Version vom
2. März 2020

1.

Ein Algorithmus
und seine Implementierung in Java

Lernziele Kapitel 1 Ein Algorithmus in Java

- Fähigkeit, elementare Java-Programme erstellen zu können
- Kenntnis von Argumenten und Methoden zu Korrektheit von Algorithmen (z.B. Invarianten, Induktion, Semantikkalkül)
- Verständnis von Prinzipien der kryptographisch gesicherten Kommunikation (Public-Key, Diffie-Hellman-Prinzip)

Thema / Inhalt

Wir lernen als „running gag" ein uraltes Verfahren zur Multiplikation von zwei Zahlen, die „**altägyptische Multiplikation**", kennen. Es ist einfach anwendbar, erstaunlich effizient und für das Dualsystem (also für CPUs heutiger Computer und Mikroprozessoren) bestens geeignet. Dass unser Java-Programm bei der Multiplikation mit 0 abstürzt, sonst aber für alle natürlichen Zahlen korrekt ist, erscheint zunächst verwunderlich. Um die **Korrektheit** zu beweisen, entwickeln wir formale (d.h. auf Prinzipien der Mathematik und der Logik) beruhende Verfahren. Die erstaunliche Einsicht: Erstens ist bei einem Algorithmus weniger das, was sich ändert, als das was gleich bleibt („Invariante") relevant; und zweitens sind Programme nicht einfach dahingeschriebene Anweisungen an einen Computer, sondern selbst mathematische Objekte, die sich mit einem **mathematischen Kalkül** behandeln und dadurch auch verifizieren lassen.

Die „altägyptische Multiplikation" lässt sich in kanonischer und einfacher Weise verallgemeinern zu einem Verfahren, das sehr effizient die Exponentialfunktion x^y realisiert. Dieses stellt den

Thema / Inhalt (2)

Kern wichtiger **kryptographischer Verfahren** dar, womit die Kommunikation in offenen Netzen verschlüsselt und abgesichert wird. Das Erstaunliche ist, dass man über das unsichere Netz mit einem Unbekannten einen Geheimschlüssel (zur Verwendung bei der späteren eigentlichen Kommunikation) vereinbaren kann, der tatsächlich „geheim" bleibt, auch wenn Andere das Verfahren genau kennen und jedes Bit der Schlüsselvereinbarung (und der nachfolgenden damit verschlüsselten Kommunikation) abhören können. Als die Geheimdienste entdeckten, dass auch total überwachte Personen gemeinsame Geheimnisse entwickeln können, waren sie elektrisiert und versuchten, daraus selbst ein streng gewahrtes Geheimnis zu machen. Geholfen hat es nicht viel, bald darauf wurde dies unabhängig auch in der akademisch-zivilen Welt entdeckt!

Ist das Prinzip der „altägyptische Multiplikation" genauso gut wie das schriftliche Multiplikationsverfahren, das wir in der Schule lernen? Was aber soll hier „gut" heissen – und kann man eine solche Qualitätsfrage unabhängig vom Kontext generell beantworten? Um eine tiefergehende Diskussion in späteren Kapiteln vorzubereiten – schliesslich geht es um den zentralen Begriff der **Komplexität** – schätzen wir ab, wie der Zeitbedarf bei der Multiplikation von x mit y funktional von den beiden Parametern x und y abhängt.

Der **Algorithmusbegriff** liefert viel Stoff für den eingeflochtenen historischen Themenstrang dieses Kapitels. Benannt nach einem persischen Mathematiker und Astronomen des 9. Jahrhunderts, ist er historisch mit den Rechenregeln bei Verwendung der indisch-arabischen Dezimalziffern (einschliesslich der Wahnsinnserfindung der Null) verbunden, wird heute aber in einem viel umfassenderen Sinne verstanden und löst Befürchtungen und Ängste aus – entscheiden evtl. rational-kalte Algorithmen im Dienste diffuser Mächte über unser Schicksal? Auch wie die Informatik selbst zu ihrem Namen kam, ist nebenbei vielleicht ganz interessant zu erfahren.

Thema / Inhalt (3)

Aber hauptsächlich dreht sich hier fast alles um **Leibniz**, den Philosophen. Denn Leibniz war keineswegs nur Philosoph, sondern eben auch Mathematiker und vieles mehr – ein echter Universalgelehrter. Ein Vordenker vieler Bestrebungen der Informatik, der (zu seinem Leidwesen, was er selbst aber bestenfalls nur ahnen konnte) seiner Zeit meilenweit voraus war und den man mit Fug und Recht zum Schutzheilgen der Informatik ernennen könnte, sollte sie jemals eines solchen bedürfen. Nach dem dreissigjährigen Krieg in eine Zeit hineingeboren, welche wieder offener war für Wissenschaft und Fortschritt, propagierte er beispielsweise das **Dualsystem** und konstruierte **Rechenmaschinen**, die die vier Grundrechenarten beherrschten. Ein mühsames Geschäft, musste doch alles rein mechanisch funktionieren, wobei Werkzeuge, Werkstoffe und Ingenieurskunst verglichen mit heute noch weit unterentwickelt waren.

Aber eigentlich geht es Leibniz sowieso um mehr als um ein praktisches Rechenwerkzeug (dessen seinerzeit noch weit in der Zukunft liegende ökonomische Wirkung er übrigens recht treffend einschätzt) – er sah dies mehr als ein erstes Beispiel auf dem Weg zur generellen Mechanisierung bzw. **Automatisierung der menschlichen Geisteskraft** (dem „Gemüt", um in der Terminologie von Leibniz zu bleiben), wobei man unterwegs dann eben noch die Logik und die menschliche Sprache formalisieren müsse... Fast meint man, ein verwegenes Programm zur Etablierung der künstlichen Intelligenz herauszuhören! Leibniz war optimistisch und glaubte, dass dies alles irgendwann gelingen sollte. In späteren drei Jahrhunderten mussten aber durch Berühmtheiten wie George **Boole**, Gottlob **Frege**, Bertrand **Russell**, David **Hilbert**, Kurt **Gödel** und Alan **Turing**, um nur einige zu nennen, noch einige Durchbrüche erzielt werden und teils auch Rückschläge eingesteckt werden – und auch heute sind wir noch nicht ganz am Ziel! Die Beiträge von Leibniz' „Nachfolgern im Geiste" sind zu umfangreich, um sie hier zu würdigen, wir werden im Bonus-Teil dieses Kapitels einige ihrer Leistungen aber zumindest andeuten.

Zum Begriff „Algorithmus“

Def.: Nach einem bestimmten Schema ablaufendes Rechenverfahren.

Allgemeiner: Eindeutige Handlungsvorschrift zur schrittweisen Lösung eines Problems bzw. einer Klasse von Problemen.

Ein klassisches Beispiel ist der euklidische Algorithmus, mit dem zu zwei beliebigen positiven ganzen Zahlen (nach endlich vielen Schritten in Form einzelner elementarer Rechenoperationen) deren grösster gemeinsamer Teiler ermittelt wird. Andere typische Beispiele sind Primzahlberechnungen, Apps zum Schachspielen oder Sortierverfahren, mit denen z.B. eine Liste von Wörtern alphabetisch (d.h., wie im Lexikon) angeordnet werden kann.

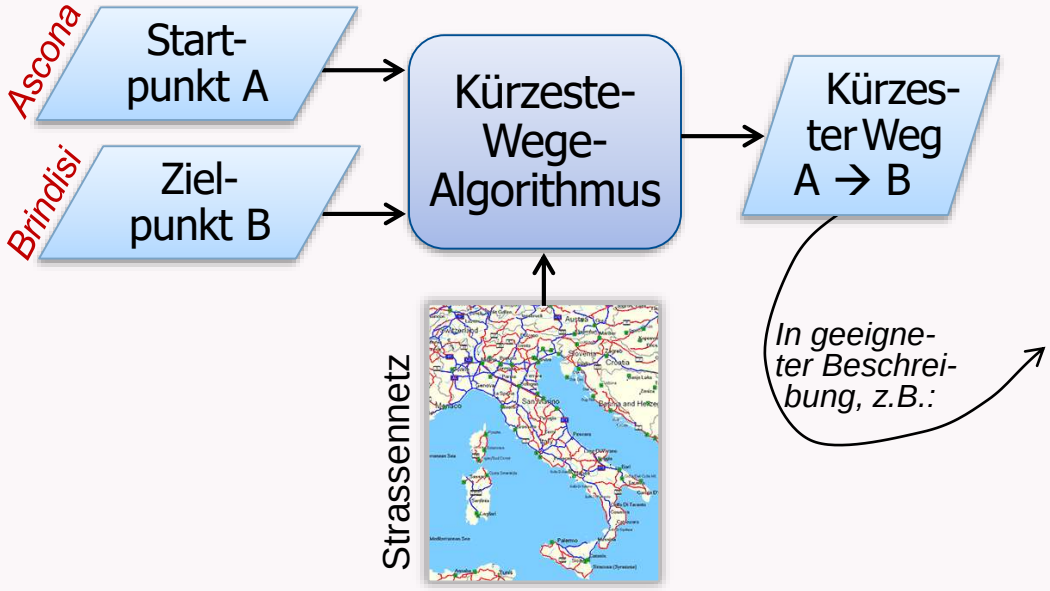
Möchte man präzise Aussagen über Algorithmen bzw. die durch sie lösbaren Probleme treffen (etwa: „für dieses Problem gibt es keinen wesentlich effizienteren Algorithmus als...“), dann muss genau festgelegt werden, welche Operationen bzw. elementaren Schritte man bei einem Algorithmus zulässt; in der Algorithmentheorie spielen daher abstrakte Maschinenmodelle (wie z.B. die klassische Turingmaschine) eine wichtige Rolle.

Algorithmen adressieren Menschen mit ihrem Kontextwissen und sachgerechten Interpretationen, sie sind umgangssprachlich oder in Form von „Pseudocode“ formuliert; Computer als Maschinen benötigen viel detailliertere und präzisere Programme.

Zum Begriff „Algorithmus“ (2)

Ein Kochrezept ist so wenig ein Algorithmus, wie eine Wegbeschreibung ein Navigationssystem ist. -- Sebastian Stiller

Ein Algorithmus kann typischerweise eine ganze Klasse parametrisierter Probleme bearbeiten. So ist die Beschreibung, wie man am schnellsten von **A**scona nach **B**rindisi kommt, kein Algorithmus im eigentlichen Sinne (auch wenn diese eine Handlungsanweisung an einen autofahrenden Menschen darstellt), sondern höchstens das Ergebnis eines Kürzeste-Wege-Algorithmus, der für einen beliebigen Startpunkt **A** den kürzesten Weg zu einem beliebigen Ziel **B** ermittelt (wobei dann Parameter **A** mit „**A**scona“ und **B** mit „**B**rindisi“ instanziiert wurde). Das zugrundeliegende Strassennetz stellt dabei typischerweise einen weiteren Eingabeparameter dar oder ist fest eingebaut.



Bis Locarno Wegweisern folgen; dann E35, A1, A14 / E45 (Autostrada Adriatica) folgen; bei Ausfahrt Bari Nord auf E55 in Richtung Bari fahren und weiter auf Autostradale Bari-Bologna; Ausfahrt Richtung Lecce / Brindisi nehmen und weiter auf Strada Statale SS379 bis Via Provinciale Lecce / Strada Comunale 80 fahren; weiter auf SS16 / SS613; Ausfahrt Brindisi Porta Lecce nehmen.

„Algorithmus“ – historische Aspekte

Historische Begriffsauffassung (ab dem Mittelalter): Rechenrezepte, speziell für die (seinerzeit neue) Dezimalzahlen-Arithmetik und die Algebra.

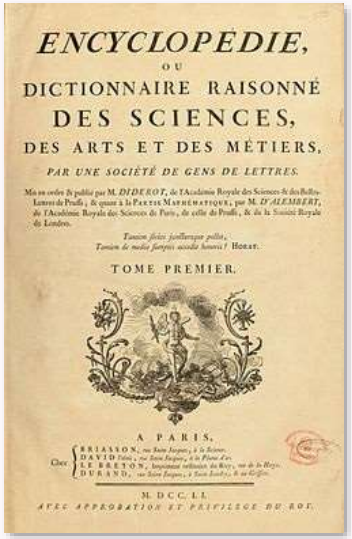
ALGORITHME, ſ. m. *terme arabe*, employé par quelques Auteurs, & ſingulierement par les Eſpagnols, pour ſignifier *la pratique de l'Algebre*. *Voyez* ALGEBRE.

Il ſe prend auſſi quelquefois pour *l'Arithmétique par chiffres*. *Voyez* ARITHMETIQUE.

L'*algorithme*, ſelon la force du mot, ſignifie proprement l'*Art de ſupputer avec juſteſſe & facilité*; il comprend les ſix regles de l'Aritmétique vulgaire. C'eſt ce qu'on appelle autrement *Logiſtique nombrante* ou *numérale*. *V.* ARITHMETIQUE, REGLE, *&c.*

Ainſi l'on dit l'*algorithme* des entiers, l'*algorithme* des fractions, l'*algorithme* des nombres ſourds.

«Encyclopédie ou Dictionnaire raisonné des sciences, des arts et des métiers» von Diderot und D'Alembert, 1751


ENCYCLOPEDIE,
OU
DICTIONNAIRE RAISONNÉ
DES SCIENCES,
DES ARTS ET DES MÉTIERS,
PAR UNE SOCIÉTÉ DE GENS DE LETTRES.
TOME PREMIER.
A PARIS,
M. DCC. LI.

„Algorithmus“ – historische Aspekte (2)

Im „Vollständigen Mathematischen Lexicon“ von Christian von Wolff (1734) findet sich ebenfalls die ältere Begriffsauffassung von der Kunst des Rechnens (vor allem der Dezimalarithmetik):

Algorithmus, unter dieser Benennung werden zusammen begriffen die 4 Rechnungs-Arten in der Rechen-Kunst, nemlich addiren, multipliciren, subtrahiren und dividiren.

Algorithmus infinitesimalis, heissen demnach die Rechnungs-Arten mit unendlichen kleinen Grössen, wie solche von dem Hrn. von Leibnitz erfunden worden.

Vollständiges
Mathematisches
LEXICON,
Darinnen alle
Kunst-Wörter und Sachen,
Welche
In der erwegenden und ausübenden Mathesi vorzukommen pflegen, deutlich erkläret;
Überall aber
Zur Historie der Mathematischen
Wissenschafften dienliche Nachrichten
eingestreuet,
Und die besten und auserlesensten Schrifften, welche iede Materie gründlich abgehandelt, angeführet:
Ferner auch
Die Mund- und Redens-Arten derer Marckscheider
auch hieher gehöriger Künstler und Handwercker, beschrieben;
Und endlich alles zum Nutzen so wohl gelehrter als ungelehrter Liebhaber der vortreflichen Mathematick eingerichtet worden.
Nebst
XXXVI. Kupffer-Tabellen.
LEIPZIG,
bey Joh. Friedrich Gleditschens sel. Sohn.
1734.

