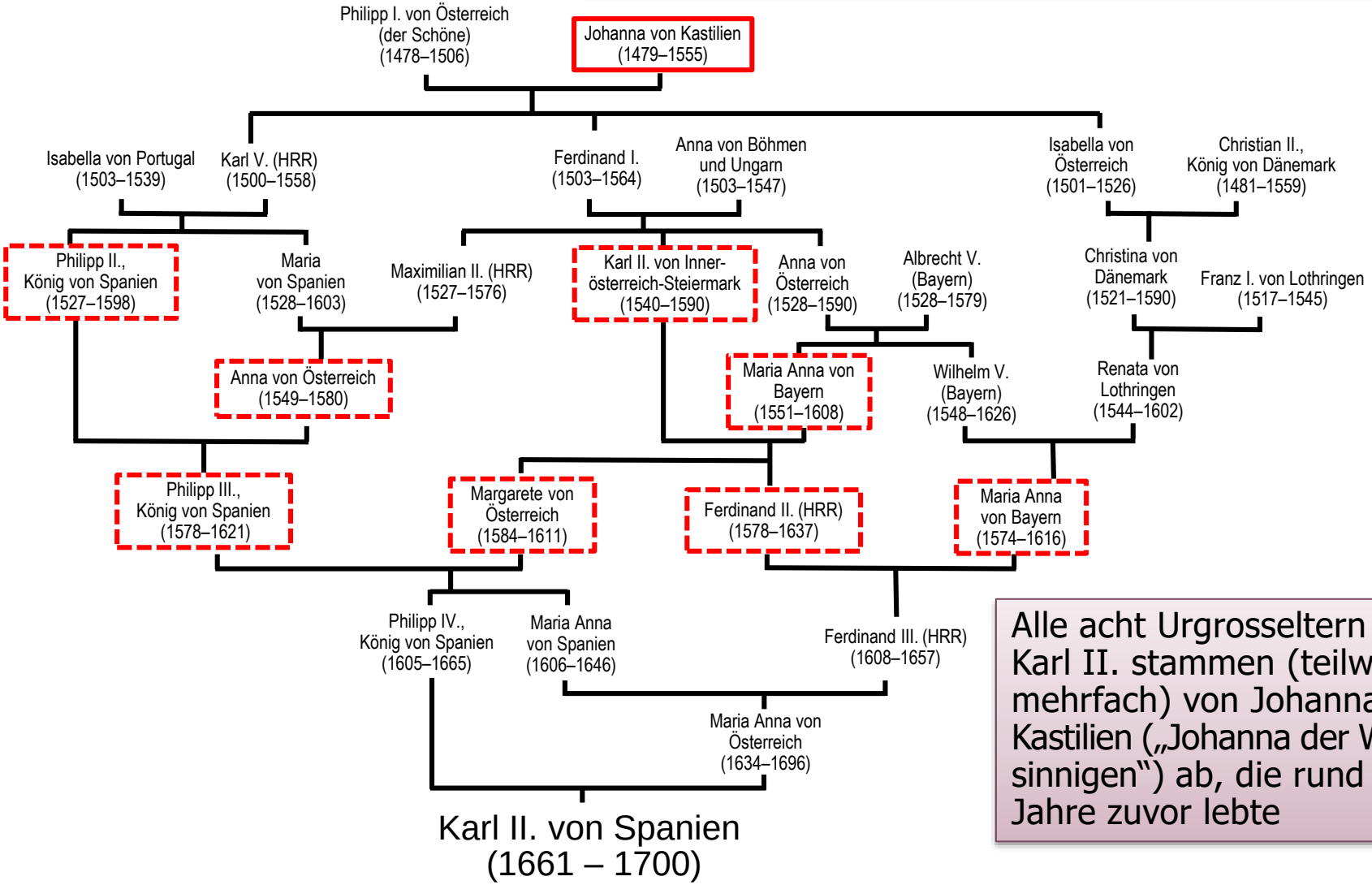
Wie gesagt:
man unterscheidet
„links“ und „rechts“:



- Es gibt 651 solche Binärbäume der Höhe 3,
- 457653 der Höhe 4,
- 210065930571 der Höhe 5,
- 44127887745696109598901 der Höhe 6,
- 1947270476915296449559659317606103024276803403 der Höhe 7,
- **...**

Stammbaum als Binärbaum?

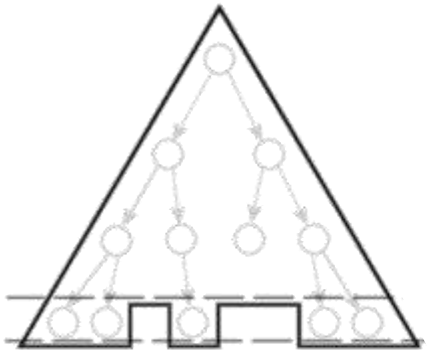
Der Stammbaum einer Person ist kein echter Binärbaum; vor 30 Generationen (etwa 500 bis 1000 Jahren) lebten keine $2^{30} \approx 1$ Milliarde Menschen; es kommt notgedrungen zu einem „Ahnenverlust“



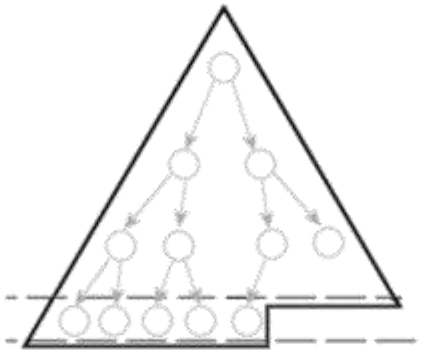
Alle acht Urgrosseltern von Karl II. stammen (teilweise mehrfach) von Johanna von Kastilien („Johanna der Wahnsinnigen“) ab, die rund 200 Jahre zuvor lebte

Ausgeglichene und vollständige Binärbäume

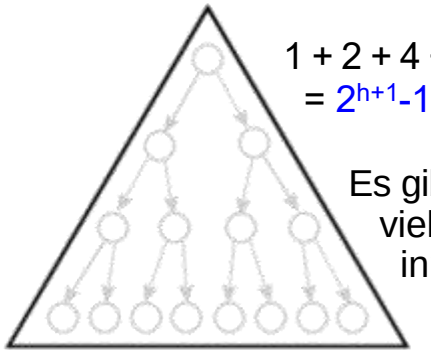
- Oft stört es, wenn Binärbäume lückenhaft oder ausgefranst sind
- Man spricht von vollständigen Binärbäumen, wenn diese für ihre Höhe die volle Anzahl an Knoten haben; dann ist jeder Knoten auf dem letzten Niveau ein Blatt und jeder Knoten auf einem kleineren Niveau hat zwei Nachfolger



ausgeglichen



fast vollständig



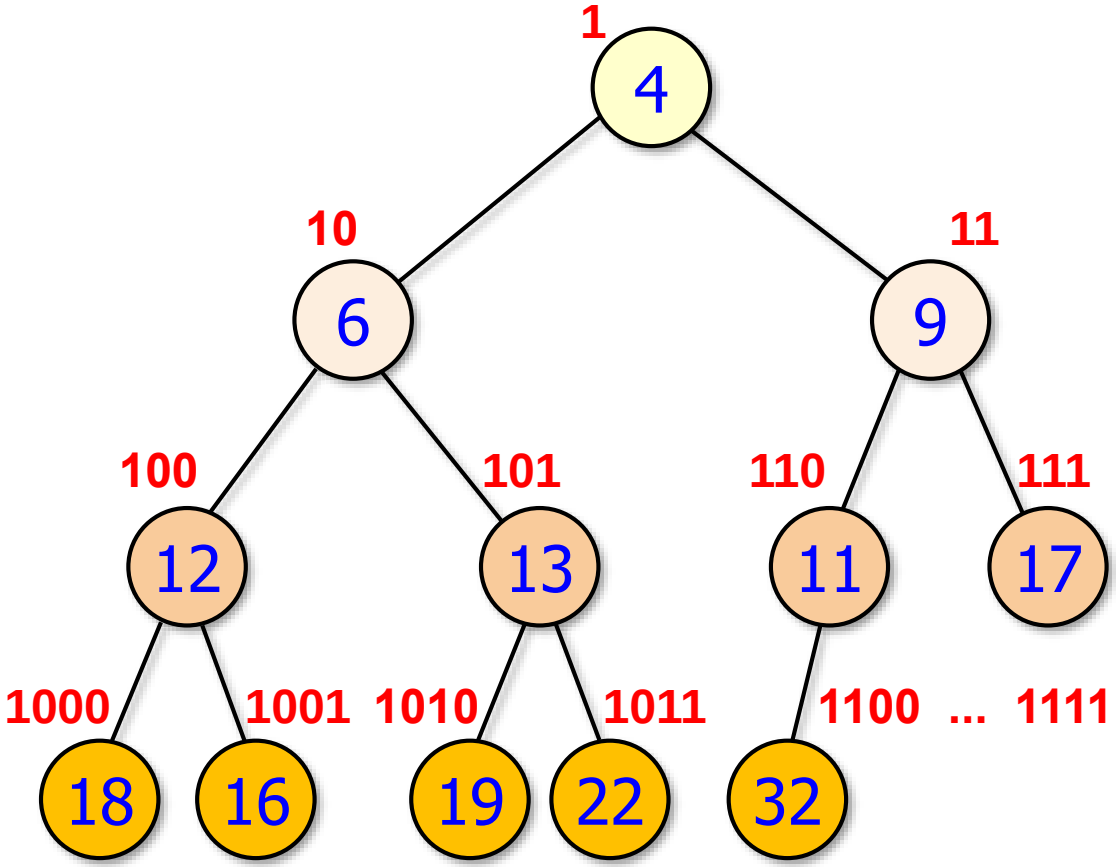
vollständig

$1 + 2 + 4 + \ldots + 2^h$
$= 2^{h+1}-1$ Knoten

Es gibt ~ genauso viele Blätter wie innere Knoten

- Ein ausgeglichener Binärbaum mit k+1 Niveaus ist bis Niveau k vollständig
- Ein fast vollständiger Binärbaum ist ein ausgeglichener Baum dessen letztes Niveau von links her lückenlos ist

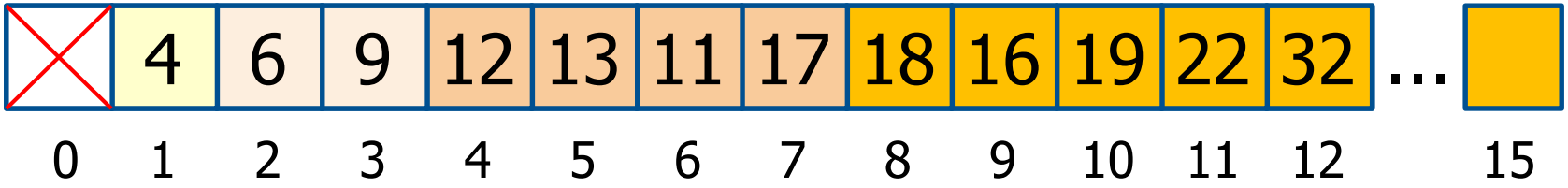
Binärbäume in Arrays



Die Idee besteht darin, die Wurzel an Stelle 1 eines Arrays zu speichern, und die beiden direkten Nachfolger von Stelle i an den Stellen 2i (Wurzel des „linken" Unterbaums) und 2i+1 (Wurzel des „rechten" Unterbaums)

Veranschaulichung mit Dualdarstellung der Indexnummer

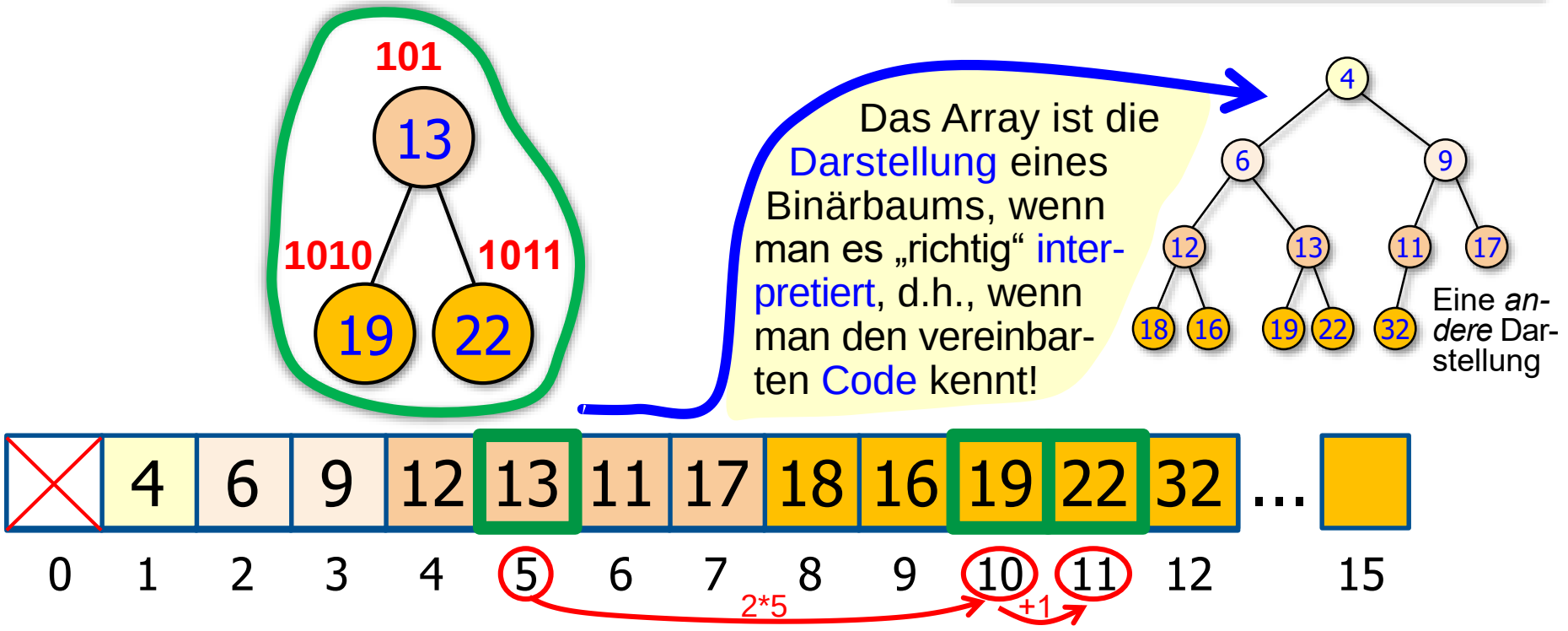
Knoten werden niveauweise sequentiell abgespeichert



Wenn der Baum „Lücken“ hat, also einige Niveaus nicht vollständig besetzt sind, müssen im Array die entsprechenden Positionen explizit als „leer“ markiert werden

Binärbäume in Arrays

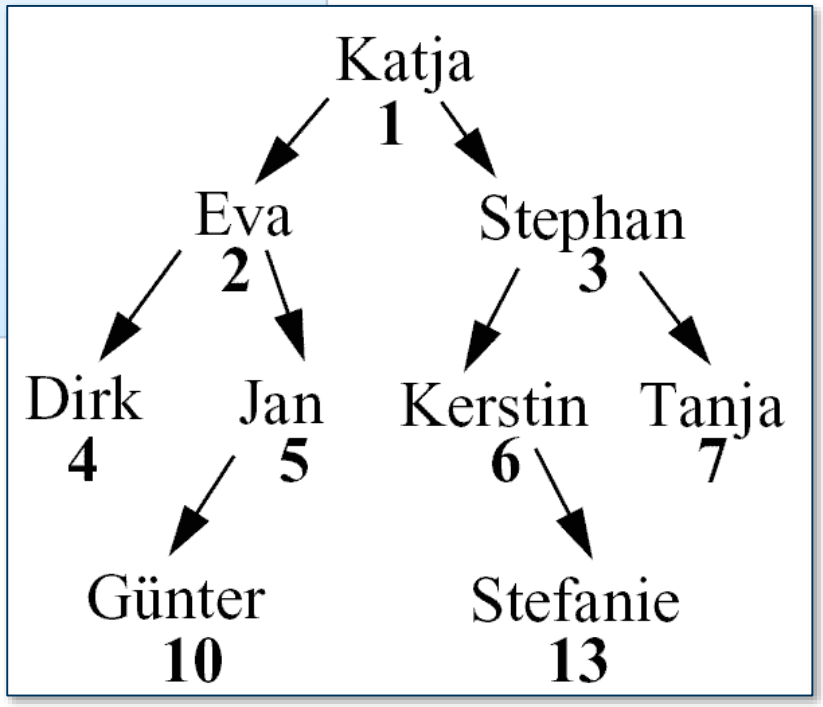
Die Idee besteht darin, die Wurzel an Stelle 1 eines Arrays zu speichern, und die beiden direkten Nachfolger von Stelle i an den Stellen 2i (Wurzel des „linken" Unterbaums) und 2i+1 (Wurzel



Binärbäume in Arrays

```
String [] Baum = new String [n+1];
Baum[1] = "Katja";
Baum[2] = "Eva";
Baum[3] = "Stephan";
Baum[4] = "Dirk";
...
```

richtig interpretiert



→ „Niveauweise“ Speicherung der Baumknoten

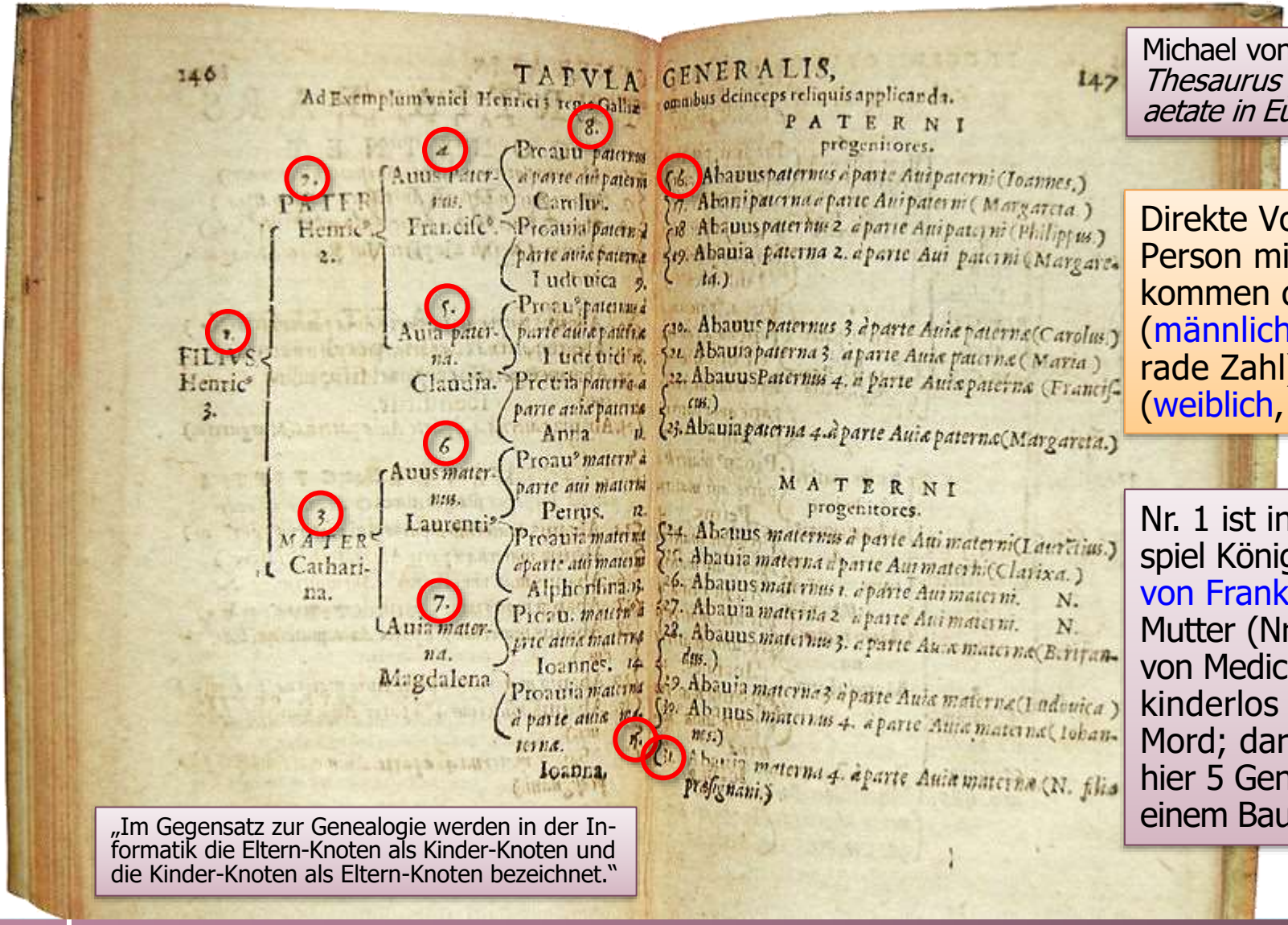
→ Verweise nur implizit

Eindeutige Darstellung von Binärbäumen

- Da bei Binärbäumen im Allg. linker und rechter Nachfolger (bzw. Unterbaum) unterschieden werden, müssen die diversen Darstellungen eindeutig erkennen lassen, bei welchem Nachfolger es sich um den rechten bzw. linken handelt – im generalisierten Sinne gilt dies auch für allgemeine Bäume mit geordneten Nachfolgern
- Insbesondere dann, wenn es nur einen einzigen Nachfolger gibt, soll klar sein, ob es sich dabei um den linken oder rechten handelt
  - Man könnte bei der Klammerdarstellung z.B. A(B(D,.),C(E,F)) bzw. A(B(.,D),C(E,F)) schreiben
  - Bei der Array-Darstellung müssen fehlende (linke und / oder rechte) Nachfolger durch ein explizit als „leer" gekennzeichnetes Element an der entsprechenden Stelle kenntlich gemacht werden
  - Bei der üblichen Graphendarstellung (Wurzel oben), ist „links" bzw. „rechts" kanonisch definiert
  - Auch bei anderen Darstellungen (z.B. Mengendiagramm; Einrückung) muss mit Konventionen oder Markierungen Eindeutigkeit erzielt werden

Kekule-Nummern bei Ahnentafeln

Stephan Kekule (1863–1933), Sohn des Chemikers August Kekulé (Benzolring): Jurist, Heraldiker und Genealoge

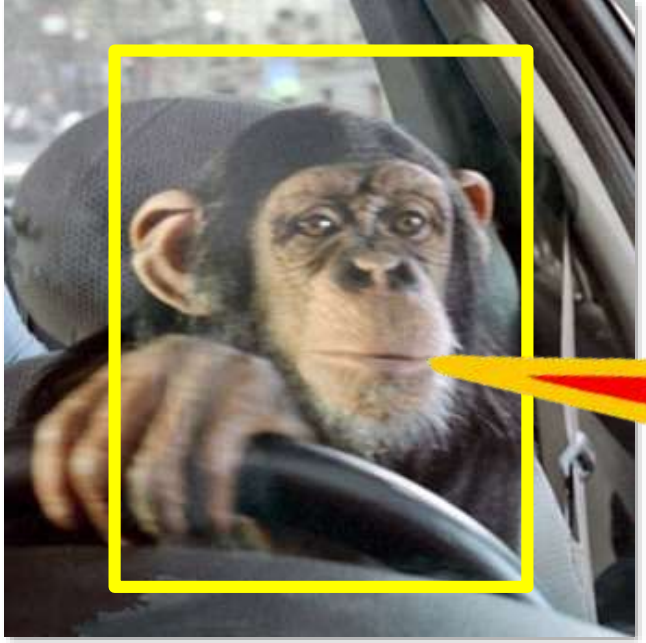


Michael von Aitzing (1590) *Thesaurus principum hac aetate in Europa viventium*

Direkte Vorfahren einer Person mit Index i bekommen die Indizes 2i (männlich, immer gerade Zahl) bzw. 2i+1 (weiblich, ungerade)

Nr. 1 ist in diesem Beispiel König Heinrich III. von Frankreich, dessen Mutter (Nr. 3) Katharina von Medici war; er starb kinderlos 1589 durch Mord; dargestellt sind hier 5 Generationen mit einem Baum der Höhe 4

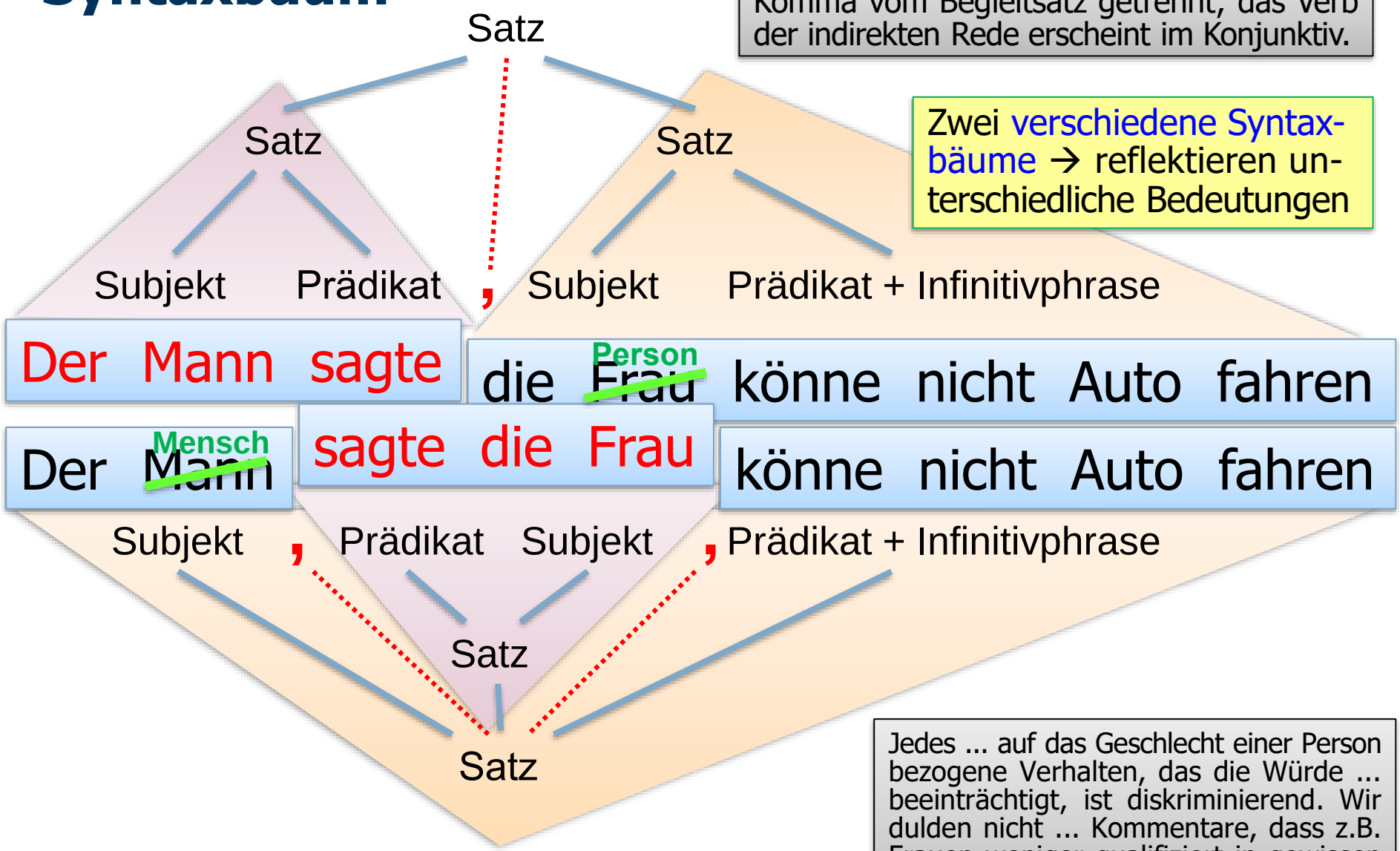
„Im Gegensatz zur Genealogie werden in der Informatik die Eltern-Knoten als Kinder-Knoten und die Kinder-Knoten als Eltern-Knoten bezeichnet."



Der **Affe** sagte die **Ziege** könne nicht Auto fahren

Syntaxbaum

Die indirekte Rede wird immer durch ein Komma vom Begleitsatz getrennt; das Verb der indirekten Rede erscheint im Konjunktiv.



Zwei verschiedene Syntax-bäume → reflektieren un-terschiedliche Bedeutungen

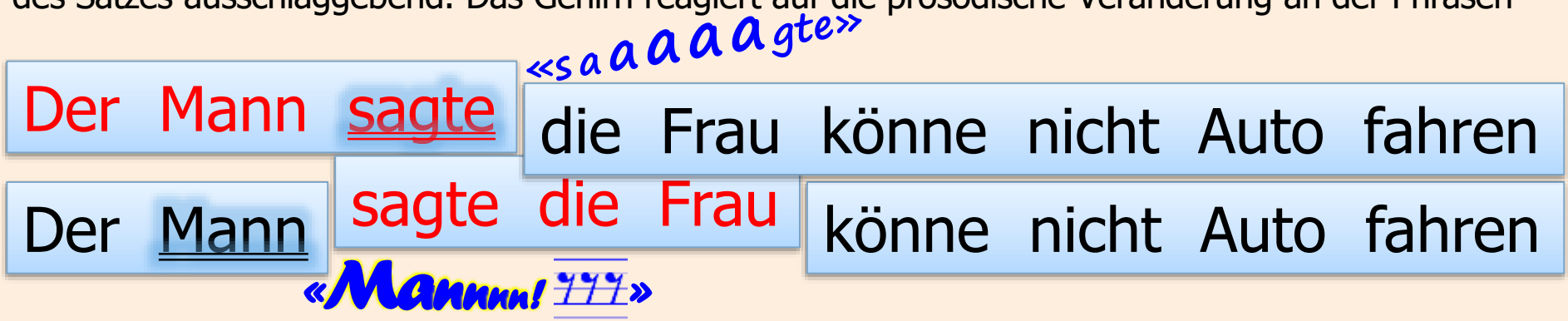
Leider löst die Substitution von „Mann“ / „Frau“ durch „Person“ nicht das Problem der syntaktischen Mehrdeutigkeit!

Jedes ... auf das Geschlecht einer Person bezogene Verhalten, das die Würde ... beeinträchtigt, ist diskriminierend. Wir dulden nicht ... Kommentare, dass z.B. Frauen weniger qualifiziert in gewissen Bereichen seien. [Code of Conduct, D-ITET]

Wenn das Gehirn Sprache verarbeitet, spielen dem kognitiven Modell der Sprachverarbeitung zufolge zwei Dinge eine Rolle: die Grammatik und die Prosodie, die Sprachmelodie. Für das Verständnis eines Satzes ist es nicht nur wichtig, dass er grammatikalisch korrekt formuliert ist, auch die Betonung bestimmt die Bedeutung.

Aus dem Zusammenspiel von grammatischer und prosodischer Information erkennt das Gehirn, welchen Sinn ein Satz ergibt. Bestimmte Areale in der linken Hirnhälfte verarbeiten dabei die Grammatik und Wörter, einzelne Bereiche in der rechten Hirnhälfte erkennen die Sprachmelodie. Zum korrekten Sprachverständnis müssen sich die Hirnhälften also austauschen.

Die Prosodie gibt in den beiden Sätzen jeweils an, wer etwas sagt, und ist somit für das Verstehen des Satzes ausschlaggebend. Das Gehirn reagiert auf die prosodische Veränderung an der Phrasen-



grenze (im Schriftsprachlichen markiert durch das Komma) mit einer spezifischen Hirnreaktion im ereigniskorrelierten Hirnpotenzial, das mittels der Elektroenzephalographie (EEG) erstmals gemessen werden konnte. Für den normal gesprochenen Satz, der neben der Satzmelodie auch syntaktische und semantische Information enthält, sind beim Verstehen die linke und die rechte Hemisphäre involviert. Die Verarbeitung der Satzmelodie alleine wird vornehmlich von der rechten Hemisphäre geleistet. Normales Sprachverstehen setzt eine zeitliche Feinabstimmung der Hirnaktivitäten in verschiedenen Arealen der linken und rechten Hemisphäre voraus. Ist eine der Hirnregionen durch eine Erkrankung geschädigt oder führt eine Hirnerkrankung zu einer zeitlichen Verzögerung einzelner Prozesse, so kann normales Sprachverstehen oder auch das Produzieren kohärenter Äußerungen nicht mehr stattfinden.

Text dieser Slide zitiert aus: www.mpg.de/541750/pressemitteilung20070116 sowie Tobias Bonhoeffer, Peter Gruss: Zukunft Gehirn, C.H.Beck, 2011

Das Geheimnis der menschlichen Sprache ist die Grammatik, die es möglich macht, mit einer begrenzten Zahl von Worten eine unbegrenzte Zahl von Sachverhalten auszudrücken. Bereits im 19. Jahrhundert entdeckte man an Gehirnen von Patienten, die zu Lebzeiten unter Sprachstörungen litten, zwei Areale, die bei der Sprachverarbeitung eine zentrale Rolle spielen. Beide lagen in der linken Hirnhälfte, das Broca-Areal und das Wernicke-Areal.

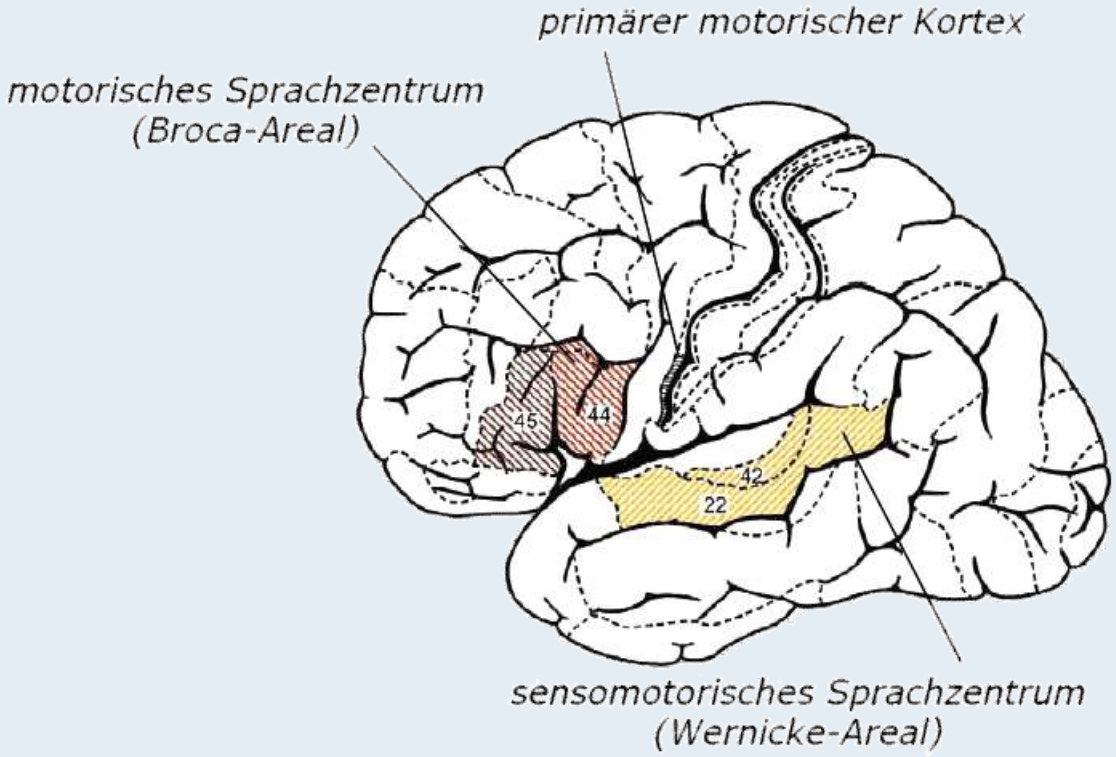
Am Max-Planck-Institut für Kognitions- und Neurowissenschaften versucht man den Geheimnissen der Sprachverarbeitung auf die Spur zu kommen. Das geht eigentlich erst seit rund 20 Jahren mit bildgebenden Verfahren. So war es möglich, das Gehirn bei der Arbeit zu beobachten. Die ursprüngliche Annahme, dass Sprache nur in der linken Hirnhälfte verarbeitet wird, konnte widerlegt werden. Sonja A. Kotz, MPI Kognitions- u. Neurowissenschaften: „Bei der Sprachverarbeitung kommunizieren sowohl die linke als auch die rechte Gehirnhälfte dergestalt, dass die Klangfarbe der Sprache durch rechts gefördert wird, während die eher analytischen Aspekte der Sprache links verarbeitet werden. Die Kommunikation dieser Gehirnhälften kommt primär über die Kommissur zustande, die die beiden Gehirnhälften verbindet."

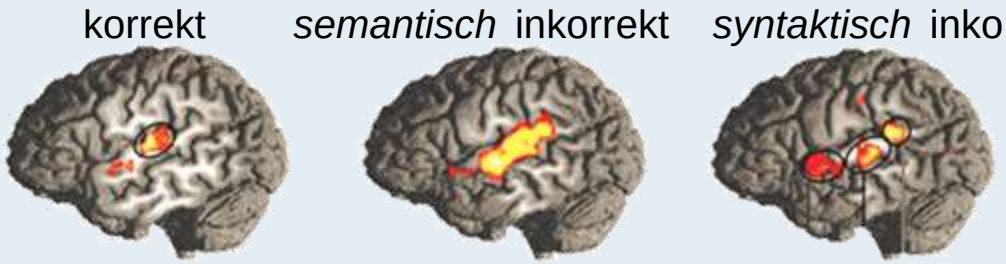
Sprache ist eine Hochleistung, an der das gesamte Gehirn mit parallelen Netzwerken arbeitet. Wird ein gesprochener Satz gehört, zum Beispiel: 'Der Mann sagt, die Frau kann nicht Auto fahren', wird die Akustik im primären auditorischen Kortex erfasst. Innerhalb von weniger als 200 Millisekunden werden dann Aspekte der Sprachmelodie in der rechten Hirnhemisphäre verarbeitet, denn es könnte ja auch das Gegenteil gemeint sein: 'Der Mann, sagt die Frau, kann nicht Auto fahren'. Parallel dazu wird die Syntax, also die Grammatik des Satzes, in der linken Hirnhälfte im Broca-Areal analysiert.

Dann aber wird es wesentlich komplizierter. Der Sinn der Worte muss erfasst werden. Die Semantik der Worte setzt sich aus vielen Bedeutungsebenen, Erinnerungen und Emotionen zusammen. [...] Bislang konnten nur wenige Details der Sprachverarbeitung entdeckt werden. So reagiert das Broca-Areal mit einer gesteigerten Aktivität, wenn ein grammatikalischer Fehler auftaucht. Die Reaktion geschieht so schnell, dass es scheint als würde das Gehirn eine Erwartungshaltung haben. Das geschieht im Übrigen auch, wenn die Testpersonen aufgefordert wird, nicht auf die Grammatik zu achten.

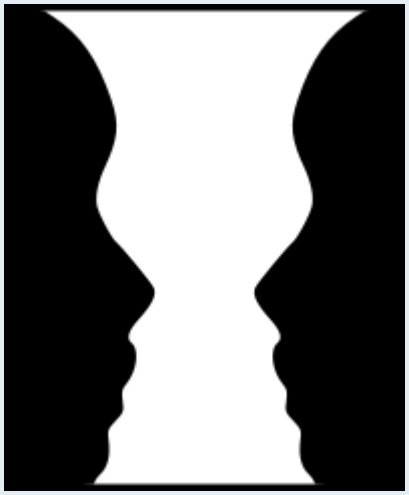
Text dieser Slide zitiert aus www.wissen.de/video/das-geheimnis-der-sprache

Sprachrelevante Hirnareale beim Menschen





Quelle: Max-Planck-Institut für Kognitions- und Neurowissenschaften, Leipzig



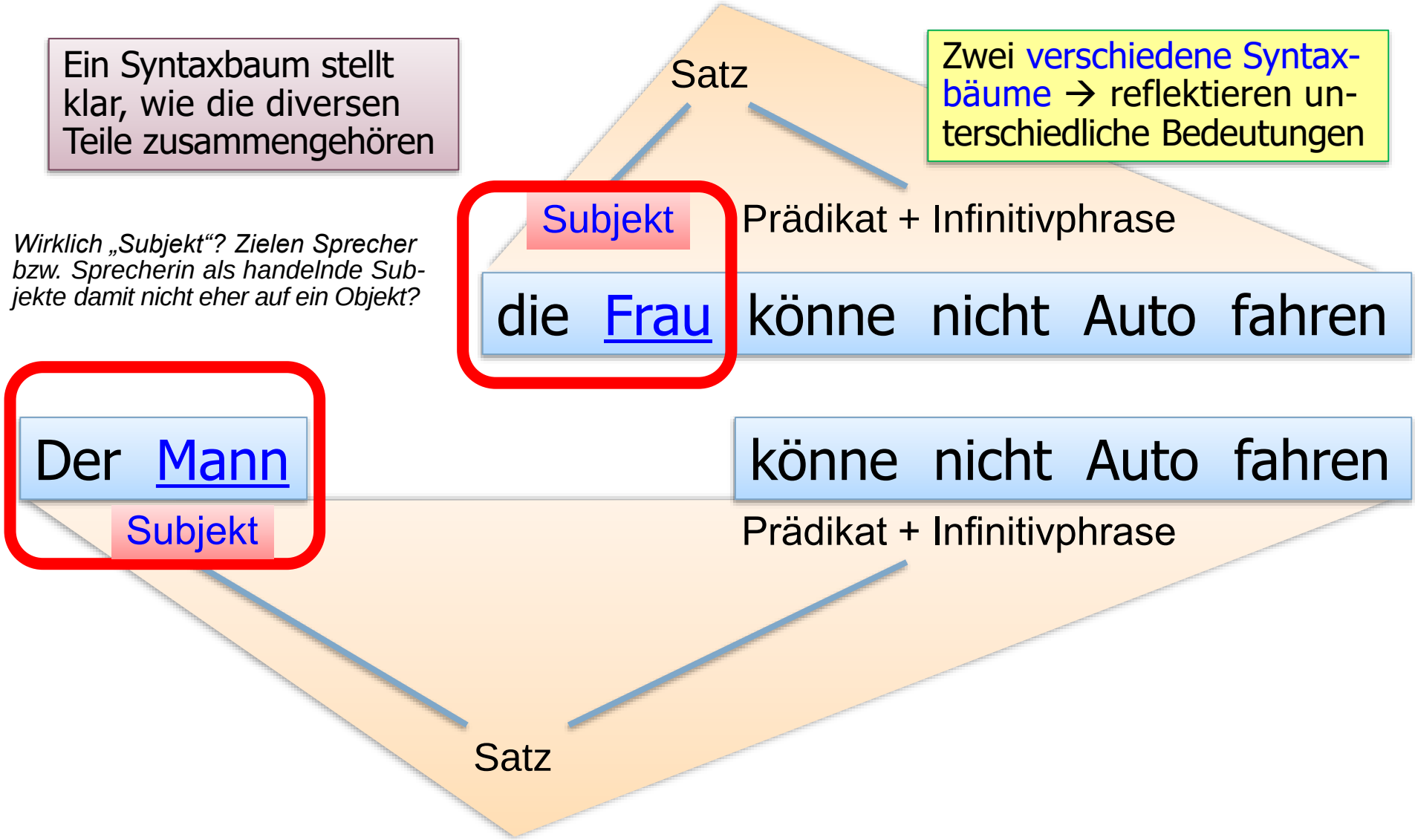
Das Gehirn versucht, die Sinneseindrücke sinnvoll zu verarbeiten und wird dadurch von Erwartungen aus dem bisher Erfahrenen und Gelernten gesteuert – unbewusste „Vor“urteile spielen dabei eine wesentliche Rolle. Ob wir Mehrdeutigkeiten überhaupt erkennen bzw. wie schnell, verrät in dieser Hinsicht etwas über unsere innere Disposition.

Syntaxbaum

Ein Syntaxbaum stellt klar, wie die diversen Teile zusammengehören

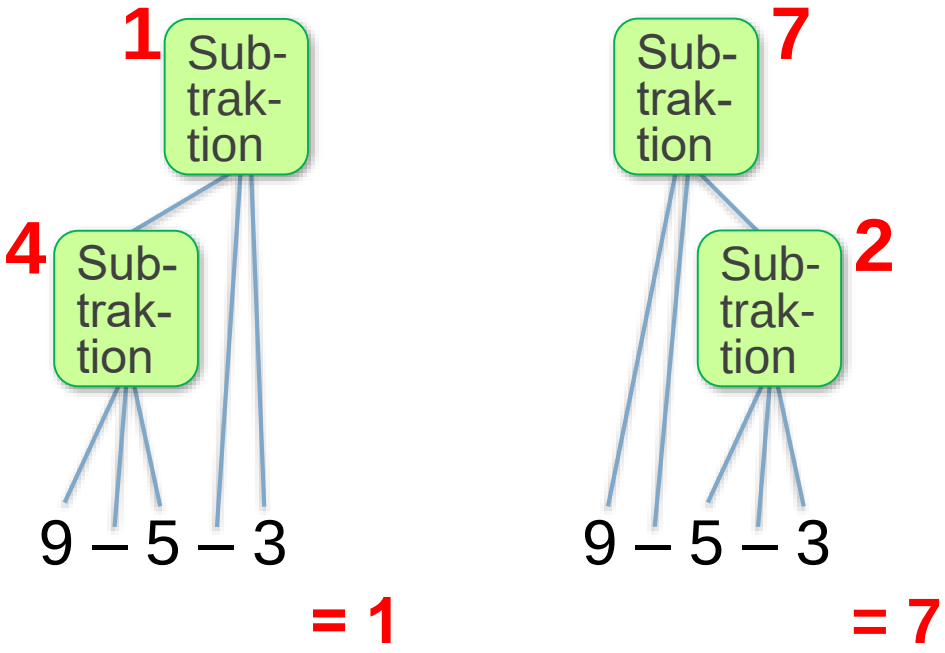
Zwei verschiedene Syntax-bäume → reflektieren un-terschiedliche Bedeutungen

Wirklich „Subjekt“? Zielen Sprecher bzw. Sprecherin als handelnde Sub-jekte damit nicht eher auf ein Objekt?



Mehrdeutigkeiten

Ein Syntaxbaum stellt klar, wie die diversen Teile zusammengehören



- Zwei verschiedene Syntaxbäume → Mehrdeutigkeit
  - Schon klar, aber <u>was</u> ist denn nun bei 9−5−3 gemeint?
- Mehrdeutigkeit beseitigen

  → Konvention (z.B. Linksassoziativität: „Es soll die linke Variante gelten“)

  → Klammern als Interpretationsvorschrift: (9−5)−3 bzw. 9−(5−3)

  Eigentlich dann sogar vollständig geklammert: ((9−5)−3) bzw. (9−(5−3))

Mehrdeutigkeiten

Er versprach, mir jedes Jahr ein neues Auto zu kaufen.
Er versprach mir, jedes Jahr ein neues Auto zu kaufen.
Er versprach mir jedes Jahr, ein neues Auto zu kaufen.

Er versprach heimlich, abzureisen.
Er versprach, heimlich abzureisen.

Hochzeit abgesagt: Er wollte, sie nicht.
Hochzeit abgesagt: Er wollte sie nicht.

Der brave Mensch denkt an sich selbst zuletzt.
Der brave Mensch denkt an sich, selbst zuletzt.

Der König begnadigt nicht, hinrichten!
Der König begnadigt, nicht hinrichten!

Wir bitten unsere Gäste, nicht zu rauchen.
Wir bitten, unsere Gäste nicht zu rauchen.

→ Satzzeichen (z.B. Kommas) als Interpretationsvorschrift

Syntaxbaum eines Programms

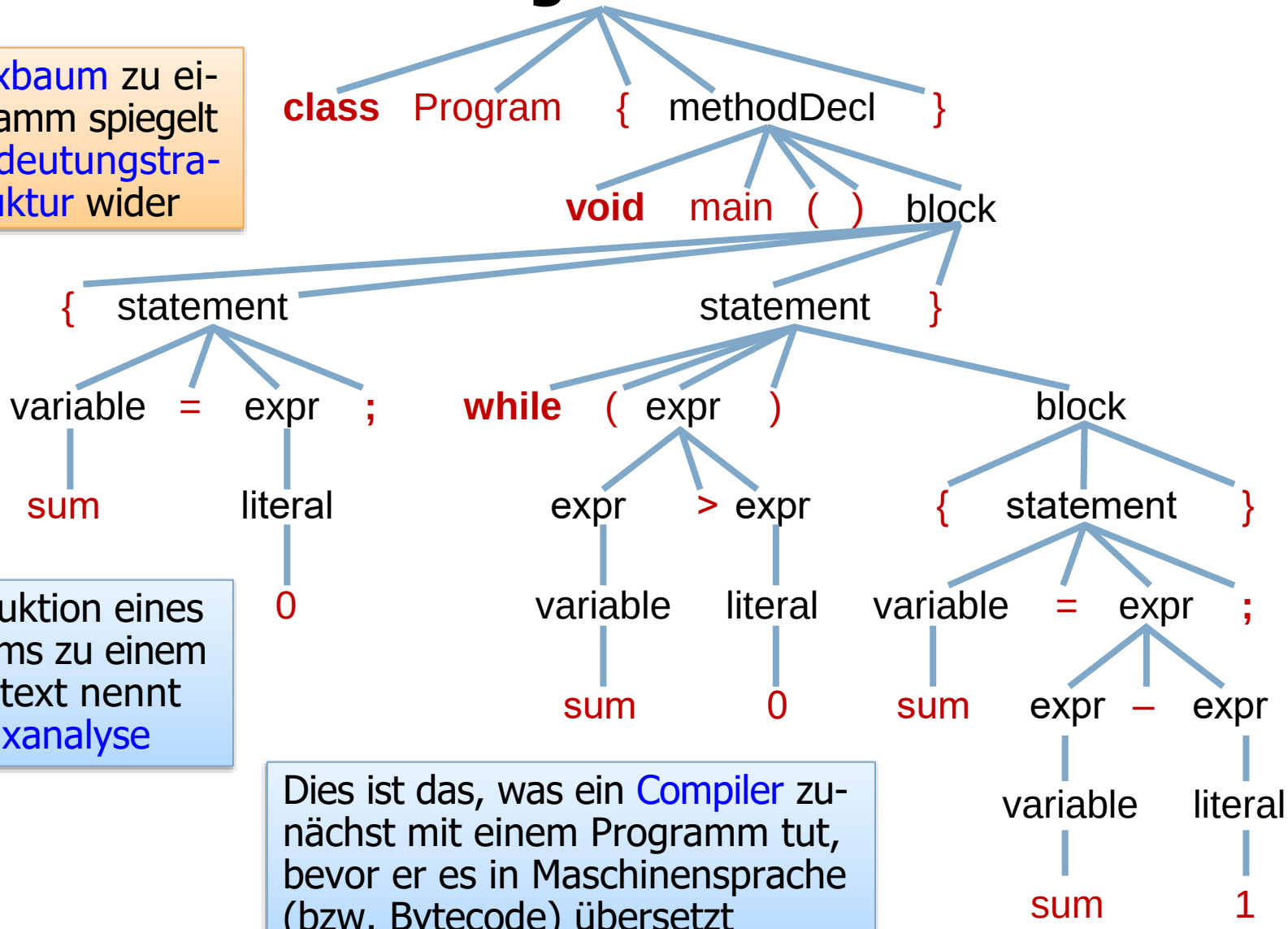
Wir zerpflücken das Programm

class Program {**void** main ( )}

{ sum = 0 ; **while** (sum> 0) {sum = sum – 1 ;} }

Syntaxbaum eines Programms

Der Syntaxbaum zu einem Programm spiegelt dessen bedeutungstragende Struktur wider



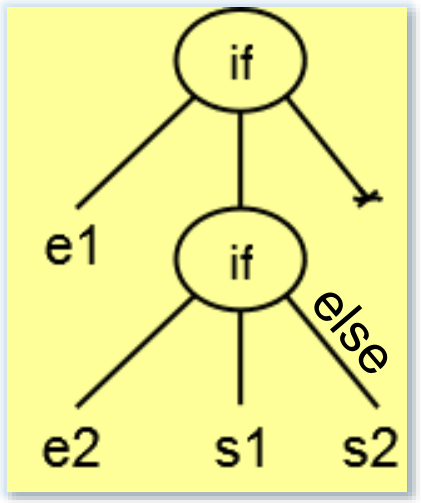
Die Konstruktion eines Syntaxbaums zu einem Programmtext nennt man Syntaxanalyse

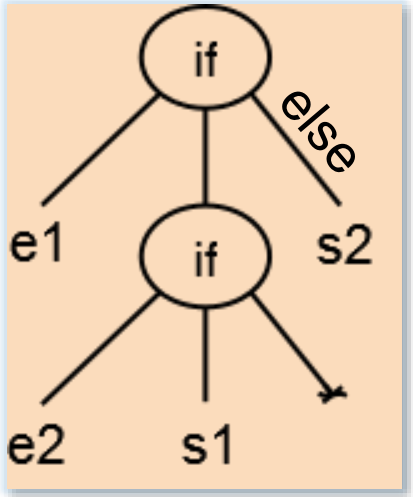
Dies ist das, was ein Compiler zunächst mit einem Programm tut, bevor er es in Maschinensprache (bzw. Bytecode) übersetzt

Die „Dangling-Else“-Mehrdeutigkeit

- Eine if-Anweisung kann optional einen else-Teil haben – zu folgendem Programmfragment gibt es daher zwei verschiedene Syntaxbäume:

```
if (age > 8)
if (age > 64) return("
        Seniorenrabatt")
else return("umsonst!");
…
```



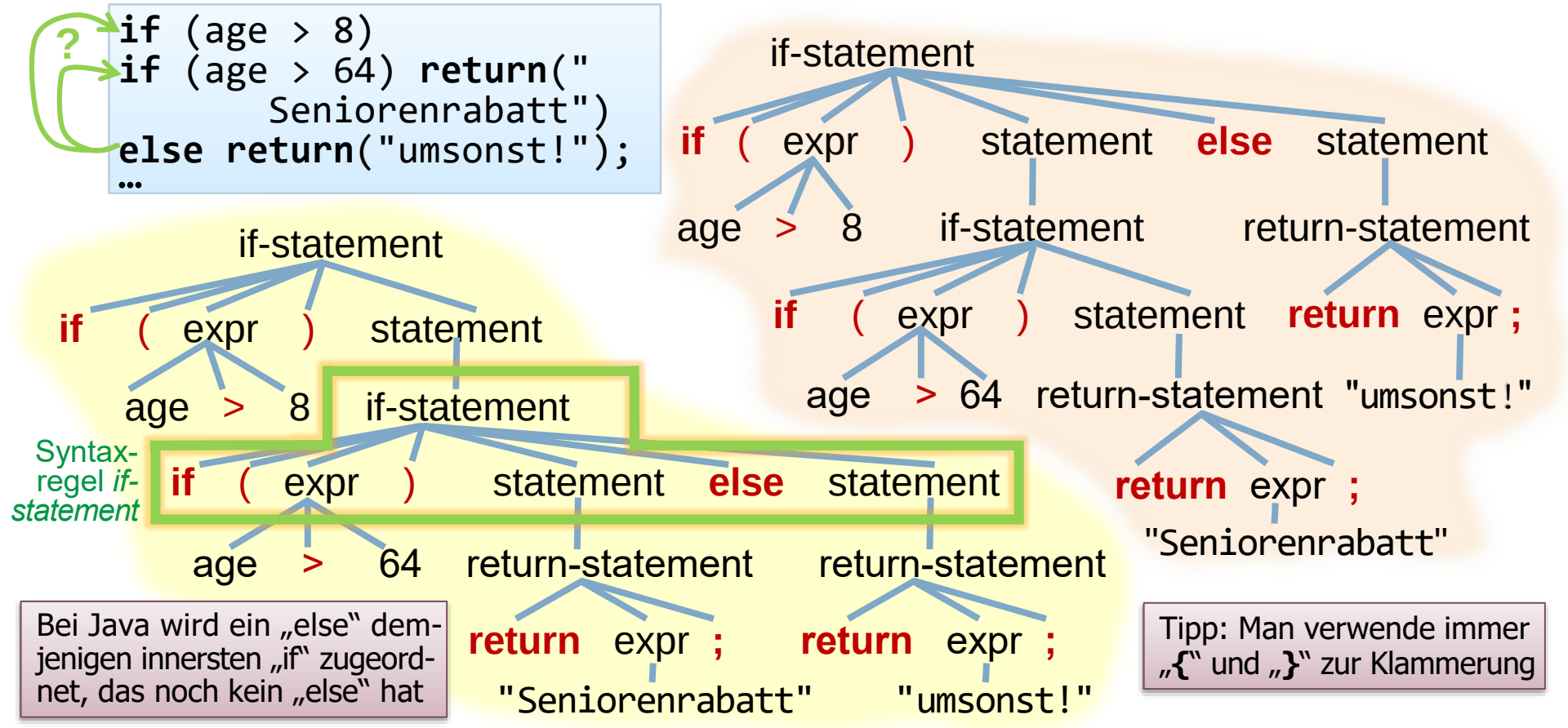


Tipp: Man verwende immer „**{**“ und „**}**“ zur Klammerung

Die „Dangling-Else“-Mehrdeutigkeit

- Eine if-Anweisung kann optional einen else-Teil haben – zu folgendem Programmfragment gibt es daher zwei verschiedene Syntaxbäume:

```
if (age > 8)
if (age > 64) return("
        Seniorenrabatt")
else return("umsonst!");
…
```



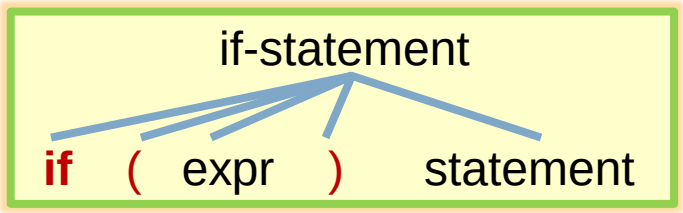
Bei Java wird ein „else“ demjenigen innersten „if“ zugeordnet, das noch kein „else“ hat

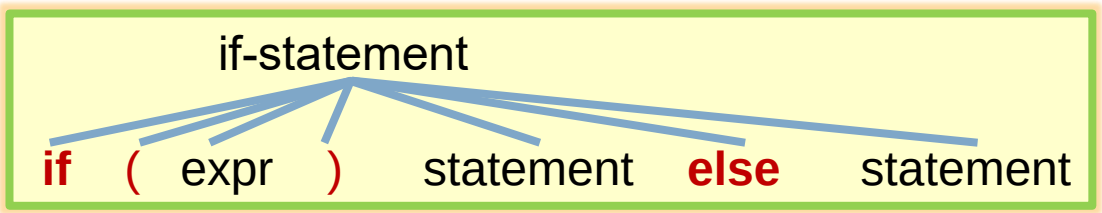
Tipp: Man verwende immer „{“ und „}“ zur Klammerung

- Diese reflektieren eine unterschiedliche Semantik (Zugehörigkeit des „else“)!

Syntaxregeln

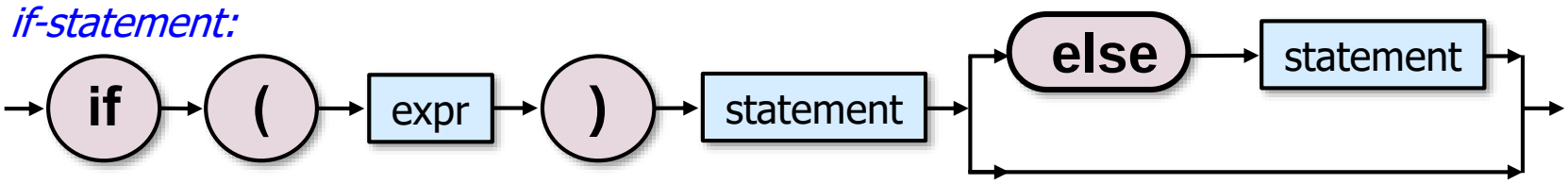
- Syntaxregeln geben an, aus welchen Bestandteilen ein bestimmtes Sprachkonstrukt aufgebaut ist, z.B.:





- Ein Syntaxbaum entsteht aus ineinander eingesetzten Regeln

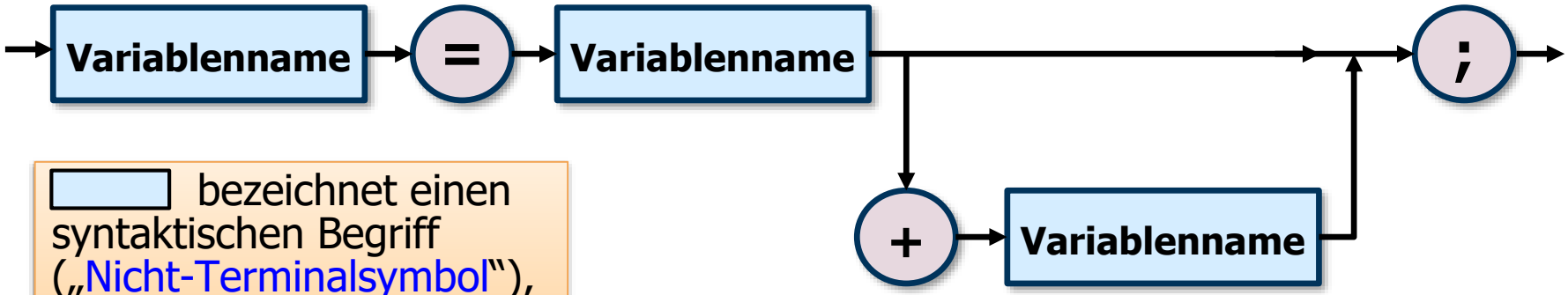
- Regeln obiger Art lassen sich auf verschiedene Weise formulieren
  - Ein Formalismus dafür stellt bspw. EBNF dar, aus „Informatik I" bekannt
  - Etwas eingängiger lassen sich Syntaxregeln in Diagrammform notieren, z.B.:



Syntaxdiagramme

Definieren die Syntax einer Sprache, also die Menge aller (korrekten) Syntaxbäume

- Zweck:
  - Nur syntaktisch korrekte Programme generieren bzw.
  - Überprüfen, ob ein Programm syntaktisch korrekt ist:
    → Versuchen, es mit einem Syntaxdiagramm zu generieren
- Ein einfaches Beispiel (Syntax einer Zuweisung):

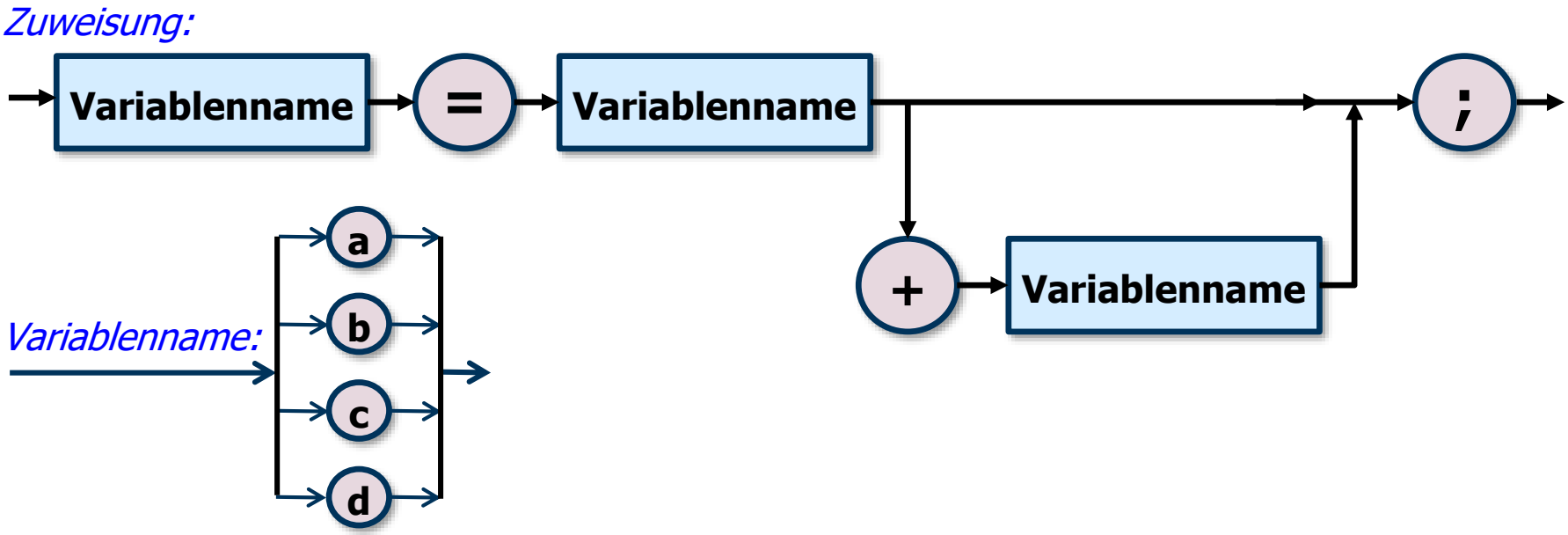


bezeichnet einen syntaktischen Begriff („Nicht-Terminalsymbol"), welcher seinerseits weiter expandiert werden muss

bezeichnet ein „Terminal-symbol", welches so „wörtlich" in einem Programm vorkommen kann

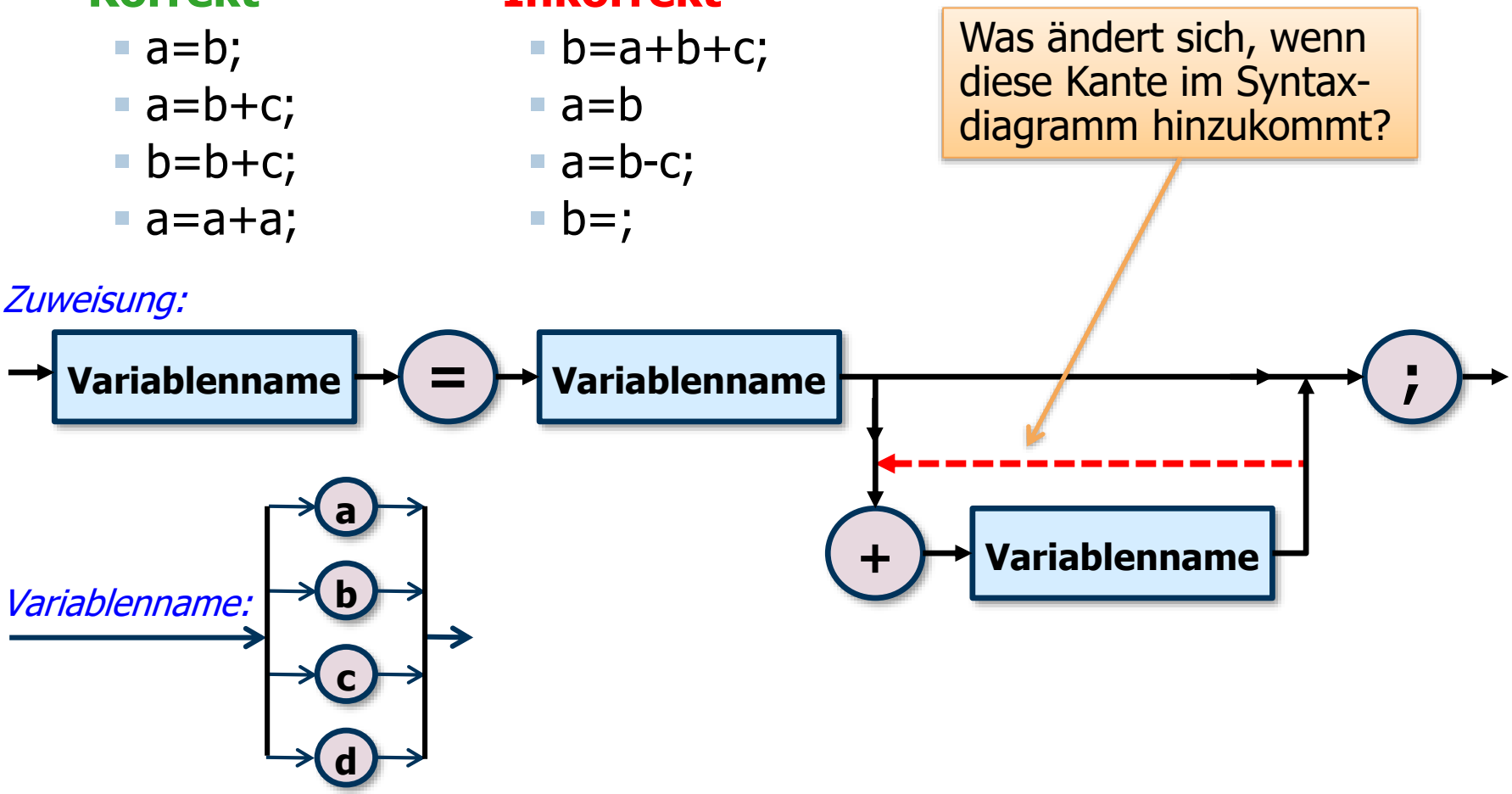
Durchlaufen von Syntaxdiagrammen

- In Pfeilrichtung durchlaufen
- Bei Verzweigungen „zweckmässige" Richtung wählen
- Durchlaufene Terminalsymbole  aufschreiben
- Wenn ein Nicht-Terminal  getroffen wird, in das zugehörige Teildiagramm „abtauchen" und nach dem „Auftauchen" weitermachen



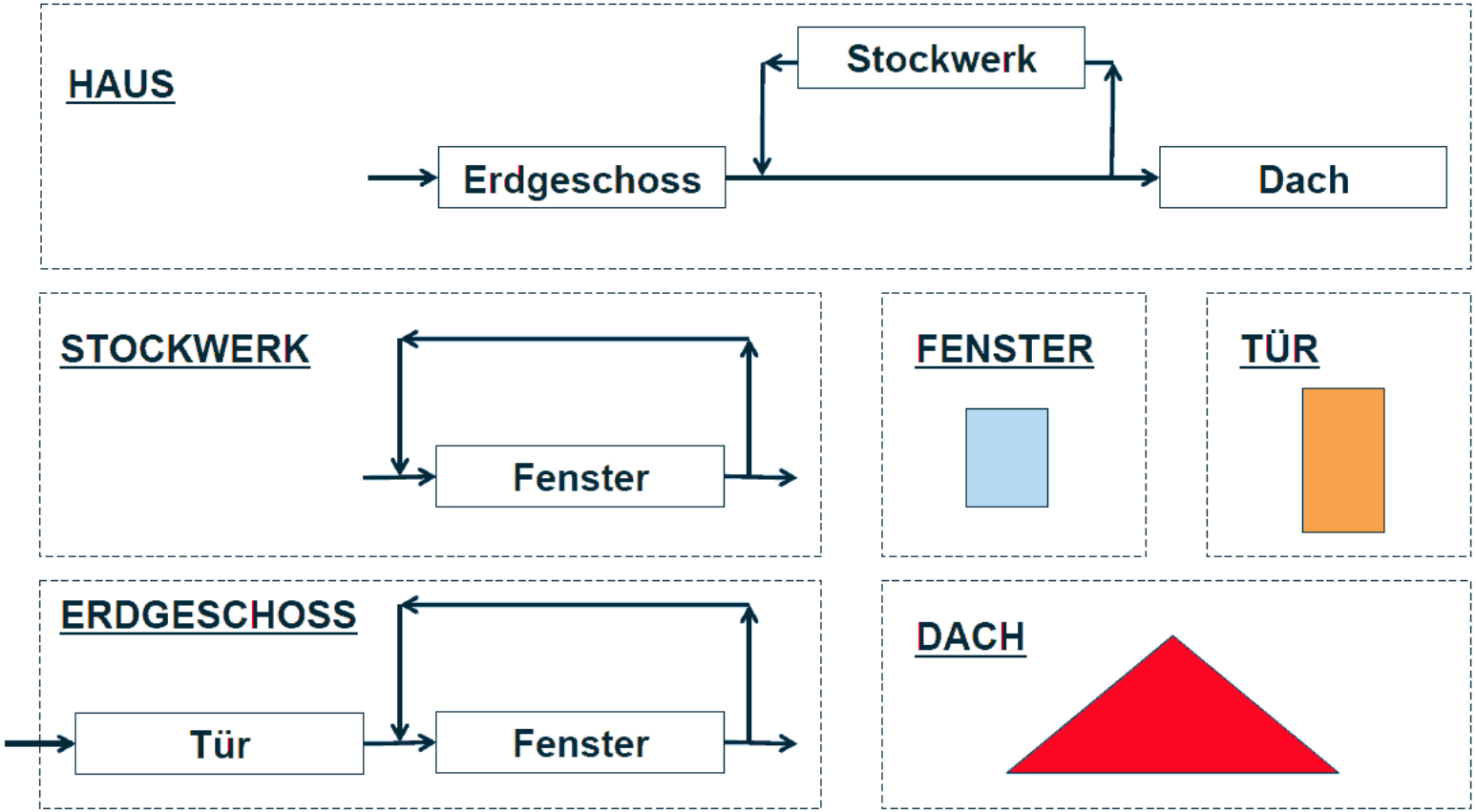
Durchlaufen von Syntaxdiagrammen

- Entsprechend diesem Syntaxdiagramm:
  - **Korrekt**
    - a=b;
    - a=b+c;
    - b=b+c;
    - a=a+a;
  - **Inkorrekt**
    - b=a+b+c;
    - a=b
    - a=b-c;
    - b=;



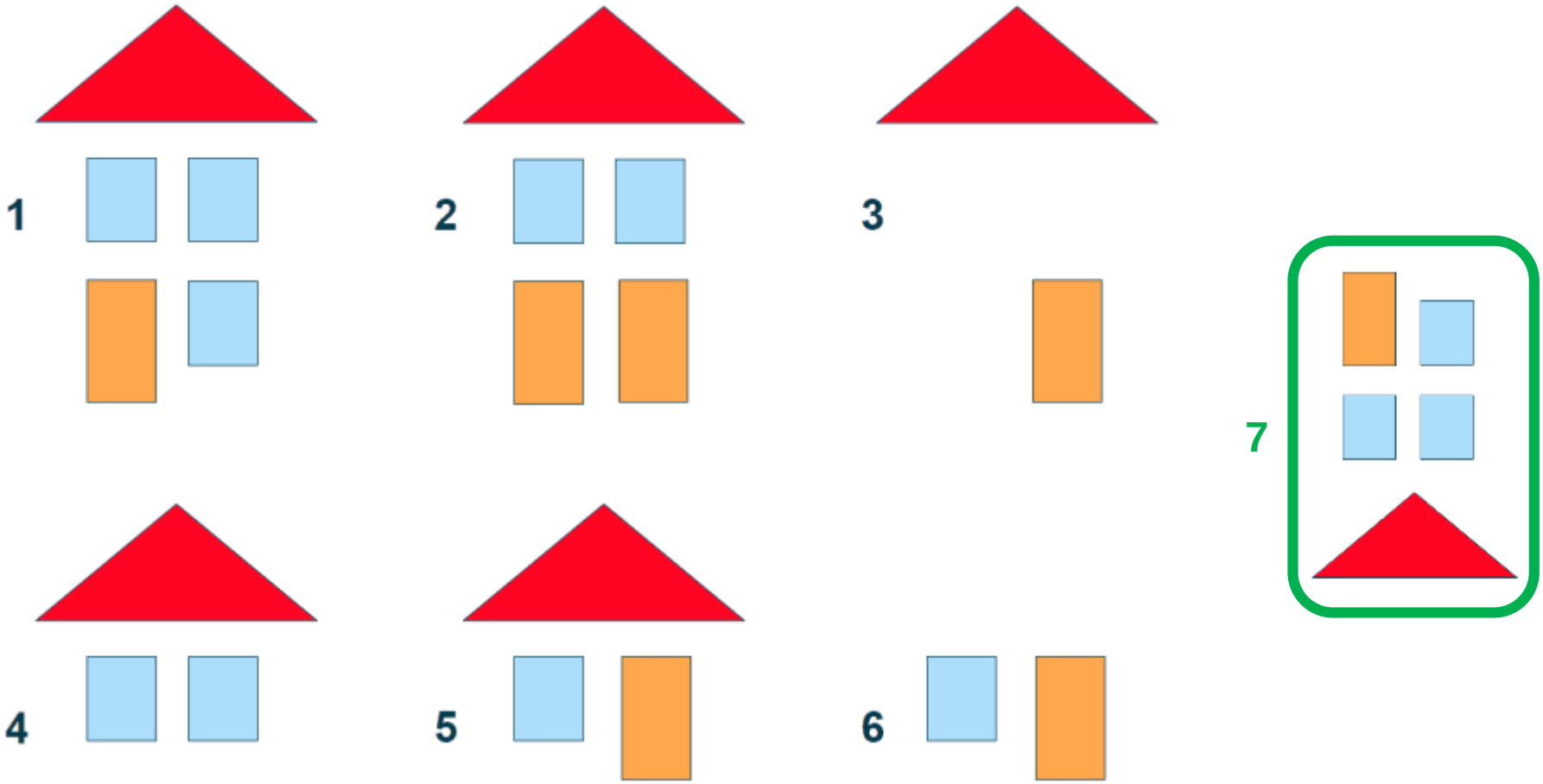
Zum selber üben: Hausbau

Architekturregeln für den Hausbau:



Zum selber üben: Hausbau (2)

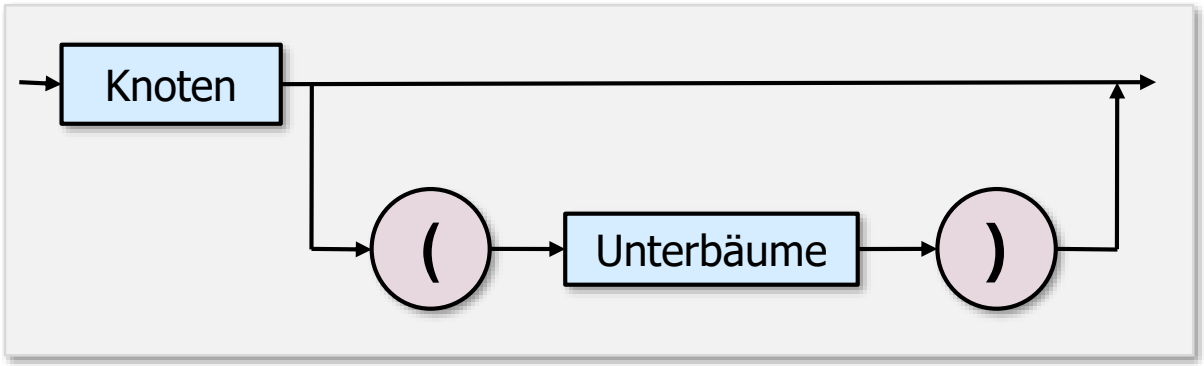
Was ist ein Haus entsprechend den Architekturregeln? Wieso?



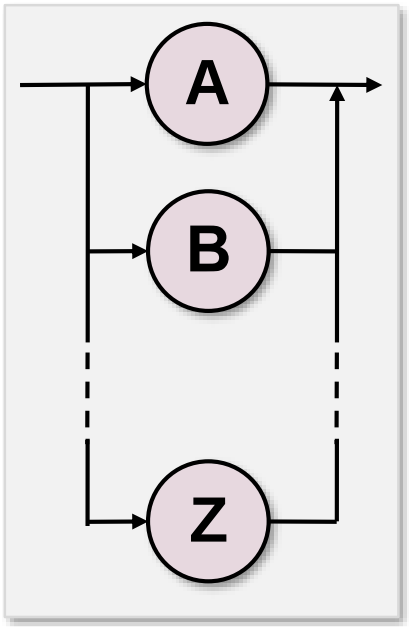
Ein Syntaxdiagramm-Beispiel: Klammerdarstellung eines Baums

Beispiel: A (B (D) , C (E , F))

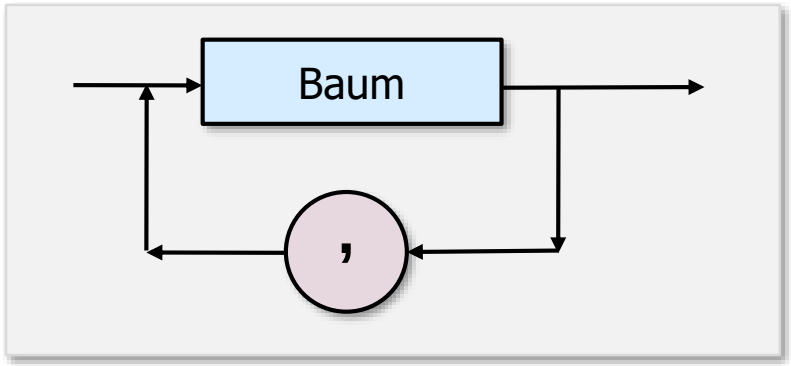
Baum:



Knoten:

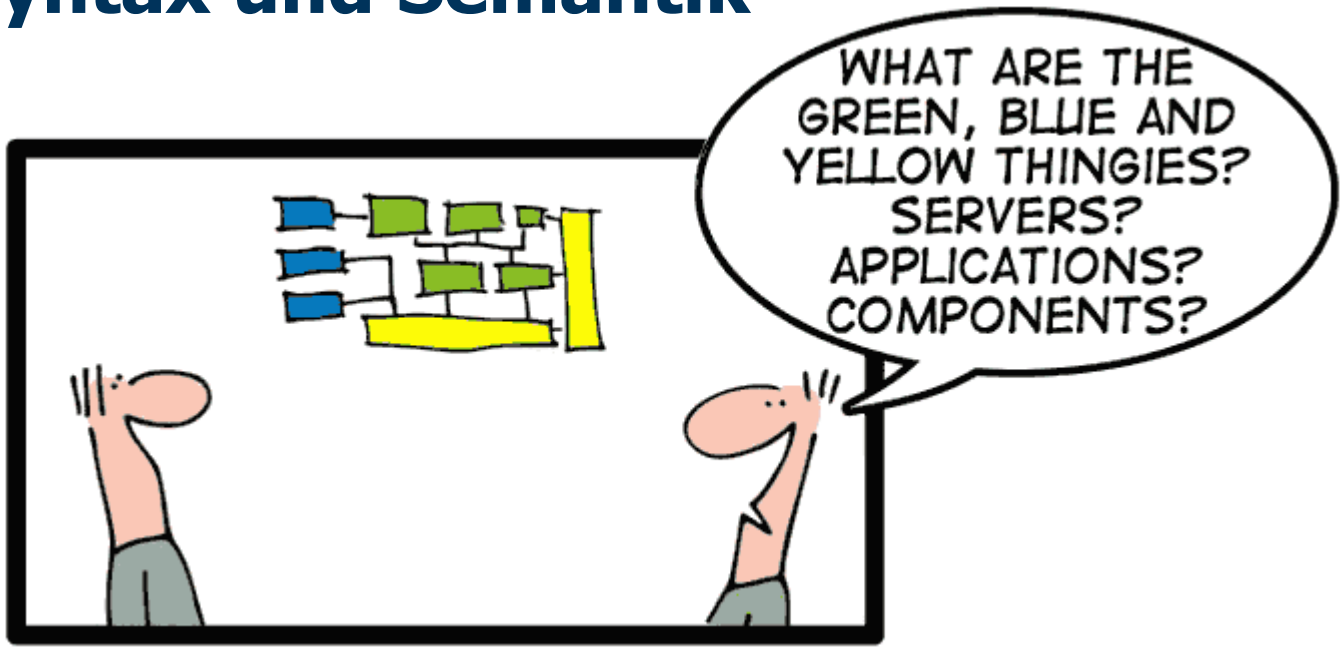


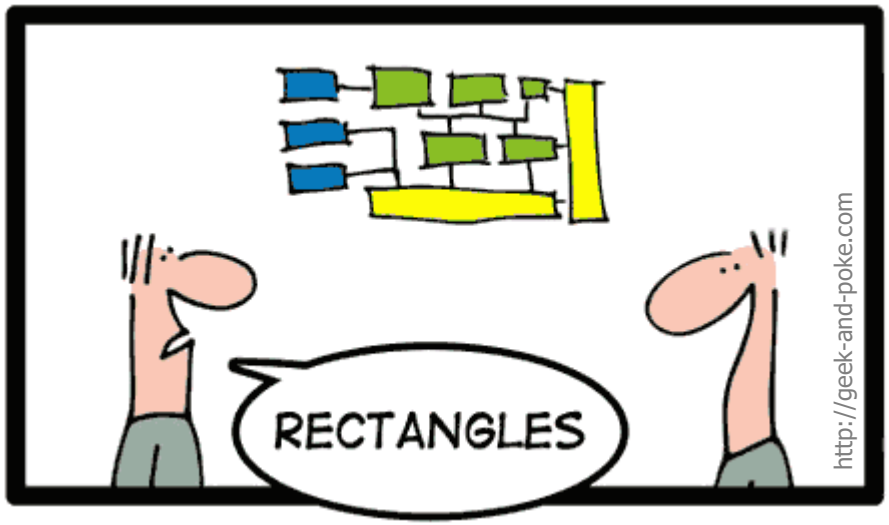
Unterbäume :



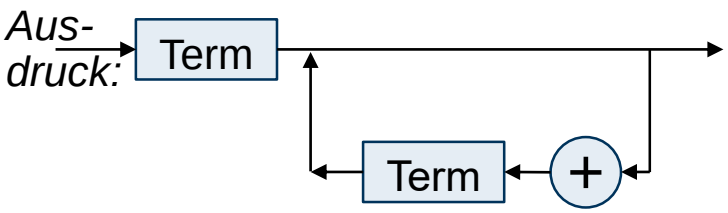
Wie könnte man Binärbäume durch ein Syntaxdiagramm darstellen?

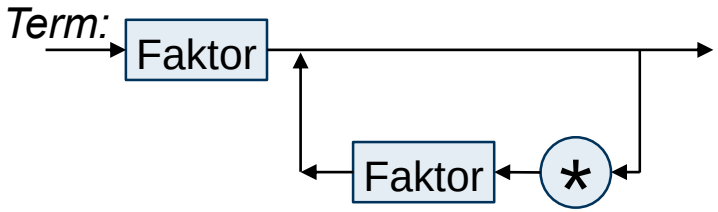
Syntax und Semantik

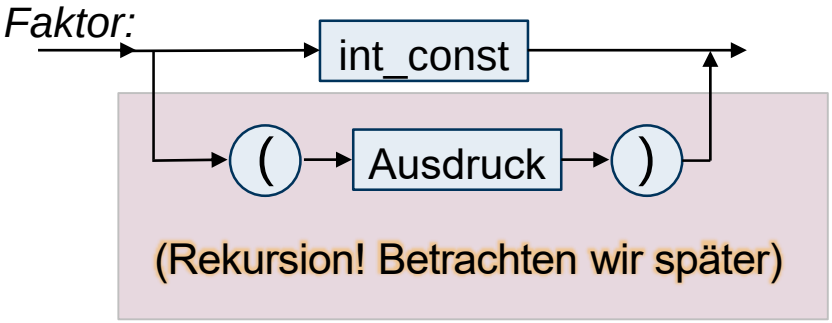


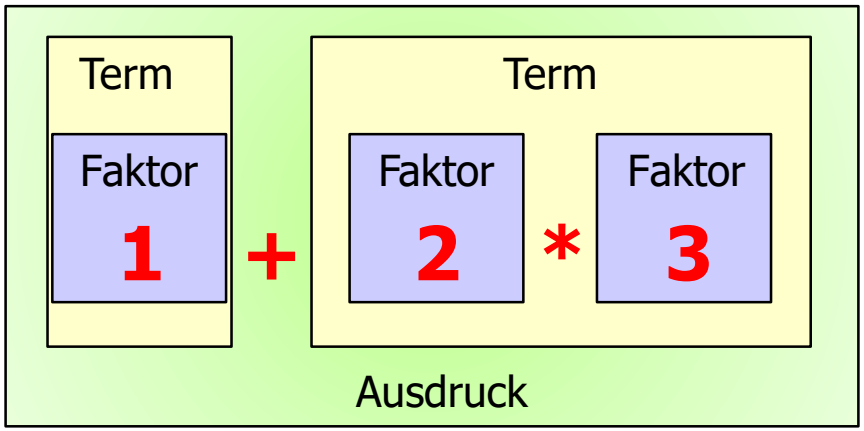


Syntax von arithmetischen Ausdrücken









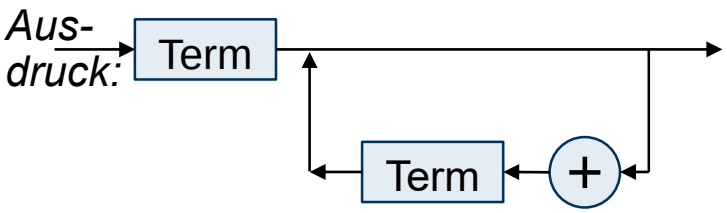
Das ist übrigens ein Baum (in Mengendiagrammdarstellung)

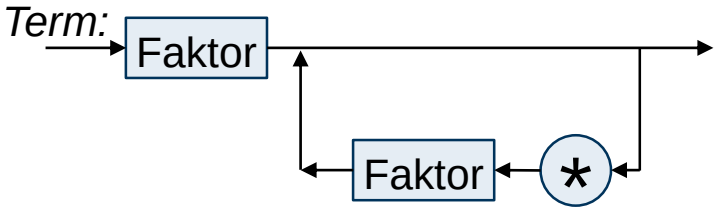
Vgl. dies mit der EBNF-Notation aus „Informatik I"!