Algorithmus, unter dieser Benennung werden zusammen begriffen die 4 Rechnungs-Arten in der Rechen-Kunst, nemlich addiren, multipliciren, subtrahiren und dividiren. Davon unter dem Wort Species mehr zu finden. Algorithmus infinitesimalis, heissen demnach die Rechnungs-Arten mit unendlichen kleinen Grössen, wie solche von dem Hrn. von Leibnitz erfunden worden, davon die Acta Erudit. an. 1684 p. 467 mehrere Nachricht geben. Alhabor, wird der Stern im Munde des grossen Hundes genennet, der sonst auch Sirius heisset, und der gröste am gantzen Himmel ist. s. Hunds-Stern.

Zur Historie:

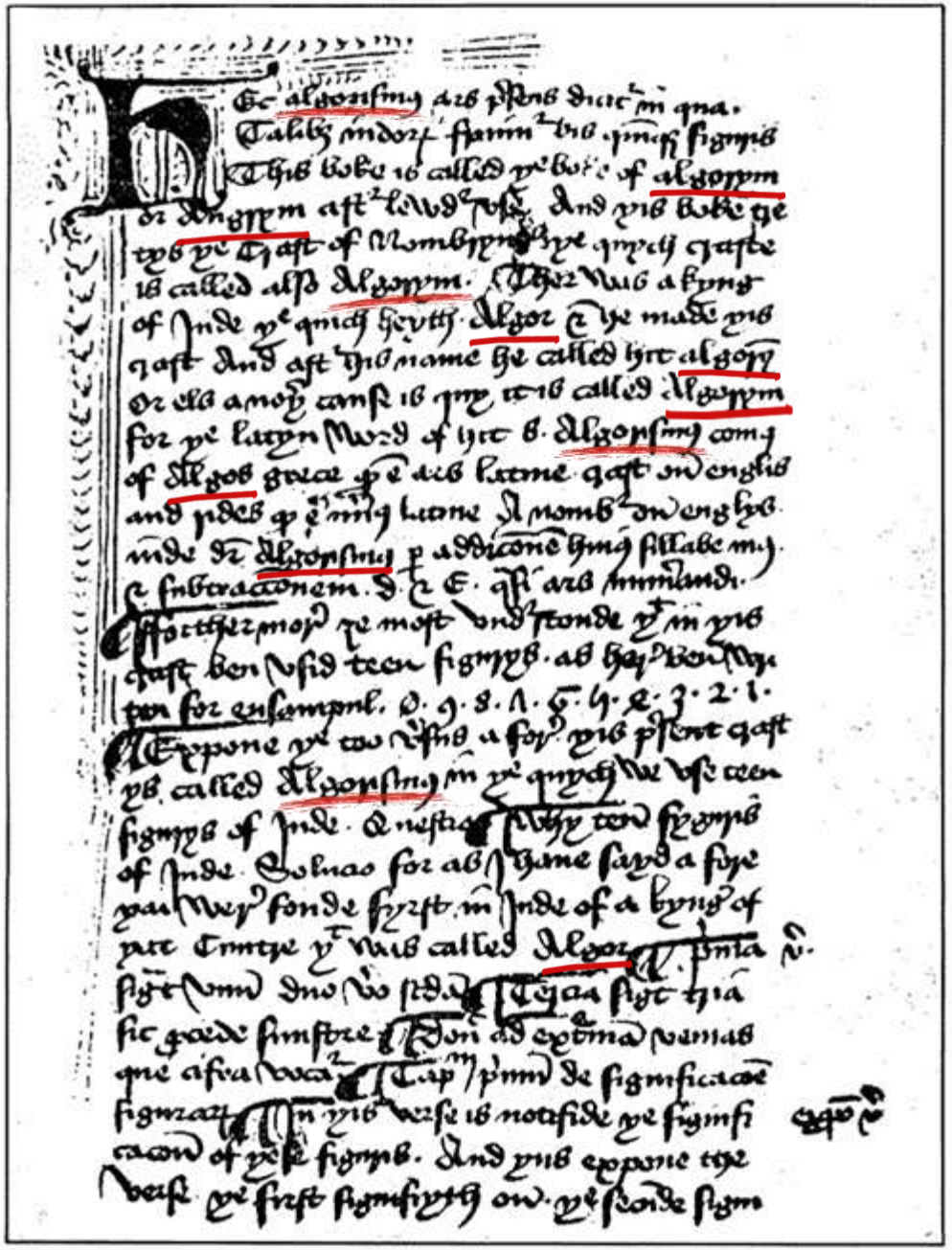
The Craft of Nombryng

Noch deutlich älter (von ca. 1300) ist ein frühenglisches Manuskript über die Kunst des Zählens und Rechnens mit indischen Ziffern, in dem der Begriff „algorism“ (bzw. „algorym“, „augrim“) auftaucht (Eggerton MS. 2622, British Museum: „It measures 7” × 5”, 29-30 lines to the page, in a rough hand. The English is N.E. Midland in dialect.“)

Es handelt sich um einen Kommentar und erweiterte Übersetzung des lateinischen Traktats *Carmen de Algorismo* von Alexandre de Villedieu (ca. 1240).

Damals populär, aber falsch, war die Annahme, dass das Wort auf einen indischen König „Algor“ zurückgeht oder vom griechischen Wort „algos“ abstammt.

Auf den nachfolgenden Seiten finden sich eine Transliteration der ersten Manuskriptseite sowie erläuternde Anmerkungen dazu aus dem Buch „History of Mathematics” von David E. Smith.



Zur Historie:
The Craft of Nombryng – Transliteration

Hec algorism*us* ars p*re*sens dicit*ur*; in quaTalib*us* indor*um* fruim*ur* bis qui*n*q*ue* figuris.[1] This boke is called þe boke of algorym, or Augrym after lewder vse. And þis boke tretys þe Craft of Nombryng, þe quych crafte is called also Algorym. Ther was a kyng of Inde, þe quich heyth Algor, & he made þis craft. And after his name he called hit algory*m*; or els anoþer cause is quy it is called Algorym, for þe latyn word of hit s. Algorism*us* come*s* of Algos, grece, q*uid* e*st* ars, latine, craft on englis, and rides, q*uid* e*st* *nu*mer*us*, latine, A nombur on englys, inde d*icitu*r Algorism*us* p*er* addic*i*one*m* hui*us* sillabe m*us* & subtracc*i*onem d & e, q*ua*si ars num*er*andi.[2]

¶ fforthermore[3] ȝe most vnd*ir*stonde þat in þis craft ben vsid teen figurys, as here bene writen for ensampul, 0 9 8 7 6 5 4 3 2 1 ¶ Expone þe too v*er*sus afor*e*[4]: this p*re*sent craft ys called Algorism*us*, in þe quych we vse teen signys of Inde. Questio. ¶ Why ten fyguris of Inde? Solucio. for as I haue sayd afore þai were fonde fyrst in Inde of a kynge of þat Cuntre, þat was called Algor. ¶ Prima sig*nifica*t unu*m*; duo ve*r*o s*ecun*da[5]: ¶ Tercia sig*nifica*t tria; sic procede sinistre. ¶ Done*c* ad extrema*m* venias, que cifra voca*tur*. ¶ Cap*itulu*m primum de significac*ion*e figurar*um*. ¶ In þis verse is notifide þe significac*i*on of þese figur*is*. And þus expone the verse. Þe first signifiyth one, þe secu*n*de signi [*fiyth tweyne]…

Die schwierig zu verstehende Null (bedeutet für sich nichts!) ist hier in anderer Farbe notiert

1,…,*5*: → *Erläuterungen nächste Slide*

Zur Historie:
The Craft of Nombryng – Erläuterungen

Am besten mit „Gedicht zur Arithmetik“ oder „Ode an die Arithmetik“ übersetzt

[1] These are the two opening lines of the *Carmen de Algorismo*, of Alexandre de Villedieu (*c.1240*). It is translated a few lines later: “This present craft is called Algorismus, in the which we use ten figures of India.”

[2] “Inde dicitur Algorismus per addicionem huius sillabe *mus* & subtraccionem *d* & *e*, quasi ars numerandi (Whence it is called Algorismus by the addition of this syllable *mus*, and the taking away of *d* and *e*, as if the art of numbering).” This idea had considerable acceptance in the 13th century.

[3] “Furthermore,” the *f* being doubled for a capital. “Furthermore you must understand that in this craft there are used ten figures.” The forms of the numerals given in the original were the common ones of the 12th and 13th centuries. The zero was not usually our form, but frequently looked more like the Greek *phi*. The 7, 5, and 4 changed materially in the latter part of the 15th century, about the time of the first printed books. The sequence here shown is found in most of the very early manuscripts, the zero or nine being at the left.

[4] “Explain the two verses afore.”

[5] “The first means one, the second two, the third means three, and thus proceed to the left until you reach the last, which is called cifra.” The author is quoting from the Carmen of Alexandre de Villedieu:

Prima significat unum; duo vero secunda;
Tertia significat tria; sic procede sinistre
Donec ad extremam venias, ques cifra vocatur.

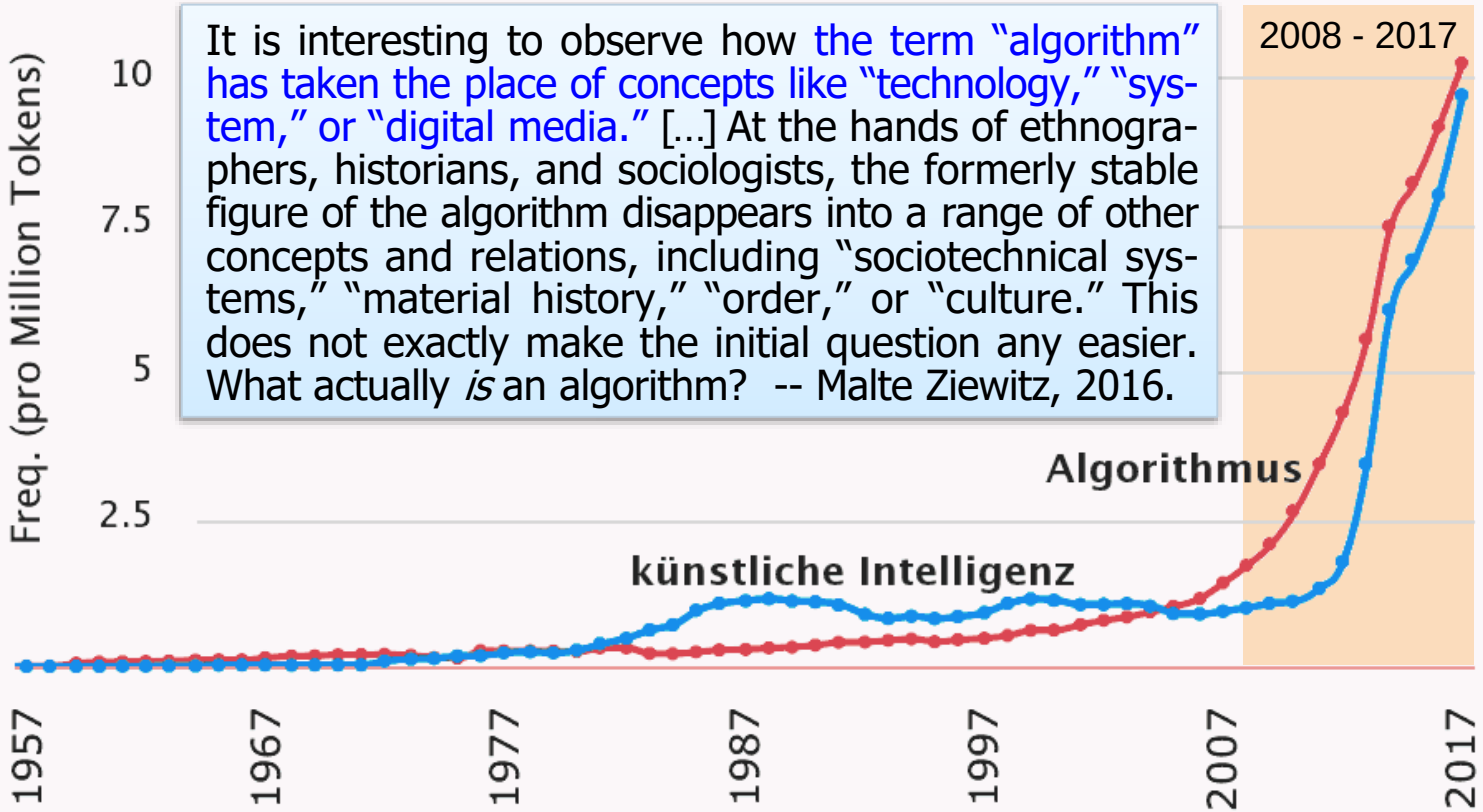
Studenten mussten im Mittelalter viel auswendig lernen, da Kopien handgeschriebener Bücher selten und kostbar waren. Rhythmen und Reime halfen dabei; sie waren der Klarheit aber abträglich – daher entstanden zu solchen Werken oft erläuternde Kommentare.

Die Null wurde also „cifra“ genannt, nach dem Arabischen „sifr“ (صِفْر) für „nichts“; daraus dann auch „Ziffer“

Popularität des Wortes „Algorithmus“

- Begriffe wie „euklidischer Algorithmus“ waren schon vor dem 20. Jh. populär
- Der Logiker Alonzo Church verwendet 1936 bereits „algorithm“ im heutigen Sinne, während Alan Turing um diese Zeit noch von „mechanical process“ spricht
- Mitte des 20. Jahrhunderts hatte sich „Algorithmus“ generell für „Rechenverfahren“ eingebürgert; Ende der 1950er-Jahren wurde zur computerbasierten Lösung numerischer Probleme die „algorithmische“ Programmiersprache ALGOL entwickelt

Popularität des Wortes „Algorithmus“ (2)



Erst vor kurzer Zeit ging „Algorithmus“ in den allgemeinen Wortschatz über, einhergehend mit einer gewissen Verschiebung des Wortgebrauchs und damit auch der Bedeutung.

Die Google-Statistik der vorherigen Seite reicht nur bis zum Jahr 2008; die Wortverlaufskurven aus dem Zeitungskorpus des „Digitalen Wörterbuchs der deutschen Sprache“ (DWDS, www.dwds.de) reichen bis in die jüngste Vergangenheit. Grundlage bildet eine Vielzahl bedeutender überregional verbreiteter deutschsprachiger Tages- und Wochenzeitungen mit über 16 Mio. Dokumenten und insgesamt ca. 375 Mio. Sätzen sowie mehr als 6 Mrd. Tokens. Das DWDS-Projekt wird von der Berlin-Brandenburgischen Akademie der Wissenschaften betrieben.