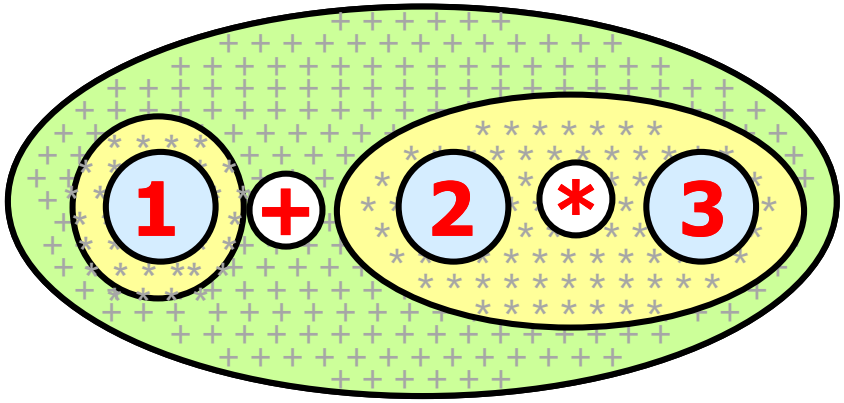
Die EBNF für Ausdrücke

```
factor     = number
           | "(" expression ")"
           | "-" factor.

term       = factor { "*" factor | "/" factor }.

expression = term { "+" term |"-" term }.
```

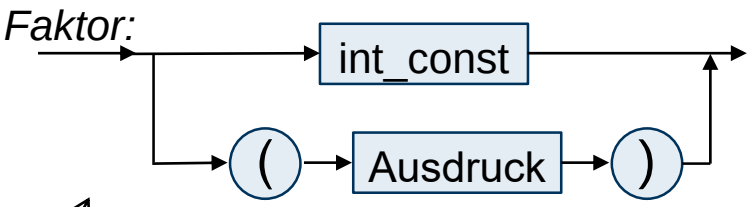
Syntax von arithmetischen Ausdrücken



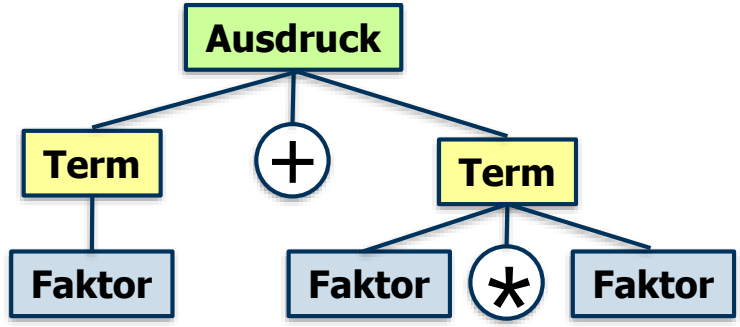




Das ist übrigens ein Baum (in Mengendiagrammdarstellung)



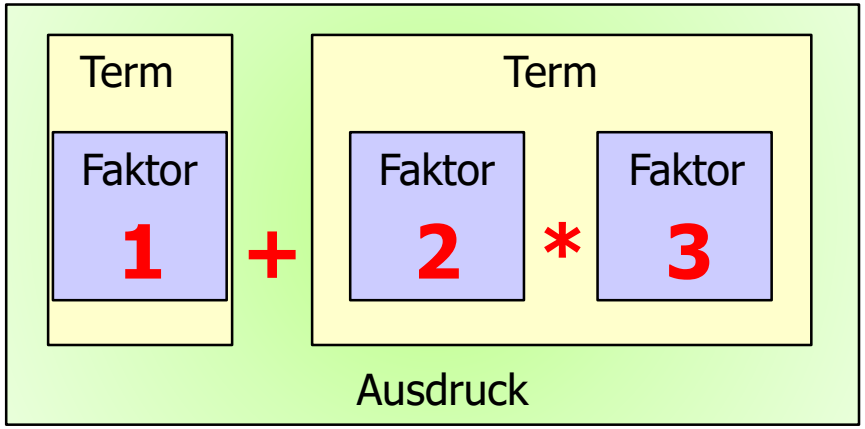
Das sind Pro-duktionsmittel

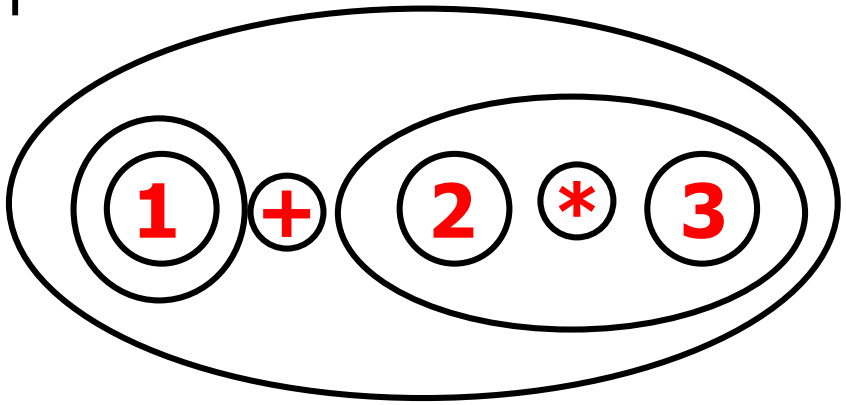


Das ist eines von vielen möglichen Produkten

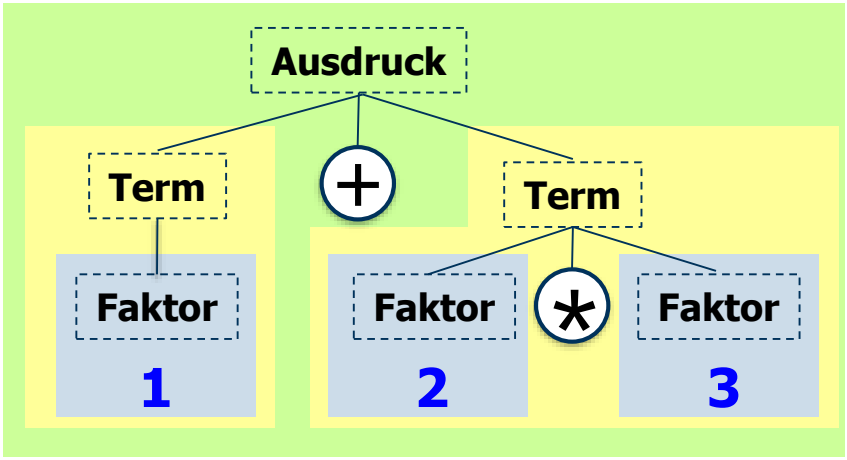
Syntax von arithmetischen Ausdrücken

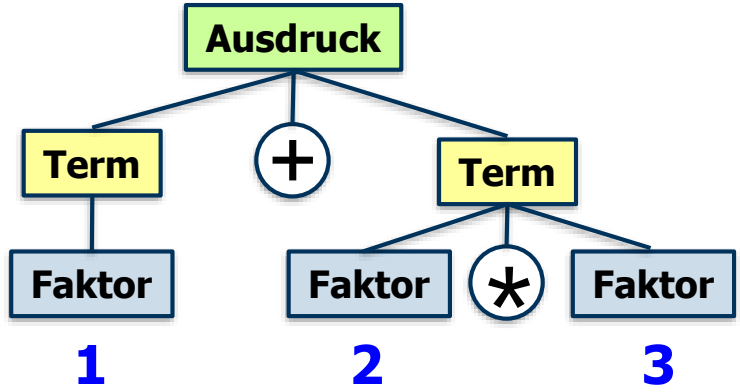
- Diverse Repräsentationen noch-mals zur Veranschaulichung:



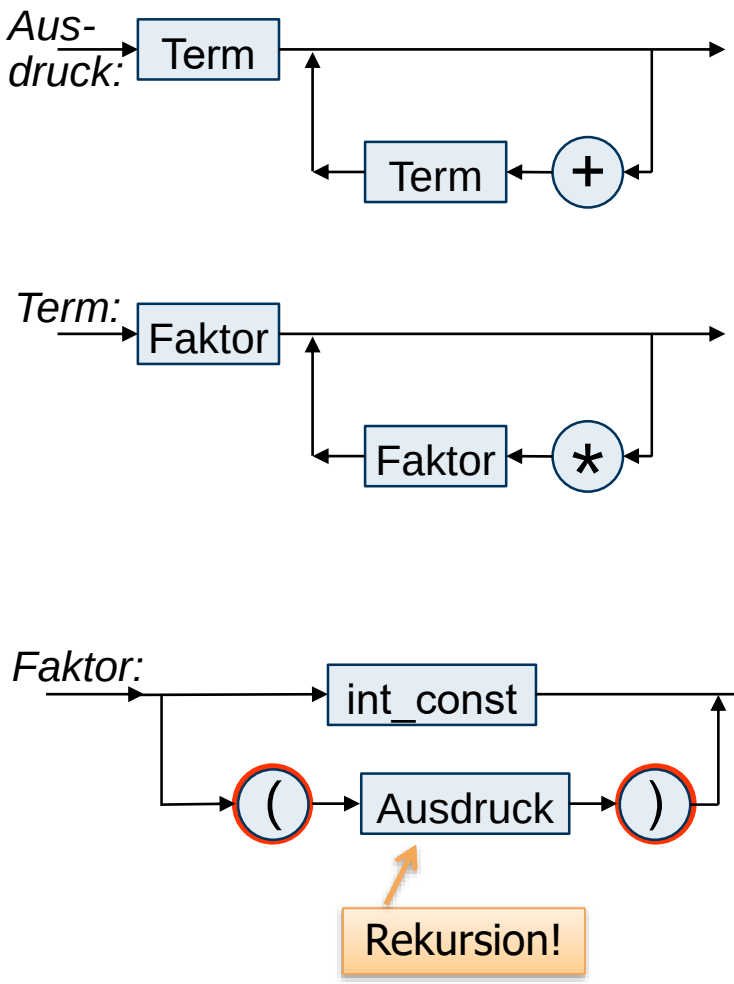


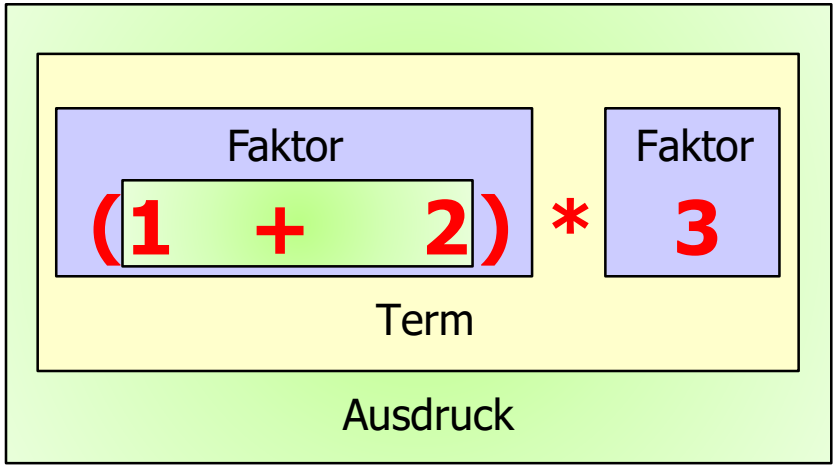
Denkübung: Diesen Baum in Klammerdarstellung angeben





Syntax von arithmetischen Ausdrücken



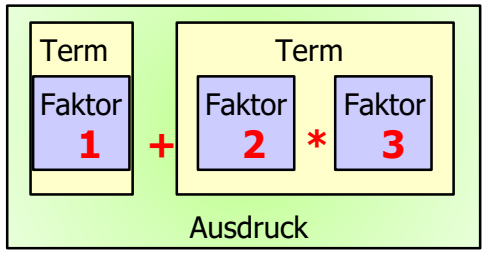


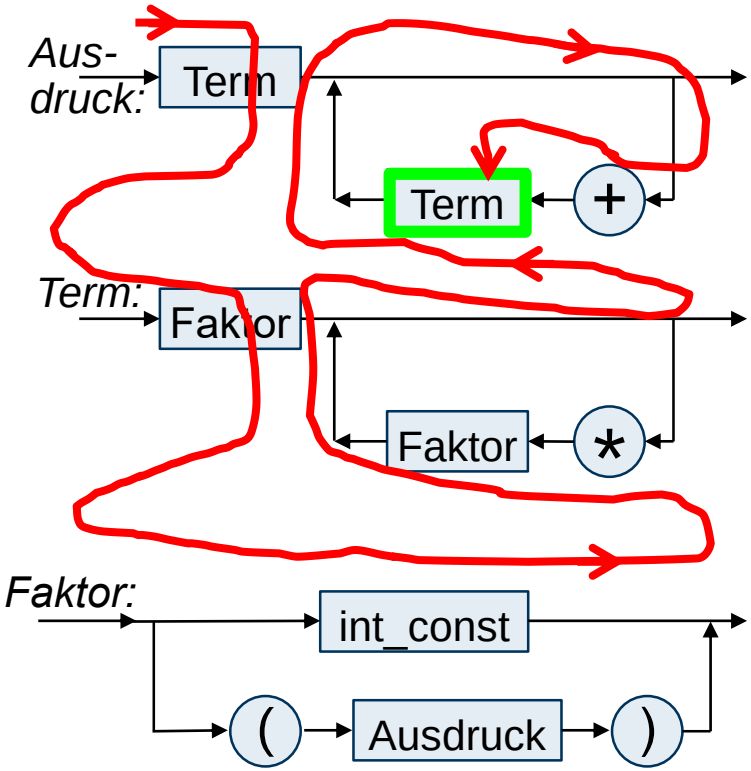
„Ausdruck“ ist (indirekt) rekursiv, so dass beliebig komplexe (d.h. zusammengesetzte) Ausdrücke gebildet werden können; die Aufspaltung in Term und Faktor ermöglicht die Darstellung der Bindungspriorität

Vgl. die beiden nachfolgenden Beispiele: wie geht daraus die Priorität, also die (evtl. implizite) „Klammerung“ der Teilausdrücke hervor?

Syntaxdiagramme und Syntaxbäume

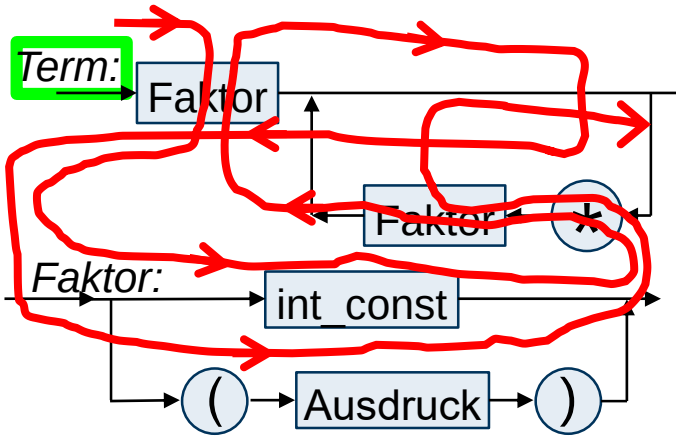
Durch eine Baumdarstellung (der Elemente eines Ausdrucks) kann der im Syntaxdiagramm durchlaufene Weg dargestellt werden:





1+2*3

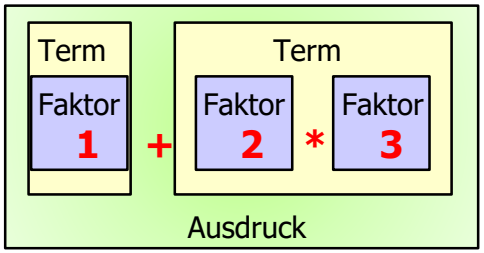
Ausdruck
Term
Faktor
int_const: 1
+
Term
Faktor
int_const: 2
*
Faktor
int_const: 3

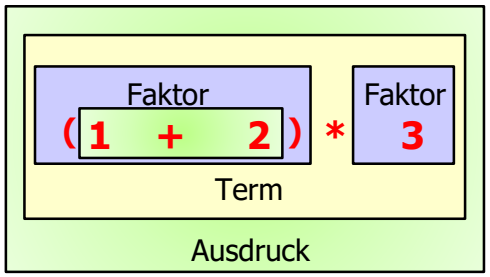


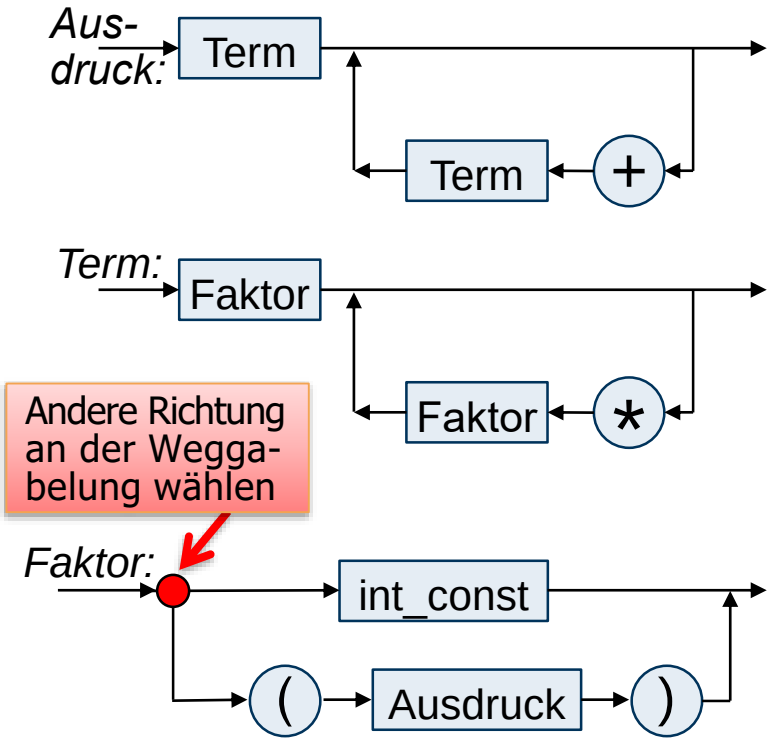
„Wegbeschreibung" für die Reise durch das Syntaxdiagramm

Syntaxbäume für 1+2*3 und (1+2)*3

Durch eine Baumdarstellung (der Elemente eines Ausdrucks) kann der im Syntaxdiagramm durchlaufene Weg dargestellt werden:







1+2*3

```
Ausdruck
  Term
    Faktor
      int_const: 1
  +
  Term
    Faktor
      int_const: 2
  *
    Faktor
      int_const: 3
```

(1+2)*3

```
Ausdruck
  Term
    Faktor
      (
      Ausdruck
        Term
          Faktor
            int_const: 1
        +
        Term
          Faktor
            int_const: 2
      )
      *
    Faktor
      int_const: 3
```

Syntaxanalyse

- Problem: Gegeben ein Textfragment, welches so aussieht, als wäre es ein Teil eines Java-Programms
  - Ist dieses wirklich syntaktisch korrekt, d.h. nach den Syntaxregeln (= Syntaxdiagramm) gebildet worden?

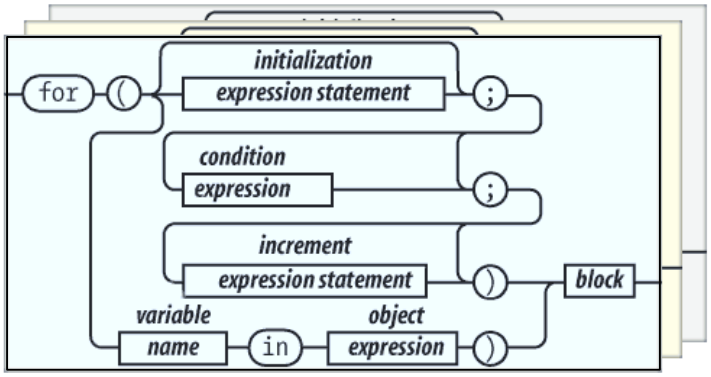
- „Lösung“: Versuche, das Syntaxdiagramm mit „Spürsinn“ so zu durchlaufen, dass das gegebene Textfragment erzeugt wird
  - Wenn dies gelingt → Textfragment syntaktisch korrekt
  - Wenn dies nicht gelingt:
    (a) → Textfragment syntaktisch nicht korrekt
    (b) → Man hat sich nicht geschickt genug angestellt

Syntaxchecker

- Problem: Einen Algorithmus angegeben, der für
  (1) beliebige Syntaxdiagramme und
  (2) einen beliebigen Text

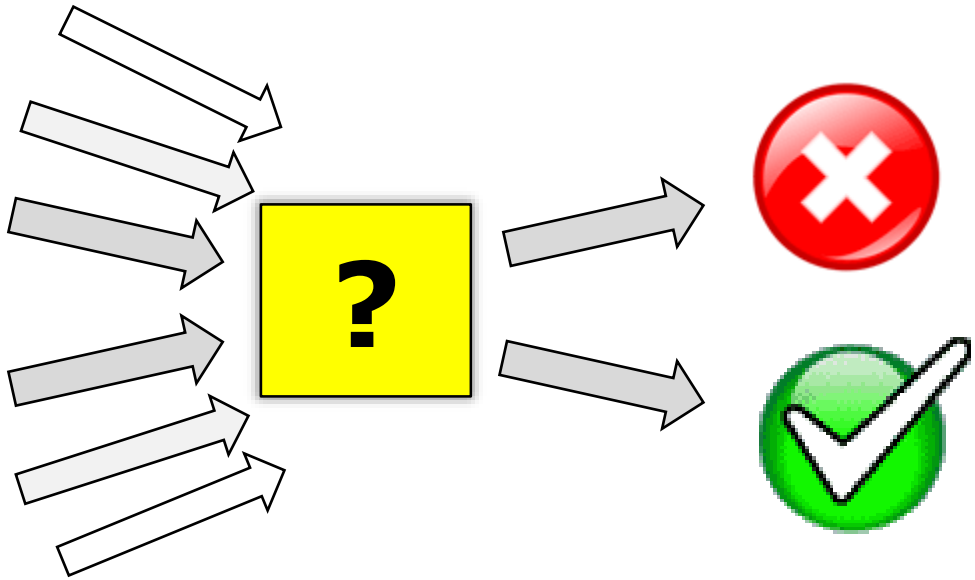
  eindeutig entscheidet, ob sich der Text mit den Diagrammen generieren lässt (→ Syntaxchecker)
  → Theorie der Syntaxanalyse → formale Sprachen, Compilerbau

(1)



(2)

```
for(int i=1;
i<=t;  i++)
```

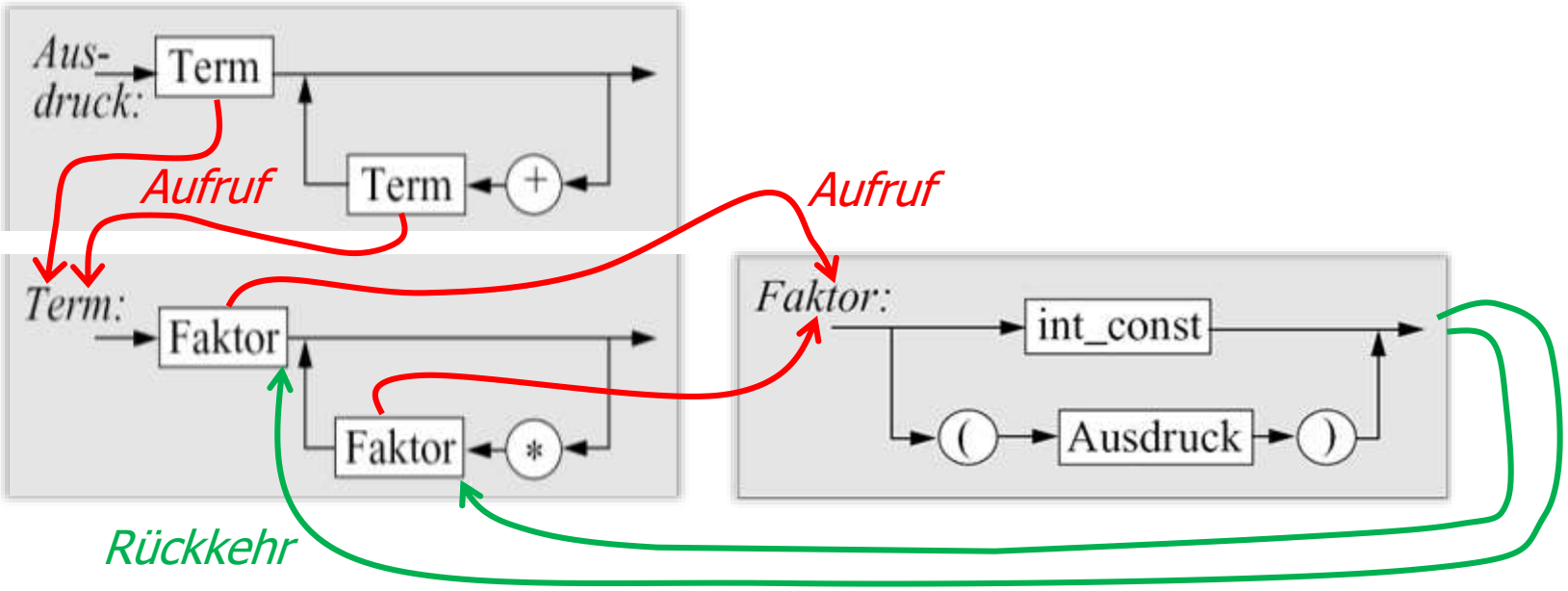


Syntaxanalyse durch „rekursiven Abstieg“

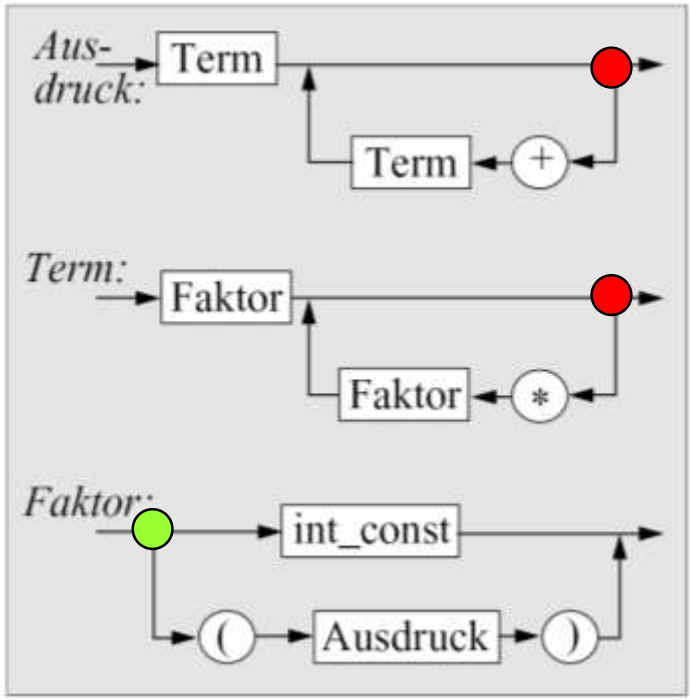
Die Idee

- „Programmierung“ von Syntaxdiagrammen: Jedes Teildiagramm durch eine Java-Methode realisieren, welche ein entsprechendes Konstrukt „versteht“

- „Absteigen“ in ein Unter-Syntaxdiagramm durch Aufruf der zugehörigen Methode (evtl. rekursiv!)



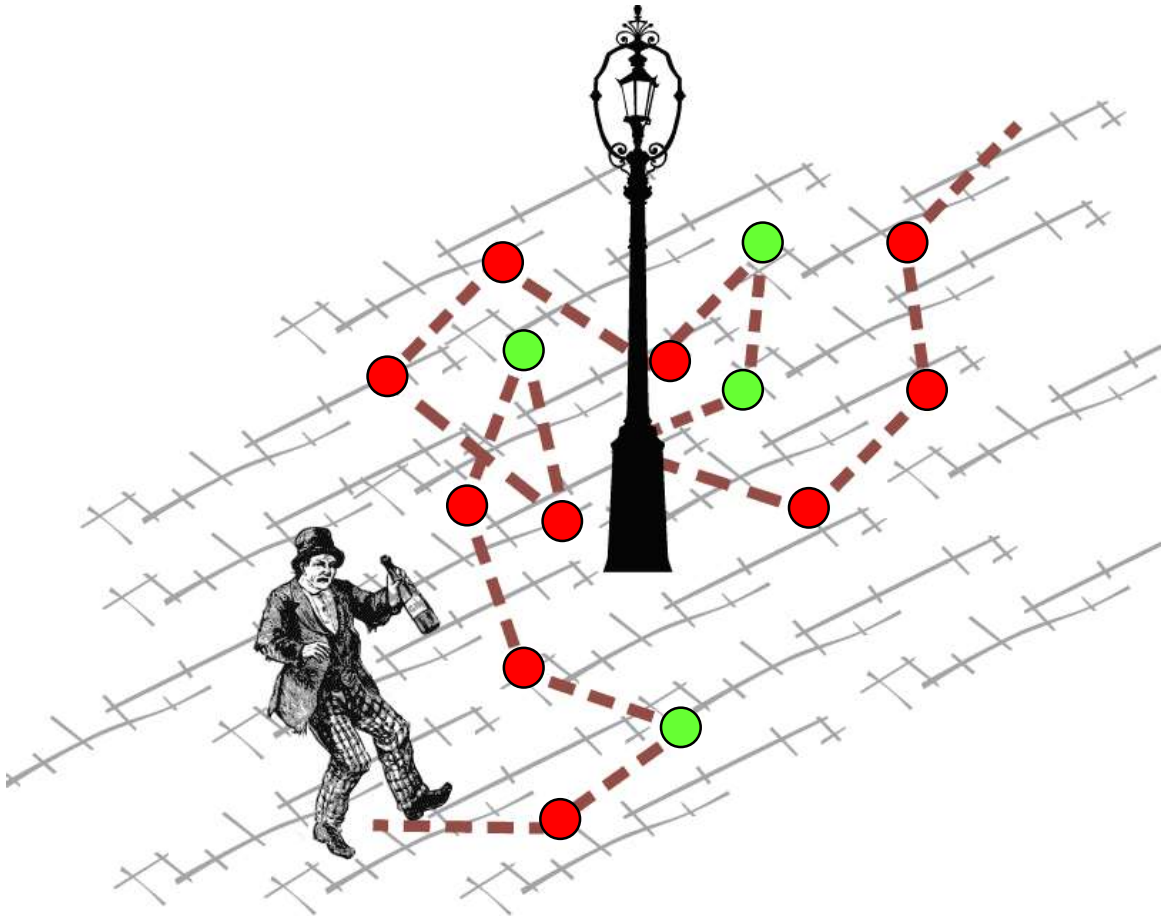
Rekursiver Abstieg
(am Beispiel arithmetischer Ausdrücke)

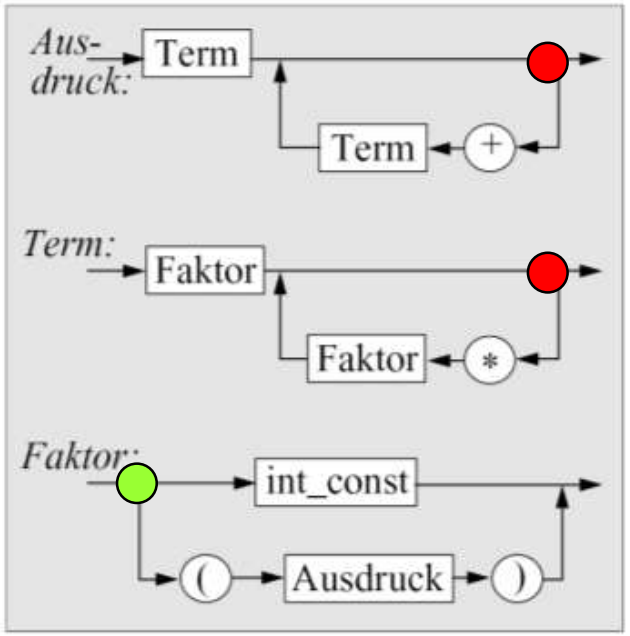


- Wiederholungen (z.B. am Ende von „Ausdruck“ und „Term“) werden durch while-Schleifen realisiert
- Verzweigungen (z.B. in „Faktor“) werden durch if-Anweisungen realisiert

Ein Generator für zufällige Ausdrücke

Wir *torkeln* durch das Syntaxdiagramm!



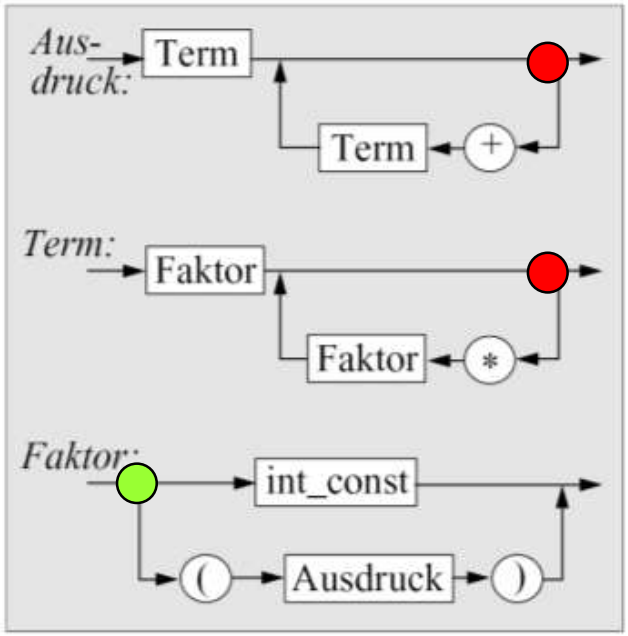


Ein Generator für zufällige Ausdrücke

Wir *torkeln* durch das Syntaxdiagramm!

```
void Ausdruck() {
  Term();
  while (randomlytrue()) {
    System.out.print("+");
    Term();
  }
}
void Term() {
  Faktor();
  while (randomlytrue()) {
    System.out.print("*");
    Faktor();
  }
}
void Faktor() {
  if (!randomlytrue()) int_const();
  else
    { System.out.print("(");
      Ausdruck(); // Rekursion
      System.out.print(")");
    }
}
void int_const(){
  System.out.print(rand.nextInt(10));
} // Zufällig zwischen 0 und 9 inkl.
```

Wie lang werden die Ausdrücke im Mittel, wenn in 40 Prozent aller Fälle „true“ geliefert wird, sonst „false“?



Mit etwas Glück und geeignetem Feintuning von randomlytrue und erhält man Ergebnisse wie (((3)*(8)*3*(4)*(((2*2)*0+8+1)*2+(1))+5*(0+2+0)+3*(0*7+(8*3))*2+0)+2+0)*7; sonst vielleicht nur eine einzige müde Ziffer oder umgekehrt einen StackOverflowError...

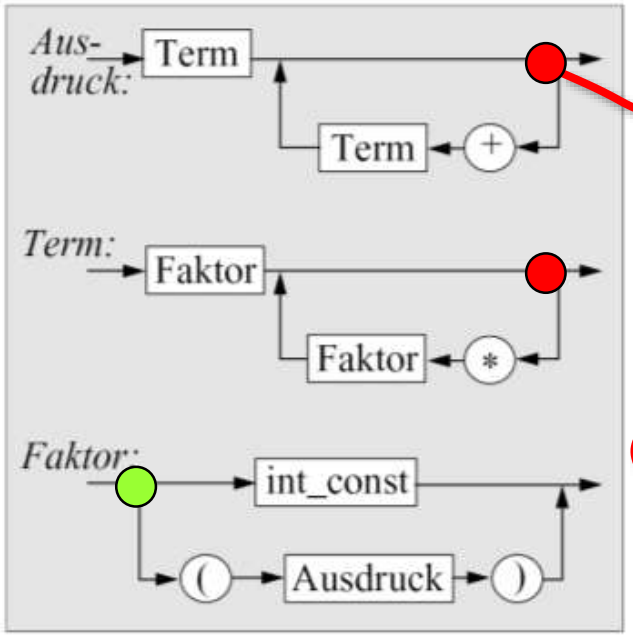
Dazu java.util.Random importieren und in main: Random rand = new Random();

Ein Parser als Java-Programm

„Zerteiler“ (lat. „pars“ = Teil), bzw. „Syntaxanalyseprogramm“

Wir greifen hier das Beispiel eines Parsers für Taschenrechnerfunktionalität aus „Informatik I“ wieder auf

Jetzt werden Zeichen nicht generiert, sondern konsumiert!




```
//Anfangsteil kommt später
  ...
  void int_const(){
    c = KbdInput.getc();
  }

  void Ausdruck(){
    Term();
    while (c == '+') {
      c = KbdInput.getc();
      Term();
    }
  }
  ...
```

Integer-Konstanten sollen hier nur aus einem einzigen Zeichen bestehen

Vorher verarbeitetes Zeichen „weglesen“; nachfolgendes Zeichen auf die Variable c einlesen

Bemerkung: KbdInput ist eine Klasse mit Methoden (wie z.B. „getc“), die nicht zum Standard-Java gehört. Die Funktionalität (Zeichen vom Keyboard lesen) davon lässt sich aber leicht mit Methoden aus dem Paket java.io.* realisieren (Übung!)

Ein Parser als Java-Programm (2)

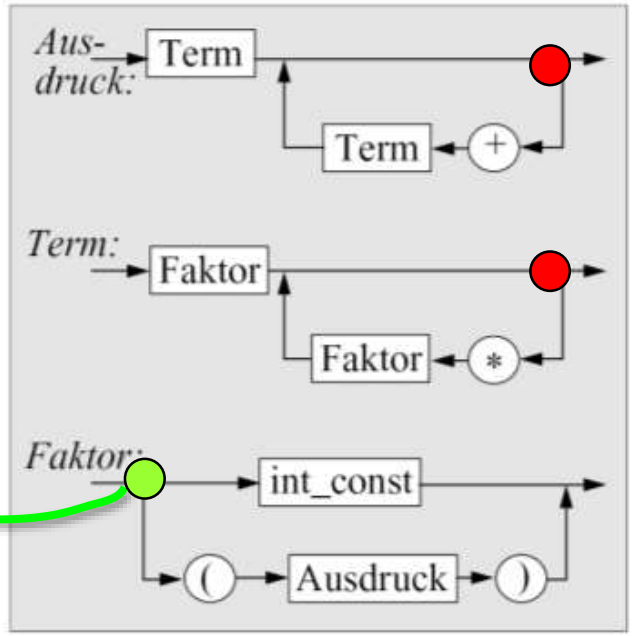
```
//Anfangsteil kommt später
  ...
  void int_const(){...}
  void Ausdruck(){...}
```

```
  void Term() {
    Faktor();
    while (c == '*') {
      c = KbdInput.getc();
      Faktor();
    }
  }
  void Faktor() {
    if ((c >='0') && (c <='9'))
      int_const();
    else
      if (c == '(') {
        c = KbdInput.getc();
        Ausdruck();
        if (c == ')')
            c = KbdInput.getc();
        else Fehler(1);
        }
      else Fehler(2);
  }
```

"Term an Faktor: *Lies einen Faktor!*"

"Faktor an int_const: *Lies eine int_const!*"

Rekursion!



Abkürzung für:
c =='0' || c =='1' || c =='2' || … || c =='9'
(da die Zifferzeichen 0 bis 9 im Zeichensatz hintereinander stehen)

```
public static void main(String args[])
{
  Parser p = new Parser();
  p.c = KdbInput.getc();
  p.Ausdruck();
}
```

"Hauptprogramm an Parser p: *Lies einen Ausdruck!*"

Ein Parser als Java-Programm – Anfangsteil

```
import packagexyz.KbdInput;

class Parser() {
  char c;

  void Fehler(int f) {
    switch f {
      case 1:
        System.out.println("Fehlende Klammer ')'");
        break;
      case 2:
        System.out.println("Falsches Zeichen;"+
          " erwartet: Ziffer oder '('");
        break;
      default:
        System.out.println("Interner Fehler");
        break;
   }
   //...
   //Die anderen Methoden
}
```

Dort sei die Methode getc definiert, die ein Zeichen als "char" einliest

Global für alle Methoden; auf c steht immer das nächste zu betrachtende Eingabezeichen („lookahead")

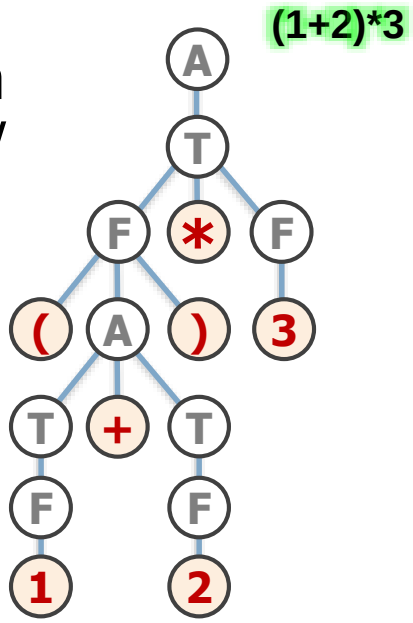
Kapselung aller Fehlermeldungen in einer einzigen Methode → leichtere Anpassung

Defensives Programmieren: mit falschem Parameter f rechnen!

Wenn nach Abarbeitung der gesamten Eingabe kein Fehler gemeldet wurde, dann lag ein (gemäss den Syntaxdiagrammen) korrekter Ausdruck vor
⇒ Programm ist ein „Syntaxchecker"

Erzeugen eines Syntaxbaums

- Man benötigt i.Allg. nicht nur einen Syntaxchecker mit dem Resultat „syntaktisch korrekt / falsch“, sondern zur weiteren Bearbeitung auch den zum analysierten Programm gehörenden Syntaxbaum
- Dazu erweitern wir das Analyseprogramm zur Ausgabe des Baums in „eingerückter Darstellung“

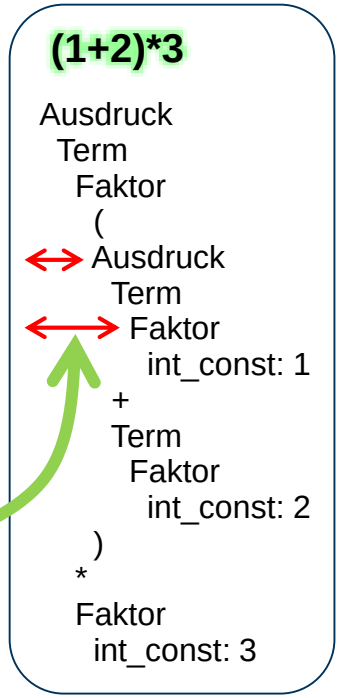



```
Ausdruck
  Term
    Faktor
      (
      Ausdruck
        Term
          Faktor
            int_const: 1
        +
        Term
          Faktor
            int_const: 2
      )
    *
    Faktor
      int_const: 3
```

Erzeugen eines Syntaxbaums (2)

- Idee: Jede Methode, die für ein Terminal oder Nicht-Terminal steht, gibt ihren eigenen Namen aus
- Und zwar eingerückt um eine Länge, die proportional zur Tiefe (= Abstand zur Wurzel) des Knotens ist
- Jede Methode bekommt dazu die momentane Tiefe als int-Parameter übergeben
- Wir nutzen eine Hilfsprozedur „out“:

```
void out(int t, String s) {
for(int i=1;i<=t;i++)
    System.out.print(" ");
  System.out.println(s);
}
```



Erzeugen eines Syntaxbaums (3)

```
void Ausdruck(int t) {
  out(t,"Ausdruck");
  Term(t+1);
  while (c == '+') {
    out(t+1,"+");
    c = KbdInput.getc();
    Term(t+1);
  }
}

void int_const(int t) {
  out(t,"int_const: " + c);
  c = KbdInput.getc();
}

// Entsprechend werden "Term"
// und "Faktor" ergänzt
```

"out" gibt den Namen entsprechend eingerückt aus

"Term" und "+" haben als Kinder von „Ausdruck" eine um 1 grössere Tiefe

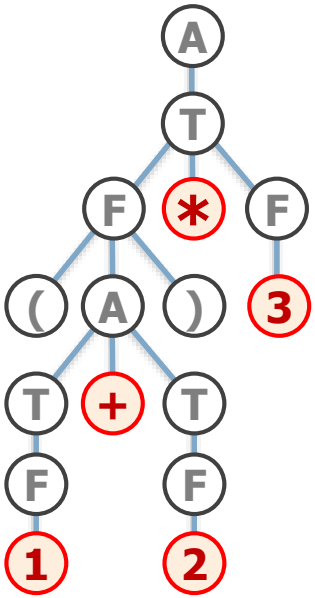
Wieso muss man t zwar erhöhen, aber nie erniedrigen?

Um den Wert der Konstanten zu erfahren

(1+2)*3

Ausdruck
- Term
  - Faktor
    - (
    - Ausdruck
      - Term
        - Faktor
          - int_const: 1
        - +
        - Term
          - Faktor
            - int_const: 2
    -)
  - *
  - Faktor
    - int_const: 3

Dies ist das Resultat



...aufgehübscht

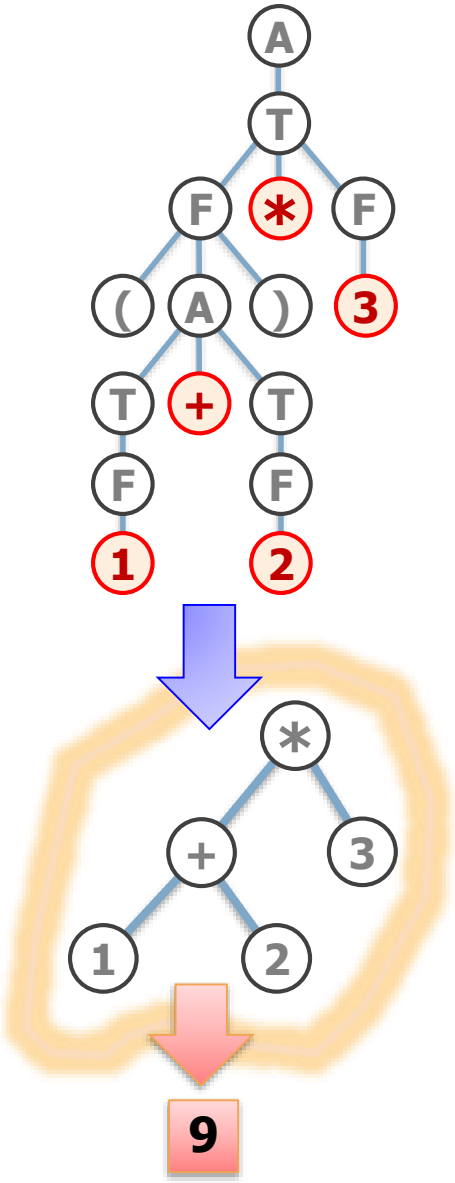
Binäre Operatorbäume

Wir kennen solche Bäume unter dem Begriff „Ausdrucksbaum“ schon aus „Informatik I“

- Operatorbäume entstehen durch Kompaktifizierung aus den eigentlichen Syntaxbäumen
  - Drücken aber noch gleichermassen die wesentliche Struktur eines Ausdrucks aus
- Operator zur Wurzel des Teilbaums hochziehen
  - Stellt damit ein sogenanntes Attribut des Knotens dar
- Unwesentliches entfernen
  - Namen von Nicht-Terminalen und Klammer-Knoten

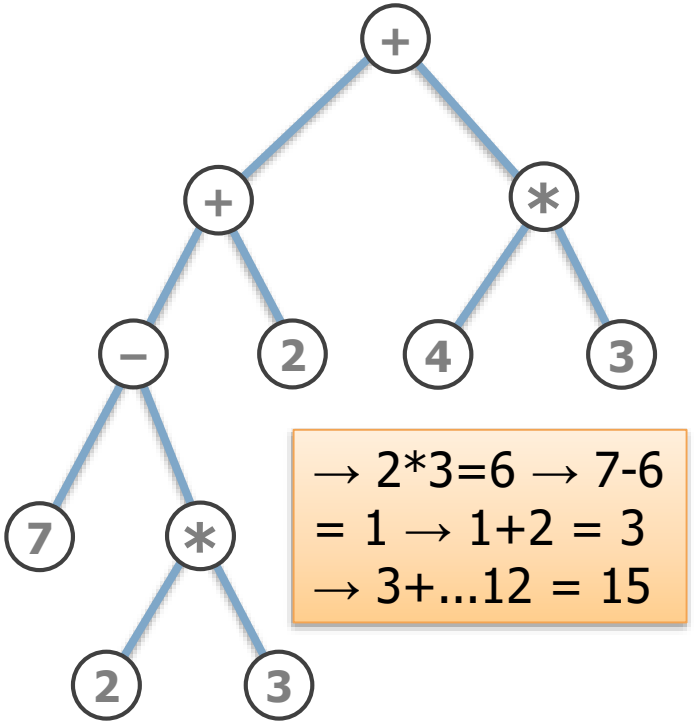
Klammer-Knoten sind unwesentlich ???

- Sind Binärbäume
  - Ein Knoten (≠ Blatt) hat nur zwei Nachfolger
  - Dazu Teilausdrücke evtl. klammern: *2*3*4 → (2*3)*4*
- Haben einen Wert
  - Diesen Wert holen wir jetzt heraus… („Auswertung“)

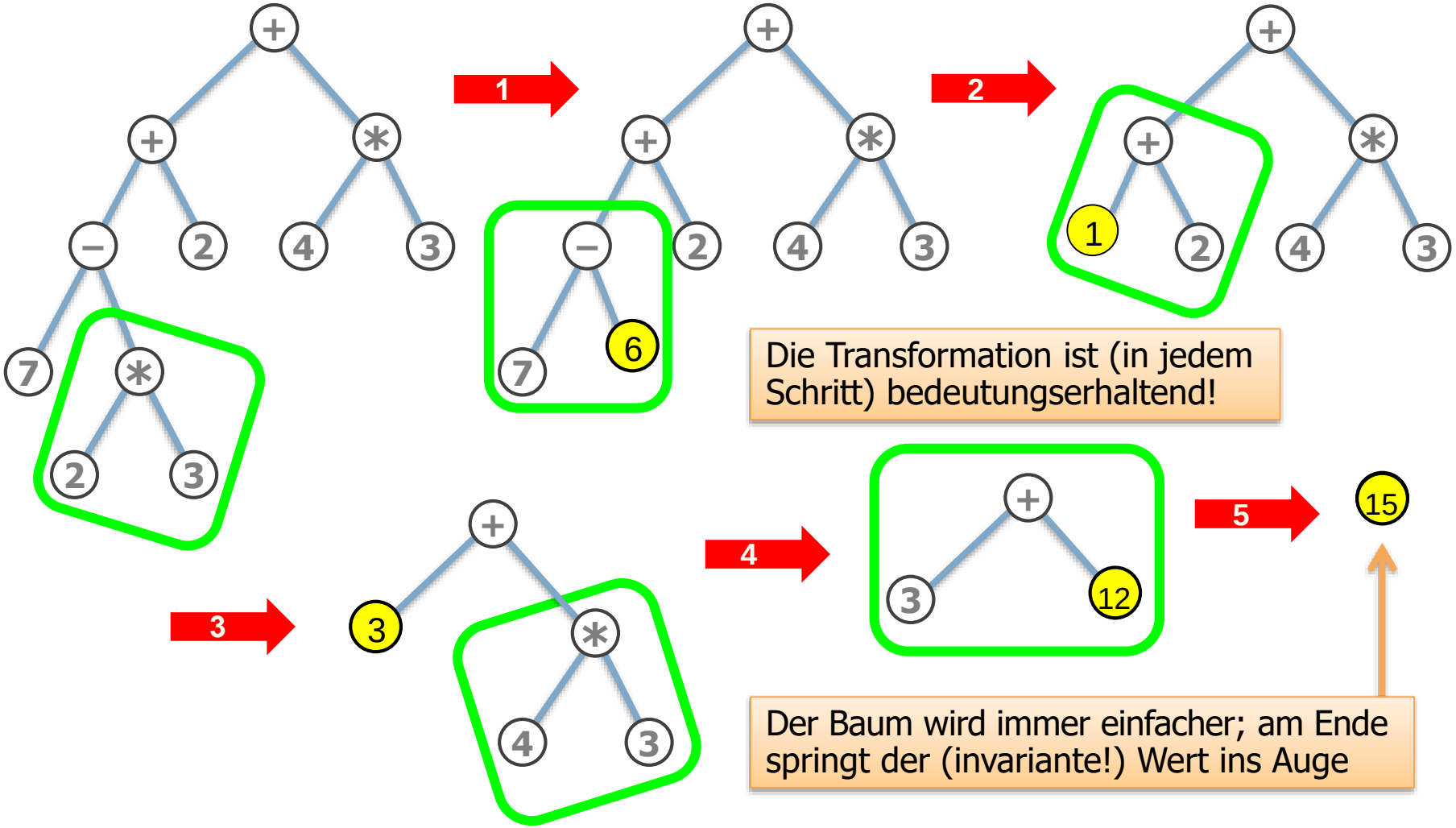


Auswertung binärer Operatorbäume

- Wenn der Baum keine Unterbäume hat, dann ist der Wert des Baums das Attribut der Wurzel (= Blatt)
- Ansonsten existiert ein linker und rechter Unterbaum – berechne (in rekursiver Weise!) „so“ den Wert des linken Unterbaums, dann „genauso“ den Wert des rechten Unterbaums und wende dann die Operation (= Attribut der Wurzel) auf diese beiden Werte an



Auswertung des Ausdrucks durch schrittweise Reduktion des Operatorbaums

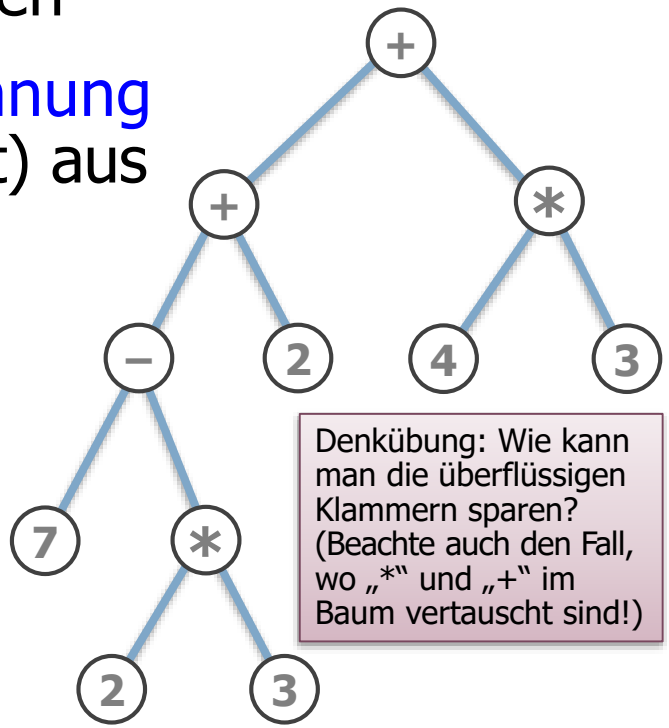


Symmetrisches Traversieren eines Binärbaums („inorder“)

Alle Knoten eines Baums „der Reihe nach“ besuchen

- Prinzip: Zuerst den linken Unterbaum traversieren, dann das Attribut der Wurzel ausgeben, schliesslich den rechten Unterbaum traversieren
- Beispielhafte Anwendung: Rückgewinnung des Ausdrucks (vollständig geklammert) aus einem Operatorbaum:
  - Falls die Wurzel Nachfolger hat:
    - Ausgabe von „(“ und wende Algorithmus auf linken Unterbaum an
  - Ausgabe des Attributs der Wurzel
  - Falls die Wurzel Nachfolger hat:
    - Wende Algorithmus auf rechten Unterbaum an und Ausgabe von „)“

Das ist dann schon fast ein „Prettyprinter“!

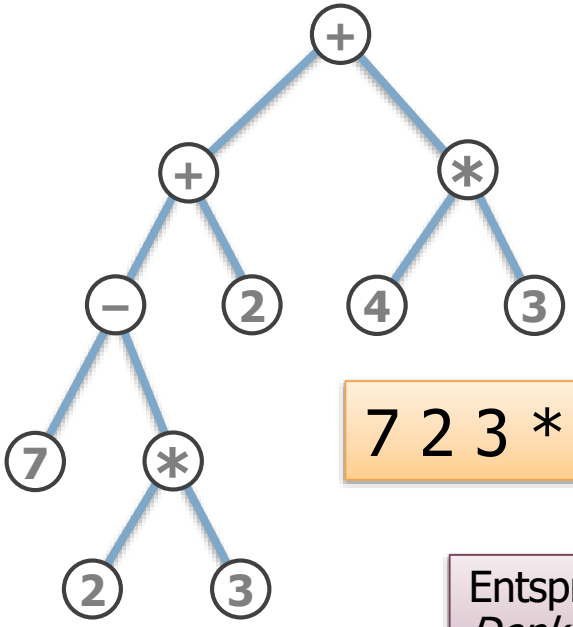


Denkübung: Wie kann man die überflüssigen Klammern sparen? (Beachte auch den Fall, wo „*“ und „+“ im Baum vertauscht sind!)

→ (((7-(2*3))+2)+(4*3))

Baumtraversierung in „postorder“

- Zuerst linken Unterbaum traversieren (falls vorhanden)
- Dann rechten Unterbaum traversieren (falls vorhanden)
- Erst danach („post“!) die Wurzel „betrachten“
  - Z.B. das bei einem Operatorbaum vorhandene Attribut ausgeben



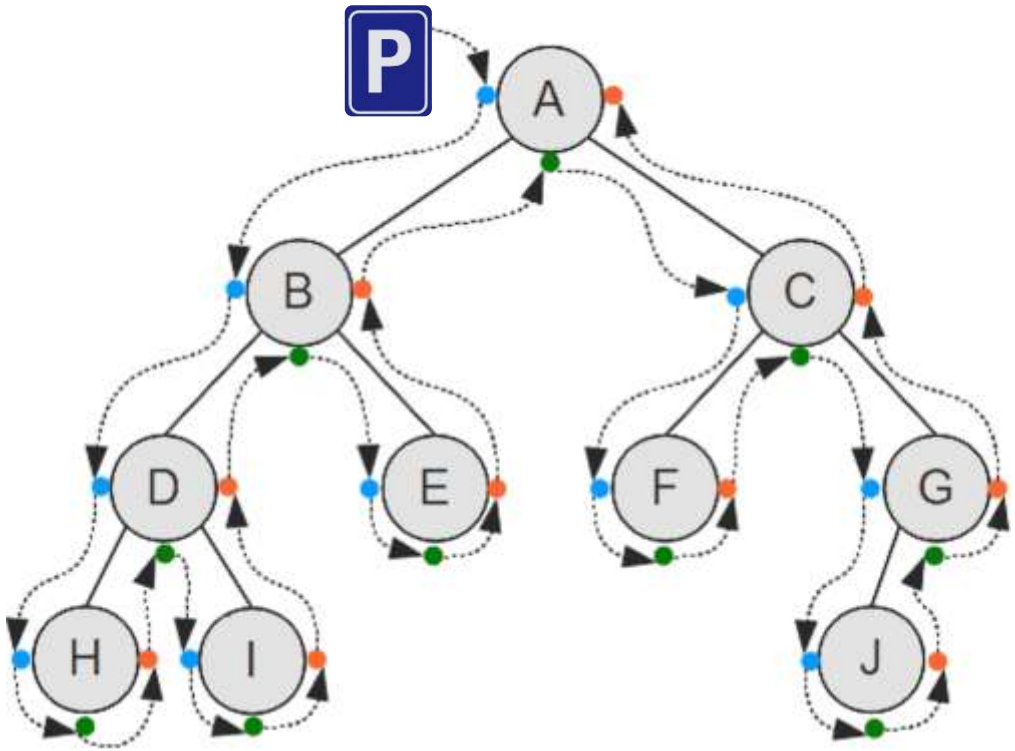
Lässt sich daraus der Operatorbaum wieder eindeutig rekonstruieren?

Die Postfix-Notation des Infix-Ausdrucks ((7-(2*3))+2)+(4*3)

7 2 3 * - 2 + 4 3 * +

Entsprechend lassen sich die „preorder“ und Präfix-Notation definieren.
Denkübung: Hängen Klammerdarstellung und preorder zusammen?

Baumtraversierung als Eulertour



- Wir stellen uns vor, dass die Kanten Teil einer Stadtmauer sind, ebenso wie die Knoten (Wehrtürme). Man beginnt die Tour beim Parkplatz in der Nähe von A. Mit der linken Hand ständig an der Mauer, läuft man nun aussen um die Stadtmauer, bis man wieder zum Parkplatz zurückkommt.
- Bei jedem Knoten des Binärbaums kommt man zunächst im Westen (blauer Punkt), sodann im Süden (grüner Punkt) und schliesslich im Osten (roter Punkt) vorbei.

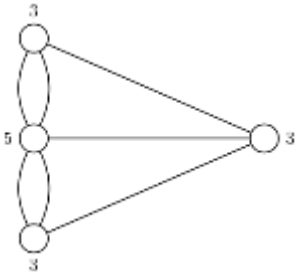
- Man erhält die Inorder-Reihenfolge HDIBEAFCJG, wenn man Knoten jeweils im Süden betrachtet; die Postorder-Reihenfolge HIDEBFJGCA, wenn man Knoten jeweils im Westen betrachtet; und die Preorder-Reihenfolge bei Betrachtung im Osten.
- Den vollständig geklammerten Infix-Ausdruck erhält man aus einem Operatorbaum mittels einer Eulertour so: Westen → „(“ ; Süden → Knoteninhalt ; Osten → „)“.

Euler und seine Tour

Schweizer Banknote (1976 – 2000) mit Leonh. Euler

Leonhard Euler (1707 – 1783) war ein bedeutender Schweizer Mathematiker und Physiker. Aufgewachsen in Basel, lebte und wirkte er anschliessend in St. Petersburg und in Berlin. Neben Analysis, Algebra und Zahlentheorie befasste er sich auch mit Anwendungen der Mathematik in anderen Disziplinen; 1736 veröffentlichte er seinen Aufsatz „Solutio problematis ad geometriam situs pertinentis“ (Lösung eines Problems zur Geometrie der Lage) mit dem bekannten Königsberger Brückenproblem, welcher als Keimzelle der Graphentheorie und Topologie gilt.

Als Eulertour (oder Eulerkreis) wird heute ein Zyklus in einem Graphen bezeichnet, welcher (entsprechend dem Königsberger Brückenproblem) alle Kanten genau ein Mal enthält. Fasst man in einem Baum jede Kante als ein Paar gegenläufig gerichteter Kanten eines Graphen auf, dann folgt aus dem von Euler gefundenen (allerdings erst später von Carl Hierholzer als hinreichend bewiesenen) Kriterium „der Grad jedes Knotens ist gerade“ (bzw. spezifischer: Eingangs- und Ausgangsgrad sind identisch), dass eine Eulertour existiert. Diese kann man an der Wurzel beginnen und enden lassen und als Tiefensuche des Baumes realisieren.



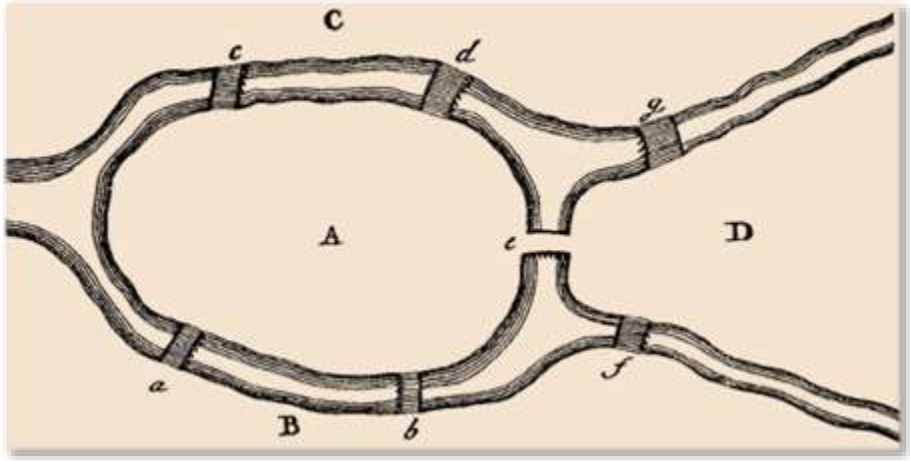


Fig. 1 aus Eulers Aufsatz (Auszug auf folgenden Slides)

Auszug aus dem Buch *Leonhard Euler. Die Geburt der Graphentheorie* von Wladimir Velminski:

Am 15. April 1707 wurde Euler als Sohn des Pfarrers Paulus Euler und dessen Frau Margaretha Bruckerin in Basel geboren. Kurz nach der Geburt des Jungen zog die Familie von Basel in das benachbarte Dorf Riehen, dessen Gemeinde der Vater fortan betreuen sollte. So wuchs der kleine Leonhard in geordneten Verhältnissen auf, wurde vom Vater im Sinne der Religion erzogen und früh durch erste Rechenaufgaben gefördert: „so trachtete er mir sogleich die erste Gründe der Mathematic beizubringen, und bediente sich zu diesem End des Christophs Rudolphs Coss mit Michael Stiefels Anmerkungen, wo rinnen ich mich einige Jahr mit allem Fleiss übte“ – erinnert sich Euler im Alter von 60 Jahren. Das Mathematikbuch von 1525, das Euler in seiner Jugend prägte, bewegt sich zwischen bloßer Rechenkunst und vollzogener Algebraisierung und macht unendliche Zahlen und ausufernde Rechnungen bereits durch erste mathematische Symbole und Kunstwörter nachvollziehbar. Und wenn heute anstatt des Wortes Summe das Zeichen Σ und anstelle der Kreiskonstanten 3.14159… π steht und die Darstellung einer Funktion mit f(x) gekennzeichnet wird, dann sind es Eulers Symbole, die uns die Mathematik anschaulich machen. Am deutlichsten sind die Spuren der mathematischen Poesie Eulers im Symbol e – später die Eulersche Zahl – auszumachen, das er als Basis der natürlichen Logarithmen 2.7182818284… eingeführt hat.

Während seines Studiums an der Universität Basel fiel Euler durch klare mathematische Lösungsstrategien auf. Der Mathematikprofessor Johann Bernoulli, Patron der bekannten Basler Mathematikerfamilie, förderte ihn und gewährte ihm Zugang zu seiner Privatbibliothek. Durch das Stöbern in den Büchern und mathematischen Korrespondenzen des Lehrers, die dieser u.a. mit Leibniz führte, blühte Euler, der Autodidakt, auf. Wenig später wurde er als Adjunkt an die St. Petersburger Akademie der Wissenschaften berufen – zu einem Gehalt von jährlich 300 Rubeln „nebst freyer Wohnung, Holtz und Licht“. Nach St. Petersburg hatten ihn die Söhne seines Mentors empfohlen – Daniel und Nikolaus Bernoulli. Mit Basel ließ der Zwanzigjährige sein vergebliches Bemühen um die dortige Physikprofessur hinter sich. Ein Schiff brachte ihn nach Mainz, eine Postkutsche von dort nach Marburg, wo er auf Empfehlung seines Lehrers Bernoulli den Allgemeingelehrten Christian Wolff besuchte. Dieser versprach Euler eine Reise ins „Paradies der Gelehrten“ – die neue Hauptstadt des Russischen Reiches.

Solutio problematis ad geometriam situs pertinentis

Eulers Aufsatz von 1736 ist lesenswert; nachfolgend drei Abschnitte (Übertragung ins Deutsche inspiriert von der Übersetzung von W. Velminski):

1. Praeter illam geometriae partem, quae circa quantitates versatur et omni tempore summo studio est exculta, alterius partis etiamnum admodum ignotae primus mentionem fecit Leibnitzius, quam *Geometriam situs* vocavit. Ista pars ab ipso in solo situ determinando situsque proprietatibus eruendis occupata esse statuitur; in quo negotio neque ad quantitates respiciendum neque calculo quantitatum utendum sit. Cuiusmodi autem problemata ad hanc situs geometriam pertineant et quali methodo in iis resolvendis uti oporteat, non satis est definitum. Quamobrem, cum nuper problematis cuiusdam mentio esset facta, quod quidem ad geometriam pertinere videbatur, at ita erat comparatum, ut neque determinationem quantitatum requireret neque solutionem calculi quantitatum ope admitteret, id ad geometriam situs referre haud dubitavi, praesertim quod in eius solutione solus situs in considerationem veniat, calculus vero nullius prorsus sit usus. Methodum ergo meam, quam ad huius generis problemata solvenda inveni, tanquam specimen Geometriae situs hic exponere constitui.

Lösung eines Problems zur Geometrie der Lage

1. Neben jenem Bereich der Geometrie, der die Grössen untersucht und zu allen Zeiten eifrig studiert wurde, gibt es noch einen anderen bis jetzt beinahe unbekannten, den Leibniz als Erster erwähnt und Geometrie der Lage (*Geometriam situs*) genannt hat. Gegenstand der Untersuchung ist das, was nur durch die Lage bestimmt werden kann und die Ergründung der Eigenschaften dieser Lage; hierbei sollen die Grössen ausser Acht gelassen und das Rechnen mit Grössen nicht angewendet werden. Welche Probleme aber zu dieser Geometrie der Lage gehören und welche Methode zu ihrer Lösung angewendet werden muss, das ist noch nicht genügend bestimmt. Als neulich ein Problem bekannt wurde, das zwar zur Geometrie zu gehören schien, aber so beschaffen war, dass es weder die Bestimmung einer Grösse erforderte, noch eine Lösung mit Hilfe des Grössenkalküls gestattete, zweifelte ich nicht daran, es der Geometrie der Lage zuzuordnen, umso mehr als zu seiner Lösung nur die Lage in Frage kommt, während eine Berechnung nicht von Nutzen ist. Somit werde ich meine Methode, die ich zur Lösung derartiger Probleme erfunden habe, hier als Muster der Geometrie der Lage vorstellen.

2. Problema autem hoc, quod mihi satis notum esse perhibebatur, erat sequens: Regiomonti in Borussia esse insulam A, der Kneiphof dictam, fluviumque eam cingentem in duos dividi ramos, quemadmodum ex figura (Fig. 1) videre licet; ramos vero huius fluvii septem instructos esse pontibus *a*, *b*, *c*, *d*, *e, f* et *g.* Circa hos pontes iam ista proponebatur quaestio, num quis cursum ita instituere queat, ut per singulos pontes semel et non plus quam semel transeat. Hocque fieri posse, mihi dictum est, alios negare alios dubitare; neminem vero affirmare. Ego ex hoc mihi sequens maxime generale formavi problema: quaecunque sit fluvii figura et distributio in ramos atque quicunque fuerit numerus pontium, invenire, utrum per singulos pontes semel tantum transiri queat an vero secus. [...]

2. Das Problem, das anscheinend recht bekannt ist, war folgendes (Fig. 1): Im preussischen Königsberg gibt es eine Insel A, genannt der Kneiphof. Der Fluss, welcher sie umfliesst, teilt sich in zwei Arme, wie dies im ersten Bild zu erkennen ist. Über die Arme dieses Flusses führen sieben Brücken *a*, *b*, *c*, *d*, *e*, *f* und *g*. Nun stellte sich die Frage, ob jemand seinen Spazierweg so einrichten könne, dass er jede der Brücken einmal und nicht mehr als einmal überschreite. Man sagte mir, dass Einige diese Möglichkeit verneinen, Andere unschlüssig sind, dass indes niemand einen Beweis erbracht habe. Hiervon abgeleitet, stellte ich mir folgende maximal verallgemeinerte Aufgabe: Herausfinden, ob man jedwede Brücke nur ein einziges Mal zu überschreiten braucht, unabhängig von Verlauf und Aufspaltung des Flusses sowie der Zahl der Brücken. [...]

20. Casu ergo quocunque proposito statim facillime poterit cognosci, utrum transitus per omnes pontes semel institui queat an non, ope huius regulae:

Si fuerint plures duabus regiones, ad quas ducentium pontium numerus est impar, tum certo affirmari potest talem transitum non dari.

Si autem ad duas tantum regiones ducentium pontium numems est impar, tunc transitus fieri poterit, si modo cursus in altera harum regionum incipiatur.

Si denique nulla omnino fuerit regio, ad quam pontes numero impares conducant, tum transitus desiderato modo institui poterit, in quacunque regione ambulandi initium ponatur.

Hac igitur data regula problemati proposito plenissime satisfit.

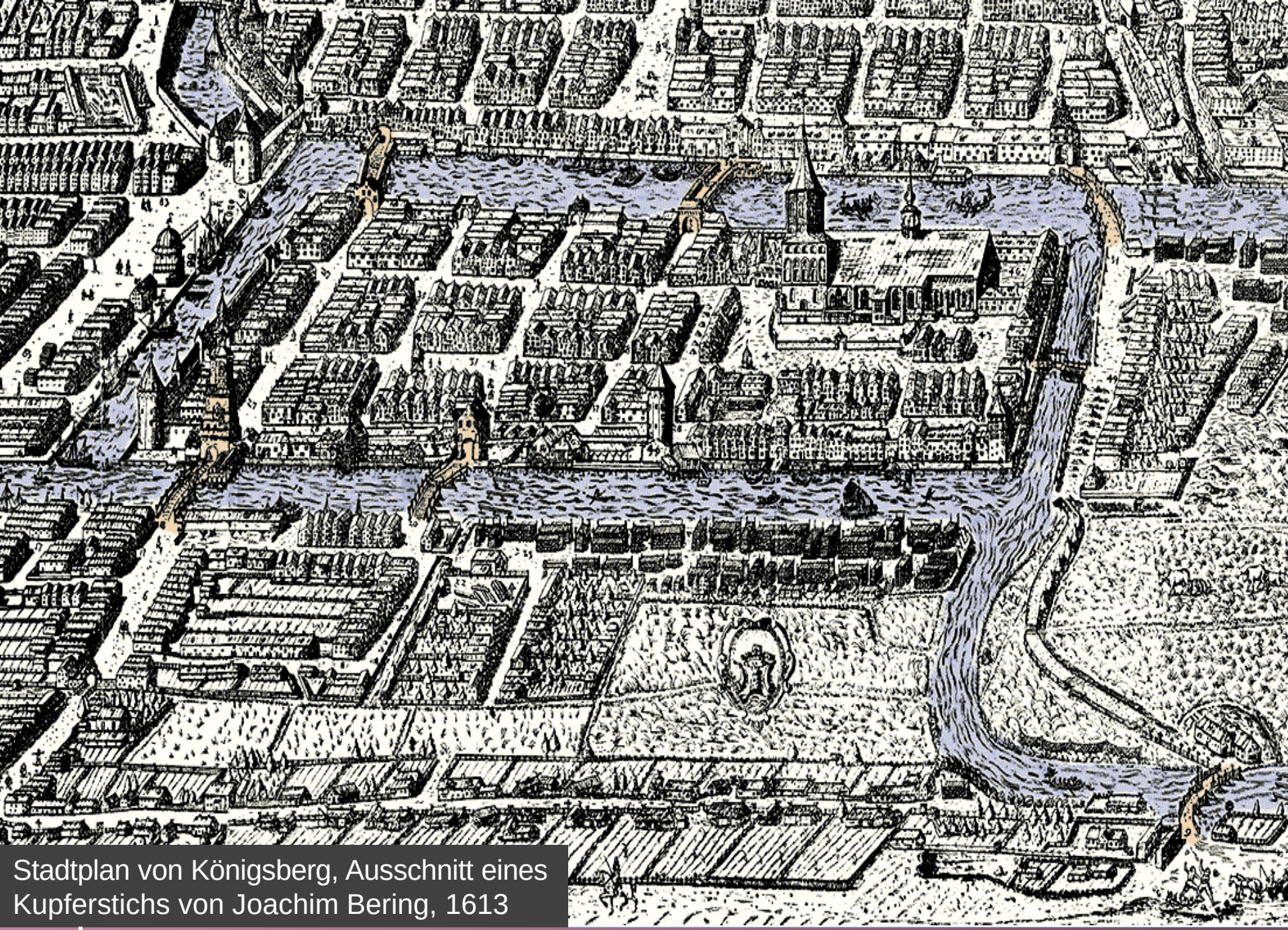
20. In einem beliebigen Fall kann man demnach mittels folgender Regeln auf das Leichteste entscheiden, ob ein einmaliger Übergang über alle Brücken möglich ist:

Wenn es mehr als zwei Gebiete gäbe, für die die Zahl der Zugangsbrücken ungerade ist, so existiert kein Weg der verlangten Art.

Wenn die Anzahl der Zugangsbrücken nur für zwei Gebiete ungerade ist, so gibt es einen Weg, vorausgesetzt, dass man in einem dieser beiden Gebiete beginnt.

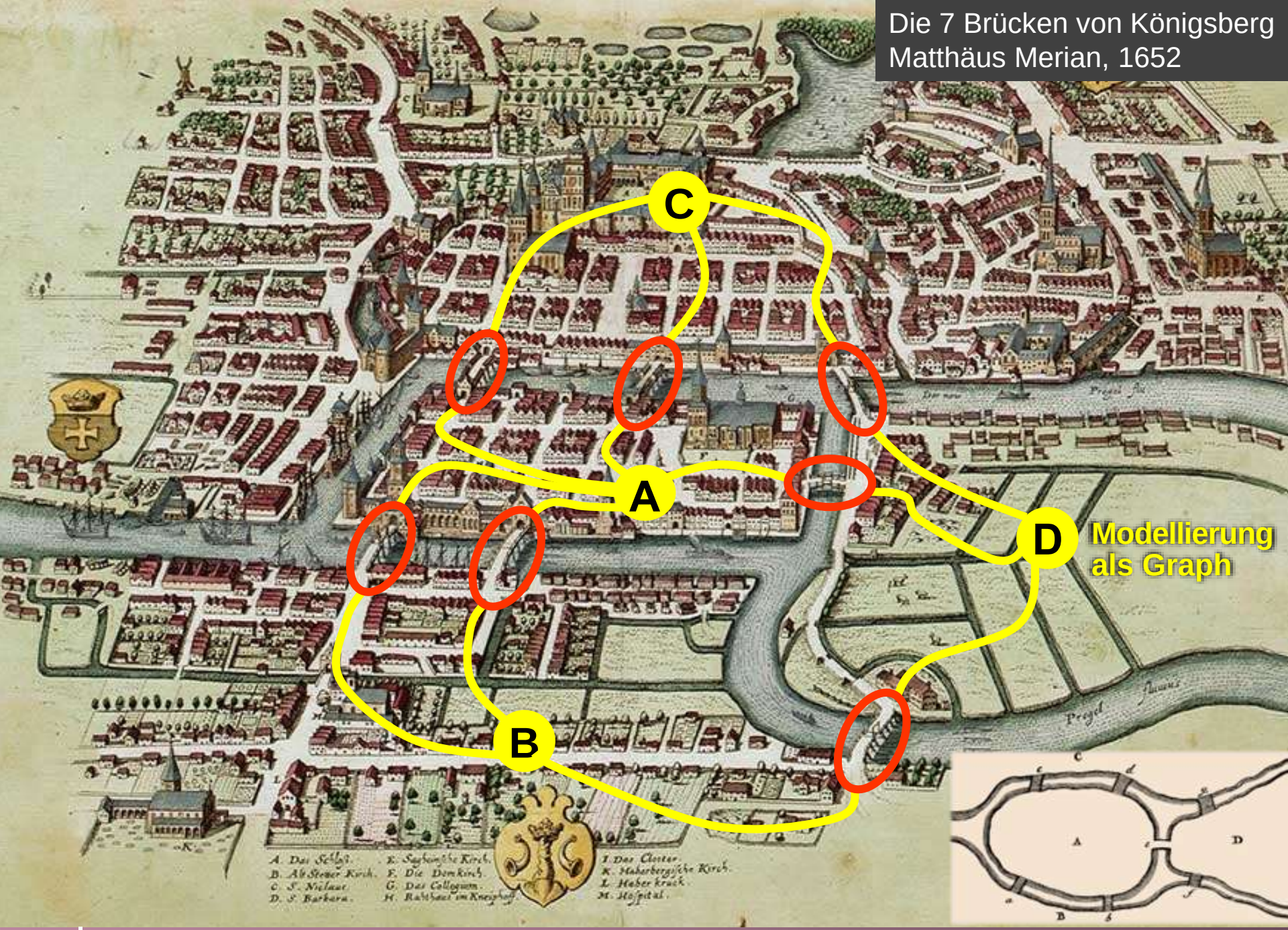
Wenn es indes überhaupt kein Gebiet gibt, für das die Zahl der Zugangsbrücken ungerade ist, so kann man den verlangten Spaziergang ausführen, gleichgültig in welchem Gebiet man beginnt.

Diese Regeln stellen somit eine vollständige Lösung des vorgelegten Problems dar.

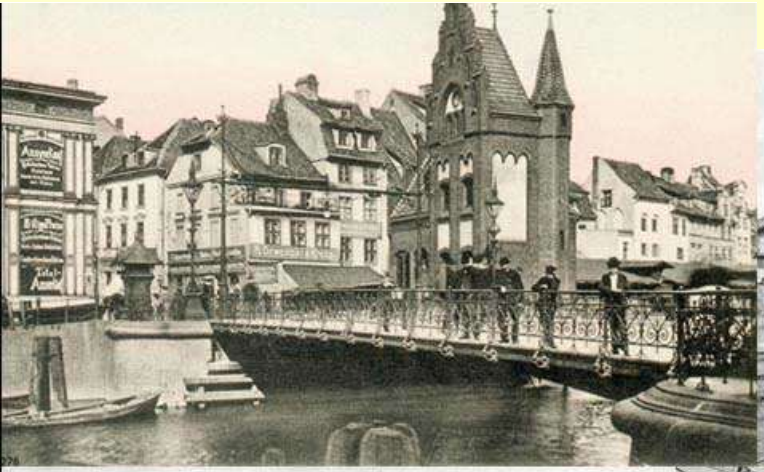


Stadtplan von Königsberg, Ausschnitt eines Kupferstichs von Joachim Bering, 1613

Die 7 Brücken von Königsberg
Matthäus Merian, 1652



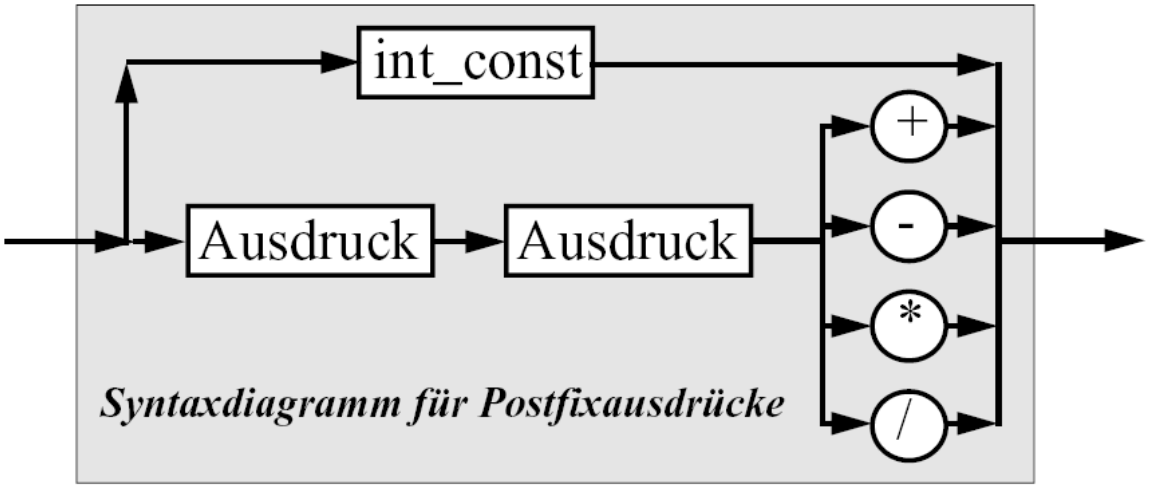
Die sieben Brücken und die erst 1905 erbaute Kaiserbrücke als 8. Brücke; bereits Euler schlug eine weitere Brücke vor, um das „Spaziergangproblem“ zu lösen



Kaliningrad heute, eine Exklave Russlands zwischen Polen und Litauen. Was ist aus den sieben Brücken geworden? Der Stadtkern wurde im zweiten Weltkrieg Ende August 1944 durch Luftangriffe fast vollständig zerstört. Die Ruinen der Kneiphof-Insel wurden in der Nachkriegszeit grossflächig abgeräumt und das eingeebnete Areal zu Grün- und Freiflächen umgewandelt; weitere Innenstadtgebiete wurden mit Hochhaussiedlungen in Plattenbauweise bebaut.

Postfix-Ausdrücke

- Bei Postfix-Ausdrücken kommt der Operator nach den zugehörigen Operanden, nicht dazwischen (infix)
  - Wird auch als „umgekehrte polnische Notation“ (UPN) bezeichnet
  - Entspr. Präfix-Ausdrücke: Operator vor seinen beiden Operanden
- Bsp: 2 3 + 4 5 * + entspricht infix (2 + 3) + (4 * 5)



Syntaxdiagramm für Postfixausdrücke

Die *Präfix-Notation* („polnische Notation“) wurde in den 1920er-Jahren vom polnischen Mathematiker Jan Lukasiewicz (1878–1956) entwickelt.

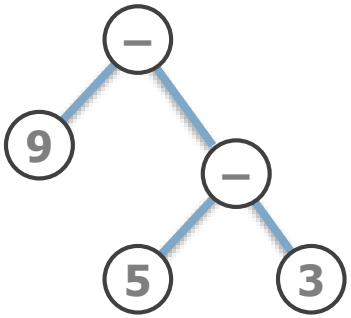
Postfix-Ausdrücke (2)

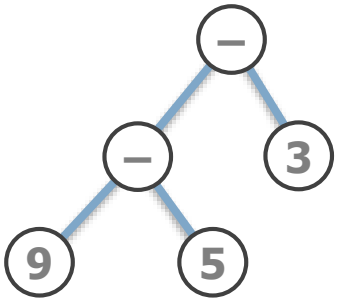
- Postfix-Ausdrücke sind für die maschinelle Verarbeitung besser geeignet als Infix-Ausdrücke (dazu später mehr)
  - Sie enthalten z.B. keine Klammern und sind dennoch eindeutig!
- Ziel daher: Automatische Umwandlung infix → postfix
- Idee am Beispiel `2 + 3 → 2 3 +`
  - D.h. von links nach rechts lesen und den Operator für den späteren Gebrauch zwischenspeichern
  - Entsprechend: `2 + Ausdruck → 2 Ausdruck +`
    - auch wenn „Ausdruck" sehr lang ist, der selbst wieder nach dem gleichen Prinzip übersetzt wird!
- Konkreter (rekursiver?) Algorithmus hierfür?
  - → Als kleine Übung

Postfix-Ausdrücke (3)

- Der Infix-Ausdruck 9-5-3 wird im Sinne der Linksassoziativität als (9-5)-3 interpretiert; wenn statt dessen 9-(5-3) gemeint ist, so sind Klammern entsprechend zu setzen. Postfix-Ausdrücke hingegen benötigen keine Klammern – einen a priori mehrdeutigen (der Infix-Notation 9-5-3 vergleichbaren) Ausdruck gibt es bei Postfix nicht!

Infix:	9-(5-3)	(9-5)-3
Postfix:	9 5 3 - -	9 5 - 3 -





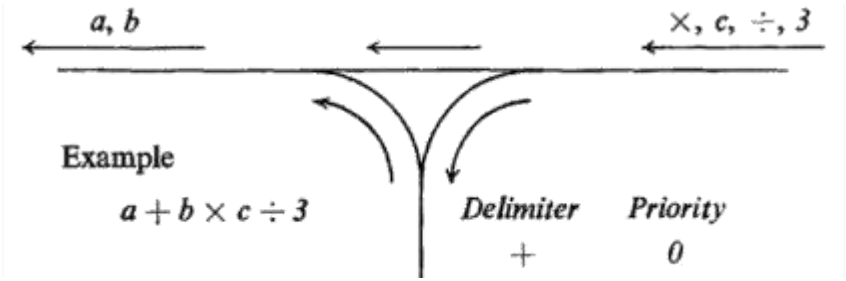
Umwandlung infix → postfix via Operatorbaum

- Eine erste Idee:
  - Infix-Ausdruck mit einem Parser analysieren
  - Dabei Operatorbaum aufbauen
  - Operatorbaum dann in postorder durchlaufen

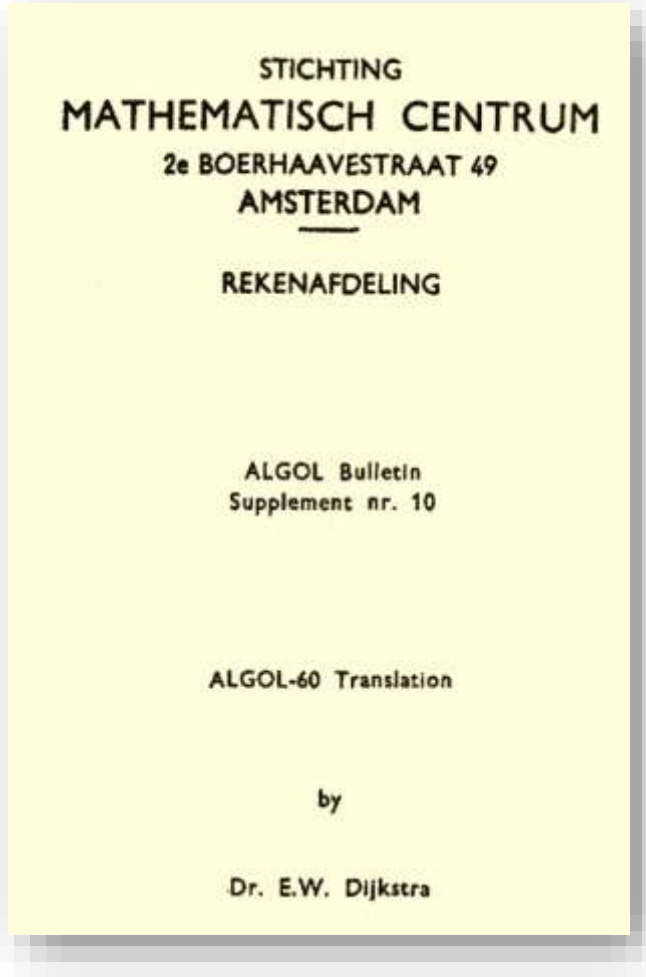
- Es geht aber auch ohne expliziten Operatorbaum!

Dijkstras Rangierbahnhof-Algorithmus

E.W. Dijkstra hatte die Postfixumwandlung 1961 in einer Veröffentlichung „Making a Translator for ALGOL 60" anhand eines Rangierbahnhofs mit Wendedreieck beschrieben: „The translation process shows much resemblance to a shunting at a three way railroad junction. At the right the symbols of the ALGOL text come in in order from left to right, at the left the successive orders of the object program are produced." [Information Bulletin No. 7, APIC, England, pp. 3-11, May 1961]



Der Input wird zeichenweise gelesen, wobei alle Zahlen direkt in die Ausgabe geschrieben werden. Falls das anstehende Zeichen ein Operationszeichen ist, wird es auf einen Stack gelegt. Falls bereits ein Operator auf dem Stack liegt, wird anhand der Operatorrangfolge und -assoziativität entschieden, ob der neue Operator direkt auf den Stack gelegt wird oder ob der Stack zuerst in den Output geleert wird. Öffnende Klammern werden ebenfalls auf den Stack gelegt, allerdings werden sie beim Entfernen nicht in den Ausgabestrom geschrieben. Bei schliessenden Klammern wird der Stack bis zum Antreffen einer öffnenden Klammer geleert.


STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

REKENAFDELING

ALGOL Bulletin
Supplement nr. 10

ALGOL-60 Translation

by

Dr. E.W. Dijkstra

Quelle: http://de.wikipedia.org/wiki/Shunting-yard-Algorithmus

Umwandlung infix → postfix mittels Stack

- Wir kommen ohne expliziten Baum aus, wenn wir beim zeichenweisen Lesen des Infix-Ausdrucks von links nach rechts einen Operator für den späteren Gebrauch zwischenspeichern

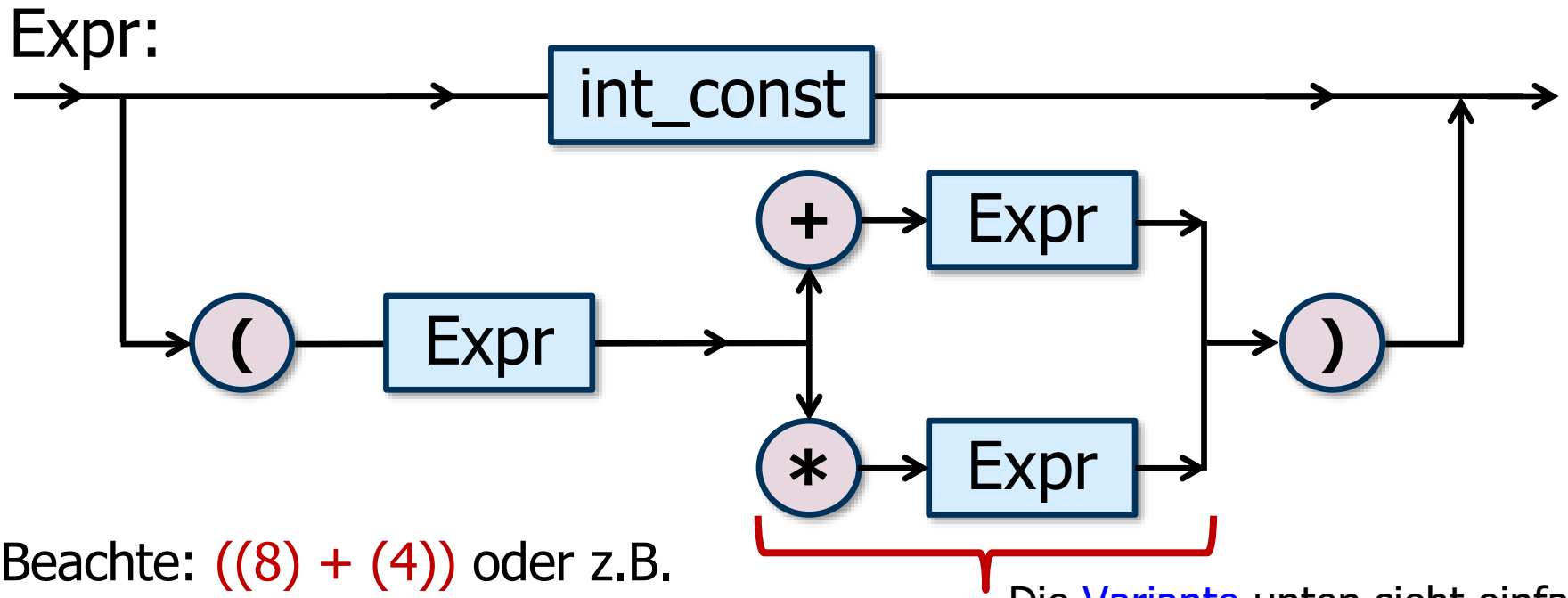
<Ausdruck1> **<OP>** <Ausdruck2>

⇒

<Ausdruck1> <Ausdruck2> **<OP>**

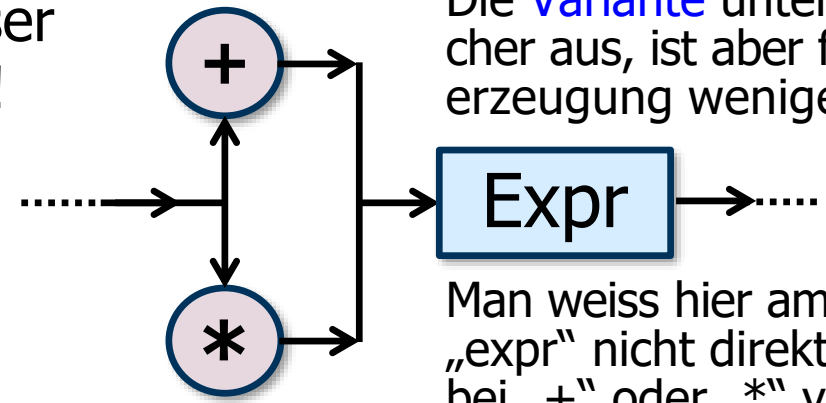
- Daher:
  - Operator in einen Stack; dort ruhen lassen
  - Inzwischen den Ausdruck2 (in analoger Weise) bearbeiten
  - Nach Ende von Ausdruck2: Operator aus dem Stack herausholen
    - Aber wie erkennt man das Ende?
    - Wir machen es uns hier einfach und fordern, dass jeder Teilausdruck geklammert ist; so erkennt man es an einer schliessenden Klammer „)"
    - Genauer: „vollständige Klammerung" → Jeder Infixoperator bringt ein Klammerpaar mit, das seinen linken und rechten Operanden umfasst

Vollständig geklammerte Infix-Ausdrücke



Beachte: ((8) + (4)) oder z.B. (3) sind entsprechend dieser „Grammatik" nicht korrekt!

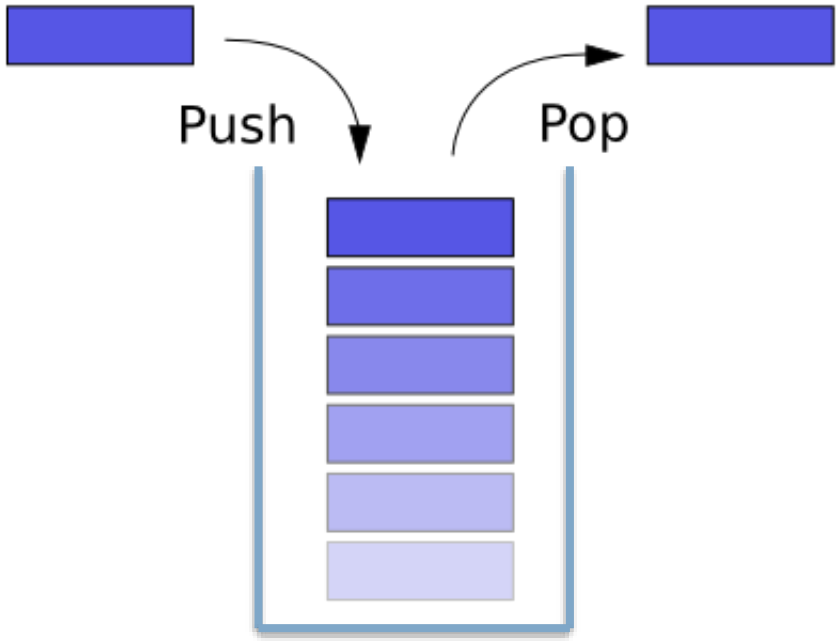
Die Variante unten sieht einfacher aus, ist aber für die Codeerzeugung weniger geeignet:



Man weiss hier am Ende von „expr" nicht direkt, ob man bei „+" oder „*" vorbeikam

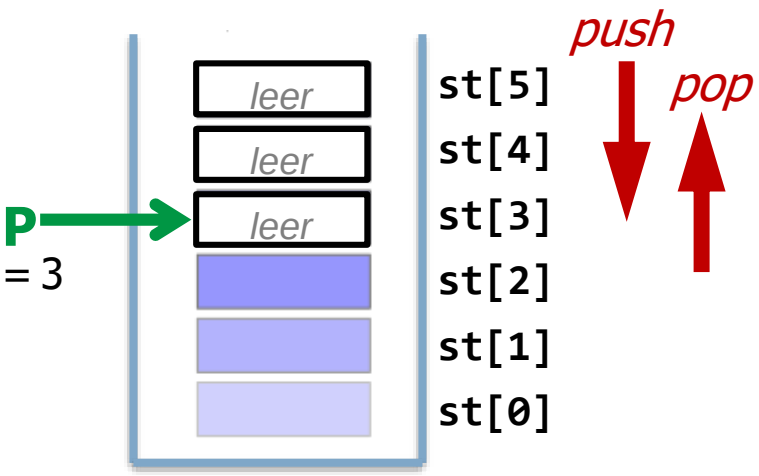
Ein Stack

Bereits aus Teil I der Vorlesung bekannt



- Stapel
- LIFO (Last In - First Out) ⟺ FIFO (First In - First Out)
- Push-down store
- Keller

Ein Stack – hier realisiert mit einem Array



- Realisiert als Klasse, die ein Array st nutzt
- Array-Grenzen sind durch 0 und length-1 abgesteckt
- p „zeigt" immer schon auf das nächste freie Element
- Hier: Stack speichert einzelne Zeichen („char-Stack")

```
class Stack {
  int p;
  char [] st;

  Stack(int size) {
    p = 0;
    st = new char[size];
  }

  void push(char c) {
    if (p >= st.length)
      System.out.println
      ("Stack Overflow");
    else
      st[p++] = c;
  }

  char pop() {
    return st[--p];
  }
}
```

Stackpointer

Konstruktor

Sonst Fehler bei Zugriff auf st[st.length]

Experten nutzen in solchen Fällen exceptions

Ein „stack underflow" sollte eigentlich auch überprüft werden!

Umwandlung infix → postfix mittels Stack

- Wir beschränken uns hier auf die Operatoren + und * sowie auf einziffrige Operanden; der Infix-Ausdruck sei vollständig geklammert

(Ausdruck1 <OP> Ausdruck2)
⇒
Ausdruck1 Ausdruck2 <OP>

- Lösungsidee:
  - Gesamtausdruck von links nach rechts zeichenweise verarbeiten
  - Operanden (d.h. Zahlen) werden sofort ausgegeben
  - Operatoren (+, *) kommen in den Stack
  - Bei jeder schliessenden Klammer: obersten Operator aus dem Stack holen (pop) und ausgeben
  - Offenbar spielen öffnende Klammern keine Rolle, daher „(" ebenso wie Leerzeichen etc. einfach überlesen!

Man spiele folgende Testfälle durch:

```
((a+b)*c)
(a+(b*c))
(a*(b+c))
```

Aber heisst das etwa, dass öffnende Klammern von vornherein überflüssig sind? Wir können diese doch kaum in allen Mathematikbüchern einfach schadlos ausradieren, oder?!

Umwandlung infix → postfix: Java-Programm

```
class Stack  // Wie gehabt...

class InfToPost {
  Stack stk = new Stack(1000);
  char c; // Lookahead-Zeichen

  boolean eof(char c) {
    return (c == (char) -1);
  }

  void convert() {
    while (!eof(c = KbdInput.getc())) {
      if ((c=='+') || (c=='*'))
        stk.push(c);
      if ((c >= '0') && (c <= '9'))
        System.out.print(" " + c);
      if (c == ')')
        System.out.print(" " + stk.pop());
    } // end while
  System.out.println();
  }
} // end class InfToPost
```

Verschachtelungstiefe von max. 1000 sollte reichen

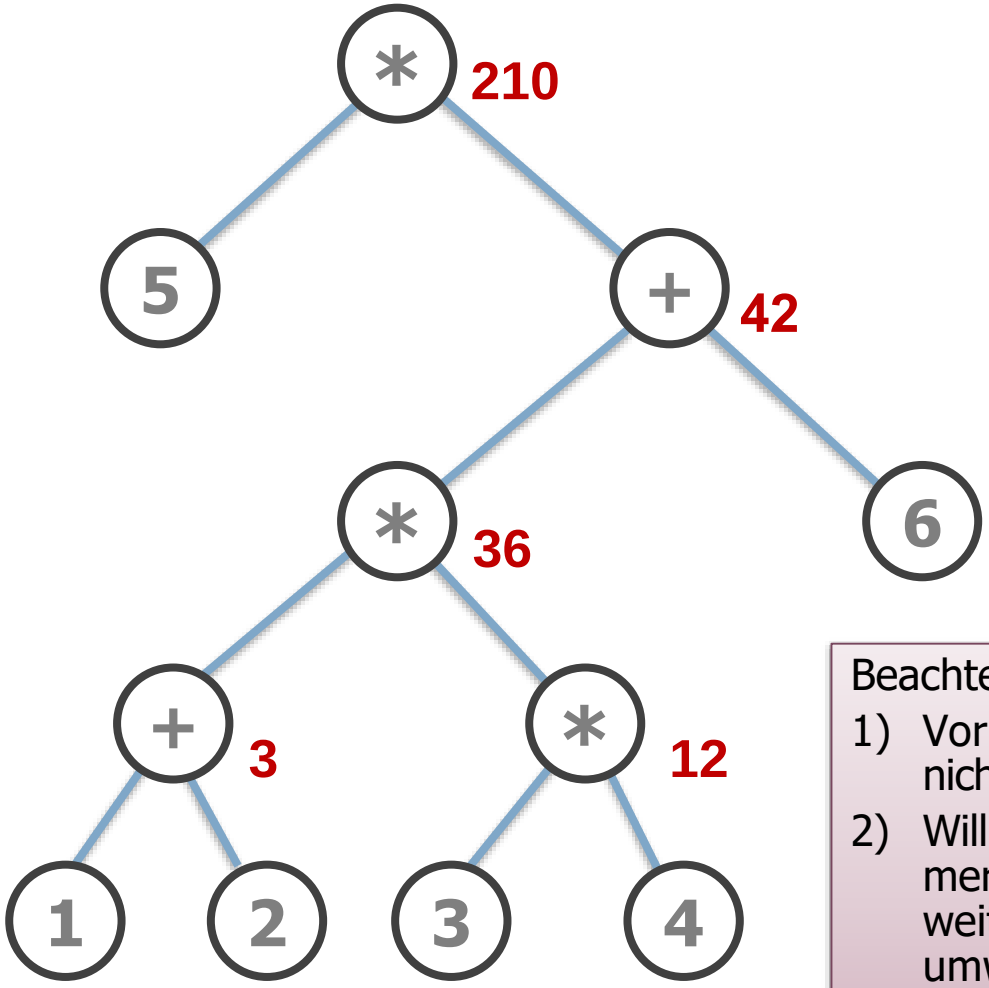
Bei Dateiende ("**e**nd **o**f **f**ile") wird ein Spezialzeichen geliefert mit dem Wert (char)-1

Als Nebeneffekt bei der Zuweisung an c wird der Wert auch an eof weitergegeben

Ziffern direkt ausgeben

Testbeispiel:
(5*(((1+2)*(3*4))+6))
wird umgewandelt in
5 1 2 + 3 4 * * 6 + *

Der Baum zu infix (5*(((1+2)*(3*4))+6))
bzw. postfix 5 1 2 + 3 4 * * 6 + *



Das ist das Testbeispiel der letzten Slide; es dient auch als Beispiel für die Postfix-Auswertung auf den nachfolgenden Slides

Beachte:

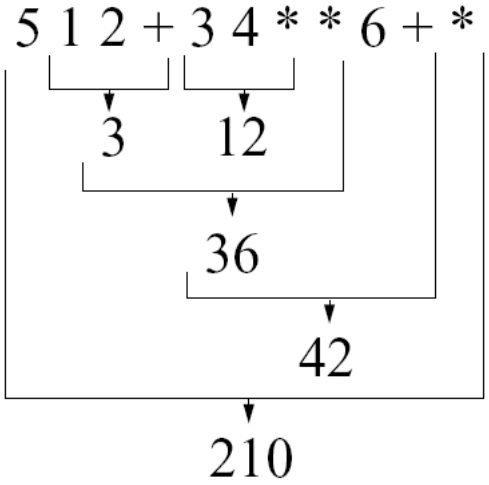
1) Vorheriges Programm funktioniert z.B. nicht bei ((3)+(4)); ist das ein Problem?
2) Will man auch nicht vollständig geklammerte Infix-Ausdrücke oder solche mit weiteren Operatoren (minus, dividiert) umwandeln, wird es etwas komplizierter; wir gehen hier aber nicht darauf ein

Auswertung von Postfix-Ausdrücken

- Idee: Zeichenweise von links nach rechts lesen, bis man auf einen Operator trifft; dann diesen auf die beiden vorangehenden Operanden (d.h. Zahlen) anwenden

Beispiel: 5 1 2 + 3 4 * * 6 + *

(Hier nur einziffrige Operanden!)

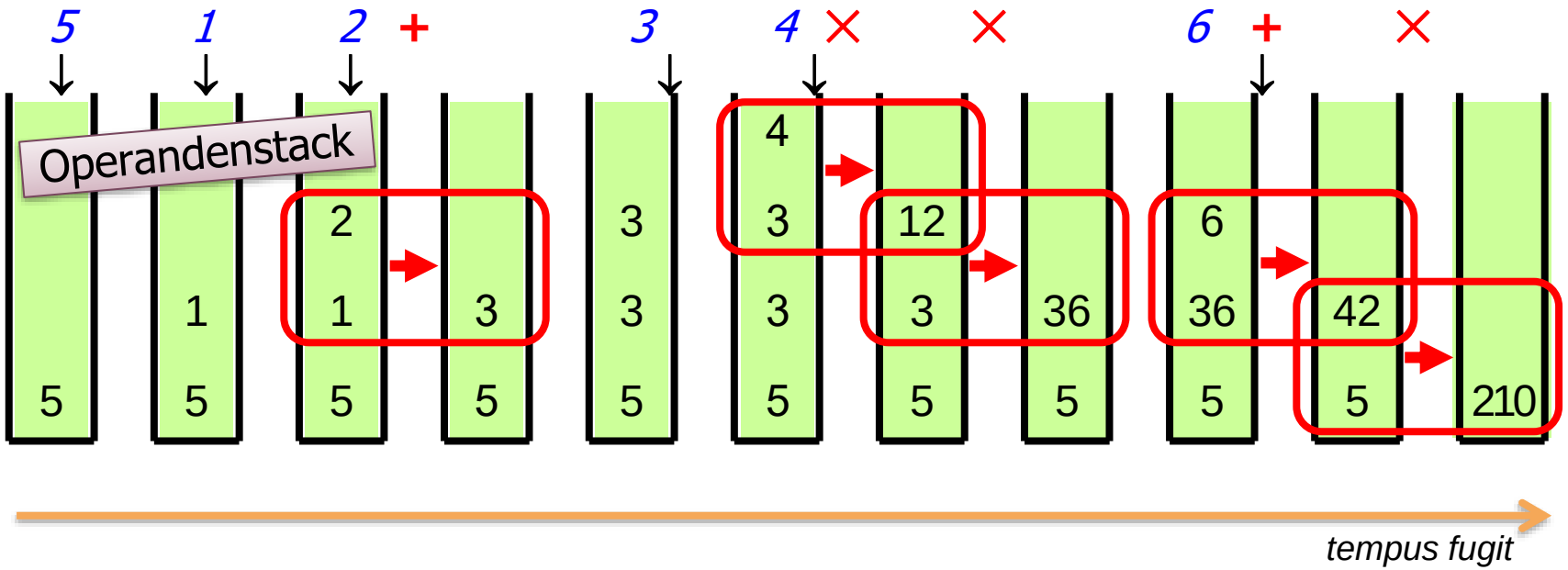


- Operanden nacheinander in einen Stack pushen
- Bei einem Operator: Die obersten beiden Operanden aus dem Stack holen und verknüpfen
- Das Resultat wieder in den Stack pushen
- Am Ende steht das Resultat alleine im Stack

Auswertung von Postfix-Ausdrücken

- Idee: Zeichenweise von links nach rechts lesen, bis man auf einen Operator trifft; dann diesen auf die beiden vorangehenden Operanden (d.h. Zahlen) anwenden

Beispiel: 5 1 2 + 3 4 * * 6 + *



Ein Postfix-Auswerter in Java

Nach der eben geschilderten Idee mittels Operandenstack

```
public static void main(String args[]) {
  Stack stk = new Stack(1000); // hier: int-Stack (für Operanden)
  int x; char c = ' '; // Lookahead-Zeichen
  while (!eof(c)) {
    if (!(c == '+' || c == '*' || (c >= '0' && c <= '9') )) {
      c = KbdInput.getc();
      continue;
    }
    if (c == '+') {
      stk.push(stack.pop() + stack.pop());
      c = KbdInput.getc();
      continue;
    }
    if (c == '*') {
      stk.push(stack.pop() * stack.pop());
      c = KbdInput.getc();
      continue;
    }
    x = 0;
    while (c >= '0' && c <= '9') {
      x = 10 * x + Character.digit(c,10);
      c = KbdInput.getc();
    }
    stk.push(x);
  }
  System.out.println(stk.pop());
}
```

Alles Fremde einfach überlesen; z.B. auch Leerzeichen, newline, ...

Die Operation '+' oder '*' werden auf die beiden obersten Stackelemente angewendet, das Ergebnis davon gleich wieder auf den Stack gepusht

Mehrziffrige Operanden zusammenbauen und den berechneten int-Wert auf den int-Stack legen (→ nächste Slide)

Ausgabe des Endergebnisses

Umwandlung von Ziffernfolgen in Zahlen

- Verschiedene Operanden sind durch Leerzeichen getrennt; mehrstellige Operanden enthalten keine Leerzeichen
- Beim Test `c >= '0' && c <= '9'` wird die Tatsache verwendet, dass die Ziffer-zeichen 0 bis 9 im Zeichensatz hintereinander stehen
- Mit `Character.digit(c,10)` wird das Zeichen in der Variablen c als Ziffer im Dezimalsystem interpretiert und nach int konvertiert

Horner-Schema für Polynome

```
while (c>='0' && c<='9')
  { x = 10*x +
    Character.digit(c,10);
    c = KbdInput.getc(); }
stk.push(x);
```

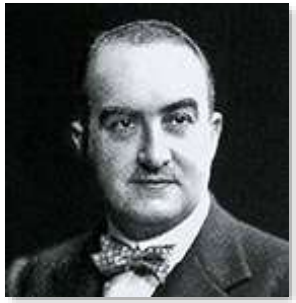
- Der Wert einer Zahl Z in Stellenschreibweise $c_n c_{n-1} \ldots c_0$ (mit Basis b > 1 und Ziffern $0 \le c_i < b$) ergibt sich aus

$$Z = \sum c_i b^i = (\ldots(c_n)b+c_{n-1})b+\ldots+c_1)b+c_0$$ (Horner-Schema: b ausklammern)

  - Wir verwenden Dezimalzahlen; daher b = 10
  - Ganz ähnlich könnte man z.B. auch binäre (b = 2) oder hexadezimale (b = 16) Operanden zulassen und nach int konvertieren – wir kennen die Binär- und Hexadezimaldarstellung ja schon aus „Informatik I"

Stichwort „Horner-Schema“

Historische Notiz

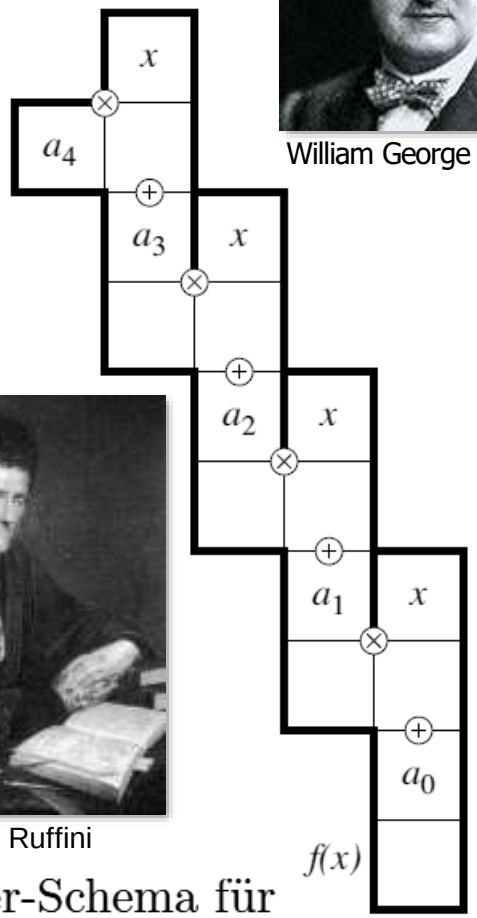


William George Horner

Das Horner-Schema nach William George Horner (1786–1837) ist ein Umformungsverfahren für Polynome, um die Berechnung von Funktionswerten zu erleichtern. Durch fortgesetztes Ausklammern der freien Polynomvariablen x wird das Polynom als Schachtelung von Produkten und Summen dargestellt. Im umgeformten Polynom kommen keine Potenzen, sondern nur noch eine minimale Anzahl von Multiplikationen und Additionen vor.

Das Horner-Schema ist 1819 der Royal Society vorgelegt worden und noch im selben Jahr in den Philosophical Transactions of the Royal Society publiziert. Allerdings war Horner nicht der Erste, der diese Methode entdeckte, sie war schon einige Jahrhunderte früher chinesischen und persischen Mathematikern bekannt. Ferner veröffentlichte Paolo Ruffini (1765 – 1822) 15 Jahre vor Horner bereits ein ähnliches Verfahren; dieses wird daher z.B. in Italien auch als regola di Ruffini bezeichnet.

Paolo Ruffini



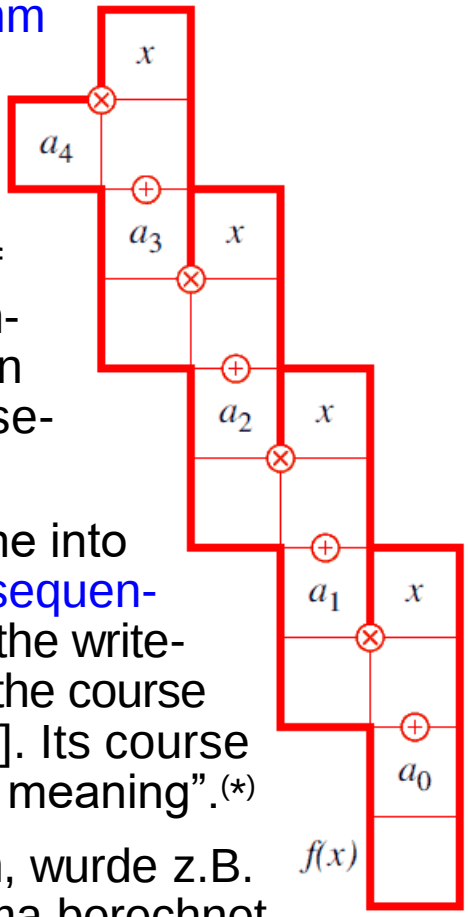
Horner-Schema für

$$f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$
$$= (((a_4x + a_3)x + a_2)x + a_1)x + a_0$$

[Quelle u.a. Wikipedia]

Rechenschemata als Proto-Programme

Als Rechenschema stellt das Horner-Schema eine Art Programm (zur einfachen und effizienten Polynomauswertung) in Form von „Anweisung an den menschlichen Geist“ dar – man fügt in das (evtl. sogar formularhaft gegebene) Schema die Parameter (also die Koeffizienten a_i und den konkreten Wert von x) an die gekennzeichneten Leerstellen ein und wendet stur (im Kopf oder auf einem Hilfsblatt, evtl. unter Zuhilfenahme einer Rechenmaschine) die vorgegebenen elementaren Rechenoperationen an, wobei man die jeweiligen Zwischenresultate an die vorgesehenen Formularplätze schreibt.



F.L. Bauer bemerkte dazu: „Long before the word software came into general use, it was necessary to describe longer, complicated sequences of calculation. For this purpose, it was customary to support the write-up of intermediate results on paper by means of forms indicating the course of calculation. For example, the well-known Horner scheme [...]. Its course of calculation can be described by a printed blank with obvious meaning”.(*)

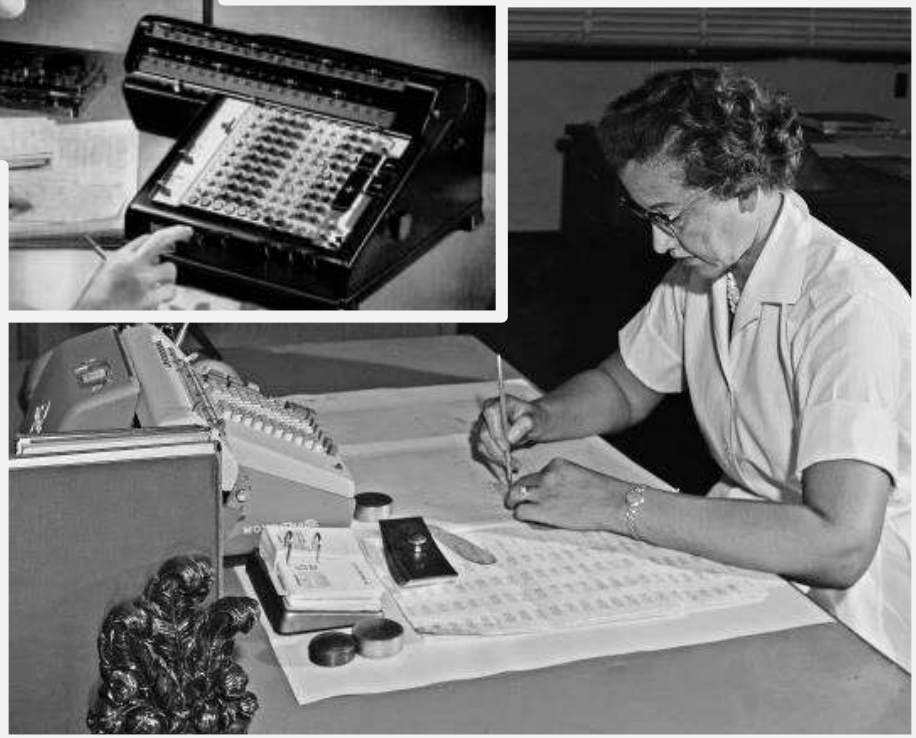
Bevor standardisierte Computerprogramme zum Einsatz kamen, wurde z.B. in Deutschland die Einkommenssteuer nach dem Horner-Schema berechnet, um Rundungsfehler bei der Berechnung zu minimieren und die Rechtssicherheit im Rahmen der Gleichbehandlung (durch ein und dasselbe Verfahren) zu gewährleisten.

*) F.L. Bauer: A computer pioneer’s talk: pioneering work in software during the 50s in Central Europe. In: *History of Computing: Software Issues*. Springer, 2002. S. 11-22.

Rechenschablonen zur Programmierung menschlicher Rechner

Langwierige, aber eher schematische Rechnungen wurden in der ersten Hälfte des 20. Jahrhunderts, also noch vor dem Aufkommen von Computern (bzw. „programmgesteuerten Rechenautomaten“), für eine zunehmende Zahl industrieller und militärischer Anwendungen erforderlich. An Alwin Walthers Institut für Praktische Mathematik der Technischen Hochschule Darmstadt entwarf Paul Terebesi 1930 einen Satz von 26 nacheinander anzuwendenden Schablonen zur schematischen Berechnung von Fourier-Koeffizienten für die harmonische Analyse periodischer Funktionen (nach dem Verfahren von Carl Runge und Hermann König mit 24 äquidistanten Ordinatenwerten), mit denen auch weniger mathematisch versierte Hilfskräfte umgehen konnten, die am Institut umfangreiche Rechnungen unter Nutzung von Tischrechenmaschinen zu erledigen hatten. Terebesis Kartonschablonen wurden 1930 publiziert und in der Folge auch von Anderen verwendet.

Menschliche Rechner



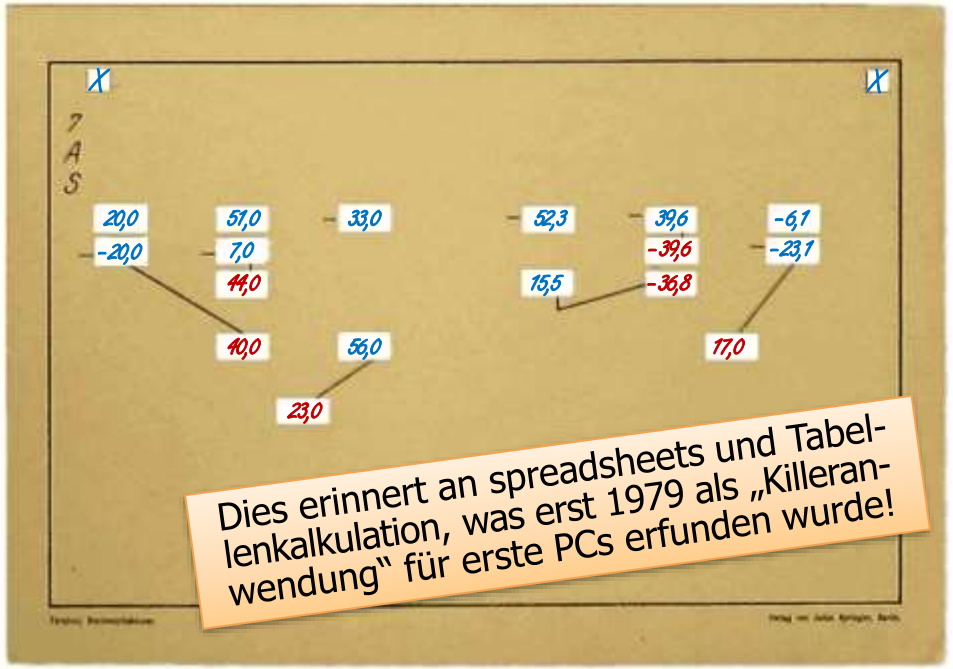
Rechenschablonen

It took several hours to fill out a 24 point stencil and today this computation is performed in micro-seconds or even more rapidly. – Philip J. Davis

Terebesi spricht vom „Mechanisieren“ des Verfahrens und meint: „Wer die vier Grundrechenarten der Volksschule beherrscht, kann mit den Schablonen auch harmonische Analyse schnell und mühelos ausführen [...] Dadurch ist es möglich, die lästige Rechenarbeit von ungeschulten Hilfskräften erledigen zu lassen.“

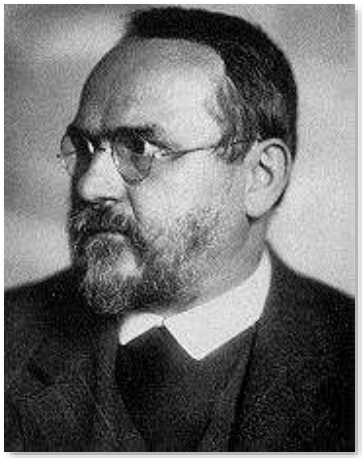
Der Astronom Karl Stumpff beschreibt in seinem Lehrbuch „Grundlagen und Methoden der Periodenforschung“ von 1937 die Methode so:

„Das ökonomische Prinzip, das durch diese Schablonen befolgt wird, besteht darin, dass jede in der Rechnung vorkommende Zahl nur ein einziges Mal hingeschrieben zu werden braucht. So ist zunächst für die 24 Beobachtungswerte selbst eine bestimmte Anordnung vorgesehen. Der Rechner legt eine Schablone, die 24 rechteckige Fenster in bestimmter Anordnung und Nummerierung enthält, auf das Rechenpapier und schreibt die gegebenen Werte in die Fenster hinein. Eine zweite Schablone dient dem nächsten Rechnungsgang: Es werden Summen und Differenzen gebildet – die Stellen, an denen sie niedergeschrieben werden müssen, sind wiederum durch Fenster bezeichnet, die Anleitung ist auf der Schablone durch Pfeile, Vorzeichen und ähnliche Hinweise vermerkt. So wird eine ganze Reihe von Schablonen in bestimmter Reihenfolge benutzt, bis schließlich die Fourier-Koeffizienten gebildet sind.“



Rechenschemata für Ungeschulte

Von vielen Zeitgenossen wird die Einfachheit des Verfahrens sowie die Anwendungsmöglichkeit durch mathematisch Ungeschulte und Praktiker hervorgehoben. Der Mathematiker Lothar Schrutka von der Technischen Hochschule Wien kommentierte seinerzeit die Nützlichkeit für die „Mechanisierung“ des Rechnens beispielsweise so: [Monatshefte für Mathematik und Physik, Dez. 1931, 38(1), S. A44 – A45]



„Für das bekannte Verfahren von Runge zur harmonischen Analyse einer periodischen Funktion ist hier eine Anordnung aufgestellt, die durch Anwendung von Schablonen (Kartonblättern mit Ausschnitten) sozusagen ‚zwangsläufig‘ gemacht ist, so dass die Durchführung der Rechnung auch Hilfskräften übertragen werden kann. [...] hat man hier den Vorteil, dass kein Formular erforderlich ist, vielmehr zum Anschreiben der Zahlen für eine besondere Aufgabe jedes Blatt Papier verwendet werden kann. [...] Auch ist dafür gesorgt, dass die Multiplikationen mit demselben Faktor stets im selben Rechnungsgang zu erledigen sind, so dass die Benutzung des Rechenschiebers oder einer Rechenmaschine in günstiger Weise erfolgen kann.“

Rudolf Zurmühl von der TU Berlin (Promotion 1939 bei Alwin Walther) meint: „Die Anlage gut überlegter und übersichtlicher Rechenschemata, die die gesamte Zahlenrechnung enthalten sollen und nach denen die Rechnung weitgehend schematisch abläuft, hilft Fehler vermeiden und erlaubt es vor allem, die Rechnung angelernten Hilfskräften zu übertragen.“ [Rudolf Zurmühl: Praktische Mathematik für Ingenieure und Physiker, Springer-Verlag, 1953]

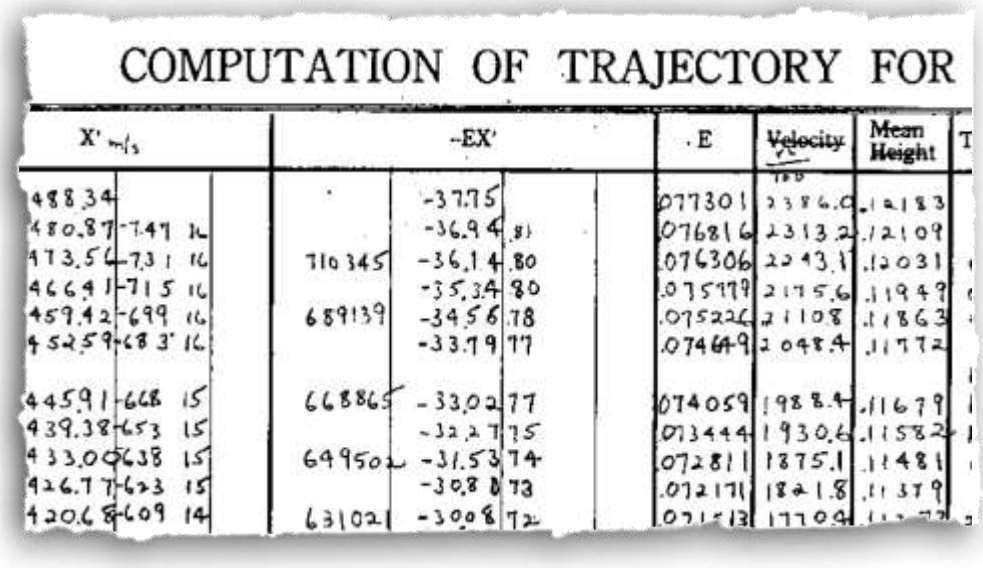
Was von mathematisch ungeschulten Hilfskräften mechanisch ausgeführt werden kann, das sollte dann aber auch ein programmierter Blechkasten können – Bauingenieurstudent Konrad Zuse wurde wohl seinerzeit durch die Rechentabellen und -formulare zur Bestimmung von Flächenmomenten bei der Baustatik auf die Idee des Computers gebracht!

Life as a Computer

Von seinem Alltag als menschlicher Rechner Ende der 1940er-Jahre am Ballistic Research Laboratory (BRL) der amerikanischen Armee in Aberdeen, Maryland, berichtet Austin Robert Brown:

“My supervisor, Gertrude Kuhlman, would give me a huge sheet of paper divided into rows and columns. Down the rows in the left-hand columns were given sets of angles, azimuths, and elevations, corresponding to observations of the rocket at successive times in its flight. Across the top of the paper the columns were numbered and labeled, such as sin A, cos A, col. 1 x col. 3, etc. Using a book of trigonometric functions and an electromechanical calculator (I usually used the Monro-Matic), I would fill in the blanks from the upper-leftmost cell to the lower-rightmost cell on the sheet of paper, turn it in to Mrs. Kuhlman, get another sheet, do the same to it, etc.”

Im Jahr 1993 merkt er an: “Nowadays we would say I filled in a spread-sheet and would have a personal (electronic) computer do the work; such tools (neither hardware nor software) did not exist in 1949.”

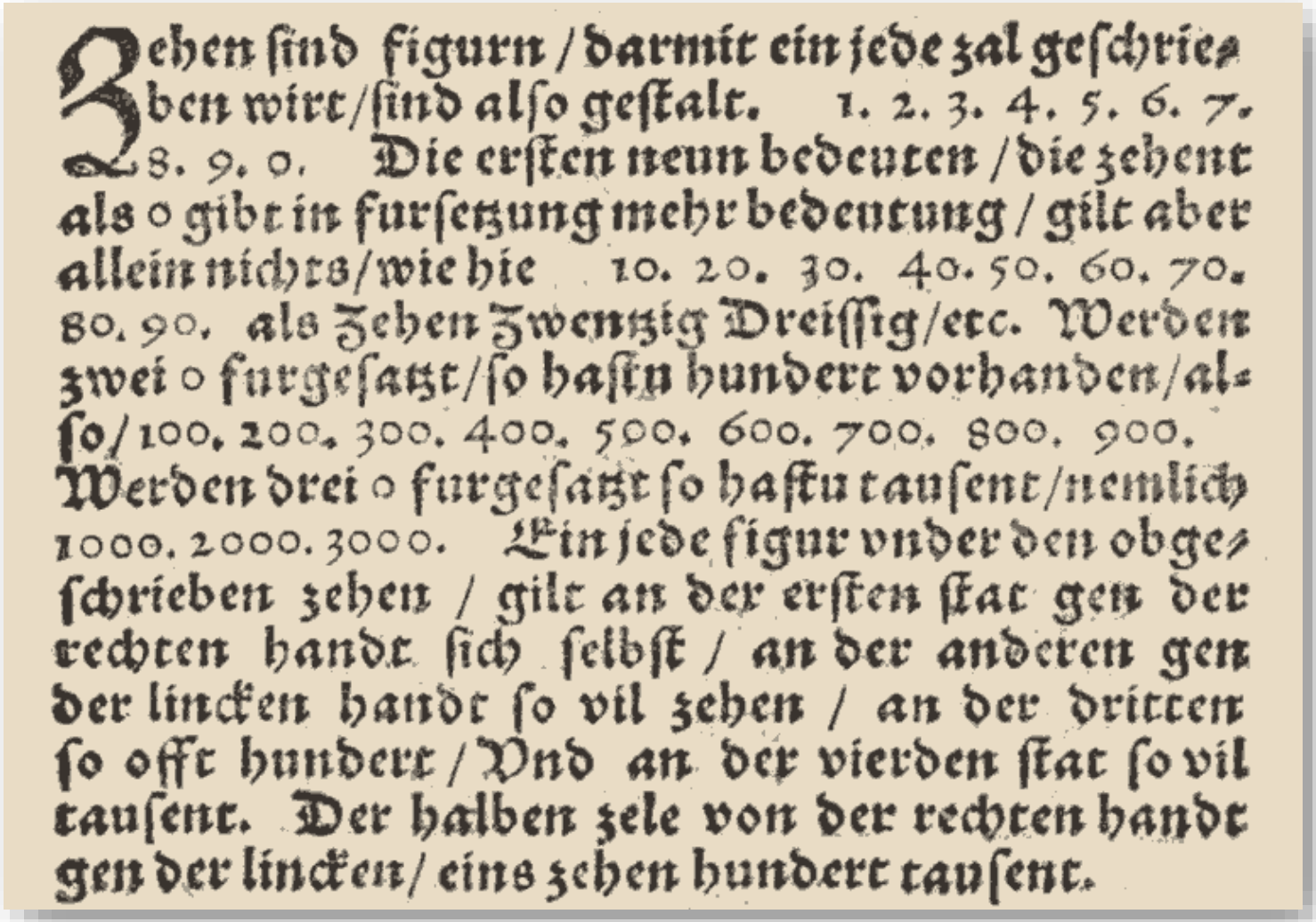

COMPUTATION OF TRAJECTORY FOR

X'			-EX'			E	Velocity	Mean Height	T
488.34				-37.75		077301	2384.0	.12183	
480.87	-7.47	16		-36.94	81	076816	2313.2	.12109	
473.56	-7.31	16	710345	-36.14	80	076306	2243.1	.12031	
466.41	-7.15	16		-35.34	80	075779	2175.6	.11949	
459.42	-6.99	16	689139	-34.56	78	075226	2110.8	.11863	
452.59	-6.83	16		-33.79	77	074649	2048.4	.11772	
445.91	-6.68	15	668865	-33.02	77	074059	1988.4	.11679	
439.38	-6.53	15		-32.27	75	073444	1930.6	.11582	
433.00	-6.38	15	649502	-31.53	74	072811	1875.1	.11481	
426.77	-6.23	15		-30.81	73	072171	1821.8	.11379	
420.68	-6.09	14	631021	-30.08	72	071513	1770.9	.11272	

[W. Barkley Fritz: ENIAC – a problem solver. IEEE Annals of the History of Computing 16.1 (1994): 25-45]

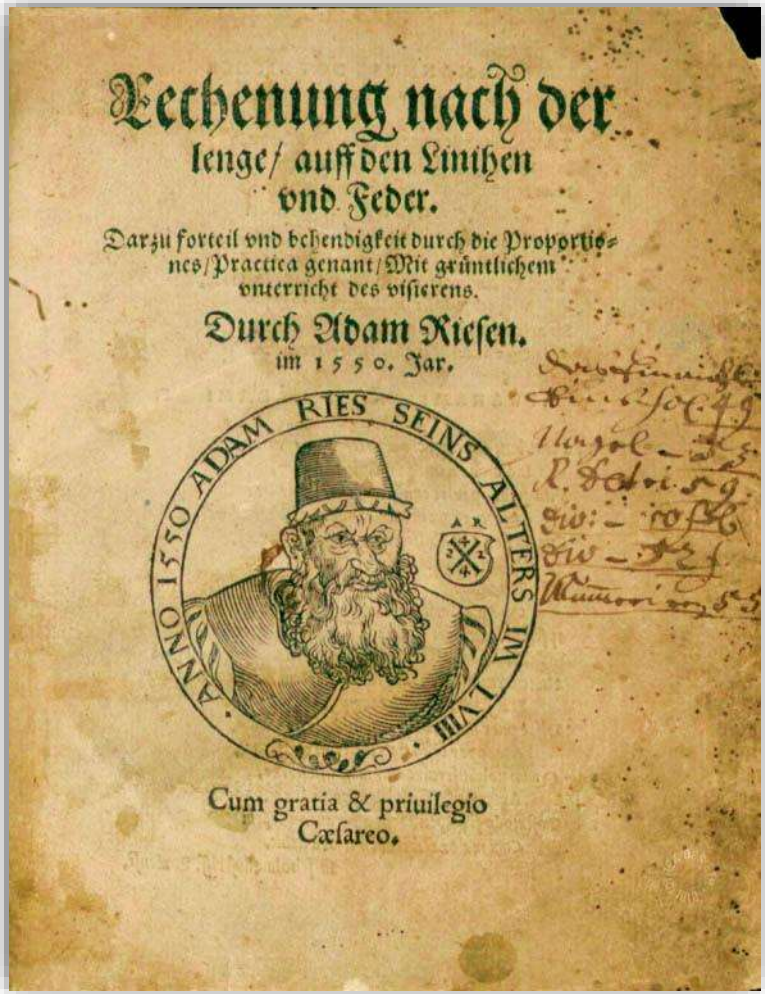
Stellenschreibweise laut Adam Riese (AD 1550)

Unser Umwandlungsschema Ziffernfolgen → Zahlen fusst auf der Stellenschreibweise.
Heute uns „selbstverständlich“, war ihre Einführung ein kultureller Meilenstein.



Zehen ſind figurn / darmit ein jede zal geſchrie-
ben wirt / ſind alſo geſtalt. 1. 2. 3. 4. 5. 6. 7.
8. 9. 0. Die erſten neun bedeuten / die zehent
als o gibt in furſetzung mehr bedeutung / gilt aber
allein nichts / wie hie 10. 20. 30. 40. 50. 60. 70.
80. 90. als Zehen Zwentzig Dreiſſig / etc. Werden
zwei o furgeſatzt / ſo haſtu hundert vorhanden / al-
ſo / 100. 200. 300. 400. 500. 600. 700. 800. 900.
Werden drei o furgeſatzt ſo haſtu tauſent / nemlich
1000. 2000. 3000. Ein jede figur vnder den obge-
ſchrieben zehen / gilt an der erſten ſtat gen der
rechten handt ſich ſelbſt / an der anderen gen
der lincken handt ſo vil zehen / an der dritten
ſo offt hundert / Vnd an der vierden ſtat ſo vil
tauſent. Der halben zele von der rechten handt
gen der lincken / eins zehen hundert tauſent.

Stellenschreibweise laut Adam Riese (AD 1550)



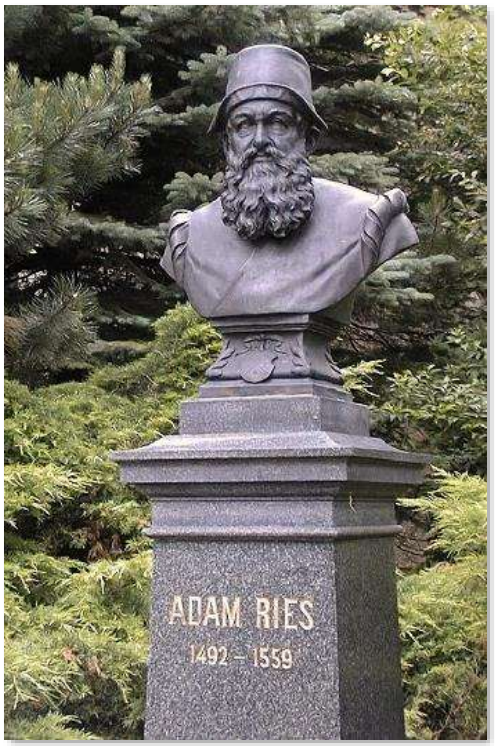
http://dx.doi.org/10.3931/e-rara-81

Adam Ries erklärt die Stellenschreibweise in seinem dritten Rechenbuch („Rechenung nach der lenge / auff den Linihen und Feder") 1550 unter der Überschrift „Numerirn / Zelen" wie folgt:

Zehen sind figurn / darmit ein jede zal geschrieben wirt / sind also gestalt. 1. 2. 3. 4. 5. 6. 7. 8. 9. 0. Die ersten neun bedeuten / die zehent als 0 gibt in fursetzung mehr bedeutung / gilt aber allein nichts / wie hie 10. 20. 30. 40. 50. 60. 70. 80. 90. als Zehen, Zwentzig, Dreissig / etc. Werden zwey 0 furgesatzt / so hastu hundert vorhanden / also 100. 200. 300. 400. 500. 600. 700. 800. 900. Werden drey 0 furgesatzt / so hastu tausent / nemlich 1000. 2000. 3000. Ein jede figur vnder den obgeschrieben zehen / gilt an der ersten stat gen der rechten handt sich selbst / an der anderen gen der lincken handt so vil zehen / an der dritten so offt hundert / Und an der vierden stat so vil tausent. Der halben zele von der rechten handt gen der lincken / eins zehen hundert tausent.

Man kann übrigens vermuten, dass die sprachliche Verwandtschaft des Zahlwortes „zehn" mit „Zehen" kein Zufall ist, vgl. die Abstammung des englischen Wortes „digit" für „Ziffer" vom lateinischen *digitus* (Finger).

Adam Ries und seine Rechenbücher

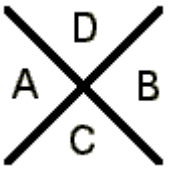


Adam Ries wurde 1492 (im Jahr, als Christoph Kolumbus Amerika entdeckte) geboren. 1518 erschien in Erfurt sein erstes Rechenbuch. Rechnen auf den Linien eines Rechenbretts, praktische Aufgaben aus dem Wirtschaftsleben, das Beherzigen des didaktischen Prinzips vom Einfachen zum Schwierigen und das ausführliche Beschreiben von Lösungsverfahren, nicht jedoch deren Begründungen, waren Kennzeichen dieses Rechenbuches. Adam Ries betrieb in Erfurt (wie auch später in Annaberg) eine Rechenschule. 1522 erschien das zweite Rechenbuch, das den Ruhm von Adam Ries begründete. Das Rechnen auf den Linien wurde darin nur noch kurz gefasst, im Mittelpunkt stand nun das Ziffernrechnen mit indisch-arabischen Ziffern. Über 120 Auflagen sind nachweisbar. 1522/23 übersiedelte Adam Ries von Erfurt nach Annaberg, wo er als Beamter der sächsischen Bergverwaltung tätig wurde. Im Jahr 1550 erschien sein drittes Rechenbuch („Rechenung nach der lenge / auff den Linihen und Feder“), das als die beste deutsche Arithmetik in der Mitte des 16. Jahrhunderts gilt. Ergänzend zu seinen früheren Büchern hat Ries hier auch das „Visieren“ behandelt, die zu seiner Zeit sehr wichtige Berechnung des Inhalts von Fässern. Adam Ries starb 1559 in Annaberg. Als Mathematiker war er auf der Höhe seiner Zeit, erbrachte jedoch praktisch keine eigenen originären Beiträge. Seine überragenden Verdienste liegen in der weiten Verbreitung des Rechnens in allen Bildungsschichten des Volkes. Er hat mit seinen Werken vor allem in Deutschland dazu beigetragen, dass die römische Zahlendarstellung als unhandlich erkannt und weitgehend durch die nach dem Stellenwertsystem strukturierten indisch-arabischen Zahlzeichen ersetzt wurde. Die Riesschen Rechenbücher kamen erst im 18. Jahrhundert allmählich ausser Gebrauch. Ries publizierte auf Deutsch und leistete damit auch einen wichtigen Beitrag zu dessen Normierung.

Rechenmeister Adam Ries

Auf beiden Briefmarken der deutschen Bundespost (1959 zum 400. Todestag und 1992 zum 500. Geburtstag) sieht man ein „Markenzeichen“ von Adam Ries, ein Kreuz mit vier Zahlen. Auf den Briefmarken sind die Zahlen paarweise gleich, was aber nicht sein muss, denn das Zahlenkreuz stellt die Neunerprobe beim Addieren dar. Für die Probe einer Rechnung c = a + b stellt man das Zahlenkreuz mit den „Neunerresten“ (Quersumme mod 9 bzw. äquivalent dazu: iterierte Quersumme) so auf:

A = Neunerrest von a;
B = Neunerrest von b;
C = Neunerrest von c;
D = A + B.



Die Probe geht auf, wenn C = D oder C + 9 = D. Dann ist die Rechnung wahrscheinlich korrekt. (Für die Probe bei einer Subtraktion oder einer Multiplikation wird das Schema leicht variiert).

Die Neunerprobe wird schon von al-Chwarizmi für die Multiplikation besprochen (dabei wird allerdings der Neunerrest direkt mit mod 9, nicht mittels Quersumme, ermittelt.)

Rechen- und Schulmeister

Rechenmeister war einen mittelalterlichen Beruf, der sich im 14. Jahrhundert in Italien aufgrund der von dort ausgehenden wirtschaftlichen Veränderungen entwickelte und im deutschen Sprachraum im 16. Jahrhundert seine grösste Verbreitung fand. Er schloss eine Ausbildungslücke zwischen dem mit dem rasch wachsenden Handel entstehenden Bedarf an elementarer Rechenfertigkeit und dem von den öffentlichen Schulen kaum angebotenen Mathematikunterricht. Um die Bedeutung der Rechenmeister zu verstehen, muss man sich die Situation des Schulunterrichts dieser Zeit vergegenwärtigen:

Das Schulwesen wurde bis gegen Ende des Mittelalters hauptsächlich vom Klerus dominiert und zwar in Form von Kloster-, Dom-, Stifts- und Pfarrschulen. In den Schulen des Mittelalters ging es nicht allein um die klassische Wissensvermittlung, sondern auch um die Orientierung der Schüler hinsichtlich Lebenseinstellung und -führung oder die Vermittlung von Idealen. Die Lateinschulen bereiteten ihre Schüler auf einen geistlichen Beruf oder ein späteres Studium an einer Universität vor; der Lateinunterricht beanspruchte die meiste Zeit, praktisch anwendbare Mathematik oder kaufmännisches Rechnen wurden nicht gelehrt.

Ab dem 14. Jahrhundert setzte sich, ausgehend vom südwestdeutschen Sprachraum (insbesondere auch der Schweiz), die deutsche Schriftsprache als Verwaltungssprache in der urbanen Administration gegenüber dem Lateinischen immer mehr durch, zudem wuchs ein selbstbewusster neuer Stand, das Stadtbürgertum, heran. Stadtbürger übten zumeist kaufmännische oder handwerkliche Tätigkeiten aus, insbesondere für reisende Fernhändler waren Kenntnisse in Schreiben und Rechnen sehr wichtig. Auch ersetzt die deutsche Sprache nach und nach die lateinischen Einträge in den kaufmännischen Aufzeichnungen. Dadurch entstand ein Bedarf zur Einrichtung von Schreibschulen, den oft so genannten teutschen Schulen, wo man das Lesen und das Schreiben der deutschen Sprache lernte. Vor allem in den Städten übernahm der Rat die Verantwortung für Erziehung und Bildung, speziell zur Ausbildung des Kaufmannsstandes entstanden aber auch viele private Schulen; die folgende Abbildung zeigt ein Reklameschild eines Schulmeisters aus Basel von 1516:

Wer jemand hie der gern welt lernen dütsch schriben vnd lâsen vß dem aller
kürzisten grundt den jeman erdencken kan do durch ein jedr der vor nit ein
büchstaben kan der mag kürtzlich vnd bald begriffen ein grundt do durch er
mag von im selber lernen sin schuld vff schriben vnd lâsen vnd wer es
nit gelernen kan so vngeschickt were den will ich vm̄ nüt vnd ver-
geben gelert haben vnd gantz nüt von im zü lon nemen er sy-
wer er well burger ouch handtwerckß gesellen frowen vnd ju-
ncfrouwen wer sin bedarff der kum̄ har jn der wirt drüwlich
gelert vm̄ ein zimlichen lon Aber die jungen knaben vnd meit
lin noch den fronvasten wie gewonheyt ist Anno m cccc xvi

Wer jemand hie der gern welt lernnen dutsch schriben und laesen uß dem aller kürzisten grundt den jeman erdencken kan do durch ein jeder der vor nit ein buochstaben kan der mag kürtzlich und bald begriffen ein grundt do durch er mag von im selber lernnen sin schuld uff schriben und laesen. Und wer es nit gelernnen kann so ungeschickt were den will ich umm nut und vergeben gelert haben und gantz nut von im zuo lon nemen er syg wer er well, burger ouch handtwerckß gesellen frowen und junckfrouwen, wer sin bedarff, der kumm har jn, der wirt druwlich gelert umm ein zimmlichen lon. Aber die jungen knaben und meitlin noch den fronuasten wie gewonehyt ist. Anno M CCCCC XVI.

Wir versuchen eine Rohübertragung in heute verständlicheres Deutsch: „Wäre jemand hier, der aus denkbar kurzfristigstem Grund, der ihm einfällt, gerne deutsch schreiben und lesen lernen will, kann hiermit jeder, der bisher nicht einen Buchstaben kennt, schnell eine Grundlage begreifen, wodurch er selbst lernen kann, seine Schuld aufzuschreiben und zu lesen[1)]. Und wer es nicht lernen kann, weil er zu ungeschickt wäre, den würde ich für nichts und vergebens gelehrt haben und von ihm gar keinen Lohn nehmen. Er sei, wer er will. Bürger oder Handwerksgesellen, Frauen und Jungfrauen – wer dessen bedarf, der komme herein, er wird für einen geziemenden Lohn getreu belehrt. Die jungen Knaben und Mädchen aber wie gewöhnlich nach[2)] den Fronfasten[3)]. 1516.“

1) Gemeint ist, seine kaufmännische Buchführung durchzuführen. 2) Gemeint ist: „erst nach“
3) Gemeint ist der Fastensonntag.

Ein Schulmeister schilt vf beiden seiten gemolt

Dieses Reklameschild ist interessant: Bemalt wurde die zweiseitige 55 x 65 cm grosse Fichtenholztafel (wobei oben nur eine Seite wiedergegeben ist, die andere Seite zeigt eine Szene mit Schulkindern) von dem bedeutenden Renaissance-Maler Hans Holbein d.J. (1497–1543) im Alter von 18 Jahren, zusammen mit seinem Bruder Ambrosius. Gemeinsam zogen sie kurz zuvor, im Jahr 1515, von Augsburg nach Basel, in der Hoffnung, in der damals blühenden Buchdruckerstadt als Illustratoren ein gutes Einkommen zu finden. Dort nahmen sie Schreib- und Lateinunterricht bei dem Theologen, Schulmeister und späteren Reformator Oswald Myconius (eigentlich „Geisshüsler“, 1488 – 1555), und für ihn malten sie die Werbetafel.

Beim Schrifttyp handelt es sich um „Textura“, der seinerzeit vor allem bei den Bibeldrucken verwendet wurde und so Qualität symbolisiert. Bemerkenswert ist, dass im Text nicht nur Bürger, sondern auch Handwerksgesellen (ohne Bürgerrecht!) sowie neben verheirateten auch unverheiratete Frauen angesprochen werden. Fast modern erscheint auch, dass keine Vorkenntnisse erwartet werden, der Lehrerfolg mit einer „Geld zurück“-Garantie verbunden war und mit der kaufmännischen Buchführung der Praxisbezug hergestellt wird.

Im Kunstmuseum Basel trägt die oben dargestellte Seite der Reklametafel den Titel: „Schulmeister erklärt zwei des Lesens unkundigen Gesellen ein Schriftstück.“ Der Kunsthistoriker Prof. Bernd Wolfgang Lindemann beschrieb das Bild so: „Dem Maler gelingt es, die Personen sehr genau zu charakterisieren. Die Burschen in stutzerhafter Kleidung, mit einmal gestreifter, einmal geschlitzter Hose, mit weit geschnittenen Hemden und knappen Wämsen, scheinen in der Tat bis heute wenig Zeit und Mühe für die allergrundlegendsten Bildungsgüter geopfert zu haben. Schon das Stillsitzen fällt ihnen schwer, besonders jenem links, der, einem eingespannten Bogen gleich, mit noch ausgestreckten linken Bein sich so niedergelassen hat, als wolle er so bald wie möglich wieder davonspringen.“

Rechenmeister

Die teutschen Schulen waren näher an der Praxis orientiert als die klerikalen Schulen und waren daher vor allem für Händler und Handwerker und deren Knaben relevant. Methodisch wurde allerdings zumeist ein vielfach wiederholtes Memorieren der Unterrichtsinhalte praktiziert und Wissen unter Zuhilfenahme der Rute, dem Markenzeichen der Schulmeister, eingepaukt. Für mathematische Bildung über das Zahlenlesen und -schreiben sowie das kleine Einmaleins (oft nur bis 5x5 durch Aufsagen im Chor) hinaus war an den teutschen Schulen oft kein Platz, sodass sich spezielle Rechenschulen (bzw. Schreib- und Rechenschulen) etablierten, denn der Bedarf an der Kenntnis des Rechnens stieg mit der Entwicklung des Handels um 1500 drastisch an. Die Geldwirtschaft hatte den Tauschhandel abgelöst, die Kaufleute mussten jetzt Buch führen und rechnen. Die Umrechnung verschiedener Währungen (Gulden, Groschen, Pfennig, Heller, Teil, Schock, Dukaten, Kreuzer, Scherf, Taler, Batzen,..), Gewichtseinheiten (Zentner, Pfund, Lot, Quent, Teil, Scheffel, Mark,…) sowie Längen- und Raummasse (Elle, Tuch, Parchant, Fuß, Meile, Klafter; Schuh, Fuder, Eimer, Kanne, Kanndel, Maß,…) war jedoch umständlich, zumal diese oft auch regional unterschiedliche Bedeutung hatten. Auch Dreisatz- und Zinsprobleme stellten sich in der Praxis.

Die Rechenmeister schlossen nach und nach die Wissenslücke. Sie unterrichteten an ihren privaten Rechenschulen Arithmetik, Elementarmathematik und kaufmännisches Rechnen in der Landessprache und schufen Rechenbücher für Unterricht / Selbststudium. Die Ausbildung zum Rechenmeister erforderte 4 bis 6 Jahre Lehrzeit. Das Abschlussexamen umfasste u.a. Bruchrechnen, Dreisatz, arithmetische und geometrische Folgen sowie Algebra.

[Quelle der letzten fünf slides, auch für kurze Paraphrasen, u.a. de.wikipedia.org sowie „Schreibmeister und Schreibenlernen im späten Mittelalter / frühe Neuzeit“ von M-C. Kreidenitsch und „Deutsches Bürgerthum im Mittelalter“ von G.L. Kriegk]

A cifre tokeneþ nothinge – Null bedeutet nichts

In verschiedenen Ländern Europas wurde versucht, durch landessprachige Texte die zehnziffrige Stellenschreibweise zu popularisieren – was allerdings vor dem 16. Jahrhundert, also bevor der Buchdruck an Dynamik gewann, ein schwieriges Unterfangen darstellte. Das nachfolgend auszugweise wiedergegebene anonyme Manuskript aus dem 14. Jhd., auf einem einzigen Pergamentblatt niedergeschrieben, stellt ein frühes Zeugnis dieser Bemühungen dar. Betont wird die Rolle der Null („cifre") und die „arabische" Schreibweise von rechts nach links. Das frühe Englisch ist gut verständlich [*tokeneth* = *bedeutet*; vgl. auch im heutigen Englisch *to betoken = anzeigen, bezeichnen* sowie die Etymologie der Stammform „token": altsächsisch „tekan"; deutsch „zeigen", „Zeichen"; idg. *deyk-]:

To alle suche even nombrys the most have cifrys as to ten, twenty, thirtty, an hundred, an thousand and suche other. But ye schal understonde that a cifre tokeneth nothinge but he maketh other the more significatyf that comith after hym. Also ye schal understonde that in nombrys composyt and in alle other nombrys that ben of diverse figurys ye schal begynne in the ritht syde and so rekene backwarde and so he schal be wryte as thus – 1000. The cifre in the ritht syde was first wryte and yit he tokeneth nothinge no the secunde no the thridde but thei maken that figure of 1 the more signyficatyf that comith after hem by as moche as he born oute of his first place where he schuld yf he stode ther tokene but one. And there he stondith nowe in the ferthe place he tokeneth a thousand as by this rewle. In the first place he tokeneth but hymself. In the secunde place he tokeneth ten tymes hymself. In the thridde place he tokeneth an hundred tymes himself. In the ferthe he tokeneth a thousand tymes himself. In the fyftthe place he tokeneth ten thousand tymes himself. In the sexte place he tokeneth an hundred thousand tymes hymself. In the seveth place he tokeneth ten hundred thousand tymes hymself, &c.

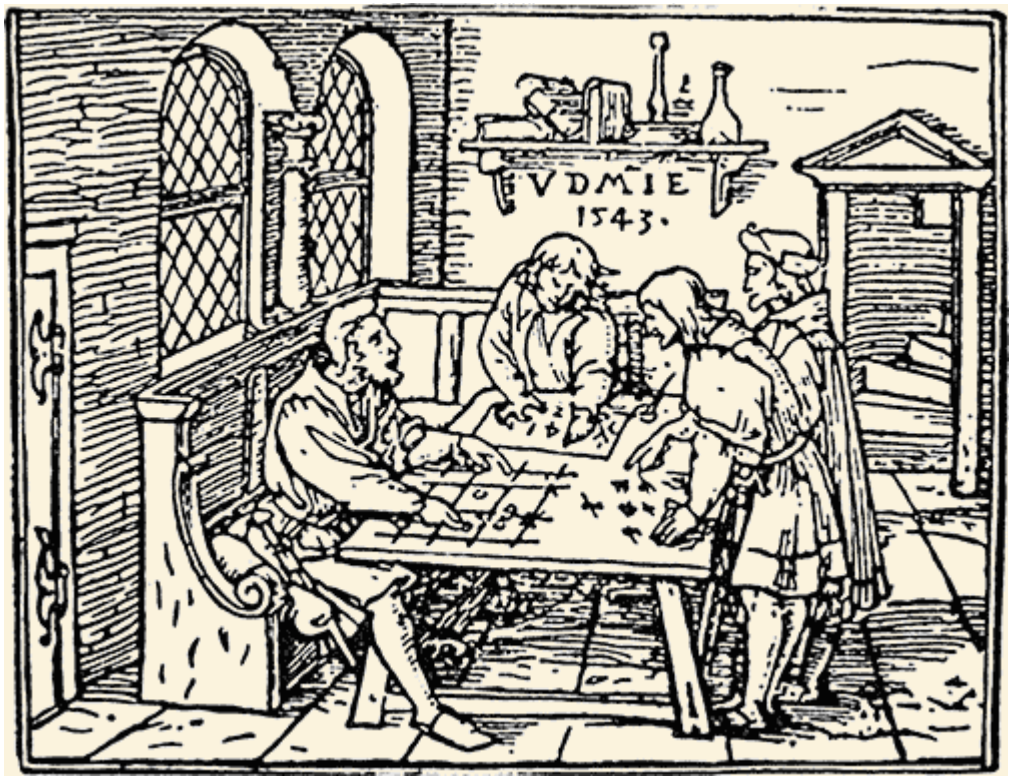
[James Orchard Halliwell-Phillipps (Ed.): Rara mathematica, 1839]

Rechnen „auf der Feder“

Im persisch-arabischen Raum schon früher üblich (vgl. al-Chwarizmi), setzten sich arabische Ziffern mit ihrer Stellenschreibweise in Europa nur langsam durch; es waren noch lange römische Zahlen sowie bei Kaufleuten Rechenbänke (mit auf Linien verschiebbaren „Rechenpfennigen“ oder aufgefädelten Steinchen in Form eines Abacus) gebräuchlich. Leonardo Fibonacci aus Pisa machte sich auf Orientreisen mit der arabischen Mathematik vertraut und verfasste dazu 1202 sein Rechenbuch „Liber abbaci“. Im seinerzeit maurischen Spanien seit dem 10. Jahrhundert gebräuchlich, lernten Rechenmeister nördlich der Alpen das arabische System erst im 15. Jahrhundert schätzen. Anfängliche Kritik am Rechnen „auf der Feder“ betraf den Papierverbrauch (Luxusgut!), leichtere Fälschbarkeit von Zahlen durch Anfügen von Ziffern sowie die heidnische Herkunft, was dem Teufel Zugang zu den Geschäften gewähre; insbesondere die Ziffer „0“ stiess auf grosse Akzeptanzprobleme. Noch 1299 verbot die Republik Venedig ihren Kaufleuten, nach diesem Verfahren zu rechnen, weil ihre Finanzbeamten der neuen Rechnungsart nicht mächtig waren.

TYPUS ARITHMETICAE, Bild von 1503: Rechts eine Rechenbank („Rechnen auf der Linie“) eines „Abakisten“, links ein mit arabischen Ziffern beschriebener Tisch eines „Algoristen“ („Rechnen auf der Feder“). Der zufriedene Algorist ist bereits fertig, während der Abakist noch missmutig rechnet. Schiedsrichterin ist die personifizierte und nicht ganz unparteiische Arithmetik, die auf ihrem Gewand Ziffern geometrischer Reihen trägt und dem Algoristen zugeneigt ist.

Rechnen „auf den Linien“

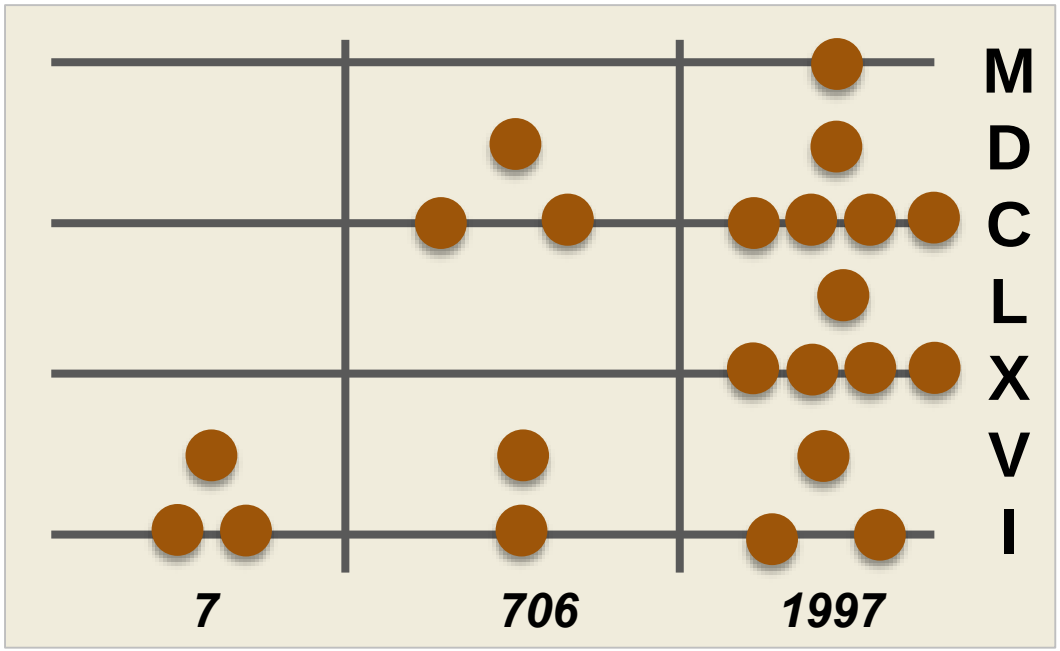


Der Wettstreit zwischen Abakisten und Algoristen in zeitgenössischer Darstellung
Aus: K. Menninger, Zahlwort und Ziffer, Bd. II.

Oskar Jursa beschrieb in einem Sachbuch der 1970er-Jahre („Kybernetik, die uns angeht“) die Vorgeschichte des Abakus in charmanter Weise so:

Die alten Griechen und vor ihnen schon die Ägypter bedienten sich einer denkbar unübersichtlichen Zahlenschreibweise. Entsprechend „handgreiflich“ waren denn auch ihre Rechenhilfen, zu denen sie ihre Zuflucht nahmen: Reiche Kaufleute besassen einen oder auch mehrere Sklaven, die mit ihren Fingern den jeweiligen Zahlenstand „festhielten“ und so einen Rechenvorgang nach dem anderen abwickelten. Weniger wohlhabende Händler mussten mit Steinchen vorlieb nehmen. Mit diesen bewegten sie ein Rechenbrett [...] Dieser sogenannte „Abakus“ wurde dann von den Römern zu einem etwa taschenbuchgrossen, handlichen Gerät weiterentwickelt.

Rechnen „auf den Linien“: Prinzip



Beispiele für Zahlendarstellungen mit „tokens“ (Rechenpfennigen), analog zum röm. Zahlensystem:

Die Zahl 7 (also VII) besteht aus einem Fünfer und zwei Einern.

706 (DCCVI) besteht aus 1 x 500, 2 x 100, 1 x 5 und 1 x 1.

Beachte: 1997 wird rein additiv als MDCCCCLXXXXVII, nicht verkürzt als MCMXCVII dargestellt; letztere Schreibweise bürgerte sich erst zu Beginn des 16. Jahrhunderts ein.

Neben dem Addieren gab es Rechenregeln für das Verdoppeln (Duplieren), Halbieren (Medieren) sowie Multiplizieren, Subtrahieren und Dividieren.

Das Addieren zweier Zahlen funktioniert so:

- Man legt in zwei Spalten des Brettes die beiden Zahlen, die addiert werden sollen.
- Dann werden die Rechenpfennige beider Zahlen zu einer einzigen, gemeinsamen Spalte zusammengeschoben.
- Danach werden die Pfennige nach evidenten Regeln neu angeordnet, z.B. von unten „eleviert“: Aus fünf Einern wird ein Fünfer, aus zwei Fünfern ein Zehner etc.
- Am Ende kann man als Ergebnis die Summe der ursprünglichen Zahlen ablesen.

Rechenpfennige

Parallele Verwendung von schriftlichem Rechnen („auf der Feder“) sowie dem Abakus mit Rechensteinchen bzw. Rechenpfennigen („auf der Linie“), 1533.

Rechenpfennige waren Münzen aus Kupfer oder Bronze, die zwar keinen Geldwert hatten, wohl aber reichhaltig mit Vignetten und Sprüchen versehen waren. Gerechnet wurde meist auf einem speziellen Tisch, der mit eingelegten Linien versehen war („Rechentisch“, engl. „reckoning table“, bzw. „Rechenbrett“ oder „Rechenbank“). Auf französisch wurden die Rechenpfennige „jetons“ genannt (von „jetter“, werfen, nämlich auf das Rechenbrett).

Rechenpfennige (2)

Die Verwendung von Rechenbrettern mit Linien, auf denen beim Rechenvorgang Steine oder spezielle Rechenpfennige verschoben wurden, wurde durch die Kreuzzüge im 13./14. Jh. nach Europa gebracht.

Der Landsknecht und Dichter Hans Wilhelm Kirchhof (1525 – 1605) verfasste 1562 seine Schwank-, Anekdoten- und Geschichtensammlung „Wendunmuth“, „darinnen fünff hundert und fünfftzig höflicher, züchtiger und lustiger historien, schimpffreden und gleichnüssen begriffen und gezogen seyn auß alten und ietzigen scribenten“. Über die Art, wie seinerzeit Rechenbretter und Rechenpfennige durch die flinken Rechenmeister gebraucht wurden, erfährt man etwas in einer der allegorischen Geschichten:

Das leben dieser zergengklichen welt und alle menschen darinn sein wie ein rechen- oder zalpfennig; auff welche linien dersel= bige gelegt, soviel und mehr gilt und zeigt er ein summ an. Ietzt ist er auff der obersten linien und bedeut ein, zwey oder zehen, biß= weilen hundert und drüber, tausend und noch mehr; bald nimpt in der, so in dahin gelegt, rückt in auff ein linien, darunder er allweg zehen mal soviel weniger gilt, als er

Der Kaufmann legt (!) Rechnung auf seiner Bank mit Rechenpfennigen; der Betrag von 3161 wird gerade zum Kummer des Kunden um 10 erhöht. (In Brügge: „Legghen en rekenen met penuingen“.)

Rechenpfennige (3)

auff der linien drüber golten hat. Ietzt ist er auff dem hundert, denn im spacio drun= der, ietzt auff dem zehen, denn auff dem ort, da er nit mehr denn eins, im hui nur ein halbs, ietzt ein gülden, ein album oder batzen, ietzt ein pfennig, heller etc. be= deut. Was darffs viel wort? Ehe sich ei= ner umbsicht, hebet der rechenmeister solchen pfenning gar hinweg, so ist er nichts mehr, denn ein ander pfennig, und ein stück messing.

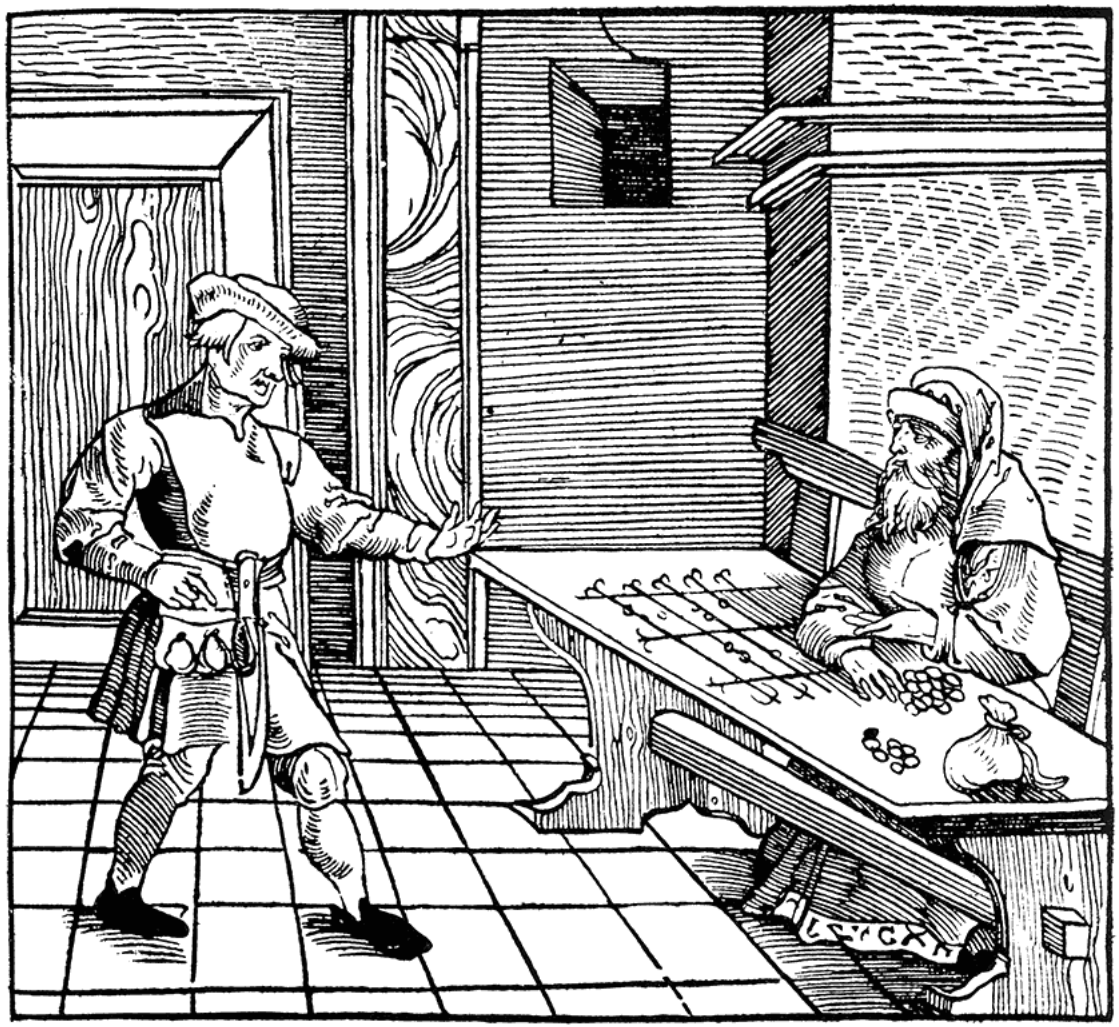
Gleich also handelt gott mit uns men= schen, und ist er der gewissest, kunst= reichst und gerechtigst rechenmeister; wir armen menschen seyn der zalpfen= ning. Denn wie das metall oder messig auß der erden kompt, also haben wir auch unsern Ursprung von der erden, unser aller mutter, und seyn derhalben einer so güt als der ander.

https://cdn.philasearch.com/A09454/E01391/0139100011.jpg

Bank und Abakus eines Geldleihers

Wenn du Geld verleihst an einen aus meinem Volk, an einen Armen neben dir, so sollst du an ihm nicht wie ein Wucherer handeln; du sollst keinerlei Zinsen von ihm nehmen [2. Mose 22, 24].

Da im Mittelalter Christen das Zinsnehmen und Pfandgeschäfte verboten waren, waren es die Juden, die als Geldleiher arbeiteten; manche von ihnen zählten später zu den wohlhabenden Bürgern, die im Bankwesen tätig waren. Aufgrund des Monopols und des oft schwer abzuschätzenden finanziellen Risikos waren die Zinsen meist hoch, weshalb die Geldgeber aus nichtjüdischer Perspektive oft als ‚Wucherjuden' verunglimpft wurden. Die katholische Kirche schaffte das Zinsverbot erst 1822 endgültig ab.



Ich bitt euch jud leicht mir zuo Hand/ Bargelt auff Bür= gen oder pfand/ Was euch gebürt gebt mit Verstand.

Bauer und jüdischer Geldleiher; Holzschnitt aus dem 1531 in Augsburg gedruckten Werk „Cicero, De officiis".

Rechentische,...

Rechentisch aus dem 16. Jahrhundert

Bildquelle: CACM Feb. 2017 (H. Bruderer) / Historisches Museum Basel
http://deliveryimages.acm.org/10.1145/2960000/2959085/figs/f1.jpg

...Rechenbretter und Rechenbänke

Engl. „counting board“ → „counter“, franz. „comptoir“ = Tisch, wo gezählt und gerechnet wird. Im 19. Jahrhundert im Englischen verallgemeinert hinsichtlich Ladentisch / -theke und später als „(kitchen) countertop“ für die (Küchen-)arbeitsplatte. Heute noch franz. „comptoir“ = „Zahltisch“ und „Handelskontor“.

Familie eines Händlers mit Wickelkind und Abakus. (Titelbild des Buches „Ein Regiment der jungen Kinder“ von Bartholomäus Metlinger, Augsburg, 1497)

Händler mit Gehilfe am Rechentisch; 17. Jhd.

Color. Holzschnitt 16. Jhd. (Herzog August Bibl.)

Das Monogramm von Virgil Solis: Ein V mit verflochtenem S im rechten Schenkel →

Das Rechenbrett symbolisiert hier einen „weisen Mann“, der als Ratgeber taugt. (Weitere Insignien stellen Federmesser, Federkiel, Schreibpult, Tintenfass und Brille dar.) Flugblatt mit einem Spruchgedicht von Hans Sachs („Ein nützlicher Rath den jungen Gsellen, so sich verheyraten wöllen“) und einem Holzschnitt von Virgil Solis; gedruckt 1549 in Nürnberg durch Hans Guldenmund.

Unter obigem Bild steht folgender Text. Er wurde von Hans Sachs (1494 – 1576), dem bekannten Dichter und Sänger aus Nürnberg (der auch die Hauptfigur in Richard Wagners Oper „Die Meistersinger von Nürnberg“ darstellt), verfasst. Hans Sachs erschuf hunderte solcher Spruchgedichte; dieses ist mit „**Rath zwischen dreyerley heyrat**“ überschrieben. Der Abakus taucht im Text nicht auf; der Illustrator verwendete ihn nur allegorisch als Symbol der Weisheit.

Nachdem ein jüngling frisch und frey
Het undter hand der heyrat drey,
Erstlich ein junckfraw schön und zart,
Nit fast reich, yedoch guter art,
Zum andern solt er im vertrawen
Zu der ehe ein junge witfrawen,
Die vor gehabt het eynen man,
Zum dritten solt er nemen an
Ein alte reich unnd wol begabt,
Die doch vor zwen mann het gehabt,
Nun ihr yede ihn haben wolt,
Nun west er nit, welche er solt
Nemen der dreyer, und thet gan
Zu einem alten weysen man
Und im die drey heyrat fürlegt.
Der weyse man seyn hand austreckt
Auff eyn fünffjering knaben mit,
Welcher auff eim steckleyn umriet
Inn der stuben, und sprach: Nun frag
Das kind, auff das es hie sag
Mit kurtzen wortten, welche frey
Auß den dreyen zu nemen sey.
Bald sprach der jünglieng zu dem knaben:
Sag, ob ich die junckfraw sol haben!
Das kneblein antwort: Wie du wildt.
Der jünglieng sprach: Sol ich die mildt
Wittfraw nemen, welche voran
Zu der ehe hat gehabt ein man?
Das kneblein sprach: Wie sie will.
Der jüngling sprach: Mir nit verhill,
Ob ich mir nemen soll die alten,
Welche auch vor hat haußgehalten
Mit zweyen mannen inn der ehe!
Rat mir, das ich mich nit vergehe!
Das kneblein warff sich bald herumb,
Rit ringweys inn der stuben umb
Unnd schry: Hüt dich! mein pferd schlecht dich.
Der weiß man sprach: O jüngling, sich!
Nun hast du deiner frag bescheyd.
Der jüngling sprach: Bey meinem ayd,
Ich hab verstanden gar kein wort
Von dem kneblein an diesem ort.
Ich bitt, wölst mir das baß erklern.
Der weiß man sprach: Von hertzen gern.
Kanst du denn erstlich nit verstan?
Da dir das kneblein zeyget an
Erstlich von der junckfrawen milt,
Da es zu dir sprach: Wie du wild,
Da maynt er, die junckfrawe gütig
Wer noch forchtsam, gschlacht und waichmütig,
Derhalb du ir wol möchst abziehen,
All eygensinnigkeit zu fliehen,
Das sie dir fein bleyb undterthan,
Das du im hauß bleybst herr und man
Und alles thet, wie du nur wolst.
Zu dem andern du mercken solst
Von der witfrawen inn der still,
Darzu der knab sagt: Wie sie will,
Meynt er, weil die wittib vorauß
Mit eym man het gehalten hauß,
Würds all ding thun nach ihrem sin,
Als die all ding wol wist vorhin
Und des haußhaltens het verstand,
Und würd ihr thun gar wee und and.
Wo du sie wolst ein anders leren,
Würd sich an dein straff nit viel keren.
Darob viel zancks sich würd erheben,
Ehe dus nach deym sinn richtest eben.
Als zu dem dritten ob der alten
Der knab das wort dir für hat ghalten:
Hüt dich, wann mein pferd das schlecht dich!
Darmit anzeygt er eygentlich,
Das es ein grosse thorheyt wer,
Das sich ein man geb inn solch gfer,
Nemb die, so vor zwen man het gehabt,
Obs gleich reich wer und wol begabt.
Bey den sie verbost und verargt,
Wer inn irm eygnen sinn verstarckt,
Das nyemand möcht biegen die frawen,
Denn allein schauffel und die hawen,
Wie man denn sagt von diesen sachen:
Alt hund böß bendig sind zu machen.
Verloren ist all trew und güt,
Zu endern ein verstockt gemüt.
Wolst dus denn bendigen mit zorn,
Mit rauffen, schlagen und rumorn,
So must du mit dem alten fratzen
Dein lebtag ziehen die strebkatzen
Oder der narr bleybn inn dem hauß.
Jüngling, nun wel dir selber auß,
Die erste, ander oder dritt,
Auff das dir inn der ehe darmit
Nit volg ein ewige nachrew,
Sonder dir durch ehliche trew
Fried, freud und freundligkeyt auffwachs
Im ehling stand! das wünscht
Hanns Sachs.