Algorithmus =?= Künstliche Intelligenz

Wo Künstliche Intelligenz draufsteht, sind oft nur Algorithmen drin. -- www.marconomy.de

Das Departement Informatik der ETH Zürich sah sich im Dezember 2019 veranlasst, die zunehmende Gleichsetzung von Algorithmen und Künstlicher Intelligenz auch innerhalb der ETH zu kritisieren. Anlass war der Entwurf des Strategie- und Entwicklungsplans 2021-2024 der ETH Zürich. In einem Brief an den Präsidenten der ETH heisst es:

We strongly urge a clear distinction between "algorithms" and "artificial intelligence", partly because it reflects reality, but mostly because the confusion of these two very different concepts in popular discourse muddies the already difficult discussion of the implications of "Digitalisierung".

For clarity, "algorithms" are a foundational concept in computer science, and refer to specific recipes or procedures for calculating values or performing actions. "Artificial Intelligence" is a broad field within computer science, and deals with automated decision making, planning, etc. "Machine learning" might be considered a subset of AI, in which algorithms are used together with training data to create models, which are then used as a basis for decision making, etc. The distinction between "algorithm" and "learned model" in ML is particularly important.

Bedeutungsverschiebung von „Algorithmus“

Im Zeitalter von Machine Learning sind Algorithmen nicht mehr einfach nur Rezepte. Wenn es schon eine Küchenanalogie sein soll, dann sind Algorithmen eher das gesamte Kochen. -- Anna Jobin

Die oben angesprochene Bedeutungsverschiebung des Begriffs „Algorithmus“ in jüngster Zeit wird derzeit noch kaum explizit thematisiert; im französischsprachigen Wikipedia (fr.wikipedia.org/wiki/Algorithme, Juli 2019) findet sich allerdings eine Aussage dazu:

> Dans la vie quotidienne, un glissement de sens s’est opéré, ces dernières années, dans la notion d’« algorithme » qui devient à la fois plus réducteur, puisque ce sont pour l’essentiel des algorithmes de gestion du big data, et d’autre part plus universel en ce sens qu’il intervient dans tous les domaines du comportement quotidien. La famille des algorithmes dont il est question effectue des calculs à partir de grandes masses de données (les big data). Ils réalisent des classements, sélectionnent des informations […]. Les implications sont nombreuses et touchent les domaines les plus variés.

Das neu aufgekommene nicht-technische Framing von Algorithmen unter ethischen und kulturellen Gesichtspunkten ist eine Quelle teilweise bizarrer Missverständnisse, etwa wenn Fachleute für Algorithmen benannt werden sollen. Die Soziologen Jonathan Roberge und Robert Seyfert sprechen in ihrem Aufsatz „Was sind Algorithmuskulturen?“ der Informatik sogar prinzipiell eine relevante Meinung dazu ab: „Wir würden sogar so weit gehen zu behaupten, dass der computerwissenschaftliche Diskurs algorithmische Praktiken konzeptuell ausschliesst.“ Die Begründung bleibt allerdings recht unscharf: „...da es ihrer DNA anhaftet, Algorithmen über Präzision und Korrektheit zu definieren. Computerwissenschaftler können Abweichungen einzig menschlichen Routinen zurechnen...“

Wer die Anweisung erhält, mit Milch, Mehl und Äpfeln etwas Schmackhaftes zu kochen, wird nicht zwangsläufig einen Apfelkuchen backen. -- Anna Jobin

Algorithmen im soziotechnischen Kontext

Die Bedeutungsverschiebung von Algorithmen wird auch bei einem Aufsatz von Rob Kitchin "Thinking critically about and researching algorithms" deutlich (2017, Auszüge):

Algorithms need to be understood as relational, contingent, contextual in nature, framed within the wider context of their socio-technical assemblage. From this perspective, 'algorithm' is one element in a broader apparatus which means it can never be understood as a technical, objective, impartial form of knowledge or mode of operation.

Code is not purely abstract and mathematical; it has significant social, political, and aesthetic dimensions, inherently framed and shaped by all kinds of decisions, politics, ideology […]. Whilst programmers might seek to maintain a high degree of mechanical objectivity – being distant, detached and impartial in how they work and thus acting independent of local customs, culture, knowledge and context – in the process of translating a task or process or calculation into an algorithm they can never fully escape these.

Wer diese Eigenschaften von Algorithmen als fabrizierte Fiktion abtue, sei "far from being objective, impartial, reliable and legitimate". Der Autor kritisiert dementsprechend auch die Herangehensweise der Informatik, die diese Eigenschaften von Algorithmen ausblendet:

In computer science texts the focus is centred on how to design an algorithm, determine its efficiency and prove its optimality from a purely technical perspective. If there is discussion of the work algorithms do in real-world contexts this concentrates on how algorithms function in practice to perform a specific task. In other words, algorithms are understood to be strictly rational concerns, marrying the certainties of mathematics with the objectivity of technology. Other knowledge about algorithms – such as their applications, effects, and circulation – is strictly out of frame. As are the complex set of decision-making processes and practices, and the wider assemblage of systems of thought, finance, politics, legal codes and regulations, materialities and infrastructures, institutions, inter-personal relations, which shape their production.

Algorithms – their usage has changed

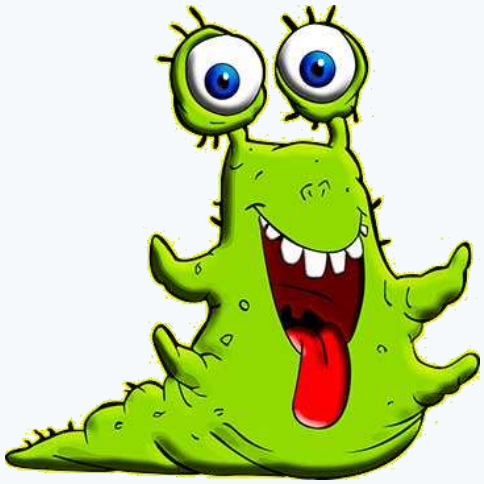
Few subjects are more constantly or fervidly discussed right now than algorithms. But what *is* an algorithm? In fact, the usage has changed in interesting ways since the rise of the internet – and search engines in particular – in the mid-1990s. At root, an algorithm is a small, simple thing; a rule used to automate the treatment of a piece of data. If *a* happens, then do *b*; if not, then do *c*. […] At core, computer programs are bundles of such algorithms. Recipes for treating data. On the micro level, nothing could be simpler. If computers appear to be performing magic, it's because they are fast, not intelligent.

Recent years have seen a more portentous and ambiguous meaning emerge, with the word "algorithm" taken to mean any large, complex decision-making software system; any means of taking an array of input – of data – and assessing it quickly, according to a given set of criteria (or "rules"). This has revolutionized areas of medicine, science, transport, communication, making it easy to understand the utopian view of computing that held sway for many years. Algorithms have made our lives better in myriad ways.

Only since 2016 has a more nuanced consideration of our new algorithmic reality begun to take shape. If we tend to discuss algorithms in almost biblical terms, as independent entities with lives of their own, it's because we have been encouraged to think of them in this way. Corporations like Facebook and Google have sold and defended their algorithms on the promise of objectivity, an ability to weigh a set of conditions with mathematical detachment and absence of fuzzy emotion. No wonder such algorithmic decision-making has spread to the granting of loans / bail / benefits / college places / job interviews and almost anything requiring choice.

[Andrew Smith: Franken-algorithms: the deadly consequences of unpredictable code, www.theguardian.com/technology/2018/aug/29/coding-algorithms-frankenalgos-program-danger]

Algorithmen: grün, schleimig, gefährlich



Sie kamen scheinbar über Nacht. Kinder lauschten verstört dem Abendgespräch der Eltern. Da war von bislang unbekannten Wesen die Rede. Sie mussten grün aussehen, vielleicht schleimig, zumal sie erst gerade den düsteren Tiefen des nahen Sees entstiegen sein mussten, in deren Strömung sie sich zuvor gleichmässig hin und her bewegt hatten – längste Zeit vollkommen unentdeckt. Aber jetzt waren sie plötzlich überall: Algorithmen! So eklig wie sie aussehen mussten, so gefährliche Wesen mussten das sein. Verführerisch seien sie. Sie stifteten Nutzen und erleichterten den Alltag. Damit zögen sie die Menschheit in ihren Bann. Anschliessend hätten sie leichtes Spiel, die Menschheit in kompletter Abhängigkeit zu beherrschen.

Christian Laux, 6.6.2019,
www.inside-it.ch/articles/54644

Algorithmen-Bashing im Feuilleton

Mir gefällt, wenn jemand von Algo-rhythmus spricht. Denke dann immer an Samba, Rumba oder Cha-Cha-Cha.
-- ElChupacabra
https://apps.derstandard.at/userprofil/postings/521489

„Algorithmus" war einmal ein unschuldiges, ein bisschen langweiliges Wort, so ähnlich wie „Grammatik" oder „Multiplikation". [...] In der Presse tauchte das Wort [...] nur dann auf, wenn jemand sagen wollte, dass da etwas Kompliziertes in einem Computer vorging, was man aus Rücksicht auf den Leser jetzt nicht so genau erklären mochte. So ging das bis zum Frühjahr 2010. Dann hielt die Kommunikationswissenschaftlerin Miriam Meckel einen Vortrag in Berlin, in dem sie den Vormarsch der Algorithmen und das Verschwinden des Zufalls beklagte. Wenige Tage später kritisierte Schirrmacher in der FAZ, dass nach dem Ausbruch des Eyjafjallajökull der Luftverkehr aufgrund von Simulationen und „sozialen Algorithmen" stillgelegt wurde. Seither ist kein Monat ohne großen Feuilletonbeitrag über das unbeaufsichtigte Treiben der Empfehlungs- und Filteralgorithmen vergangen, und seit dem Erscheinen von Eli Parisers Buch über die „Filter Bubble" Mitte 2011 ist „Algorithmus" auf dem besten Weg zum Schulhofschimpfwort. -- Kathrin Passig im Januar 2012 im Blog der Süddeutschen Zeitung („Zur Kritik an Algorithmen"). [Siehe dazu auch: Miriam Meckel (2011): Wie der Zufall aus unserem digitalen Leben verschwindet, www.spiegel.de/spiegel/print/d-80451034.html]

Pauschale Kritik an Algorithmen, wie sie in den letzten zwei Jahren häufig zu lesen war, ist ungefähr so sinnvoll wie Kritik am „Rechnen" oder – wie in den 1980er Jahren – an „den Computern". Man müsste erst mal präzise benennen, welche Verfahren man meint. Das scheitert in der Regel an der fehlenden Anschauung und dem fehlenden technischen Verständnis der Kritiker. -- Kathrin Passig im Januar 2012 in der Berliner Gazette, https://berlinergazette.de/kritik-an-algorithmen/

Algokratie

Der Journalist Adrian Lobe verfasst regelmässig Artikel für bekannte Tages- und Wochenzeitungen; im Januar 2018 schrieb er in der Süddeutschen Zeitung unter „Vorgekautes Denken" u.a.:

Klammheimlich haben sich in den letzten Jahren algorithmische Entscheidungssysteme in unseren Alltag geschlichen, die als zentrale Steuerungsinstanzen fungieren. Sie entscheiden autoritativ, ob wir bei der Bank einen Kredit bekommen, welche Informationen wir sehen und, wie in einigen US-Bundesstaaten, wie hoch die Haftstrafe ausfällt. Die technischen Systeme entscheiden über Freiheit oder Unfreiheit und mithin über ureigenste Rechte des Menschen – obwohl sie dazu gar nicht legitimiert sind.

Der Stanford-Soziologe A. Aneesh prägte den Begriff der „Algokratie", eine Herrschaftsform, bei der Programmcodes eine politische Steuerung implementieren. Durch die Abtretung von Wertentscheidungen an soziotechnische Systeme wächst Konzernen wie Google oder Facebook eine politische Macht zu. [...] Der Angriff auf die offene Gesellschaft besteht nicht allein darin, dass die große Masse aus politischen Entscheidungsprozessen ausgeschlossen wird, sondern, dass das Geschäftsmodell des Silicon Valley das Erbe der Aufklärung aufs Spiel setzt: das vernunftgeleitete, für jeden nachvollziehbare Überprüfen und Hinterfragen von Quellen. Die algorithmischen Prozeduren, die unter Ausschluss der Öffentlichkeit stattfinden, sind eine Rückkehr zu jenen Praktiken, wie sie bereits in der mittelalterlichen Geistlichkeit verbreitet waren und leisten einer Refeudalisierung der Gesellschaft Vorschub.

Das Problem ist, dass die Undurchsichtigkeit algorithmischer Prozeduren eine der Grundvoraussetzungen für das Funktionieren der Informationsökonomie ist. Google beruft sich auf die Schutzbehauptung, dass bei einer Offenlegung seines Algorithmus Spammer ihre Splitter in die oberen Suchränge platzieren könnten und die informationelle Architektur kollabieren würde. Es ist ein systemimmanenter Widerspruch, bei dem niemand weiß, wie er aufzulösen wäre.

Der langjährige Google-Chef Eric Schmidt formulierte 2005 das Ziel, für jede Suchanfrage nur noch einen Treffer anzuzeigen. „Wir sollten in der Lage sein, sofort die richtige Antwort zu geben. Wir sollten wissen, was jemand meint." [...] Die Automatisierung des Denkens, die zwischen jeder Programmierzeile zu lesen ist, ist nicht nur ein antiaufklärerisches Vorhaben, sondern auch ein Einfallstor für Autoritarismus. [...] Wenn Schmidt behauptet, Mehrfachantworten seien ein „Bug", also ein Fehler im System, ist das eine Absage an jede Form von Meinungspluralismus.

Herrschaft der Algorithmen?

Mit einem Algorithmus kann man nicht verhandeln. -- Anna Jobin

Jürgen Kuri, Redakteur bei der Computerzeitschrift c't und heise online, schrieb bereits 2010 in der Frankfurter Allgemeine Zeitung über die aufkommende Sorge vor der algorithmischen Herrschaft – unter der Überschrift „Herrschaft der Algorithmen – Die Welt bleibt unberechenbar":

Algorithmen beherrschen die Welt, die Gesellschaft, unser Leben, online wie offline. Hedge-Fonds entscheiden über Wohl und Wehe von Märkten, Firmen und ganzen Volkswirtschaften anhand der Berechnungen, mit denen die Algorithmen der Finanzmathematik die Welt erklären. Die selbständigen Transaktionen der automatisierten Börsensoftware lösen Auf- und Abwärtsbewegungen der Aktienindizes, ja ihren plötzlichen Absturz aus. Staaten und Staatengemeinschaften beurteilen die Lage anhand von Simulationen, in denen Algorithmen aus der Vergangenheit die Zukunft vorauszusagen versuchen. Beratergremien nehmen Politikern mittels Modellen Entscheidungen ab, deren Algorithmen mit historischen Wetterdaten, aktuellen Messwerten, archäologischen Gesteinsanalysen und anderen Daten, die sich zu von Menschen nicht mehr erfassbaren Gebirgen auftürmen, Aussagen über das künftige Klima treffen.