Abakus: Rechnen auf der Linie (im Sand)

The manipulation of pebbles in the dust, or the use of a finger or stylus in fine dust or sand spread upon a table, is known to have been used as an aid to calculation from very early times. The Semitic word abaq (dust) seems to be the root of our modern word abacus. From the Semitic, the word has been adopted by the Greeks who used αβαξ to denote a flat surface or table upon which to draw their calculating lines. The term then appears to have spread to the Romans who called their table an abacus. Because most early arithmetic was done on the abacus, the term became synonymous with "arithmetic" and we find such oddities as Leonardo of Pisa (Fibonacci) publishing a book in 1202 called "Liber Abaci" (The Book of the Abacus), which did not deal with the abacus at all but was designed to show how the new Hindu-Arabic numerals could be used for calculation. [Michael R. Williams]

Wirt und Gast rechnen ab: Drei Krüge Wein und mehr

Kurz vor dem Jahr 200 beschreibt der griechische Autor Athenaios in seinem Werk Δειπνοσοφισταὶ (*Gastmahl der Gelehrten*) die Verwendung eine Abakus für eine umfangreiche Rechnung, wo nicht nur addiert wird, sondern auch die unterschiedlichen Wertigkeiten der Münzen verrechnet werden müssen. Von einem Gast erbittet der Koch als Wirt eine Kostenbeteiligung an einem Gemeinschaftsmahl:

Gast: Solange du mir nicht für alles *en détail* Rechenschaft ablegst, kriegst du von mir nicht einen einzigen Heller!
Wirt: Logo. Ich hol' den Abakus und Rechensteine.
Gast: Dann leg' los!
Wirt: Der rohe Salzfisch macht fünf Groschen.
Gast: Weiter.
Wirt: Die Muscheln – sieben Groschen.
Gast: Ist gegessen. Weiter.
Wirt: Seeigel – einen Obolos.
Gast: OK.
Wirt: Dann kam wohl der Rettich.
Gast: Den du so angepriesen hast!
Wirt: Dafür habe ich zwei Oboloi gezahlt.
Gast: Kein Kommentar.
Wirt: Dann die Fischwürfel: Drei Oboloi.
Gast: Geschenkt! Aber vergasst Du die Endivien?
Wirt: Mein Bester, du kennst nicht den Markt; die Würmer haben alles Grünzeug ruiniert!
Gast: Berechnest du deswegen den Salzfisch zum doppelten Preis?
Wirt: Nein, das liegt am Händler; geh' und frag' ihn selbst. Der Aal macht zehn Oboloi.
Gast: Nicht zu viel. Und weiter.
Wirt: Den Bratfisch kaufte ich für eine Drachme.
Gast: Preise wie Fieber. Mal tief, nur um dann um so schneller wieder anzusteigen.
Wirt: Dazu der Wein. Als ihr schon trunken wart, besorgt' ich noch drei Krüge, jeder zu zehn Oboloi.

[Das verschlug dem Gast anscheinend die Sprache; das griech. Original findet sich hier: http://gallica.bnf.fr/ark:/12148/bpt6k251078/f312]

Abakus in einer holländischen Schule, ca. 1930

Anstatt Kalksteinchen („calculi" → „Kalkül") auf Linien im Sand zu verschieben, hat man jetzt Kügelchen in einem Gestell

Abakus in einer deutschen Schule, 1951

Abakus in einer chinesischen Schule

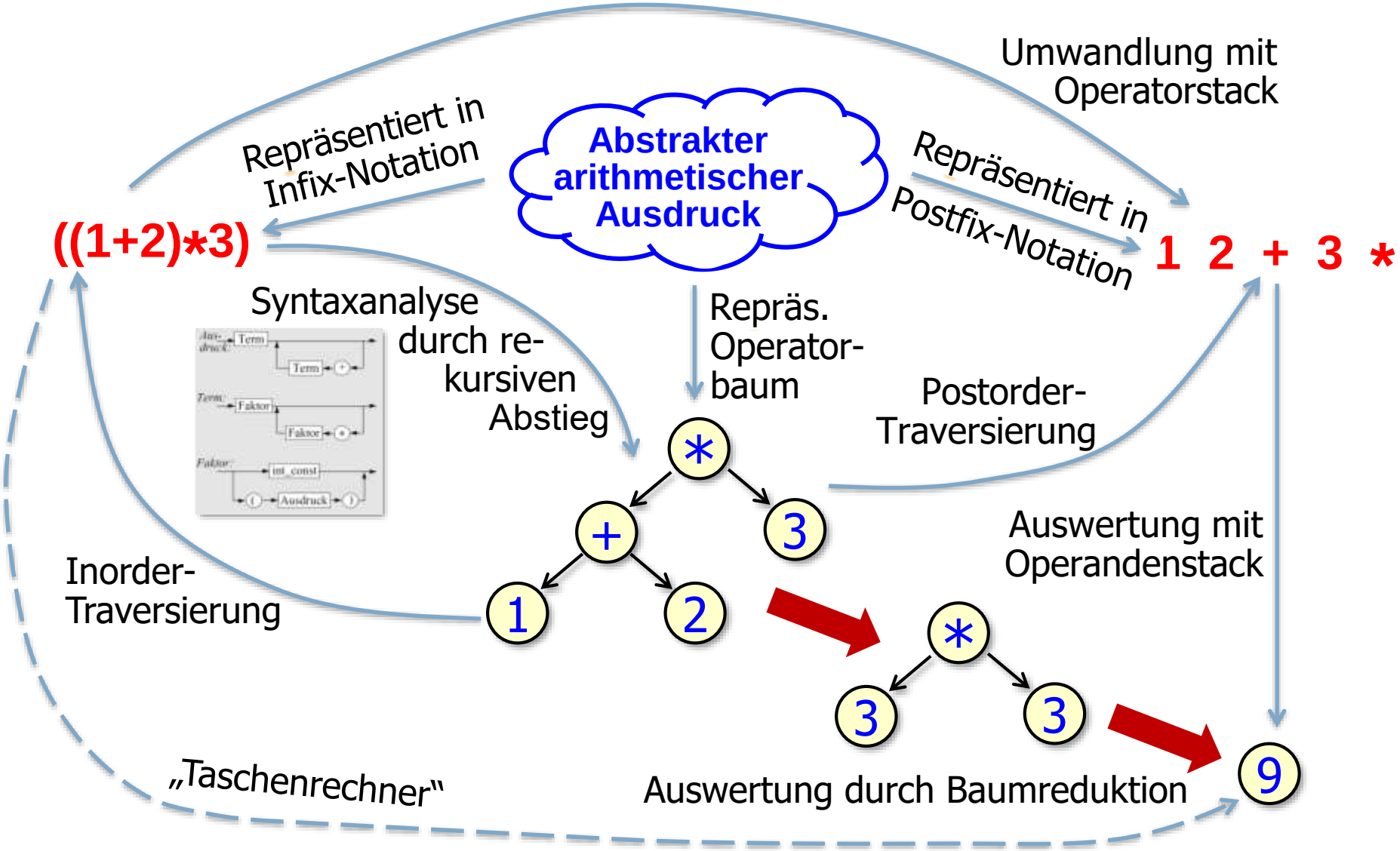
Abakus-Verkäufer in St. Petersburg, ca. 1860

Fotografie von William Carrick (1827 – 1878), National Galleries Scotland

William Carrick (Вильям Андреевич Каррик) was a Scottish-Russian artist and photographer. Carrick made a name for himself capturing pictures of Russian life and pioneering Russian ethnographic photography. [en.wikipedia.org]

www.museumsyndicate.com/images/4/33854.jpg

Überblicksgemälde zu arithmetischen Ausdrücken



Überblicksgemälde zu arithmetischen Ausdrücken

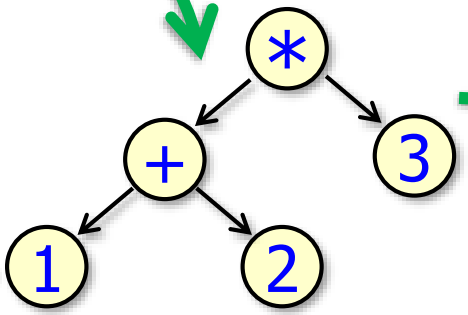
Als Übung: Algorithmus postfix → infix

Darstellungen systematisch ineinander umwandeln

((1+2)*3)

1 2 + 3 *

Transformation lässt Bedeutung invariant

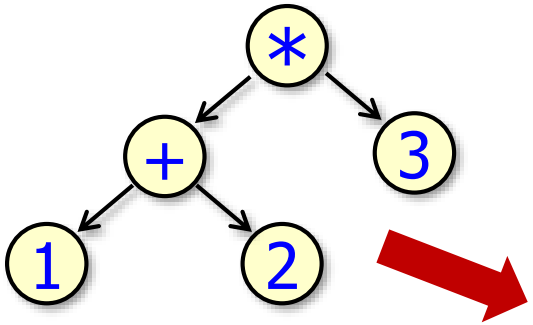


Überblicksgemälde zu arithmetischen Ausdrücken

Auswertung des Ausdrucks in diversen Repräsentationen

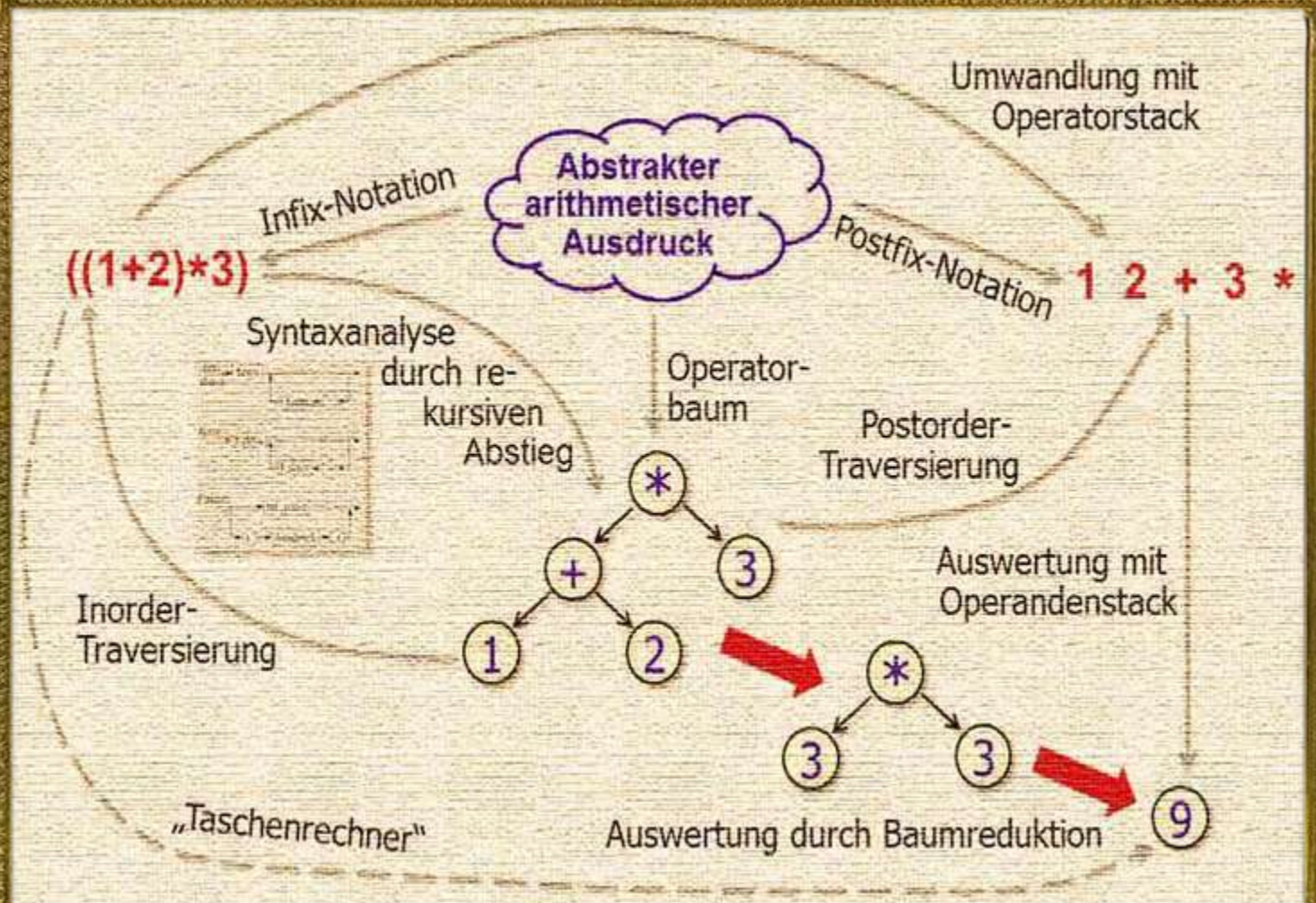
((1+2)*3)

Transformation lässt den Wert invariant



1 2 + 3 *

9


Umwandlung mit
Operatorstack
Infix-Notation
Abstrakter
arithmetischer
Ausdruck
Postfix-Notation
((1+2)*3)
1 2 + 3 *
Syntaxanalyse
durch re-
kursiven
Abstieg
Operator-
baum
Postorder-
Traversierung
*
+
3
1
2
Auswertung mit
Operandenstack
Inorder-
Traversierung
*
3
3
9
„Taschenrechner"
Auswertung durch Baumreduktion

Codegenerierung für Infix-Ausdrücke

- Aufgabe eines Compilers ist nicht nur die Syntaxprüfung, sondern auch „Code“ für eine Zielmaschine zu erzeugen

  Anweisungen in „codierter“ Form, die von einer Maschine ausgeführt werden können

- Die Codegenerierung für mathematische Formeln war Anfang der 1950er-Jahre, als es noch keine höheren Programmiersprachen und Compiler gab, ein echtes Problem
  - Dieses Problem wurde von Corrado Böhm in seiner (1954 veröffentlichten) Dissertation „Calculatrices digitales. Du déchiffrage de formules logico-mathématiques par la machine même dans la conception du programme“ an der ETH Zürich angegangen

CALCULATRICES DIGITALES

DU DÉCHIFFRAGE DE FORMULES LOGICO-MATHÉMATIQUES PAR LA MACHINE MÊME DANS LA CONCEPTION DU PROGRAMME

THÈSE

PRÉSENTÉE

À L'ÉCOLE POLYTECHNIQUE FÉDÉRALE, ZURICH,

POUR L'OBTENTION DU

GRADE DE DOCTEUR ÈS SCIENCES MATHÉMATIQUES

PAR

CORRADO BÖHM, ing. électr. dipl. EPUL
de Milan (Italie)

Rapporteur : Prof. Dr. E. STIEFEL
Co-rapporteur : Prof. Dr. P. BERNAYS

"Böhm's dissertation was especially remarkable because he not only described a complete compiler, he also defined that compiler in its own language!" (Donald E. Knuth)

Du déchiffrage de formules logico-mathématiques

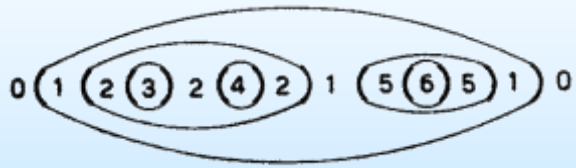
Einige Sätze aus der Dissertation

- Le support matériel sur lequel le programme a été enregistré sous forme de succession de formules est placé a l' entrée de la machine et la calculatrice exécute une série de calculs qui fournissent le même programme, mais enregistré sous forme d' instructions codifiées.
- Il s' agit d' un problème arithmétique susceptible d' être résolu par la calculatrice même, par l' application du programme de la «codification automatique».
- Soit une formule du type $(((a\ op_1\ b)\ op_2\ (c\ op_3\ d))\ op_4\ ((f^0\ op_5\ g)\ op_6\ h))$.

$f^o\ op_5\ g \rightarrow x_6$
$x_6\ op_6\ h \rightarrow x_5$
$c\ op_3\ d \rightarrow x_4$
$a\ op_1\ b \rightarrow x_3$
$x_3\ op_2\ x_4 \rightarrow x_2$
$x_2\ op_4\ x_5 \rightarrow x_1$
$x_1 \rightarrow x$

Il s' agit de construire une suite de nombres-instructions correspondant aux formules

…est en outre susceptible d'une représentation graphique



Corrado Böhm – CV
(1923 – 2017)

Aus der Dissertation

CORRADO BÖHM, citoyen italien, né à Milan le 17-1-23. En 1941 il obtint son diplôme d' études secondaires au Lycée scientifique «Vittorio Veneto» de Milan.

Entré en 1942 à l' Ecole Polytechnique de l' Université de Lausanne (Suisse) il en sortit fin 1946 avec le diplôme d' ingénieur électricien.

Il fut engagé en 1947 en tant qu' assistant à l' École Polytechnique Fédérale de Zurich de M. R. DUBS (Professeur d' Hydraulique et de Machines hydrauliques) pendant trois semestres et les trois suivants de M. E. STIEFEL (Professeur de Géométrie et Directeur de l' Institut de Mathématiques appliquées).

Pendant cette période, tout en approfondissant ses propres connaissances en mathématiques, il fréquenta un cours de spécialisation chez I. B. M. sur les machines à cartes perforées. Il se familiarisa ensuite avec les analogues machines BULL.

En 1949 il fut envoyé a Neukirchen (Allemagne) pour étudier sur place la machine à relais construite par M. ZUSE, machine qui fut adoptée ensuite par l' Ecole Polytechnique même.

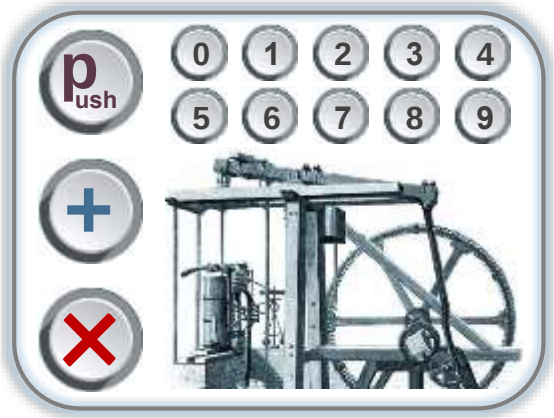
Corrado Böhm (1923 – 2017) emigrierte in die Schweiz, um den faschistischen Rassengesetzen zu entgehen, die sich gegen Juden richteten. 1951 kehrte er wieder nach Italien zurück, ging zunächst zu Olivetti, dann 1953 an das Istituto per le Applicazioni del Calcolo (IAC) des Consiglio Nazionale delle Ricerche in Rom, wo 1955 (u.a. von Dietrich Prinz) einer der ersten Computer Italiens (Mark 1 der Firma Ferranti aus Manchester) installiert wurde. In den 1960er Jahren lehrte er auch an den Universitäten Rom und Pisa und wandte sich der theoretischen Informatik (Lambda-Kalkül) zu. 1970 wurde er Professor für Informatik in Turin, 1974 Professor an der Universität „La Sapienza" in Rom. 1966 veröffentlichte er mit seinem Schüler Giuseppe Jacopini das sogen. Böhm-Jacopini-Theorem in einem Aufsatz "Flow diagrams, Turing machines and languages with only two formation rules" zur strukturierten Programmierung – es zeigt, dass man immer ohne expliziten Sprungbefehl ("goto") auskommt.

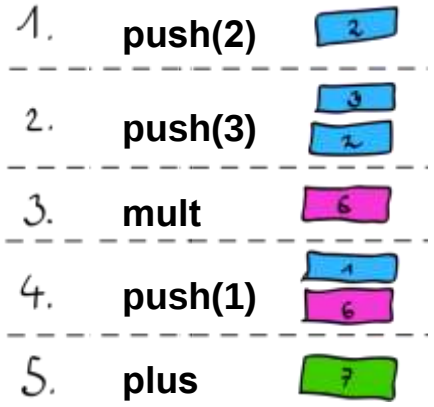
Codegenerierung für Infix-Ausdrücke

- Aufgabe eines Compilers ist nicht nur die Syntaxprüfung, sondern auch „Code“ für eine Maschine zu erzeugen

  Anweisungen in „codierter“ Form, die von einer Maschine ausgeführt werden können

- Als Zielmaschine für die Übersetzung von Infix-Ausdrücken postulieren wir hier eine Stackmaschine mit 3 Operationen:
  - **push(i)** – Int-Wert i in den Stack stopfen
  - **plus** – Obersten zwei Stackelemente durch ihre *Summe* ersetzen
  - **mult** – *… Produkt* …

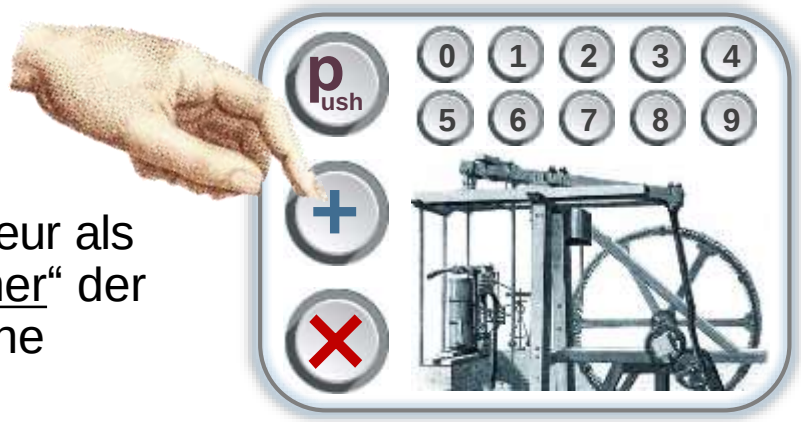




Programmgesteuertes Stackmaschinenbedienen

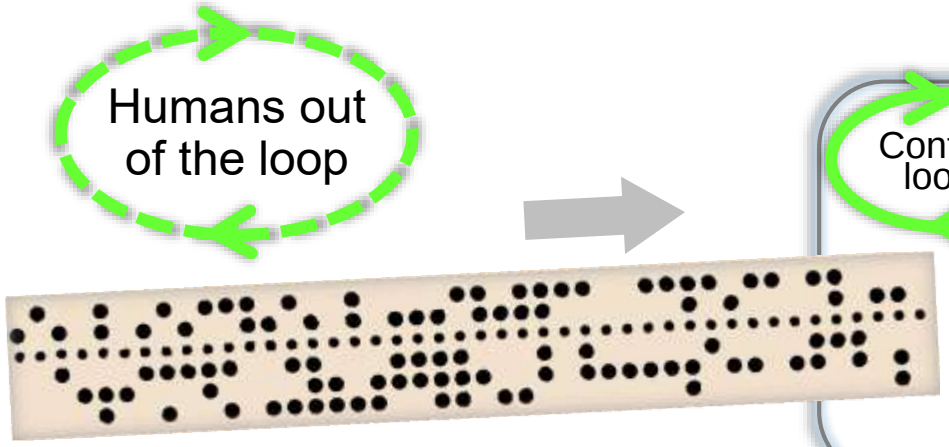
```
push(5)
push(1)
push(2)
plus
push(3)
push(4)
mult
mult
push(6)
plus
mult
```

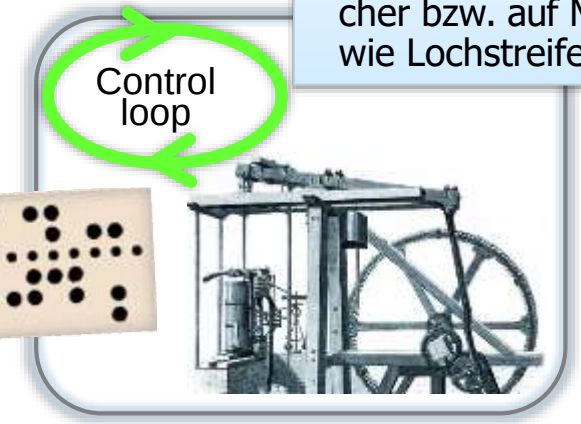
Operateur als „Bediener“ der Maschine



Programmgesteuerte Automaten

```
push(5)
push(1)
push(2)
plus
push(3)
push(4)
mult
mult
push(6)
plus
mult
```



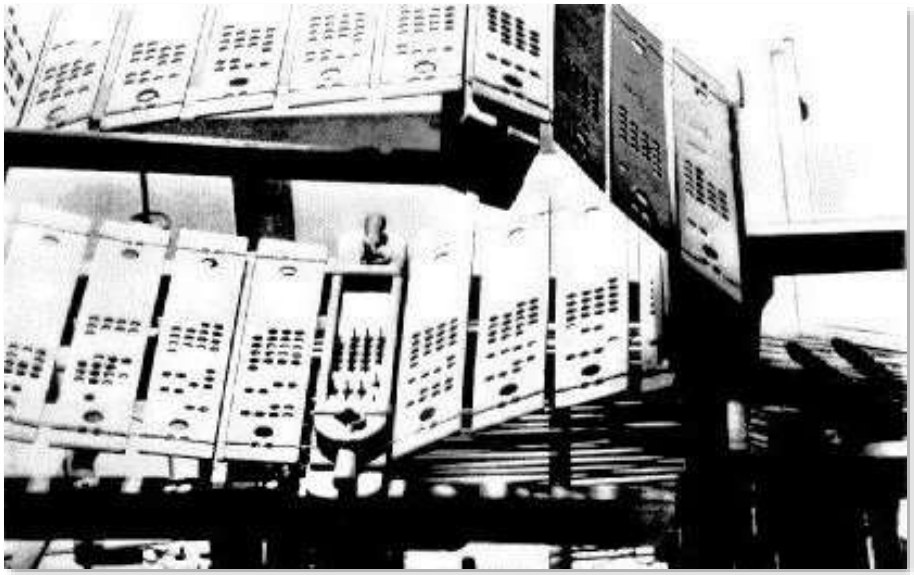


Nötig hierfür:
- Antriebsenergie
- Maschinenlesbares Programm (im Speicher bzw. auf Medium wie Lochstreifen etc.)

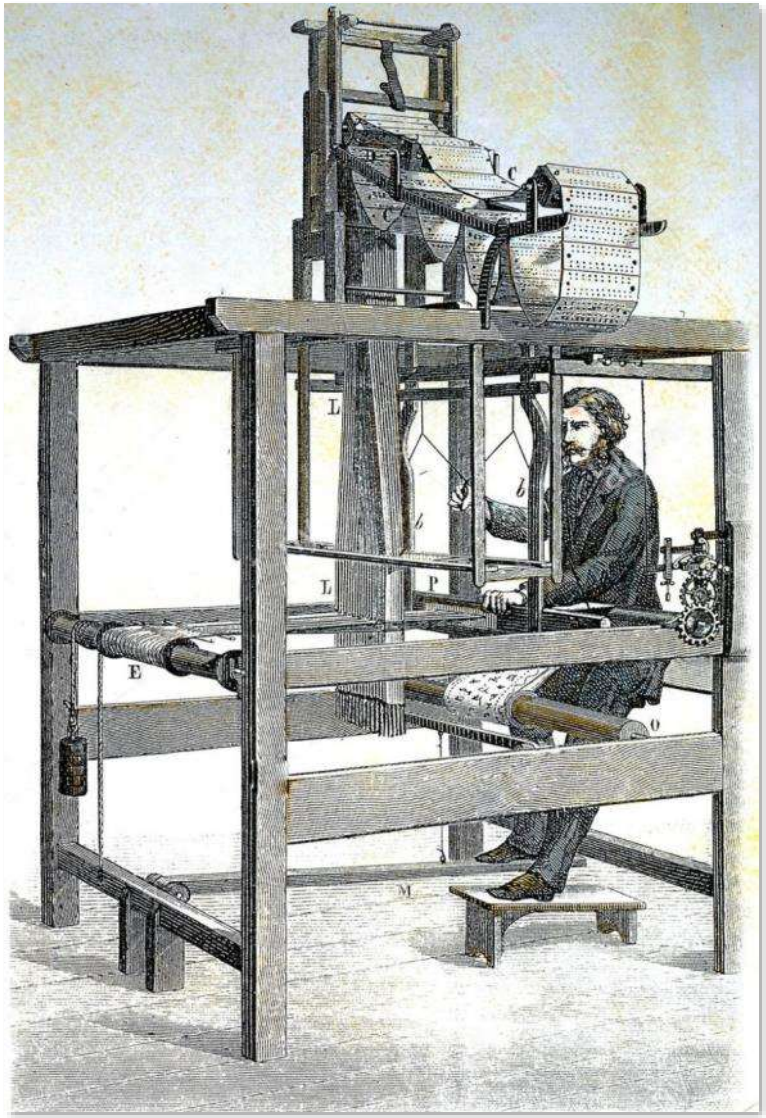
- 1801 **Jacquard**: Lochkarten-Ablaufsteuerung für mechanische Webstühle
- 1835 Charles **Babbage**: Konzept programmierbarer Rechenmaschinen
  - Mit kettenförmig aneinandergehängten Lochkarten
- Seit der Antike gibt es Automaten, die eine eingebaute Ablaufsteuerung besitzen (wie z.B. Spieluhren), allerdings *nicht frei programmierbar* sind
  - Vgl. dazu etwa https://de.wikipedia.org/wiki/Geschichte_der_Automaten

Programmgesteuerter Webstuhl

Joseph Marie Jacquard konstruiert 1801 in Lyon eine Ablaufsteuerung für mechanische Webstühle mit Lochkartenstreifen. Die Maschine stiess auf erbitterten Widerstand der Seidenweber, die um ihre Arbeit fürchteten; 1806 findet in Lyon auf Befehl des Zunftmeisters eine öffentliche „Hinrichtung“ eines Jacquard-Webstuhl statt, der zerschlagen und verbrannt wird, Jacquard selbst wird tätlich angegriffen. 1812 gab es bereits 11000 solche automatisierten Webstühle in Frankreich, er verbreitete sich dann rasch über England in andere Länder. C. Babbage liess sich bei seinen automatischen Rechenmaschinen durch Jacquards Erfindung inspirieren.



http://jaimeluna.angelfire.com/Images/telar_de_jacquard1.jpg



http://history-computer.com/Dreamers/images/jacquard-loom1.jpg

Stichwort „Automat“

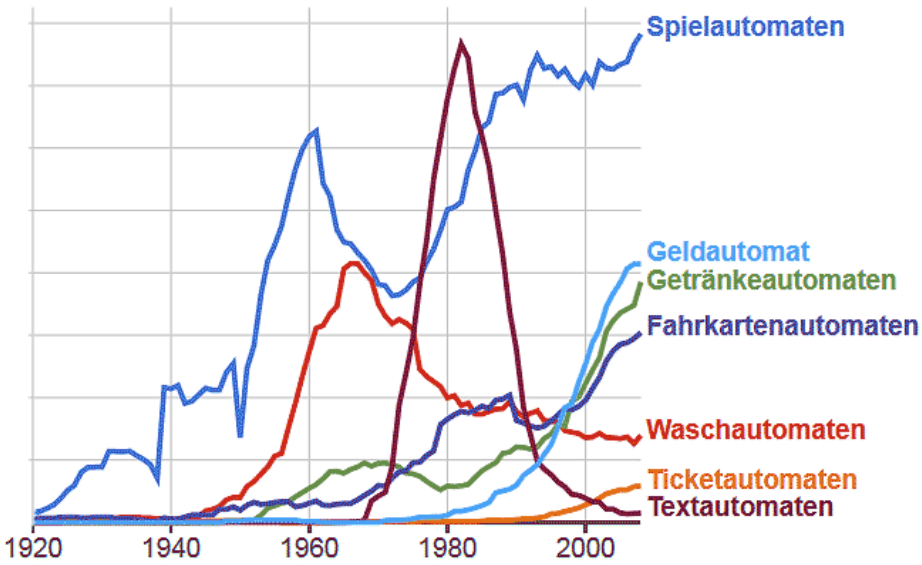
- Maschine, die vorbestimmte Abläufe selbsttätig („automatisch“) ausführt
  - Im engeren Sinne oft ein Apparat, der (z.B. nach Münzeinwurf) selbsttätig Waren abgibt oder eine Dienstleistung erbringt
- In der Informatik: eine abstrakte Maschine als Modell eines digitalen, zeitdiskreten Rechners (bzw. wesentlicher Komponenten davon)
  - Relevant vor allem in den Teilgebieten Digitaltechnik, Berechenbarkeits-theorie, Komplexitätstheorie und Automatentheorie
- Das Fremdwort „Automat“ erscheint zuerst im 16. Jh. in den Formen *Automata* und *Automaton*; die eingedeutschte Form setzt sich dann im 18. Jh. unter dem Einfluss des französischen Wortes „automate“ durch
  - Etymologie: Aus dem Griechischen αὐτόματος (automatos), „von selbst geschehend“, oder „aus sich selber denkend und handelnd“, mit den beiden Wortbestandteilen *auto* und *matos*.
  - *Aut-* bzw. *auto-* von *autos* = „selbst“; vgl. Automobil, Automorphismus, Autodidakt, Autofokus, Autist, autonom, autark etc.
  - *Matos:* altgriechisches Partizip aus der indogerm. Sprachwurzel „*men“ für die Wortsippe um „denken, wollen, (geistig) erregt sein, Antrieb“ etc.; daraus z.B. auch *Manie* und *munter* (im Sinne von „aufgeregt, lebhaft“)

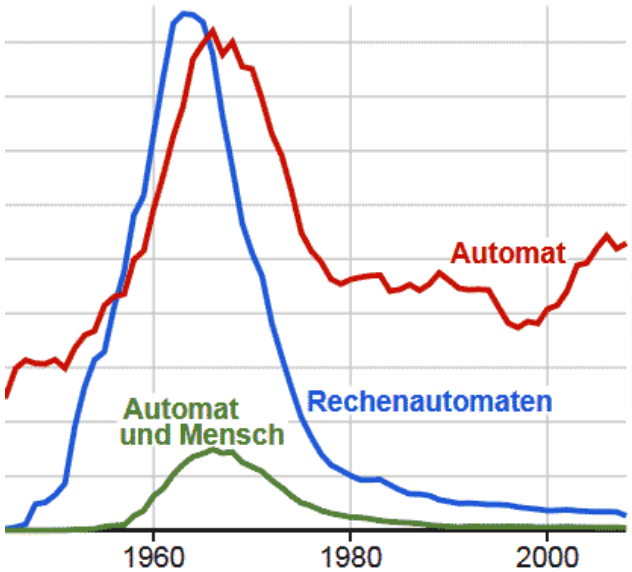
Stichwort „Automat“ (2)

Automat (vom griech. αὐτόματος, aus eigenem Antriebe handelnd), bedeutet 1) überhaupt jede sich selbst bewegende mechanische Vorrichtung, die eine Zeit lang ohne Einwirkung von außen, durch die im Innern verborgenen Kräfte (Federn, Gewichte u. s. w.) in Bewegung gesetzt wird; z. B. Uhren, Planetarien (s. d.) und dergl. künstliche Räderwerke, weshalb auch die Uhrmacherkunst Automatopoetica genannt wird. Die Schachmaschine ist daher kein A., da immer menschliche Einwirkungen von außen, wenn auch noch so fein versteckt, dabei nöthig sind. 2) Im engern Sinne versteht man aber unter A. ein mechanisches Kunstwerk, welches vermittelst eines innern Mechanismus die Thätigkeit lebender Wesen, der Menschen oder Thiere, nachahmt und meist auch an Gestalt diesen nachgebildet ist.

Meyers Konversations-Lexikon („Das große Conversations-Lexicon für die gebildeten Stände. In Verbindung mit Staatsmännern, Gelehrten, Künstlern und Technikern herausgegeben von J. Meyer.“), 4. Band, 2. Abt., 1844

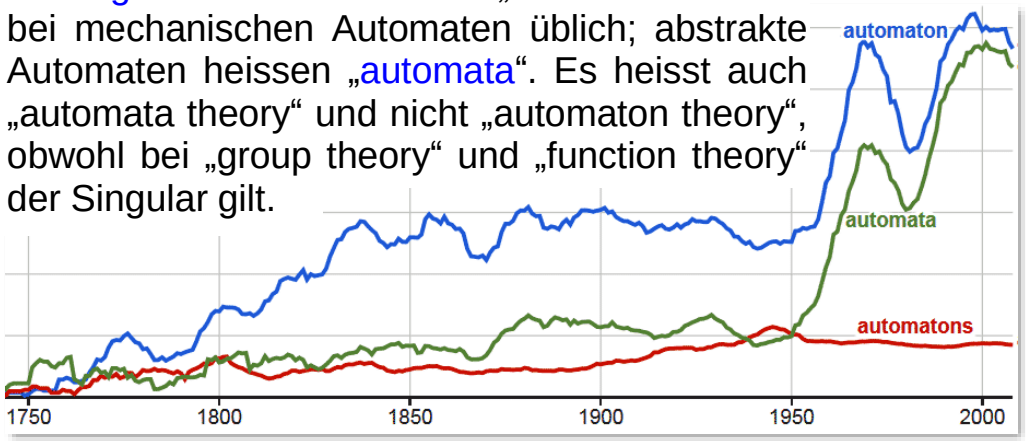
Automatenzeitalter?





Quelle jeweils: Google Books

Im Englischen ist der Plural „automatons“ eher bei mechanischen Automaten üblich; abstrakte Automaten heissen „automata“. Es heisst auch „automata theory“ und nicht „automaton theory“, obwohl bei „group theory“ und „function theory“ der Singular gilt.



Geld- und Spielautomaten sind im Aufstieg, den Textautomaten (vgl. folgende Seite) und den Rechenautomaten war als Begriff nur ein kurzer Höhepunkt gegönnt. Spielmaterial für eigene Experimente: Fotoautomaten, Verkaufsautomaten (engl.: „vending machines“, franz.: „distributeurs“), Kaffeeautomaten, Kellerautomaten, Bankautomaten, Lungenautomat, Drehautomat, endlicher Automat (engl.: finite state machine), Vollautomat, Halbautomat, Automatenrestaurant,...

Textautomaten

Textautomaten (bzw. Schreibautomaten) waren speziell für die Textverarbeitung konzipierte Computer, die das Verfassen von Schriftstücken im Geschäftsalltag rationalisierten. Stereotype Formulierungen und Texte wurden einmalig eingegeben und unter einer Codenummer abgespeichert. Textbausteine konnten kombiniert und individuell ergänzt werden. Die Ära der Textautomaten begann Mitte der 1960er Jahre mit den Speicherschreibmaschinen, im Laufe der 1970er Jahre kamen die Bildschirmausgabe sowie Disketten als Speichermedium hinzu. Die Ära endete relativ abrupt mit Erscheinen des PCs in den 1980ern.

http://museum.ipsj.or.jp/computer/word/images/0030_01_l.jpg

www.imamuseum.org/blog/wp-content/uploads/2014/02/ibm_selectric_action.jpg

www.theparisreview.org/blog/2016/05/11/this-faithful-machine/

1964 begann IBM mit der Auslieferung seines „Magnetic Tape Selectric Typewriter“, der MT/ST, eine Kugelkopfmaschine mit einem Beistelltisch, auf dem eine wuchtige Mechanik zwei Bandlaufwerke steuerte, die mit Speicherkassetten bestückt wurden. Damit kam die Textverarbeitung in heutiger Form in die Welt. Denn mit der MT/ST konnten Absätze neu geschrieben und umkopiert oder Textblöcke verschoben werden. Als wichtigstes Feature entpuppte sich der „Serienbrief“. Die Magnetbänder konnten jeweils 28000 Zeichen speichern, entsprechend rund 12 A4-Seiten. Einmal auf die Bänder gespeicherte Texte konnten mit 900 Anschlägen pro Minute vom rasenden Kugelkopf verarbeitet werden. 42719 Mark kostete das Wunderwerk. IBMs grösster Kunde in Deutschland wurde die Allianz, wo Sachbearbeiter fortan keine Texte mehr diktierten, sondern in einem umfangreichen Floskel-Handbuch die entsprechenden Nummern zusammensuchten und nur die Nummern der Textbausteine nannten. „Sterbeband 14 23 56“ ergab ein Schreiben, das den Angehörigen das Beileid aussprach, die zügige Abwicklung der Versicherungsleistung versprach und einige Dokumente anforderte.

Textquelle (gekürzt):
www.heise.de/ct/artikel/Der-Mensch-denkt-die-Maschine-arbeitet-302172.html

http://creativepro.com/scanning-around-gene-back-when-typesetting-was-craft/#

Spätere Systeme nutzten anstelle von Magnetband andere Speichertechniken wie Disketten (oben: 8-Zoll-Floppy) oder Magnetkarten (links, 1972).

https://digitaltmuseum.no/011015239434/7-0-ibm-op-fotografier

Stackmaschinen

- Stackmaschinen sind meistens in Software realisiert und werden deshalb auch virtuelle Maschinen (VM) genannt. Klassischerweise wurden sie entworfen, um Implementierungen von Programmiersprachen (wie z.B. Pascal) leichter zu portieren.
- Sie bieten Befehle zur Auswertung von Ausdrücken an, welche oben auf einem Operanden-stack („Auswertungskeller") stattfindet. Operanden von arithmetischen und logischen Befehlen sowie Vergleichen stehen in den obersten Zellen des Operandenstacks oder sind Direktoperanden, also Konstanten. Deshalb enthalten die meisten dieser Befehle keine Speicheradressen. Programme für Stackmaschinen sind daher sehr kompakt, was günstig für den Speicherbedarf und die notwendige Bandbreite beim Versenden über ein Kommu-nikationsnetz und Herunterladen solcher Programme ist.
- Der Speicher ist in einen Laufzeitstack und in eine Halde („Heap") aufgeteilt. Stackmaschi-nen bieten daher auch Befehle zur Organisation von Laufzeitstack und Heap an.
- Eine Implementierung (oder „Emulation") der virtuellen Maschine auf einem „echten" Hardwareprozessor ist relativ einfach und effizient; es werden nur wenige Register benötigt.
- Ein prominentes Beispiel einer Stackmaschine ist die virtuelle Java-Maschine (JVM). Sie besitzt neben dem Operandenstack einen Laufzeitstack aus „frames" (Kellerrahmen); dieser enthält zu jedem Zeitpunkt der Programmausführung einen solchen frame für jede Methode, deren Ausführung begonnen, aber noch nicht abgeschlossen wurde. In diesem Rahmen liegen Werte der Funktionsparameter, Werte lokaler Variablen und Referenzen in den Heap hinein auf dort abgelegte Arrays und Objektinstanzen.

Codegenerierung: Stackmaschine als Zielsystem

- Wir wollen unser Analyseprogramm an den „richtigen“ Stellen so mit Anweisungen zur Codeerzeugung ausstatten, dass Folgendes generiert wird:

Programm in „Maschinensprache“

`1+2*3`	`(1+2)*3`
`push(1)`	`push(1)`
`push(2)`	`push(2)`
`push(3)`	`plus`
`mult`	`push(3)`
`plus`	`mult`

Die Zielmaschine kann in Hardware realisiert sein, oder als virtuelle Maschine durch ein Laufzeit- oder Betriebssystem simuliert werden.

Oder der generierte Code wird weiter in Code für eine echte Zielmaschine transformiert.

`5*(((1+2)*(3*4))+6)`

`push(5)`	5			
`push(1)`	5	1		
`push(2)`	5	1	2	
`plus`	5	3		
`push(3)`	5	3	3	
`push(4)`	5	3	3	4
`mult`	5	3	12	
`mult`	5	36		
`push(6)`	5	36	6	
`plus`	5	42		
`mult`	210			

Stackmaschine wendet die Operation (+, *) an, nachdem die Operanden in den Stack gebracht wurden → postfix!

`5 1 2 + 3 4 * * 6 + *`

Das ist eigentlich eine Übersetzung nach postfix!

Der Parser mit Codeerzeugung

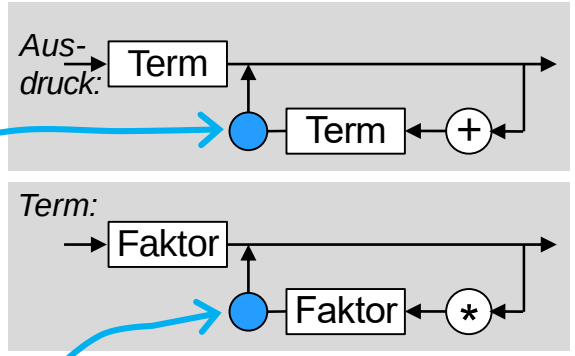
```
void int_const(){ // hier nur einziffrig
  System.out.println("push(" + c + ")");
  c = KbdInput.getc();
}
```

```
void Ausdruck(){
  Term();
  while (c == '+') {
    c = KbdInput.getc();
    Term();
    System.out.println("plus");
  }
}
```

```
void Term() {
  Faktor();
  while (c == '*') {
    c = KbdInput.getc();
    Faktor();
    System.out.println("mult");
  }
}
```

Codegenerierung für die Stackmaschine (als Postfix-Operation!)

Codegenerierung an der korrekten Stelle im Diagramm bzw. der zugehörigen Methode – z.B., nachdem ein zweiter (bzw. dritter etc.) Term analysiert wurde:



Auch bei dieser Infix zu Postfix-Übersetzung wird (implizit) ein Stack benutzt:

Der Laufzeitstack von Java, in dem die Rücksprungadressen abgelegt sind!

Ein Interpreter für Infix-Ausdrücke

5*(((1+2)*(3*4))+6)

- Statt die Operationen einer Stackmaschine auszugeben (also z.B. in eine Datei zu schreiben und sie anschliessend von einer Stackmaschine ausführen zu lassen), können wir auch gleich push, plus, mult etc. auf einem Stack ausführen und so den Ausdruck „on the fly" schritthaltend zur Analyse auswerten: `stk.push(stk.pop() * stk.pop());`
  - Wir simulieren quasi nebenbei schrittweise die Zielmaschine

```
push(5)
push(1)
push(2)
plus
push(3)
push(4)
mult
mult
push(6)
plus
mult
```

- Wir bekommen damit statt eines Compilers (=Übersetzer) einen Interpreter für Infix-Ausdrücke

- Zur Realisierung dieses „Taschenrechners" verwenden wir wieder unsere Service-Klasse „Stack" →

In der Vorlesung „Informatik I" wurde die Logik des Taschenrechners nicht mittels explizitem Stack realisiert, sondern es wurde eine rekursive Funktion verwendet: *double expression (double v, char sign , std :: istream& is)*

Ein Interpreter für Infix-Ausdrücke (2)

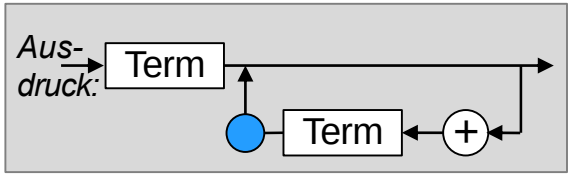
```
class Stack // wie gehabt ein int-Stack
class Parser {
  Stack stk = new Stack(1000); char c;

  void int_const() {  // einstellig
    stk.push(Character.digit(c,10));
    c.KbdInput.getc();
  }

  void Ausdruck() {
    Term();
    while (c == '+') {
      c = KbdInput.getc(); Term();
      stk.push(stk.pop() + stk.pop());
    }
  }

  void Term() {
    Faktor();
    while (c == '*') {
      c = KbdInput.getc(); Faktor();
      stk.push(stk.pop() * stk.pop());
    }
  }

  ... main ...
    Parser p = new Parser();
    p.c = KbdInput.getc();
    p.Ausdruck();
    System.out.println(p.stk.pop());
}
```

Anstatt System.out.println("plus");

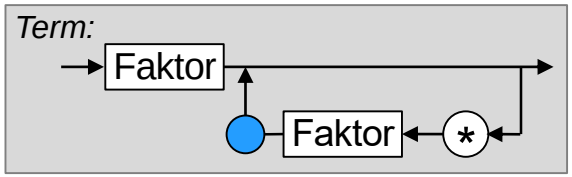
Was ist mit der Methode „Faktor()"?

Damit haben wir nun einen Taschenrechner für geklammerte Ausdrücke realisiert!

Umwandlung eines char in eine Zahl zur Basis 10



Die obersten beiden Stackelemente werden durch deren Summe ersetzt



Hier wird das Resultat ausgegeben

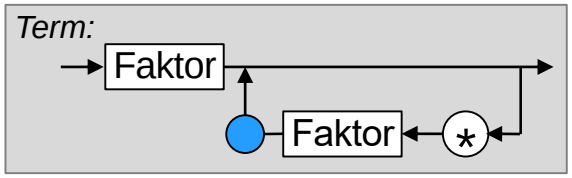
Déjà-vu?

Hier nochmals zum Vergleich der entsprechende Teil des Ansatzes aus „Informatik I“

Terme auswerten

```
double term (std::istream& is)
{
  double value = factor (is);
  while(true){
    if (consume(is, '*'))
      value *= factor (is);
    else if (consume(is, '/'))
      value /= factor(is)
    else
      return value;
  }
}
```

term = factor { "*" factor | "/" factor }




```
// Term = Faktor |
// Faktor "*" Faktor
void Term() {
  Faktor();
  while (c == '*') {
    c = KbdInput.getc();
    Faktor();
    stk.push(stk.pop()
      * stk.pop());
  }
}
```

BUNDESREPUBLIK DEUTSCHLAND

KL. 42 m 14

INTERNAT. KL. G 06 f

DEUTSCHES PATENTAMT

AUSLEGESCHRIFT 1 094 019

B 44122 IX/42m

ANMELDETAG: 30. MÄRZ 1957

BEKANNTMACHUNG DER ANMELDUNG UND AUSGABE DER AUSLEGESCHRIFT: 1. DEZEMBER 1960

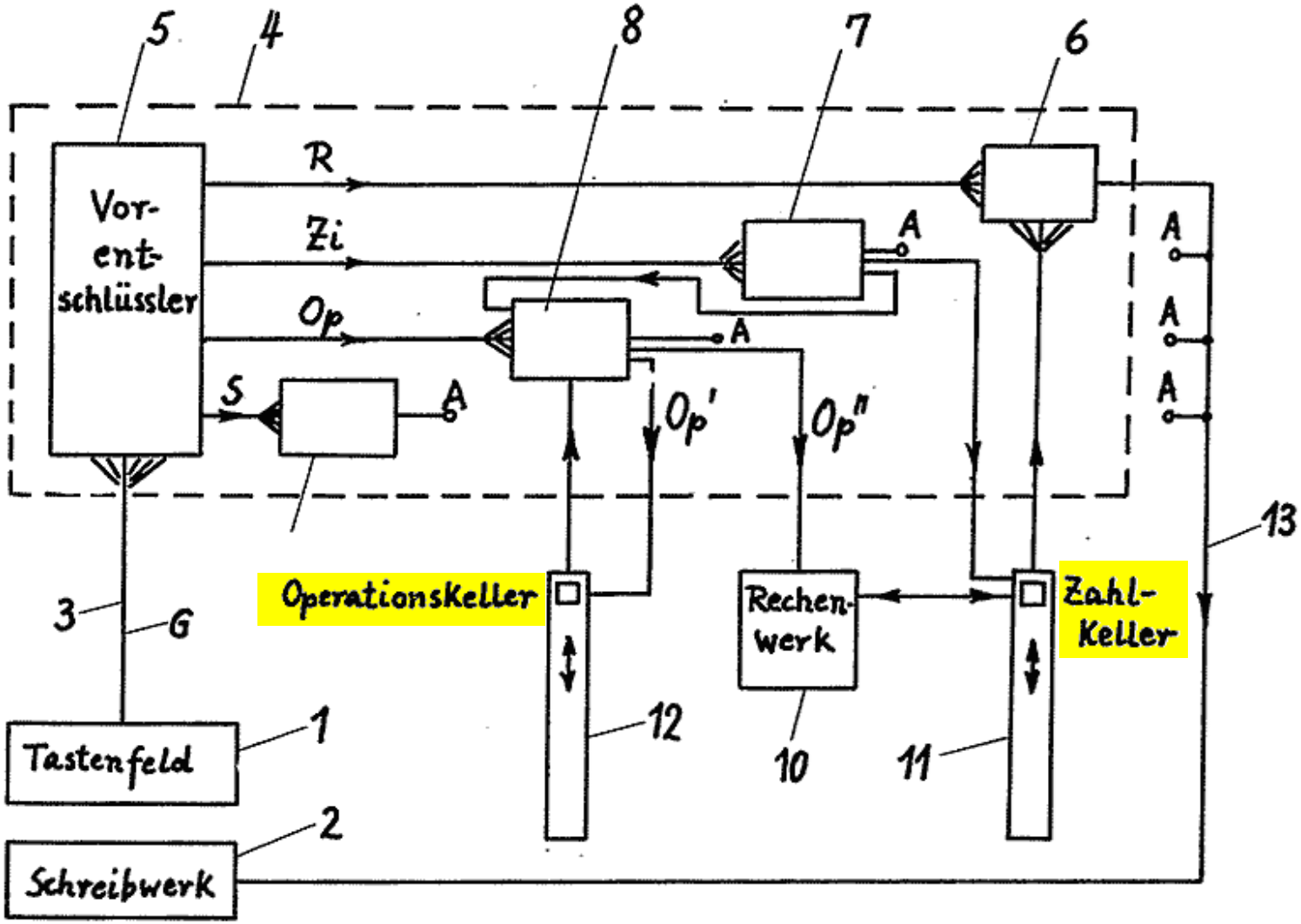
Verfahren zur automatischen Verarbeitung von kodierten Daten und Rechenmaschine zur Ausübung des Verfahrens

Anmelder:

Dr. Friedrich Ludwig Bauer,
München, Pörtschacherstr. 40,
und Dr. Klaus Samelson,
München, Hiltenspergerstr. 19

Historische Notiz

Ein etwas längerer geschichtlicher Einschub zu F. L. Bauer, Rutishauser, Speiser, Zuse, den Anfängen von Compilern, der Z4, der ERMETH und Babbages Analytical Engine sowie dem ersten Programm.



Bem.: Die klassische deutsche Bezeichnung für „Stack“ lautet „Keller“ oder „Stapel“ (vgl. dazu das „Einkellern“ als Tätigkeit)

PATENTANSPRÜCHE:

1. Verfahren zur automatischen Verarbeitung von kodierten Daten, z. B. arithmetischen Formeln in üblicher Schreibweise, die als kodierte Zeichen Klammern, Operationssymbole, Zahlen und Variable gemischt, Dezimalkommas, Indizes, Entscheidungssymbole sowie Formelnummern enthalten, in einer datenverarbeitenden Maschine mit einer Eingabe- und einer Ausgabevorrichtung, **dadurch gekennzeichnet,** daß die den einzelnen Zeichen entsprechenden Signale in der Reihenfolge der Aufschreibung einem Analysator (**4**, **28**) zugeführt und in diesem entsprechend der Reihenfolge des Eingangs geprüft werden, ob die Operationen sofort ausführbar sind oder ob der Eingang weiterer Signale abgewartet werden muß, daß in diesem letzteren Fall die noch nicht verarbeitbaren Zeichen in einen Speicher (Keller) eingeführt werden und daß beim Eintreffen neuer Zeichen im Analysator (**4**, **28**), die die Ausführung einer Operation mit gespeicherten Zeichen ermöglichen, diese gespeicherten Zeichen in der durch die Art der Einführung festgelegten umgekehrten Reihenfolge entnommen und verarbeitet werden.

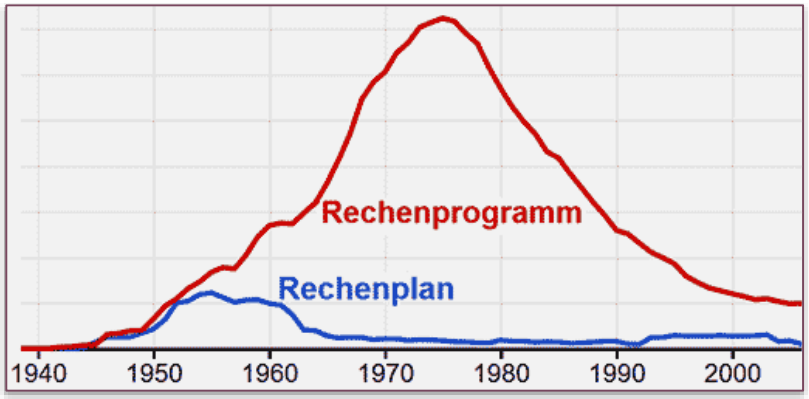
2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, daß im Analysator die Formelzeichen nach Ziffernsymbolen (Zahlen) und Operationssymbolen getrennt sind und, sofern sie zurückgestellt werden müssen, zwei verschiedenen speicherfähigen Vorrichtungen (**11**, **12**), vorzugsweise zwei verschiedenen »Kellern«, nämlich dem Zahlkeller (**11**) und dem Operationskeller (**12**), zugeführt werden.

3. Verfahren nach Ansprüchen 1 und 2, dadurch gekennzeichnet, daß die in dem Zahlkeller (**11**) bzw. dem Operationskeller (**12**) neu eintreffenden Zeichen sich jeweils an die Spitze der entsprechenden Sequenz setzen und die Entnahme eines Zeichens automatisch durch Wegnahme von der Spitze der entsprechenden Sequenz erfolgt.

Wie kam es zu diesem Patent? Friedrich L. Bauer erinnert sich:

„Im Jahr 1951 erschien eine bahnbrechende, aufregende Zürcher Publikation von Heinz Rutishauser – ein Programm, das Programme produziert:“

- *Über automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen.* Z. Angew. Math. u. Mech. 31(8/9):255, Aug./Sept. 1951



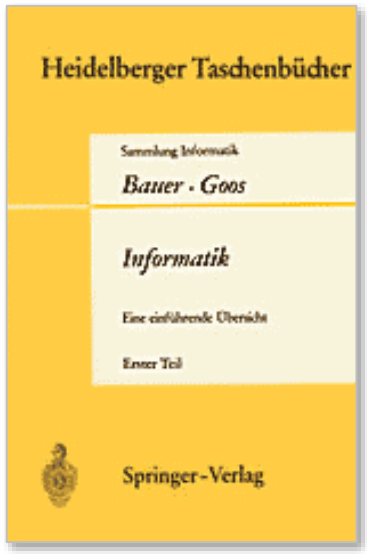
Friedrich Ludwig Bauer

Heinz Rutishauser

„War man zunächst froh, überhaupt Rechenautomaten zu besitzen, so musste man bald erkennen, dass man ihre Vorteile mit einer zeitraubenden, Unsicherheit tragenden Programmierungsarbeit bezahlte.“ [Wilhelm Kämmerer, *Ziffern-Rechenautomat mit Programmierung nach mathematischem Formelbild*, 1958]

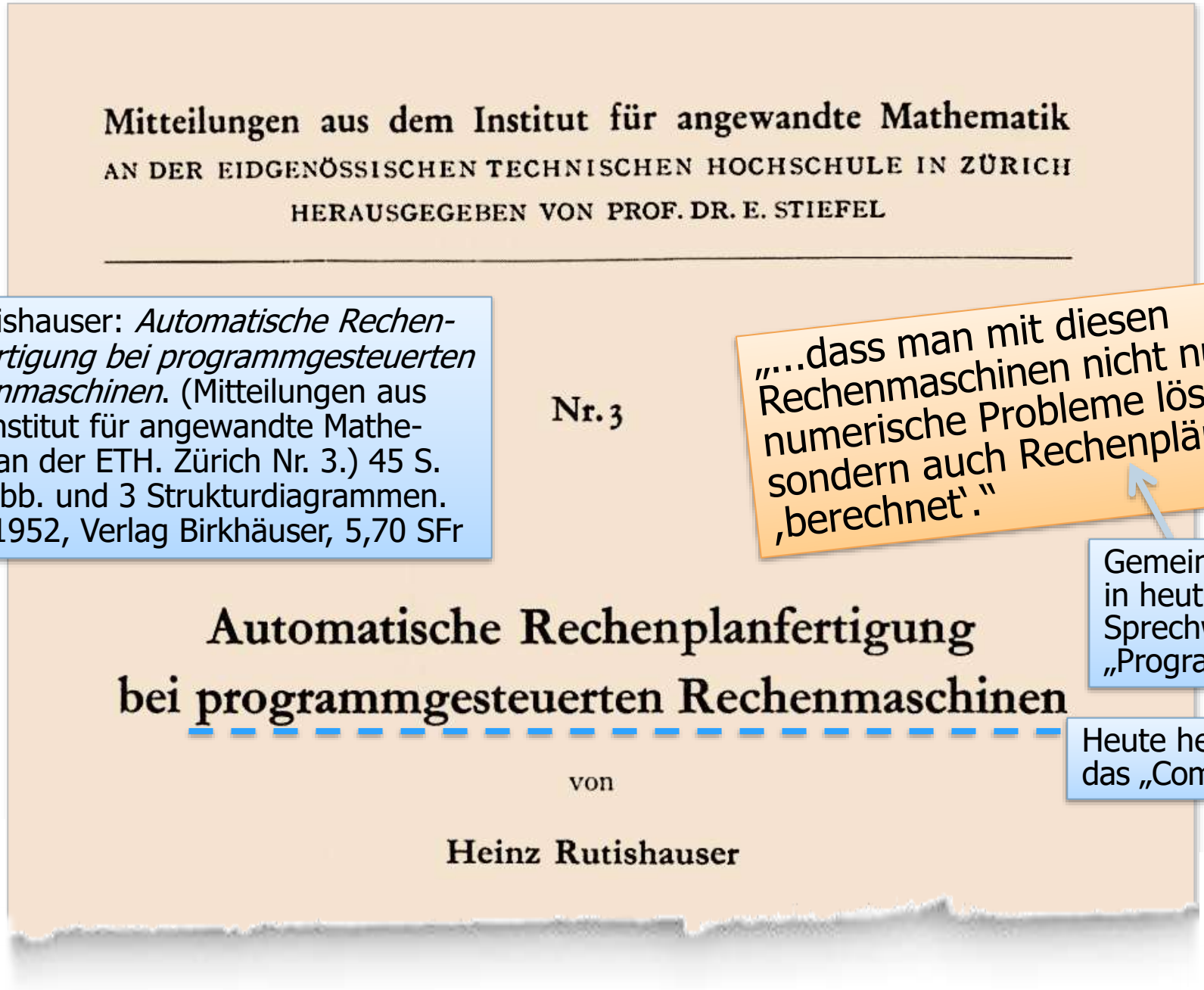
Friedrich Ludwig Bauer (1924 – 2015)

Friedrich Ludwig Bauer (1924–2015) war ein deutscher Informatik-Pionier. Er konstruierte in den 1950er Jahren Verschlüsselungsmaschinen sowie eine elektromechanische „aussagenlogische Maschine“ (STANISLAUS). Ab 1956 beteiligte er sich an der internationalen Zusammenarbeit zur Schaffung der Programmiersprache Algol 60, 1957 erfand er das Prinzip des Stacks.



Er hielt 1967 an der TU München die erste offizielle Informatikvorlesung Deutschlands und prägte 1968 den Begriff „Software Engineering“.

ETH-Professor Walter Gander bemerkte in einem Nachruf: „Bauer war einer der letzten ‘Allwissenden’ in der Informatik: Er hatte die ganze Entwicklung von den ersten Relais- und Röhrenmaschinen bis hin zum Laptop nicht nur miterlebt, sondern auch mitgestaltet.“



Mitteilungen aus dem Institut für angewandte Mathematik

AN DER EIDGENÖSSISCHEN TECHNISCHEN HOCHSCHULE IN ZÜRICH

HERAUSGEGEBEN VON PROF. DR. E. STIEFEL

Nr. 3

Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen

von

Heinz Rutishauser

H. Rutishauser: *Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen*. (Mitteilungen aus dem Institut für angewandte Mathematik an der ETH. Zürich Nr. 3.) 45 S. m. 8 Abb. und 3 Strukturdiagrammen. Basel 1952, Verlag Birkhäuser, 5,70 SFr

„...dass man mit diesen Rechenmaschinen nicht nur numerische Probleme löst, sondern auch Rechenpläne ‚berechnet'."

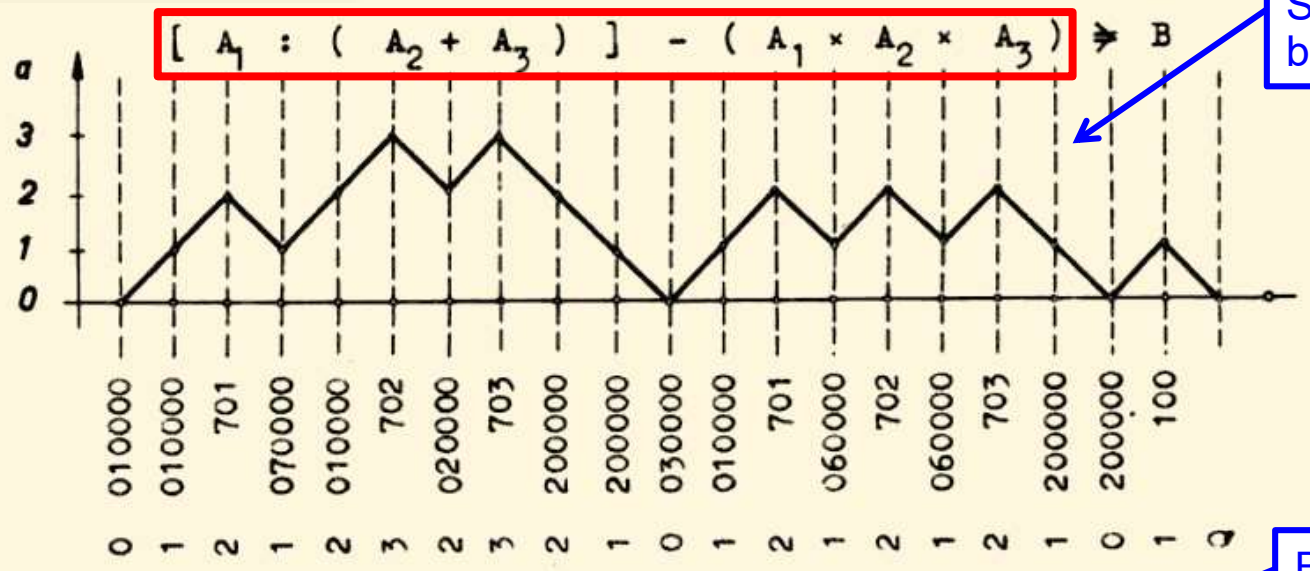
Gemeint sind in heutiger Sprechweise „Programme"

Heute heisst das „Computer"

Auszug aus Rutishausers „Automatische Rechenplanfertigung“

Für den als Beispiel angeführten Klammerausdruck (2.1) zeigt die folgende Aufstellung die Zahlfolgen a_k und b_k, sowie eine graphische Darstellung der a_k :

Fig. 2



Syntax-baum

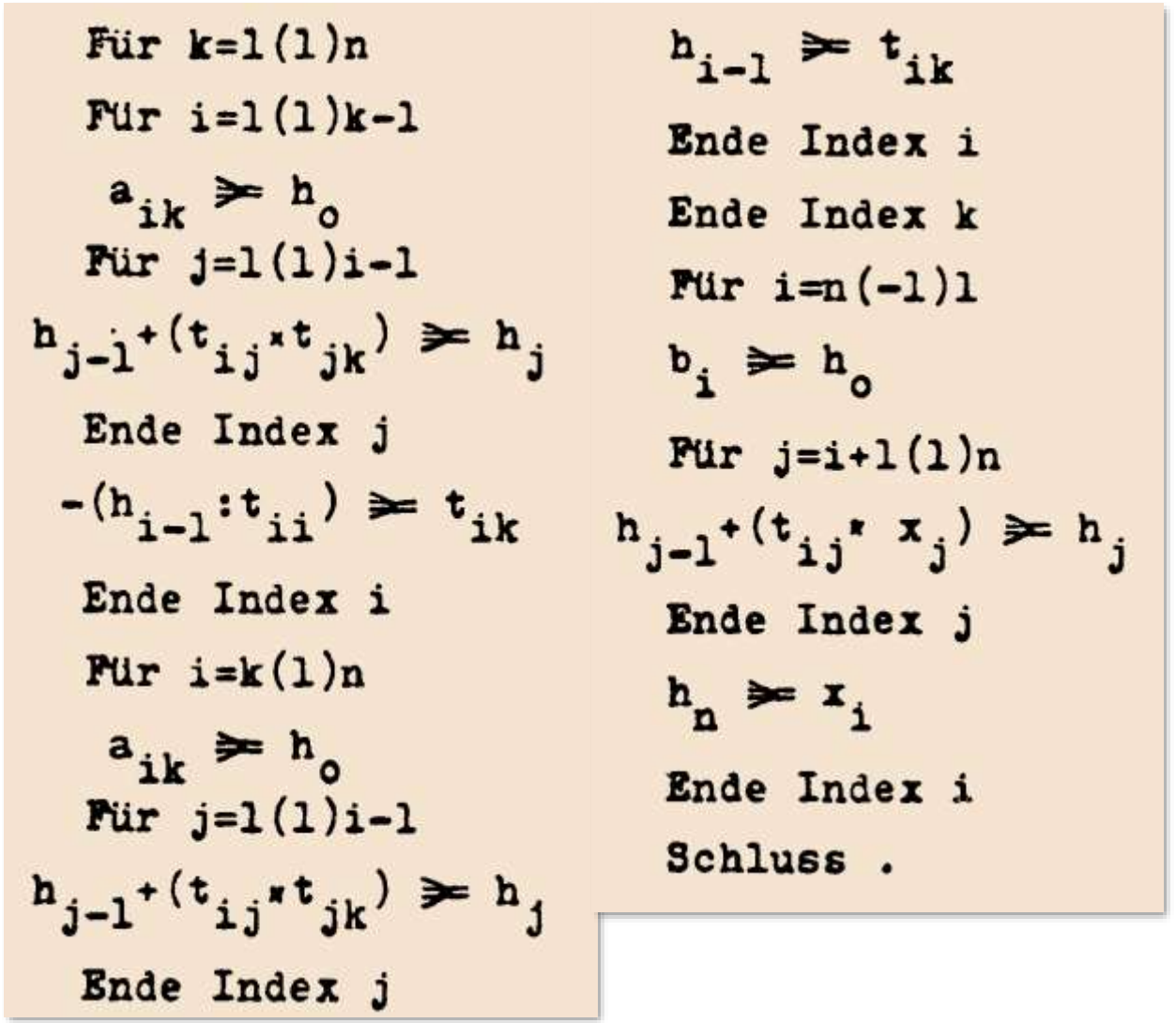
Blätter

Innere Baum-knoten

Dabei ist angenommen, dass die Zahlen A_i, bzw. B in den Zellen 700+i, bzw. 100 gespeichert seien. Aus der graphischen Darstellung erkennt man sofort, dass den "Bergspitzen" die Operanden des Klammerausdruckes, den "Tälern" dagegen die Operationszeichen entsprechen, während die Klammern in den "Hängen" liegen.

Ansätze algorithmischer Programmiersprachen

Rutishauser entwickelte früh eine Notation für numerische Algorithmen, wovon zentrale Elemente später in die von ihm wesentlich mitgestaltete Sprache Algol eingeflossen sind. Hier aus obiger Veröffentlichung ein Beispiel mit geschachtelten „for"-Schleifen (Gauß-Banachiewicz-Eliminationsverfahren)



Friedrich L. Bauer erinnert sich…

„Rutishauser nahm noch an, dass die Formel explizit geklammert ist, und unter dieser Annahme zeigte Corrado Böhm 1952, dass man die Auswertung auch sequentiell, normalerweise von links nach rechts, vornehmen kann. In üblicher Schreibweise wird jedoch unter Annahme einer Präzedenz der Multiplikation über der Addition und der Subtraktion auf die vollständige Klammerung verzichtet. In FORTRAN wurde ab 1954 (P.B. Sheridan) durch einen vorgeschalteten Durchlauf die Klammerung vervollständigt. L. Kalmar machte dazu den witzigen Vorschlag, das Multiplikationszeichen × überall durch die Folge)×(zu ersetzen (und die ganze Formel extra einzuklammern). Die entstehenden redundanten Klammern störten Kalmar nicht.“ [Aus: F.L. Bauer: Frühe Zeugnisse der „Software“. Informatik-Spektrum, 2006, 29(6), S. 433-441]

Kämmerer schreibt 1958 zu Rutishausers Methode: „Das Charakteristische dieses Verfahrens ist ein fortgesetztes Durchmustern mit einem immer wieder von links nach rechts wechselnden Augenspiel“. Tatsächlich hat das Verfahren einen mit der Eingabelänge quadratischen Zeitaufwand. F.L. Bauer bezeichnete es in Anspielung an die Prozession in Echternach (Luxemburg) als „Springprozession Rutishausers“.

Friedrich L. Bauer erinnert sich weiter

„Wir verbesserten das Programm von Rutishauser, das eine Springprozession über die Formel vollführte, zu einem streng sequentiellen Verfahren, das auch auf die vollständige Klammerung verzichtete unter Einführung von Operator-Rangordnungen. Das wurde die Grundlage unserer späteren Patentanmeldung von 1957, wobei es das ‚Kellerprinzip' verwendete, nämlich den im Rechner STANISLAUS eingeführten mehrgeschossigen Speicher vom (last in - first out)-Typus.

Klaus Samelson hatte auch die Idee, neben dem ‚Zahlkeller' für Zwischenergebnisse des STANISLAUS auch einen ‚Operationskeller' zu verwenden, um die jeweils ihres Ranges wegen ‚zurückgestellten' Operationen zu kellern."

„Wie sehr das Kellerprinzip inzwischen die Informatik durchdrungen hat, zeigt folgender kleiner Vorfall: Als ich kürzlich einem jungen Mitarbeiter gegenüber äusserte, Prof. Bauer habe den Kellerspeicher erfunden, fragte er: ‚Was gab es denn da zu erfinden? Das ist doch einfach ein Stack!' " -- Fritz Lehmann

Friedrich L. Bauer in einem Interview 1987

"We knew how to parse mechanically a formula written in Polish or in reverse Polish notation. So when we discussed in 1954 or 1955, Samelson and myself, how to do algebraic compilations, the whole question was: now we have parentheses, and what we do now with parentheses? The solution seemed to be obvious one day; I cannot say whether Samelson or myself... One of us said to the other one, «It's quite simple. We have to push down the parentheses too; because before we had pushed down our intermediate results for Polish notation, so we have to push down the parentheses too.» That meant that we had an on-line method for transforming algebraic notation into one-address code or three-address code.

Now we were aware at that time of Rutishauser's paper from about 1951 on formula translation. Rutishauser had a different method. He had a method that would walk back and forth and would work down from the top of what he called the parenthesis mountain. Our method seemed to be much simpler; in particular, it didn't walk back and forth – it was monotonously running. It went proportional to n if n is the length of the formula, and not to n^2 – which happened with Rutishauser's method. So we considered it to be much simpler and much more efficient.

That started the whole thing on our side of the ALGOL program language – because very soon we found out that not only our arithmetic formula could be parsed this way, but practically everything that you would want to write down in program, provided it has nested structure."

Friedrich L. Bauer schrieb schon 1960…

Rutishauser [4] was first in recognizing that an effective solution of the programming problem requires introducing conventional notation (commonly called "problem oriented language" today) into programming and having the computer do all subsequent phases of translation work. His proposal published in 1952 [5] and the subsequent work of Böhm [14] found no immediate response.

4. RUTISHAUSER, H. Über automatische Rechenplanfertigung bei programmgesteuerten Rechenanlagen. *Z. Angew. Math. Mech.* *31* (1951), 255.

5. RUTISHAUSER, H.: Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen. Mitt. Inst. f. Angew. Math. der ETH Zurich, Nr. 3 (1952).

Auszug aus: K. Samelson, F. L. Bauer: *Sequential Formula Translation*, Communications of the ACM 3(2), pp. 76-83, Feb. 1960

"Computation of a program"

JOURNAL
OF THE
ASSOCIATION FOR COMPUTING
MACHINERY

VOL. 2 *January, 1955* NO. 1

SOME PROGRAMMING TECHNIQUES FOR THE ERMETH*

By HEINZ RUTISHAUSER
Swiss Federal Institute of Technology, Zürich, Switzerland

At the Swiss Federal Institute of Technology, Zürich, Switzerland, an electronic digital computer (ERMETH) with floating decimal arithmetic and a 10000 word magnetic drum storage[4] is now under construction. Director

3. *Computation of a Program.* Based on the computing experience with the Z4 it is to be expected that we shall have to solve many problems

The customer has to write down the formulas of his problem in a special but quite natural manner, which we call the *algorithmic writing* of the problem. An exact definition will be published later; here we give as an example the algorithmic writing for the approximate calculation of log x with arithmetic-geometric means:* **

For $k=0$: $x^2+1 \Rightarrow a_k$; $2x \Rightarrow b_k$

For $k=0(1)m$: $\frac{1}{2}(a_k+b_k) \Rightarrow a_{k+1}$; $\sqrt{a_{k+1} \cdot b_k} \Rightarrow b_{k+1}$

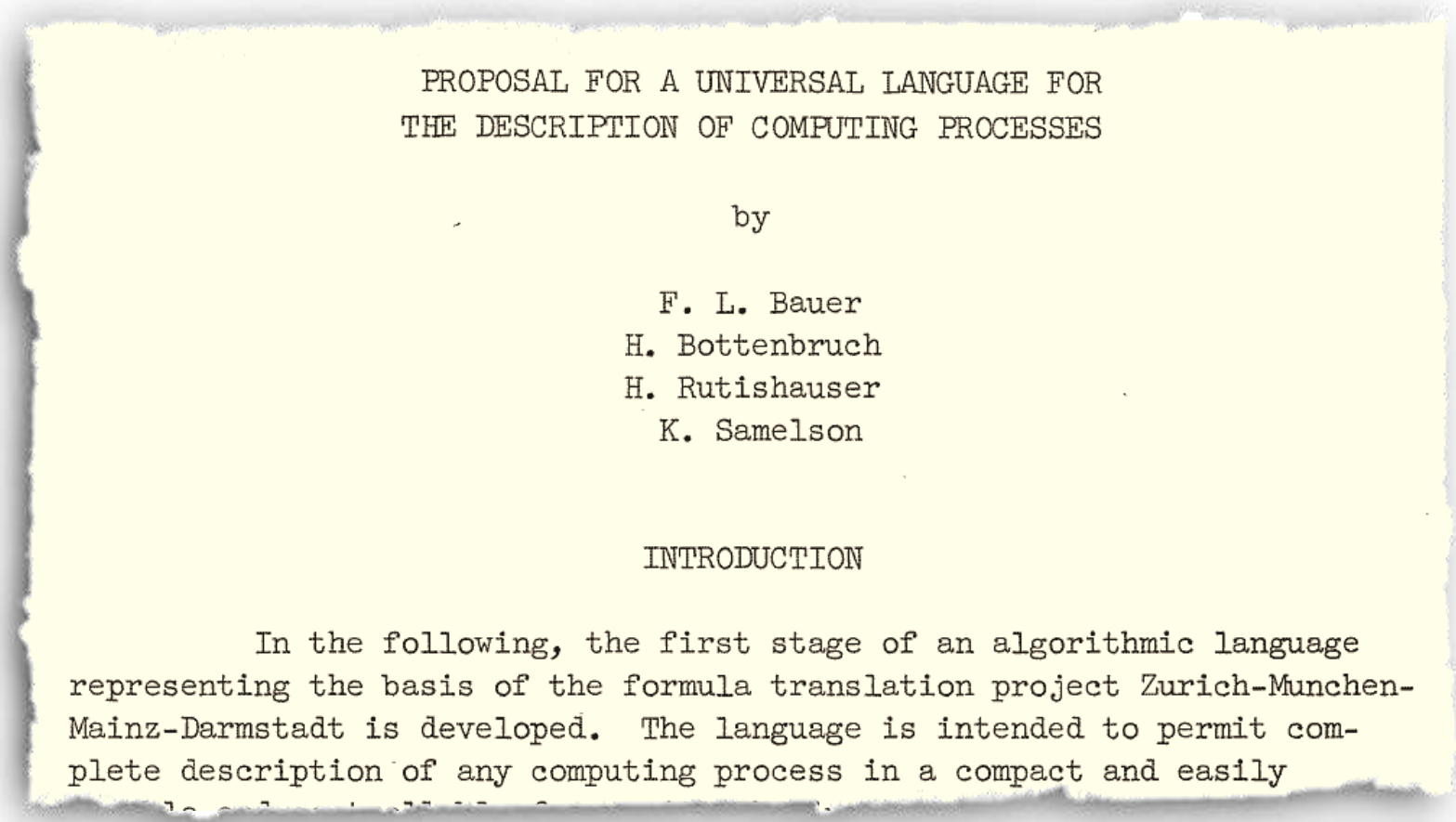
$k=n$, if $|a_k - b_k| - 10^{-5} \; b_k < 0$

*This is rather an example for the algorithmic writing than the best method for the computation of log x.

**The symbol $\Rightarrow$ has the meaning that a number is to be computed according to the formula on the left side of "$\Rightarrow$" and is to be denoted by the symbol on the right side of "$\Rightarrow$". The symbol => was first used in this sense by K. Zuse.

Schlüsselwörter "for", "if" etc.

If this description of the problem is arithmetized (*i. e.*, the symbols "For" ; "if" ; "=>" ; "(" ; etc. are replaced by appropriate numbers), the machine itself may compute the program including the conditional orders which control the exits from the induction loops of the flow diagram. A description of a possible method for computing a program is given in Rutishauser,[2] especially §4.* Of course the same method can also be



PROPOSAL FOR A UNIVERSAL LANGUAGE FOR
THE DESCRIPTION OF COMPUTING PROCESSES

by

F. L. Bauer
H. Bottenbruch
H. Rutishauser
K. Samelson

INTRODUCTION

In the following, the first stage of an algorithmic language representing the basis of the formula translation project Zurich-Munchen-Mainz-Darmstadt is developed. The language is intended to permit complete description of any computing process in a compact and easily

Vom 16. bis 22. Oktober 1957 trafen sich auf Einladung von Rutishauser u.a. er, Bauer, Hermann Bottenbruch (Doktorand bei Alwin Walther in Darmstadt), Klaus Samelson (Mainz und TU München) in Lugano, um gemeinsam weiter am Entwurf einer algorithmischen Programmiersprache zu arbeiten. Kurz danach erschien das *Proposal for a universal language for the description of computing processes*, mit welchem die Algol-Entwicklung im eigentlichen Sinne, unter Einbezug amerikanischer und weiterer europäischer Wissenschaftler, eingeleitet wurde.

$$(a - b) \times c \Rightarrow t \qquad (1)$$
$$t + s - (a \times c) + (b \times b) \Rightarrow z$$

wherefrom the machine carries out translation into the machine program.

The main principle was to write to the left of the "ergibt"-symbol "=>" the arithmetic expression to be evaluated and to the right the designation of the new quantity defined by the calculation. This "ergibt" symbol corresponds better to the dynamic process of computing than the usual equality symbol. Especially it can be used in situations where the latter may lead to contradictions, e.g., s + 2 => s ("old" s plus 2 yields "new" s).

For description of loops generated by running subscripts, a symbolization of the following form was proposed (example for multiplication of a matrix by a vector):

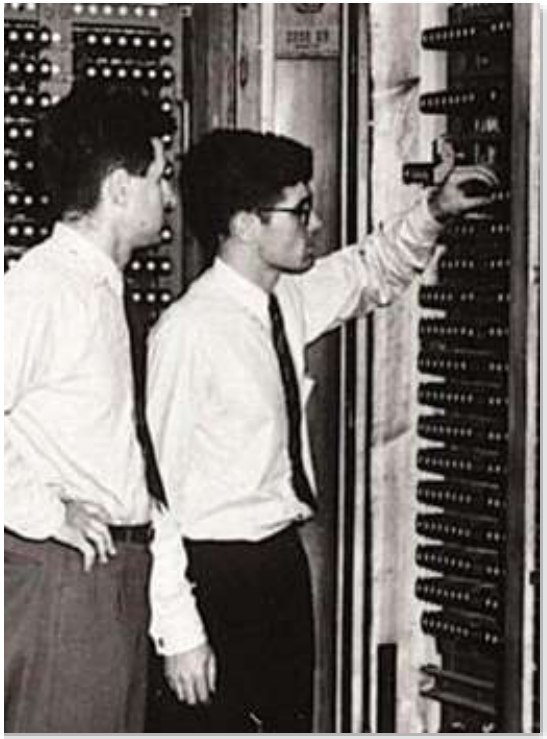
$$\begin{array}{lll} \text{Für } i = 1(1)\ n: & & \\ \text{Für } j = o: & o \Rightarrow h_j & \\ \text{Für } j = 1(1)\ n: & h_{j-1} + (a_{ij} \times x_j) \Rightarrow h_j & (2) \\ \text{Für } j = n & h_j \Rightarrow y_i & \end{array}$$

2. Principles of the New Language

The new language follows essentially the ideas given by Rutishauser in his original paper as stated in Section 1: A formula

Man erkennt, dass das Proposal auf den vorangehenden Arbeiten von Rutishauser beruht; der Vorschlag, die Variablenzuweisung durch „=>" auszudrücken (statt durch „=" wie bereits in Fortran) wird allerdings später leider nicht übernommen.

Ambros Speiser erinnert sich...
...an Zuses Z4-Computer, Rutishauser und dessen Compiler



Rutishauser (li.) und Speiser (re.)

Rutishauser, who was exceptionally creative, devised a way of letting the Z4 run as a compiler, a mode of operation which Zuse had never intended. For this purpose, the necessary instructions were interpreted as numbers and stored in the memory. Then, a compiler program calculated the program and punched it out on a tape. All this required certain hardware changes. Rutishauser compiled a program with as many as 4000 instructions. Zuse was quite impressed when we showed him this achievement.

It was my job to make the necessary wiring changes. I vividly remember the hours it took me to find out which of the perhaps 30,000 soldering joints had to be changed to implement Rutishauser's ideas!

[Konrad Zuse's Z4: Architecture, Programming, and Modifications at the ETH Zurich]

Autocode: Instructions take the form of words

Programme in algebraischer und algorithmischer Notation sowie zugehörige „Compiler" werden fast zeitgleich in England erfunden:

Auszug aus Kapitel 5 "Programming for high-speed digital calculating machines" von John Makepeace Bennett, Alick Edwards Glennie, veröffentlicht in "Faster Than Thought" (Hg: Bertram Vivian Bowden), S. 112-113, 1953:

AUTOMATIC CODING

It has been found possible to use the Manchester machine to convert programmes written in a notation very similar to simple algebraic notation into its own code. The programme, in this simple form, is fed into the machine under the control of a special input routine which makes the translation into the code of the machine. The notation has been designed to be as near as possible to the usual notation of algebra, so that the construction and checking of programmes is made easy; this lessens the difficulty that is found in constructing large and complicated programmes.

The notation consists of two parts; one for the description of the numerical calculations, and the other for the description of the organization of the calculations into a completely automatic process. The description of the numerical calculation is in the form of algebraic "equations" using the simple basic operations of addition, subtraction, and multiplication, as in the following equation which describes the addition

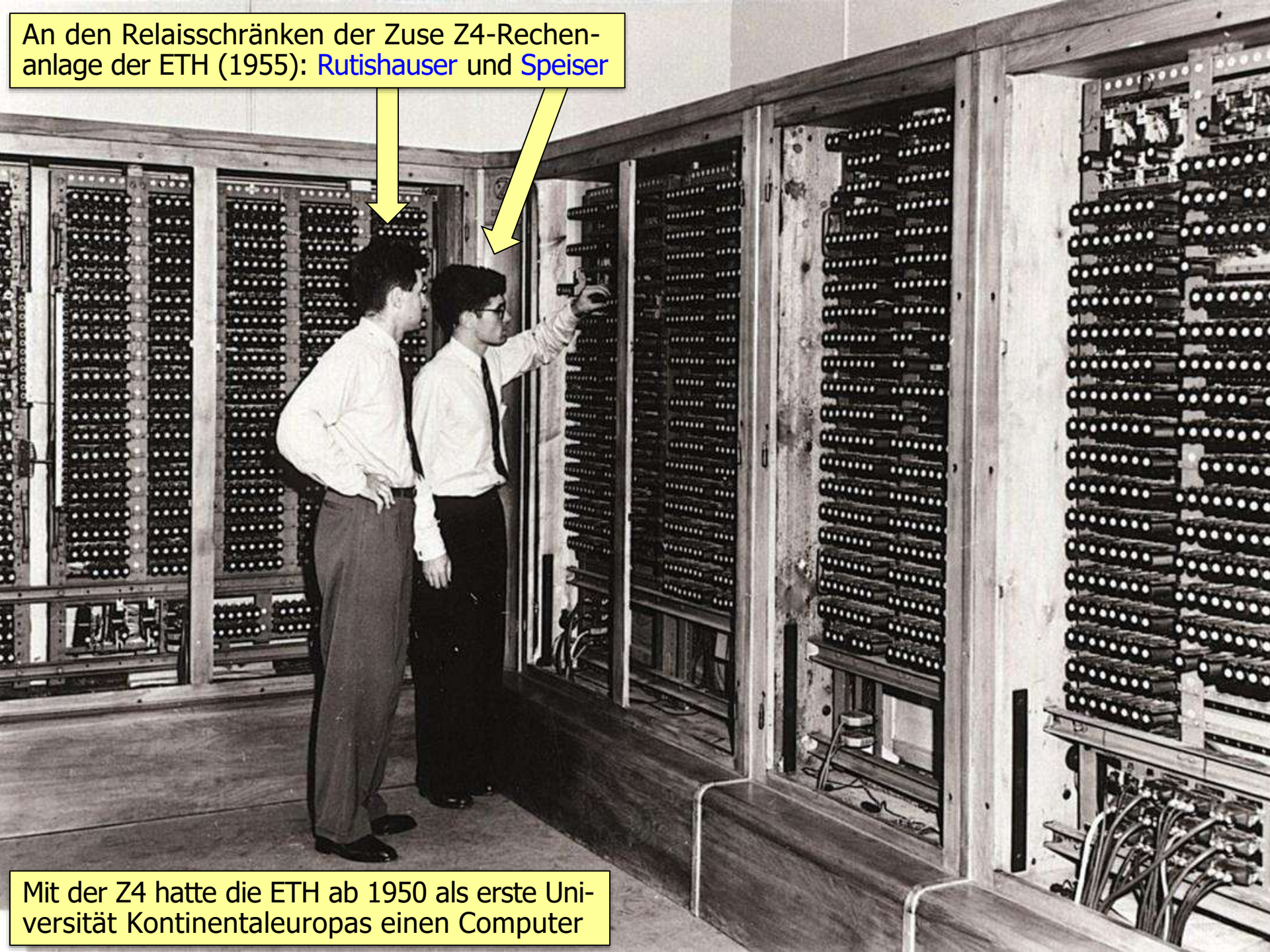
of several numbers contained in storage locations for which the code letters are *x, y, z* and so on. It is–

$$+\ x + y + z + a + b \rightarrow c$$

The result of the sum is to be found in the storage location *c*. These locations are not fixed, but may be defined according to need. This type of notation is sufficient for the description of most numerical calculations.

For the description of the organization, which forms a considerable part of any programme, instructions take the form of words (in English) which, when interpreted by the machine, cause the correct instructions in the machine's code to be synthesized. Thus if a subroutine is required during the calculation, for printing the results or the calculation of auxiliary functions, it is sufficient to write the word *subroutine* followed by a number describing which subroutine is meant. By an extension of this technique it would be possible to call for the particular subroutine by name (e.g. *cosine* for a subroutine for calculating cosines). This has not yet been done as the gain in convenience would be too small to warrant the trouble. In the system now in use, English words are used to specify transfers of control, counting and many of the more common programme tricks. The total number of descriptive words is 13, and this has been found adequate for most purposes.

Programming with such a system for making the machine do its own coding does not lead to the most economical or fastest programmes, but the loss in "efficiency" is not more than about 10 per cent and is a small price for the convenience that results.


An den Relaisschränken der Zuse Z4-Rechen-
anlage der ETH (1955): Rutishauser und Speiser
Mit der Z4 hatte die ETH ab 1950 als erste Uni-
versität Kontinentaleuropas einen Computer

Heinz Rutishauser (1918 – 1970)

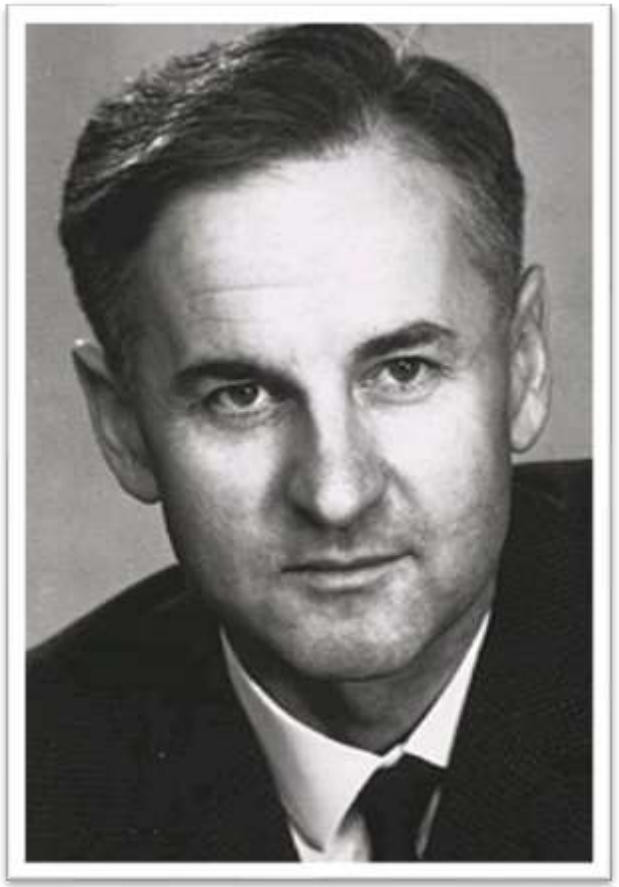
- Geboren 1918 in Weinfelden
- 1924 – 1936 Kantonsschule Frauenfeld
- Ab 1936 Studium an der ETH
- 1942 Diplom in Mathematik

Heinz Rutishauser (1918 – 1970)

- 1948 Dissertation an der ETH
- 1949–1955 ERMETH-Entwicklung (Elektronische Rechenmaschine der ETH; mit Ambros Speiser bei Prof. Stiefel)
  - Ab 1955 Professor an der ETH
  - Entscheidende Beiträge zur Programmiersprache ALGOL
- Ab 1968 Leiter der neu gegründeten ETH-Fachgruppe „Computerwissenschaften“

Heinz Rutishauser an der Konsole von Zuses Z4-Rechenanlage, 1950 (Bild: ETH-Bibliothek).

Heinz Rutishauser



„1968 war Rutishauser im Besitz eines Rufes nach München auf eine Professur, die mit der Leitung des Leibniz-Rechenzentrums verbunden war. Es gelang mir im Mai 1968 nicht, ihn zu überreden, den Ruf anzunehmen; er nannte mir damals gesundheitliche Gründe und seine Befürchtungen waren, wie sich leider herausstellte, nicht grundlos. Dass er in Zürich verblieb, hat dann dem Aufbau der zunächst Computerwissenschaften genannten Informatik sehr geholfen.“

Aus F.L. Bauer „Computer und Algebra“ (in: *Zwanzig Jahre Institut für Informatik*, Bericht des Instituts für Informatik, ETH Zürich, 1988)

Informationsverarbeitung und ETH Zürich haben einen unersetzlichen Verlust erlitten: am 10. November 1970 starb Prof. Dr. Heinz Rutishauser, mitten in seiner Arbeit am Institutsschreibtisch an einem Herzleiden, das ihn schon lange gequält hatte, und dennoch völlig unerwartet. Nicht nur seine Familie bleibt in tiefem Schmerz zurück: wohl selten hat ein Mensch eine solche Lücke hinterlassen und soviel Empfindung bei Mitarbeitern und Kollegen. Heinz Rutishauser war eine ungewöhnliche Mischung von Genialität und Menschlichkeit; seine Bescheidenheit und Güte hielten ihn so sehr in der Unauffälligkeit, daß die Fachwelt erst allmählich bemerken wird, was sie an ihm verloren hat. [...]

Die besondere Liebe Rutishausers, schreibt sein Lehrer Walter Saxer, galt den Algorithmen; in ihrer Erfindung war er ein wahrer Meister. Tatsächlich bilden viele von ihm aufgestellten Algorithmen den Grundbestand jedes einschlägigen Lehrbuchs. Von dieser Basis aus wurde Rutishauser zum Pionier der numerischen Analysis. [...]

Rutishausers Habilitation „Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen“ ist der Beginn der Programmierungssprachen und der Ausgangspunkt für ALGOL. Sein Vorschlag, die Rechenmaschine selbst für die Rechenplanfertigung heranzuziehen, erscheint uns heute so selbstverständlich, daß wir uns nicht vorstellen können, welchen Fortschritt gegenüber den Programmboxes der frühen Maschinen dieser Gedanke bedeutet hat. [...]

Wer ihn in seiner bedächtigen, aber urteilssicheren, wortkargen, aber gedankenreichen Art gekannt hat, seinen feinen Humor und seine behutsame Geschicklichkeit, Differenzen zu bereinigen, weiß, daß wir einen genialen und charakterstarken Menschen verloren haben.

Nachruf, verfasst von H. Zemanek, Elektronische Rechenanlagen 12, S. 305, Dezember 1970

Am 10. November 1970, an seinem Schreibtisch in der ETH sitzend, starb an einem akuten Herzversagen Dr. HEINZ RUTISHAUSER, ord. Professor für Angewandte Mathematik, der große Meister der Numerischen Mathematik und der Computer-Wissenschaften, der Informatik. Die Eidgenössische Technische Hochschule zu Zürich, die so viele bedeutende Gelehrte hervor- und zum Wirken gebracht hat, besaß in RUTISHAUSER einen Mann, dessen Ruhm über die Welt verbreitet ist.

Die ungewöhnlichen Fähigkeiten RUTISHAUSERS zeigten sich seinen akademischen Lehrern schon während des Studiums, und mehr noch in seiner ausgezeichneten Dissertation, die er unter SAXER auf dem Gebiet der Funktionentheorie fertigte. Seine Habilitationsschrift im Jahre 1951 brachte dann sogleich den ganz großen Wurf: Unter dem Titel *„Automatische Rechenplanfertigung bei programmgesteuerten Rechenmaschinen“* legte er die Grundlage für die Entwicklung höherer Programmiersprachen und deren Übersetzung. Es heißt darin: „*…gewann … die Überzeugung, daß es möglich sein müsse, die programmgesteuerte Rechenmaschine selbst dank ihrer Vielseitigkeit als Planfertigungsgerät zu verwenden. Dies würde also bedeuten, daß man mit diesen Rechenmaschinen nicht nur numerische Probleme löst, sondern auch Rechenpläne ‚berechnet‘.“* In genialer Weise die Anregung aufgreifend, die ZUSE mit seinem „Plankalkül“ gegeben hatte, zeigt RUTISHAUSER sogleich die Lösung durch die Maschine selbst und stößt darin, weit über VON NEUMANN hinaus, zur vollständigen Abschaffung des Unterschieds zwischen Daten und Programm vor.

Mehr zu Heinz Rutishauser:

Hanna Rutishauser: Numerik, ALGOL und die Schweizer Hochalpen. Zur Arbeit an der Biografie von Heinz Rutishauser (1918–1970). Informatik-Spektrum, Okt. 2013, 36(5), 463-468, http://dx.doi.org/10.1007/s00287-013-0730-z

Nachruf, verfasst von F.L. Bauer, Computing 7, 129-130 (1971)

Konrad Zuse (1910 – 1995) präsentiert seinen Z4-Computer

„Immerhin besass das verschlafene Zürich durch die ratternde Z4 ein, wenn auch bescheidenes, Nachtleben. Ich selbst besass einen Schlüssel zum Hauptgebäude der ETH, und manches Mal bin ich spät in der Nacht durch die einsamen Züricher Gassen gegangen, um nach der Z4 zu sehen. Es war ein eigenartiges Gefühl, in die menschenleere ETH einzutreten und bereits im Parterre zu hören, dass die Z4 im obersten Stock noch einwandfrei arbeitete.“

Die Z4 an der ETH Zürich

I regard the computing machine as being not in the category of the large astronomical telescope, which is a pleasant but optional luxury for a university, but rather in the category of the electronuclear particle accelerator, which is a necessity for any university.
-- Louis N. Ridenour, Vice President Lockheed, 1947

„Bei längeren Rechenprozessen werden der Maschine Befehle über die Art und Aufeinanderfolge der auszuführenden arithmetischen Operationen gegeben, indem die nötigen Instruktionen in einem einfachen Code auf dem im Bild sichtbaren Lochstreifen vom Mathematiker eingelocht werden. Das Gerät führt dann die ganze Rechnung vollautomatisch durch, indem Zwischenresultate in dem Speicherwerk hinten links aufbewahrt und Schlußresultate auf der elektrischen Schreibmaschine gedruckt werden.“ -- Eduard Stiefel

„Programmgesteuertes Rechengerät“ ist eine treffende Bezeichnung früher Computer

NEUE ZÜRCHER ZEITUNG

TECHNIK

Mittwoch, 30. August 1950 Blatt 5
Mittagausgabe Nr. 1796 (50)

Das programmgesteuerte Rechengerät an der Eidgenössischen Technischen Hochschule in Zürich

Die Entwicklung programmgesteuerter Rechenmaschinen in den Vereinigten Staaten von Amerika wurde in den Artikeln „Elektronische Rechenmaschinen“ (vgl. Nr. 2140 der „N. Z. Z.“ vom 13. Oktober 1948) und „Die neueste elektronische Rechenmaschine“ (vgl. Nr. 871 der „N. Z. Z.“ vom 26. April 1950) behandelt. Nachstehend soll von einem Gerät deutscher Herkunft — Zuse K.-G., Neukirchen — die Rede sein, welches im Juli dieses Jahres am Institut für angewandte Mathematik der Eidgenössischen Technischen Hochschule in Zürich, das unter der Leitung von Prof. Dr. E. Stiefel steht, in Betrieb genommen wurde. Damit ist dieses Institut in der Lage, dem in der Schweiz immer stärker werdenden Bedürfnis nach einer leistungsfähigen Zentralstelle für numerische Rechnungen wenigstens teilweise gerecht zu werden. Bereits sind einige mathematische Probleme behandelt worden, und die Erledigung vieler anderer Aufgaben ist vorbereitet.

Merkmale des Gerätes

Das Gerät ist ein Glied in dem längeren Entwicklungsprogramm des Ingenieurs *Konrad Zuse*; es wurde im Auftrag des *Institutes für angewandte Mathematik* der E. T. H. unter Berücksichtigung von dessen Wünschen und Ideen von Zuse als *„Modell Z 4“* konstruiert. Die ursprüngliche Entwicklung in Deutschland erfolgte in den Kriegsjahren und verlief völlig unabhängig von den Untersuchungen in den Vereinigten Staaten. Es ist überaus interessant festzustellen, wie für die meisten wichtigen funktionellen Probleme beiderorts genau dieselbe Lösung gefunden wurde, wie aber anderseits gewissen Fragen sekundärer Wichtigkeit eine ganz unterschiedliche Bedeutung beigemessen wurde.

Eine kurze technische Charakterisierung lautet wie folgt: Elektromechanisch arbeitendes Gerät mit 2200 Relais, 21 Schrittschaltern und einem Speicher für 64 Zahlen, welcher mit neuartigen, mechanischen Schaltgliedern arbeitet; Verwendung des Dualsystems und der halblogarithmischen Darstellung; Multiplikationszeit 2,5 Sekunden; Programmsteuerung mit Hilfe zweier Lochstreifen, auf die wahlweise umgeschaltet werden kann; Eingabe von Zahlen durch eine Tastatur oder durch einen Lochstreifen; Abgabe der Resultate durch Lampenfeld, Lochstreifen oder Druckwerk.

Das duale Zahlsystem

Allgemein wird programmgesteuerten Rechengeräten häufig das duale Zahlsystem zugrunde gelegt, welches nur die zwei Zahlsymbole 0 und 1 verwendet, während das bekannte Dezimalsystem

Lesen wir eine Dezimalzahl von rechts nach links, so erhöht sich das Gewicht von Stelle zu Stelle um den Faktor 10. Im *Dualsystem* ist nun einfach dieser Faktor 10 durch 2 zu ersetzen. Also bedeutet die (nunmehr duale) Zahl abcd,efg den Ausdruck:

$$a \cdot 2^3 + b \cdot 2^2 + c \cdot 2^1 + d \cdot 2^0 + e \cdot 2^{-1} + f \cdot 2^{-2} + g \cdot 2^{-3}$$

Die Zahl 1 wird in beiden Systemen gleich dargestellt. Um jedoch duale von dezimalen Zahlen deutlich zu trennen, schreiben wir die duale 1 als L. — Dagegen weicht schon die 2 ab, indem sie dual L0 lautet; denn dies bedeutet ja $1 \cdot 2^1 + 0 \cdot 2^0 = 2$. Wenn einer Zahl (ohne Stellen nach dem Komma) rechts eine Null zugefügt wird, so vergrößert sie sich um den Faktor 2 (und nicht, wie im Dezimalsystem, um den Faktor 10). Auf diese Weise kann aus L0 = 2 auf einfachste Weise gebildet werden: L00 = 4, L000 = 8, L0000 = 16, usw.

Die Dualzahl L0L0L bedeutet nun also:

$$1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 21$$

Ganz analog sind etwaige Stellen nach dem Komma zu interpretieren; so wird L, 0LL wie folgt übersetzt:

$$1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1 + 1/4 + 1/8 = 1{,}375$$

Der große Vorteil, der das Dualsystem für Rechenautomaten so geeignet macht, nämlich die Reduktion der Anzahl der verwendeten Symbole auf nur zwei, wird allerdings durch einen Nachteil erkauft: Es braucht mehr Stellen, um eine bestimmte Zahl darzustellen. Die einstellige Zahl 8

Aenderung des Maßstabes durchgerechnet werden können.

Die beschriebene Darstellung bringt eine gewisse Komplikation der Rechenoperationen mit sich. So müssen vor einer Addition die beiden Summanden zunächst so verschoben werden, daß ihre Kommata untereinander zu liegen kommen, was an Hand eines Beispiels erläutert werden soll. Damit der Leser nicht durch das ungewohnte duale Zahlsystem verwirrt wird, ist das Beispiel im Dezimalsystem durchgeführt; doch wird daran erinnert, daß das Gerät in Wirklichkeit mit dualen Zahlen rechnet.

Es soll also etwa addiert werden: $2{,}345678 \times 10^2 + 9{,}876543 \times 10^{-1}$ (Man beachte, daß die eigentliche Zahl stets zwischen 1 und 10 liegt, also das Komma nach der ersten Stelle hat.) Nun müssen die beiden Summanden „ausgerichtet“ werden, d. h. die beiden Exponenten sind einander gleich zu machen, und zwar erhält der kleinere Exponent den Wert des größeren, also 2. Die Zahlen lauten nun, richtig untereinander geschrieben und addiert, wie folgt:

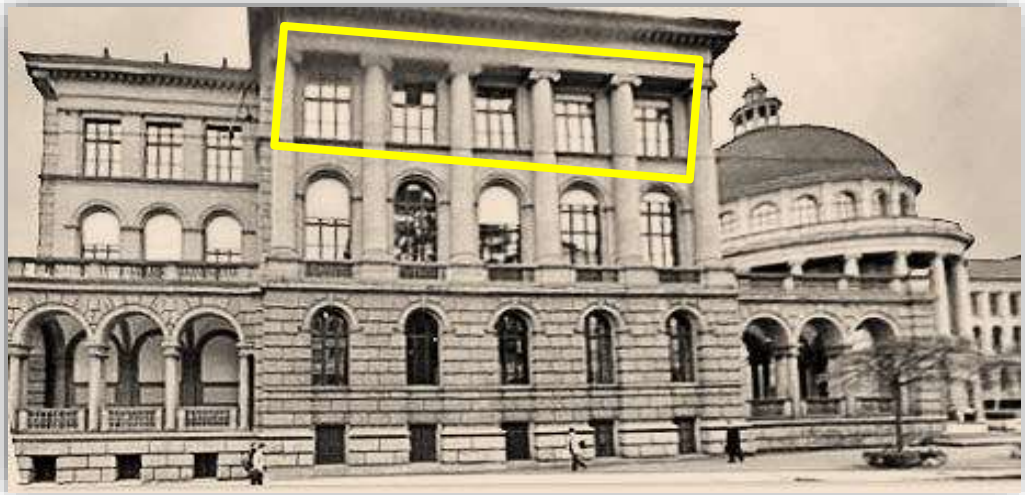
$$\begin{array}{r} 2{,}345678 \times 10^2 \\ 0{,}009876 \times 10^2 \\ \hline 2{,}355554 \times 10^2 \end{array}$$

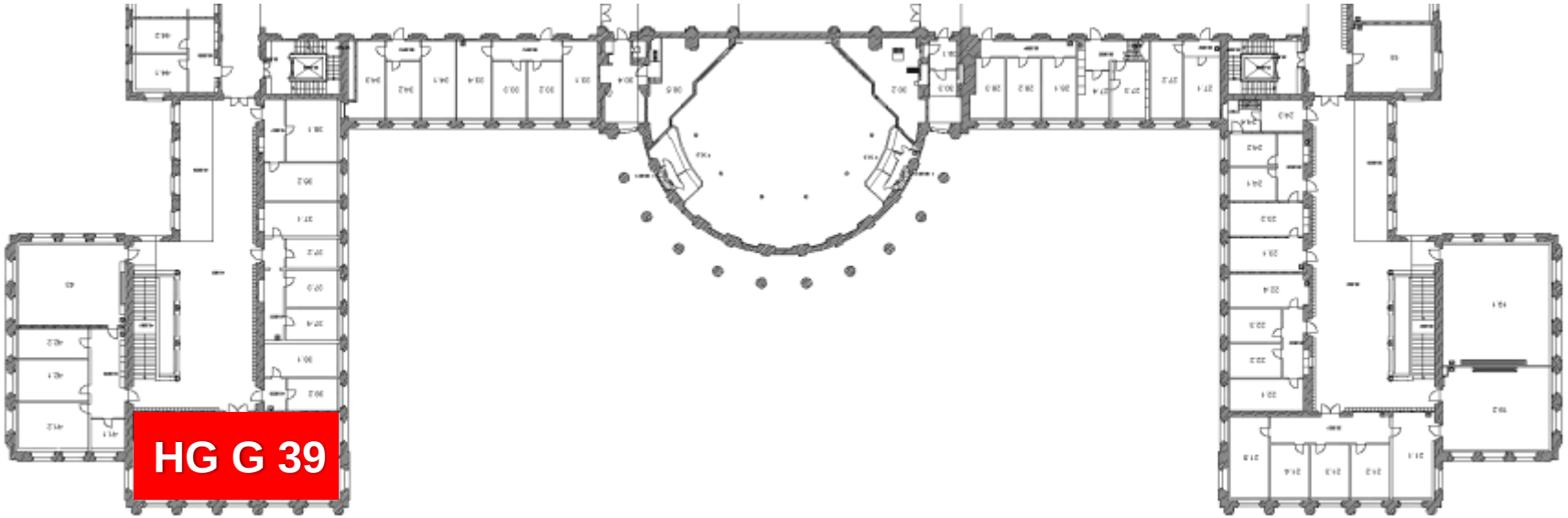
Es ist ersichtlich, daß bei der kleineren der beiden Zahlen rechts einige Stellen abgeschnitten werden mußten; denn wenn die Summanden siebenstellig gegeben waren, so soll auch das Resultat nicht mehr als sieben Stellen enthalten.

Abb. 2. Der Schaltpult bei der Fertigung eines Rechenplanes. Die Abtaster für den Lochstreifen sind deutlich sichtbar.

Befehle können „bedingt“ gegeben werden, d. h. ihre Ausführung wird von der Natur eines errechneten Resultates abhängig gemacht. Erst dadurch werden die außerordentlich weiten Perspek-

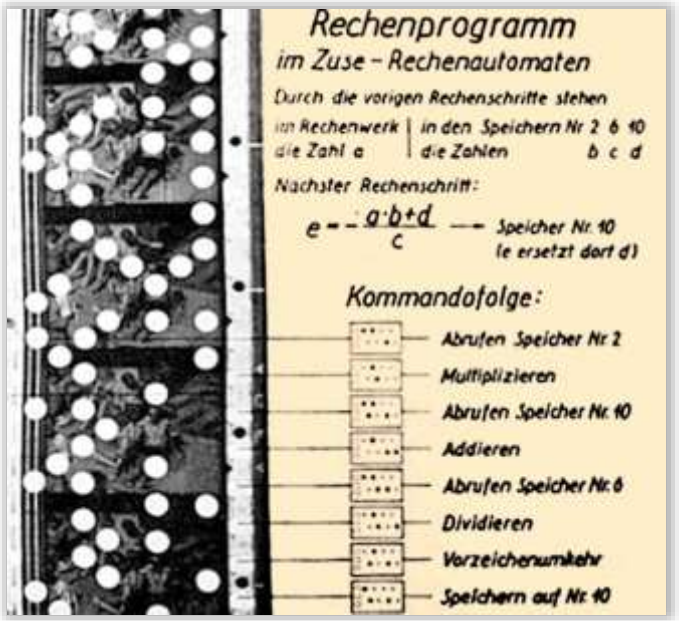
Die Z4 an der ETH Zürich





Der elektromechanische Computer Z4 von Zuse

- Konstruiert 1942–1945 in Berlin
- 2200 Telefonrelais, Gewicht ca. 1 t
- 64 mechanische Speicherplätze für Zahlen
  - Aber kein Speicher für Befehle
- Programme („Rechenpläne“) wurden auf Lochstreifen (alte Kinofilme) gestanzt
- Rechenwerk im Dualsystem, Zahlen in Gleitpunktdarstellung
- Ca. 1 s / Befehl; ca. 3 s / arithm. Operation
  - Programmieren mit Herstellen der Lochstreifen benötigte oft sowieso den grössten Zeitanteil, → geringe Geschwindigkeit kein grosses Problem



„Das Programm wurde auf ausgedienten Kinofilmstreifen gelocht, die wir oft auf interessante Bilder abgesucht haben. Doch wesentlich war, dass man diese Filme zu Schleifen zusammenkleben konnte und damit die Möglichkeit hatte, da es zwei Abtaststellen gab, bis zu zwei Programmschleifen zu bilden und ineinander zu schachteln.“
-- Peter Läuchli, ab 1953 Assistent bei Eduard Stiefel

Bildquellen (auch nächste Seite): www.spiegel.de/fotostrecke/fotostrecke-56183-3.html, www.ethistory.ethz.ch/rueckblicke/departemente/dinfk/bilder/1951_z4-lochstreifen.jpg

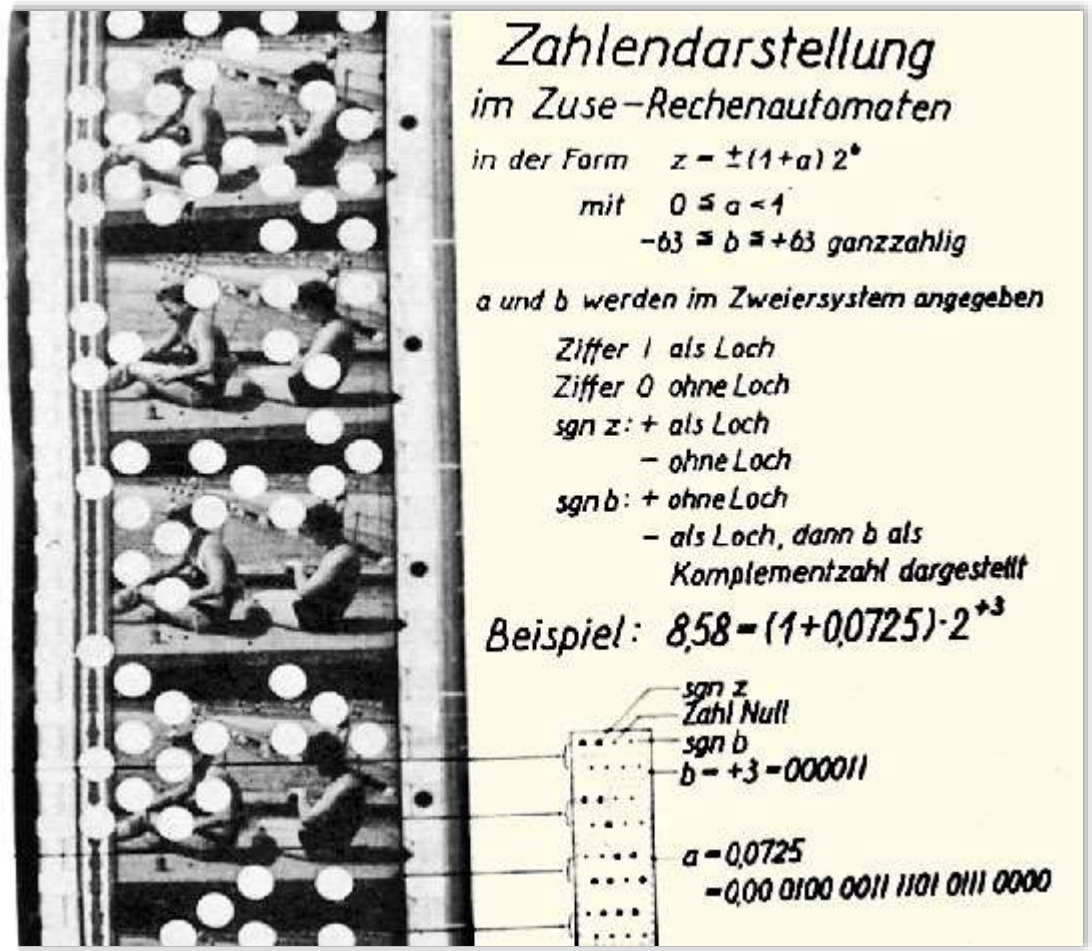
„Heute weiss jeder Programmierer, wie Programmschleifen zu schreiben sind. Aber keiner weiss mehr, dass Schleifen nicht einfach mit Uhu zusammengeklebt werden dürfen, weil sie sonst in den Lesegeräten stecken bleiben. Man muss den Klebstoff mit Aceton verdünnen.“ [Rudolf Kippenhahn]

Abtaster für Kommando-streifen

http://zuse.zib.de/

Rechts: Abtaster für das Hauptprogramm; links: Abtaster für ein Unter-programm als Schleife

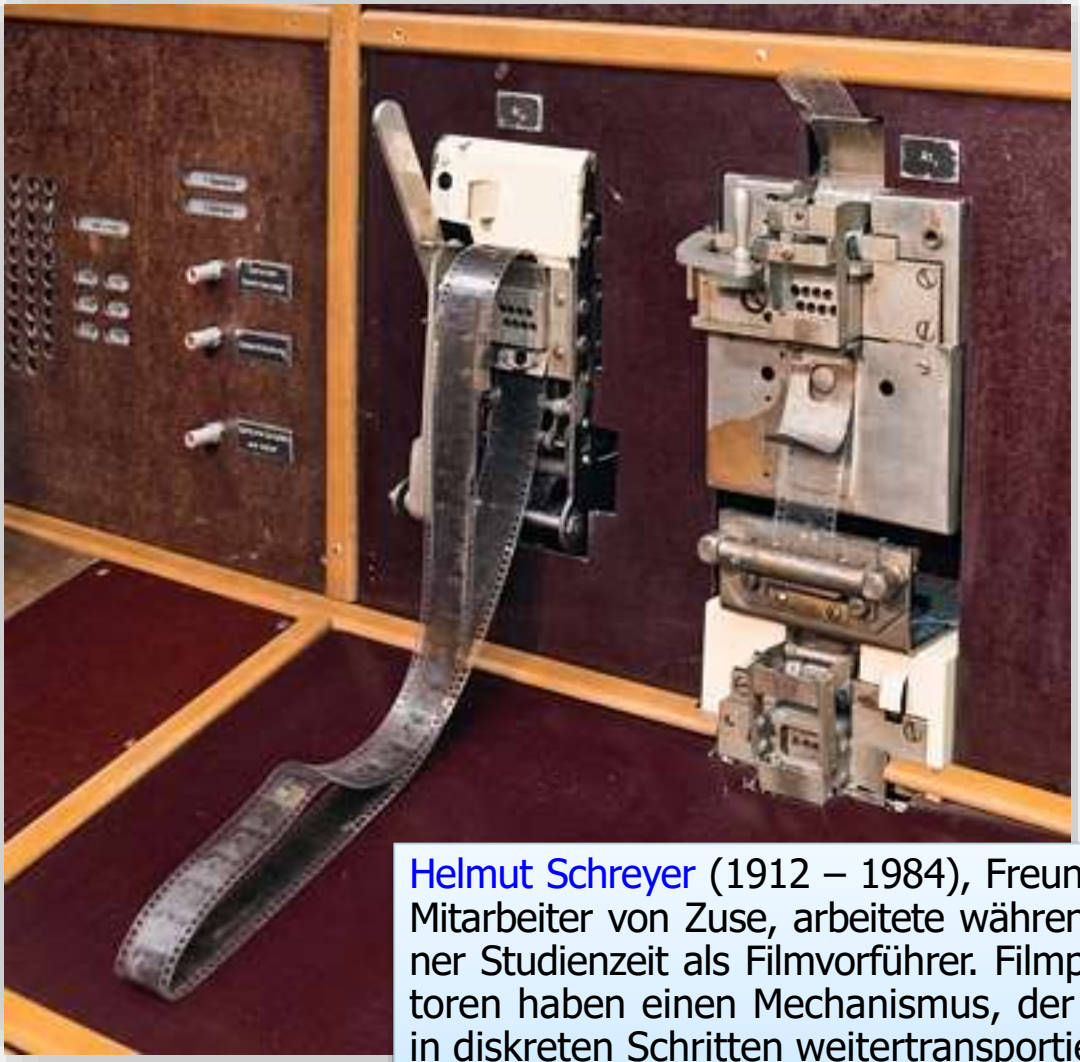
Z4: Lochfilm-Zahlencode



Um grosse Löcher und sicheres Abtasten zu erreichen, wurden je 4 Codelochungen in einer Spalte zu den benachbarten versetzt angeordnet, und die 8 Lochspalten einer Zahl in halblogarithmischer Notation ineinander geschachtelt.

Eine Schwimmbadszene (in der Tat meldet cloud.google.com/vision/ von Google “Sun Tanning 95%”; dagegen findet www.captionbot.ai von Microsoft “I think it’s a group of people sitting in a field”)

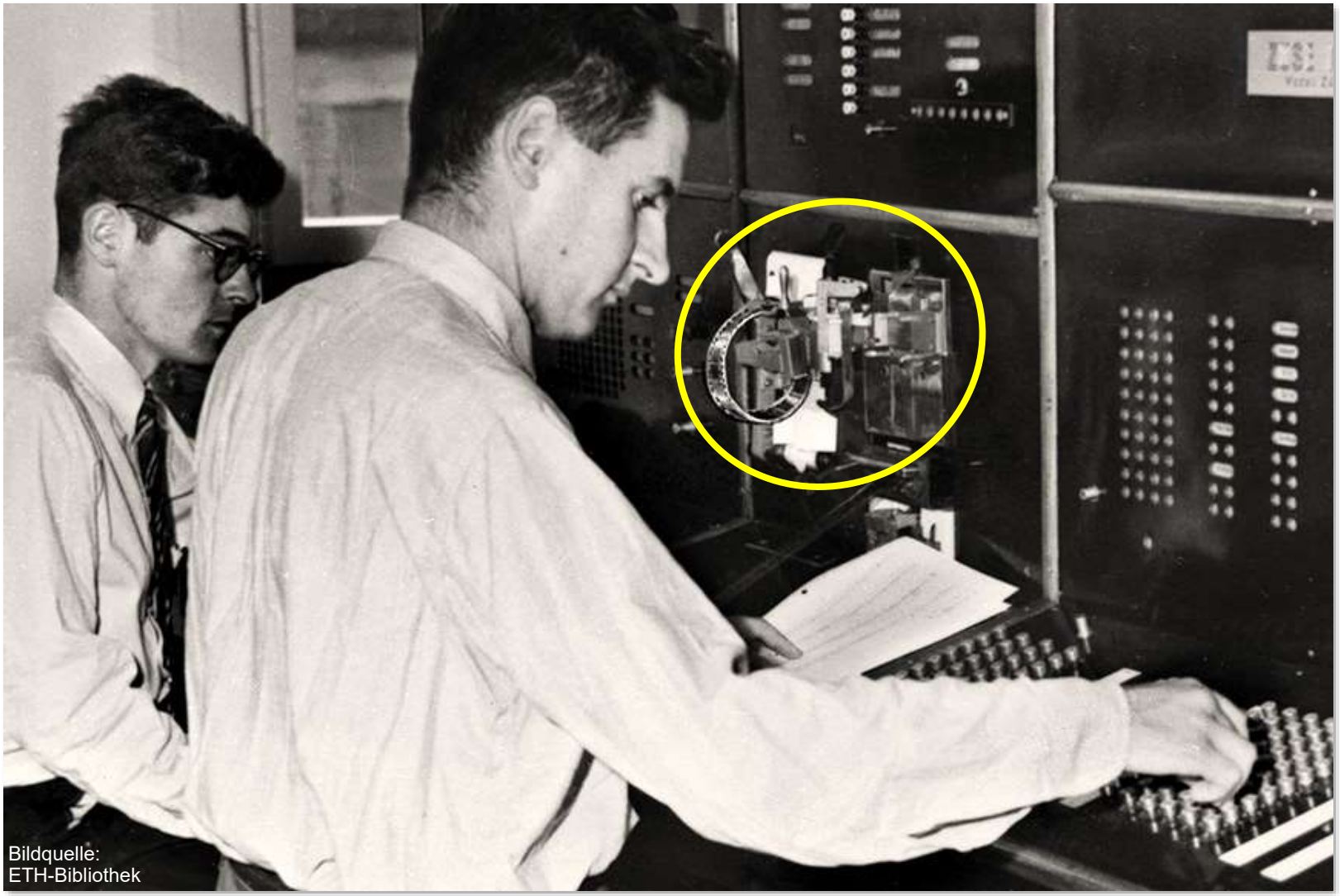
Z4: Programmbefehle auf gelochten Kinofilmstreifen



Helmut Schreyer (1912 – 1984), Freund und Mitarbeiter von Zuse, arbeitete während seiner Studienzeit als Filmvorführer. Filmprojektoren haben einen Mechanismus, der Filme in diskreten Schritten weitertransportiert.

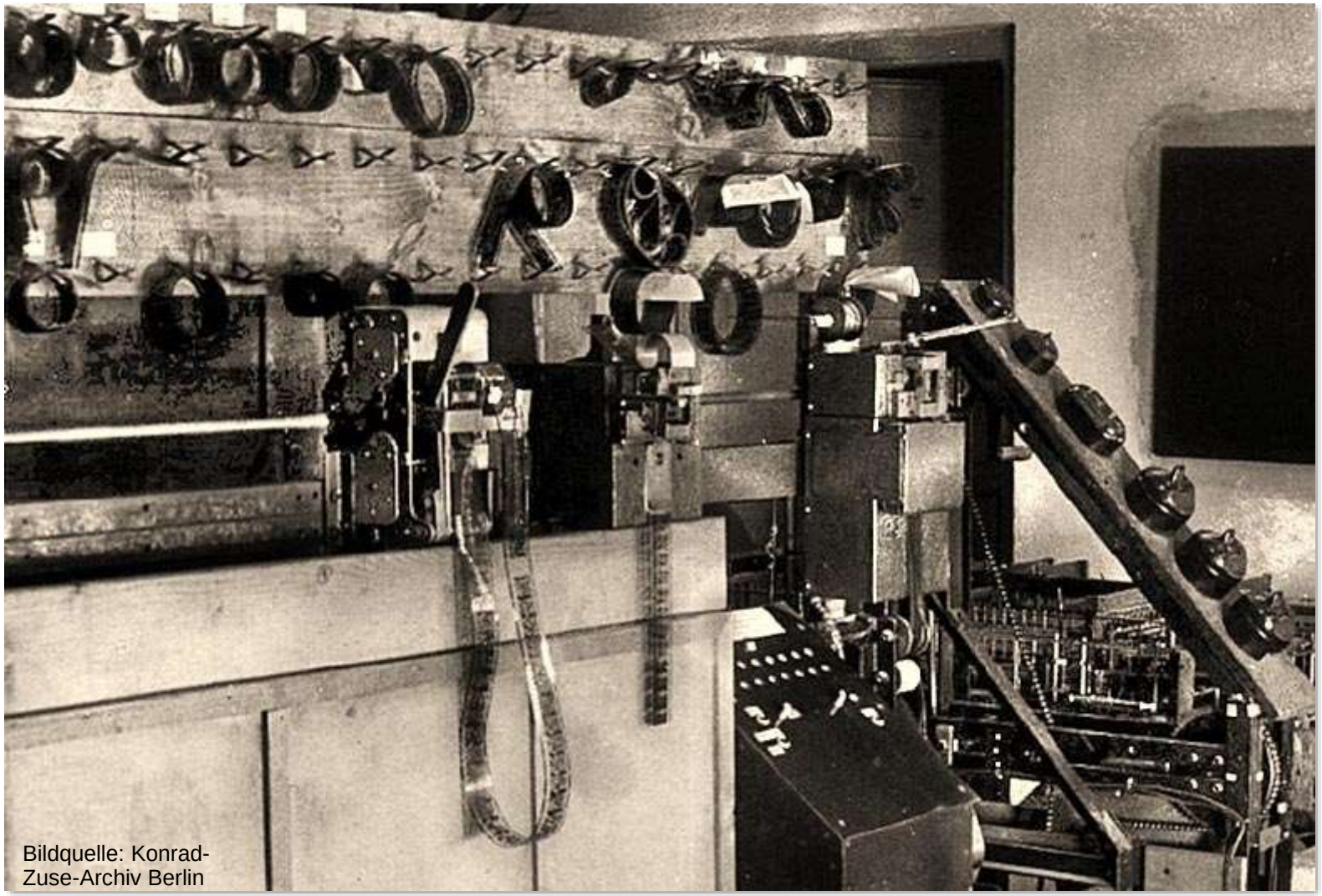
Bildquelle: www.deutsches-museum.de/en/exhibitions/communication/computers/universal-computers/

Z4: Programm auf gelochten Kinofilmstreifen



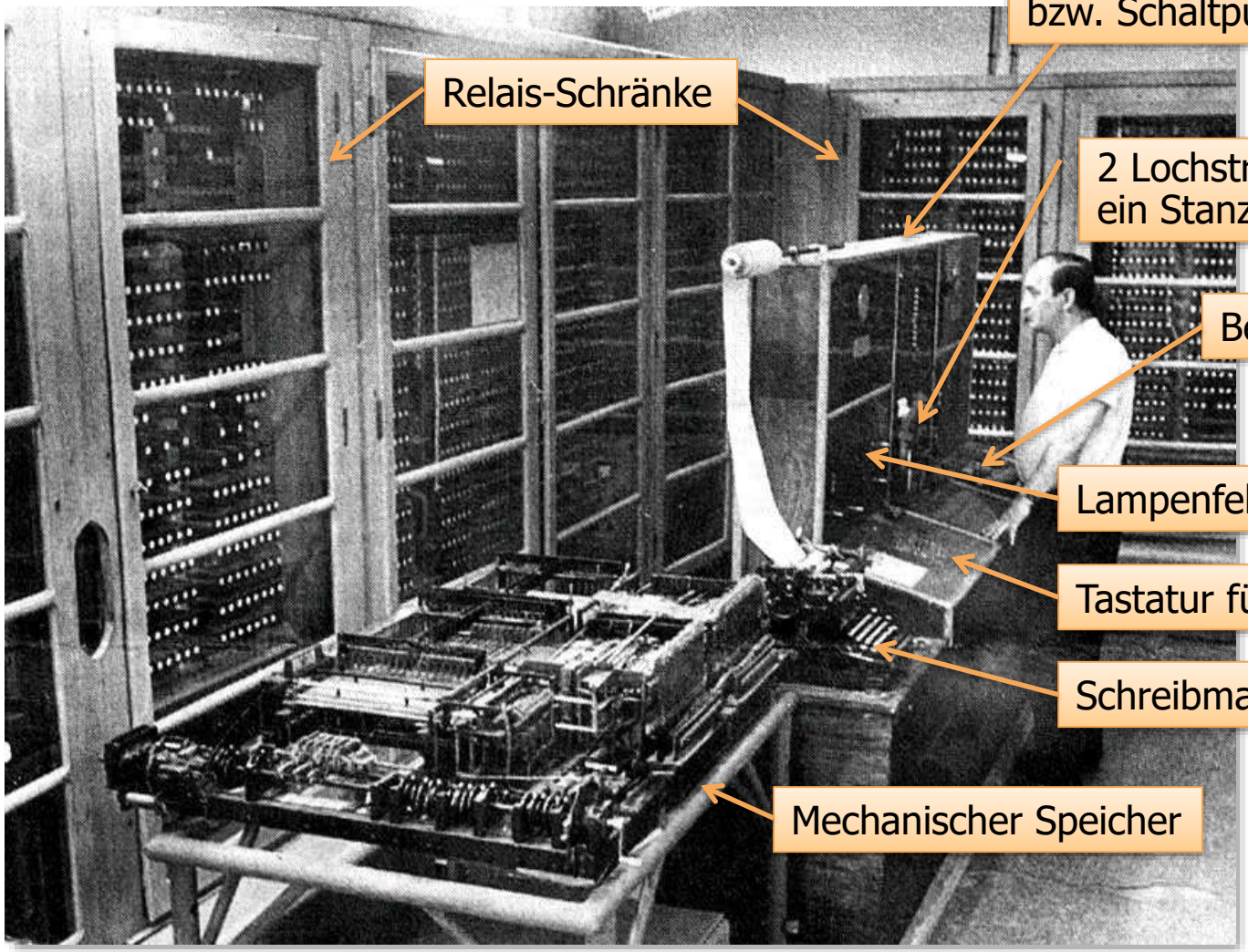
Speiser und Rutishauser an der Konsole der Z4; Film-Lochstreifenabtaster in Bildmitte

Z4: Programmlager statt Programmbibliothek



Z4-Programme, aufgehängt an Nägeln in einem Mehllager in Hopferau (Allgäu), 1949

Die Konfiguration der Z4

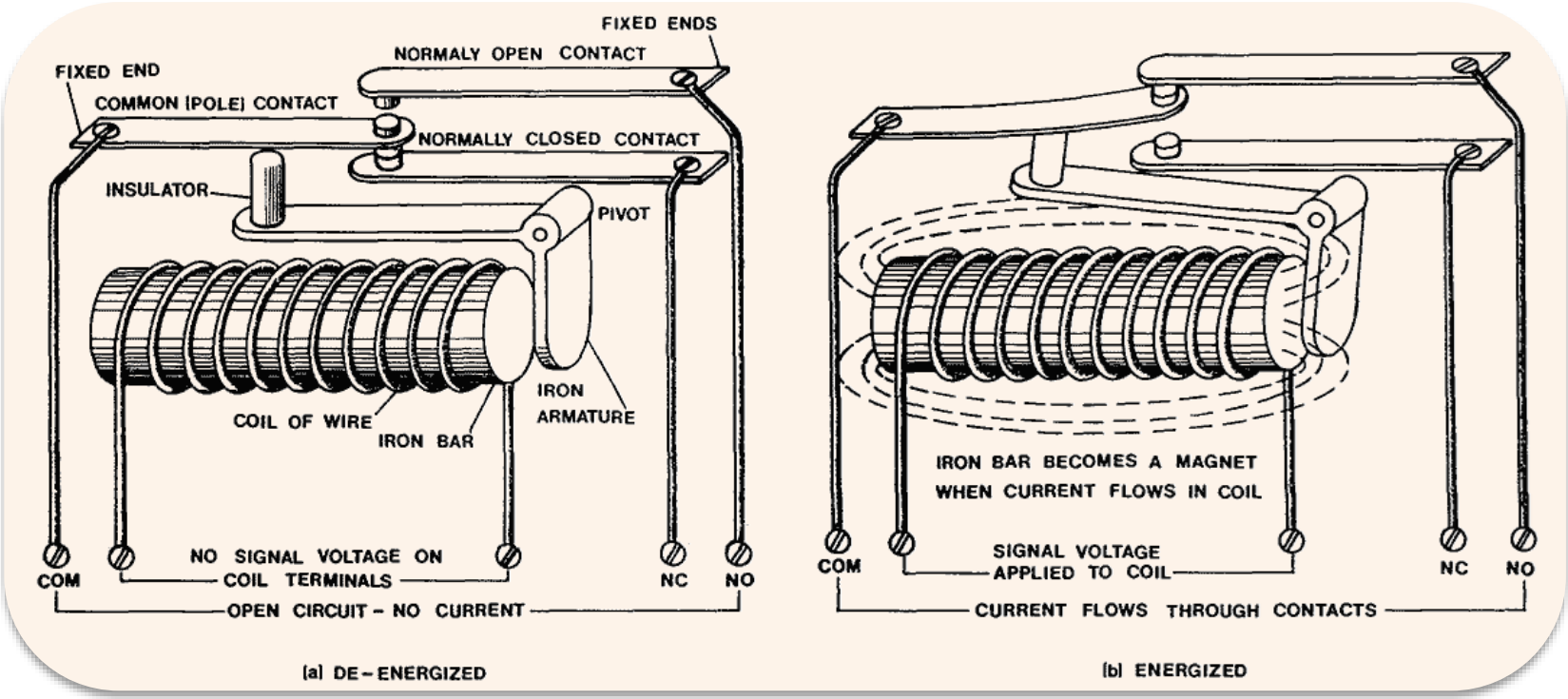


http://farm3.staticflickr.com/2208/1814569165_5348377b80_o_d.jpg

Elektromechanische Relais

Zum weiter Nachforschen: Wann und zu welchem Zweck wurden Relais erstmalig genutzt? Wie kam es zu dem Namen?

Ein Relais ist ein durch elektrischen Strom betriebener, elektromagnetisch wirkender, fernbetätigter Schalter mit in der Regel zwei Schaltstellungen. Ein Steuerstrom in der Erregerspule erzeugt einen magnetischen Fluss durch den ferromagnetischen Kern und einen daran befindlichen, beweglich gelagerten, ebenfalls ferromagnetischen Anker. An einem Luftspalt kommt es zur Krafteinwirkung auf den Anker, wodurch dieser einen oder mehrere Kontakte in Laststromkreisen schaltet. Der Anker wird durch Federkraft in die Ausgangslage zurückversetzt, sobald die Spule nicht mehr erregt ist. [Wikipedia]



Typical relay used in telephone switching. The cylinder in front is the magnet; behind are the contacts. Drawings: Edwin Collen.

Kein bedingter Sprung

- Die Z4 besass ursprünglich keinen bedingten Sprung oder bedingte Verzweigung, was im Vorfeld von den ETH-Experten kritisiert wurde:

c) Bei sehr vielen längeren Rechnungen hängt der Gang der Rechnung vom numerischen Wert der Zwischenresultate ab. Die Rechenresultate der Maschine beeinflussen also die später zu gebenden Befehle (bedingte Befehle). Die Zusesche Maschine ist dafür nicht besonders eingerichtet, d.h. besitzt keine Einrichtung, zwischen verschieden möglichen Befehlsfolgen automatisch auszuwählen. In Amerika wird dem grösstes Gewicht gegeben. Dieser Nachteil beeinträchtigt die mathematische Flexibilität der Maschine.

10. Es ist ein neuer Befehl (Teilschlusszeichen) vorzusehen, der bewirkt, dass der Streifen bis zum nächsten Startzeichen läuft, so dass die eventuell dazwischenliegenden Befehle vom Abtaster nicht abgenommen werden. Die Ausführung dieses Befehls sollte durch den Ablauf der Rechnung gesteuert werden können. (Bedingte Befehle.) Auch dieser

c) Einbau eines richtigen bedingten Befehls, der ausgelöst wird, wenn ein Rechenresultat positiv (oder auch imaginär, oder genau Null) ist, und der je nach Wahl entweder "Teilschluss" oder "Stop" oder Uebergang auf den andern Abtaster, gemäss b), bewirkt.

- Daraufhin baute Zuse vor der Auslieferung der Z4 an die ETH noch den bedingten Sprungbefehl ein

Die Z4, beschrieben von Eduard Stiefel

„Im Vordergrund steht das Schaltpult; es enthält in seinem Mittelstück zwei Abtaster und einen Locher für die Lochstreifen, links die Tastatur zum Eingeben von Zahlen und ein Lampenfeld zum Ablesen von solchen. Der rechte Teil dient der Herstellung von Programmen; durch Betätigung der unten sichtbaren Tasten werden die Befehle auf einen Filmstreifen gelocht.

Hinter der elektrisch gesteuerten Schreibmaschine ist das Speicherwerk angebracht […]. Rechts und im Hintergrund stehen die Schränke für Rechenwerk und Speicherwerk; die Z_4 verwendet als Schaltelemente gewöhnliche Telefon-Relais und Schrittschalter.

[…] zeigt die beiden Abtaster während des Durchrechnens eines mathematischen Problems. Im rechten Abtaster liegt das Hauptprogramm, auf welchem etwa ein Integrationsschritt zur Lösung einer Differentialgleichung programmiert ist. Der linke Abtaster verarbeitet ein Unterprogramm, und zwar das Ausziehen einer Wurzel auf iterativem Weg. Man beachte, daß dieses Programm eine endlose Schleife ist; ein einmaliger Umlauf desselben ergibt einen Iterationsschritt. Während die Rechnung automatisch abläuft, kann man nun folgendes Spiel beobachten. Sobald das Hauptprogramm an die Stelle kommt, wo eine Wurzel berechnet werden muß, bleibt es stehen und veranlaßt durch einen Sprungbefehl das Anlaufen des Unterprogramms. (Unbedingter Sprung.) Dieses macht so viele Umläufe, das heißt berechnet so viele immer bessere Annäherungen an den Wurzelwert, bis die Rechnung steht; dann nimmt das Hauptprogramm seine Arbeit wieder auf.“

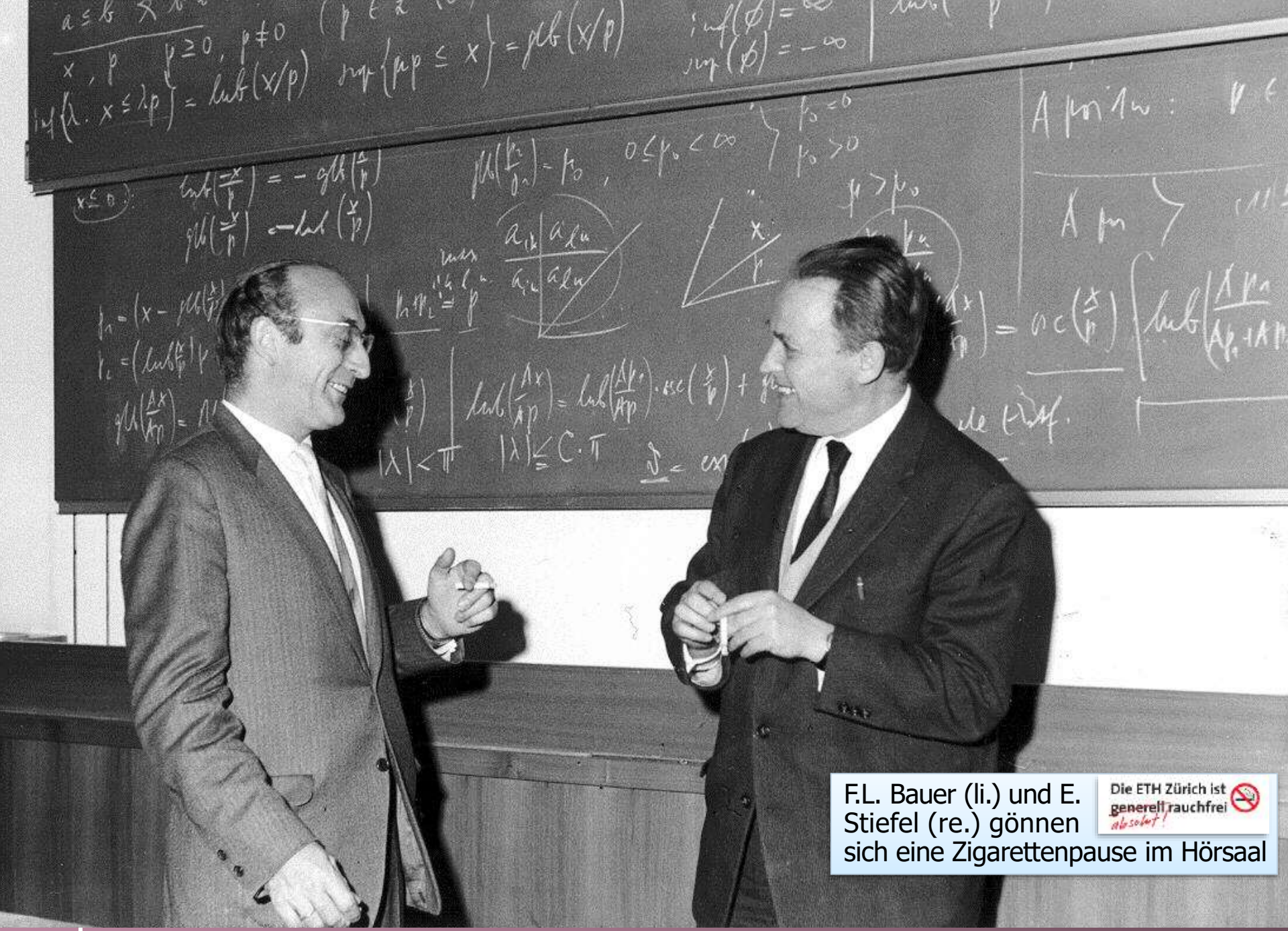
In: Rechenautomaten im Dienste der Technik. Erfahrungen mit dem Zuse-Rechenautomaten Z4. Sonderdruck aus Arbeitsgemeinschaft für Forschung des Landes Nordrhein-Westfalen, Bd. 45, Köln 1954.

Eduard Stiefel (1909 – 1978)

http://ba.e-pics.ethz.ch/#1517139612819_3

Studium der Mathematik und Physik an der ETH Zürich, Diplom 1931. Danach in Hamburg sowie Göttingen. Später Assistent von Walter Saxer an der ETH, dort 1935 Promotion bei Heinz Hopf. Ordentlicher Professor für höhere Mathematik an der ETH 1943; Leitung des 1948 von ihm gegründeten Instituts für angewandte Mathematik. Verfasste u.a. das Standardwerk „Einführung in die Numerische Mathematik".

Laut www.genealogy.math.ndsu.nodak.edu haben bei Eduard Stiefel mindestens 63 Doktoranden promoviert, darunter Ambros Speiser, Peter Henrici, Corrado Böhm, Urs Hochstrasser, Peter Läuchli, Carl August Zehnder und Jörg Waldvogel.



F.L. Bauer (li.) und E. Stiefel (re.) gönnen sich eine Zigarettenpause im Hörsaal

Die ETH Zürich ist generell rauchfrei absolut!

Eduard-Stiefel-Weg ✕ John-von-Neumann-Weg

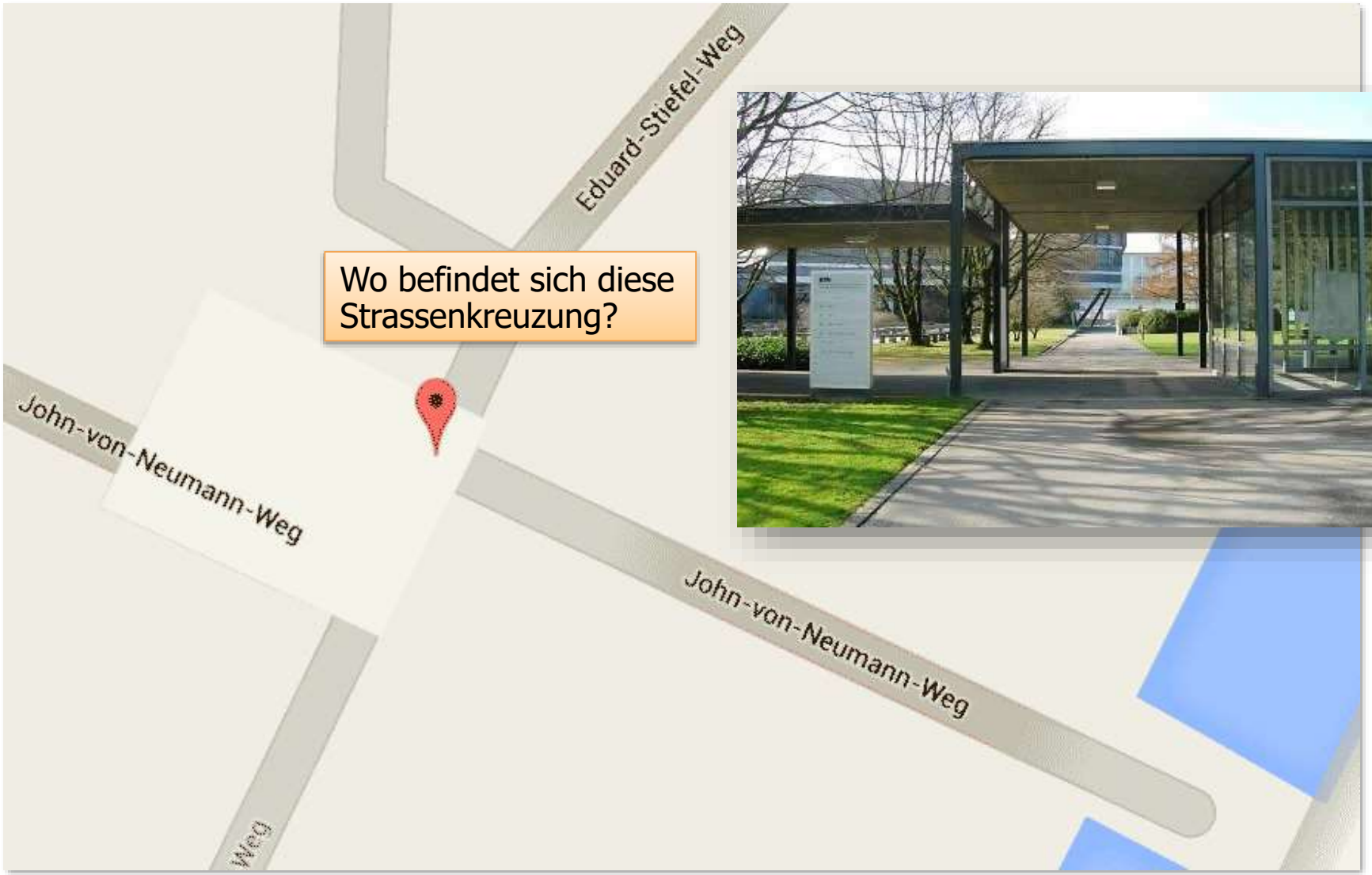
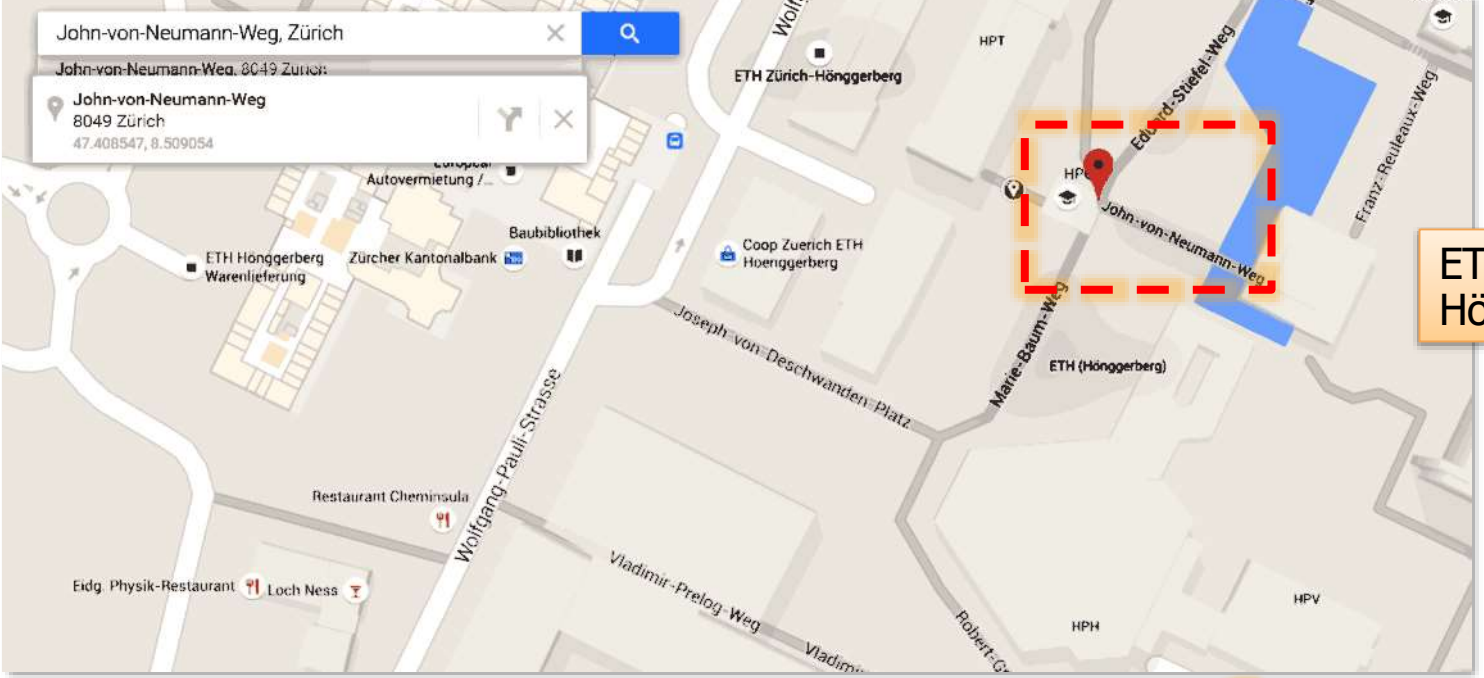
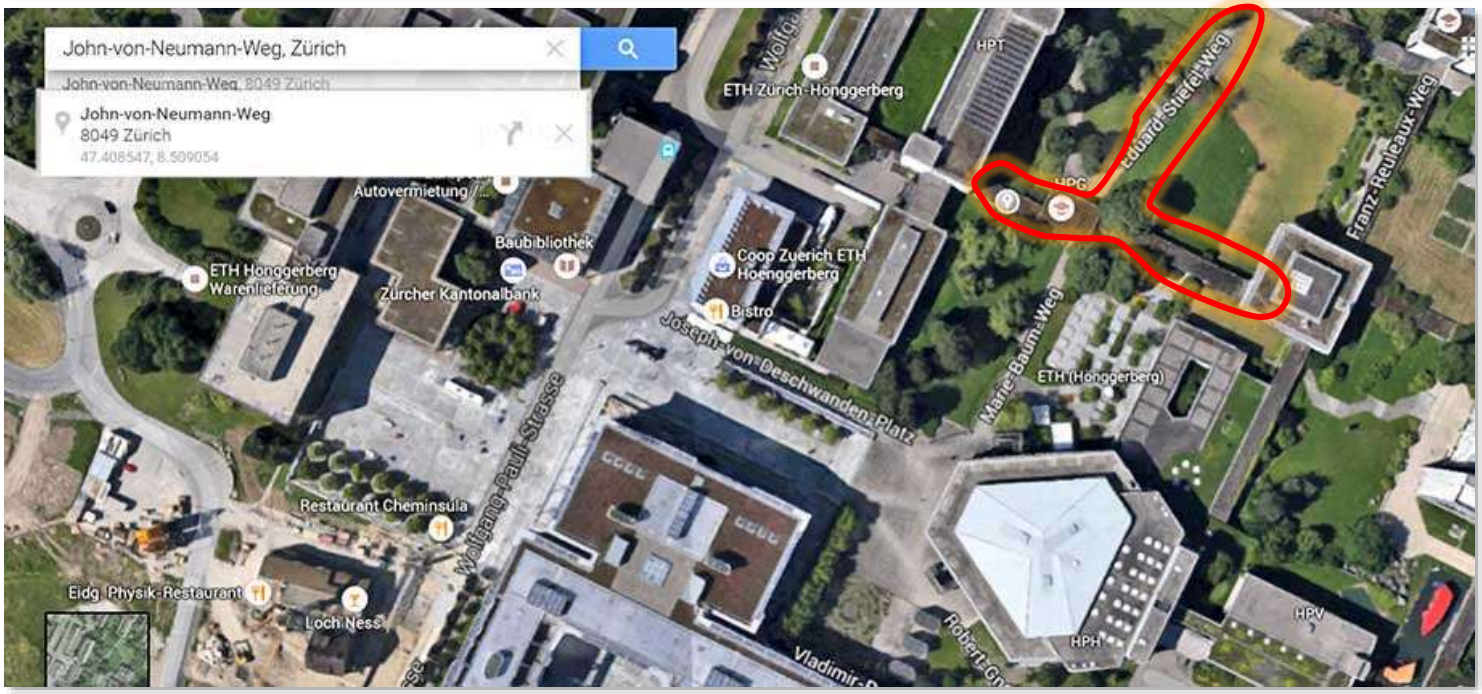


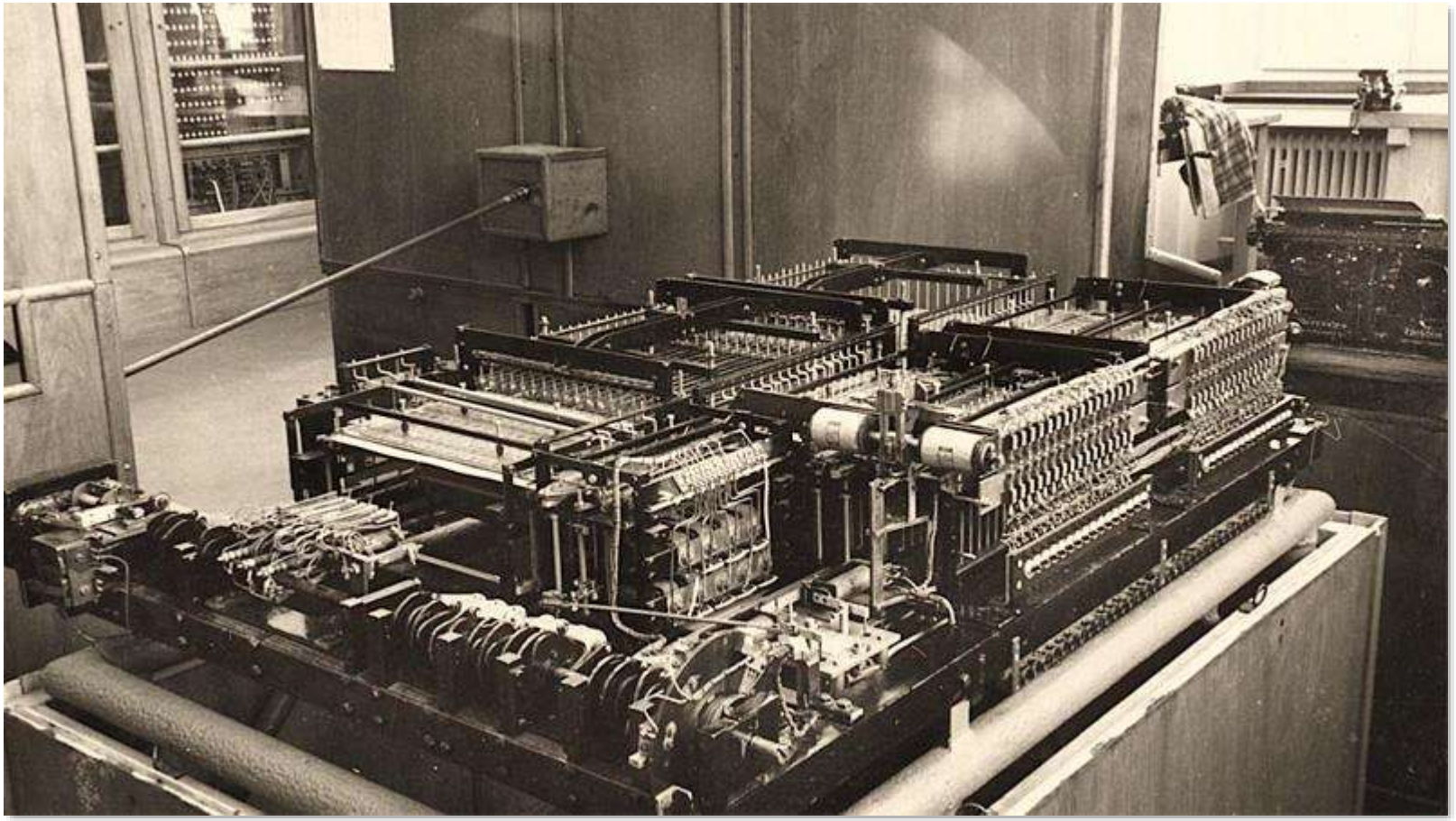
Foto: www.alt-zueri.ch/turicum/strassen/j/john_von_neumann_weg/john_von_neumann_weg.html, Bildarchiv Dürst, Zürich (cc)



ETH Zürich, Hönggerberg



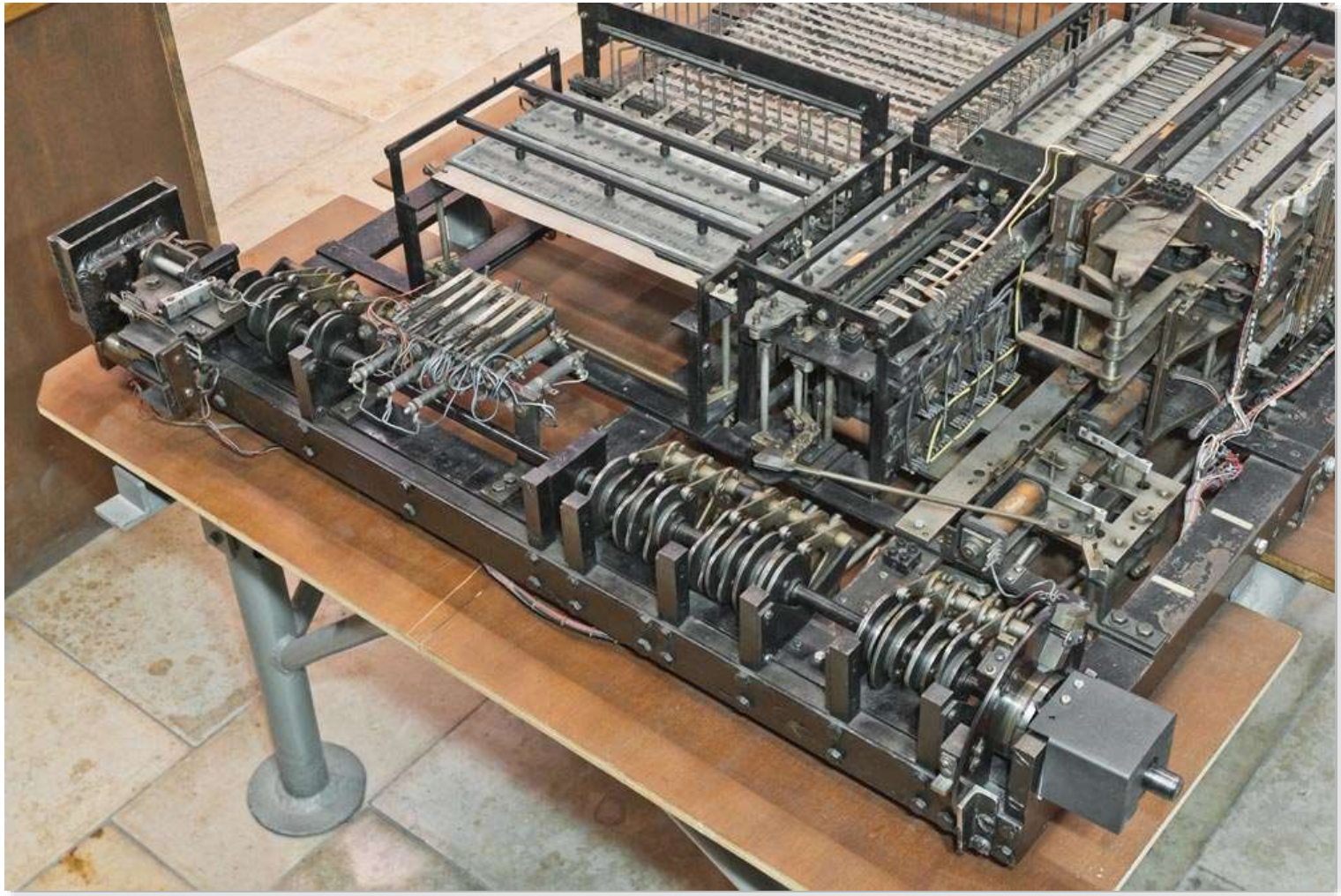
Der mechanische Speicher der Z4



Vorne links die mechanische Antriebswelle, mitte links das Wählwerk (zum Auswählen der gewünschten Speicherzelle), hinten links der eigentliche Speicher (2048 Bit), rechts die Einstell- und Ablesevorrichtung

www.digitalbrainstorming.ch/weblog/Abb.%201%20Mechanischer%20Speicher%20Z4-pano.jpg
www.e-pics.ethz.ch/index/ETHBIB.Bildarchiv/ETHBIB.Bildarchiv_Ans_03676_8398.html

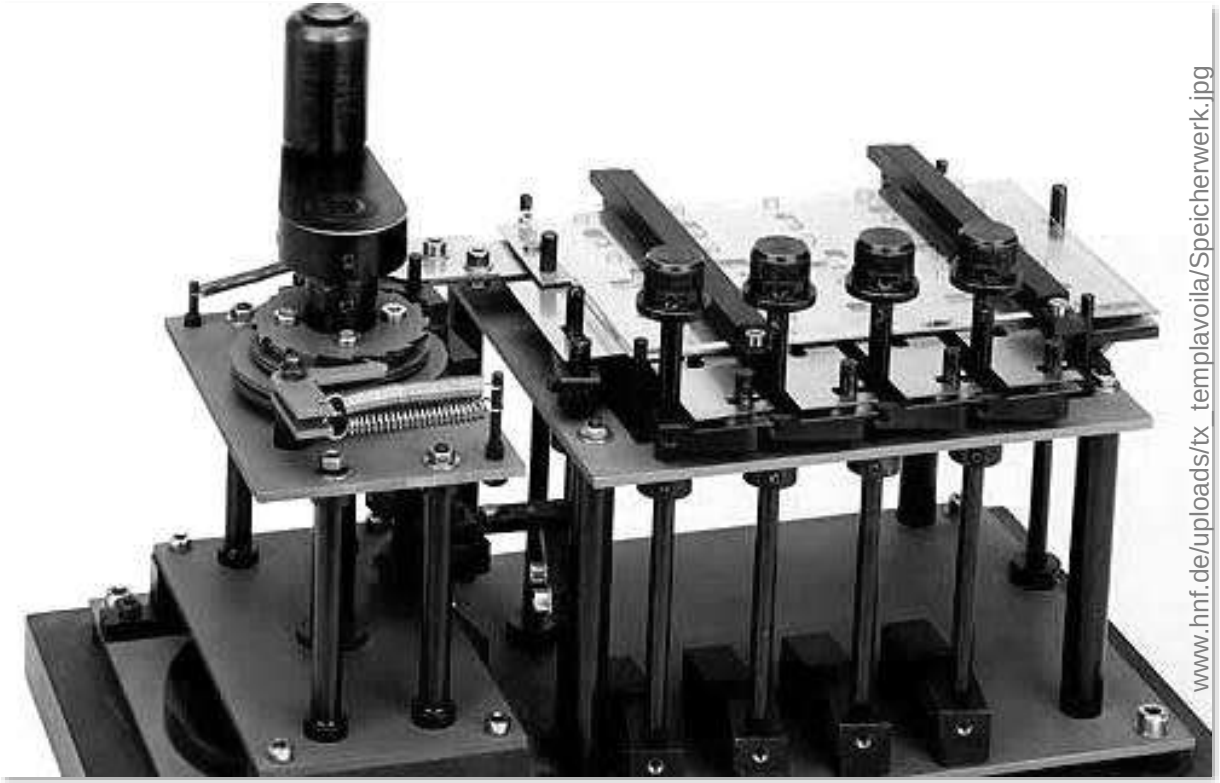
Der mechanische Speicher der Z4 (2)



Aufbau im Deutschen Museum, München

www.deutsches-museum.de/en/exhibitions/communication/computers/universal-computers

Ein Teil des mechanischen Speichers der Z4



www.hnf.de/uploads/tx_templavoila/Speicherwerk.jpg

„Am Dienstag, dem 11. Juli 1950, wurde eine Sendung von schweren Gestellen und Kästen, die mit einem Eisenbahnwagen von Deutschland gekommen war, in die ETH getragen... Ein Stück dieser Sendung war absonderlich in jeder Beziehung: Ein Apparat, dicht gepackt mit Hebeln, Blechen, Federn, und gefüllt mit etwa 3000 präzis gearbeiteten Stahlstiften. Es ist anzunehmen, dass ausserhalb Deutschlands nicht mehr als ein halbes Dutzend Menschen in ihrem Leben je etwas auch nur entfernt Ähnliches gesehen hatten. Die Sendung war der Relais-Rechner Z4 des deutschen Computer-Pioniers Konrad Zuse, der seltsame Apparat war der mechanische Speicher. Mit diesem Tag nahm die Informatik an der ETH ihren Anfang."

Ambrosius Speiser: 95 Semester ETH – der Weg zur Informatik, ETH Zürich, 1992

Zuhören beim Programmablauf

Durch genaues Zuhören bekam man manche Aufschlüsse über den Programmablauf. Deutlich konnte man das Ticken des Programmabtasters, das Klappern der Relais im Rechenwerk und das Klirren der Speicheroperationen unterscheiden. Mit einiger Übung konnte man zu jeder Zeit sagen, ob eine Addition, eine Multiplikation oder eine Division im Gang war. Längere Umspeichervorgänge oder Sprünge – also das Überspringen einer Folge von Befehlen – waren sofort erkenntlich. Ein Blick durch die Glasfront auf zwei Schrittschalter half, Divisionen von Quadratwurzeln zu unterscheiden. Ein Steckenbleiben des Programmes – etwa wegen falscher Befehlsfolgen – war sofort hörbar.

Ambrosius Speiser: Die Z4 an der ETH Zürich: ein Stück Technik- und Mathematikgeschichte. In: Elemente der Mathematik 36 (1981), Heft 6, Seiten 145-176.

Konrad Zuse

- Konrad Zuse (1910–1995) schloss 1935 sein Ingenieurstudium in Berlin ab. Danach arbeitete er zunächst als Statiker in der Flugzeugindustrie, gab diese Stelle jedoch bald auf und richtete eine Erfinderwerkstatt in der elterlichen Wohnung ein.
- Mit seiner Entwicklung der Z3 im Jahre 1941 baute er den ersten vollautomatischen, programmgesteuerten und frei programmierbaren, in binärer Gleitkommarechnung arbeitenden Computer.
- Zu Kriegsende 1945 Flucht aus Berlin via Göttingen in das Allgäu, wobei er den zuletzt entstandenen Rechner Z4 retten konnte.
- Am 13. Juli 1949 besuchte Prof. Eduard Stiefel von der ETH Zürich Zuse im Allgäu und liess sich die Z4 vorführen. Durch ihn kam ein Mietvertrag mit der ETH zustande, der Zuse die notwendigen Mittel verschaffte, um die Zuse KG zu gründen.
- Zuse erhielt 1991 die Ehrendoktorwürde der ETH Zürich.

Quelle u.a. Wikipedia

K. Zuse: „Die Rechenmaschine des Ingenieurs“
(Unveröffentlichtes Manuskript, 30.1.1936, Auszug)

Die Anforderungen, die an die ideale Rechenmaschine des Ingenieurs gestellt werden müssen, gehen aus den folgenden Überlegungen hervor:

Der Ingenieur hat viel mit festen Formeln zu arbeiten, die immer wiederkehren. Man hat gewisse Ausgangswerte, und die Arbeit besteht nun darin, durch bestimmte, für eine Formel immer gleiche Aufeinanderfolge von Grundrechenarten zwischen bestimmten Zahlen das Resultat zu berechnen. [...]

In dieser Art läßt sich für jede beliebig lange Rechnung ein „Rechnungsplan“ aufstellen, indem im voraus die aufeinanderfolgenden Rechenoperationen dem Charakter und der Reihe nach aufgezeichnet werden und die im Verlauf der Rechnung auftretenden Zahlen fortlaufend numeriert, oder nach einem anderen Schema geordnet werden. [...]

Der Ingenieur braucht Rechenmaschinen, die diese Rechenoperationen automatisch ausführen, indem der Rechenplan auf einem Lochstreifen festgehalten wird, der die Befehle für die einzelnen Rechenoperationen selbsttätig und nacheinander an die Maschine gibt.

Die Maschine muß auf „Befehl“ des Lochstreifens jede verlangte Grundrechnung vollautomatisch ausführen. Ferner muß die Maschine über ein Speicherwerk verfügen, in welchem die während der Rechnung auftretenden Zahlen der Nummer nach geordnet werden können, und aus denen durch ein mechanisches Wählwerk jede gewünschte Zahl abgelesen werden kann. Das Speichern und Ablesen der Zahlen wird ebenfalls durch den Befehlslochstreifen dirigiert. [...]

Pläne von allgemeiner Bedeutung entsprechen den heutigen Formelsammlungen und gehören zum dauernden Planbestand eines Büros. [...] Ein bestimmter Einzelplan kann seine Ausgangswerte von verschiedenen anderen Plänen beziehen und seine Resultatwerte weitergeben. [...] Die Beziehungen der Pläne untereinander müssen wie elektrische Stecker aneinander passen.

„Mechanisierung schematischer Denkaufgaben“
(K. Zuse in einem Informationsblatt von 1947)

Die von Dipl. Ing. Konrad Zuse begonnene Gerätentwicklung ermöglicht es, über das Rechnen mit Zahlen hinausgehend, das gesamte Gebiet der schematischen kombinatorischen Denkoperationen zu mechanisieren.

Wir erkennen, welches große Aufgabengebiet der maschinellen Lösung erschlossen wird, wenn wir uns vergegenwärtigen, dass eine der wichtigsten geistigen Funktionen die Auswertung irgendwelcher gegebener Angaben, wie Beobachtungen, Daten usw. ist, welche man z.B. nach bestimmten Gesichtspunkten ordnet, aus ihnen nach gewissen Vorschriften Schlüsse zieht, und auf diese Weise durch Ausführen „kombinatorischer Denkaufgaben“ zu folgerichtigen Ergebnissen kommt.

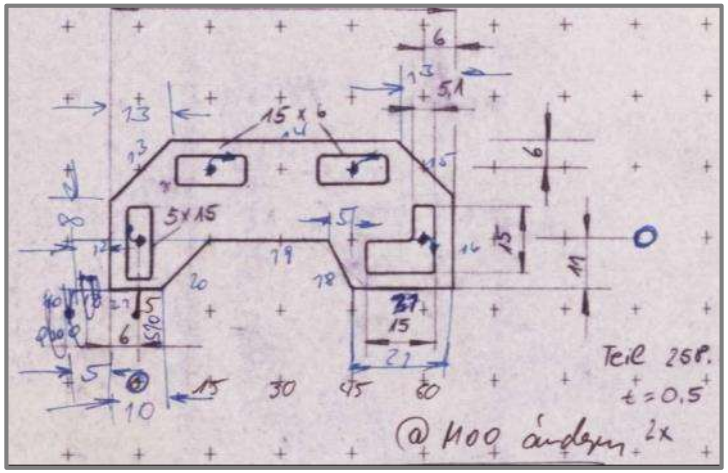
Zuse hat mit der von ihm begonnenen Entwicklung den Weg zur Konstruktion von Geräten beschritten, die nach den Anweisungen einer allgemeinen mathematischen „Zeichensprache“ rechnen können. Diese Zeichensprache ist auf kein bestimmtes Anwendungsgebiet spezialisiert, indem sie z.B. die Gesetzmäßigkeiten der Addition und der anderen Operationen der Zahlenrechnung mit den gleichen Mitteln darstellt, wie etwa die Bewegungsgesetze der Figuren des Schachspiels oder die Gedankengänge beim Entwurf einer Brücke.

Wie der Begriff „Rechnen“ in der Umgangssprache auch auf Operationen angewendet wird, die mit Zahlen nichts zu tun haben brauchen, indem wir z.B. sagen, dass wir mit dem Eintreten dieser und jener Ereignisse „rechnen“, so ist auch der Aufgabenbereich der Zuse-Rechenmaschinen ein allgemeiner, in welchem die Zahlenrechnung zwar eine große Bedeutung hat, sie aber nur einen Teil der mechanisierbaren schematischen Denkaufgaben darstellt. [...]

Mit der Mechanisierung schematischer geistiger Arbeiten auf breiter Grundlage beginnt ein neuer Abschnitt der Technik. Die in diesem Zusammenhang auftretenden Probleme sind so umfangreich, dass eine Generation von Wissenschaftlern, Technikern und Wirtschaftlern erforderlich sein wird, um sie erschöpfend zu bearbeiten.