Scoring-Algorithmen bestimmen anhand persönlicher Zahlungsmoral, individuellen Umfelds, Wohn- und Arbeitssituation die Kreditwürdigkeit eines Bürgers. In per WLAN vernetzten Kraftfahrzeugen entscheiden Algorithmen, welche Autobahn die Strecke mit den wenigsten Staus verspricht und wie schnell oder langsam der Wagen fahren muss, um effizient und schnell ans Ziel zu kommen. Smartphone-Apps zeigen anhand von Bevölkerungsdaten und Kriminalitätsstatistik, ob es eine gute Idee ist, die schicke Wohnung ausgerechnet in diesem oder jenem Wohnviertel zu beziehen. Empfehlungsalgorithmen sagen uns, welche Musik wir hören wollen, welches Buch wir lesen möchten, welche Menschen wir treffen sollen. Die Maschinen, die Algorithmen berechnen unser Leben und unsere Zukunft: So ist es, so wird es sein.

www.faz.net/aktuell/feuilleton/herrschaft-der-algorithmen-die-welt-bleibt-unberechenbar-1996485.html

Algorithmic Bias

Beherrschen uns Algorithmen wirklich? Viele Leute, die unentwegt das Wort „Algorithmus" im Munde führen, wissen nicht wirklich, wovon sie reden. Algorithmen als solchen Macht zuzusprechen, ist ein neo-primitiver Technoanimismus und nicht ungefährlich. -- Eduard Kaeser, NZZ vom 23.11.2017

Der Journalist (u.a. Neon, Focus, Wired, SZ, Bayerischer Rundfunk) und „bekennende Nerd/Geek" Michael Moorstedt schrieb unter dem Titel „Maschinenverstand" in der SZ vom 9. Aug. 2017:

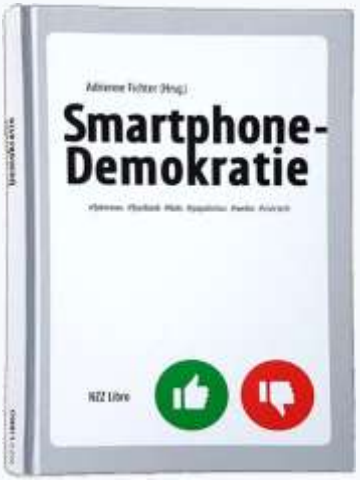
Algorithmen regulieren und organisieren unser Leben online und immer häufiger auch offline. Daten und Zahlen werden eingegeben, und heraus kommt eine Anweisung: Wer bekommt einen Kredit, einen Job und eine Zahnzusatzversicherung? All das entscheiden immer häufiger nicht die netten Sachbearbeiter, sondern Algorithmen, nüchtern und vermeintlich unbestechlich. [...]

Manchmal wirken Algorithmen dabei bevormundend. Etwa wenn Facebook oder Google mal wieder anhand von längst zurückliegenden Klicks entscheiden, was der Nutzer auf seinem Bildschirm zu sehen bekommt oder nicht. Manchmal sind Algorithmen regelrecht kriminell, etwa wenn sie die Abgaskontrollanlage eines Dieselfahrzeugs dahingehend manipulieren, dass das Auto in Prüfsituationen weniger Stickoxide emittiert. Hin und wieder können sie sogar aufs Tiefste in das Leben der von ihnen Betroffenen eingreifen. In den USA berücksichtigen Gerichte bei der Frage, ob ein Straftäter rückfällig wird oder nicht, die Empfehlungen eines Algorithmus des Softwareherstellers Equivant. Der berechnete [...] bei Afroamerikanern fast doppelt so häufig wie bei Weissen fälschlicherweise eine hohe Rückfallgefahr. Gleichzeitig wurde ein späteres, erneutes Vergehen weisser Straftäter fast doppelt so oft nicht vorhergesagt wie bei Schwarzen.

Von „Algorithmic Bias" ist dann die Rede, also von Befangenheit oder Vorurteilen eines Programms. Aufgrund von Beispielen wie diesem forderte Justizminister Heiko Maas deshalb vor Kurzem auf einer Digitalkonferenz ein „digitales Antidiskriminierungsgesetz" und „vorurteilsfreies Programmieren". „Im Rechtsstaat sind alle Entscheidungen begründungspflichtig. Denn nur so kann überprüft werden, ob die Grundlagen, auf denen sie getroffen wurden, richtig, rechtmässig und auch verhältnismässig sind", sagte Maas. „Eine solche Überprüfbarkeit brauchen wir auch, wenn Algorithmen Entscheidungen vorbereiten." [...]

www.sueddeutsche.de/digital/kuenstliche-intelligenz-wie-algorithmen-hass-und-vorurteile-zementieren-1.3620668

Neutrale Algorithmen?



Auszug aus: Anna Jobin: Von A(pfelkuchen) bis Z(ollkontrolle): Weshalb Algorithmen nicht neutral sind. In: Adrienne Fichter (ed.): Die Smartphone-Demokratie. Zürich, NZZ Libro, 2017:

Vor über 30 Jahren publizierte der Technikhistoriker Melvin Kranzberg seine Thesen zur Technologie, wovon die erste lautet: «Technologie ist nicht gut oder schlecht und erst recht nicht neutral.» Seit einigen Jahren hören wir vermehrt von Algorithmen – ein Thema, das sich offenbar immer grösserer Beliebtheit erfreut. Die Diskussionen darüber sind oft polarisiert:

Während in den Augen der einen Algorithmen die Verantwortung für so vieles in unserem täglichen Leben tragen – und sie daher konsequent als entweder heilbringende Technologie oder Sündenbock beschrieben werden – vertreten andere die Meinung, Algorithmen seien nichts Neues, schon immer da gewesen und einfach zu einer modischen Bezeichnung von grundsätzlich agnostischer Computertechnologie geworden. [...]

Jenseits von Extrembeispielen nehmen algorithmische Prozesse in unserem Alltag tatsächlich einen wichtigen Platz ein. Sie berechnen Wettervorhersagen, handeln mit Aktien, steuern Verkehrsampeln und vieles mehr. Ihre Auswirkungen reichen von banal bis lebensentscheidend. Was all diese Prozesse jedoch gemeinsam haben, ist die Tatsache, dass uns durch ihren Einsatz Entscheidungen aus der Hand genommen werden. [...]

Wo entschieden wird, werden Werte gewichtet, und eine in jeder Hinsicht neutrale Entscheidung gibt es nicht. [...] Algorithmen sind nicht gut oder schlecht, aber die Werte, die sie – mit Absicht oder ungewollt – abbilden, können es sein.

Algorithmen: Macht, Ideologie & Fairness

Algorithmen bilden Wertvorstellungen ab – egal, ob diese Abbildungen als solche durchdacht und gewollt sind oder nicht. -- Anna Jobin

Auch Matthias Sander befasst sich in der NZZ vom 27.6.2019 mit der Macht der Algorithmen und spricht anhand eines prominenten Beispiels ein grundsätzliches Dilemma an: Welcher „Ideologie" soll ein „fairer" Algorithmus folgen, wenn die jeweiligen Alternativen unvereinbar sind? (Auszug):

Wenn Algorithmen (mit)entscheiden, [...] welcher Mörder nach Ablauf seiner Gefängnisstrafe verwahrt wird oder nicht, dann ist die Dringlichkeit von gesellschaftlichen Debatten zum Thema sofort klar. Zumal Software wie zum "predictive policing", der vorausschauenden Verbrechensbekämpfung, bereits vieler-orts eingesetzt wird, auch in der Schweiz. Mit Erfolg – und zuweilen wenig wünschenswerten Folgen.

[...] Die Software Compas berechnet Rückfallwahrscheinlichkeiten von Straftätern. Sie wurde in den USA als rassistisch kritisiert, weil sie Schwarzen fast doppelt so oft wie Weissen fälschlicherweise unterstellte, rückfällig zu werden. Allerdings sagt der Algorithmus laut den Entwicklern mit nahezu gleicher Treffsicherheit für Schwarze wie Weisse die tatsächliche Rückfallquote voraus. Dieser vermeintliche Widerspruch hat einen simplen statistischen Ursprung: Die Kriminalitätsrate von Schwarzen ist höher als die von Weissen.

Das Beispiel zeigt, warum wir wissen sollten, wie ein konkreter Algorithmus funktioniert. Die Entwickler von Compas hatten das Ziel, dass über alle Bevölkerungsgruppen hinweg Rückfallgefährdete mit der gleichen Wahrscheinlichkeit zu Recht ins Gefängnis müssen. Die Kritiker hingegen wollen nicht, dass gut zu resozialisierende Schwarze öfter einsitzen als vergleichbare Weisse. Was davon ist fairer? Beide Ansprüche sind laut Mathematikern unvereinbar. Die Frage muss politisch beantwortet werden.

„Blackbox Algorithmus“

Was bei Algorithmen hinten herauskommt, weiss niemand so genau – und warum, noch viel weniger. – Schlagzeile der NZZ, 18.5.2018

Mario Martini 2017 in „Algorithmen als Herausforderung für die Rechtsordnung.“ (JZ, 1017 ff.):

Wie die Algorithmen funktionieren, denen wir die Organisation unseres Lebens immer stärker anvertrauen, verstehen wir immer weniger. Wie aber wir funktionieren, verstehen umgekehrt die Algorithmen im Maschinenraum moderner Softwareanwendungen immer besser. [...]

Eine intransparente und dadurch für Betroffene nicht nachvollziehbare Entscheidungsfindung birgt Gefahren für gesellschaftliche Grundwerte. Die „Blackbox Algorithmus" beschwört nicht nur das dumpfe Gefühl herauf, überwacht zu werden und die Selbstbestimmungsmacht darüber zu verlieren, wer welche persönlichen Daten sammeln, auslesen und daraus Schlüsse ziehen darf. Sie kann auch die Anwandlung auslösen, nach undurchsichtigen Entscheidungskriterien diskriminiert zu werden oder zum Objekt sublimer Steuerung zu degenerieren. Gleichzeitig entwaffnet der fehlende Einblick in das Arsenal einer Softwareanwendung den Verbraucher: Die Rechtmäßigkeit von Entscheidungen kann nur prüfen, wer die Datengrundlage, Handlungsabfolge und Gewichtung der Entscheidungskriterien kennt – und versteht.

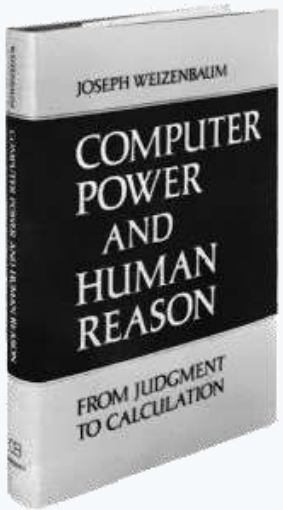
Algorithmen beruhen auf menschlichen Modellierungen, in die auch Ansichten, Neigungen und Wertmuster ihrer Schöpfer einfließen; sie sind nicht per se objektiv oder neutral. Ihre Wertungen folgen strukturell den Zielmustern des Entscheiders und damit typischerweise eher der ökonomischen Rationalität ihrer Schöpfer als Wertvorstellungen des Gemeinwesens. [...]

In vielen Kontexten sind Algorithmen den Schwankungen und Vorurteilen menschlicher Entscheidungsfindung zwar überlegen, diskriminierungsfrei sind sie aber nicht; ihnen fehlt ein ethischer Kompass. [...] Ihre Kontrolle entpuppt sich bei lernfähigen Systemen technisch gleichzeitig immer mehr als Quadratur des Kreises. [...] Wie sie zu ihren Ergebnissen gelangen, bleibt von außen nicht einsehbar und mithin auch für Kontrollmechanismen nicht nachvollziehbar. [...] Als Gegengift gegen die Intransparenz, die algorithmenbasierten Entscheidungsverfahren eigen ist, kann eine Begründungspflicht wirken. [...] Die Implementierung einer [...] Begründungspflicht in Softwareanwendungen wird Programmierer vor Herausforderungen stellen. Insbesondere bei komplexen maschinellen Lernverfahren neuronaler Netze können selbst ihre Schöpfer häufig nur im Nachhinein sagen, dass es zu einer Entscheidung gekommen ist – nicht aber, aus welchen Gründen. Was technisch schwierig erscheint, muss deshalb aber noch nicht im normativen Sinne unmöglich sein.

Blackbox Software?

Die mangelnde Durchschaubarkeit von Softwaresystemen ist allerdings kein neues Problem, das erst im Kontext von „Algorithmen" oder im Bereich der Künstlichen Intelligenz auftritt. Schon 1987 äusserte sich beispielsweise der Bremer Informatikprofessor Klaus Haefner (Jahrg. 1936) in einem vom Magazin „Der Spiegel" (10/1987) initiierten und veröffentlichten Streitgespräch mit dem kritischen Wissenschaftler Joseph Weizenbaum (1923-2008) dazu so: „Leider, leider werden immer neue Systeme gebaut, die so kompliziert sind, dass wir sie nicht mehr durchschauen können. [...] Dass der Mensch unberechenbare Seiten hat, damit konnten wir bislang leben. Ein für uns unberechenbar rechnender Computer, das ist gefährlich und widerspricht der Zielrichtung der Evolution." Weizenbaum selbst hatte Mitte der 1970er-Jahre in seinem Buch „Computer power and human reason" („Die Macht der Computer und die Ohnmacht der Vernunft") schon dringend vor unverständlichen Programmen und deren gesellschaftlichen Konsequenzen gewarnt.