Vor der Z4: Die Z1, Zuses erste Maschine (1936 – '38)

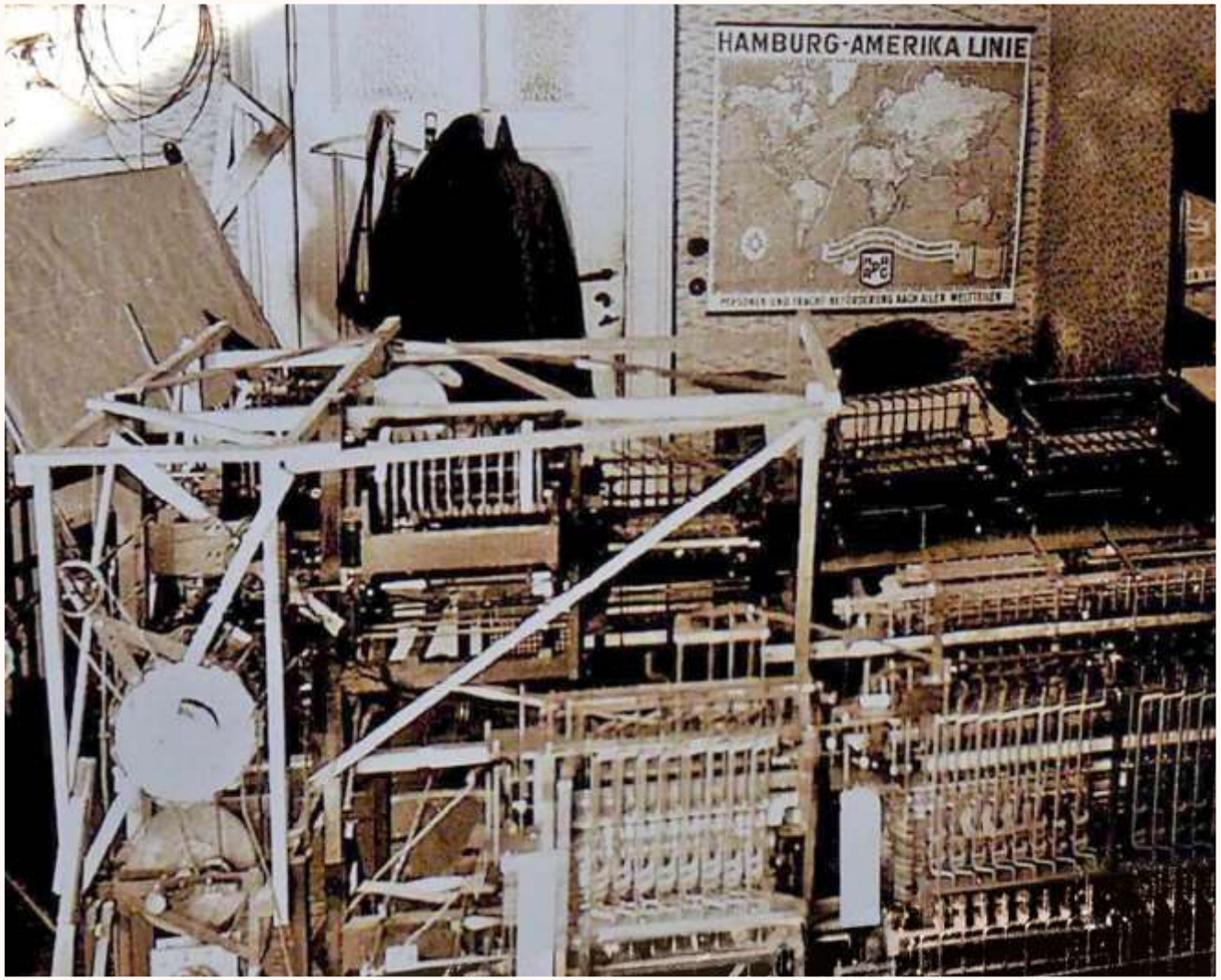
- Motivation: Automatisierung zeitaufwändiger baustatischer Berechnungen (Zuse studierte bis 1935 Bauingenieurswesen in Berlin)
- Merkmale der Z1 als Vorläufer moderner Computer
  - Bistabile mechanische Schaltglieder in Form beweglicher Metallplatten
  - Antrieb durch Staubsaugermotor
  - Von-Neumann-Architektur
  - Zahlendarstellung: dual und Gleitpunkt
  - Automat. Umwandlung Dezimal → Dual
  - Binär verschlüsselte 8-Bit-Befehle
  - Programme von gelochten Filmstreifen
  - Kein bedingter Sprungbefehl
  - Taktfrequenz: 1Hz (→ Multiplikation ca. 20s)
  - Ca. 30000 Bauteile (Weichbleche)
  - Mechanischer Speicher: 16 24-Bit-Worte (ähnlich zum Speicher der Z4)



Bauteilskizze von Zuse (http://zuse-z1.zib.de)

- Rechenwerk unzuverlässig (Verhaken mechanischer Schaltglieder)
  - Spätere Z3 (1941) analoge Architektur, aber elektromechanisch mit Relais

Die Z1 im Wohnzimmer der Eltern



https://heureka-stories.de/images/Zuse/Zuse-Wohnzimmer.JPG

„Die Leistungen Konrad Zuses sind deshalb besonders hoch einzuschätzen, weil er aus eigenen Mitteln, in der elterlichen Wohnstube seine grundlegende Entwicklung durchführte, zeitweise noch zum Militärdienst eingezogen wurde und trotzdem nicht verzagte, bis er schließlich die Realisierungsmöglichkeit programmgesteuerter Rechenautomaten mit seinem funktionierenden Modell nachgewiesen hatte.“ [Karl Steinbuch]

„Wer vor der Maschine steht, kapituliert auch heute noch beim Versuch, die Funktionsweise von mehreren Hundert Blechen, Stangen und Rädern allein durch Anschauung verstehen zu wollen. Die Z1 wirkt wie ein versteinertes, unergründliches Uhrwerk.“
[Rojas; Röder; Nguyen]

Z1-Nachbau

Nachbau der Z1 (Bild: Raúl Rojas) von Zuse selbst vorgenommen; das Original wurde bei einem Bombenangriff auf Berlin im Dezember 1943 zerstört. Rechts die acht Speicherebenen, links die zwölf Ebenen für den Prozessor. Im unteren Teil das Hebelwerk, das den Takt an alle Maschinenkomponenten weiterleitet. „Im einzigen überlebenden Video der laufenden Maschine erweckt dies den Anschein von beweglichen Teilen eines Webstuhls. Allerdings webte diese Maschine Zahlen“ [Rojas]

Dadurch, dass Zuse das Dualsystem verwendete, konnte er im Unterschied zu den Konstrukteuren klassischer mechanischer Rechenmaschinen (wie Leibniz oder Babbage) auf Zahnräder und Getriebe zugunsten einfacherer bistabiler Schalt- und Speicherelemente verzichten.

Konrad Zuse

„Es ist für uns heute noch erstaunlich, wie dem jungen Diplomingenieur Konrad Zuse 1936 eine solch moderne Konstruktion gelang. Die Erbauer der ENIAC waren eine Schar von Ingenieuren, darunter gestandene Professoren. Was sie erschufen ist aber im Vergleich zur Z1 ein Dinosaurier. Die Z1 ist, trotz der mechanischen Ausführung, eine ausgereiftere Konstruktion als vieles, was bis 1945 gebaut wurde. Noch schlankere Rechnerarchitekturen wurden erst beim Nachfolger der ENIAC, der EDVAC, konzipiert, aber da hatte schon ein John von Neumann seine Finger im Spiel. Er lebte von 1926 bis 1929 in Berlin und war der jüngste Privatdozent an der Berliner Universität. Vielleicht sind der Erstsemestler Zuse und der Privatdozent sich irgendwann über den Weg gelaufen?"

Raúl Rojas, Julian Röder, Hai Nguyen: Die Prozessorarchitektur der Rechenmaschine Z1. Informatik-Spektrum (2014) 37: 341-347

Bild: http://people.idsia.ch/~juergen/zuse4.jpg

Wer erfand den Computer?

- Das hängt davon ab, was man unter „Computer“ versteht:

- Funktionsfähig oder nur Konzept (wie z.B. Babbages Analytical Engine)?
- Digital (binär, dezimal,...) oder auch analog (→ „ungenau“!)?
- Mechanisch, elektromechanisch, elektronisch (Röhren, Transistoren)?
- Universalrechner („all purpose computer“), d.h. frei programmierbar und Ausführung beliebiger Algorithmen im Rahmen des verfügbaren Speicherplatzes (sogen. „Turing-Vollständigkeit“) oder konstruiert speziell für eine bestimmte Aufgabenklasse?
- Programm: Auf (auswechselbarem) Steckbrett, auf Lochstreifen (als Nur-Lese-Speicher) oder zur Laufzeit im Speicher und damit im Prinzip durch Software selbst veränderbar (Speicherprogrammierung: Programm = Daten, Von-Neumann-Prinzip)?

OPREMA-Computer mit Programmsteckta-feln (Carl Zeiss Jena)

Wer erfand den Computer?

- Das hängt davon ab, was man unter „Computer“ versteht
  - Tabelle bei https://de.wikipedia.org/wiki/Zuse_Z3 [März 2018]:

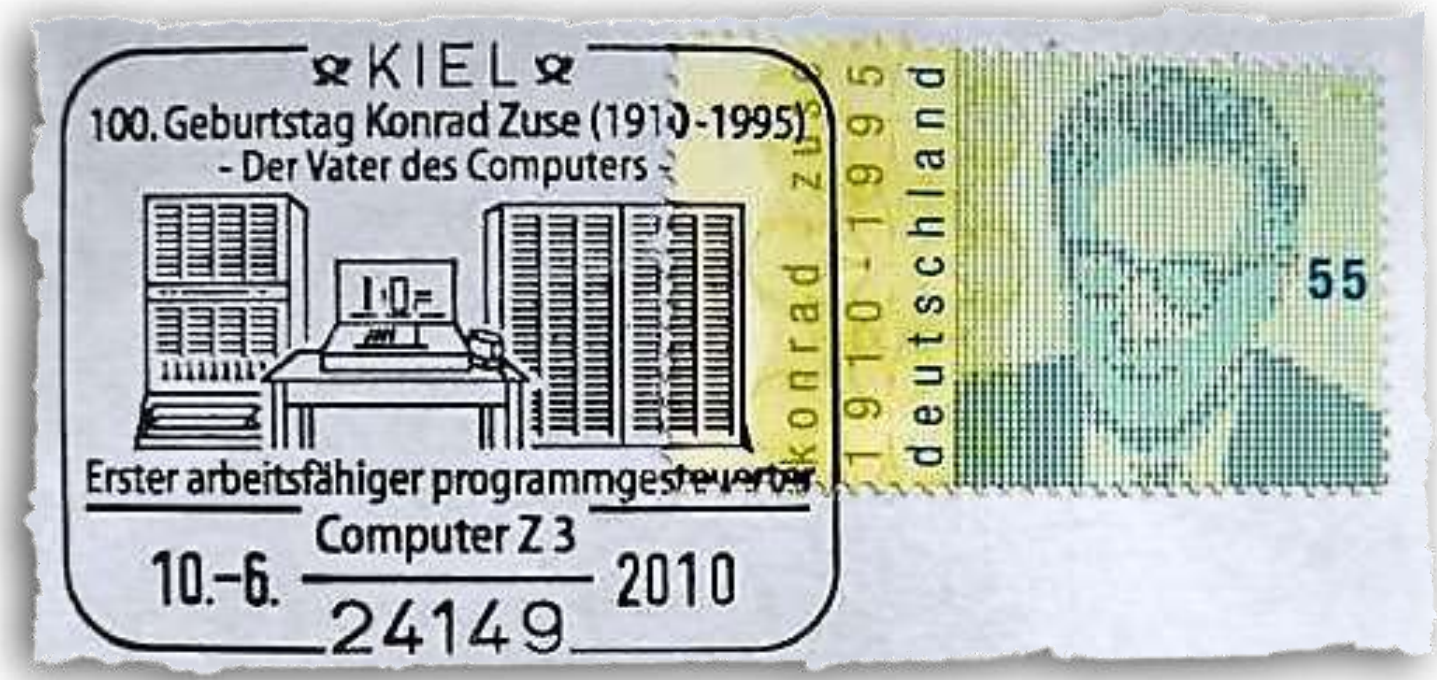
Computer	Land	Inbetrieb-nahme	Gleitkomma-arithmetik	Binär	Elektro-nisch	Programmierbar	Turing-mächtig
Zuse Z3	Deutschland	Mai 1941	Ja	Ja	Nein	Ja, durch Lochstreifen	Ja, ohne Praxisnutzen
Atanasoff-Berry-Comp.	USA	Sommer 1941	Nein	Ja	Ja	Nein	Nein
Colossus	UK	1943	Nein	Ja	Ja	Teilweise, durch Neuverkabelung	Nein
Mark I	USA	1944	Nein	Nein	Nein	Ja, durch Lochstreifen	Ja
Zuse Z4	Deutschland	März 1945	Ja	Ja	Nein	Ja, durch Lochstreifen	Ja
ENIAC	USA	1946	Nein	Nein	Ja	Teilweise, durch Neuverkabelung	Ja
		1948	Nein	Nein	Ja	Ja, durch Wider-standsmatrix	Ja

“Any ‘first’ is problematical. What ‘first’ means depends on precise definitions of fuzzy concepts. Historian of computing and former Computer History Museum chief curator Michael Williams famously said that anything can be a first if you put enough adjectives before the noun. Does the ‘first computer’ need to be electronic? Have a program stored in memory? Have been built? Be general-purpose?”

https://computerhistory.org/blog/programming-the-eniac-an-example-of-why-computer-history-is-hard/

Wer erfand den digitalen Universalcomputer?

- Zuse: Z3 (am 21. Dez. 1943 durch Bombentreffer zerstört; funktionsfähiger Nachbau von 1962 im Deutschen Museum); elektromechanisch
  - Patentanmeldung von 1941 durch Zuse, Ablehnung durch das deutsche Bundespatentgericht „mangels Erfindungshöhe“: Die Bauteile wie Relais und Schrittschalter seien bereits vorhanden gewesen, Zuse habe sie einfach auf eine neue Art zusammengebaut



Wer erfand den digitalen Universalcomputer?

- Zuse: Z3 (am 21. Dez. 1943 durch Bombentreffer zerstört; funktionsfähiger Nachbau von 1962 im Deutschen Museum); elektromechanisch
  - Patentanmeldung von 1941 durch Zuse, Ablehnung durch das deutsche Bundespatentgericht „mangels Erfindungshöhe“: Die Bauteile wie Relais und Schrittschalter seien bereits vorhanden gewesen, Zuse habe sie einfach auf eine neue Art zusammengebaut
- 1944: Mark I (bzw. ASCC – Automatic Sequence Controlled Computer) durch Howard Aiken; elektromechanisch
- 1946: ENIAC (Electronic Numerical Integrator and Computer) durch John Presper Eckert und John William Mauchly; elektronisch (Elektronenröhren)

- 1973 wurde durch ein umstrittenes, aber unangefochtenes US-Gerichtsurteil in einem Patentstreit John Atanasoff zum Erfinder des ersten automatischen elektronischen digitalen Computers bestimmt, er konstruierte 1937 – 1941 eine nicht programmierbare Maschine zum Lösen linearer Gleichungssysteme (Atanasoff-Berry-Computer, „ABC“)
- Colossus (1943, Max Newman und Tommy Flowers) war ebenfalls kein Universalrechner, sondern speziell zur Dechiffrierung von geheimen Nachrichten des deutschen Militärs im Zweiten Weltkrieg konstruiert

Höhere Mathematik auf Knöpfen

DER SPIEGEL , 7. Juli 1949 (Auszug)

Gewerbeausweis vom Nov. 1942
(Deutsches Museum, München)

Als er sich als Berliner TH-Student mit langen statischen Berechnungen herumquälte, kam ihm der Gedanke, „die geistigen Kräfte des Menschen zu verstärken, indem Maschinen zur Lösung von Aufgaben herangezogen werden, die bisher einen großen Teil der geistigen Arbeitskraft gebunden hatten".

„Ich war zu faul zum Berechnen", interpretiert der lange Mann in den geflickten Mechanikerhosen heute seinen eigenen Broschürentext.

Aus 15 000 Meter Draht, 20 000 Lötstellen, 2000 Telephonrelais, 2500 kg Material, 250 000 Mark und 100 000 Arbeitsstunden entstand ein Gerät, das einen normalen Möbelwagen bequem ausfüllt. In diesem Möbelwagen fährt V 4 jetzt nach Hünfeld (Kreis Fulda). Dort soll sie der deutschen Wissenschaft rechnen helfen.

Originalname der Z4

In den USA existieren ähnliche Geräte wie V 4 unter dem Namen „Maschinen-Gehirn". Sie arbeiten aber auf anderer Basis. Zuse ersetzte 18000 US-Elektronenröhren durch mechanische Einrichtungen. Die amerikanische Maschine kostet 400000 Dollar, wiegt 20 Tonnen und hat Stromlinienform.

Auf den 149 Knöpfen und Tasten des V-4-Kommandostandes spielt Zuse höhere Mathematik. Daneben hängen unzählige verschieden gelochte Filmstreifen. Jede Lochung bedeutet eine mathematische Formel, von der einfachsten Wurzel bis zur schwierigsten Gleichung mit zehn Unbekannten.

Höhere Mathematik auf Knöpfen

Um eine Rechenaufgabe zu lösen, wird der entsprechende Streifen in V 4 eingehängt. Paulas flinke Hände tippen auf der Tastatur die jeweiligen Zahlen, die je nach Bedarf von der V 4 addiert, subtrahiert, multipliziert, dividiert oder gewurzelt oder alles auf einmal werden.

Wenn die 60 Volt Gleichstrom durch die 20 Kilometer Draht gejagt sind, leuchtet das Ergebnis hinter weißen Glasscheiben auf. Eine schwierige Berechnung, zu der ein Stab von Wissenschaftlern Tage braucht, löst V 4 in Minuten.

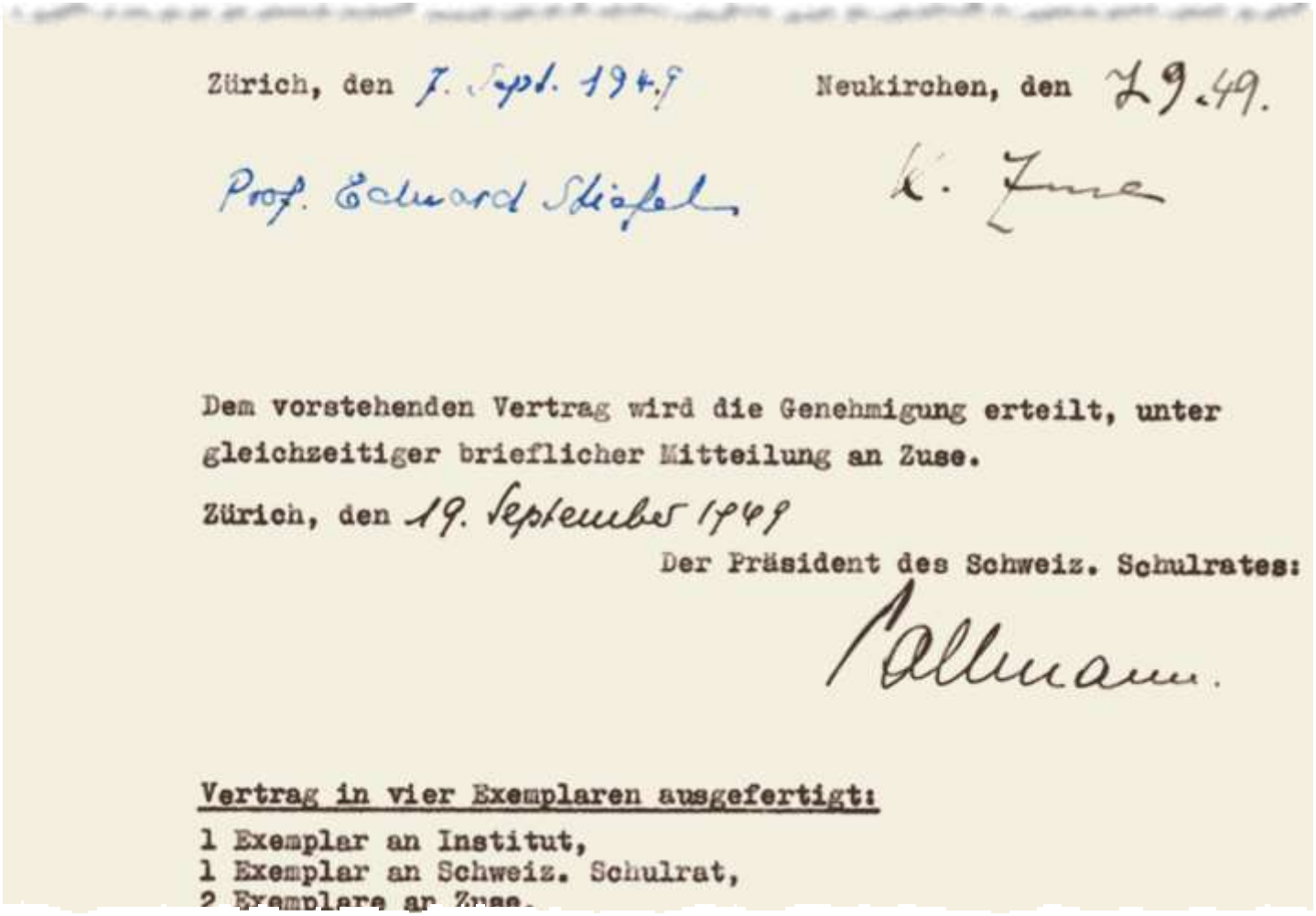
Zur V 4 gehört noch ein „mechanisches Gedächtnis" in der Größe eines mittleren Kleiderschrankes. Dessen Einzelteile schnitt Zuse aus US-Konservenblech. Das Gedächtnis „notiert" automatisch Zwischenergebnisse.

Konrad Zuse braucht 20 000 DM, um der V 4 ein New-Look-Kleid zu geben. Im Augenblick sieht sie aus wie eine Großstadttelephonzentrale nach einem Erdbeben. Außerdem sucht er Geldgeber, um den Serienbau der Rechen- und der Problemmaschine zu beginnen. „20 000 DM zum Weiterexperimentieren täten es auch schon."

Das Ausland zeigt sich an V 4 interessiert. Aber Zuse wartet ab.

„Eines Tages im Jahr 1949 entsteigt ein Herr einem vornehmen Automobil mit Schweizer Nummer. Professor Eduard Stiefel ist Vorsteher des neugegründeten Instituts für Angewandte Mathematik der ETH in Zürich, er hat von der Rechenmaschine Wind bekommen, die irgendwo im Allgäu herumstehen soll. Stiefel diktiert Zuse eine Differentialgleichung, Zuse knipst das Programm auf einen Filmstreifen, legt ihn in den Leser, die Z4 klappert und rattert, das Resultat ist richtig." [Emil Zopfi in*: Zuse Z4, der Computer aus der Asche*, http://zopfi.ch/zusez4/]

Der Mietvertrag mit der ETH wurde am 7.9.1949 im Bahnhofbuffet des Badischen Bahnhofs in Basel unterzeichnet; ein ETH-Schulratsprotokoll vom Okt. 1949 notiert: „Im Juli 1949 erfuhr die Kommission von einer Rechenmaschine des deutschen Ingenieurs Zuse, die von der ETH zu aussergewöhnlich günstigen Bedingungen übernommen werden konnte. Prof. Stiefel und Dr. Lattmann besichtigten die Apparatur. Sie rühmten besonders deren mathematische Disposition…"



Zürich, den 7. Sept. 1949 Neukirchen, den 29.49.

Prof. Eduard Stiefel K. Zuse

Dem vorstehenden Vertrag wird die Genehmigung erteilt, unter gleichzeitiger brieflicher Mitteilung an Zuse.

Zürich, den 19. September 1949

Der Präsident des Schweiz. Schulrates:

Lattmann.

Vertrag in vier Exemplaren ausgefertigt:

1 Exemplar an Institut,
1 Exemplar an Schweiz. Schulrat,
2 Exemplare an Zuse.

Quelle: Archiv der ETH Zürich

2. Die im Vertrag unter Ziff. 3 erwähnten Instandstellungs- und Ergänzungsarbeiten, die von Zuse bis zur Uebernahme der Maschine ausgeführt werden müssen, sind:

a) Ueberholung und Instandstellung aller elektrischen und mechanischen Einrichtungen;

b) Ersetzung abgenützter Teile (Relais, Schrittschalter, usw.), sodass der Zustand des Materials für einen dauernden Betrieb an der ETH genügt;

c) Ausbau des Registers auf eine Kapazität von 64 Worten;

d) Erhöhung der Rechengeschwindigkeit ~~auf 60 Operationen pro Minute;~~ entsprechend den techn. Möglichkeiten. Stiefel.

e) Einbau einer Vorrichtung für die Eingabe numerischer Werte auf Lochstreifen (input);

f) Einbau einer Vorrichtung für das Stanzen von Resultaten auf Lochstreifen (output);

g) Einbau eines Schreibwerkes zum Drucken der Resultate;

h) Herstellung einer für den Transport und die Aufstellung in Zürich geeigneten äusseren Form.

Quelle: Archiv der ETH Zürich

Am 11.7.1950 wurde die generalüberholte Z4 schliesslich an der ETH Zürich installiert.

Zuse erinnert sich an die Inbetriebnahme der Z4

Eine besondere Episode stellt die Inbetriebnahme der Z4 an der ETH in Zürich dar. Die Eröffnung erregte in der Schweiz allgemeines Interesse. Stiefel hatte alles gut vorbereitet und etwa I00 Gäste aus Industrie und Wissenschaft eingeladen. Wie üblich, wurde bis zum Vortage an dem Gerät eifrig gebastelt. Am Vormittage des Eröffnungstages wurden noch einmal alle wichtigen Programme durchgespielt. Alles lief einwandfrei. Die Vorführung sollte nachmittags 16.00 Uhr stattfinden, und so fanden wir noch Zeit zu einem ruhigen Mittagessen, das, wie in der Schweiz üblich, sehr gut war.

Nach dem Essen passierte nun allerdings etwas zunächst völlig Unerklärliches. Bereits nach Einstellung der einfachsten Aufgaben fing das Gerät plötzlich an zu bocken. Funken sprühten hier und dort, und es verbreitete sich allmählich der bekannte Geruch von 'Elektrikerbraten'. So etwas war mir in meiner ganzen Praxis mit Computern noch nicht passiert. Fassungslos standen wir vor diesen Erscheinungen. Um 16.00 Uhr sollte die Vorführung stattfinden. Stiefel hat in diesen Augenblicken sich sicher innerlich gefragt, ob es denn wirklich richtig war, eine deutsche Maschine nach Zürich zu holen.

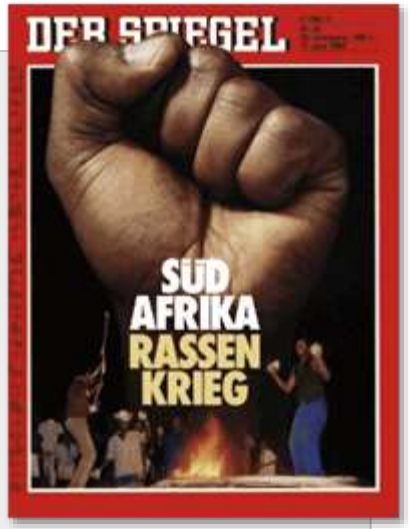
Schliesslich hatte Speiser die rettende Idee [...]. Nachdem dieser Fehler entdeckt war, hatten wir noch eine halbe Stunde Zeit, die Folgen zu beseitigen. In aller Eile wurden die durchgeschmorten Drähte durch fliegende Leitungen ersetzt. Der brenzliche Geruch konnte durch Öffnen der Fenster und Hereinlassen der schönen Schweizer Luft beseitigt werden. Um 16.00 Uhr begann pünktlich die Vorführung und alles lief einwandfrei.

All das gehört noch zur Pionierzeit des Computers. Die Jahre in Zürich habe ich in bester Erinnerung. Aus der anfänglichen Zusammenarbeit entwickelte sich eine echte Freundschaft. Speiser sorgte für eine erstklassige technische Wartung des Gerätes. Rutishauser entwickelte sich zu einem routinierten Programmierer und sicher haben seine Arbeiten über die ETH und die Z4 hinaus Bedeutung erlangt.

[Aus: ZAMP, Vol. 30, 1979, S. 399-403]

Abenteuerliche Geräte aus Altmaterial
DER SPIEGEL 17. Juni 1985 (Auszug)

Spiegel-Titel 25/1985



Zuse war in den 30er Jahren in Berlin so etwas Ähnliches wie heute die Garagen-Genies des Silicon Valley in den USA, die aus dem Brut-Klima der kalifornischen Hippie-Kultur zu Milliardenunternehmern aufstiegen. Doch seiner Karriere fehlt das dollarschwere Happy-End.

Er war ein begabter Laienschauspieler und Maler, versuchte sich vergeblich als Reklamezeichner und trieb sich in kommunistischen Jugendgruppen herum. Ohne große Begeisterung absolvierte er ein Studium als Bauingenieur. Von Rechenmaschinen hatte er keine Ahnung. Die endlosen Rechnereien, die sein Studium ihm abverlangte, waren ihm ein Gräuel. So entwickelte sich die Idee, einen Rechenautomaten zu bauen.

Im Jugendstil-Wohnzimmer seiner Eltern bastelte Zuse sein erstes Gerät zusammen. Die abenteuerliche Konstruktion, mit einem mechanischen Rechenwerk, war mit der Laubsäge aus Blech geschnitten und füllte den Esstisch. Hauptamtlich arbeitete er als Statiker der Flugzeugwerke an Flugabwehrraketen.

Die Mannschaft seiner eigenen Firma war eine kriegsbedingt merkwürdige Auslese. Ein Konstrukteur kam aus dem Irrenhaus, der Buchhalter war wegen Betrügereien vorbestraft. Der erste Programmierer Deutschlands, wenn nicht der Welt, war ein Blinder.

1967 ging die Zuse KG an Siemens. Der deutsche Elektro-Multi ließ die Entwicklungslinie der Zuse-Rechner einschlafen. Der Firmenname Zuse wurde gelöscht. Reich ist der Computer-Erfinder mit dem Verkauf seiner hochverschuldeten Firma nicht geworden. Seinen Lebensunterhalt sicherte er fortan durch Berater-Verträge und den Verkauf seiner Ölbilder.

1991: Ehrendoktorwürde der ETH für Zuse

Rechts:
Konrad Zuse

Links:
Prof. Walter Gander, 1991 Vorsteher der Abteilung IIIC (Informatik) der ETH Zürich

Portrait Prof. Dr. K. Zuse

Von Prof. Walter Gander, Vorsteher der Abteilung für Informatik

Konrad Zuse ist am 22. Juni 1910 in Berlin geboren. 1927 studierte er an der technischen Hochschule Berlin-Charlottenburg Maschinenbau, dann Architektur, und 1935 schloss er als Bauingenieur ab. Während des Studiums hatte er sich über den Aufwand ausgedehnter statischer Berechnungen geärgert. Er begann deshalb schon 1933 mit Überlegungen, wie man solche Berechnungen mechanisieren könnte. 1936 richtete er eine Erfinderwerkstatt in der Wohnung seiner Eltern ein, mit dem Ziel, einen programmgesteuerten Rechner zu konstruieren.

Prof. Dr. Konrad Zuse ist der Erbauer des ersten frei programmierbaren, in binärer Gleitpunktarithmetik arbeitenden Computers, der Z3. Sie wurde 1941, am 25. November, vor 50 Jahren, Fachleuten vorgestellt. Es gibt keine andere Wissenschaft als die Informatik, deren Begründer noch am Leben ist und die in so kurzer Zeit eine so grosse Bedeutung erlangt hat.

Im Bereich der Wissenschaft hatte die Schweiz mit der ETH Persönlichkeiten, die nicht nach innen geschaut haben, wie das in der Politik der Fall war. Prof. Stiefel erkannte an der ETH Zürich die grosse zukünftige Bedeutung der Computer, und es gelang ihm, sowohl den damaligen ETH-Präsidenten davon zu überzeugen als auch von Prof. Zuse die Z4 für 5 Jahre zu mieten.

Die ETH war damit die einzige Hochschule in Europa, die einen Computer in Betrieb hatte. Dies führte zu Zusammenarbeit von Prof. Zuse mit den späteren Professoren Rutishauser und Speiser. Es war der Grundstein für die erfolgreiche Entwicklung der numerischen Mathematik und Informatik der ETH Zürich.

Abgedruckt in: OUTPUT, Nr. 1/92, S. 25

ETH-Schulratsprotokoll vom 8.10.1949 zur Z4

Das Arbeiten mit modernen Rechenmaschinen nahm in den letzten Jahren einen ungeahnten Aufschwung. Es entwickelte sich geradezu eine neue Richtung der Mathematik, die experimentelle Mathematik. Zahlreiche mathematische Aufgaben lassen sich nur mit Hilfe der modernen Maschinen lösen. Die Forschung, die Industrie, die Banken und die Versicherungsgesellschaften beginnen heute mit solchen Rechenmaschinen zu arbeiten. Die Schweiz steht in der Entwicklung solcher Rechenmaschinen und in der Anwendung gegenüber dem Ausland in grossem Rückstand. Die schweizerische Präzisions-Industrie würde sich für dieses Gebiet ausserordentlich eignen. Die Kommission setzte sich am Anfang zwei Ziele:

a) die Entwicklung neuer Rechenmaschinen
b) die Weiterbildung bestehender Systeme.

Im Juli 1949 erfuhr die Kommission von einer Rechenmaschine des deutschen Ingenieurs Zuse, die von der E.T.H. zu ausserordentlich günstigen Bedingungen übernommen werden könnte. Prof. Stiefel und Dr. Lattmann besichtigten die Apparatur. Sie rühmten besonders deren mathematische Disposition und erklär-

ETH-Schulratsprotokoll vom 8.10.1949 zur Z4 (2)

Der Präsident: Die Rechenmaschine soll mehreren Zwecken dienen. Zunächst ist hervorzuheben, dass bestimmte mathematische Aufgaben nur maschinell in nützlicher Frist gelöst werden können. Das Institut kann mit der Maschine somit Aufträge von Professoren der E.T.H., aber auch von dritter Seite, z.B. von Banken, Versicherungen und Industriefirmen ausführen. Sodann aber hoffen wir, dass die Maschine der schweizerischen Industrie Anregungen für weitere Konstruktionen geben werde. Wenn auch in der Schweiz vielleicht der Bau von grossen Rechenmaschinen nicht in Frage kommt, so kann unsere Präzisionsinstrumenten-Industrie zweifellos doch direkt und indirekt aus der genauen Kenntnis moderner Rechenmaschinen Nutzen ziehen.

ETH-Schulratsprotokoll vom 12.7.1952 zur Z4

Die Maschine hat seither die gestellten Erwartungen in diesen drei Punkten erfüllt: So führte sie die mathematische Berechnung zahlreicher Probleme durch, die ohne Maschine nicht oder nur in gröberer Annäherung bzw. mit zu grossem Zeitaufwand hätten gelöst werden können. Wir erwähnen u.a.: Inversion von Matrizen, Integration von Schwingungsdifferentialgleichungen, spezielle Relaxationsverfahren zur Lösung partieller Differentialgleichungen, Berechnung gewisser elektrischer Netzwerke mit Potentiometern. Im Auftrag wurden u.a. folgende Aufgaben gelöst: Berechnungen der Spannungen in einer Talsperre, Raketenflug, quantenmechanische Untersuchung von Naphtalinmolekülen, Hilfsrechnung für die Hochfrequenztechnik, Strahlendurchgang durch optisches System, Ausgleich photogrammetrischer Streifenaufnahmen, Schwingungen vierachsiger Lokomotiven. Deformation von Flugzeugflügeln; kritische Drehzahlen von Turbinenaggregaten, Abflussregulierung der drei Juraseen. Einschwingvorgänge einer Servosteuerung, usw. - Im Institut liegen zahlreiche Aufträge vor, die mangels Zeit und Personal zurückgestellt werden mussten.

Z4: Rückblick

Die Z4 war an der ETH Zürich von 1950 bis 1954/55 in Betrieb. 1981 schrieb A. Speiser dazu:

Wenn wir auf die Zürcher Zeit der Z4 zurückblicken, so können wir abschätzen, dass sie während dieser vier Jahre ungefähr 15 Millionen Rechenoperationen ausgeführt hat. Die heutige Anlage an der ETH erbringt – wenn man die zentralen und die peripheren Prozessoren zusammenrechnet – die gleiche Rechenleistung in einigen Sekunden. Es ist ganz klar, dass während der vier Z4-Jahre zusammengerechnet für die Programmierung mehr Sorgfalt aufgewendet wurde und dass auch mehr wissenschaftlich bedeutende und praktisch nützliche Ergebnisse entstanden sind als heute während einiger Sekunden Betriebszeit. Diese Feststellung enthält im Grund eine Selbstverständlichkeit und ist nicht eine Kritik an den heutigen Benutzern. Der Vergleich zeigt aber, wie grundlegend sich das digitale Rechnen und die numerische Mathematik im Verlauf eines Vierteljahrhunderts gewandelt haben.

[Ambrosius Speiser: Die Z4 an der ETH Zürich: ein Stück Technik- und Mathematikgeschichte. In: Elemente der Mathematik 36 (1981), Heft 6, Seiten 145-176 (überarbeitetes Referat, gehalten an der ETH Zürich am 16. Juni 1981)]

Externe Benutzer mussten übrigens einen Rappen pro ausgeführtem Befehl bezahlen. Die Z4 geht nach ihrer Zeit an der ETH an das ISL, ein Rüstungsforschungsinstitut in St. Louis im elsässischen Dreiländereck (auf einem verlassenen Fabrikgeländes in der Nähe des Flughafens Basel-Mulhouse eingerichtet), in dem Frankreich ab August 1945, also unmittelbar nach Ende des zweiten Weltkriegs, einige zig deutsche Rüstungswissenschaftler vom ehemaligen Institut für Technische Physik und Ballistik der Technischen Akademie der Luftwaffe (Berlin-Gatow) beschäftigt, die (sehr zum Missfallen der lokalen Bevölkerung, die diesen Personen eine Mitschuld am gerade durchlittenen Krieg gab) mit ihren Familien auf der gegenüberliegenden deutschen Rheinseite, in Weil am Rhein (seinerzeit französische Besatzungszone) privilegiert untergebracht waren und täglich in einem verplombten Bus über Schweizer Gebiet zur Arbeit fuhren. Die Z4 wurde in St. Louis vermutlich für ballistische und gasdynamische Rechnungen eingesetzt.

Heute befindet sich die Z4 im Deutschen Museum in München.

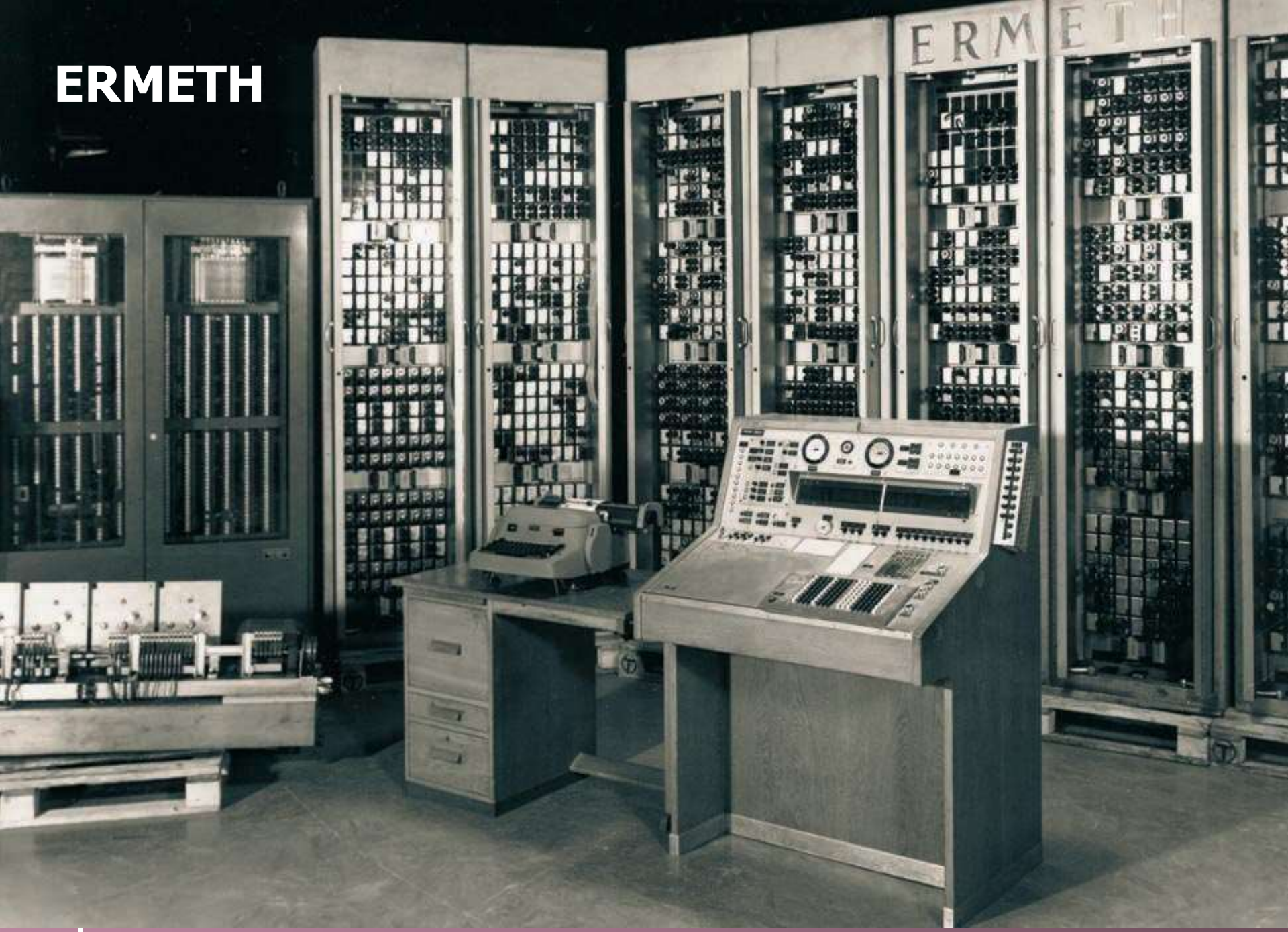
Z4: Rückblick (2)

When the scientific results started to flow out of Zurich in 1951 and 1952, some criticism was voiced in Germany to the effect that the Z4 should have been kept at home rather than letting it go abroad. In retrospect, the explanation for what happened is quite clear to me: In 1950, universities were still suffering from the consequences of the war. Buildings were badly damaged, equipment was almost non-existent and, accordingly, the limited funds had to be spent on the urgent needs of the day in order to keep university life at an acceptable level. The sum of 50,000 Swiss Francs that we paid for Z4 was clearly beyond the reach of any of the [German] universities.

But German interest in what was being done in Zurich had been awakened and our ERMETH plans, in particular, were closely followed. When the first industrial machines appeared in Germany, we found that a number of ERMETH ideas had been adopted. To us, it was a source of grief that our work was followed with much more interest in Germany than in our own country. To be sure, Swiss industry was polite and cooperative when we needed help, but they showed no interest in our results – nobody is a prophet in his own country!

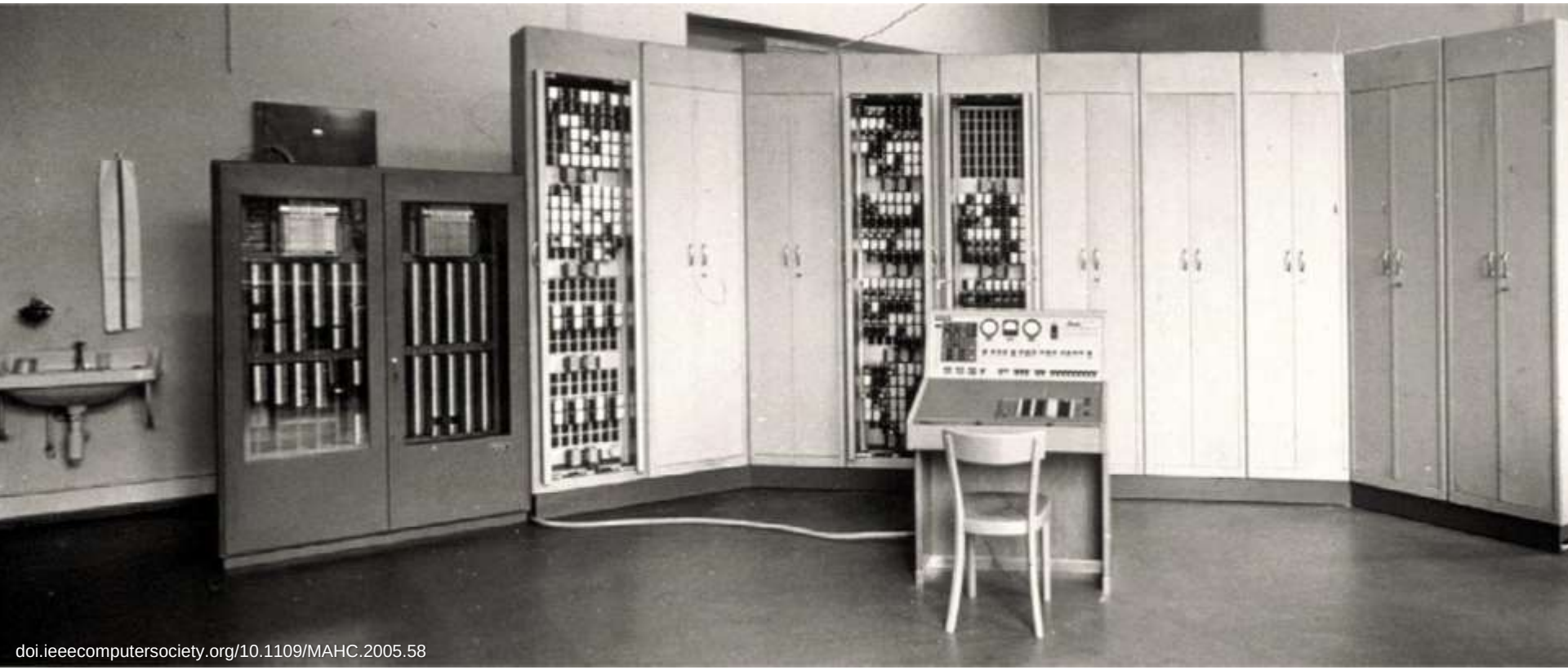
[Ambrosius Speiser: Konrad Zuse's Z4: Architecture, Programming, and Modifications at the ETH Zurich. In: The first computers (edited by Raúl Rojas and Ulf Hashagen). MIT Press, 2000, pp. 263-276.]

ERMETH



ERMETH – Elektronische Rechenmaschine der ETH

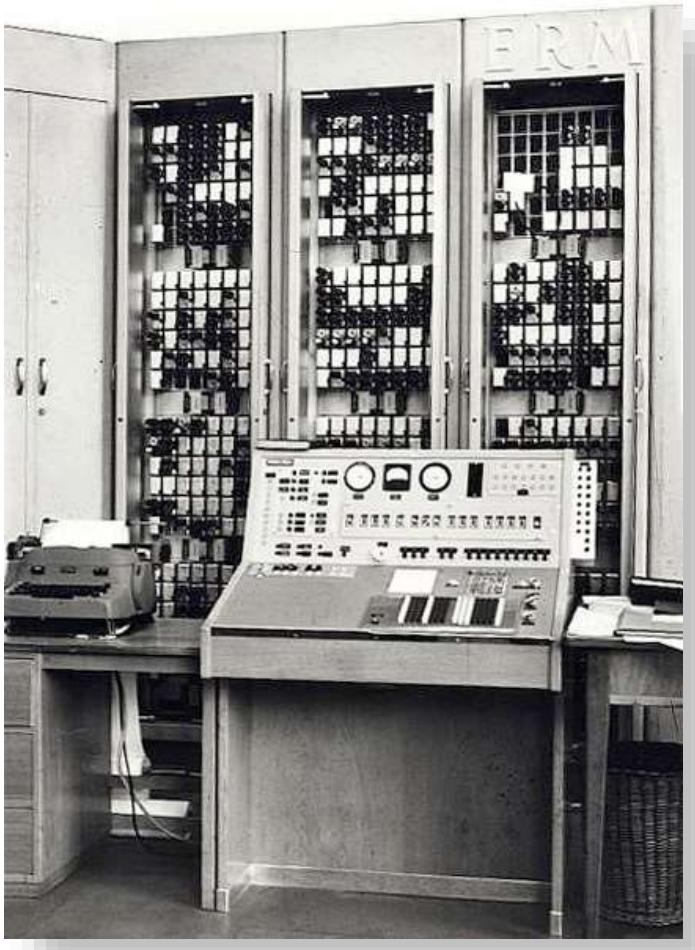
Nachfolger der Z4, gebaut 1952 – 1956 an der ETH Zürich und genutzt bis 1963



„Hinsichtlich der Organisation dieses Geräts war uns das Prinzip wegleitend, daß jeder Ingenieur nach einigen Stunden Instruktion in der Lage sein soll, sein Problem mit Hilfe dieser Maschine zu lösen, so daß das schädliche Staunen vor dem «elektronischen Hirn» durch nützliche Arbeit ohne speziell geschultes Personal abgelöst wird.“ -- Eduard Stiefel

ERMETH – Elektronische Rechenmaschine der ETH

ERMETH, gebaut 1952 – 1956 an der ETH Zürich und genutzt bis 1963, Nachfolgesystem der Z4.



„Bereits kurz nach der Gründung des Instituts für angewandte Mathematik im Jahr 1948 begann die Planung zum Bau eines eigenen Computers. Zu Beginn der 1950er Jahre waren auf kommerzieller Basis noch keine programmierbaren Rechner mit Speicher erhältlich, die für wissenschaftliches Rechnen geeignet gewesen wären. Deshalb auch die Idee zu einer Eigenentwicklung von Grund auf…"

www.ethistory.ethz.ch/rueckblicke/departemente/dinfk/weitere_seiten/ermeth/index_DE

Das ERMETH-Entwicklungsteam 1963

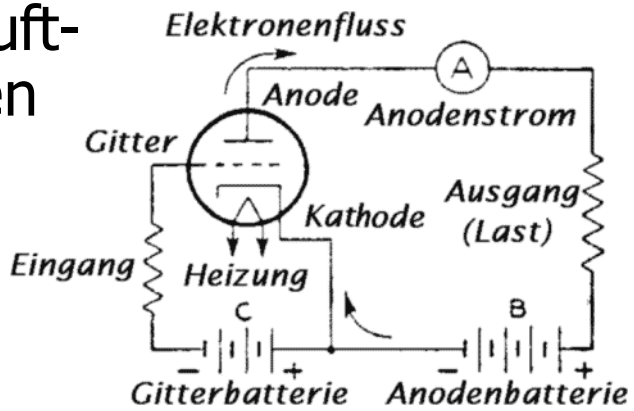
Stehend von links: Youssef Boutros, Peter Gantenbein, Alfred (Fredy) Stofer, Tibor Siklossy, Eduard Stiefel, Max Rössler, Jörg Waldvogel, Dimitri Koutroufiotis, Paul Szigeti, Hans-Rudolf Schwarz, Alfred Schai, Carl August (Gusti) Zehnder, Alfed Ganz, Roshdi Abdel-Rahmann Amer; sitzend von links: Annemarie Meier, Josef Schär, Heinz Rutishauser, Peter Läuchli, Hans Amman.

ERMETH

- 1800 Elektronenröhren
  - Doppeltrioden E90CC für Inverter, Flip-Flops und Verstärker
  - 6400 Germaniumdioden OA55 für AND- und OR-Gatter
  - 500 Relais für die Ein-/Ausgabe (Eingabe über Lochkarten)
- Magnettrommel: Arbeitsspeicher für 10000 Wörter à 14 Ziffern
  - 1.5 Tonnen, Ø 28 cm, 45 cm hoch, 6000 U/min, 200 Magnetspuren
- Ca. 60 Befehle / s → ca. 100 Mal schneller als die Z4, aber ca. eine Milliarde Mal langsamer als heutige Laptops
- 30 kW Leistungsaufnahme; Fläche von 50 m^2
  - Die ERMETH reagierte empfindlich auf Schwankungen der Netzspannung, etwa wenn morgens die Trams den Betrieb aufnahmen
- Entwicklung kostete eine Million Franken

Elektronenröhren als elektronische Bauelemente

- Elektronenröhren bestehen aus einem luftleeren Glaskolben mit Metall-Elektroden
  - Am. Engl.: „vacuum tube“ (vgl. dazu auch „YouTube“ nach *cathode ray tube* = CRT)
  - Dioden als einfachste Elektronenröhren (mit nur einer Anode und einer Kathode als Elektroden) wirken als Gleichrichter



- Aus der beheizten Kathode treten Elektronen aus und werden durch ein elektrisches Feld zur Anode bewegt
  - Trioden besitzen als weitere Elektrode ein Gitter, mit dem der Strom von der Kathode zur Anode gesteuert werden kann (Wirkung analog zu einem Transistor)
- Elektronenröhren dienen der Erzeugung, Verstärkung oder Modulation elektrischer Signale
  - Waren (ab ca. 1910) bis zur Einführung des Transistors die einzigen aktiven elektronischen Bauelemente
  - Wurden in Verstärkern, Radiogeräten, Sendern, Computern (*Elektronen-* bzw. *Röhrenrechner*) verwendet

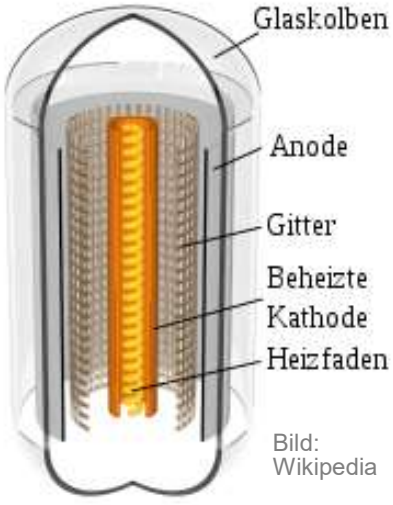


Bild: Wikipedia

Elektronenröhren als elektronische Bauelemente (2)

Brit. Engl. für „Elektronenröhre“

A valve consists of an evacuated glass envelope containing a number of electrodes. These are connected to the outside by wires passing though special seals. The innermost electrode is the cathode. This consists of a metal tube coated with a material that emits electrons when it is heated. The heat is provided by a tungsten wire, situated inside the cathode and connected to a 6 or 12 V supply.

In the simplest form of valve, called a diode, the cathode is surrounded by a metal cylinder called the anode. If the anode is connected to a voltage that is positive relative to the cathode, the anode attracts electrons from the cathode, and a current flows. Current cannot flow in the other direction.

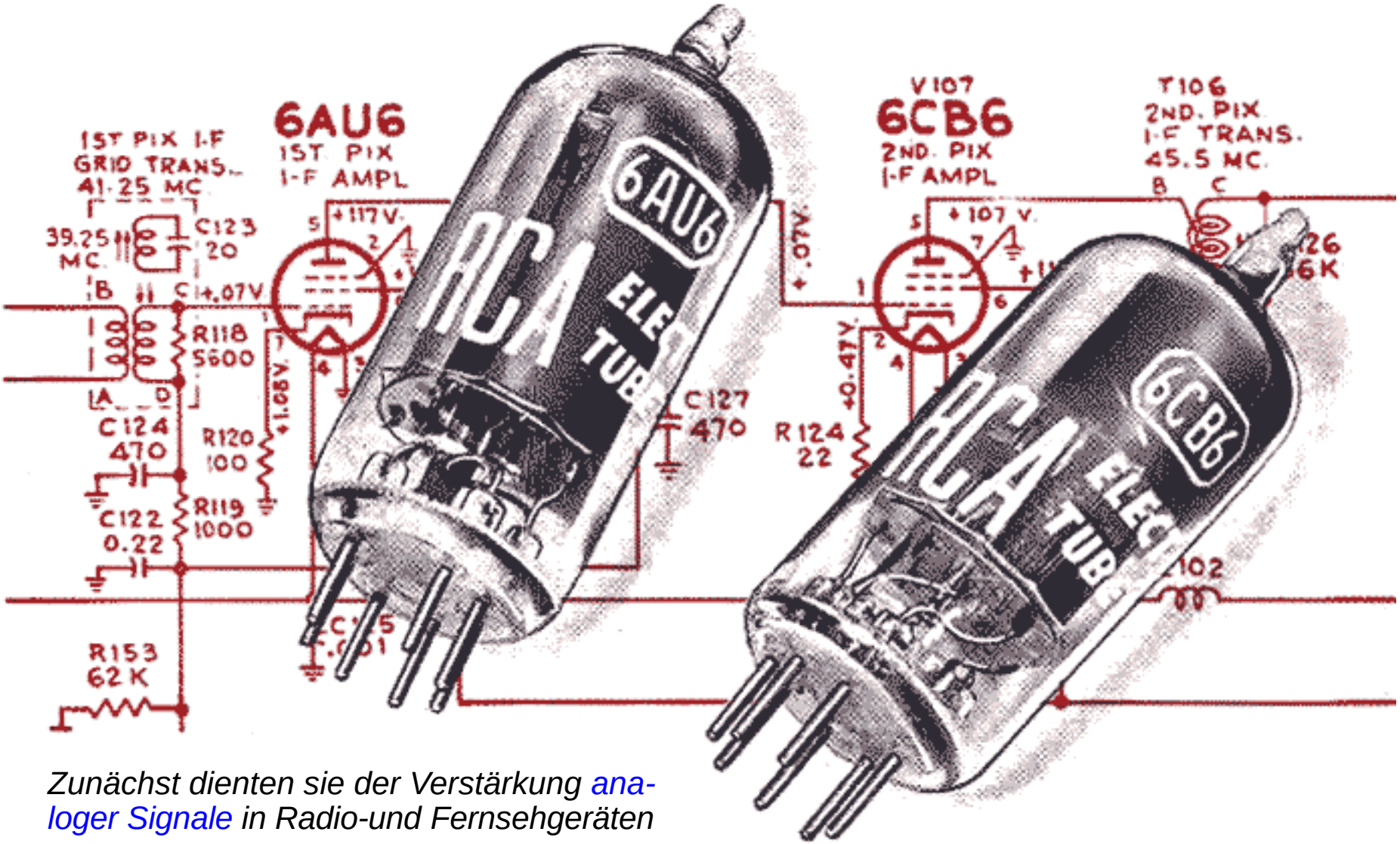
In the type of valve called a triode, a grid of fine wires is inserted between the cathode and the anode. A voltage on this grid (normally a few volts negative relative to the cathode) can control the flow of electrons to the anode. This enables the valve to be used as an amplifier, the current depending on both the grid voltage and the anode voltage.

Valves were initially developed as linear amplifiers. When a valve is used in this way, the anode current is related to the grid voltage. For small signal amplitudes, the distortion is small. Valves are used either as radio frequency or as audio frequency amplifiers.

When television started in the 1930s, it was an easy step to use valves as on–off or change-over switches. When the valve is used in this way, the current is either on (at some controlled level) or off. This is how valves were used in digital computers.

[Auszug aus: David O. Clayden: How valves work.]

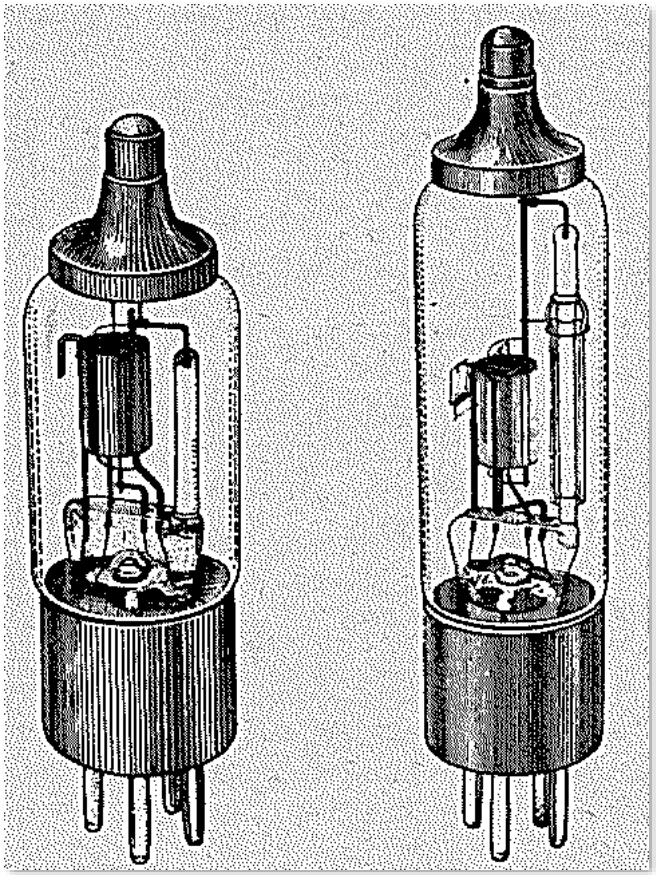
Elektronenröhren als elektronische Bauelemente (3)



Zunächst dienten sie der Verstärkung ana-loger Signale in Radio-und Fernsehgeräten

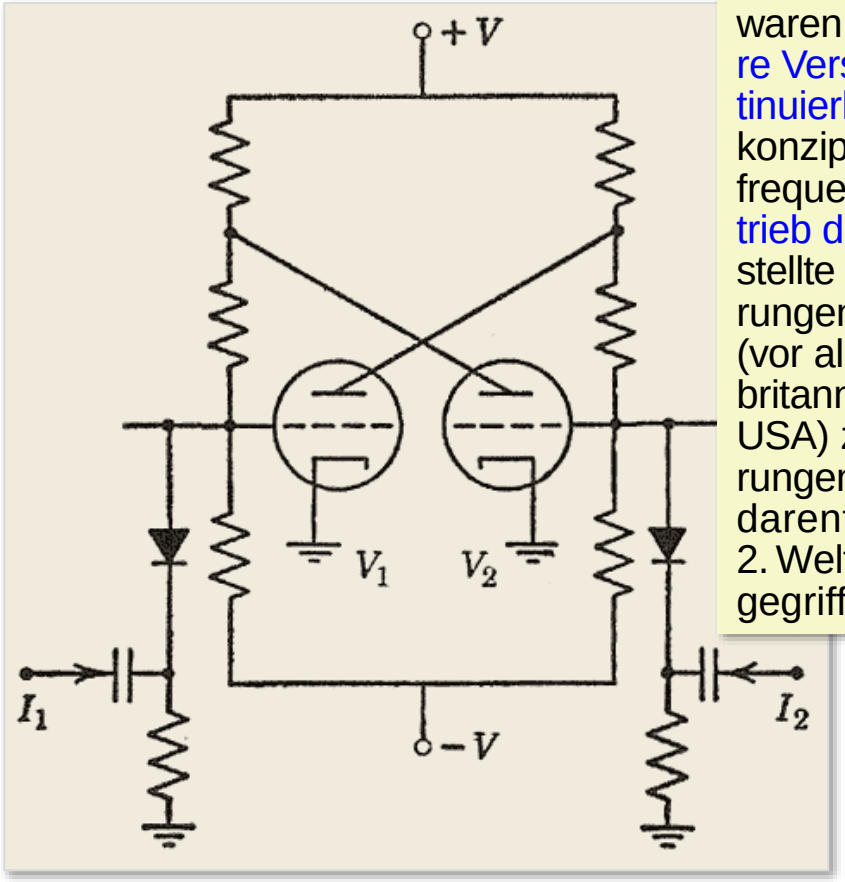
http://digital.hagley.org/islandora/object/islandora%3A2167044

Elektronenröhren als elektronische Bauelemente (4)



Röhren von AEG-OSRAM, 1936

https://upload.wikimedia.org/wikipedia/commons/7/78/Elektrometerroehren.PNG

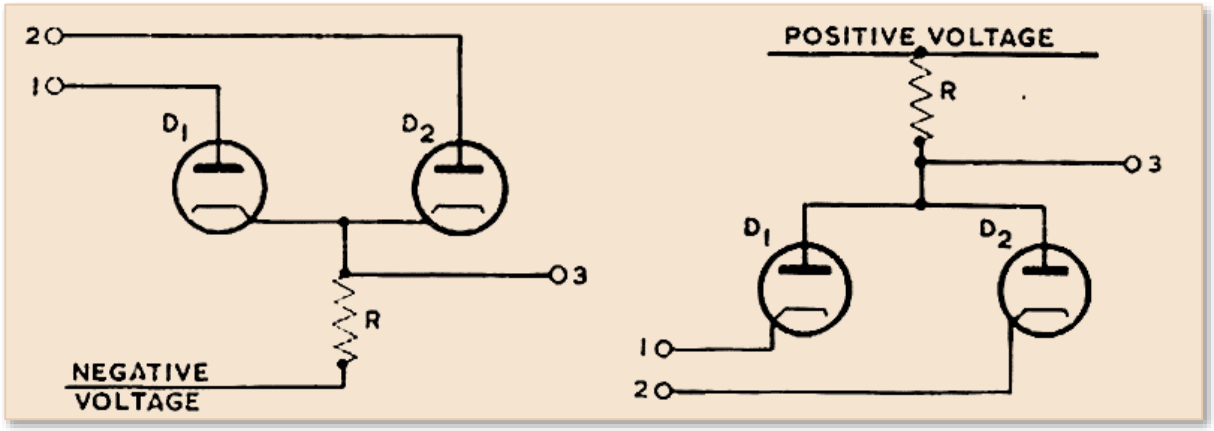


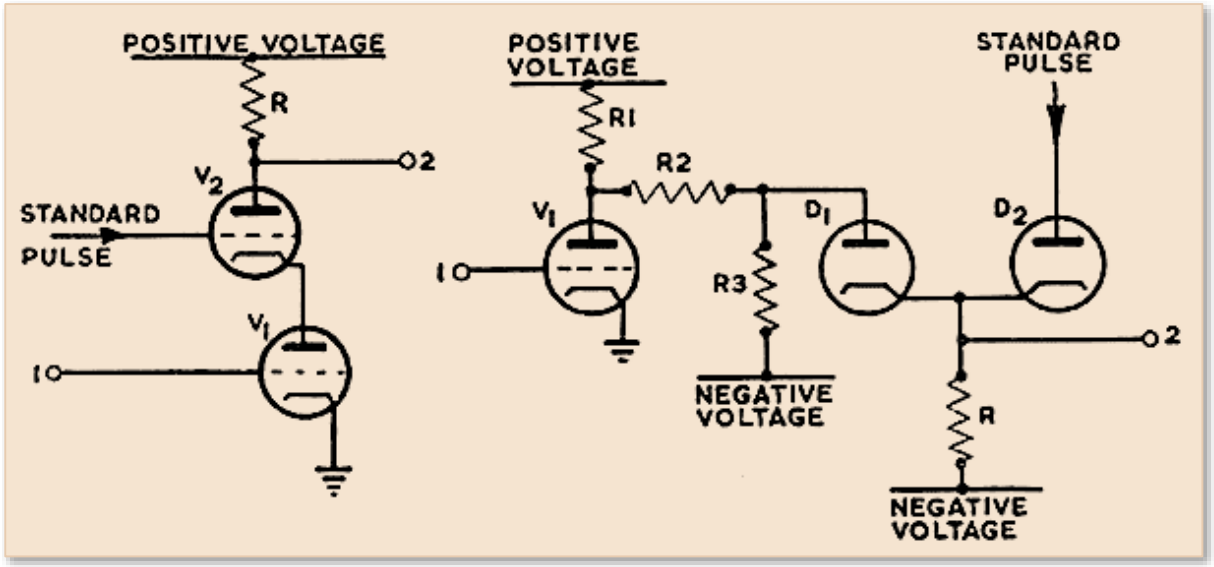
Eine Digitalschaltung: Flip-Flop mit Trioden

[Montgomery Phister: Logical design of digital computers, 1958]

Elektronenröhren waren für die lineare Verstärkung kontinuierlicher Signale konzipiert; der hochfrequente Impulsbetrieb der Digitallogik stellte neue Anforderungen; hier konnte (vor allem in Grossbritannien und den USA) z.T. auf Erfahrungen aus der Radarentwicklung im 2. Weltkrieg zurückgegriffen werden.

Elektronenröhren als elektronische Bauelemente (5)





Schaltungsprinzipien für die logische Und-Funktion (links oben), die logische Oder-Funktion (rechts oben), sowie zwei Varianten der logischen Negation (unten) mit Röhren.

Quelle: "Faster Than Thought" (Hg: Bertram Vivian Bowden), 1953

Elektronenröhren als elektronische Bauelemente (6)

Steckmodule mit Elektronenröhren aus einem IBM-Mainframe der 1950er-Jahre

www.emsp.tu-berlin.de/fileadmin/fg232/Lehre/MixedSignal/Bilder/Pionierzeit/603.jpg

Flip-Flop-Röhrensteck-einheit einer IBM 650

Elektronenröhren als elektronische Bauelemente (7)

The early radio engineers were concerned with sine waves of various frequencies — radio, intermediate, audio — and nothing else. By the 1930s cathode ray tubes were coming into use and bringing with them new and strange wave forms, particularly time bases and strobes. Primitive analogue computing devices were also appearing. A new term, 'electronics', was coined for the new technology.

Electronic techniques were much to the fore in ionosphere research and in television. They were vigorously exploited during the war for radar and other applications and, by the end of the war, knowledge of electronics had become widespread.

Although their experience in other applications of electronics stood them in good stead, computer designers soon found they had to learn a few new tricks, such as how to handle non-repetitive wave forms.

Obviously vacuum tubes would be used for amplifiers and this seemed straightforward enough. However, the output was at a much higher voltage than the input, and the interstage coupling circuits had to allow for this. The designer could either use capacitors or pulse transformers for interstage coupling, with diodes for zero restoration (otherwise called clamping), or he could use a resistor chain, perhaps with capacitors for frequency correction.

When all seemed set for a great future with vacuum tubes, transistors came along and we were all back at square one.

[Auszug aus: Maurice Wilkes: Recollections of early vacuum tube circuits.]

„Die Fabrikation von Hochvakuum-Röhren“

Einige Auszüge aus einem rund 100 Jahre alten Artikel der Telefunken-Zeitung (Heft 19, Feb. 1920, S. 14 – 26) des Physikers Hans Rukop (1883 – 1958), leitender Angestellter bei Telefunken – nicht nur der Inhalt, sondern auch die Diktion stellen ein Zeitzeugnis dar:

Das Wort „Fabrikation“ löst bei einem Manne der Technik, der sein Leben über Revolverbänken und Automaten verbringt, alsbald Vorstellungen aus wie: Stanzen, Ziehen, Sandstrahlen, Abstechen, Bohren, Lackieren, Zusammensetzen. Aber bei Hochvakuumröhren ist das eine andere Sache: Hochvakuumröhren verlangen zum großen Teil eine „physikalische Fabrikation“, die jedem echten Fabrikationsmann ein Dorn im Auge ist; und trotzdem wartet auf den tüchtigen, phantasievollen Massenfabrikanten gerade bei Röhren viele und dankbare Arbeit, die sich nicht auf dem fast abgegrasten Gebiete der Ziehpressen bewegt.

Heute umfaßt die Röhrenfabrik 3500 Quadratmeter – drei Stockwerke eines großen Hauses in der Friedrichstraße. Es ist wohl klar, daß in den ersten Monaten des Werdens die Fabrikation eine absolute Handarbeit war, bei der man bezüglich Einhaltung der vorgeschriebenen Dimensionen auf das Wohlwollen des Mechanikers – der die ersten Elektroden aus Zehnpfennigstücken hämmerte – und des Glasbläsers angewiesen war. Das ist auch nicht anders möglich, solange man nicht mit Bestimmtheit weiß, daß man von einer vorliegenden Type wenigstens einige hundert fabrizieren kann.

Wir haben bei Verstärkerröhren stets Bleiglas verwendet, das sich wunderschön verarbeiten ließ. Weiter: was für Metall verwendet man für die Durchführungen? Platin ist das klassische Material hierfür, und wir haben es stets, wenigstens in den ausgelieferten Röhren, benutzt. Die Platin-Ersatzmetalle haben uns nur großes Leid gebracht. Eine einzige gesprungene Röhre im Ofen, und das ganze Quantum ist hinüber. Der Ausfall wächst ins Gigantische!

Für die Glühkathoden verwendet man Wolframfäden, wie sie in Glühlampen benutzt werden.

„Die Fabrikation von Hochvakuum-Röhren“ (2)

Aber die Wolframfäden sind eine kleine Wissenschaft für sich, und die paar Geheimnisse, die man über sie und darüber hinaus hat, sind für 99 Prozent aller Leser ganz uninteressant, und das eine Prozent, dem sie interessant sind, soll sie doch lieber nicht erfahren!

Für Anoden braucht man dünne Bleche. Solche sind etwa aus folgenden teuren Metallen zu haben: Platin, Iridium, Gold, Palladium, Silber, Tantal, Molybdän, Wolfram. [...] Es bleiben daher ernstlich nur Kupfer, Nickel und Eisen übrig, die alle drei einigermaßen selbst für Hochvakuum benutzbar sind.

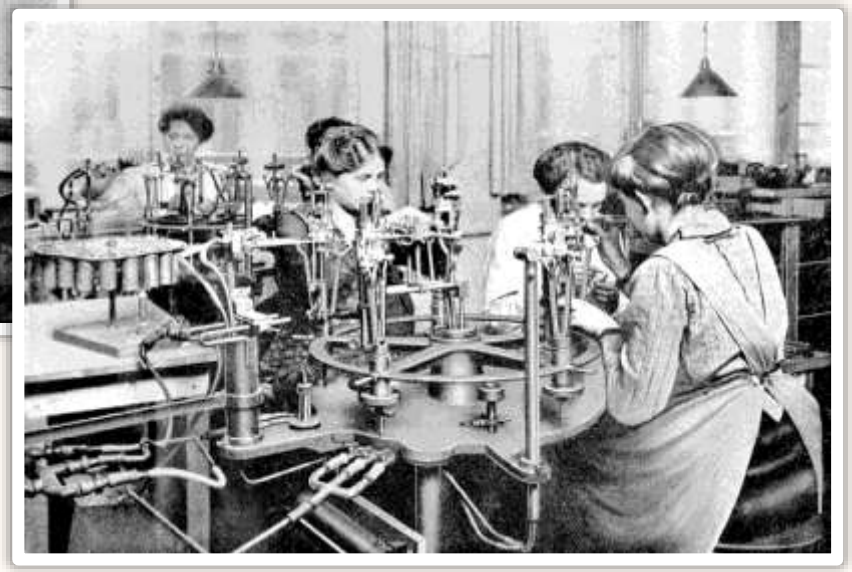
Die richtige Fabrikation von Verstärkerröhren in großen Zahlen muß notwendigerweise von der Handarbeit abgehen, die viel Leute und viel Platz braucht und dementsprechend teuer ist; dabei ist die Gleichmäßigkeit immer fraglich.

Die Glasbläser arbeiten vorzugsweise nach Augenmaß, wobei doch ernstliche Differenzen von den erforderlichen Längen, Abständen, Dicken usw. unvermeidlich sind. Dies ist mit Rücksicht auf die Flammen und die allseitige Zugänglichkeit der zu verschmelzenden Teile oft schwierig zu bewerkstelligen.

„Die Fabrikation von Hochvakuum-Röhren“ (3)

Biegen, Aufsetzen und Schweißen der Elektroden

Die erste Andeutung der entstehenden Röhre ist ein etwa 5 cm langes Stück Glasrohr, das an einer Anschlagvorrichtung abgeschnitten, dann an einer zweiten drehbankähnlichen Vorrichtung in der Gasflamme an einem Ende zu einem sogenannten Teller erweitert wird.

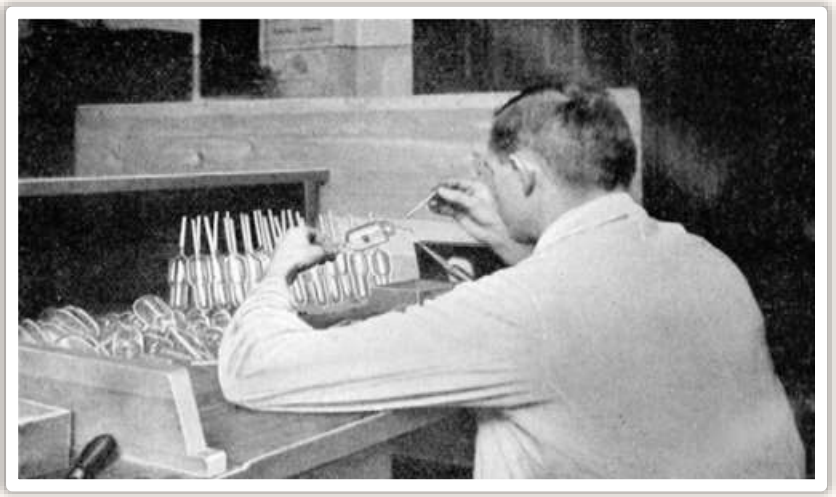


Fußquetschmaschinen

Dieses Tellerrohr wird zum „Fuß“ verarbeitet. D.h., die sämtlichen Zuleitungen für die Elektroden im Innern der Röhre, sowie die Halter zur Befestigung der Elektroden werden vakuumdicht in das Rohr eingeschmolzen.

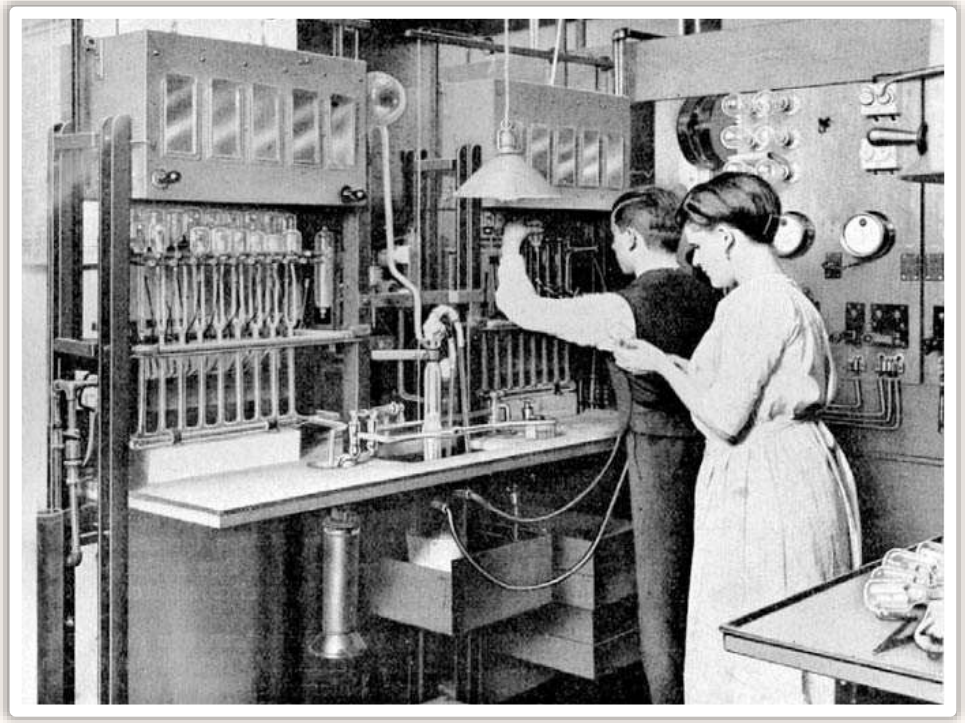
Der Fuß wandert nach der Werkstatt zum Aufsetzen der Elektroden. Hier wird er in mehreren kleinen raffinierten Maschinellen bearbeitet, die ihm unter Bedienung von „zarter Hand“ zuerst die Elektrodenhalter zurechtbiegen, dann allmählich erst das Gitter, dann die Anoden, schließlich den Glühfaden einsetzen.

„Die Fabrikation von Hochvakuum-Röhren“ (4)



Einschmelzen des Fußes in den Kolben

Der mit allen Elektroden versehene Fuß kommt nun zum ersten Mal in die Hände des Glasbläsers. Fertig aus der Glashütte bezogene Kolben werden teilweise maschinell mit einem Röhrchen zum Anfassen und Blasen versehen, das überschüssige Glas wird abgestochen, der Fuß eingeschmolzen, schließlich wird der Evakuieransatz (Stengel) eingeschmolzen. Die Röhre ist dann reif zum Evakuieren.



Einsetzen der Röhren in den Evakuierofen

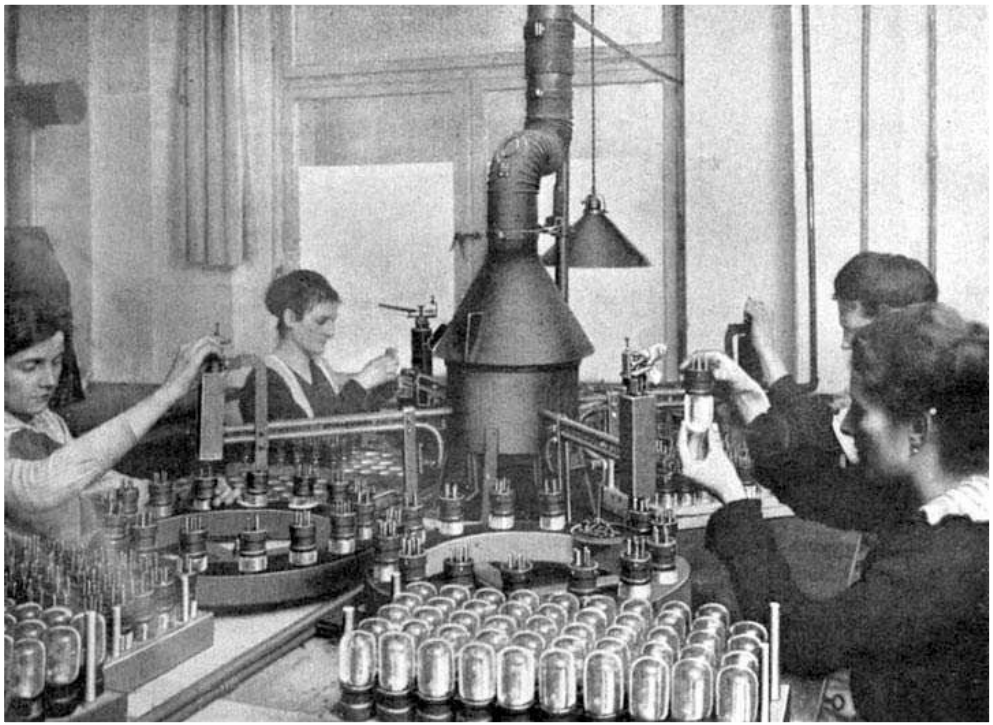
Ja, wenn das Evakuieren nicht wäre! Hier ist die Quelle fast allen Übels. Das Evakuieren geschieht zwar nach einem bestimmten Rezept, aber es verlangt große Aufmerksamkeit und guten Blick für die kleinsten Anormalitäten. Die gestengelten Röhren werden zum Evakuieren auf eine Glasgabel so aufgeschmolzen, daß sie zu je 20 Stück zusammen in einem Ofen sitzen.

„Die Fabrikation von Hochvakuum-Röhren“ (5)

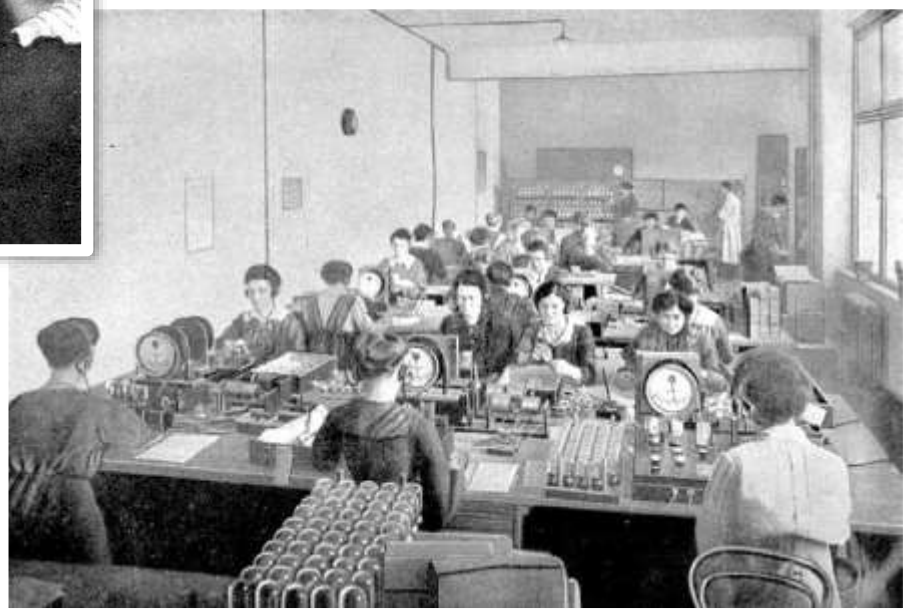
Jede einzelne Röhre wird einer Vakuum-messung unterzogen.

Hierauf kommt sie mehrere Stunden in die „Dauerprobe“, wo sie mit übernormaler Anodenspannung brennen muß. Die Dauerprobe ist für die Röhre etwa das, was für die Menschen im Mittelalter die „peinliche Befragung“ war.

„Die Fabrikation von Hochvakuum-Röhren“ (6)

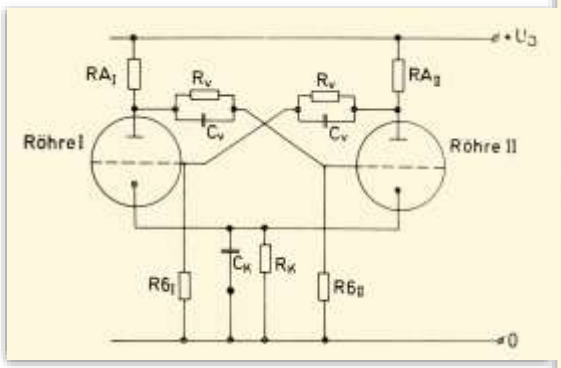


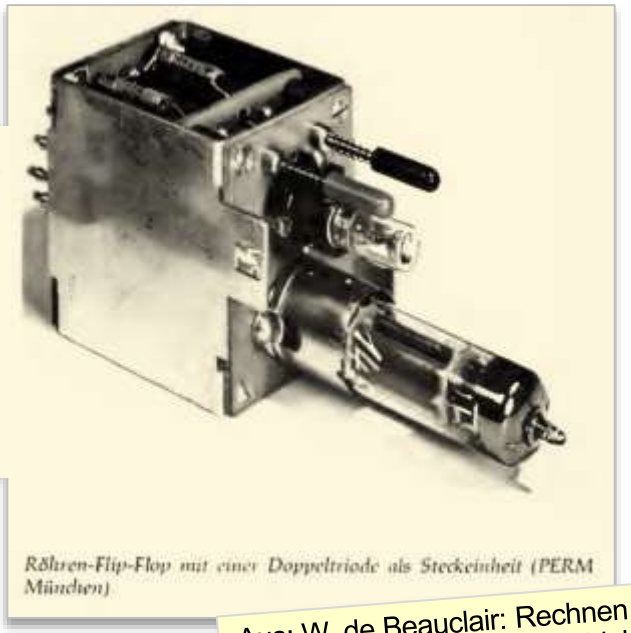
Die brauchbar befundenen Röhren werden nun gesockelt, gestempelt, geätzt, beklebt usw., worauf sie natürlich zum dritten Mal geprüft werden müssen, denn hierbei kommt noch mancher Fehler vor. Im ganzen wird infolge der scharfen Prüfbedingungen ein erklecklicher Prozentsatz ausgeschieden.



Jede Röhre wird fein säuberlich in eine Schachtel gelegt. Die Schachtel ist aus echtem Pappdeckel-Ersatz. Früher, als die Röhren noch so selten waren, wie die Ananas auf dem Kartoffelfeld, gab es eine wunderschöne Samtschachtel dazu. Ja, das samtene Zeitalter ist wohl vorbei, jetzt kommt das pappdeckelne.

Elektronenröhren / Transistoren als elektronische Bauelemente



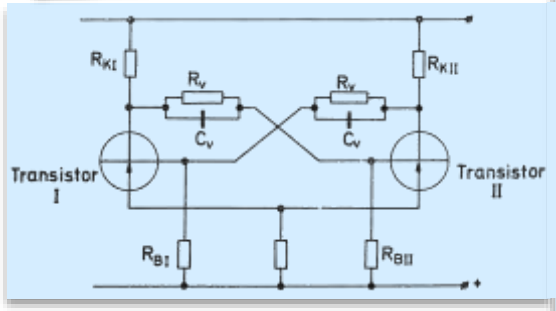


Röhren-Flip-Flop mit einer Doppeltriode als Steckeinheit (PERM München)

Flip-Flops mit Röhren sowie mit Transistoren; ein Vorläufer der Flip-Flop-Schaltung wurde 1918 von William H. Eccles und Frank W. Jordan bei Untersuchungen rückgekoppelter Röhren-Verstärker entdeckt (sogenannte Eccles-Jordan-Triggerschaltung)

Aus: W. de Beauclair: Rechnen mit Maschinen – Eine Bildgeschichte der Rechentechnik, 1968

Flipflop steht nicht nur für eine Schaltung, sondern auch für einen Schuh, dessen Bezeichnung wahrscheinlich mit dem Namen der Schaltung zusammenhängt. In Kalifornien muss es zu jener Personalunion von Ingenieur und Surfer gekommen sein, die nicht nur dafür sorgte, dass man in den Datenströmen zu surfen glaubt, sondern auch einer Sandale zum Namen einer Schaltung verhalf. --- Stefan Heidenreich



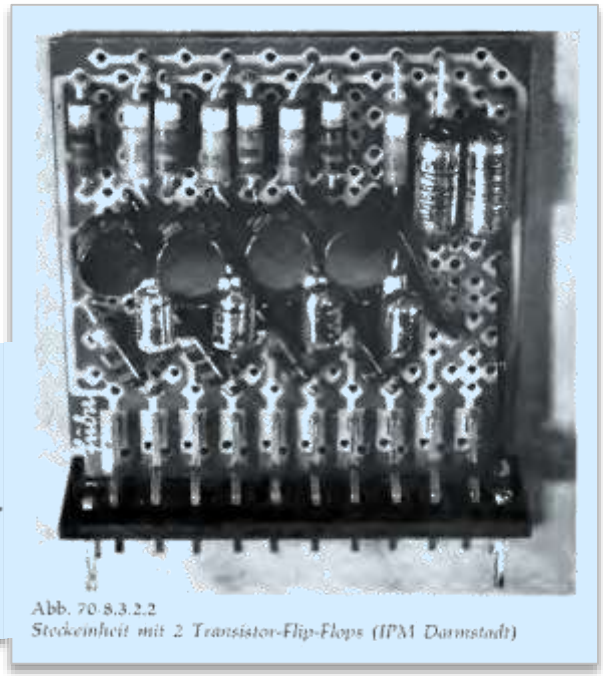
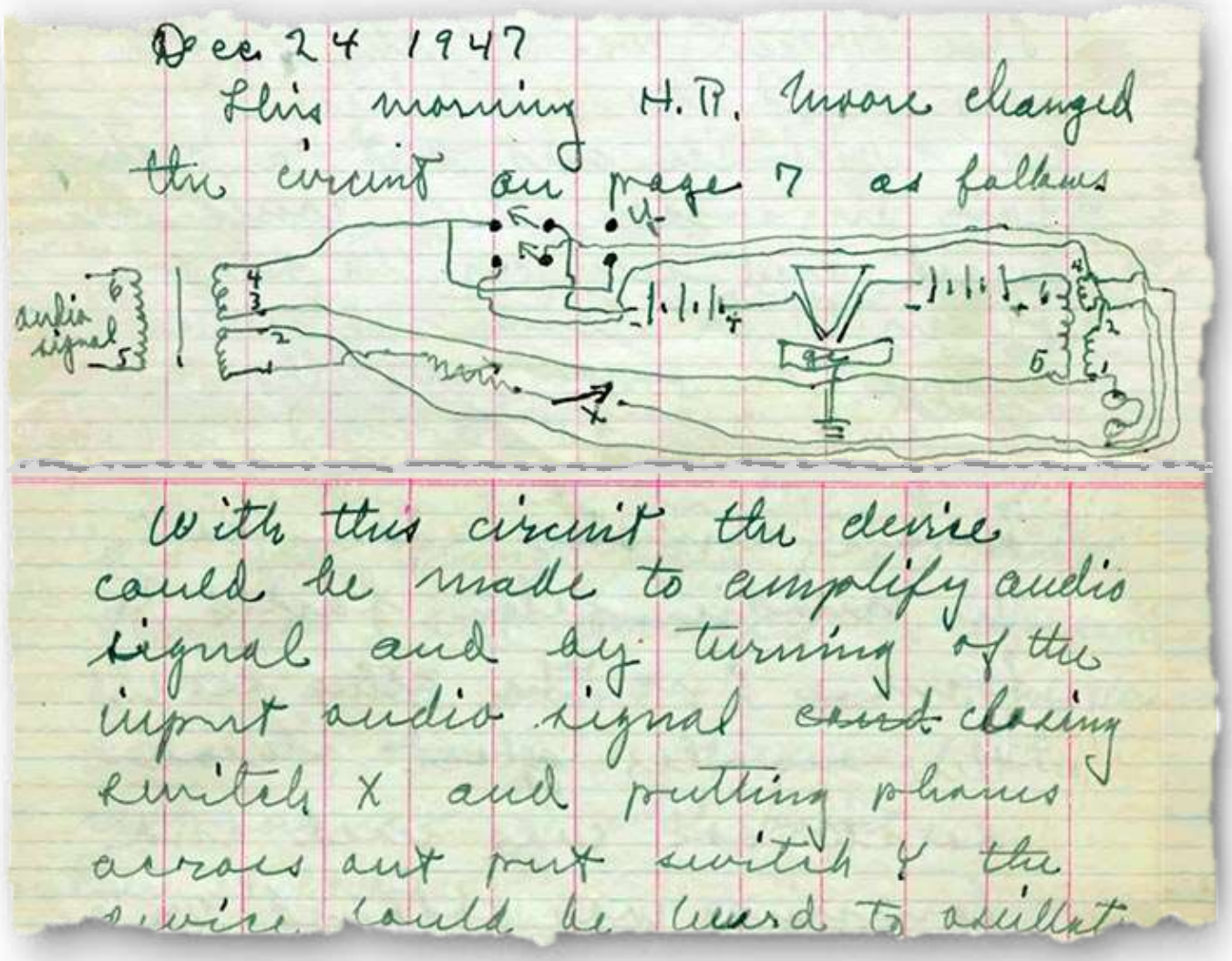


Abb. 70.8.3.2.2
Steckeinheit mit 2 Transistor-Flip-Flops (IPM Darmstadt)

Die Erfindung des Transistors, Weihnachten 1947



Aus dem Laborbuch von Walter Brattain (Bell-Labs) an Heiligabend 1947.

With this circuit the device could be made to amplify audio signal and by turning of […] the device could be made to oscillate.

Wie der Transistor zu seinem Namen kam

Bell Labs, where the transistor was invented, basically needed a new term to describe this new device that had up until that point been referred to as a “semiconductor triode”. The committee within Bell Labs that was set up to standardize terminology across the company were unable to decide on a name for it, so, they held a ballot across the entire company to vote for the name.

On the subject of a generic name to be applied to this class of devices, the committee is unable to make an unanimous recommendation. A discussion of some proposed names is given here.

Semiconductor triode. This is considered to be a fairly good name, being satisfactorily descriptive, but a shorter name would be preferable. The "triode" describes the three element device; if more elements were added it might be a tetrode or pentode, for instance. A single point contact rectifier might be referred to as a semiconductor diode in line with this terminology.

Surface States triode. This is in the same class as the first name suggested above; it is descriptive, but is not brief.

Crystal triode. The objection to this is that the term "crystal" is usually associated with the piezoelectric types, such as quartz.

Wie der Transistor zu seinem Namen kam (2)

Solid triode. This has the advantage of brevity, and is descriptive in the sense that the device may be explained by the physics of the solid state, and also that the active element is a solid rather than vacuum or gas filled. However, the word "solid" also commonly means sturdy, massive, rugged, or strong, which terms are contradictory to the actual physical characteristics of the unit.

Iotatron. This term satisfactorily conveys the sense of a minute element, as contrasted to the previous name. However, in view of the many vacuum or gas filled devices such as thyratrons, dynatrons, transitrons, etc., it lacks the distinguishing property which would differentiate it from such devices.

Transistor. This is an abbreviated combination of the words "transconductance" or "transfer", and "varistor". The device logically belongs in the varistor family, and has the transconductance or transfer impedance of a device having gain, so that this combination is descriptive.

Accompanying this memorandum is a ballot. It is suggested that each person to whom the memorandum is routed, fill out the ballot and return it, in order that the resultant vote may be used by the committee as the basis of a recommendation for a generic name.

Press Release from Bell Telephone Laboratories

Thursday, July 1, 1948. – An amazingly simple device, capable of performing efficiently nearly all the functions of an ordinary vacuum tube, was demonstrated for the first time yesterday at Bell Telephone Laboratories where it was invented.

Known as the Transistor, the device works on an entirely new physical principle discovered by the Laboratories in the course of fundamental research into the electrical properties of solids. Although the device is still in the laboratory stage, Bell scientists and engineers expect it may have far-reaching significance in electronics and electrical communication.

The whole apparatus is housed in a tiny cylinder less than an inch long. It will serve as an amplifier or an oscillator -- yet it bears almost no resemblance to the vacuum tube now used to do these basic jobs. It has no vacuum, no glass envelope, no grid, no plate, no cathode and therefore no warm-up delay.

Two hair-thin wires touching a pinhead of a solid semiconductive material soldered to a metal base, are the principal parts of the Transistor. These are enclosed in a simple, metal cylinder not much larger than a shoe-lace tip. More than a hundred of them can easily be held in the palm of the hand. […]

Yesterday's demonstration emphasized some of the many uses the Transistor may have in telephone communication, as well as its ready adaptability to the electronic techniques of radio, television, and public address systems. In one demonstration, a Transistor was used to amplify the electrical speech waves traveling between two telephones, a function now performed by vacuum tubes. In another, the audience heard a radio broadcast from a set constructed entirely without vacuum tubes, but using instead several of the tiny Transistors to provide amplification. A Transistor was also used to generate a standard frequency tone, thus demonstrating its role as an oscillator. […]

Der Transistor – eine Halbleitertriode

The Transistor, A Semi-Conductor Triode

J. Bardeen and W. H. Brattain
Bell Telephone Laboratories, Murray Hill, New Jersey
June 25, 1948

A THREE-ELEMENT electronic device which utilizes a newly discovered principle involving a semiconductor as the basic element is described. It may be employed as an amplifier, oscillator, and for other purposes for which vacuum tubes are ordinarily used. The device consists of three electrodes placed on a block of germanium[1] as shown schematically in Fig. 1. Two, called the emitter and collector, are of the point-contact rectifier type and are placed in close proximity (separation ~.005 to .025 cm) on the upper surface. The third is a large area low resistance contact on the base.

The germanium is prepared in the same way as that used for high back-voltage rectifiers.[2] In this form it is an *N*-type or excess semi-conductor with a resistivity of the order of 10 ohm cm. In the original studies, the upper surface was subjected to an additional anodic oxidation in a glycol borate solution[3] after it had been ground and etched in the usual way. The oxide is washed off and plays no direct role. It has since been found that other surface treatments are equally effective. Both tungsten and phosphor bronze points have been used. The collector point may be electrically formed by passing large currents in the reverse direction.

Each point, when connected separately with the base electrode, has characteristics similar to those of the high

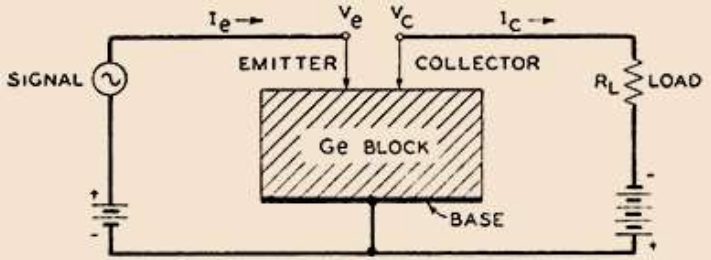
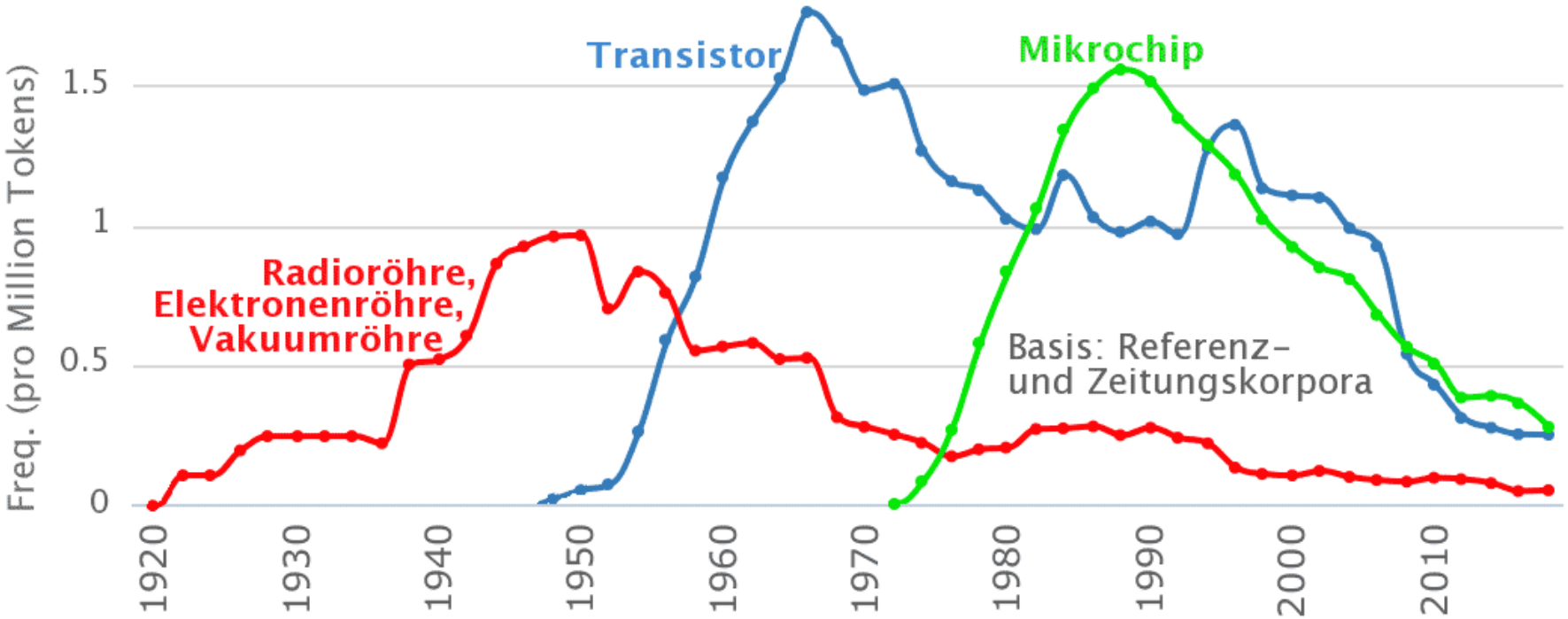


FIG. 1. Schematic of semi-conductor triode.

Das erste paper zum Transistor, in:
Physical Review 74.2 (1948): 230

Röhren, Transistoren, Mikrochips



Wortverlaufskurven aus dem Korpus des „Digitalen Wörterbuchs der deutschen Sprache“ (DWDS, www.dwds.de).

Transistors came along and we were all back at square one

TRANSISTORS

ARE IN STOCK

FOR IMMEDIATE DELIVERY

IN ANY QUANTITY...

AT RADIO SHACK!

ACTUAL SIZE

CK722 RAYTHEON

CK721 RAYTHEON

Raytheon PNP germanium junction transistors are now available for the first time. Heralded as the most revolutionary device since the vacuum tube, PNP transistors are now being used in hearing aids and other units *now being sold* or shortly to reach the market. Every lab and technician with a future in this business should become familiar NOW with this remarkable Raytheon product!

AV. CHARACTERISTICS AT 30° C

Description	CK721	CK722
Collector voltage (volts)	-1.5	-1.5
Collector current (ma)	-0.5	-0.5
Base current* (ua)	-6	-20
Current amplif. factor*	40	12
Power gain* (db)	38	30
Noise factor* (1000 cy) (db)	22	22

*Grounded emitter connection

CK721 (#38-386)

$12.50

CK722 (#38-387)

$7.60

Eine der ersten Verkaufsanzeigen zu Transistoren im März 1953

The CK722 was the first low-cost PNP germanium junction transistor available to the general public. It was introduced by Raytheon in early 1953 for $7.60 each; the price was reduced to $3.50 in late 1954 and to $0.99 in 1956. The original CK722 were direct fallouts from CK718 hearing aid transistors, that did not meet specifications. These fallouts were later stamped with CK721 or CK722 numbers. In the 1950s and 1960s, countless “build it yourself” articles were published in the popular electronics press and electronics/ hobbyist magazines describing how to use the CK722 to build all types of devices such as radios, oscillators, electronic voltmeters, and photoelectric alarms. [Wikipedia; www.ck722museum.com]

ERMETH: Röhre E90CC, Germaniumdiode OA55

SIEMENS RÖHREN

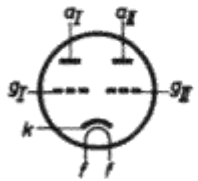
DOPPELTRIODE

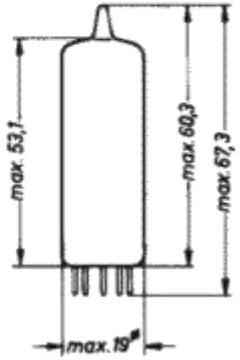
Art und Verwendung

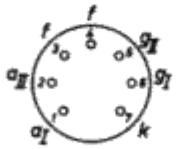
Doppeltriode mit gemeinsamer Kathode, besonders geeignet für bistabile Kippstufen und Multivibratoren in Rechen- und Zählgeräten.

Qualitätsmerkmale

Lange Lebensdauer (> 10 000 Std.)
Große Zuverlässigkeit (p ≈ 1,5 °/oo je 1000 Std.)
Enge Toleranzen
Zwischenschichtfreie Spezialkathode



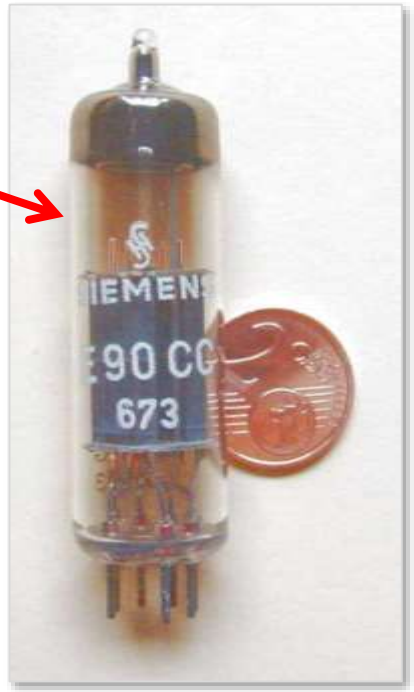




7-Stift-Miniatur

Sockel: Miniatur
Kolben: DIN 41537, Form A, Nenngröße 50

Gewicht: ca. 15 g
Einbau : beliebig



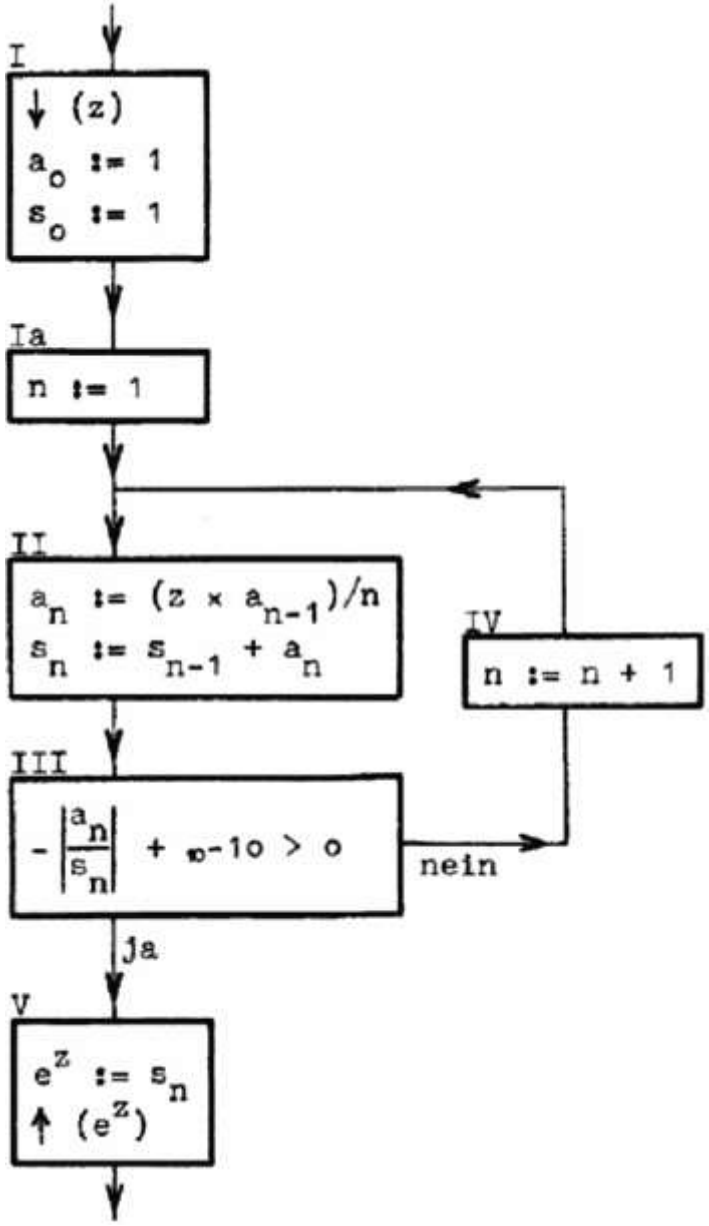
ERMETH-Programm für e^z

Institut für angewandte Mathematik

Programm 9999

Berechnung von e^z

000 0*			9999						
1 I	↓		0	(z)	S			z	
2	A		9001		S			a	Vorbereitung
3 (Ia)	S			s	S			n	
4* II	A			a	x			z	
5	:			n	S			a	eigentliche Rechnung
6	+			s	S			s	
7 III	A			a	:			s	
8*	|x|				-1				Diskriminante
9	+			10^{-10}	C+			V	
10 IV	A			n	+		9001		Erhöhen von n
1	S			n	C			II	
2* V	A			s	↑		0	(e^z)	Resultat anzeigen
3	Fin		0						
4	49	0	1000		00	0	0000		10^{-10} : Konstante
5									Plansumme
6*							-0		



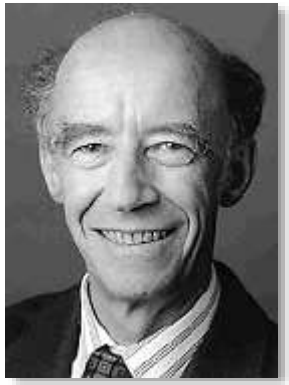
Entwicklung der ERMETH – Peter Läuchli erinnert sich

„1955 feierte die ETH ihr 100jähriges Jubiläum. Unser Rechner war leider noch keineswegs fertig. Da die Türen der Institute für Besucherströme geöffnet wurden, beeilten wir uns, wenigstens das Schaltpult der ERMETH soweit bereitzumachen, dass man auf den grossen Ziffernrädern 14-stellige Zahlen anzeigen konnte. [...]

Später, als die ERMETH schon einigermassen funktionierte, fanden Stock und ich einen logischen Fehler in der Schaltung für die Festkomma-Division, der sich aber nur in sehr seltenen Fällen auswirken konnte. Nach langem Abwägen entschlossen wir uns, auf die sehr mühsame und aufwendige Korrektur des Fehlers zu verzichten und unser Produkt mit diesem Makel behaftet den Benutzern zu überlassen."

Peter Läuchli (Jahrgang 1928) studierte Mathematik und Physik an der ETH Zürich („trotz meiner offiziellen Studienrichtung Physik tendierte ich mit der Zeit eher auf die mathematische Seite und erhielt auch die Erlaubnis, bei Herrn Professor Stiefel schriftlich zu diplomieren"). 1953 wurde er Assistent am Institut für Angewandte Mathematik bei Eduard Stiefel (1959 Dissertation „Iterative Lösung und Fehlerabschätzung in der Ausgleichsrechnung"); von 1964 bis 1993 war er Professor an der ETH (ab 1983 als Ordinarius für Informatik). 1968 gründete er zusammen mit Heinz Rutishauser und Niklaus Wirth die „Fachgruppe für Computerwissenschaften", aus der später schliesslich das Departement für Informatik an der ETH Zürich hervorging.

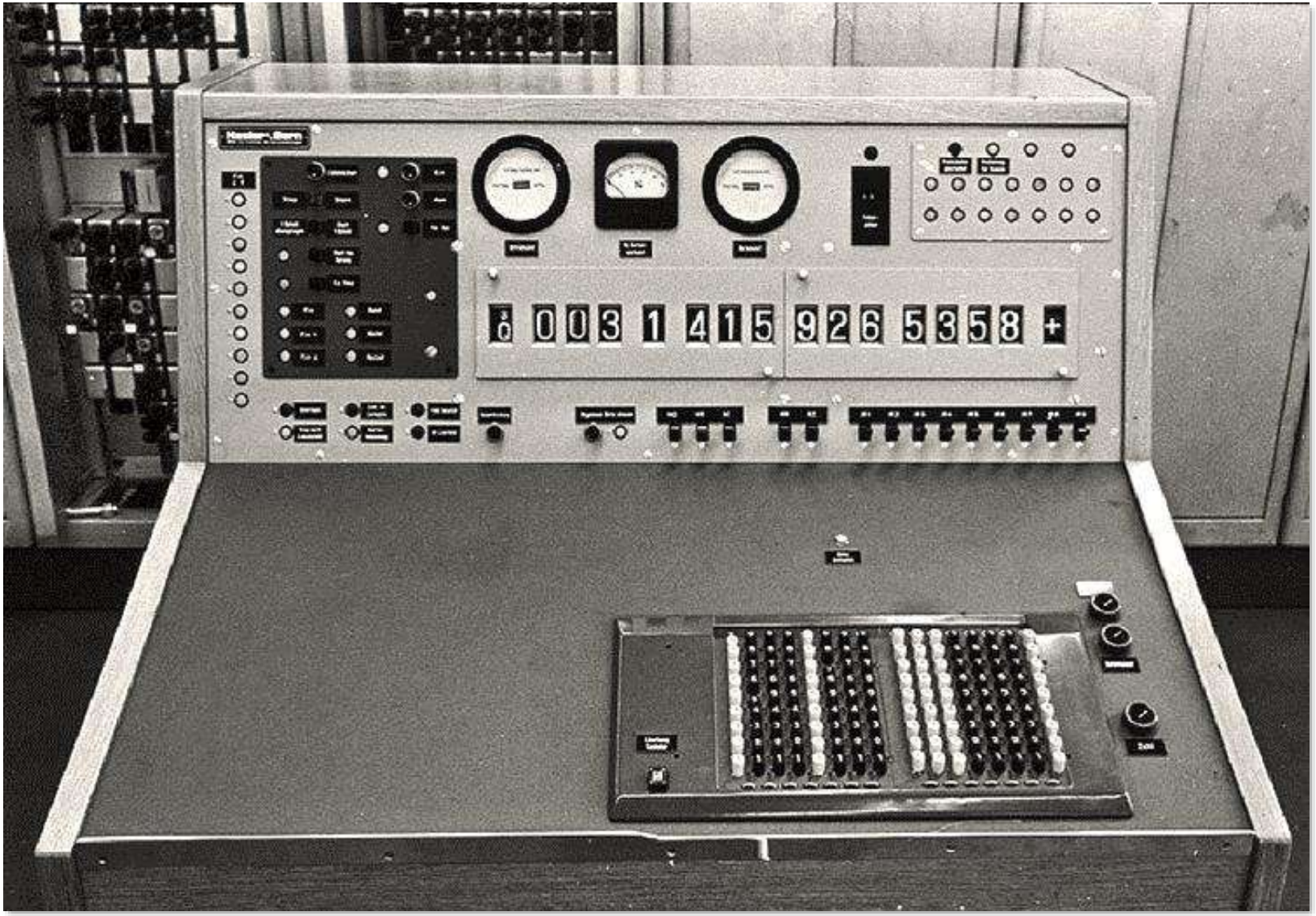
Nutzung der ERMETH – Jörg Waldvogel erinnert sich



Jörg Waldvogel, Jahrgang 1938 und emeritierter Professor am Seminar für Angewandte Mathematik der ETH Zürich, erinnert sich:

Zu meinem ersten Forschungsgebiet bin ich eher zufällig gekommen, nämlich durch das von Prof. Eduard Stiefel am Institut für Angewandte Mathematik der ETH 1961 angebotene Proseminar über ausgewählte Kapitel der Himmelsmechanik. Da der mathematische Stil von E. Stiefel mir sehr zusagte, bat ich ihn um ein Diplomthema. In meiner Diplomarbeit im Jahr 1962 berechnete ich auf der ERMETH Flugbahnen von der Erde zum Mond. Die Technik basierte auf der numerischen Integration von Systemen von Differentialgleichungen und war durch Handrechnungen, auch mit der Hilfe von mechanischen MADAS-Rechnern, nicht mehr zu bewerkstelligen. Die Berechnung einer Bahn brauchte auf der ERMETH mehrere Stunden, die ich nur nachts bekommen konnte. Auf einem Tisch im Maschinenraum schlief ich jeweils auf einer Luftmatratze während eines Runs. Jede Nacht um 0.30 Uhr ertönte das Warnsignal und die Rechnung stand still: Das Ausschalten des elektrischen Netzwerkes der Zürcher Trambetriebe hatte jeweils eine Spannungsschwankung verursacht, welche die Rechnung auf der ERMETH stoppte. Zum Glück konnte ich jeweils das Programm ohne Verlust am selben Ort wieder starten.

Die ERMETH-Konsole mit grossen Ziffernrädern

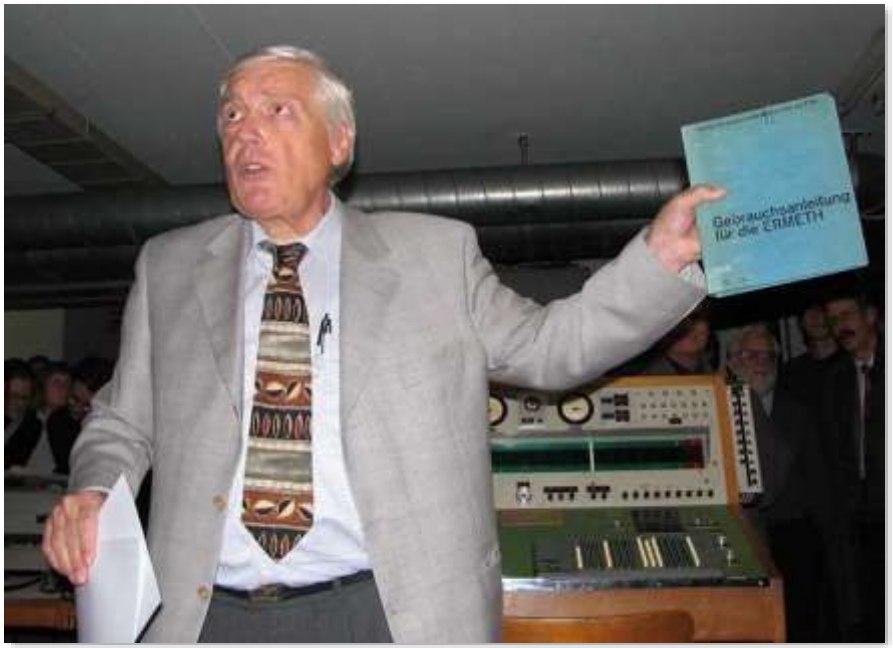


Prof. C. A. Zehnder mit ERMETH, Nov. 2006

In den Kellerräumen des IFW-Gebäudes der ETH Zürich bei der ERMETH-Übergabe an das Museum für Kommunikation in Bern

Carl August **Zehnder** war 1979–2003 Professor für Informatik an der ETH Zürich; bereits als Student programmierte er 1958 an der ERMETH

Die Bedeutung der ERMETH – C. A. Zehnder erinnert sich



„Wie wichtig ... [die ERMETH] für die Zürcher Hochschulen damals war, kann nur ermessen, wer die Alternativen dazu betrachtet. Ingenieure rechneten damals primär mit Rechenschieber (3-stellig) und Logarithmentafel (5-stellig), bevorzugt auch mit mechanischen oder elektromechanischen Addier- oder gar 4-Spezies-Rechenmaschinen. Von letzteren (Typ Madas) hatte um 1960 die gesamte ETH Zürich ungefähr 50 Stück (die meisten bei den Geodäten); eine einzige Division dauerte damit etliche Sekunden, alle Operanden mussten einzeln eingetastet, alle Ergebnisse von Hand aufgeschrieben werden.

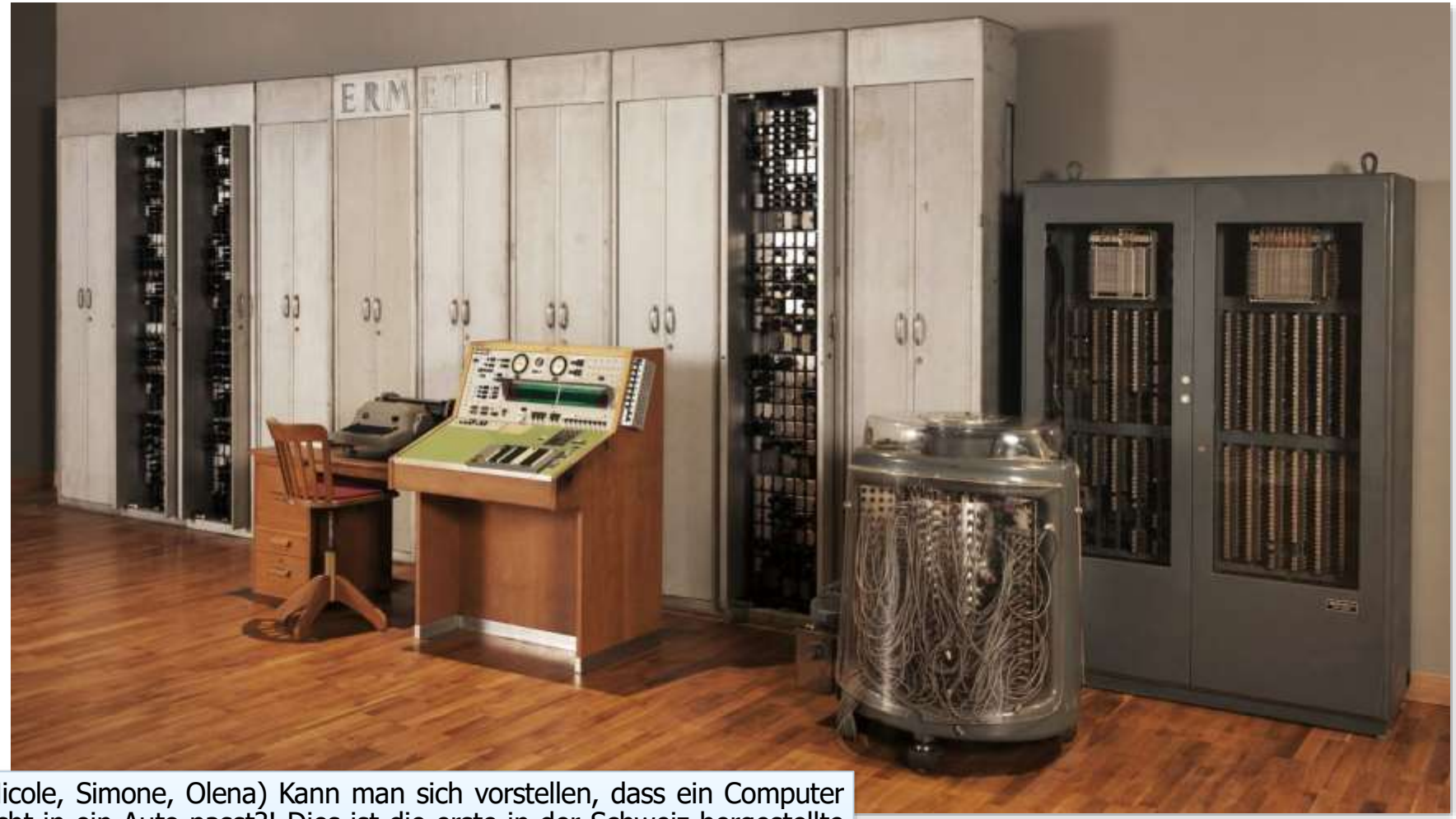
Demgegenüber bildete der Einsatz von Rechenautomaten auf jeden Fall einen gewaltigen Fortschritt, auch wenn bereits relativ einfache programmgesteuerte Berechnungen Stunden dauern konnten. Erst jetzt konnten nämlich ganze Operationsketten mit Schleifen, Alternativen und Unterprogrammen systematisch aufgebaut (programmiert) und dann vollautomatisch laufengelassen und auch variiert und wiederholt werden. Gleichzeitig liessen sich problemlos viel höhere Genauigkeiten erreichen (die ERMETH z.B. arbeitete 14-stellig dezimal). Dabei war sich damals jeder Benützer eines Rechenautomaten ständig nicht nur der Vorteile, sondern auch sehr direkt der Grenzen dieses Instruments bewusst. Der Wissenschaftler suchte den Lösungsweg (Algorithmus) für sein Problem meist selber, er programmierte diesen nachher aus, tastete das Programm und die Daten in die Maschine, testete das Programm aus und bediente wiederum selber das gelegentlich auch störrische Gerät – manchmal auch nachtsüber. Die knappe Grösse des Arbeitsspeichers, die Langsamkeit der Maschine und des Druckers und ähnliche Grenzen zwangen zu äusserster Zurückhaltung bei der Abgrenzung des zu rechnenden Problems; nicht zu gross und nicht zu kompliziert musste es sein.“ [C.A. Zehnder (1992): Frühe Gefechtssimulationen in der Schweiz.]

ERMETH im Museum für Kommunikation, Bern
(2006 – 2016)

(Filip, Marco, Joel) Die ERMETH war der erste in der Schweiz gebaute Computer. Er lief unter dem Saurier-Motto: gigantischer Körper, winziges Hirn. Die ERMETH konnte trotz ihrer eindrucksvollen Grösse nur rechnen. Aus heutiger Sicht ist diese Maschine völlig unnütz. Ein moderner Taschenrechner ist deutlich schneller, kann viel mehr, hat mehr Speicher und man kann ihn sogar programmieren. www.digi-news.ch/schulen/20080228/

https://iart.ch/documents/10176/37731/As_Time_Goes_Byte_-_ERMETH.jpg

ERMETH im Museum für Kommunikation, Bern



http://blog.nationalmuseum.ch/wp-content/uploads/2018/01/ERMETH-Farbe-MFK.jpg

(Nicole, Simone, Olena) Kann man sich vorstellen, dass ein Computer nicht in ein Auto passt?! Dies ist die erste in der Schweiz hergestellte elektronische Rechenmaschine. Die ETH in Zürich konstruierte die ca. 7.5m lange, 3.5m hohe und 40cm breite Maschine. Trotz dieser Grösse hatte sie praktisch keinen Nutzen. www.digi-news.ch/schulen/20080228/

Mehr zur ERMETH: Hans Neukom: *ERMETH: The First Swiss Computer.* IEEE Annals of the History of Computing, 27(4), Oct. 2005, 5-22

Zwischen 2007 und 2016 haben Schulklassen das Museum für Kommunikation in Bern besucht und dabei u.a. die ERMETH besichtigt. Ihre Erinnerungen und Eindrücke haben die Schülerinnen und Schüler bei digi-news.ch notiert. Hier einige Auszüge, die die ERMETH betreffen:

(Melanie & Karin) Vor allem Ingenieure, Mathematiker, Chemiker, Maschinenbauer und die Armee benutzten diese Maschine, obwohl man damit nur rechnen konnte. Sie wiegt rund 3.5 Tonnen, hat aber nur einen Speicherplatz von 80 Kilobytes. Wir reden immer vom Urzeitmonster, dabei ist diese Kiste bloss 52 Jahre alt. Also etwa gleich alt wie unsere Eltern.

(Anina, Marietta) Er war gross, hässlich und einige Tonnen schwer.

(Funda, Burcu, Vanessa) Ermeth ist der erste in der Schweiz gebaute Computer. Es sah alles kompliziert aus und war sehr gross und teuer. Der Preis war 1000000.- Franken. Das Gerät war damals einzigartig und nicht erhältlich.

(Jennifer, Leonie, Milena) Es ist erstaunlich was diese Maschine alles benötigt, um zu funktionieren. Unter anderem benötigt die Maschine sehr viel Strom und Energie. Wenn es Sie interessiert kommen Sie doch bitte ins Kommunikationsmuseum und erkundigen Sie sich selbst.

(Sarah und Celine) Der erste Computer der Welt wurde 1952 bis 1955 gebaut. Der Wichtigste von den Mitarbeitern war: Nickolas Wirth, sowie auch der Chef Proffesor Speiser. Hier konnte man nur Zahlen schreiben, er konnte dadurch nur für Mathematik eingesetzt werden. Es ist eine Abkürzung für: Erste Rechenmaschine der E.T.H Zürich. Dies ist die Firma/Universität die den Computer gebaut hat.

(Noddy, Linus, Joël) Früher war die Benutzung eines Computer viel anspruchsvoller als heutzutage. Deshalb spielte die Bedienungsanleitung eine sehr grosse Rolle.

(Stefanie, Fabienne, Simona) Damals war die Erfindung des Computers nur für sehr wenige und sehr reiche Menschen von Bedeutung und das Mitführen eines Computers in Form eines Laptops war noch undenkbar. Weil in den Röhrchen für ihre Verhältnisse sehr viel Spannung herrschte, kam es schnell zu Überhitzungen, die die Lebensdauer der Elektronenröhrchen verkürzten. Um solche Ausfälle zu vermeiden wurde weitergeforscht.

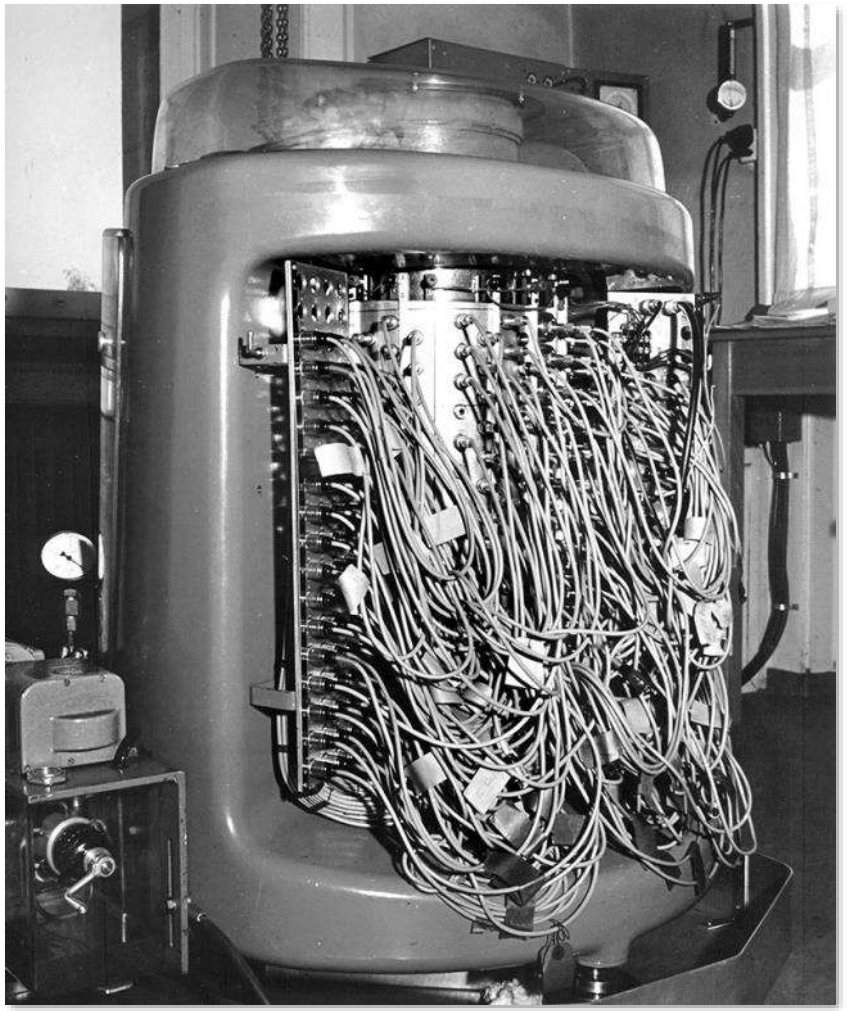
(Severin, Robert, Marc) Der Rechner ERMETH wurde zum Rechnen gebraucht. Da es nur einen auf der Welt gibt kann man ihn nicht kaufen.

(Susanne, Njomza und Annisha) Die ERMETH war nicht geeignet für anspruchsvolle Arbeiten. Wir haben eine Umfrage gestartet, bei der wir den Leuten ein Bild der ERMETH gezeigt haben. Die meisten der befragten Personen wussten nicht, was auf dem Bild abgebildet ist und daher auch nicht, wofür man diesen Apparat verwendete. Viele vermuteten jedoch, dass es sich dabei um eine Telefonzentrale oder Radiostation handelt.

(Sarah und Miryam) Die Maschine war hauptsächlich von Nutzen als Heizung im Winter wenn es kalt war.

Der ERMETH-Magnettrommelspeicher – Ambrosius Speiser erinnert sich

„...etwas, das uns wirklich Sorgen gemacht hat, das uns gelegentlich den Schlaf gekostet hat. Es ist eindeutig die Magnettrommel, der Speicher. Das war ein sehr schwieriges, gleichzeitig mechanisches und elektrisches und elektronisches Problem. Das ist eine Trommel, die rotiert mit 6000 Umdrehungen pro Minute. Die Magnetköpfe sind nur ein hundertstel Millimeter von der Oberfläche der Trommel entfernt. Also, die geringste Wärmeausdehnung würde dazu führen, dass die Magnetköpfe berühren und verkratzen und die Oberfläche beschädigen. Wir haben es glücklicherweise gelöst, aber es hat uns ziemlich viel Kopfzerbrechen bereitet. ... Wenn ich jetzt ... die Ermeth anschaue, und im Besondern die Magnettrommel, so muss ich mir sagen, ich verstehe nicht ganz, dass wir das gewagt haben. ... Es bestand ein gewisses Risiko, dass es ein Fiasko wird, und zwar das Ganze. Dann wären wir gegenüber dem Geldgeber dagestanden, der uns gegen eine Million Franken gegeben hat, und das Ergebnis ist fast Null. Aber damals, in diesem Alter, da war ich noch nicht 30 Jahre alt, und da ist man eben etwas sorgloser."



Ambrosius Speiser (1922 – 2003)

- Studium der Elektrotechnik an der ETH
- 1950 Dissertation bei Eduard Stiefel
  - *Entwurf eines elektronischen Rechengerätes unter besonderer Berücksichtigung der Erfordernis eines minimalen Materialaufwandes bei gegebener mathematischer Leistungsfähigkeit*
- Technische Leitung der ERMETH-Entwicklung
- Gründungsdirektor (1955) des IBM-Forschungslabors in Rüschlikon
- Gründungsdirektor (1966) des Forschungszentrums von Brown, Boveri & Cie. (später: ABB) in Dättwil bei Baden
- 1986 Ehrendoktorwürde der ETH Zürich

Ambrosius Speiser: Lebenslauf (aus der Dissertation)

Ich wurde am 15. November 1922 in Basel geboren; mein Vater war damals Inhaber eines Import- und Exportgeschäftes in London, wo ich die ersten 9 Jahre meines Lebens verbrachte und die ersten Schuljahre absolvierte. 1931 siedelten meine Eltern nach Baden über. Dort besuchte ich die Primarschule und die Bezirksschule, und anschliessend die Aargauische Kantonsschule in Aarau, an welcher ich 1942 die Maturität (Typus B) erwarb.

In den folgenden 2 1/2 Jahren leistete ich Militärdienst und avancierte zum Leutnant der Fliegerabwehrtruppe; ferner studierte ich ein Semester an der Universität Zürich Mathematik und Physik. 1944 immatrikulierte ich mich an der Abteilung IIIb der Eidgenössischen Technischen Hochschule, wo ich den normalen Studiengang ohne Unterbruch absolvierte und Ende 1948 das Diplom als Elektro-Ingenieur erhielt. Während des Studiums befasste ich mich zu Hause viel mit Versuchen auf dem Gebiet der Hochfrequenztechnik; unter Anderem konstruierte und betrieb ich eine Kurzwellen- Sende- und -Empfangs-Station. Ausserdem studierte ich das Gebiet der elektrischen Rechenhilfsmittel gründlich; das Thema meiner Diplomarbeit entstammte diesem Gebiet.

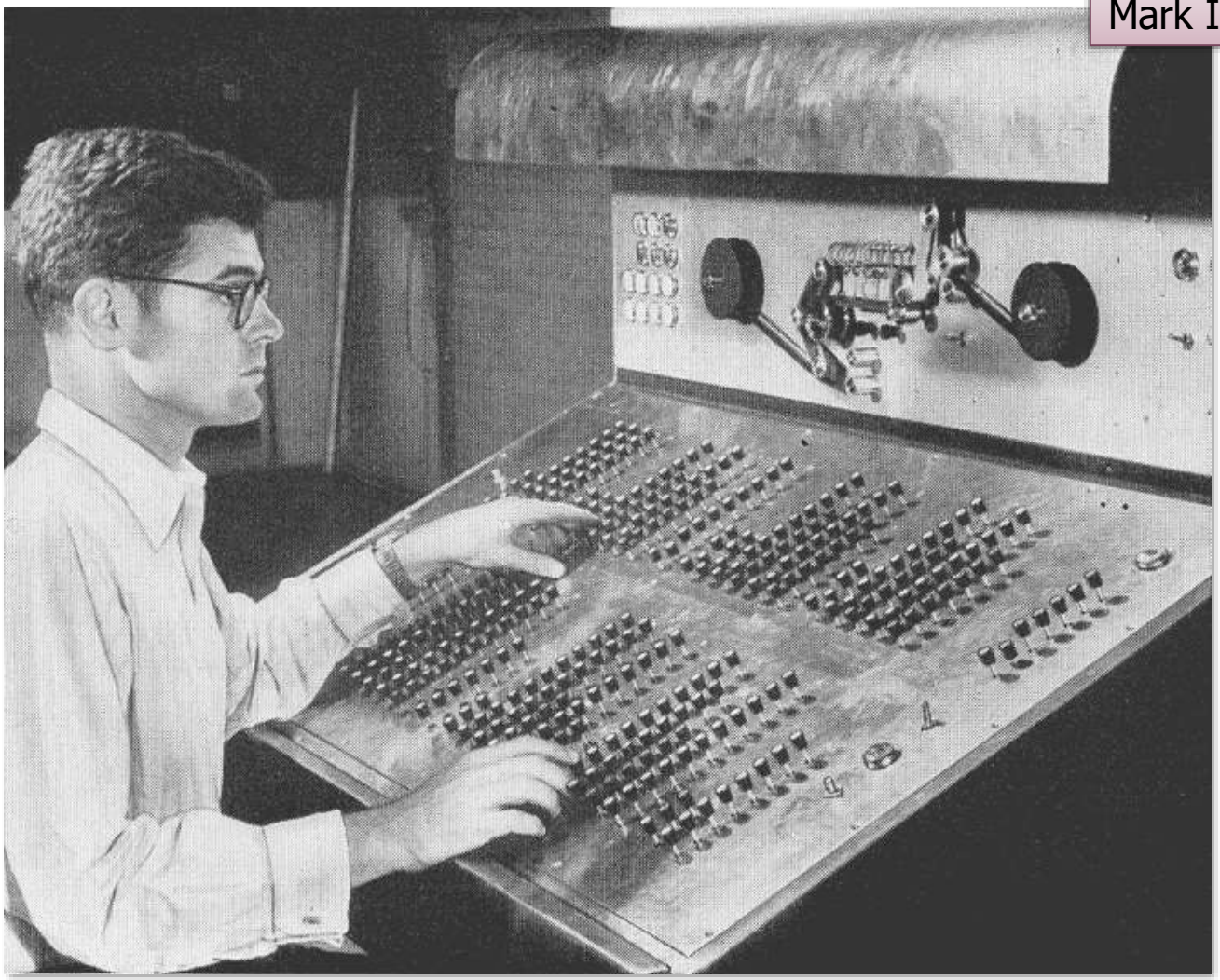
1947 und 1948 verbrachte ich je zwei Monate in England, um mich über ausländische Fortschritte auf dem Gebiet der Elektronik und der elektrischen Rechenhilfsmittel zu orientieren. Das Jahr 1949 verbrachte ich in den Vereinigten Staaten mit dem besonderen Zweck der Ausbildung im Bau programmgesteuerter Rechenautomaten. Dies geschah vornehmlich an der Columbia University (New York), der Harvard University (Cambridge) und dem Institute for Advanced Study (Princeton, New Jersey); doch besuchte ich auch an verschiedenen andern Orten Persönlichkeiten und Betriebe.

Seit Januar 1950 bin ich als wissenschaftlicher Mitarbeiter am Institut für angewandte Mathematik der E.T.H. beschäftigt; neben anderen Arbeiten beendete ich in dieser Zeit meine Promotionsarbeit, zu welcher mir Herr Professor Stiefel zu Beginn meines Amerika-Aufenthaltes das Thema gegeben hatte.

Zürich, 2. Juni 1950

Ambrosius Speiser

Speiser an der „coding box“ des Mark III-Computers in Harvard.



The “coding box” consists of a six by ten foot panel of over 200 keys, each with a number or mathematical symbol on it. Using the keyboard with its familiar symbols, a mathematician can record on a magnetic tape all the commands the machine needs to solve his problem. Essentially, he “copies” his equations on the keys of the coding machine. By means of the “coding box,” an operator can feed a problem into Mark III in a fraction of the time required by Mark II and any other calculating machines in use at the present time. *The Harvard Crimson, Sep. 26, 1949*

„Coding box“ oft „Chiffriermaschine“ auf Deutsch. „Übrigens hatte der deutsche Ingenieur K. Zuse auf seiner im Jahre 1945 fertigen programmgesteuerten Maschine bereits ein ähnliches Planfertigungsgerät.“ [Rutishauser]

Ist die Informatik in der Schweiz verschlafen worden?

„Es ist eine Tatsache, dass die grosse Bedeutung der kommerziellen Datenverarbeitung als Lehr- und Forschungsgebiet nicht erkannt wurde, und zwar nicht nur in ihren Anfängen, sondern auch noch zu einer Zeit, als ihr Umfang jenen des wissenschaftlichen Rechnens weit hinter sich gelassen hatte. Der Computer wurde nur als Instrument für technisch-wissenschaftliche Berechnungen und, schwergewichtig, für die Forschung in numerischer Mathematik gesehen. Tatsächlich dauerte es viel zu lange, bis man bereit war, die Informatik als ein Lehr- und Forschungsgebiet anzuerkennen, das wirklich alle Anwendungen der Computer einschliesst. Angesichts der grossen praktischen Bedeutung des Gebietes haben diese Vorgänge erhebliche Nachteile auf der gesamtschweizerischen Ebene zur Folge gehabt. Aus dieser Sicht hat der Vorwurf, die Informatik sei verschlafen worden, leider seine Berechtigung.“

Ambrosius Speiser: 95 Semester ETH – der Weg zur Informatik, ETH Zürich, 1992

Informatik-Innovationen aus der Schweiz

Gregor Henger, NZZ vom 11. Januar 2008 [Auszug]

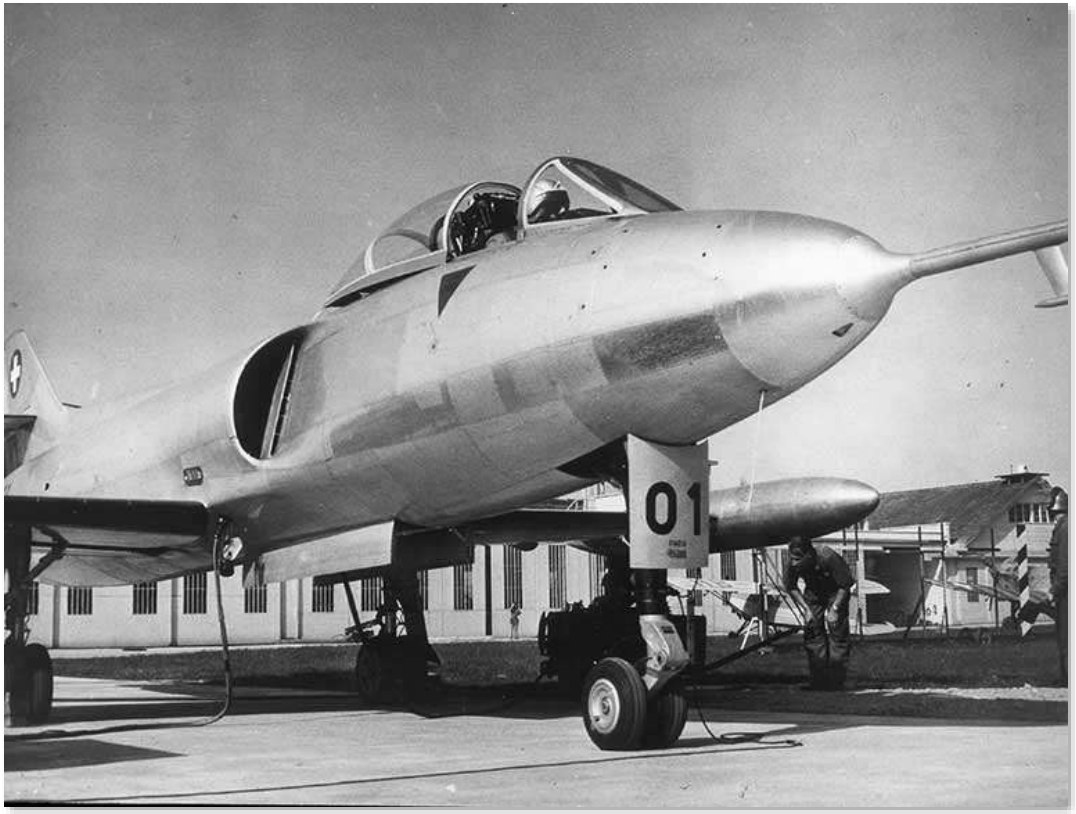
Die Informatik in der Schweiz verdankt ihre Geburt und frühe Blüte vor allem der Weitsicht und Tatkraft des Mathematikprofessors Eduard Stiefel. Dieser gründete 1948 das Institut für angewandte Mathematik der ETH. Er beschaffte 1950 aus dem kriegsverwüsteten Deutschland mietweise die in einem Pferdestall eingelagerte, mit elektromechanischen Relais funktionierende Rechenanlage Z4 des Computerpioniers Konrad Zuse und engagierte ihn als Berater. Zuvor hatte Stiefel seine Assistenten, den Mathematiker Heinz Rutishauser und den Elektroingenieur Ambros Speiser, auf mehrmonatige Erkundungsreise in die USA geschickt, wo diese sich frühe Computer anschauten.

Grande-Dixence-Staumauer im Wallis, 1962 (Bild: ETH-Bibliothek)

Durch den Handstreich Stiefels wurde die ETH zur ersten Hochschule auf dem europäischen Kontinent, in der ein Computer zum Einsatz kam. Trotz der vergleichsweise geringen Rechengeschwindigkeit der Z4 war die technische und wissenschaftliche Ausbeute des Z4-Betriebs gross. Die Maschine wurde zu Berechnungen für die Statik der Staumauer der Grande Dixence

im Wallis und für Eisenbahnbrücken verwendet, zur Bestimmung kritischer Wellendrehzahlen für Turbinen und Generatoren in Kraftwerken sowie zur Analyse von Flattererscheinungen an den Flügeln im später abgebrochenen P-16-Projekt zum Bau eines schweizerischen Düsenjägers.

Die wissenschaftliche Z4-Ausbeute erlangte internationale Ausstrahlung. Heinz Rutishauser erarbeitete an der Z4 das Konzept des Compilers, einer Schlüsselkomponente beim Einsatz aller höheren Programmiersprachen. Stiefel erfand eine Näherungsmethode zur Lösung von Gleichungssystemen, deren theoretisch genaue Berechnung infolge hohen Aufwands vorher gar nicht versucht worden war. Und Rutishauser ergriff die Initiative zur Entwicklung der wissenschaftlich-technischen Programmiersprache Algol, deren erste Version an einer internationalen Konferenz 1958 an der ETH in Zürich verabschiedet wurde.



Der Schweizer Jagdbomber P-16, 1955 (Bild: ETH-Bibliothek)

Stiefel war auch die treibende Kraft hinter dem 1955 begonnenen Eigenbau der legendären Ermeth, der elektronischen Rechenmaschine der ETH. Rutishauser entwickelte die Ermeth-Arithmetik, während Speiser als technischer Verantwortlicher die Architektur konzipierte und den

Bau der Maschine leitete. Er wurde 1957, noch vor der endgültigen Fertigstellung der Ermeth, zum Gründungsdirektor des ersten nichtamerikanischen IBM-Forschungslabors berufen. […]

1968 wurde Niklaus Wirth als Professor für Computerwissenschaft an die ETH berufen. Schon zwei Jahre später präsentierte er die Programmiersprache Pascal. Sie erreichte bald weltweite Popularität. Bei einem Forschungsaufenthalt im Palo Alto Research Center (PARC) von Xerox lernte Wirth die dort entwickelte Alto-Arbeitsstation kennen, die über eine grafische Benutzeroberfläche verfügte und netzwerkfähig war. Er beschloss, diese zukunftsträchtigen Errungenschaften im Rahmen eines Forschungsprojekts am Institut für Computersysteme aufzugreifen. Das Lilith genannte Projekt begann 1977. Wirth und seine Forschungsgruppe bauten die Hardware und programmierten die gesamte Arbeitsstation – den Mikrocode des Prozessors, das Betriebssystem sowie Anwendungsprogramme – durchgängig in der auf Pascal beruhenden Programmiersprache Modula-2. Dazu wurde am Institut für Computersysteme das erste Ethernet-Lokalnetz der Schweiz installiert. […]

Nach mehreren vergeblichen Vorstössen gelang es Wirth und seinen Professorenkollegen von der Fachgruppe der Computerwissenschaften 1981 endlich, die ETH-Schulleitung zur Einrichtung eines regulären Studienganges für Informatik mit einem Diplomabschluss zu bewegen. Zuvor war ein Nebenfach-Programmierunterricht auf verschiedene Fachrichtungen verteilt. Die neugeschaffene Abteilung zog schnell zahlreiche Studierende an und wurde innerhalb weniger Jahre zum Departement Informatik ausgebaut. […]

Im Lauf seiner langen Tätigkeit an der ETH hatte Wirth höchste akademische Ehren erlangt, er erhielt 1984 den als Informatik-Nobelpreis geltenden Turing Award und 1988 den Computer Pioneer Award des amerikanischen Institute of Electrical and Electronic Engineering (IEEE) sowie später eine lange Reihe von Ehrendoktortiteln. […]

www.nzz.ch/aktuell/startseite/informatik-innovationen-aus-der-schweiz-1.649089

ETH in den 1950ern – Heinz Waldburger erinnert sich

Quelle: Herbert Bruderer: Konrad Zuse und die Schweiz. Oldenbourg-Verlag, 2012

- Studium an der ETH, ab 1953 Hilfsassistent von Rutishauser und Stiefel
  - Später Informatikchef der Firma Nestlé
- „Im ETH-Vorlesungsverzeichnis des Sommersemesters 1952 wurde angekündigt: *Praktikum an der programmgesteuerten Rechenmaschine Z4 (Zuse) am Institut für angewandte Mathematik. Prof. Stiefel, Dr. Rutishauser, Dr. Speiser.*"
- „Meine Vorlesungsnotizen zeigen, dass Stiefel nicht in Erscheinung trat. Rutishausers 56 Seiten betrafen die folgende Themen: Struktur einer programmgesteuerten Rechenmaschine, externes Rechenprogramm, Rechenbefehle, Flussdiagramm, numerische Anwendungen, sauber strukturiertes Programmieren, Rechnen mit Befehlen. Hinzu kamen die 16 Seiten der *Bedienungsanweisung Z4.* […] Die 20 Seiten Notizen zu Speisers Vorträgen erklärten das Funktionieren von elektrischen und elektronischen digitalen Schaltungen und deren Kombinationen für die Rechen- und Datenübertragungsfunktionen…"

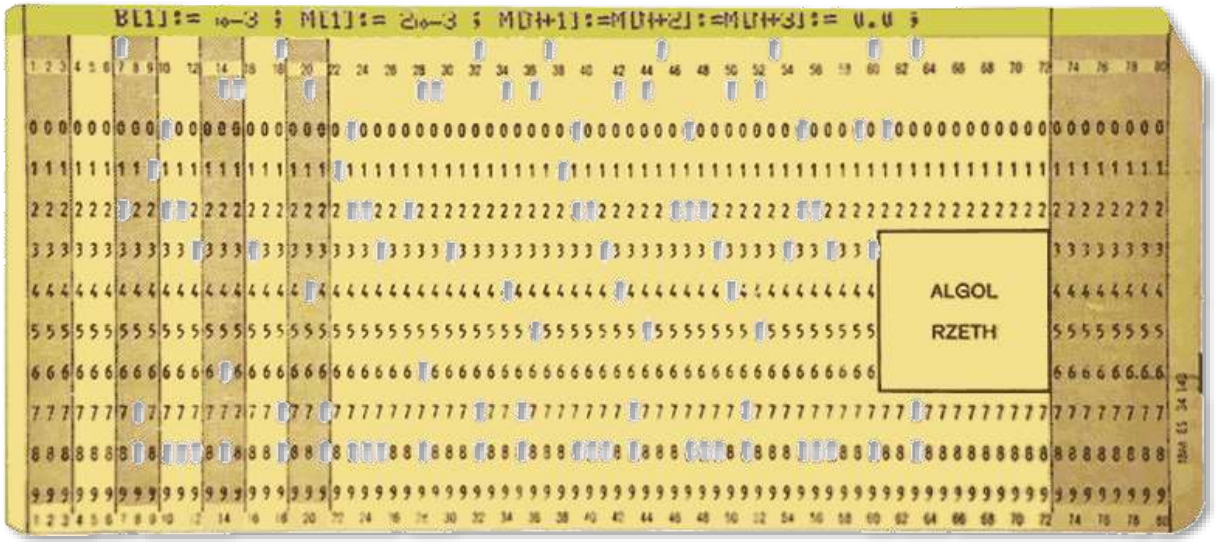
ETH in den 1950ern – Heinz Waldburger erinnert sich

- „Das Wechselbad von Theorie und Praxis an der ETH war äusserst anregend. Rutishauser war dabei ein echter Meister, zurückhaltend, zusammenführend, stets hilfsbereit. Zuses geniale Erfindungen und sein einzigartiger Rechenautomat liessen an der ETH dank Stiefel, Rutishauser und Speiser eine eigenständige schweizerische Informatikkultur wachsen.
- Die Studierenden in den ersten Z4- und ERMETH-Jahren waren sich wohl nicht bewusst, dass meines Wissens 1952 in Zürich die erste Informatikvorlesung auf dem europäischen Kontinent stattfand."
- Nicht die beim Bau von Prozessoren und Magnetspeichern erworbenen technischen Kenntnisse überlebten, sondern Rutishausers Programmiergrundsätze. Sie flossen in die von ihm geschaffene Programmiersprache Algol ein.

Im Sommersemester 1954 hielt Rutishauser dann eine zweistündige Lehrveranstaltung „Einführung in die Praxis des programmgesteuerten Rechnens", im Wintersemester 1955/56 eine zweistündige Lehrveranstaltung „Programmgesteuertes Rechnen". Zum Weiterlesen der Geschichte der Computer und der Informatik an der ETH Zürich: Andreas Nef, Tobias Wildi: *Informatik an der ETH Zürich 1948–1981*, Preprints zur Kulturgeschichte der Technik / 2007 / 21, ETH Zürich, Institut für Geschichte / Technikgeschichte. Auch in: Peter Haber (Hrsg.): *Computergeschichte Schweiz – eine Bestandesaufnahme*, Zürich, 2009.
www.tg.ethz.ch/dokumente/pdf_Preprints/Preprint21.pdf

Wie es nach der ERMETH weiterging: Das ETH-Rechenzentrum

Ende der 1950er-Jahre stiess die Rechenkapazität der ERMETH an ihre Grenzen. Als verschiedene ETH-Institute Anträge zum Kauf eigener Computer stellten, wurde es nötig, den Computerbetrieb an der ETH neu zu gestalten. Im Mai 1963 gründete die ETH ein Rechenzentrum, die die Bedarfe zusammenfassen sollte, denn „Doppelspurigkeiten und anderweitig ungerechtfertigte Anschaffungen sind unbedingt zu vermeiden", wie ein Schulratsprotokoll schon 1959 vermeldete. Der Anstoss dazu kam vom Leiter des Instituts für angewandte Mathematik, Prof. Stiefel, der als Vorsitzender der technischen Kommission für Rechengeräte an der ETH agierte. Die Betreuung der Rechenanlage und ihrer Benutzer oblag fortan nicht mehr dem Institut für Angewandte Mathematik, sondern dem neu geschaffenen Rechenzentrum, dessen erster Leiter Alfred Schai wurde, der schon am ERMETH-Projekt beteiligt war.



Amtssprache des RZ-ETH ist die Formelsprache „ALGOL". Übungsarbeiten der Studierenden werden in „ALGOL" geschrieben und von der Rechenanlage automatisch korrigiert.
-- Schweiz. Bauzeitung, 30.9.1967

Algol-Lochkarte des ETH-Rechenzentrums

Das ETH-Rechenzentrum (2)

1964 wurde mit einer vier Millionen Franken teuren CDC 1604A von Control Data der erste kommerzielle Computer an der ETH installiert, im gleichen ehemaligen Zeichensaal des Hauptgebäudes, wo zuvor die ERMETH ihre Dienste geleistet hatte. Mit ein Grund für die Wahl der CDC 1604A war der sehr leistungsfähige Algol-Compiler der Maschine. Alfred Schai schrieb dazu: „Die ersten Betriebsjahre des RZETH standen ganz im Zeichen der Programmiersprache Algol 60, an deren Entwicklung die ETH mit H. Rutishauser massgebend beteiligt war. [...] Für die meisten an dieser Sprachentwicklung nicht direkt Beteiligten gestaltete sich das Aneignen dieser Sprache als recht mühsam. Mit speziellen Algolprogrammierkursen und mit der Aufnahme des Programmierens in die Lehrpläne sowie vereinfachter Darstellung der Algolsyntax mit Syntaxdiagrammen versuchte man Algol 60 in den Griff zu bekommen. Das gelang in der Folge recht gut, und bald erkannte und schätzte man ihre Qualitäten für Numerik, Programmiermethodik und Compilertechnik.“ Mit der CDC 1604A wurde auch an der ETH das Wort „Computer“ salonfähig, vorher sprach man von „programmierbaren Rechnern“ oder „Rechenautomaten“.

Eine CDC 1604A, ca. 1965 (hier allerdings im Rechenzentrum der TH Hannover)

Das ETH-Rechenzentrum (3)

Obwohl die CDC 1604A rund 400 Mal leistungsfähiger als die ERMETH war, musste nach einem halbjährigen Einschichtbetrieb auf den Zweischichtbetrieb übergegangen werden. Die Benutzung der Anlage, vor allem durch die Chemie und die Reaktorforschung, entwickelte sich zwischen 1964 und 1966 derart stürmisch, dass die Anlage bereits nach zwei Betriebsjahren vollkommen überlastet war. Das Rechenzentrum erhielt Anfang der 1970er-Jahre nicht nur ein neues eigenständiges Gebäude in der Clausiusstrasse, sondern auch einen neuen „Riesencomputer“, ein CDC 6400/6500-Doppelsystem.

1971: Das Rechenzentrum in der Clausiussstrasse

Am Ende der 1960er-Jahre kam die „Time Sharing“-Technik auf. Das Schulratsprotokoll vom 4.2.1967 vermerkt dazu: „Dieses Prinzip der Computer-Benützung wird als der eigentliche Sprung nach vorn bezeichnet und es wird der Erwartung Ausdruck gegeben, dass die durch Time Sharing gewährleistete Benützung des Computers als Diskussionspartner bei der Ausarbeitung von wissenschaftlichen Theorien und technischen Entwürfen von grösstem Einfluss auf Wissenschaft und Technik sein wird. Das Prinzip besteht, einfach ausgedrückt, darin, dass an einen zentralen Computer die Benützer dezentralisiert, durch Telefon- oder Drahtleitungen, verbunden sind. Jeder Benützer hat praktisch sofortigen Zugang zur Anlage und die Resultate werden ihm praktisch ohne jeden Zeitverlust geliefert.“

Das ETH-Rechenzentrum (4)

1970: Aufbau des CDC 6400/6500-Computers von Control Data im neuen Rechenzentrumsgebäude („RZ“) in der Clausiussstrasse 59

Das ETH-Rechenzentrum (5)

Allerdings war die Realisierung von Time Sharing komplizierter als gedacht. Rechenzentrumsleiter Alfred Schai schreibt rückblickend dazu: „Das Betriebssystem gestattete zwar diese Betriebsart; die Zuverlässigkeit und Beeinträchtigung der gesamten Systemleistung aber waren so schlecht, dass die Computerkommission im April 1971 beschloss, ein eigenes, nicht voll-interaktives, aber effizientes Konsolesystem in Betrieb zu nehmen.“

Im Erdgeschoss des Gebäudes befand sich der Computerraum mit einer Empore als Besuchergalerie

Das ETH-Rechenzentrum (6)

„Programme auf Lochkarten gestanzt, im Rechenzentrum abgegeben und nach ein bis zwei Tagen ein Resultat oder aber eine Fehlermeldung auf einem Stück Endlospapier zurück erhalten." -- Thomas Ottmann

Kundenbetrieb im „Lochkartenraum" des Rechenzentrums (vgl. heute Hörsaal F 21)

Seit 1964 wurden an der ETH Teile der Studentenadministration automatisiert. Die Rechenzentrumscomputer wurden nun nicht mehr nur für Forschungszwecke, sondern auch für administrative Aufgaben genutzt. Zunächst erfolgte die Automatisierung der Prüfungspläne. In einem Schulratsprotokoll von 1966 wird vermerkt: „Da und dort wurde der Wunsch geäussert, zwischen zwei Prüfungen sollten zwei prüfungsfreie Tage eingeschoben werden, was jedoch nach der neuen Regelung weniger leicht realisierbar ist als früher, bei Aufstellung der Prüfungspläne durch den Computer kann nicht allen Wünschen entsprochen werden." Das ETH-Rechenzentrum sah seine Aufgabe damals nicht nur im Betrieb der Computer, sondern bot den Nutzern auch Dienstleistungen an. Dazu gehören die Syntaxfehler-Suche („der Kunde darf, zu den angegebenen Zeiten, die RZ-Beratung beanspruchen, wenn er Schwierigkeiten beim Auffinden und Beheben von Fehlern hat"), die System-Beratung („Arbeiten mit Magnetbändern, Lochstreifen, Permanent-Files und Intercom"), die Numerische Beratung („die Beratergruppe der Fachgruppe für Computer-Wissenschaften nimmt sich der numerischen Probleme der Kunden an und hilft ihnen mit Hinweisen auf bessere numerische Lösungsmöglichkeiten beim Einsparen von Rechenzeit und beim Erreichen einer angemessenen numerischen Genauigkeit – sofern der Kunde die Ratschläge auch annimmt; ausser den numerischen Fragen behandeln die Herren aus dieser Beratungsgruppe auch sehr schwierige

Das ETH-Rechenzentrum (7)

und komplizierte Algol-Probleme“) sowie die Datenverarbeitende Beratung („alle Probleme, die mit der Verarbeitung von grossen Datenmengen in Zusammenhang stehen“). Generell werden die Ratsuchenden dann noch gebeten, „Nachsicht walten zu lassen, wenn ein Berater das Problem nicht gleich lösen kann, da die Berater weder Götter noch Halbgötter sind“.

Das Rechenzentrum als organisatorische Einheit wurde 1986 aufgelöst und in die neu gegründete Abteilung *Informatikdienste* überführt. Noch Jahrzehnte später stehen viele ETH-Server gut geschützt in den tiefen Kellerräumen des RZ.

CDC 6400 / 6500 des ETH-Rechenzentrums , ca. 1975

Wer erfand das Programmieren?

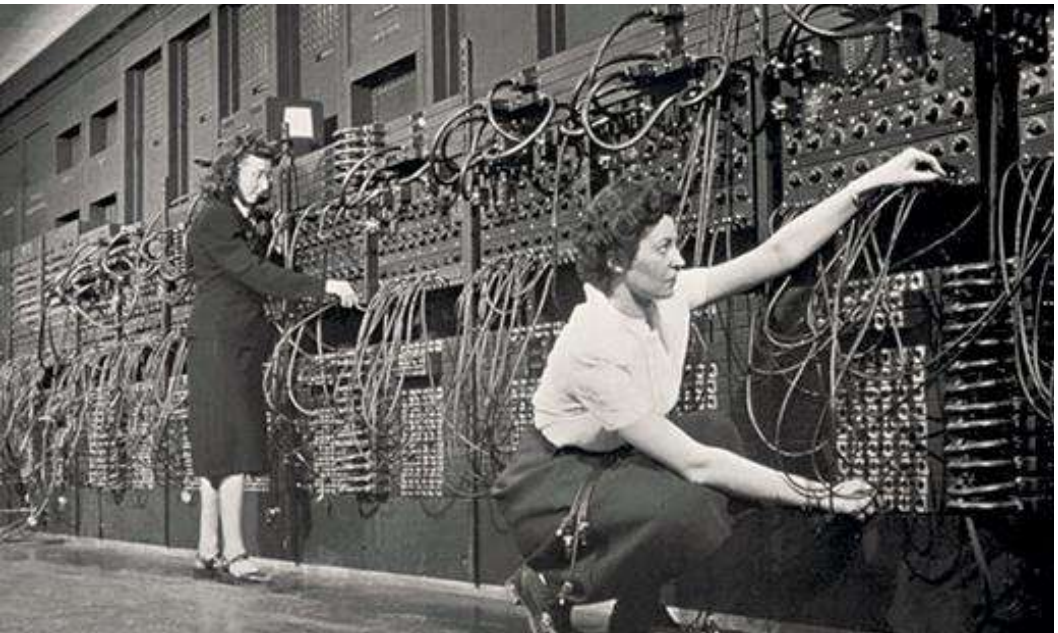
Noch mehr Historie: Die Erfindung des Programmierens

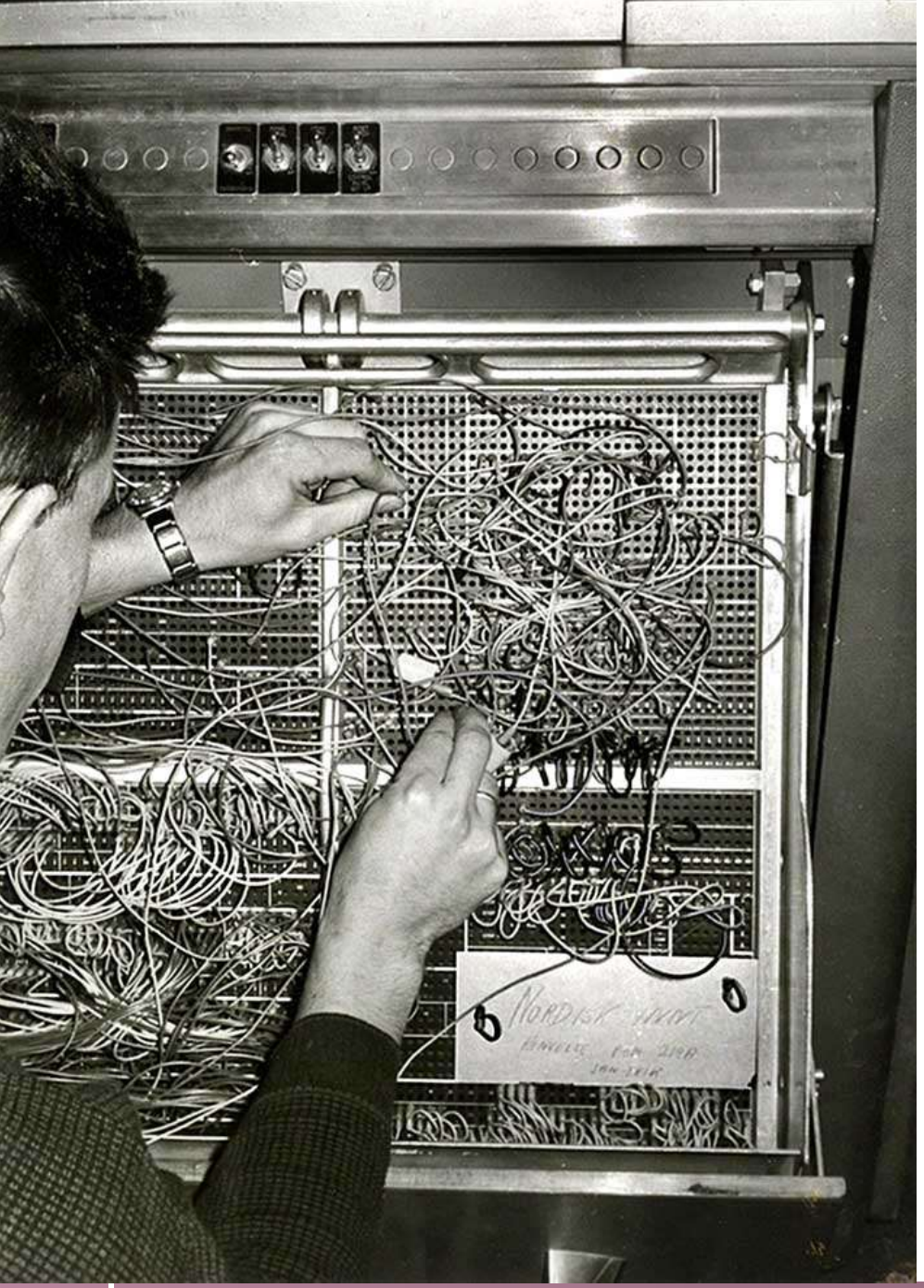
Was war das weltweit erste Computerprogramm und wer erstellte es? Dies hängt davon ab, was wir genau unter „Computer" verstehen wollen und was genau „programmieren" und „Programm" heissen soll. Sicherlich wurden die ersten elektromechanischen und elektronischen Computer der 1940er-Jahre bereits programmiert: Der von Konrad Zuse 1941 konstruierte elektromechanische Digitalrechner Z3 (ein Vorläufer des ab 1950 an der ETH Zürich verwendeten Z4-Computers) wurde über Lochstreifen programmiert, u.a. wurde seinerzeit ein Programm für die Berechnung einer komplexen Matrix geschrieben, das zur Berechnung von kritischen Flatterfrequenzen bei Flugzeugen verwendet wurde.

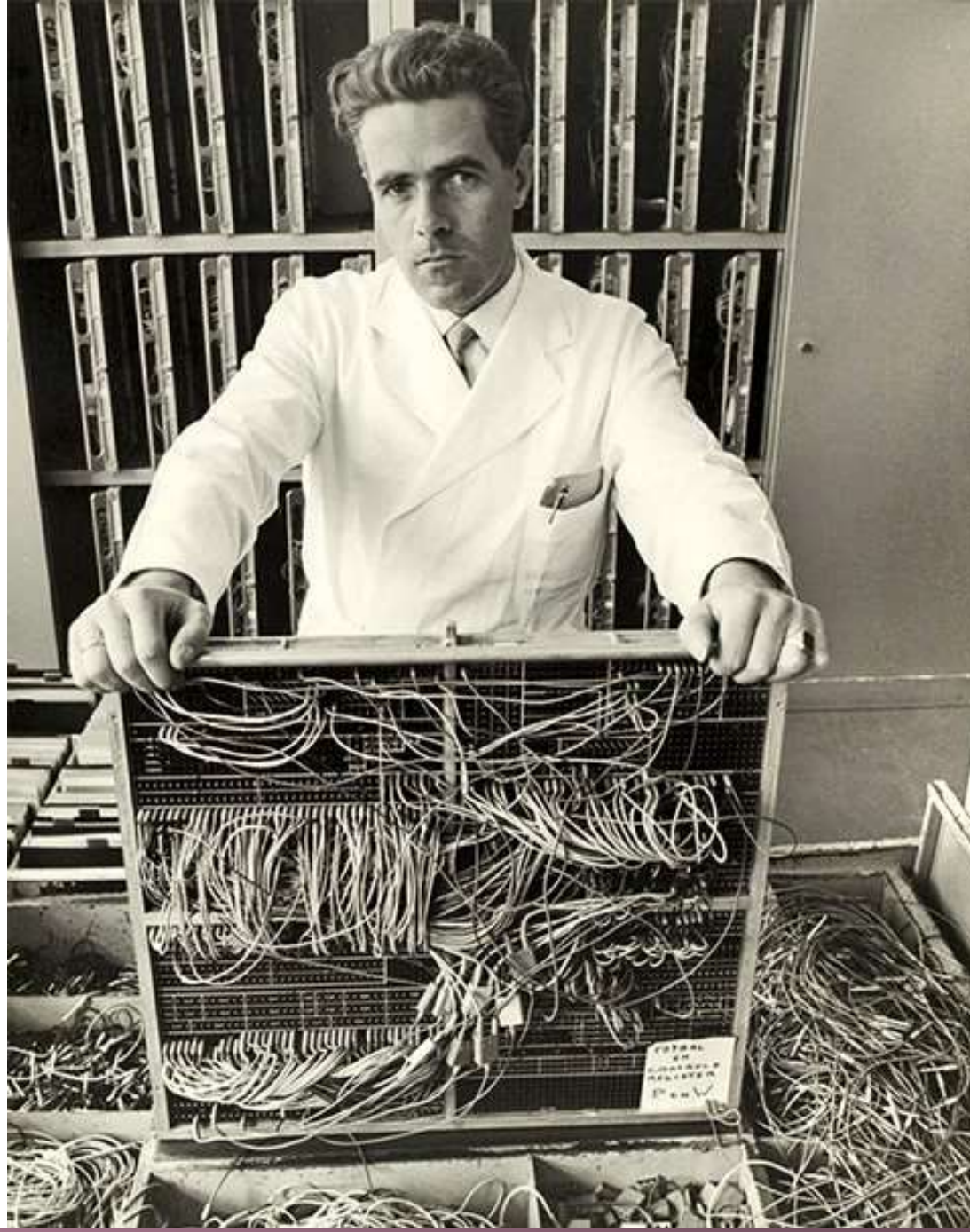
Die im zweiten Weltkrieg entstandenen tonnenschweren und raumfüllenden elektromechanischen Rechner Colossus (Tommy Flowers, Grossbritannien, Spezialrechner zur Dechiffrierung von geheimen Funknachrichten des deutschen Militärs in Bletchley Park) und Mark I, früher ASCC – Automatic Sequence Controlled Calculator – genannt, (Howard Aiken, USA) waren ebenfalls programmierbar; Mark I mit Lochstreifen, Colossus nur teilweise und eingeschränkt durch Neuverkabelung.

ENIAC („Electronic Numerical Integrator and Computer"), der erste rein elektronische Universalcomputer (USA, J. Presper Eckert und John W. Mauchly), der aber erst nach dem zweiten Weltkrieg funktionsfähig war, war eher mühsam, durch teilweise Neuverkabelung und später durch den Austausch von „Steckbrettern" (in Form von Widerstandsmatrizen) programmierbar.


Programmierung
mit Steckbrettern





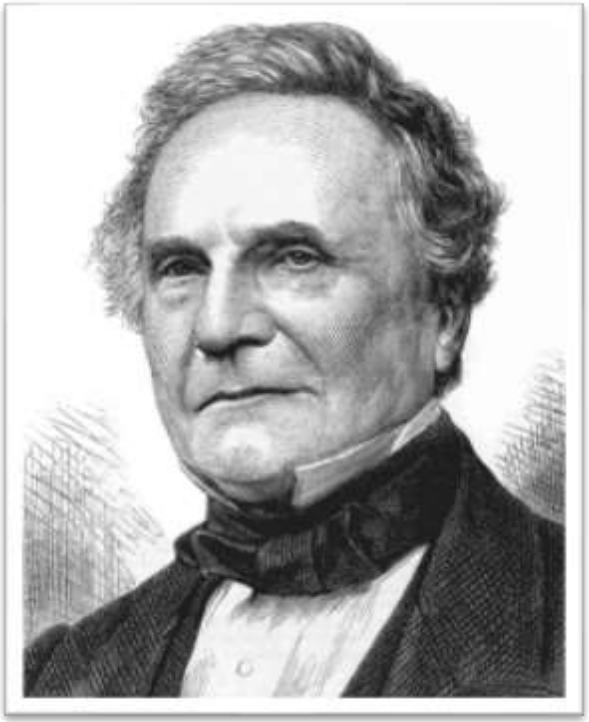


Besichtigung eines Programms


Programmierung
mit Lochstreifen

Das allererste Computerprogramm? (1)

Nicht als Computerprogramm wollen wir Rechenverfahren auffassen (die seit Jahrhunderten in Rechenbüchern zu finden sind), auch nicht Rechenanweisungen an „Rechenknechte", die die Grundrechenarten schematisch auf vorgegebene Zahlen anwendeten, etwa zur Interpolation bei der arbeitsteiligen Erzeugung von Tafelwerken für Winkelfunktionen, Logarithmen sowie astronomische Berechnungen. Andererseits muss man zugeben, dass die weiter oben erwähnten Schablonen zur schematischen Berechnung von Fourier-Koeffizienten von 1930 durchaus konkrete, in einzelne Schritte aufgelöste Rechenbefehle an (menschliche) „Rechner" bzw. „Rechnerinnen" waren, mit denen eine nichttriviale Aufgabe gelöst wurde. Tabelliermaschinen von Hollerith bzw. IBM, die zur Datenverarbeitung mit Lochkarten dienten, konnten ab ca. 1928 über Schalttafeln (bzw. „Steckbretter" oder „plug boards") konfiguriert werden; dies mag man evtl. auch als „Programmieren" bezeichnen. Aber man assoziiert mit „Computerprogramm" heute sicherlich eher etwas, das einen Algorithmus beschreibt und vollautomatisch von einer Maschine ausgeführt wird.

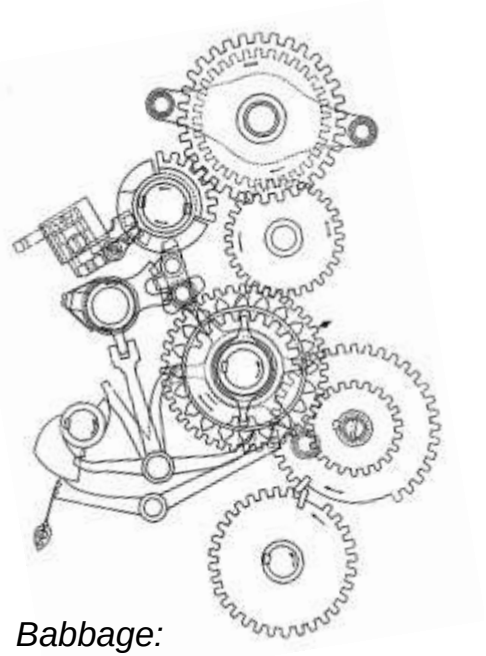


Interessant sind diesbezüglich Programme für die von Charles Babbage (1791 – 1871) ab Mitte der 1830er-Jahre konzipierte „Analytical Engine". Babbage war ein vielseitig interessierter, sehr geschäftiger Mathematiker und genialer Erfinder, zudem Philosoph, Geograph und Ökonom, aber auch ein eigensinniger Querulant und Misanthrop. („If your ideas are one hundred years in advance of your times, it is never easy to get your contemporaries to understand them; but Babbage clearly made matters far worse than they need have been by his intolerance, irascibility, vainness and unwillingness to compromise" urteilte Henry Turner über ihn.)