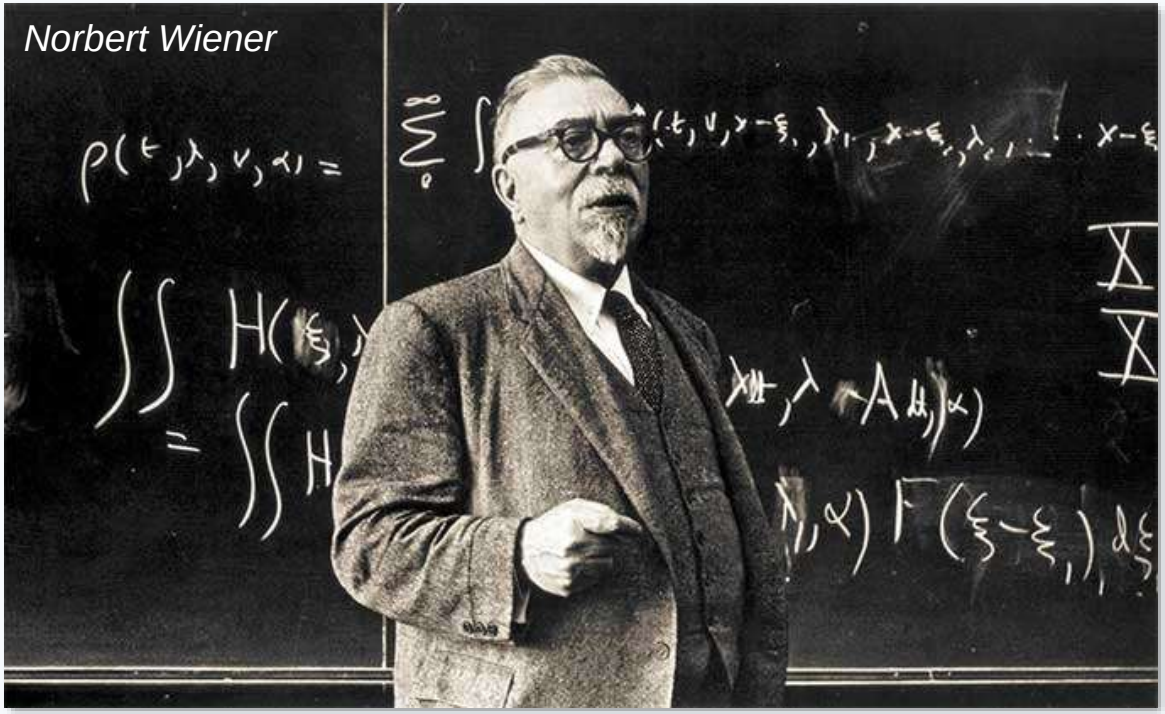
Klaus Haefner



Joseph Weizenbaum

Blackbox Maschine?

Noch früher, bereits 1960, hat Norbert Wiener (der Begründer der Kybernetik, von dem später noch ausführlich die Rede sein wird) in seinem Artikel „Some Moral and Technical Consequences of Automation" auf die praktische Nicht-Nachvollziehbarkeit von Computerberechnungen hingewiesen: „It may well be that in principle we cannot make any machine the elements of whose behavior we cannot comprehend sooner or later. This does not mean in any way that we shall be able to comprehend these elements in substantially less time than the time required for operation of the machine, or even within any given number of years or generations. […] This means that though machines are theoretically subject to human criticism, such criticism may be ineffective until long after it is relevant."


Norbert Wiener

Zum Aspekt von Algorithmen als Blackbox und den weiteren Kontext dazu sei auf den exzellenten Artikel von Kathrin Passig *Fünfzig Jahre Black Box* in Merkur, Nr. 823 (Dezember 2017) verwiesen.

Die Computertechnik ist eine Blackbox, auf der geschrieben steht: „Designed in California, made in China". Es sind, so könnte man glauben, junge Milliardäre aus Kalifornien und altgediente Parteisoldaten aus Peking – fremde Mächte –, die das Innenleben der Geräte gestalten, die zunehmend unser Leben bestimmen.

– Stefan Betschon, NZZ 16.7.2019

Feindbild Algorithmus

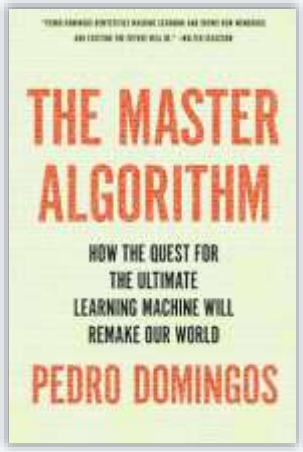
Man könnte einfach „Software“ sagen, aber das klingt eben nicht so sinister wie „Algorithmus“, sondern nur ein bisschen nach angestaubten Disketten. Aber was soll man machen, in der Umgangssprache verschwindet die Software zugunsten des Algorithmus, und wer gegen den Sprachwandel protestiert, der kann auch gleich gegen Ebbe und Flut demonstrieren. -- Kathrin Passig, 2017

Das Wort Algorithmus ist mittlerweile so negativ besetzt wie der Name Darth Vader. Algorithmen wird Macht zugesprochen, manchmal sogar „unheimliche Macht". Sie „greifen an" und werden zur mindestens potenziellen Bedrohung erklärt, zu Herrschern über Menschen und Schicksale, zu „Imperien". [Die Zeit, 14.10.2017, in „Feindbild Algorithmus“]

Im Februar 2019 wurden über zehntausend Personen in allen EU-Ländern nach dem Begriff „Algorithmus“ befragt. 15 Prozent haben den Begriff überhaupt noch nie gehört und 33 Prozent haben ihn zwar schon einmal vernommen, wissen aber nicht, was er bedeutet. Am ehesten verbinden sie damit auf die Person zugeschnittene Online-Werbung oder Anwendungen bei Dating-Plattformen. Polen kommt laut Studie auf den höchsten Kenntniswert, die Briten hingegen wissen besonders wenig über Algorithmen.

Algorithmus: Imageproblem

Was könnte man tun, um dem Wort „Algorithmus" ein besseres Image zu verpassen? Am einfachsten wäre es wohl, einfach ein anderes, freundlicheres Wort zu benutzen. Dies hat sich in der Geschichte immer wieder bewährt: Bei der Bundeswehr ist aus einem schrecklichen „Krieg" der an eine gemütliche, holzkohleofengewärmte Amtsstube erinnernde „Verteidigungsfall" geworden; die mittelalterliche, voraufklärerische „Folter" ist den zackigen „verschärften Verhörmethoden" gewichen...
Wie wäre es mit „Berechnungsvorschrift"? Zu dirigistisch. Vielleicht „Rechenanleitung"? Schon besser, klingt aber etwas nach einer Mathematiknachhilfestunde. Letzter Vorschlag: „Kochrezepte für Computer". Da stellt man sich unwillkürlich tomatensossebespritzte Roboter in einer Küche vor und ist damit schon mal in einer positiven mentalen Grundhaltung. Am Ende bleiben wir dann doch bei „Algorithmus". [Till Tantau]

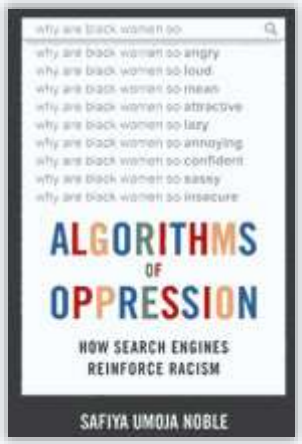


Algorithmen brauchen keine Computer

Algorithmen sind dafür zuständig, Probleme mit Hilfe einer speziellen Vorgehensweise zu lösen.
-- webhelm.de/algorithmen/

In seinem lesenswerten Buch „Planet der Algorithmen“ greift Sebastian Stiller die Popularisierung und postmoderne Interpretation des Begriffs „Algorithmus“ zunächst auf: „*Algorithmen haben einen schlechten Ruf. Man nennt sie in einem Atemzug mit Gleichmacherei, Grosskonzernen, Bespitzelung und Bedrohung.*“ Und an anderer Stelle schreibt er: „*Algorithmus ist ein Modewort. Es steht für »irgendetwas mit Computern«. Im Talkshow-Vokabular ersetzt es heute das, was in den 1980ern »Computerprogramm« hiess. Der Algorithmus dient als Leerstelle für alles, was man nicht so genau verstanden hat. Ähnlich umschwärmt ist wohl nur der traditionelle Liebling der Populärwissenschaft, die Formel. Der Algorithmus wirkt jedoch aktiver und bedrohlicher, deutet ein dunkles Geschehen an, das sich uns entzieht, weil wir nicht eingeweiht sind, nicht genug Fachwissen haben oder nicht so denken wie diejenigen, die mit Algorithmen arbeiten.*“

Bevor er selbst auf Algorithmen im Informatik-Sinne sowie deren Bedeutung eingeht, setzt er Algorithmen in den richtigen Bezug zum Computer: „*Algorithmen brauchen keine Computer. Der Mensch kennt Algorithmen spätestens, seit er rechnen kann. Ein Algorithmus ist ein Teil unseres Denkens, den wir so gut verstanden haben, dass wir ihn getrost auslagern können. Wir lassen denken. Dafür sind dann die Computer gut.*“



Logarithmus

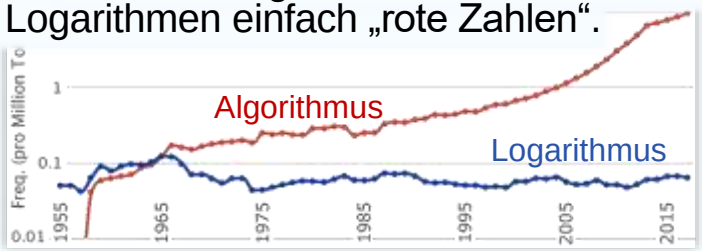
Süddeutsche Zeitung, 8. Dez. 2017

KORREKTUREN

»» In „Nonstop“ vom 5. Dezember auf Seite 18 war fälschlicherweise die Rede von Logarithmen, mit deren Hilfe automatisch Ticketpreise für Flugreise und Expresszüge ermittelt würden. Es handelt sich natürlich um Algorithmen.

Logarithmus ist ein Anagramm von *Algorithmus*.

Logarithmus, aus griech. *λόγος* (*lógos*) „Verhältnis, Berechnung“ und *ἀριθμός* (*arithmós*) „Zahl“; der neulateinische Begriff wurde erstmals 1614 vom schottischen Mathematiker John Napier verwendet. Die „Verhältniszahl“ drückt die Beziehung zwischen der arithmetischen und der geometrischen Reihe aus. Der Schweizer Jost Bürgi hatte zwar schon früher eine Logarithmentafel erstellt, aber erst 1620 veröffentlicht. Da darin die Logarithmen in roter Schrift ausgeführt sind, nannte er seine Logarithmen einfach „rote Zahlen“.



DWDS, halblogarithmisch

Kleines Feuilleton.

= [Die Dreihundertjahrfeier der Logarithmen.] In das Jahr 1914 fällt die 300. Wiederkehr des Tages, an dem Lord Napier seine „Mirifici Logarithmorum Canonis descriptio“ erscheinen ließ, das erste Druckwerk, in dem der Begriff des Logarithmus erörtert wird. Die „Royal Society“ in Edinburgh veranstaltet zur Erinnerung an dieses Ereignis am heutigen Tage einen Kongreß, zu dem eine Reihe von Fachleuten des In- und Auslandes eingeladen sind. John Napier, Laird of Merchiston, hat mit seinem Buche zweifellos die Grundlage zu allen mathematischen Verfahren gelegt, die eine Vereinfachung des Rechnens zum Ziele haben, und auf die Tabellen seines Werkes gehen die heutigen Logarithmentafeln zurück. Professor Max Simon (Straßburg) hat bereits in der „Frankfurter Zeitung“ (Zweites Morgenblatt vom 7. Juli 1913) eine ausführliche geschichtliche Darstellung der Entwicklung des logarithmischen Rechnens und den weittragenden Folgen für die mathematische und astronomische Wissenschaft gegeben. In diesen Tagen der Feiern zur Erinnerung an den großen englischen Mathematiker sollte man aber nicht vergessen, daß der Schweizer Joost Bürgi, der sich aus eigner Kraft vom Uhrmachergesellen zum Astronomen vom Range Tycho de Brahes erhoben, der vor Galilei den Proportionalzirkel erfunden hat, der gleichzeitig mit Stevin die Dezimalbruchrechnung und als erster die abgekürzte Multiplikation ersann, die Logarithmen mindestens fünf Jahre vor Napier gefunden hat. Seiner allzu großen Bescheidenheit wegen versäumte er jedoch, seine Erfindung rechtzeitig in die Oeffentlichkeit und damit ihre Priorität zur Anerkennung zu bringen.

Frankfurter Zeitung (Zweites Morgenblatt vom 24. Juli 1914)

Kernidee eines Algorithmus für den Logarithmus:

$$\ln x = 2\left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5} + \ldots\right)$$

2 4 3 1 Logarithmus

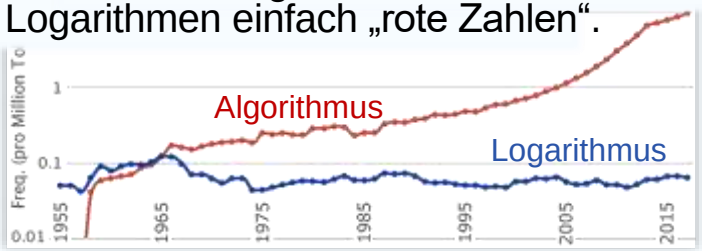
Süddeutsche Zeitung, 8. Dez. 2017

KORREKTUREN

»→ In „Nonstop“ vom 5. Dezember auf Seite 18 war fälschlicherweise die Rede von Logarithmen, mit deren Hilfe automatisch Ticketpreise für Flugreise und Expresszüge ermittelt würden. Es handelt sich natürlich um Algorithmen.

Logarithmus ist ein Anagramm von *Algorithmus*.

Logarithmus, aus griech. *λόγος* (*lógos*) „Verhältnis, Berechnung“ und *ἀριθμός* (*arithmós*) „Zahl“; der neulateinische Begriff wurde erstmals 1614 vom schottischen Mathematiker John Napier verwendet. Die „Verhältniszahl“ drückt die Beziehung zwischen der arithmetischen und der geometrischen Reihe aus. Der Schweizer Jost Bürgi hatte zwar schon früher eine Logarithmentafel erstellt, aber erst 1620 veröffentlicht. Da darin die Logarithmen in roter Schrift ausgeführt sind, nannte er seine Logarithmen einfach „rote Zahlen“.



DWDS, halblogarithmisch

Kleines Feuilleton.