Das allererste Computerprogramm? (2)

Über den Beginn seiner Arbeit an Rechenautomaten berichtete Babbage in seiner Autobiographie: „Die Idee, dass es möglich sein müsste, Tabellen maschinell zu errechnen, kam mir, soweit ich mich erinnere, erstmals im Jahre 1820 oder 1821. Sie verdankte sich folgender Situation: Die Astronomical Society hatte für die Ausarbeitung bestimmter Tafeln eine Kommission ernannt, die aus Sir J. Herschel und mir bestand. Wir hatten uns auf die passenden Formeln geeinigt und sie zwei im Rechnen geübten Leuten übergeben, um die Berechnungen durchführen zu lassen. Eines Abends trafen wir uns, um die errechneten Tabellen zu vergleichen, und da wir viele Unstimmigkeiten fanden, äusserte ich gegenüber meinem Freund, dass ich wünschte, es gäbe ein dampfgetriebenes Rechnen, worauf er mir beipflichtete und meinte, so etwas liege durchaus im Bereich des Möglichen."

Babbage versuchte sich zunächst an einer einfacheren Maschine, der „Difference Engine". Diese war noch nicht allgemein programmierbar, sondern spezialisiert auf die Erstellung von Tafeln für Logarithmen, trigonometrische Funktionen etc. Das Prinzip der Maschine beruht auf dem Faktum, dass ein Polynom n-ten Grades konstante Differenzen n-ter Ordnung hat, die sukzessive zur Funktionsberechnung genutzt werden können. Bei allgemeinen Funktionen werden diese Differenzen höherer Ordnung nicht konstant, aber relativ klein, so dass (dank Weierstraß bzw. seines Approximationssatzes) interpolativ zurückgerechnet werden kann. Im Laufe der Zeit reifte bei Babbage der Gedanke, dass man das automatisierte Rechnen auch viel allgemeiner fassen kann und durch eine Maschine realisieren kann. Er nannte die neue Maschine „Analytical Engine". Der Begriff „analytical" sollte betonen, dass die neue Maschine wesentlich allgemeiner ist und nicht nur arithmetische und numerische, sondern algebraische und formelmässige Prinzipien verkörpert.

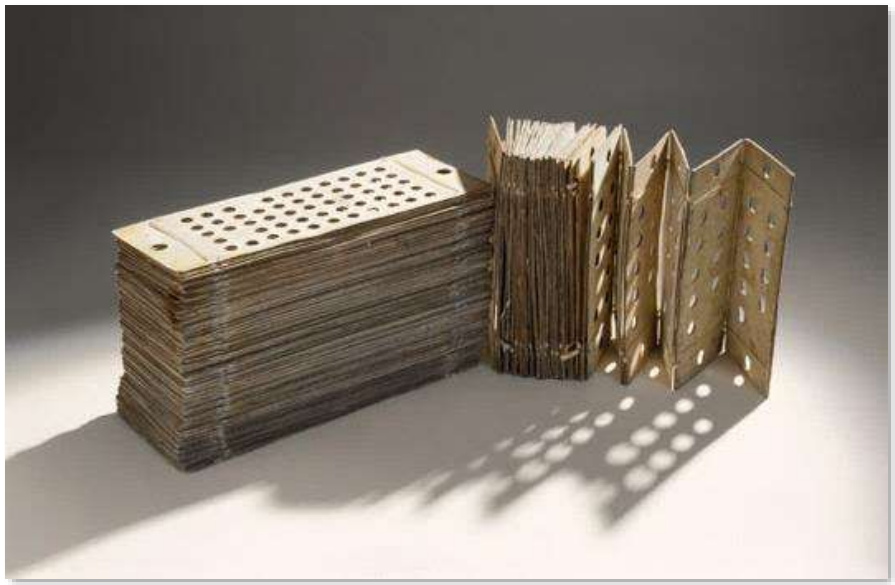


Babbage: Method of Carrying the Tens

Das allererste Computerprogramm? (3)

Obwohl rein mechanisch aufgebaut, weist die Analytical Engine bezüglich der Grundarchitektur alle Merkmale eines heutigen elektronischen Computers auf: Sie ist gegliedert in ein Rechenwerk („the mill"), das die vier Grundrechenarten im Dezimalsystem durchführen sollte, sowie einen Speicher für 1000 Variablen zu 50 Dezimalstellen, in dem auch von der Maschine erarbeitete Zwischenergebnisse gespeichert werden sollten. Die Eingabe der Befehle und Daten sollte über Lochkarten erfolgen, eine Methode, die in der damaligen Zeit der Steuerung mechanischer Webstühle („Jacquard-Webstühle") diente. Es sollte auch ein Drucker angeschlossen werden. Die gesamte Mechanik der Analytical Engine hätte von einer Dampfmaschine angetrieben werden können, sie wäre über 30 Meter lang und 10 Meter breit geworden.

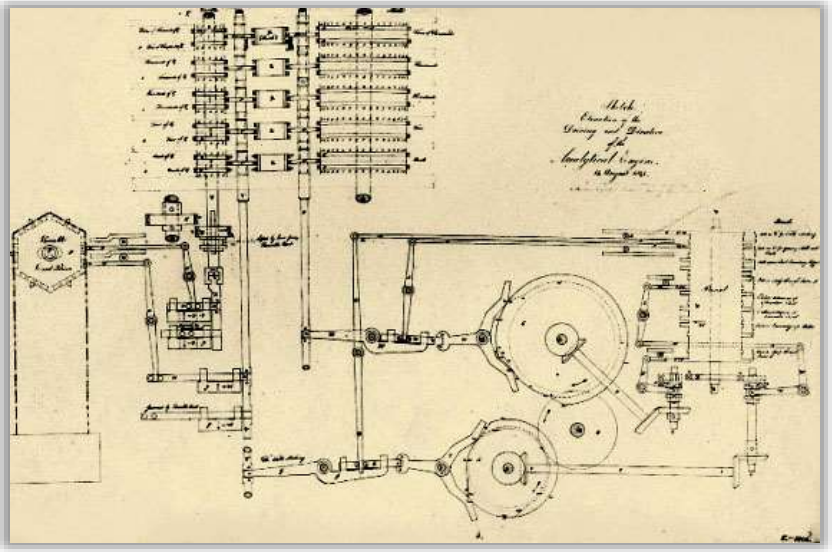
Während seiner Beschäftigung mit der Difference Engine kam Babbage zur Erkenntnis, dass weit weniger „Hardwareaufwand" notwendig wäre, wenn die Maschine sich selbst aufgrund berechneter Zwischenergebnisse rekonfigurieren könnte („a locomotive that lays down its own railway"). Ferner sollte die Möglichkeit bestehen, Rechenergebnisse als Operanden in nachfolgende Berechnungen einfliessen zu lassen. Dies erforderte eine flexible Ablaufsteuerung, die Babbage in Form von kettenförmig zusammengehängter Lochkarten realisieren wollte.



Das allererste Computerprogramm? (4)

Im Sommer 1836 notiert Babbage in seinem Skizzenbuch: „It is easier to punch pasteboard than to screw on a multitude of studs. When once the formula has been made and verified, it need never be made again until worn out. The change from one formula to another, when both have been previously made, is done in a very short time. There will be no backing of the drums, and the Jacquard pasteboards will circulate. Every formula ever put into the machine will be preserved. The extent of the formulae is almost unbounded." Tatsächlich stellte er sich später eine ganze Bibliothek aus Programmen vor, die in Form einer Kette aus aneinandergefädelten gelochten Pappkärtchen realisiert sind.

Um die kompliziertere Maschine zu bauen, stellte Babbage Konstruktionszeichner und Mechaniker ein und erwarb extra ein Haus auf einem grossen Grundstück mit Werkstätten, einer Schmiede sowie einer Giesserei. Es gab dennoch zu grosse technische Probleme, und nur wenige Komponenten wurden wirklich gebaut. Hinzu kamen finanzielle Probleme, da ihm die staatliche Förderung nicht in ausreichendem Masse gewährt wurde. Schon 1841 war er selbst pessimistisch hinsichtlich der Realisierung der Maschine, er schrieb an Alexander von Humboldt: „Es besteht keine Aussicht, dass die Maschine zu meinen Lebzeiten jemals gebaut wird, und ich weiss nicht einmal, was



Lochkartensteuerung (links) und Hilfstrommel zur Getriebesteuerung (rechts) der Analytical Engine

Das allererste Computerprogramm? (5)

mit den Konstruktionszeichnungen nach meinem Tod werden soll." Tatsächlich empfahl schliesslich ein Komitee der British Association for the Advancement of Science abschliessend, die Analytical Engine nicht zu bauen: „Wir sind der Ansicht, dass die Mühen, die Herr Babbage [...] auf seine Analytical Engine verwandte, ein Wunder an technischer Ingeniösität und technischem Vermögen sind. [...] Wir sind zu dem Schluss gekommen, dass es uns beim derzeitigen Konstruktionsstand der Maschine nicht möglich ist, irgendeine vernünftige Schätzung ihrer Kosten oder ihrer Festigkeit und Dauerhaftigkeit abzugeben." Die Analytical Engine war ihrer Zeit weit voraus; "his stillborn machine […] lay, forgotten, in bits and pieces in dusty libraries for a hundred years" (G.J.E. Rawlins).

Doch zurück ins Jahr 1840. In diesem Jahr traf der italienische Ingenieur Luigi Federico Menabrea, der später Ministerpräsident von Italien wurde, Babbage auf einem Kongress der dortigen Akademie der Wissenschaften, wo dieser über seine Maschine referierte. Als Folge seiner anschliessenden Beschäftigung damit veröffentlichte Menabrea 1842 in Genf eine auf französisch verfasste (und auch heute noch sehr lesenswerte) Beschreibung der Analytical Engine („Notions sur la machine analytique de Charles Babbage", Bibliothèque universelle de Genève, Nouvelle Série, 41^{e} tome, Oct. 1842, 352-376), und geht dabei vor allem auch auf ihre Programmierung ein.

Das allererste Computerprogramm? (6)

Er schreibt, dass Babbage einen „Wahnsinnsgedanken" hatte und stellt fest, dass es sich bei der Analytical Engine um weit mehr als um eine Rechenmaschine handelt: „Il ne s'est proposé rien moins que de construire une machine capable d'exécuter, non-seulement les calculs arithmétiques ; mais encore les calculs analytiques." Die Maschine „devait avoir la généralité de l'écriture algébrique même, et que, pour cette raison, il nomme *machine analytique*".

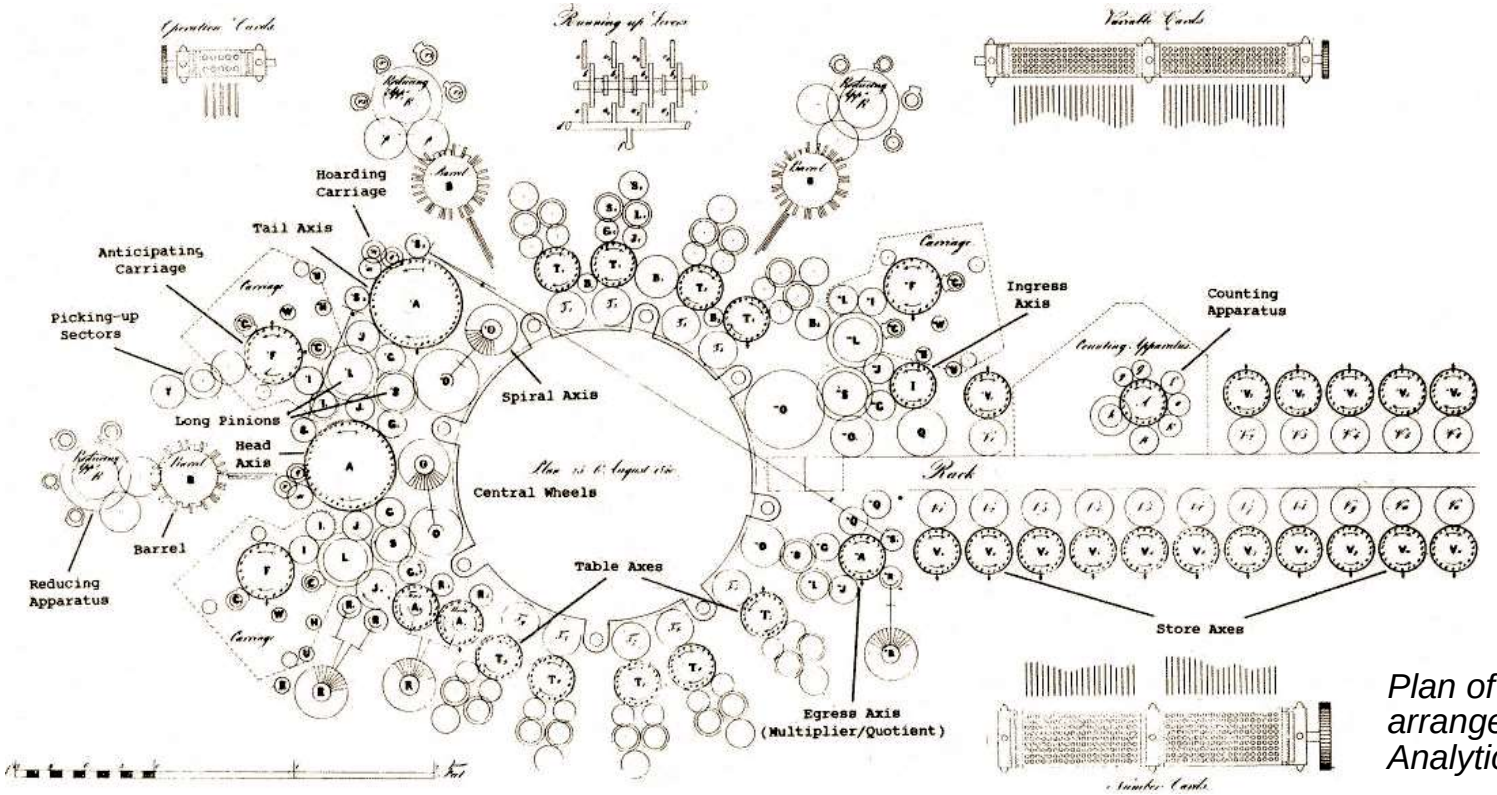
Bei algebraischen Methoden müssen mehrere Einzelschritte in geeigneter Weise nacheinander ausgeführt werden und Zwischenresultate an nachfolgende Rechenschritte weiterverteilt werden – Menschen seien hier jedoch langsam und fehleranfällig, daher solle die Maschine dies selbst leiten: „Mais si la main de l'homme était obligée d'intervenir pour diriger chacune de ces opérations partielles, il n'y aurait rien de gagné sous le rapport de l'exactitude et de l'économie de temps ; il faudra donc encore que la machine ait la propriété d'exécuter elle-même toutes les opérations successives nécessaires."

Der Begriff „Programm" oder „Programmierung" taucht bei Menabrea zwar dem Namen nach noch nicht auf, doch charakterisiert er dieses „Prinzip" als „assez général pour que, si on l'applique à la machine, celle-ci soit capable de traduire mécaniquement les opérations qui lui seraient indiquées par l'écriture algébrique."

Wie erfolgt die programmierte Steuerung der einzelnen Operationen? „Comment la machine peut d'elle-même et sans avoir recours à la main de l'homme, prendre les dispositions successives convenables pour opérer? La solution de ce problème a été empruntée à l'appareil de Jacquard, en usage pour la confection des étoffes brochées."

Das allererste Computerprogramm? (7)

Die Lochkartensteuerung beschreibt Menabrea so: „Les cartons étant mis en mouvement, disposent successivement les diverses pièces de la machine selon la nature des opérations à faire, et la machine les exécute en même temps au moyen des mécanismes dont elle est composée.“ Die Variablen befinden sich in einem Speicher und werden fallweise zum Rechenwerk transportiert bzw. aus diesem zurück in den Speicher: „...passent alternativement du moulin au magasin et du magasin au moulin, pour y subir les transformations requises par la nature du calcul à exécuter.“



Plan of the general arrangement of the Analytical Engine

Das allererste Computerprogramm? (8)

Sodann gibt Menabrea ein Beispiel – die Lösung eines linearen Gleichungssystems mit zwei Unbekannten x, y: $mx+ny=d$; $m'x+n'y=d'$. Das zugehörige kommentierte Programm notiert er in Form einer Tabelle, wo in elf sukzessiven Schritten mit den Elementaroperationen Multiplikation, Subtraktion und Division das Ergebnis (also die Werte für x und y) berechnet werden. Die 6 Koeffizienten der Gleichung werden durch Variablen V_0 bis V_5 repräsentiert, Zwischenresultate auf den Hilfsvariablen V_6 bis V_{14} abgespeichert und die Werte der berechneten Unbekannten resultieren auf den Variablen V_{15} und V_{16}. Jede Programmzeile nennt die arithmetische Operation, die Operanden sowie die zu verwendende Ergebnisvariable. Dies entspricht recht genau den erst 100 Jahre später für tatsächliche Computer verwendeten Assemblerprogrammen.

Colonnes sur lesquelles sont écrites les données primitives du problème.	Nombre des opérations.	CARTONS des opérations. Nombre des cartons des opérations	CARTONS des opérations. Signe indiquant la nature des opér.	CARTONS DES VARIABLES. COLONNES soumises aux opérations	Colonnes qui reçoivent le résultat des opérations	Indication des nouvelles colonnes sur lesquelles sont écrites les variables.	MARCHE des OPÉRATIONS.
$V_0 = m$	I	I	×	$V_0 \times V_4 =$	$V_6 \ldots$	V_0 sur V_0; V_4 id. V_4	$V_6 = mn'$
$V_1 = n$	2	»	×	$V_3 \times V_1 =$	$V_7 \ldots$	V_1 id. V_1; V_3 id. V_3	$V_7 = m'n$
$V_2 = d$	3	»	×	$V_2 \times V_4 =$	$V_8 \ldots$	V_2 id. V_2; V_4 id.	$V_8 = dn'$
$V_3 = m'$	4	»	×	$V_5 \times V_1 =$	$V_9 \ldots$	V_1 id. ....; V_5 id. V_5	$V_9 = d'n$
$V_4 = n'$	5	»	×	$V_0 \times V_5 =$	$V_{10} \ldots$	V_0 id. ....; V_5 id.	$V_{10} = d'm$
$V_5 = d'$	6	»	×	$V_2 \times V_3 =$	$V_{11} \ldots$	V_2 id. ....; V_3 id.	$V_{11} = dm'$
	7	2	–	$V_6 - V_7 =$	$V_{12} \ldots$	V_6 id. ....; V_7 id.	$V_{12} = mn' - m'n$
	8	»	–	$V_8 - V_9 =$	$V_{13} \ldots$	V_8 id. ....; V_9 id.	$V_{13} = dn' - d'n$
	9	»	–	$V_{10} - V_{11} =$	$V_{14} \ldots$	V_{10} id. ....; V_{11} id.	$V_{14} = d'm - dm'$
	10	3	:	$\frac{V_{13}}{V_{12}} =$	$V_{15} \ldots$	V_{13} id. ....; V_{12} id. V_{12}	$V_{15} = \frac{dn' - d'n}{mn' - m'n} = x$
	11	»	:	$\frac{V_{14}}{V_{12}} =$	$V_{16} \ldots$	V_{12} id. ....; V_{14} id.	$V_{16} = \frac{d'm - dm'}{mn' - m'n} = y$

Das allererste Computerprogramm? (9)

Dass das Programm auf nützliche Art „optimiert" ist, fällt erst bei genauerem Hinsehen auf: Erst kommen alle Multiplikationen, dann alle Subtraktionen und schliesslich die Divisionen; auf diese Weise werden insgesamt nur drei Operationslochkarten benötigt.

Mit einigem Grund kann man dieses Programm von Menabrea als das erste veröffentlichte Computerprogramm ansehen – auch wenn es sich bei der Analytical Engine nur um eine „virtuelle Maschine" handelt, die zur damaligen Zeit nicht wirklich existierte – und mit dem damaligen technischen Möglichkeiten (vor allem der noch nicht weit genug entwickelten Feinmechanik für diesen Zweck) vielleicht auch gar nicht realisiert werden konnte.

Menabrea kommt anschliessend im Text noch auf kompliziertere Berechnungen zu sprechen und beschreibt u.a., wie Schleifen mit einer rückwärts zählenden Variablen realisiert werden können: „Les opérations indiquées se répéteront jusqu'à ce que le compteur ne marque plus que zéro." Auch bedingte Befehle erwähnt er, allerdings wird dies in einer Fussnote der englischen Übersetzung des Artikels (von Ada Lovelace) klarer ausgedrückt: „The engine is capable, under certain circumstances, of feeling about to discover which of two or more possible contingencies has occurred, and of then shaping its future course accordingly." Die Berechnung der Kreiszahl π sowie der Bernoulli-Zahlen erwähnt er nur beiläufig – solche Zahlen könnten auch vorberechnet werden und dann bei Bedarf direkt in die „Mühle" eingespeist werden.

Programme seien nach Menabrea „une autre forme d'écriture analytique", und die Programmierung („l'emploi des cartons") sieht er als das entscheidende und mächtige Prinzip der Analytical Engine an: „L'emploi des cartons offre une généralité égale à celle des

Das allererste Computerprogramm? (10)

formules algébriques“. Mit anderen Worten: Die Programmierbarkeit verleiht der Maschine Universalcharakter.

Wenn man diese Universalmaschine einmal gebaut hätte, ginge es nur noch um das Programmieren: „Une fois que la machine sera construite, la difficulté se reportera donc sur la confection des cartons“. Optimistisch meinte Menabrea zu den Programmen seinerzeit: „Comme ceux-ci ne sont que la traduction de formules algébriques, par le moyen de simples notations il sera facile d’en confier l’exécution à un ouvrier.“ Was die Übersetzung von Formeln in Maschinensprache angeht, hatte er recht – noch nicht einmal Arbeiter brauchte man dazu ab der Mitte des 20. Jahrhunderts, sondern die Maschine selbst erledigte dies mit Compilern. Dass dies nicht alles ist und knapp 130 Jahre später Begriffe wie „Software-Krise“ und „Software-Engineering“ entstehen würden, war seinerzeit natürlich nicht absehbar.

Am Ende seines Artikels gerät Menabrea ins Schwärmen: Ein Instrument zu schaffen, das die menschliche Schwäche in Form von Langsamkeit und Fehleranfälligkeit beim Rechnen ausgleichen würde, sei höchst nützlich: „Combien d’observations précieuses restent inutiles aux progrès des sciences, parce qu’il n’y a pas de forces suffisantes pour en calculer les résultats !“

Hier betritt nun Ada Augusta King, Countess of Lovelace (geb. Byron) (1815 – 1852), eine Tochter des bekannten britischen Dichters Lord Byron, die Bühne. Ihre mathematisch interessierte Mutter, die Geometrie und Astronomie studiert hatte, ermöglichte Ada eine naturwissenschaftliche Ausbildung, in deren Verlauf sie die die Universalgelehrte und Wissenschaftsautorin Mary Somerville und auch Charles Babbage kennenlernte. Eine

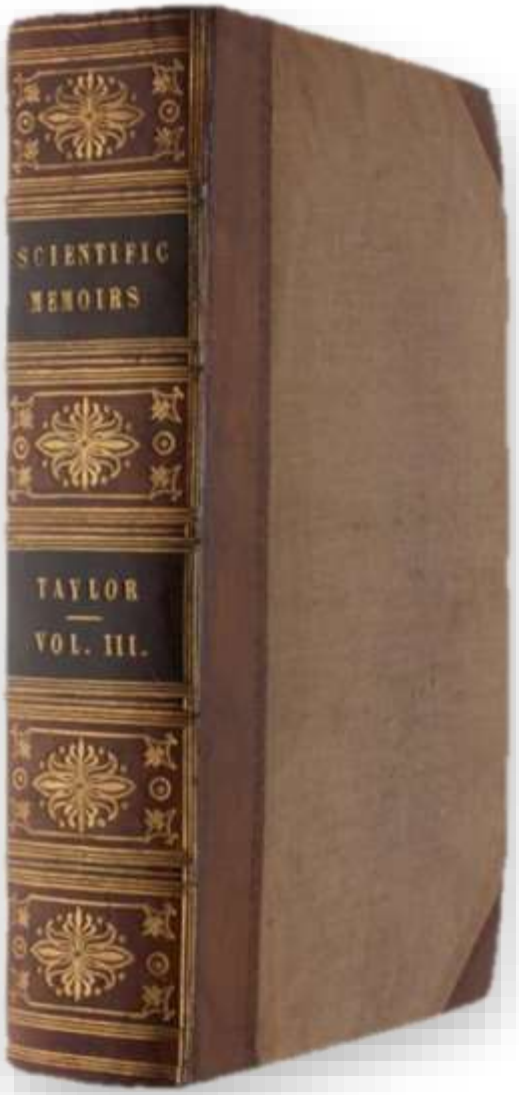
Das allererste Computerprogramm? (11)

engere Zusammenarbeit mit Babbage entstand allerdings erst im Rahmen ihrer Übersetzung von Menabreas Veröffentlichung („Sketch of the Analytical Engine invented by Charles Babbage", Taylor's Scientific Memoirs, Vol. III, pp. 666-731, 1843). Zu dieser Übersetzung hatte ihr der bekannte Experimentalphysiker Charles Wheatstone geraten, ein Freund der Familie, der für die britische Zeitschrift „Taylor's Scientific Memoirs" arbeitete, welche auf die Publikation von Übersetzungen fremdsprachiger Artikel spezialisiert war. Der Artikel wurde dabei von Ada Lovelace mit ausführlichen Annotationen versehen, die fast drei Mal so lang wie der ursprüngliche Text von Menabrea waren und damit quasi eine eigenständige Abhandlung darstellen.

Ada Lovelace – Aquarell (Ausschnitt)
von Alfred Edward Chalon, 1840

"*Programming is a particularly British skill. In fact, we invented it.* Thus spoke the narrator of an influential 1978 BBC documentary on the challenges of the information age." [James Sumner]

Das allererste Computerprogramm? (12)



In ihren „notes by the translator" präsentiert Ada Lovelace zusätzliche Details und verdeutlicht dabei auch einige der Aussagen von Menabrea. So betont sie etwa die Leistungsfähigkeit der Analytical Engine mit folgenden Worten: „The engine may be described as being the material expression of any indefinite function of any degree of generality and complexity." Oder: „It can arbitrarily substitute any formula for any other". Auch die Tatsache, dass es nicht nur um numerische Rechnungen geht, streicht sie heraus: „Symbolical results are not less the necessary and logical consequences of operations performed upon symbolical data, than are numerical results when the data are numerical."

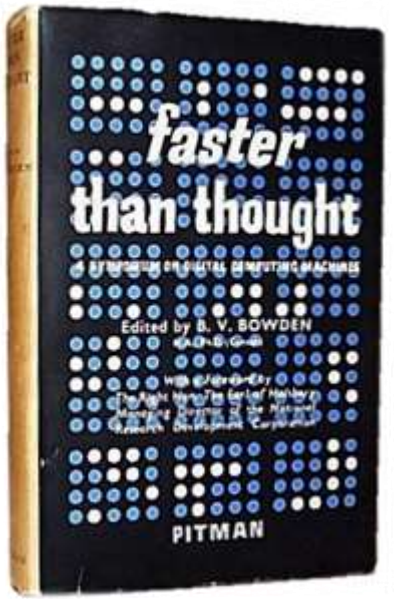
Vor allem drei grundlegende und weitreichende Bemerkungen in ihren Annotationen zu Menabreas Bericht begründen den Ruhm von Ada Lovelace. Die *erste Bemerkung* betrifft den entscheidenden Unterschied zwischen einer blossen Rechenmaschine und einem programmierbaren Computer: „The bounds of arithmetic were however outstepped the moment the idea of applying the cards had occurred; and the Analytical Engine does not occupy common ground with mere calculating machines. It holds a position wholly its own; and the considerations it suggests are most interesting in their nature."

Das allererste Computerprogramm? (13)

Die *zweite Bemerkung* betrifft die Möglichkeit, mit einem Computer wie der Analytical Engine mehr als nur arithmetische Aufgaben bearbeiten zu können: „It might act upon other things besides number, were objects found whose mutual fundamental relations could be expressed by those of the abstract science of operations, and which should be also susceptible of adaptations to the action of the operating notation and mechanism of the engine."

Die *dritte Bemerkung* bleibt ein wenig vage, wird aber aufgrund ihres poetischen Charakters besonders gerne zitiert (und dabei vielleicht sogar überinterpretiert): „The Analytical Engine weaves algebraical patterns just as the Jacquard-loom weaves flowers and leaves."

Ferner enthalten die Annotationen von Ada Lovelace ein Programm zur Berechnung von Bernoulli-Zahlen mit der Analytical Engine – genauer gesagt, wird die Bernoulli-Zahl B_7 berechnet und beschrieben, wie sich dieses Schema generalisieren lässt. (Es enthält übrigens einen kleinen „Bug": In Zeile 4 sind die Variablen V_4 und V_5 vertauscht.) Dieses Programm wird im Wesentlichen in der gleichen Notation verfasst wie das Programm zur Lösung linearer Gleichungssysteme von Menabrea. Vor allem aufgrund dieses Programms wird Ada gern mit dem Ehrentitel „erste Programmiererin" dekoriert, auch wenn Menabrea sein oben angegebenes Programm bereits früher veröffentlichte.



In diesem Sammelband von 1953 wird der Aufsatz vom Menabrea und Ada Lovelace erneut veröffentlicht – erst dadurch gewann er im angehenden Computerzeitalter an Bekanntheit und Bedeutung

„Faster than thought" ist ein geflügeltes Wort aus Shakespeares „Wintermärchen": Die Liebe solle schneller wachsen als „thought or time"

ADA AUGUSTA
The Countess of Lovelace

Frontispiece

FASTER THAN THOUGHT

A SYMPOSIUM ON
DIGITAL COMPUTING MACHINES

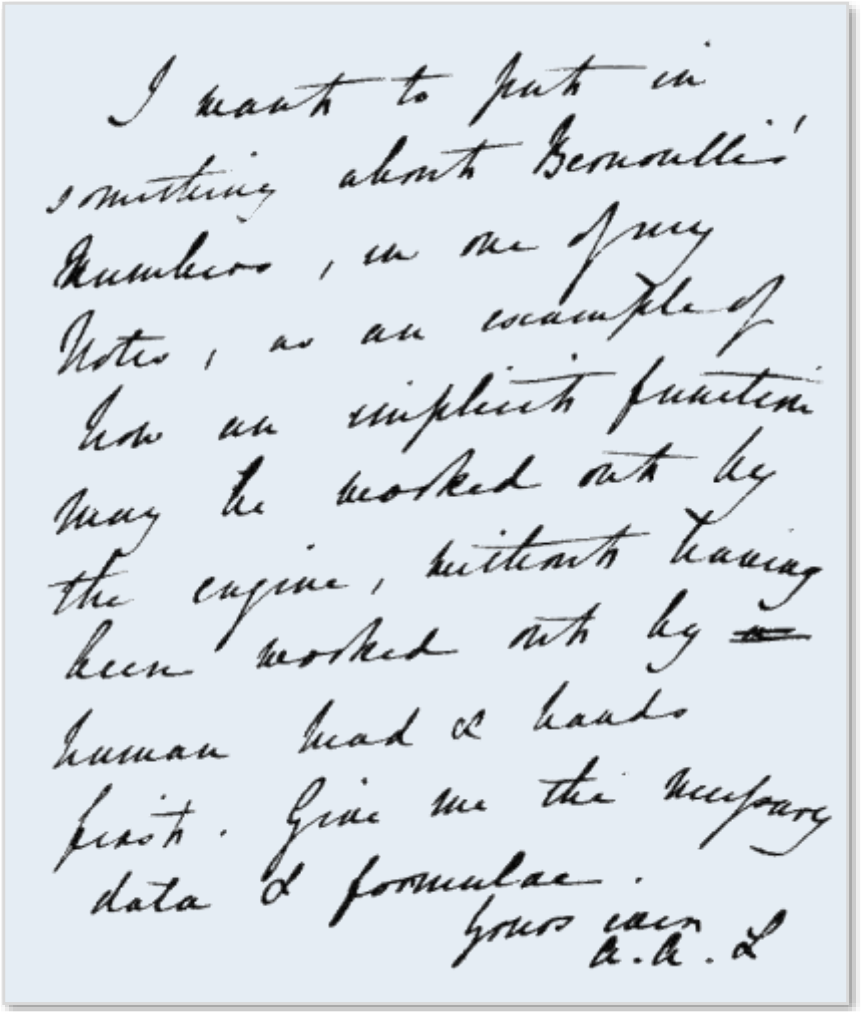
EDITED BY
B. V. BOWDEN
M.A., Ph.D. (Cantab.)

With a Foreword by
THE RIGHT HON. THE EARL OF HALSBURY
Managing Director of the National Research Development Corporation

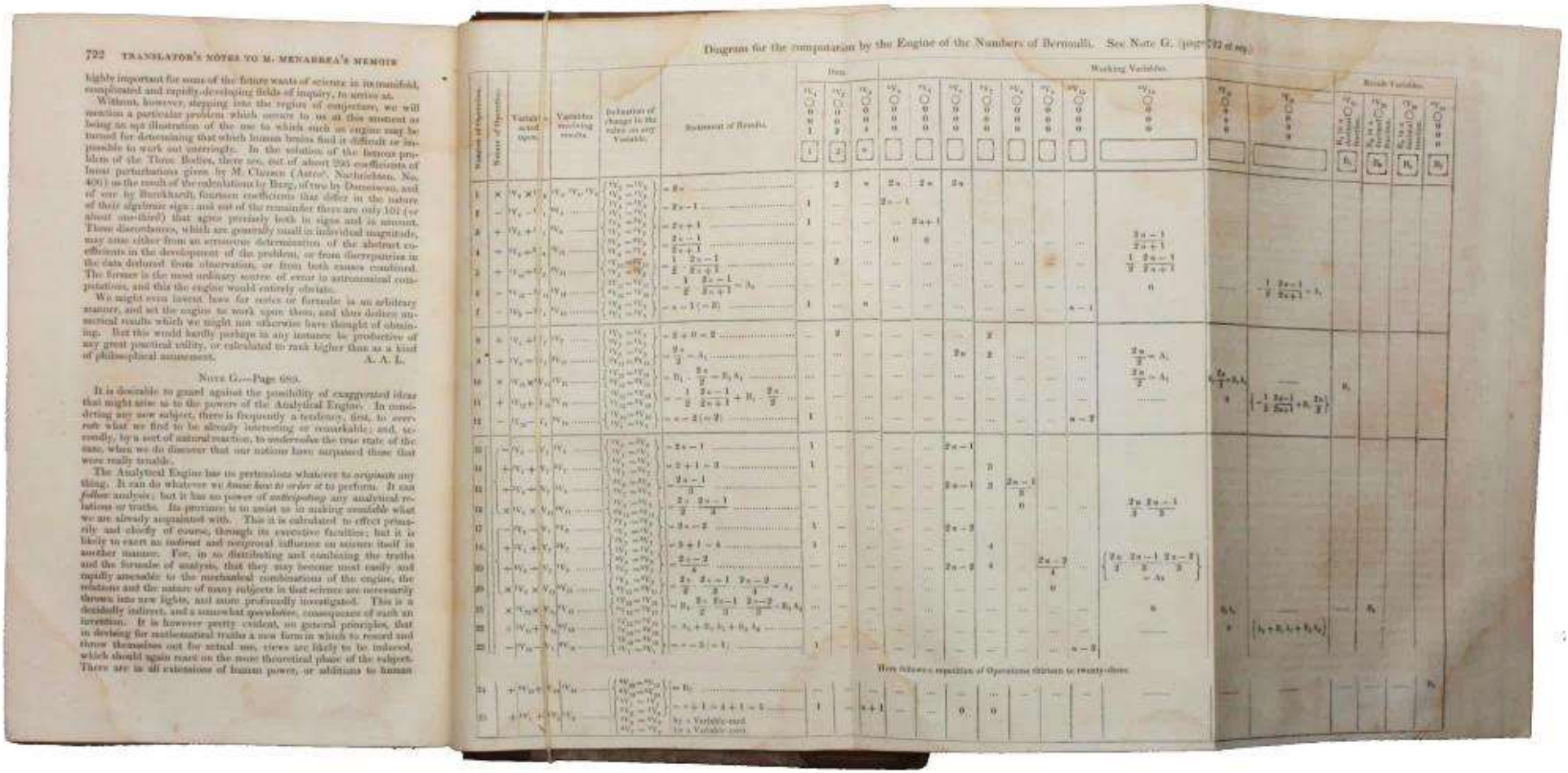
"Creating a space for programming in the grand national narrative had been one of Vivian Bowden's priorities in Faster Than Thought – hence his relentless championing of the conceptual achievements of Charles Babbage, and in particular of the discussion of applications by Ada Lovelace, whose portrait he included as a frontis illustration (this coverage was, in later years, the foundation of the distinctly ahistorical depiction of Lovelace as the 'first computer programmer')." [James Sumner]

Das allererste Computerprogramm? (14)

Unter Fachleuten ist im Übrigen nicht ganz unumstritten, ob das Bernoulli-Programm tatsächlich bzw. hauptsächlich von Ada (oder etwa von Babbage) entwickelt wurde. In einem Brief vom 10. Juli 1843 an Babbage bittet Ada Lovelace jedenfalls um die Formeln zur Berechnung: *„I want to put in something about Bernoulli's Numbers, in one of my Notes, as an example of how an implicit function may be worked out by the engine, without having been worked out by human head & hands first. Give me the necessary data & formulae.*" Und Babbages Autobiographie, welche erst einige Jahre nach Adas frühem Tod entstand, vermerkt: „We discussed together



Das allererste Computerprogramm? (15)



the various illustrations that might be introduced: I suggested several, but the selection was entirely her own. So also was the algebraic working out of the different problems, except, indeed, that relating to the numbers of Bernoulli, which I had offered to do to save Lady Lovelace the trouble. This she sent back to me for an amendment, having detected a grave mistake which I had made in the process."

Das allererste Computerprogramm? (16)

ber of Operati	of Operation	Variables acted upon	Variables receiving results	Indication of change in the value on any Variable	Statement of Results
1	×	${}^1V_2 \times {}^1V_3$	${}^1V_4, {}^1V_5, {}^1V_6$	${}^1V_2 = {}^1V_2$; ${}^1V_3 = {}^1V_3$	$= 2n$ .....
2	−	${}^1V_4 - {}^1V_1$	${}^2V_4$	${}^1V_4 = {}^2V_4$; ${}^1V_1 = {}^1V_1$	$= 2n - 1$ .....
3	+	${}^1V_5 + {}^1V_1$	${}^2V_5$	${}^1V_5 = {}^2V_5$; ${}^1V_1 = {}^1V_1$	$= 2n + 1$ .....
4	÷	~~${}^2V_5$~~ ÷ ~~${}^2V_4$~~ (V4 V5)	${}^1V_{11}$	${}^2V_5 = {}^0V_5$; ${}^2V_4 = {}^0V_4$	$= \frac{2n-1}{2n+1}$ ..
5	÷	${}^1V_{11} \div {}^1V_2$	${}^2V_{11}$	${}^1V_{11} = {}^2V_{11}$; ${}^1V_2 = {}^1V_2$	$= \frac{1}{2} \cdot \frac{2n-1}{2n+1}$ .....
6	−	${}^0V_{13} - {}^2V_{11}$	${}^1V_{13}$	${}^2V_{11} = {}^0V_{11}$; ${}^0V_{13} = {}^1V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$ .....
7	−	${}^1V_3 - {}^1V_1$	${}^1V_{10}$	${}^1V_3 = {}^1V_3$; ${}^1V_1 = {}^1V_1$	$= n - 1 (= 3)$ .....
8	+	${}^1V_2 + {}^0V_7$	${}^1V_7$	${}^1V_2 = {}^1V_2$; ${}^0V_7 = {}^1V_7$	$= 2 + 0 = 2$ .....
9	÷	${}^1V_6 \div {}^1V_7$	${}^3V_{11}$	${}^1V_6 = {}^1V_6$; ${}^0V_{11} = {}^3V_{11}$	$= \frac{2n}{2} = A_1$ .....
10	×	${}^1V_{21} \times {}^3V_{11}$	${}^1V_{12}$	${}^1V_{21} = {}^1V_{21}$; ${}^3V_{11} = {}^3V_{11}$	$= B_1 \cdot \frac{2n}{2} = B_1 A_1$ .....
11	+	${}^1V_{12} + {}^1V_{13}$	${}^2V_{13}$	${}^1V_{12} = {}^0V_{12}$; ${}^1V_{13} = {}^2V_{13}$	$= -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$ .....
12	−	${}^1V_{10} - {}^1V_1$	${}^2V_{10}$	${}^1V_{10} = {}^2V_{10}$; ${}^1V_1 = {}^1V_1$	$= n - 2 (= 2)$ .....

Programm zur Berechnung der Bernoulli-Zahlen (erster Teil)

Ein kleiner „Bug" in Zeile 4: die Variablen V_4, V_5 sind vertauscht

Def. Bernoulli-Zahlen als Taylorreihe-Koeffizienten:

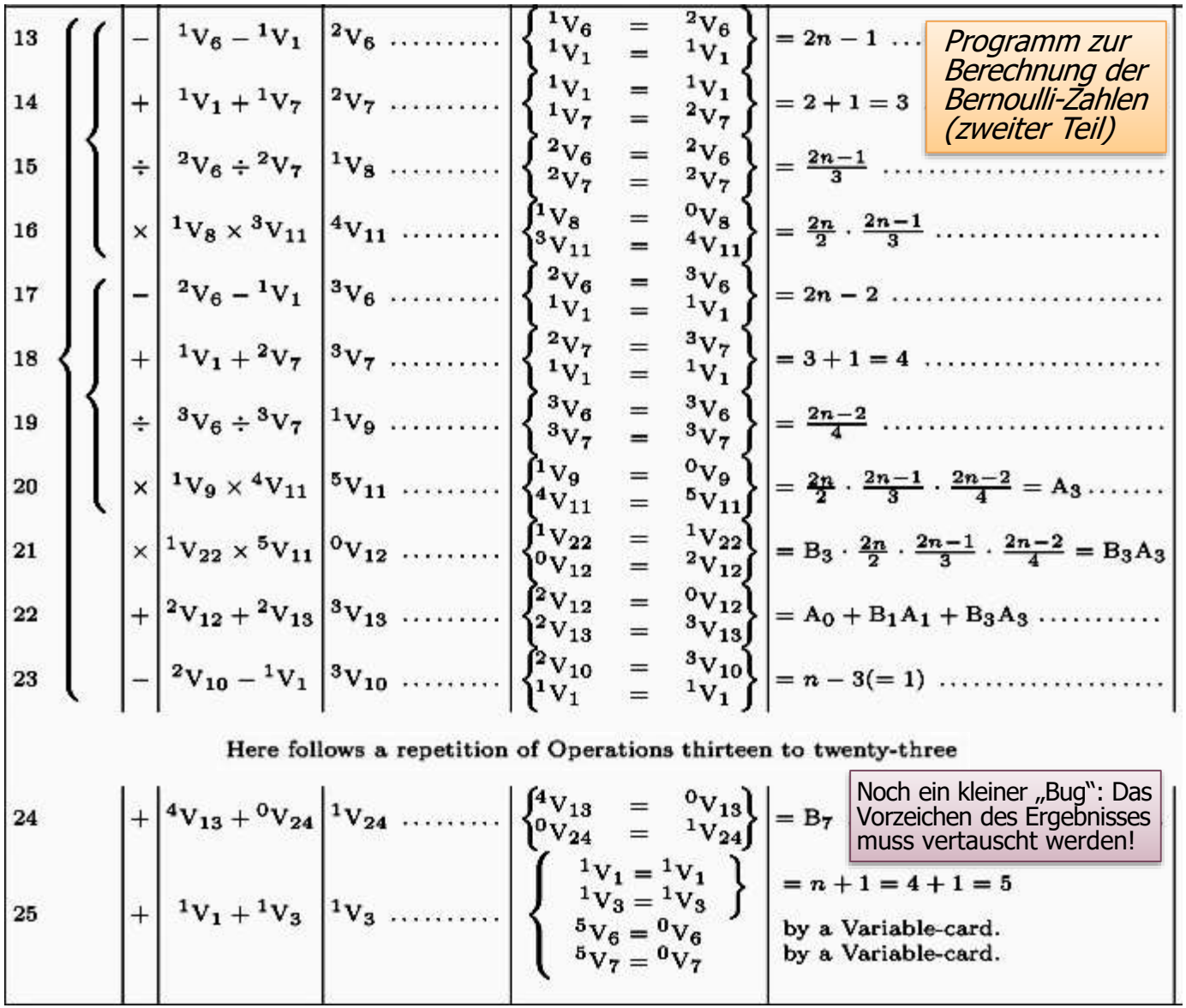
$$\frac{x}{e^x - 1} = \sum_{n \geq 0} B_n \frac{x^n}{n!}$$

Daraus in „Note G" von Ada Lovelace durch Umformung die „formula (8)" für die rekursive Berechnung einer Bernoulli-Zahl B_{2n-1} aus B_1, B_3, B_5,...:

$$0 = -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \frac{2n}{2} + B_3 \frac{2n(2n-1)(2n-2)}{2 \cdot 3 \cdot 4} + B_5 \frac{2n(2n-1)(2n-2)\cdots}{2 \cdot 3 \cdot 4 \cdot \cdots} \cdots \frac{\cdots(2n-3)(2n-4)}{\cdots \cdot 5 \cdot 6} + \cdots + B_{2n-1}$$

(Für ein konkretes n nehmen alle grösseren Summanden den Wert 0 an)

Das allererste Computerprogramm? (17)



13	–	$^1V_6 - {}^1V_1$	$^2V_6$	$^1V_6 = {}^2V_6$, $^1V_1 = {}^1V_1$	$= 2n - 1$ ...	
14	+	$^1V_1 + {}^1V_7$	$^2V_7$	$^1V_1 = {}^1V_1$, $^1V_7 = {}^2V_7$	$= 2 + 1 = 3$	
15	÷	$^2V_6 \div {}^2V_7$	$^1V_8$	$^2V_6 = {}^2V_6$, $^2V_7 = {}^2V_7$	$= \frac{2n-1}{3}$	
16	×	$^1V_8 \times {}^3V_{11}$	$^4V_{11}$	$^1V_8 = {}^0V_8$, $^3V_{11} = {}^4V_{11}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3}$	
17	–	$^2V_6 - {}^1V_1$	$^3V_6$	$^2V_6 = {}^3V_6$, $^1V_1 = {}^1V_1$	$= 2n - 2$	
18	+	$^1V_1 + {}^2V_7$	$^3V_7$	$^2V_7 = {}^3V_7$, $^1V_1 = {}^1V_1$	$= 3 + 1 = 4$	
19	÷	$^3V_6 \div {}^3V_7$	$^1V_9$	$^3V_6 = {}^3V_6$, $^3V_7 = {}^3V_7$	$= \frac{2n-2}{4}$	
20	×	$^1V_9 \times {}^4V_{11}$	$^5V_{11}$	$^1V_9 = {}^0V_9$, $^4V_{11} = {}^5V_{11}$	$= \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = A_3$	
21	×	$^1V_{22} \times {}^5V_{11}$	$^0V_{12}$	$^1V_{22} = {}^1V_{22}$, $^0V_{12} = {}^2V_{12}$	$= B_3 \cdot \frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = B_3A_3$	
22	+	$^2V_{12} + {}^2V_{13}$	$^3V_{13}$	$^2V_{12} = {}^0V_{12}$, $^2V_{13} = {}^3V_{13}$	$= A_0 + B_1A_1 + B_3A_3$	
23	–	$^2V_{10} - {}^1V_1$	$^3V_{10}$	$^2V_{10} = {}^3V_{10}$, $^1V_1 = {}^1V_1$	$= n - 3(= 1)$	

Here follows a repetition of Operations thirteen to twenty-three

24	+	$^4V_{13} + {}^0V_{24}$	$^1V_{24}$	$^4V_{13} = {}^0V_{13}$, $^0V_{24} = {}^1V_{24}$	$= B_7$	Noch ein kleiner „Bug“: Das Vorzeichen des Ergebnisses muss vertauscht werden!
25	+	$^1V_1 + {}^1V_3$	$^1V_3$	$^1V_1 = {}^1V_1$, $^1V_3 = {}^1V_3$, $^5V_6 = {}^0V_6$, $^5V_7 = {}^0V_7$	$= n + 1 = 4 + 1 = 5$	by a Variable-card. by a Variable-card.

Programm zur Berechnung der Bernoulli-Zahlen (zweiter Teil)

„…a Table, containing the details of the computation for B_7, (B_1, B_3, B_5 being supposed given).“

„…the numerical value of every Number of Bernoulli in succession, from the very beginning, ad infinitum, by the following series of computations–

- 1st Series.– Let n = 1, and calculate (8) for this value of n. The result is B_1.
- 2nd Series.– Let n = 2. Calculate (8) for this value of n, substituting the value of B_1 just obtained. The result is B_3.
- 3rd Series.– Let n = 3. Calculate (8) for this value of n, substituting the values of B_1, B_3 before obtained. The result is B_5. And so on, to any extent.“

Das allererste Computerprogramm? (18)

Die folgende Interpretation des Programms für die Bernoulli-Zahlen ist hilfreich. (Die innere „Schleife" geht über Zeilen 13 bis 23, die äussere befindet sich aber nicht im Programm selbst. Die Variable V_{13} dient als Akkumulator für die Summanden in „formula (8)". Man beachte, dass in der *zweiten* Iteration der inneren „Schleife" V_{23} anstelle von V_{22} verwendet werden muss!)

Computing each Bernoulli number one at a time constituted the outer loop of the program, to use modern computer programming parlance. To compute the fractional value next to each Bernoulli number, Ada used a second loop. She started by dividing the first factor of the numerator by the first factor of the denominator and storing that value. She then divided the second factor of the numerator by the second factor of the denominator and multiplied that value by the previously stored value. These steps were repeated until the value of the fraction was completely calculated, at which point it was multiplied by the appropriate Bernoulli number.

When the program begins, there are six variables in use: V_1, V_2, V_3, V_{21}, V_{22} and V_{23}. (The superscripts in the chart indicate the number of times the variable has been used.) The values of these variables are 1, 2, n (which in this case is 4, because Ada is calculating B_7, the fourth Bernoulli number), B_1, B_3 and B_5. V_{10} is used to store the number of iterations left to perform. At the first iteration, V_{10} is equal to $n-1$, at the second iteration, V_{10} is equal to $n-2$, and so on. When V_{10} equals 1, the loop stops, and the program is finished computing the Bernoulli number.

The first six operations calculate $(1/2) \times (2n-1) / (2n+1)$ and store the value in V_{13}. Operation 7 subtracts 1 from n and assigns it to V_{10} because the first iteration is complete. Operations 8, 9 and 10 calculate $2n/2$ and multiply it by B_1, which was calculated earlier and stored in V_{21}; they store the value in V_{12}. Operation 11 takes V_{12} and adds it to V_{13}, and operation 12 subtracts 2 from n and stores that value in V_{10}, because the second iteration is complete. Operations 13 through 21 calculate the next value and multiply it by B_5. [Muss wohl B_3 heissen]

One flaw in Ada's program is that she does not use a variable to keep track of each iteration while calculating the fractional values, as she does with V_{10} when she computes the product of the fractions by the previous Bernoulli numbers.

Eugene Eric Kim, Betty Alexandra Toole: *Ada and the First Computer*. Scientific American, 280 (1999): 76-81.

Das allererste Computerprogramm? (19)

Das Bernoulli-Programm – hier einmal moderner in Java:

```
[0]       n = 4; B1 = 1.0/6.0; B3 = -1.0/30.0; B5 = 1.0/42.0;
//-------------------------------------------------------------
[1-6]     A0 = -0.5 * (2*n - 1) / (2*n + 1);
[7]       i = n-1;
//-------------------------------------------------------------
[8-9]     A1 = n;
[10-11]   V13 = A0 + A1 * B1;
[12]      i = i-1;
//-------------------------------------------------------------
[1;8-9]   V6 = 2*n; V7 = 2; V11 = A1;  // Loop inits
//-------------------------------------------------------------
[13-14]   V6 = V6 - 1;  V7 = V7 + 1;
[15-16]   Z = (V6 / V7) * V11;
[17-18]   V6 = V6 - 1;  V7 = V7 + 1;
[19-20]   V11 = (V6 / V7) * Z;  // V11 = A3, A5,...
[21-22]   V13 = V13 + V11 * B3; // B5 in 2nd iteration
[23]      i = i-1;  // Leave repetition loop when i = 0
//   Here follows a repetition of Operations 13 to 23
//-------------------------------------------------------------
      System.out.println(-V13); // Result B7
```

Mit: **int** i, n;
double A0, A1, V6, V7, V11, V13, Z, B1, B3, B5;

Gegenüber dem Original-Programm wurden i.w. komplexe Ausdrücke in einer einzigen Zeile geschrieben anstatt jede Maschinenoperation in einer neuen Zeile

Das allererste Computerprogramm? (20)

Auch ob Menabrea sein 1842 veröffentlichtes Programm zur Lösung eines linearen Gleichungssystems mit der Analytical Engine wirklich selbst erdacht und programmiert hat, oder ob es nicht eigentlich sogar von Babbage stammt, der es bei seinen Vorträgen in Turin zur Illustration verwendet haben könnte, bleibt zweifelhaft. Denn Babbage schreibt in seinen Memoiren, wie er in Turin ausführlich mit Menabrea und anderen Wissenschaftlern über die Analytical Engine diskutierte: „Around the room were hung the formula, the drawings, notations, and other illustrations which I had brought with me. I began on the first day to give a short outline of the idea. My friends asked from time to time further explanations of parts I had not made sufficiently clear. [...] It was during these meetings that my highly valued friend, M. Menabrea, collected the materials for that lucid and admirable description which he subsequently published." Einige der Materialien, die Babbage seinerzeit mitgebracht hatte, werden noch in der Akademie in Turin verwahrt; vgl. dazu das Video "Charles Babbage and the Academy of Sciences of Torino", www.youtube.com/watch?v=_p0cSQD26Xk

Babbage hatte jedenfalls die Veranstaltung in Turin explizit dazu benutzt, für seine Maschine „Reklame" zu machen; die Veröffentlichung von Menabrea war mit Babbage abgestimmt. Dieser schrieb später Angelo Sismoda: „Mit der Entdeckung der Analytischen Maschine bin ich meinem Land, und sogar, wie ich fürchte, meinem Zeitalter, so weit voraus, dass es für den Erfolg der Maschine von grosser Bedeutung ist, dass sie nicht allein von meinem unbestätigten Zeugnis abhängt. Ich habe deshalb das Treffen von Turin gewählt, um die Sache zu publizieren."

Und Allan Bromley, der in den 1980er-Jahren das Archivmaterial zu Babbage im Science Museum in London studiert hatte (siehe *Allan G. Bromley: Charles Babbage's analytical engine, 1838; Annals of the History of Computing 4.3, 1982, 196-217* sowie *Allan G. Bromley: The Evolution of Babbage's Calculating Engines; Annals of the History of Computing 9.2, 1987, 113-136*) schreibt: „About 50 fragments of 'programs' (the modern term is somewhat misleading) for the Analytical Engine exist. They cover a variety of appli-

Das allererste Computerprogramm? (21)

cations such as the evaluation of simple formulas, recurrence relations, manipulation of power and trigonometric series, and the solution of simultaneous equations."
Und an anderer Stelle: „Babbage prepared, in total, relatively few user-level programs, and many of them seem to be attempts to clarify difficulties of which he was aware but could not overcome. [...] Some two dozen programs for the Analytical Engine exist dated between 1837 and 1840. [...] One group of programs deals with the tabulation of polynomials by difference techniques. [...] A second group of programs deals with the tabulation of iterative formulas of varying complexity. [...] A third group of programs deals with the solution of simultaneous equations by variants of Gaussian elimination. Part of the group explores various methods of doing the row reductions, with a view to minimizing the computing time. [...] These programs were reported by Babbage at Turin in 1840, written up by Menabrea, and translated into English by Augusta Ada King. That work is all essentially derivative, however; the programming ideas, and almost all of the examples, were developed by Babbage by 1840." So gesehen hatten vielleicht sowohl Menabrea als auch Ada Lovelace den gleichen Gehilfen (oder gar Ghostwriter?) für ihre Programme – Charles Babbage, der Erfinder des programmgesteuerten Digitalcomputers.

Charles Babbage (1791 – 1871)

Charles Babbage was the son of a banker, who was himself the son and grandson of goldsmiths. [...] In 1810 he entered Trinity College, Cambridge. [...] He discovered that he already knew more of mathematics than his tutors. [...]

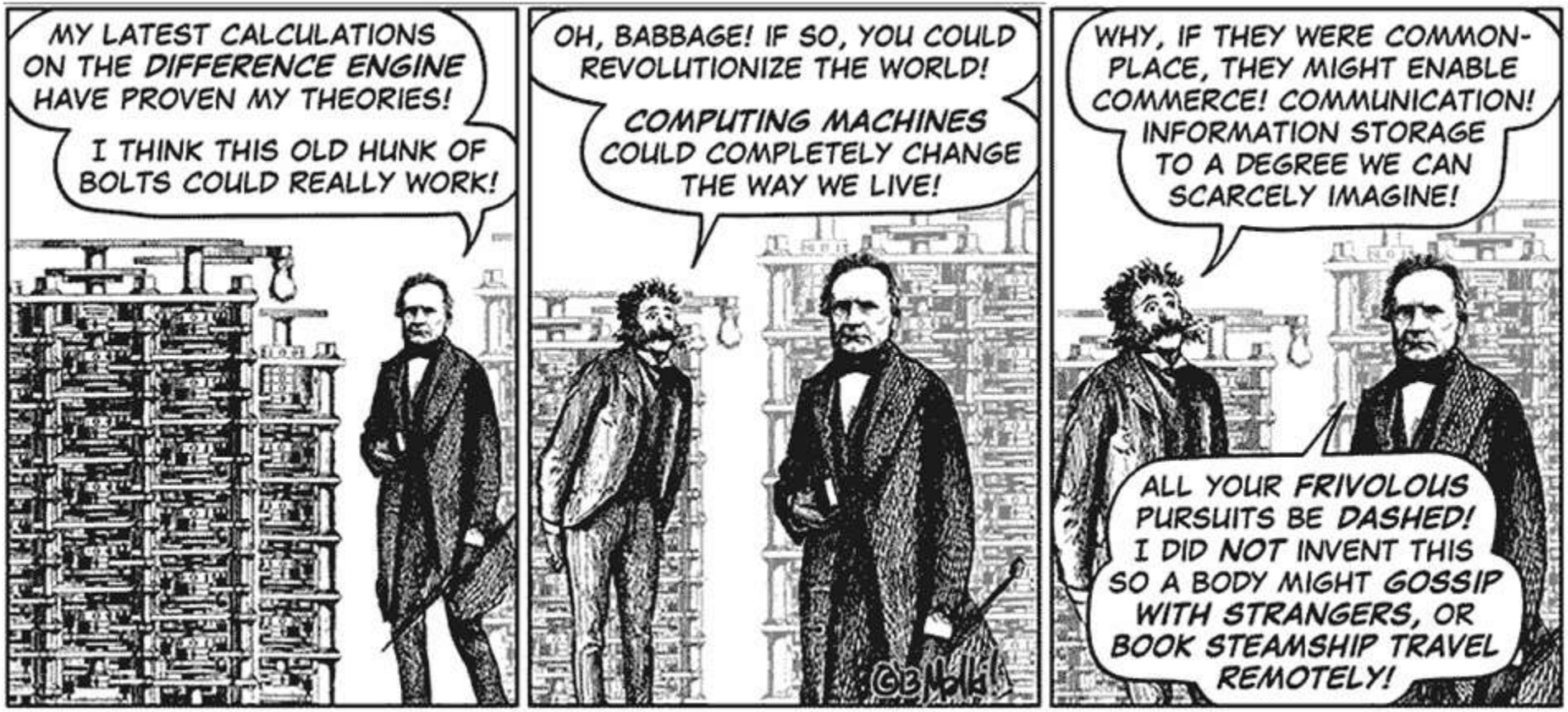
No one doubted that Charles Babbage was brilliant. Nor did anyone quite understand the nature of his genius, which remained out of focus for a long time. What did he hope to achieve? For that matter, what, exactly, was his vocation? [...]

Babbage did not quite belong in his time, which called itself the Steam Age or the Machine Age. He did revel in the uses of steam and machinery and considered himself a thoroughly modern man, but he also pursued an assortment of hobbies and obsessions — cipher cracking, lock picking, lighthouses, tree rings, the post — whose logic became clearer a century later. Examining the economics of the mail, he pursued a counterintuitive insight, that the significant cost comes not from the physical transport of paper packets but from their “verification” — the calculation of distances and the collection of correct fees — and thus he invented the modern idea of standardized postal rates. He loved boating, by which he meant not “the manual labor of rowing but the more intellectual art of sailing.” He was a train buff. He devised a railroad recording device that used inking pens to trace curves on sheets of paper a thousand feet long: a combination seismograph and speedometer, inscribing the history of a train’s velocity and all the bumps and shakes along the way. [...] He did become expert on the manufacture of Nottingham lace; also the use of gunpowder in quarrying limestone; precision glass cutting with diamonds; and all known uses of machinery to produce power, save time, and communicate signals. He analyzed hydraulic presses, air pumps, gas meters, and screw cutters. [...]

[James Gleick – The information: a history, a theory, a flood. Pantheon Books, 2011.]

Charles Babbage

Als junger Mann wollte er die Bibel testen und wissen, warum Jesus über das Wasser gehen konnte. Für diesen Zweck baute er sich entsprechende Schuhe; es gelangen ihm einige erstaunliche Schritte. Er wollte auch wissen, wie es für ein Brot ist, wenn es gebacken wird. Dazu legte er sich in einen geheizten Backofen, musste allerdings abbrechen, als es ihm zu knusprig wurde. -- Elmar Schenkel



http://wondermark.com/953 (David Malki)

Non est facta pro his qui olera aut pisculos vendunt.

-- Gottfried Wilhelm Leibniz über seine eigene Rechenmaschine

The EDSAC Film (1951)

www.youtube.com/watch?v=6v4Juzn10gM

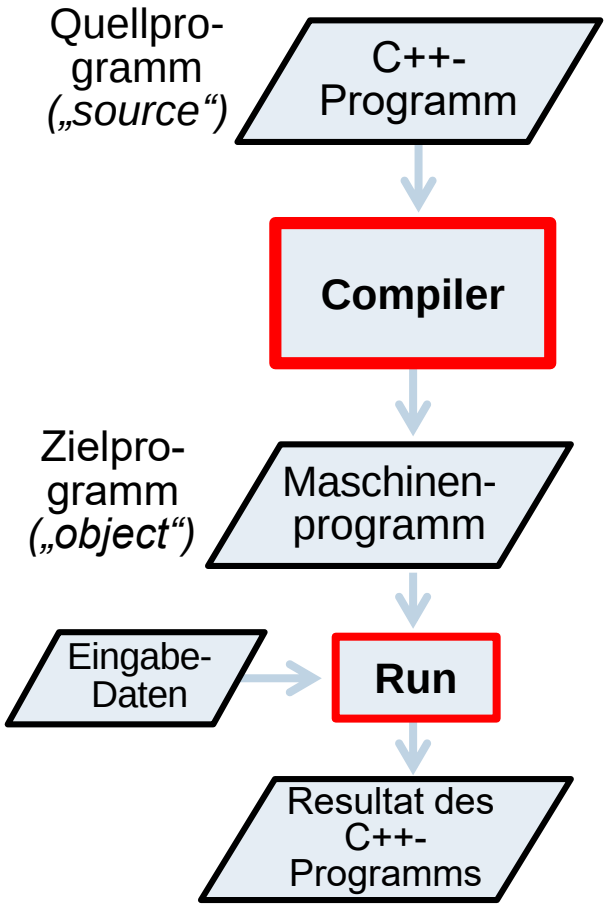
Wenn man wissen will, wie das Programmieren eines Computers um 1950 ablief, dann möge man sich den von Maurice Wilkes (1913–2010), einem Computerpionier aus Cambridge, der den EDSAC-Computer konstruierte und 1967 den Turing Award erhielt, kommentierten Film ansehen.

Ende der längeren historischen Notiz

Compiler

Von Compilern haben wir ja bereits ausführlich (auch in „Informatik I“) Gebrauch gemacht

- Übersetzt ein Programm einer (höheren) Programmier-sprache (z.B. C++) ganz in Maschinensprache




```
sc (char x[], char y[])
{ int i; i=0;
  while ((x[i]=y[i]) != 0) i++; }
```

```
sc: subi $sp, $sp, 4
    sw $s0, 0($sp)
    add $s0, $zero, $zero
L1: add $t1, $a1, $s0
    lb $t2, 0($t1)
    add $t3, $a0, $s0
    sb $t2, 0($t3)
    addi $s0, $s0, 1
    bne $t2, $zero, L1
    lw $s0, 0($sp)
    add1 $sp, $sp,4
    jr $ra
```

Eine Maschine kann keine Programme einer höheren Sprache ausführen, sondern nur Maschinenprogramme

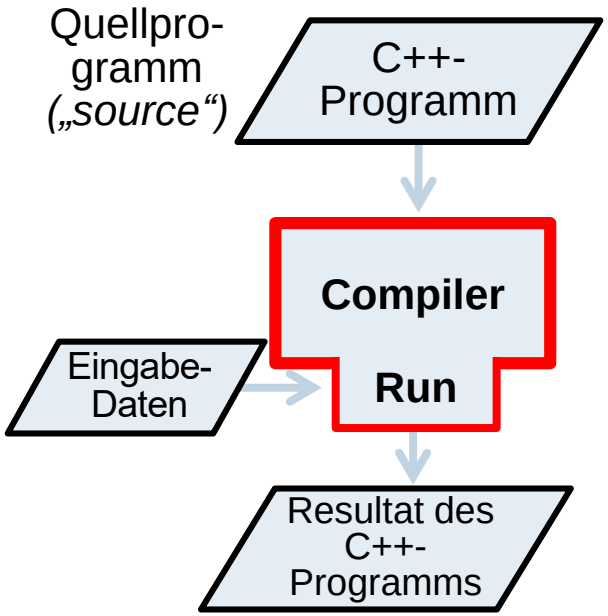
Die Maschinensprache ist Menschen nicht zumutbar

Transformation muss be-deutungsäquivalent sein!

Maschinenprogramm hier in „symbolischer“ Form; eigent-lich ist jede Zeile eine Folge aus 0 und 1 („Maschinencode“)

Compiler

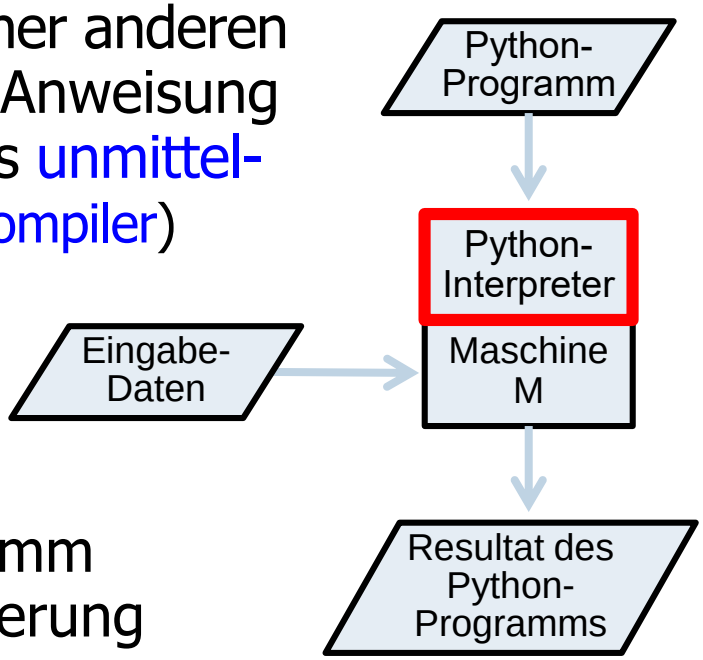
- Übersetzt ein Programm einer (höheren) Programmier-sprache (z.B. C++) ganz in Maschinensprache



→ Interpreter

Interpreter

- Programm, welches ein Programm einer anderen Programmiersprache Anweisung für Anweisung nur intern analysiert und diese jeweils unmittelbar ausführt (im Gegensatz zu einem Compiler)
- Typisch für Dialog-, Skript- und Kommandosprachen
  - Z.B. Perl, Python, oder Ruby
- Vorteil: Einfaches Testen, da Programm sofort (weiter) ausführbar nach Änderung
- Nachteil: Programmausführung dauert länger als die Ausführung übersetzter (compilierter) Programme
  - Analyse, Adressberechnung etc. wird z.B. bei Schleifen neu bei jedem Durchlauf (statt einmalig) gemacht!

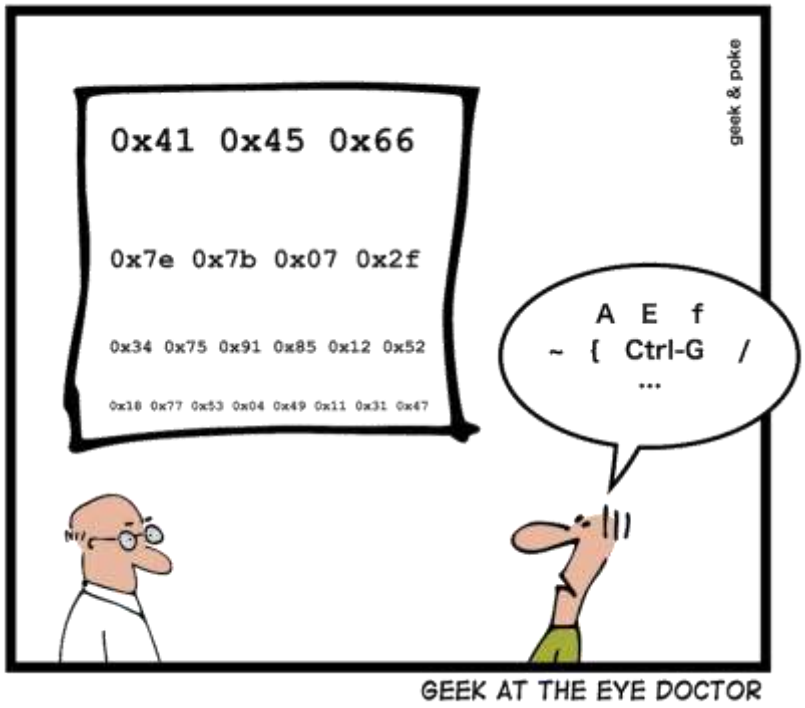


Java stellt eine Zwischenform dar: Java wird in Bytecode übersetzt, dieser wird dann interpretiert

Java-Bytecode

- Bytecode ist die Maschinensprache der Java-VM
  - Bei der Java-VM handelt es sich um eine Stackmaschine

- Bytecode ist ziemlich kompakt: Die meisten Instruktionen („Operationen“) sind nur 1 Byte (= 8 Bit) lang
  - Kennzeichnung durch einen 8-Bit-Operationscode („Befehlsschlüssel“)
  - Haben zusätzlich auch eine „symbolische“ Bezeichnung, z.B.:
    - **add** mit Opcode 01100000 (dezimal 96, hexadezimal 0x60)
    - **pop** mit Opcode 01010111
    - **iconst_3** mit Opcode 00000100



Übersetzung in Java-Bytecode

```
int abc (int p) {
  int i;
  i = 5;
  int j = 7;
  int k = j+i+j+i+3;
  return k;
}
```

iconst und bipush wirken analog

```
0  iconst_5
1  istore_1   i
2  bipush 7
4  istore_2   j
5  iload_2    j
6  iload_1    i
7  iadd
8  iload_2    j
9  iadd
10 iload_1    i
11 iadd
12 iconst_3   k
13 iadd
14 istore_3   k
15 iload_3    k
16 ireturn
```

- Die Speicherplätze für Variablen werden vom Compiler durchnummeriert
  - p auf Platz 0, i auf 1, j auf 2, k auf 3
- Diese Nummern werden dann bei istore und iload als „Adressen“ verwendet
- Übergabe des Rückgabewertes via Stack
  - Daher vor „ireturn“ noch „iload_3“ für k

Übersetzung in Java-Bytecode

```
int abc (int p) {
  int i;
  i = 5;
  int j = 7;
  int k = j+i+j+i+3;
  return k;
}
```

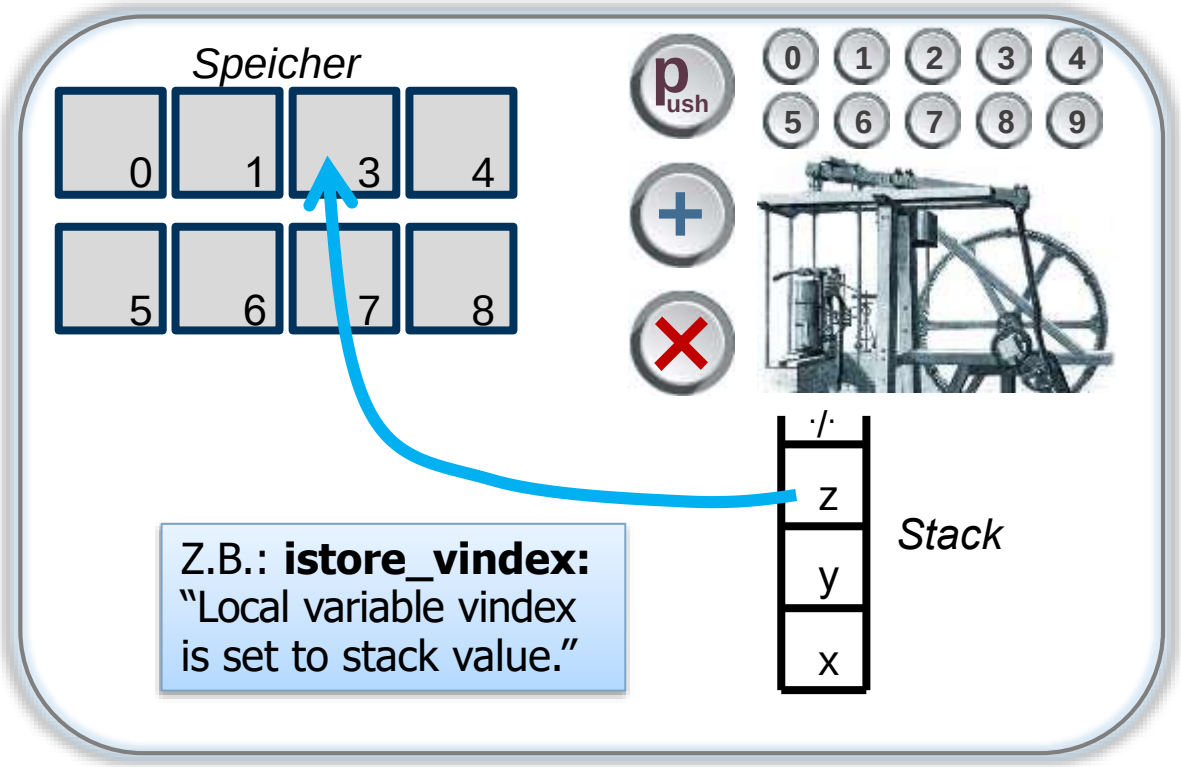
```
0  iconst_5
1  istore_1   i
2  bipush 7
4  istore_2   j
5  iload_2    j
6  iload_1    i
7  iadd
8  iload_2    j
9  iadd
10 iload_1    i
11 iadd
12 iconst_3   k
13 iadd
14 istore_3   k
15 iload_3    k
16 ireturn
```

Vollständig in Binärcode weiterübersetzt sieht das Ergebnis so aus:

```
0000 1000 0011 1100 0001 0000 0000 0111
0011 1101 0001 1100 0001 1011 0110 0000
0001 1100 0110 0000 0001 1011 0110 0000
0000 0110 0110 0000 0011 1110 0001 1101
1010 1100
```

Übersetzung in Java-Bytecode

- Um die Wirkung der Befehlssequenz zu verstehen, denken wir uns eine Stackmaschine
  - Jeder einzelne Befehl ist als Operation mit dieser Maschine festgelegt („operationale Semantik“)




```
 0  iconst_5
 1  istore_1   i
 2  bipush 7
 4  istore_2   j
 5  iload_2    j
 6  iload_1    i
 7  iadd
 8  iload_2    j
 9  iadd
10  iload_1    i
11  iadd
12  iconst_3   k
13  iadd
14  istore_3   k
15  iload_3    k
16  ireturn
```

Übersetzung in Java-Bytecode – zweites Beispiel

```
void q() {
  int i=0, j=0;
  boolean b = false;
  b = (4*i + (3+j)*2) > j+7*i;
  b = b & (33 > 2+j);
}
```

Postfix!

```
 0  iconst_0
 1  istore_0      i
 2  iconst_0
 3  istore_1      j
 4  iconst_0
 5  istore_2      b
 6  iconst_4
 7  iload_0
 8  imul
 9  iconst_3
10  iload_1
11  iadd
12  iconst_2
13  imul
14  iadd
15  iload_1
16  bipush 7
18  iload_0
19  imul
20  iadd
21  if_icmpgt 28
24  iconst_0
25  goto 29
28  iconst_1
29  istore_2
30  iload_2
31  iconst_2
32  iload_1
33  iadd
34  bipush 33
36  if_icmplt 43
39  iconst_0
40  goto 44
43  iconst_1
44  iand
45  istore_2
```

Man kann sich den Java-Bytecode der Klasse xx, den der Java-Compiler (z.B. in der Datei xx.class) generiert hat, in symbolischer Form anzeigen lassen. Bei Linux etwa mit dem Konsolekommando

```
javap -c xx
```

Unter Eclipse gibt es auch diverse Byte-code-Viewer / -Visualizer als plug-ins.

Tipp: Dieses ausprobieren, man macht interessante Entdeckungen!

Denkübung: Wie ändert sich der Bytecode bei „&&“ statt „&“? Wieso?

VM-Instruktionen

- Instruktionen operieren i.Allg. auf dem Operandenstack
  - Aber einige dienen auch nur der Steuerung des Kontrollflusses
- Instruktionen werden durch einen Operationscode gekennzeichnet (1 Byte, daher max. 256 Instruktionen)
  - Evtl. enthalten nachfolgende Bytes auch Operanden (z.B. Index einer Variablen oder einen direkten Wert, wie z.B. bei bipush); einige Instruktionen sind daher auch zwei oder mehr Byte lang
- Neben nachfolgenden Beispielen gibt es noch weitere
  - Siehe z.B. *Tim Lindholm et al.: The Java Virtual Machine Specification:* https://docs.oracle.com/javase/specs/index.html
  - Oder bei Wikipedia *Java bytecode instruction listings:* https://en.wikipedia.org/wiki/Java_bytecode_instruction_listings

Eine Auswahl von VM-Instruktionen

iconst_m1
Push the integer -1 onto the stack.

iconst_0, ... iconst_5
Push the integer 0,..., 5 onto the stack.

iload_vindex
The value of the local variable at vindex in the current Java frame is pushed onto the operand stack.

istore_vindex
Local variable vindex in the current Java frame is set to stack value.

nop
Do nothing

bipush byte
Push one-byte signed integer.

pop
Pop top stack word from the stack.

iadd
Top stack values must be integers. Two values are added and replaced on the stack by their integer sum.

imul
...replaced on the stack by their integer product.

vindex: die Variablen eines „frames" sind durchnummeriert; vindex ist dabei die Nummer („index") einer Variablen

Ein Bytecode-Beispiel: Fakultätsfunktion rekursiv

$$f(n) = \begin{cases} 1 & \textit{falls } n = 1 \\ n \times f(n-1) & \textit{falls } n > 1 \end{cases}$$

```
0  load_0
1  iconst_1
2  if_icmpne  5
3  iconst_1
4  ireturn

5  iload_0
6  iload_0
7  iconst_1
8  isub

9  invokestatic  #29

10 imul
11 ireturn
```

Der Parameter n

Man kann Bytecode wie jede andere Programmiersprache benutzen und Programme schreiben – das ist allerdings doch etwas mühsam und fehleranfällig

Methodenaufruf: Die Bezeichnung der Methode erfolgt indirekt über einen Index (hier #29) einer Namenstabelle, wo die Methodennamen aufgeführt sind – hier ist es rekursiv die Methode selbst.

Oben auf dem Stack steht n-1 (als Parameter), darunter n.

Die Parameter für die gerufene Methode werden konventionsgemäss vorher auf den Stack gelegt, das Ergebnis findet sich danach ebenfalls auf dem Stack.

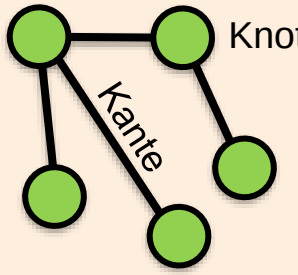
Eine kleine „Reverse-Engineering“-Aufgabe

Welche Funktion des in Speicherplatz 0 abgelegten Parameters berechnet folgendes Bytecode-Programm?

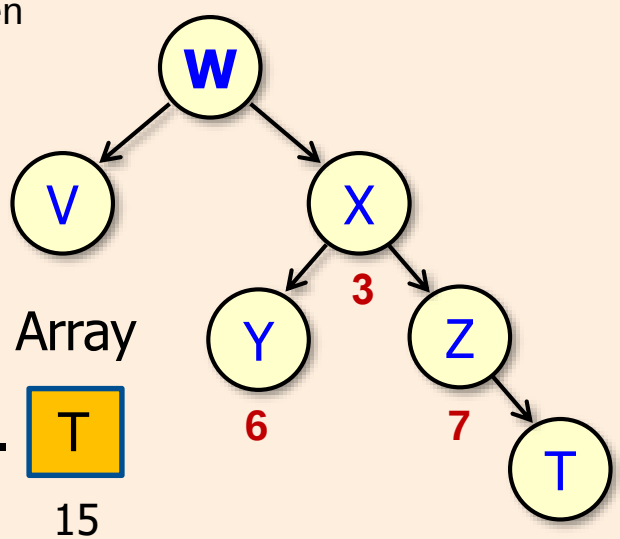
```
0  iconst_1
1  istore_1
2  iload_0     // push parameter (variable 0) on stack
3  ifle 14     // go to 14 if negative or null
6  iload_1
7  iload_0
8  imul
9  istore_1
10 iinc 0, -1  // decrement variable 0 by 1
11 goto 2
14 iload_1
15 ireturn
```

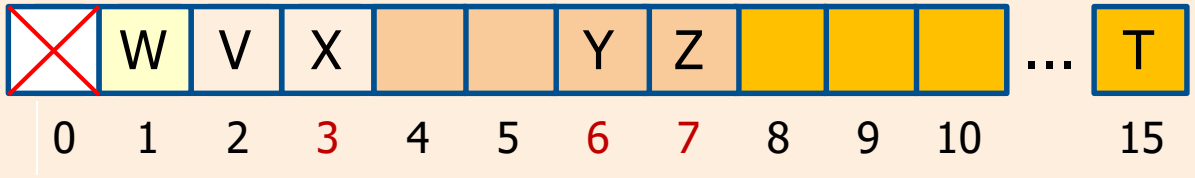
Resümee des Kapitels

- Bäume
  - Definition, Charakterisierung

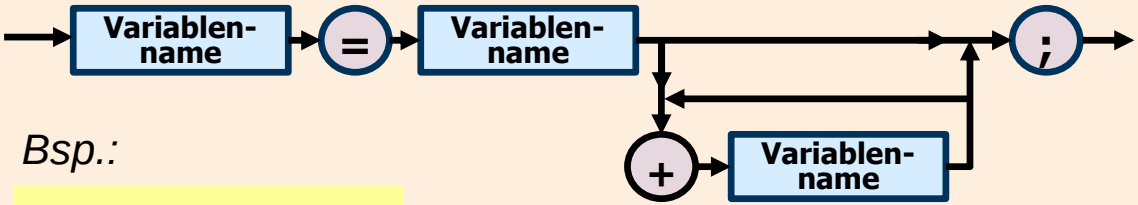


- Wurzelbäume
  - Diverse äquivalente Darstellungen
  - Binärbaum niveauweise repräsentiert in einem Array





- Syntaxdiagramme
  - Generierung aller syntaktisch korrekten Programme
    (das ist eine unendliche Menge!)



Bsp.:

a = b + c + d ;

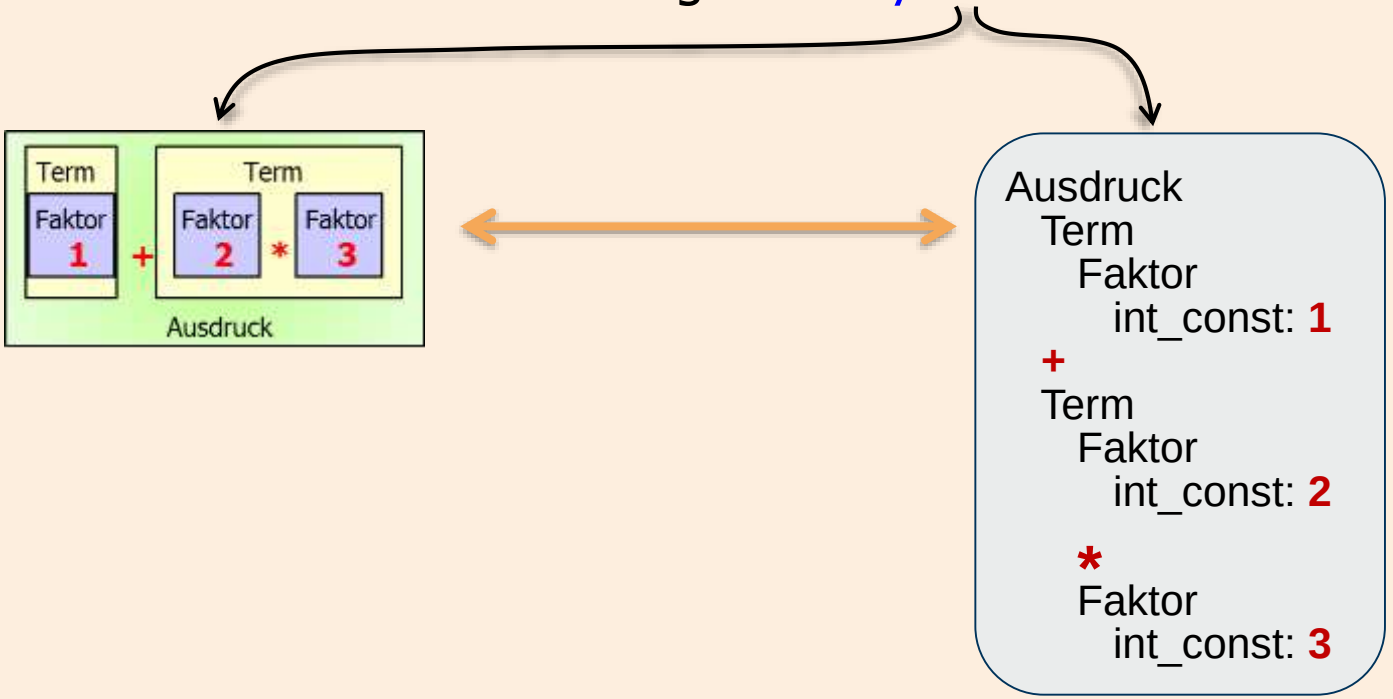
Resümee des Kapitels (2)

- Syntaxanalyse arithmetischer Ausdrücke

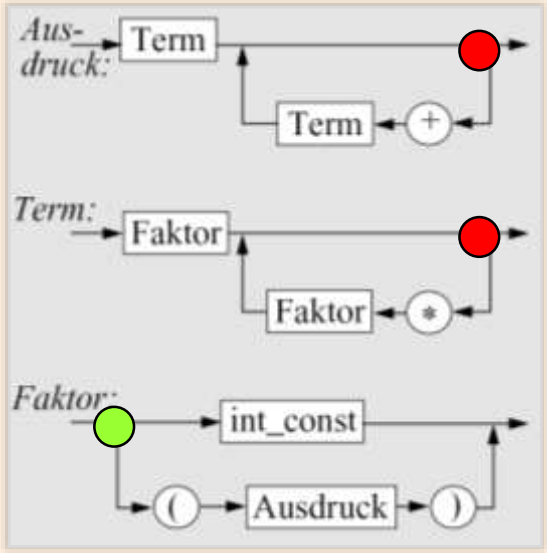
  Beispiel: **1 + 2 * 3**

  → Struktur explizit machen (z.B. Bindungspriorität; implizite Hierarchie)

  → Automatische Generierung eines Syntaxbaums

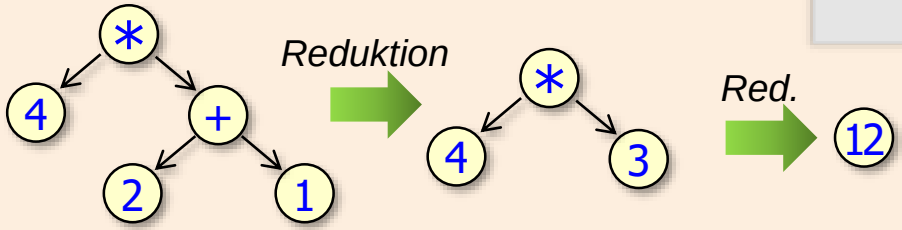


Resümee des Kapitels (3)



- Syntaxanalyse durch rekursiven Abstieg
  → Parser als „programmierte" Syntaxdiagramme

- Auswertung von Operatorbäumen



- Traversieren von Bäumen in inorder bzw. postorder
  - Liefert bei Operatorbäumen den Ausdruck in Infix- bzw. Postfixnotation

Resümee des Kapitels (4)

Transformation von Darstellungen

- Compiler-Aspekte
  - Infix → Postfix
  - Postfix-Auswertung
  - } mit Stack
  - Ziffernfolgen → Zahlen: Horner-Schema

- Infix-Ausdrücke
  - Codegenerierung für eine Stackmaschine
  - Interpreter („Taschenrechner")

(1+2)*3 →

```
push(1)
push(2)
plus
push(3)
mult
```

```
while (c == '+') { …
  stk.push(stk.pop() + stk.pop());
}
```

- Übersetzung Java → Bytecode
  - Java-VM als Bytecode-Interpreter

```
0 iconst_5
1 istore_1
2 bipush 7
3 istore_2
4 iload_2
5 iload_1
6 iadd
```