= [Die Dreihundertjahrfeier der Logarithmen.] In das Jahr 1914 fällt die 300. Wiederkehr des Tages, an dem Lord *Napier* seine „Mirifici Logarithmorum Canonis descriptio“ erscheinen liess, das erste Druckwerk, in dem der Begriff des Logarithmus erörtert wird. Die „Royal Society“ in *Edinburgh* veranstaltet zur Erinnerung an dieses Ereignis am heutigen Tage einen *Kongress*, zu dem eine Reihe von Fachleuten des In- und Anstandes eingeladen sind. John Napier, Laird of Merchiston, hat mit seinem Buche zweifellos die Grundlage zu allen mathematischen Verfahren gelegt, die eine Vereinfachung des Rechnens zum Ziele haben, und auf die Tabellen seines Wertes gehen die heutigen Logarithmentafeln zurück. Professor Max *Simon* (Straßburg) hat bereits in der „Frankfurter Zeitung“ (Zweites Morgenblatt vom 7. Juli 1913) eine ausführliche geschichtliche Darstellung der Entwicklung des logarithmischen Rechnens und den weittragenden Folgen für die mathematische und astronomische Wissenschaft gegeben. In diesen Tagen der Feiern zur Erinnerung an den grossen englischen Mathematiker sollte man aber nicht vergessen, dass der Schweizer Joost *Bürgi*, der sich aus eigner Kraft vom Uhrmachergesellen zum Astronomen vom Range Tycho de Brahes erhoben, der vor Galilei den Proportionalzirkel erfunden hat, der gleichzeitig mit *Stevin* die Dezimalbruchrechnung und als erster die abgekürzte Multiplikation ersann, die Logarithmen mindestens *fünf Jahre vor Napier gefunden* hat. Seiner allzu grossen Bescheidenheit wegen versäumte er jedoch, seine Erfindung rechtzeitig in die Oeffentlichkeit und damit ihre Priorität zur Anerkennung zu bringen.

Frankfurter Zeitung (Zweites Morgenblatt vom 24. Juli 1914)

Kernidee eines Algorithmus für den Logarithmus:

$$\ln x = 2\left(\frac{x-1}{x+1} + \frac{(x-1)^3}{3(x+1)^3} + \frac{(x-1)^5}{5(x+1)^5} + \ldots\right)$$

Tous les vœux des mathématiciens: log, sin, tan

Tous les mathématiciens, tous les astronomes, et ils étaient alors en grand nombre, sentaient à chaque instant le besoin de trouver quelque invention qui simplifiât les effroyables calculs numériques auxquels ils étaient sans cesse contraints de se livrer pour la résolution des triangles célestes, seule application des mathématiques que l'on connût alors. [...] Et en effet, lorsque l'on songe à ce que devait être le calcul numérique des Tables de sinus et de tangentes naturels, pour un rayon exprimé par un million, ou même par dix millions de parties, comme on en construisait alors, quand on songe que tout cela exigeait de continuelles divisions et multiplications, qui devaient impitoyablement s'exécuter au complet, sans faire grâce d'aucun chiffre sur les plus grands nombres, on comprend très bien que tous les vœux des mathématiciens tendissent à se délivrer d'un si lourd fardeau. [Jean-Baptiste Biot]

Vorrede an den Treuherzigen Leser: Betrachtendt derowegen die Aigenschafft und Correspondenz der Progressen als der Arithmetischen mit der Geometrischen, das was in der ist Multiplizieren ist in jener nur Addieren, und was ist in der dividieren, in Jehner Subtrahieren, und was in der ist Radicem quadratam Extrahieren, in Jener ist nur halbieren, Radicem Cubicam Extrahieren, nur in 3 dividieren, Radicem Zensi in 4 dividieren, Sursolidam in 5. Und also fort in Anderen quantitaten, so habe Ich nichts Nützlichres erachtet, dan dise Tabulen. [Jost Bürgi: Kurzer Bericht der Progress Tabulen wie dieselbige nützlich in Allerley Rechnung zugebrauchen.]

Jost Bürgi (1552–1632) und die Logarithmen

Der im obigen Zeitungsartikel erwähnte „eigentliche Erfinder" der Logarithmen, Jost Bürgi – Uhrmacher, Instrumentenbauer, Astronom und Mathematiker – ist eine sehr interessante und oft unterschätzte Person. Bürgi stammt aus Lichtensteig im Toggenburg, erhielt nur eine einfache Schulausbildung und konnte daher kein Latein und schrieb schlecht Deutsch – ein grosses Handicap in seiner späteren wissenschaftlichen Karriere. Das Uhrmacherhandwerk lernte er als fahrender Geselle an verschiedenen Orten; seine mathematischen Kenntnisse erwarb er wohl zu einem guten Teil autodidaktisch.

Kupferstich (Ausschnitt) des niederländischen Künstlers Egidius Sadeler, ein Freund Bürgis

1579, im Alter von 27 Jahren, wurde Bürgi als Hofuhrmacher und Astronom bei Landgraf Wilhelm IV. von Hessen-Kassel (genannt „der Weise") angestellt. 1585 baute er die erste Uhr mit Sekundenzeiger – zu einer Zeit, in der noch kaum jemand eine Vorstellung von einer Sekunde hatte. Nun konnte man die Zeit buchstäblich vergehen sehen! Das Uhrwerk variiert in 24 Stunden um

höchstens eine Minute – bisher ist man bei den besten Uhren eine Abweichung von einer Viertelstunde gewohnt. Mit Bürgi wird die Uhr daher zum wissenschaftlichen Messinstrument, insbesondere für die Astronomie. Dank seiner Präzisionsinstrumente konnte Bürgi die Durchgangszeit und Distanz von Fixsternen, Planeten und anderen Himmelskörpern weit genauer bestimmen, als dies zuvor möglich gewesen war. Bürgi arbeitet mit den berühmten Astronomen Tycho Brahe und Johannes Kepler zusammen, die von seinen präzisen Instrumenten und seinen Mathematikkenntnissen und praktischen Rechenmethoden stark profitierten. Berühmt über die engere Fachwelt hinaus wurde er vor allem durch seine Himmelsgloben. Auf seiner Globus-Uhr von 1594 kann man die aktuellen Positionen von über 1000 Sternen ablesen – dabei hat die Kugel mit kompliziertem mechanischem Innenleben nur 14.2 cm Durchmesser.

Bürgi veröffentlichte seine Logarithmentafel 1620, nachdem John Napier bereits 1614 ein Buch über Logarithmen veröffentlicht hatte. Entstanden ist Bürgis Tafel allerdings schon Jahre früher, und offensichtlich auch von ihm und Vertrauten wie Johannes Kepler praktisch genutzt worden. 1620 tobte allerdings bereits der Dreissigjährige Krieg, die Druckauflage war klein, vielleicht handelte es sich bei den wenigen erhaltenen Exemplaren auch erst nur um Probedrucke. Denn die im Titel angekündigte Nutzungsanleitung („Unterricht") ist von Bürgi zwar verfasst, aber offenbar nie gedruckt worden. Und ohne eine solche Anleitung ist die Ta-

Aritmetische und geometrische Progress-Tabulen, sambt gründlichem unterricht, wie solche nützlich in allerley Rechnungen zugebrauchen und verstanden werden sol. Gedruckt, in der Alten Stadt Prag, bey Paul Sessen, der Löblichen Universitet Buchdruckern, Im Jahr, 1620.

belle höchstens für Mathematiker, die mit dem Prinzip bereits vertraut waren, nützlich. Auch insofern ging der Ruhm bezüglich der Logarithmen an John Napier, Bürgi geriet später fast in Vergessenheit.

L'invention des logarithmes, en réduisant le temps passé aux calculs de quelques mois à quelques jours, double pour ainsi dire la vie des astronomes. -- Pierre-Simon Laplace

Unabhängig vom Erfinder waren seinerzeit allerdings die Logarithmen sowie genaue (und möglichst fehlerfreie) Logarithmentafeln entscheidend für Fortschritte in der Astronomie. Denn die Bestimmung von Sternpositionen, Planetenbahnen und (für die Nautik wichtige) Ephemeriden beruht auf der sphärischen Trigonometrie. Deren praktische Anwendung erforderte aber langwierige Berechnungen, darunter insbesondere viele Multiplikationen mit langen Zahlen. Bei Multiplikationen entsteht ein Aufwand, der quadratisch mit der Zahl der Ziffern wächst, im Unterschied zum nur linearen Aufwand bei der Addition. Multiplikationen dauern daher lange, vor allem aber stellen sie aufgrund der hohen Zahl von Elementaroperationen auch eine bedeutende Quelle von Rechenfehlern dar. Dadurch, dass ab dem 16. Jahrhundert der Schiffsverkehr auf hoher See zunahm und die Nachfrage nach genauen nautischen Tafeln antrieb, aber auch dadurch, dass präzisere astronomische Instrumente mehr Messwerte mit mehr signifikanten Ziffern lieferten, wurde das astronomische Rechenproblem drängend. Eine Methode wie die Logarithmen, die es gestattet, Multiplikationen auf Additionen zurückzuführen (und es auch ermöglicht, das Wurzelziehen und Potenzieren wesentlich zu erleichtern), stellte daher eine aus heutiger Sicht kaum zu überschätzende Erleichterung für die menschlichen Rechner dar.

Nun war es seinerzeit unökonomisch und praktisch kaum durchführbar, den Logarithmus (oder Anti-Logarithmus) als Funktionswert erst bei Bedarf (und jedes Mal neu) zu berechnen. Gleiches galt auch für die schon wesentlich länger verwendeten trigonometrischen Funktionen (wie beispielsweise die Sinusfunktion). Indem die Werte „ein für alle Mal" berechnet und tabelliert wurden, konnten aktuelle Rechnungen durch Nutzung dieses wertvollen konservierten Wissens wesentlich beschleunigt werden. Rechentafeln hat es daher bereits früh in der Geschichte

gegeben, und die Tabulierung der Logarithmen war ein logischer Schritt, der sich auf Grund des wissenschaftlich-technischen Fortschrittes im 17. Jahrhundert ergab, um Astronomen, Landvermessern, Navigatoren etc. als Rechenhilfsmittel zur Verfügung zu stehen. Logarithmentafeln und Tafeln für trigonometrische und ähnliche Funktionen waren dann bis in die 1970er-Jahre in Gebrauch. Erst mit dem Aufkommen von elektronischen Taschenrechnern (sowie von Software-Bibliotheken – „libraries" –, die Unterprogramme zur Berechnung der Funktionen auf Computern enthielten) war es möglich, diese Funktionswerte „just in time" zu berechnen. An der expliziten Verwendung von Logarithmen als Hilfsmittel zur Abkürzung und Beschleunigung von Rechnungen bestand auch kein Bedarf mehr, so dass die Tafeln, ihr Gebrauch, aber auch ihre ehemalige grosse Bedeutung praktisch in Vergessenheit geriet.

Bürgis Tafel umfasst 58 Seiten und enthält 23023 Korrespondenzen zwischen roten Zahlen, einer arithmetische Folge, und schwarzen Zahlen, einer geometrische Folge – daher auch die Bezeichnung „Progress-Tabulen". Die roten Zahlen stellen damit die Logarithmen der entsprechenden schwarzen Zahlen dar; den Dezimalpunkt muss man sich aber jeweils geeignet hinzudenken. Bürgi schien, im Unterschied zu Napier, das mathematische Konzept einer Logarithmusfunktion samt Basis noch gar nicht wirklich realisiert zu haben; man kann aber leicht rekonstruieren, dass die (implizite) Basis seiner Logarithmen 2.71814593 ist; die Ziffernfolge der Zahl taucht im Titelbildausschnitt im schwarzen Ring auf. Diese Basis ist fast e (2.71828183), Bürgi wählte wohl aus rechenpraktischen Gründen für die Erstellung seiner Tabelle einen leicht anderen Wert – für den Zweck und die praktische Anwendung der Tabellen

als Rechenhilfsmittel ist der Wert der Basis sowieso unerheblich. (Setzt man bei der im Zentrum des Titelblatts aufgeführten „ganzen Schwarzen Zahl" den Dezimalpunkt so, dass 10.0 entsteht, und berechnet man „naiv" den natürlichen Logarithmus davon, dann erhält man mit 2.30258509 fast die Ziffernfolge der „ganzen Roten Zahl" – der Unterschied bei den hinteren Dezimalstellen ist dabei seiner leicht von e abweichenden Basis geschuldet.)

In den folgenden Jahrzehnten und Jahrhunderten wurden Tafeln für Logarithmen und trigonometrische Funktionen immer wieder neue berechnet – meist mit mehr signifikanten Stellen und um Rechen- und Druckfehler früherer Tabellen zu korrigieren. Interessant sind in dieser Hinsicht die 1911 von Julius Bauschinger, Direktor der Kaiserlichen Sternwarte in Strassburg, und Jean Peters, Observator des Königlich-Astronomischen Recheninstituts in Berlin, herausgegeben „Logarithmisch-Trigonometrische Tafeln mit 8 Dezimalstellen enthaltend die Logarithmen aller Zahlen von 1 bis 200000 und die Logarithmen der trigonometrischen Funktionen für jede Sexagesimalsekunde des Quadranten". Das Besondere dieser Tafeln war, dass sie nach der Differenzenmethode mit einer speziell dafür gebauten mechanischen Rechenmaschine berechnet und auch gleich ausgedruckt wurden – dies sollte viele potentiellen Fehlerquellen vermeiden.

Wie kam nun aber Bürgi auf das Prinzip der Vereinfachung von Operationen wie der Multiplikation durch Addition mittels Logarithmen? Man kann vermuten, dass ihm als Motivation dafür eine andere Methode diente, die Anfang des 16. Jahrhunderts unter dem Begriff „Prosthaphärese" (aus dem Griechischen πρόσθεσις für Addition und ἀφαίρεσις für Subtraktion) im Kreise der rechnenden Astronomen, welche naturgemäss Experten auf dem Gebiet der sphärische Trigonometrie waren, bekannt wurde. Es basiert auf trigonometrischen Identitäten, die in heutiger Formelschreibweise etwa so notiert werden können:

- $\cos(\alpha) \times \cos(\beta) = \frac{1}{2} [\cos(\alpha + \beta) + \cos(\alpha - \beta)]$
- $\sin(\alpha) \times \sin(\beta) = \frac{1}{2} [\sin(90° - \alpha + \beta) - \sin(90° - \alpha - \beta)]$

Für die Zwecke der Astronomie und Nautik, die an sich viel Sinus und Kosinus zu multiplizieren haben, eine gar nicht üble Methode; und doch umständlich. -- Otto Toeplitz

Dies bedeutet, dass man ein Produkt mittels Nachschlagen in Sinus- oder Kosinustafeln und einfachen Operationen (Addition, Subtraktion, Halbierung) berechnen konnte! In seiner Schrift Fundamentum Astronomiae schreibt Bürgi in etwas eigenwilliger Orthographie dazu:

„Multipliciren vnd Diuidiern [...] vf eine viell leichttere vnnd behendere artt durch die prosthaphaeresin, Nemlich durch addiern vnd Subtrahiern durch hilff der Sinuum wie volgtt […]".

Bei der Anwendung benötigt man zusätzlich zum Tabellennachschlagen mindestens vier (einfache) arithmetische Operationen. Dies noch weiter zu vereinfachen, war sicherlich ein Ziel von Bürgi. Mit seinen „Progress-Tabulen", also der Logarithmentafel, schaffte er es dann tatsächlich, dies auf eine einzige solche Operation zu reduzieren! (Schon 1544 wies Micheal Stifel allerdings darauf hin, dass Multiplikationsoperationen in einer geometrischen Reihe durch Additionen in einer zugeordneten arithmetischen Reihe abgebildet werden können.)

Nun beruht die Rechengenauigkeit bei Anwendung der Prosthaphärese stark auf der Genauigkeit der verwendeten Sinustafel. Ferner spielen die Winkelfunktionen natürlich ganz generell eine fundamentale Rolle in der sphärischen Trigonometrie und damit der Astronomie. (Wegen $\cos(x) = \sin(x + 90°)$ und $\tan(x) = \sin(x) / \cos(x)$ sowie $\sec(x) = 1/\cos(x)$ genügt im Prinzip die Tabellierung des Sinus, um daraus die Werte der anderen Winkelfunktionen zu gewinnen.) Bevor Bürgi sich den Logarithmen zuwandte, galt sein Bemühen daher der effizienten Berechnung einer Sinustafel, dem „Canon sinuum".

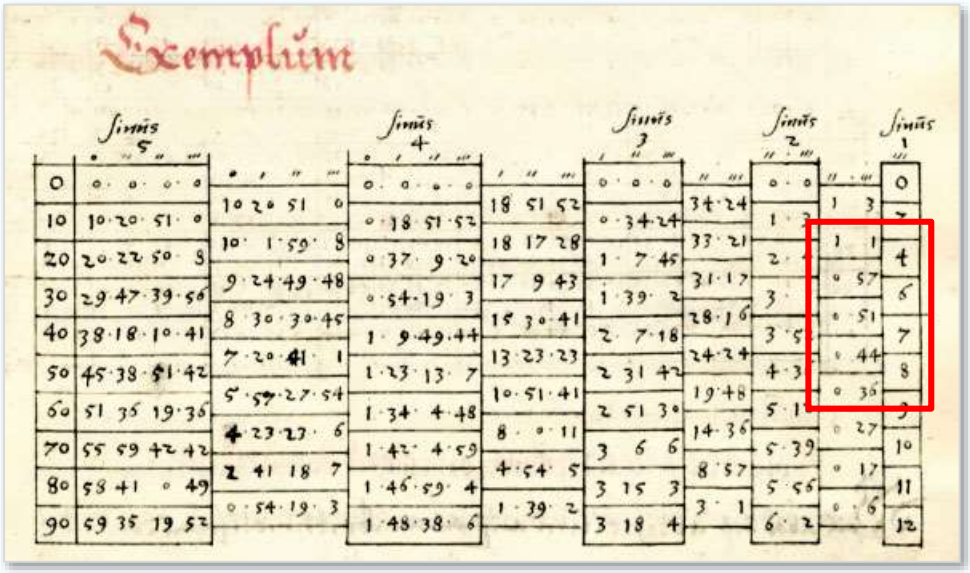
Zur Berechnung von Sinuswerten wurde jahrhundertelang ein geometrisches Verfahren verwendet, das der griechische Mathematiker und Philosoph Ptolemäus beschrieb (aber schon vor ihm verwendet wurde und um 820 von Muhammed al-Chwarizmi in Babylon tradiert wurde). Die Methode nutzt ineinander geschachtelte Polynome, die zugehörigen Berechnungen waren allerdings langwierig und mühsam.

Bürgi fand nun einen innovativen und schnell konvergierenden Algorithmus, der einfach anzuwenden ist und nur die Grundoperationen Addition und Halbierung verwendet. Dazu wählt er statt des klassischen geometrischen Prinzips einen effizienter anzuwendenden arithmetisch-

algebraischen Ansatz, der auf Differenzenfolgen beruht. (Bürgi selbst schreibt, dass „diß artificium auß einem Arithmetischen vnd nicht auß einem Geometrischen grunde herfleußet".) Ohne komplizierte Divisionen und Interpolationen ermöglicht Bürgis Methode eine auf beliebig viele Stellen genaue Bestimmung der Sinuswerte. Zu einer Zeit, da Divisionen sehr zeitraubend und fehleranfällig waren, bedeutete diese Vereinfachung einen enormen Fortschritt. Bürgis damit erstellter „Canon sinuum" führt in Schritten von jeweils zwei Bogensekunden 81000 Sinuswerte zwischen 0 und 45 Grad auf. Früher hätte eine solche Arbeit ein Jahrzehnt gedauert.

Bürgi hält seinen Algorithmus, von ihm als „Kunstweg" bezeichnet, allerdings geheim. Nur in seinem frühneuhochdeutschen Manuskript „Fundamentum Astronomiae", das er 1592 in Prag Kaiser Rudolph II übergibt, beschreibt er ihn und charakterisiert den Fortschritt mit einem gewissen Stolz – im Vergleich zur unsicheren, fehleranfälligen, mühsamen und arbeitsreichen klassischen Methode sei sein Algorithmus besser, sicherer, leichter und auch lustiger: „welchs Mathematisch Kunststuck [...] von vnnß ist excogitiert vnd außgesinnett wordenn. [...] Vnnd wirdt also auff solche weiße mitt beschwerlicher muhe vnd arbeitt der ganze Canon vermachett, mitt welcherer weiß sich die alttenn biß auff dieße vnsere Zeitt In soviell hundertt Jahrenn beholffenn. Weil sie es nicht beßer habenn erfindenn konnen, welchere weiß aber eben so vngewiß vnnd bauefellig, also muheselig vnd arbeitsam, ist. Wollen derhalbenn die sache auf eine andere beßere vnd richtiger, auch sowoll leichter vnd lustiger, weis numehr fur die handt nehmen."

Ohne den Algorithmus zur Ermittlung der Sinustafel hier vollständig beschreiben zu wollen (siehe dazu weiter unten die Literaturangaben), seien nachfolgend einige kürzere Passagen aus Bürgis Beschreibung aufgeführt. Im Wesentlichen baut man schematisch eine Zahlentabelle auf. (Mit einem Tabellenkalkulationsprogramm wie Excel lässt sich heutzutage eine entsprechend „selbstausfüllende Tabelle" leicht programmieren.) Erstaunlich erscheint schon die Initialisierung des Algorithmus: „Anfenglich setze etzliche Zallen deines gefallens vber einander" – man beginnt also mit willkürlichen (nicht-negativen) „Zufallszahlen", die in eine Spalte gesetzt werden. Sodann erzeugt man iterativ neue halbversetzte Spalten, im Wesent-



Exemplum

...
36 + 8 → 44
44 + 7 → 51
51 + 6 → 57
57 + 4 → 61
61 sexagesimal → 1 1
...

lichen durch Addieren von zwei Zahlen der Spalte rechts davon. („Umgekehrt" gelesen ergeben sich auf diese Weise dann Differenzen, die die mathematische Rechtfertigung des Verfahrens induzieren!) „Vnd je lenger vnnd mehr du" rechnest, desto „scherffer vnnd gewißer daraus die verhaltung" der Ergebniswerte, „biß sie entlich [...] schier vnuerenderlich pleiben. Welchs den eine eigentliche vnnd gewiße anzeigung wirdt geben", dass die gewünschte Approximation der Ergebnisses erreicht wurde.

Die Idee hinter dem erstaunlichen Verfahren (das erst vor wenigen Jahren enträtselt werden konnte, als das verschollene Manuskript nach 427 Jahren wiederentdeckt wurde) kann man als Umkehrung der Berechnungsmethode der zweiten Differenzenfolge begreifen. Bei der Sinusfunktion sind diese zweiten Differenzen ein konstantes (negatives) Vielfaches des Funktionswertes an derselben Stelle (für die zweite Ableitung gilt sin''(x) = -sin(x)). Da die Differenzbildung von nahezu identischen Werten zu Auslöschungen und numerischen Instabilitäten führt, konnte Bürgi hoffen, dass der umgekehrte Weg stabil und konvergent ist. Ob ihm dies aber explizit bewusst war?

Lesenswert im Zusammenhang mit dem Algorithmus zur Konstruktion der Sinustafel ist der NZZ-Artikel „Ein «Kunstweg» zur Berechnung von Sinuswerten“ vom 26.1.2016 (online: „Jost Bürgi oder die Kunst der Trigonometrie“, 29.1.2016, www.nzz.ch/wissenschaft/astronomie/ein-kunstweg-zur-berechnung-von-sinuswerten-1.18685618) oder auch der wissenschaftliche Aufsatz „Jost Bürgi's method for calculating sines“ von M. Folkerts, D. Launert, A. Thom [Historia Mathematica 43(2), 2016, 133-147] und „How Bürgi computed the sines of all integer angles simultaneously in 1586“ von Grégoire Nicollier [Math. Semesterber. 65(1), 2018, 15-34].

> Und weil mir auß mangel der sprachen die thür zu den authoribus nit alzeitt offen gestanden, wie andern, hab jch etwas mehr, als etwa die glehrte vnd belesene meinen eigenen gedanckhen nachhengen vnd newe wege suechen müessen. -- Jost Bürgi

Der Pionier der Astronomie in der Schweiz, Rudolf Wolf, urteilte in einem Vortrag am 4. Januar 1872 im Rathaus in Zürich über Bürgi so: „Sein mathematisches Talent endlich liess Bürgi zu Gunsten der mühsamen Berechnungen, welche die damaligen Beobachtungsmethoden erforderten, in denkwürdiger Weise leuchten, und für den praktischen Rechner der Jetztzeit, der sich bequemer Rechnungsvorschriften, Hülfstafeln und dergleichen erfreut, gehört unser Toggenburger entschieden zu den Heiligen: Nicht nur verdankt man ihm nämlich das immense Werk einer nach ihm eigenthümlichen Methode berechneten Tafel, welche für jede zweite Sekunde den Sinus auf acht Stellen gab und nach Kepplers Urtheil damals alle ähnlichen Tafeln an Genauigkeit weit überragte, — sondern namentlich auch die Erfindung, oder zum allerwenigsten die Miterfindung, der Logarithmen.“

Rudolf Wolf (1816 – 1893) war Professor für Astronomie an der ETH und von 1864 bis 1893 Direktor der von ihm gegründeten Eidgenössischen Sternwarte (Semper-Sternwarte der ETH in der Schmelzbergstrasse). Er amtierte zudem als erster Direktor der Meteorologischen Zentralanstalt der Schweiz.

Zum Wort „Algorithmus“

Algorithmen im Sinne von Rechenverfahren gab es allerdings schon vor al-Chwarizmi (z.B. euklidischer Algorithmus; Euklid lebte im 3. Jhd. v. Chr.); al-Chwarizmi ist Namenspatron, nicht aber Erfinder des Algorithmen-Begriffs.

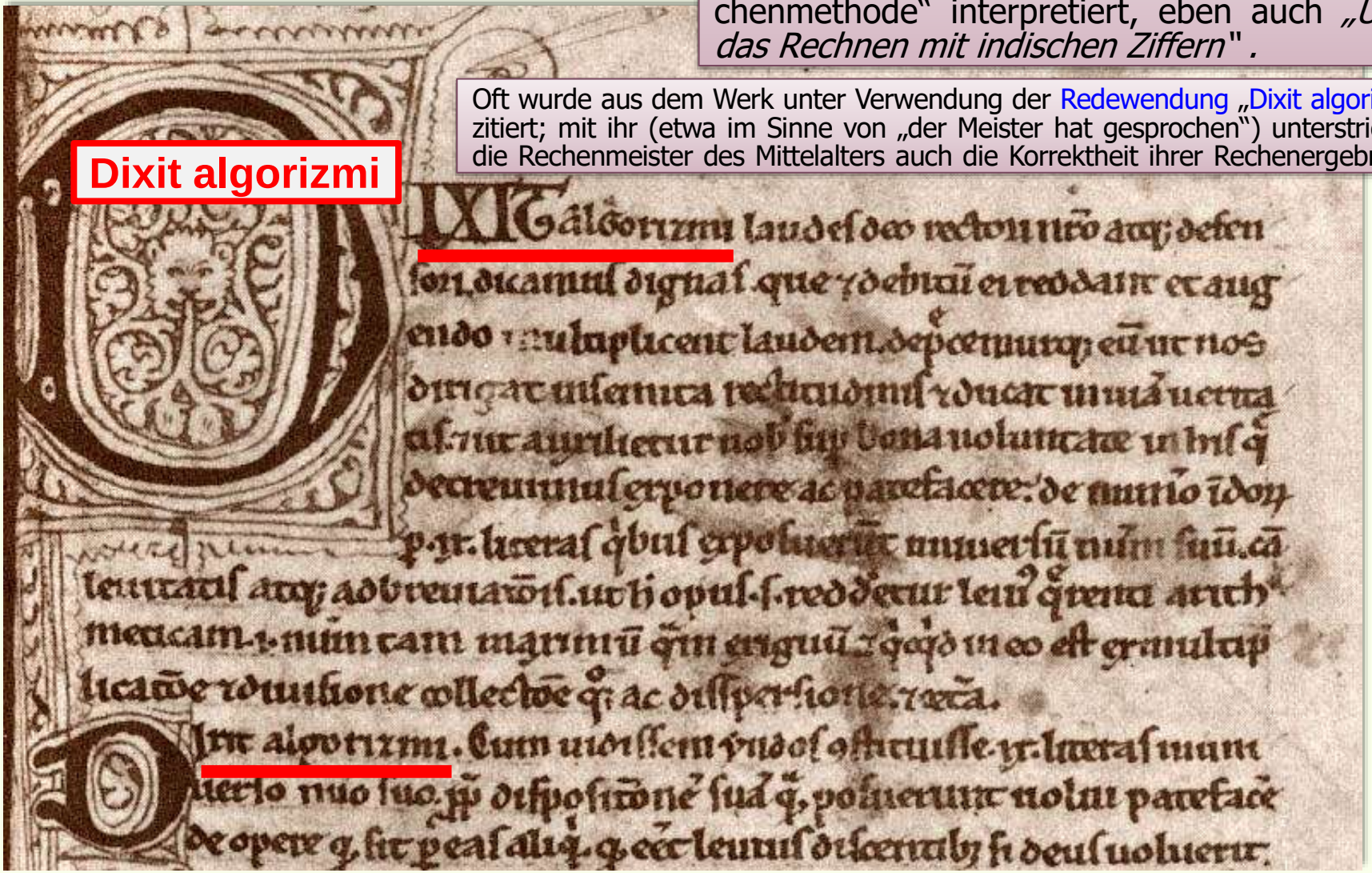
- Muhammed al-Chwarizmi (oder Al-Hwarizmī), ca. 780–850, persisch-arabischer Mathematiker, Astronom und Geograph, lebte in Bagdad.
- Beiname al-Chwarizmi → Herkunft aus Choresmien bzw. Xorazm: Landschaft südlich des Aralsees (heute zu Usbekistan und teilweise Turkmenistan gehörend)
- Sein auf Arabisch verfasstes Buch *Über das Rechnen mit indischen Ziffern* erläutert das Rechnen mit Dezimalzahlen, gibt Rechenregeln an und führt die Ziffer 0 aus dem indischen ins arabische Zahlensystem ein.

- In lateinischer Übersetzung des Werkes wird der Autor al-Choarismi oder algorizmi bzw. latinisiert Algorismus genannt; daraus entstand schliesslich die Bezeichnung „Algorithmus“, zunächst aber für die Kunst des Rechnens mit Dezimalzahlen („arabischen Ziffern“). In Webster’s New World Dictionary wurde letzteres bis 1957 mit „algorism“ bezeichnet.
- Aus dem Wort „al-dschabr“ im Titel eines seiner anderen Bücher, *Al-kitāb al-muchtasar fi hisab al-dschabr wa-l-muqabala* („Das kurzgefasste Buch vom Rechen durch Ergänzung und Ausgleich“, الكتاب المختصر في حساب الجبر والمقابلة), entstand „Algebra“ als Begriff. („Ergänzung“ meint, negative Werte auf die andere Seite einer Gleichung zu bringen, „Ausgleich“ das Eliminieren gleicher Grössen auf beiden Seiten).

Dixit Algorizmi – Also sprach Algorizmi

Lateinische Übersetzung und Bearbeitung aus dem Mittelalter des (im Original verschollenen) Buches *Algoritmi de numero Indorum* („*al-Chwarizmi über die indischen Zahlen*" oder, wenn man das „i" am Wortende als Pluralform eines Wortes „Algoritmus" mit der Bedeutung „Rechenmethode" interpretiert, eben auch „*Über das Rechnen mit indischen Ziffern*" .

Oft wurde aus dem Werk unter Verwendung der Redewendung „Dixit algorizmi" zitiert; mit ihr (etwa im Sinne von „der Meister hat gesprochen") unterstrichen die Rechenmeister des Mittelalters auch die Korrektheit ihrer Rechenergebnisse



Transliteration und Übersetzung ins Deutsche

Dixit Algorizmi: laudes Deo rectori nostro atque defensori dicamus dignas, que et debitum ei reddant, et augendo multiplicent laudem. Deprecemurque eum ut nos dirigat in semita rectitudinis et ducat in viam veritatis, et ut auxilietur nobis super bona voluntate in his que decrevimus exponere ac patefacere de numero Indorum per .ix. literas quibus exposuerunt universum numerum suum, causa levitatis atque a[d]breviationis, ut hoc opus s[cilicet] redderetur levius querenti arithmeticam, id est numerum tam maximum quam exiguum et quicquid in eo est ex multiplicatione et divisione, collectione quoque ac dis[s]persione et cetera.

Dixit Algorizmi: Cum vidissem Yndos constituisse .ix. literas in universo numero suo propter dispositionem suam quam posuerunt, volui patefacere de opere quod fit per eas aliquid quod esset levius discentibus, si Deus voluerit.

Algorizmi sprach: Wir wollen Gott, unserem Herrn und Beistand, das ihm zukommende Lob aussprechen, das ihm das Geschuldete abstattet und durch Vermehren sein Lob vervielfältigt. Und wir wollen ihn bitten, dass er uns auf den Pfad der Geradlinigkeit und auf den Weg der Wahrheit führt und dass er uns hilft bei unserer guten Absicht hinsichtlich dessen, was wir beschlossen haben darzulegen und zu erörtern über die Rechenweise der Inder mit Hilfe von 9 Symbolen, mit denen sie jede einzelne Zahl um der Leichtigkeit und abgekürzten Form willen darstellen, damit nämlich dieses Verfahren leichter wird für denjenigen, der sich um die Arithmetik bemüht, d.h. sowohl um eine grosse als auch eine sehr kleine Zahl und um all das, was mit ihr geschieht an Multiplikation und Division, Addition und Zerlegung, und um die übrigen Dinge.

Algorizmi sprach: Als ich sah, dass die Inder 9 Symbole für jede ihrer Zahlen aufgestellt hatten, um sie nach ihrem System darzustellen, da wollte ich von dem Verfahren, das mit jenen [Symbolen] geschieht, etwas offenkundig machen, was leichter für die Lernenden sein würde, so Gott will.

Al-Chwarizmi ➔ „Algorithmus“

Oft wird statt „Dixit Algorizmi“ auch „Dixit Algoritmi“ oder „Dixit Algorismi“ geschrieben. Zur Namensschreibung merkt der Orientalist und Wissenschaftshistoriker Julius Ruska (1867 – 1949), Professor in Heidelberg und Direktor des Berliner Forschungsinstituts für Geschichte der Naturwissenschaften (in „Zur ältesten arabischen Algebra und Rechenkunst. Sitzungsberichte der Heidelberger Akademie der Wissenschaften, Phil.-hist. Klasse, Jahrg. 1917, Carl Winter’s Universitätsbuchhandlung, 1917“), folgendes an:

„Man kann eine ganze Musterkarte von Umschreibungen dieses Namens zusammenstellen; so schreibt COLEBROOKE (1817) Khuwárezmí, ROSEN (1831) of Khowarezm, WOEPCKE (1863) Alkhârizmî, MARRE (1865) Al Khârezm, RODET (1878) Al-Khârizmi, HANKEL (1874) al Ḫovārezmī, CANTOR (1880) Alchwarizmî, VAN VLOTEN (1895) Al-Khowarezmi, SUTER (1900) el-Chowâresmî oder Chwâresmi, BJÖRNBO (1905) Alkwarizmi, WIEDEMANN (1906) al Châriẓmî, BOSMANS (1906) El-Chowārizmī, KARPINSKI (1910) Al-Khowarizmi, SMITH (1911) Al-Khowārazmī, ENESTRÖM Alkhwarizmi und Alkwarismi. Dazu kommen die alten Formen Alchoarismi in der Übersetzung des Gerhard von Cremona, Alghuarizim, Alguarizin, Algaurizm, Algaurizin in den Handschriften der Übersetzung des Robert Castrensis (Bibl. Math. 3. F., Bd. 11, 1910/11, S. 130; die Formen mit *in* und *im* sind offenbar aus den Formen mit *m* und *mi* entstanden, die mit *au* aus *ua*), Alghoarismi, Algoarismi, Algorismi in der Pratica Arismetrice des Johannes Hispalensis, Algoritmi im Traktat de numero Indorum, Volkstümliche Weiterbildung führt von Algorismus zu *augorisme* und *augrim* (SMITH-KARPINSKI, The hindu-arabic numerals, Boston 1911, S. 120, 121).
Alle Formen mit Khwa- und Chwa- beruhen auf buchstäblicher Umschreibung der Zeichengruppe خوا, die der persischen Sprache eigentümlich ist und etwa *khō* gesprochen wird (vgl. Χωρασμία, Χοράσμιοι). Da der zweite Vokal von خوارزم Ḫwārazm nach VULLERS (S. 736) ein Fatḥa ist, lautet die genaue Umschreibung der Nisbe al-Ḫwārazmī oder Alḫwārazmī. Die Wiedergabe von خ durch k, von ز durch s ist falsch, die Umschreibung von خ durch ch oder kh ist nicht eindeutig genug, der allgemeinen Verwendung von Ḫ und ḫ stehen typographische Schwierigkeiten gegenüber. [...] Will man die geschichtlichen Zusammenhänge betonen, so wird man besser Algoritmi oder Algorithmus schreiben.“

Al-Chwarizmi

Julius Ruska gibt in seiner Abhandlung noch einige Einblicke in das Leben von Al-Chwarizmi, soweit dies überhaupt bekannt ist bzw. sich aus seinen Schriften und dem Zeitkontext ermitteln lässt:

„Daß [al-Chwarizmi] eine wissenschaftliche Persönlichkeit von starker Eigenart war, beweisen seine wirklich für „Fachleute“ verfaßten astronomischen und geographischen Werke, soweit sie uns ein glücklicher Zufall erhalten hat. [...] ...einer Erdkarte, und zwar höchst wahrscheinlich zu dem großen Kartenwerk, das [der Kalif] Alma’mun durch eine Gesellschaft von Gelehrten herstellen ließ, die sich wohl aus allen Provinzen des Kalifats zusammenfanden. Es scheint die besondere Aufgabe des Astronomen [al-Chwarizmi] gewesen zu sein, den Inhalt des Kartenwerks in ähnlicher Weise in Buchform zu bringen, wie dies vorher Ptolemaeus für griechische Karten [...] getan hatte. [...]

http://archive.computerhistory.org/resources/still-image/Khuwarizmi-Muhammad_ibn_Musa/statue_Khuwarizmi.102649723.lg.jpg

Al-Chwarizmi: Denkmal in Xiva (Usbekistan)

Al-Chwarizmi (2)

So gewinnt das Bild, das wir von [seinem] wissenschaftlichem Lebenswerk erhalten, mehr und mehr an Inhalt. Wir sehen unsern Autor inmitten einer hochstrebenden, von allen Seiten wissenschaftliche Anregung suchenden und empfangenden Gelehrtenwelt, als Mitglied einer Akademie, die in der von Harun al Rasid begründeten, von Alma'mun freigebig geförderten Bibliothek zu Bagdad ihren geistigen Mittelpunkt hatte. In erster Linie Astronom und Astrolog, stützte sich der einem altpersischen Geschlecht entstammende Gelehrte zunächst wohl auf persische und indische Überlieferung. Indische Werke über die mathematischen Hilfswissenschaften und persönlicher Verkehr mit indischen Astronomen mögen ihm die Anregung zur Abfassung der beiden Schriften über die indische Rechenkunst und über die wichtigsten Kapitel des angewandten Rechnens gegeben haben."

http://upload.wikimedia.org/wikipedia/commons/a/a6/Khwarizmi_Amirkabir_University_of_Technology.png

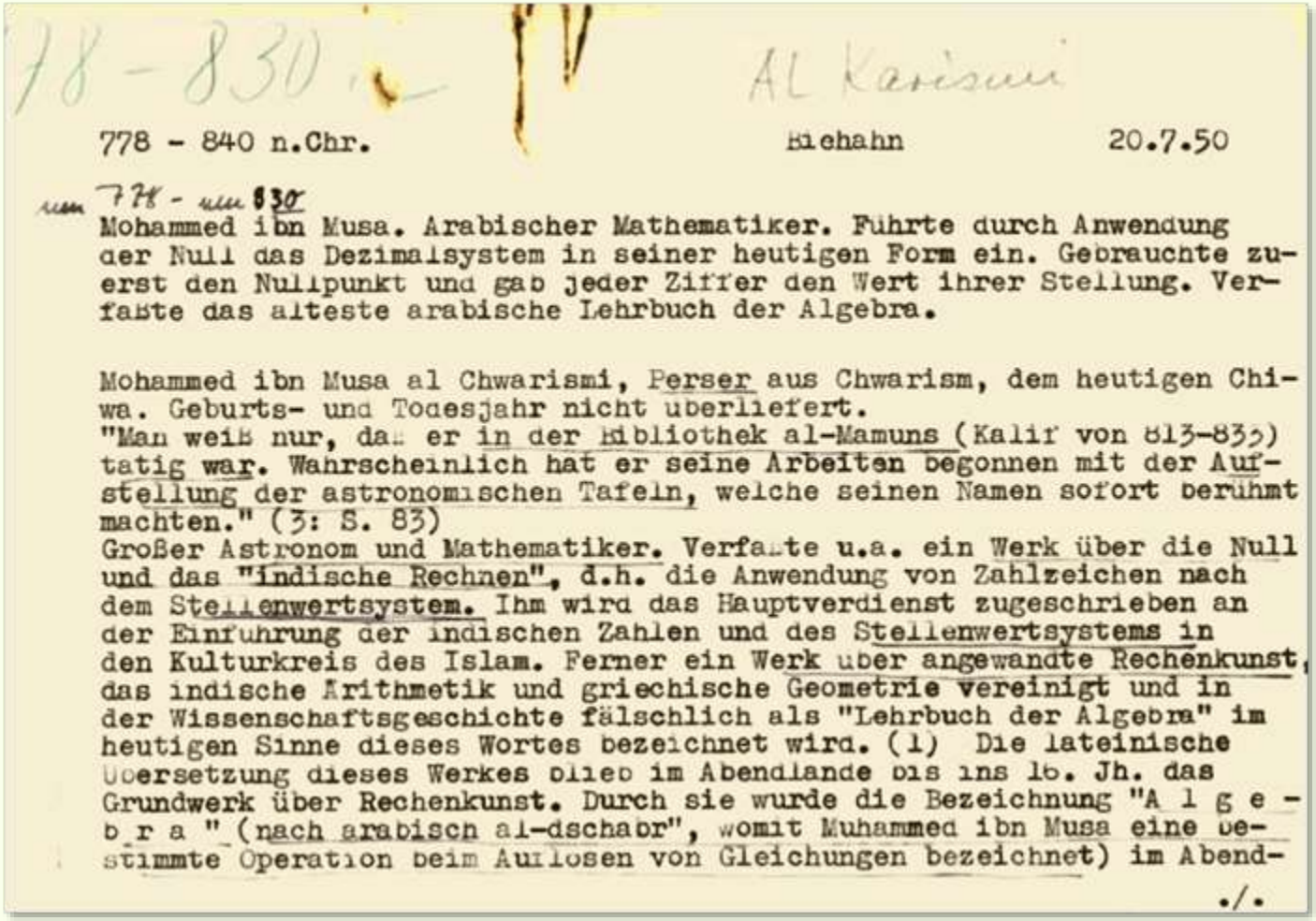
Al-Chwarizmi: Denkmal in Teheran (Iran)

Bagdad im Jahre 800

www.muslimheritage.com/uploads/Mathematical_Works_of_Nasir_Al-Din_Al-Tusi_04.jpg

Um das Jahr 800, in Europa wurde Karl der Grosse gerade zum Kaiser gekrönt, blühten in Bagdad die Wissenschaften und Künste auf. Die Kalifen Harun ar-Raschid und sein Sohn und Nachfolger Al-Ma'mun förderten sehr die Entwicklung der Wissenschaft. Aus ihrer Bibliothek heraus wurde das „Haus der Weisheit" gegründet, eine Art Akademie mit zugehörigem Observatorium, die auch viele christliche und jüdische Wissenschaftler und Philosophen anzog. Al-Chwarizmi war Mitglied dieser illustren Gelehrtengemeinschaft. Zahlreiche griechische, aber auch persische und indische Werke zur Logik, Mathematik, Medizin und Astronomie wurden ins Arabische übersetzt. Begünstigt wurde dies durch die Entwicklung der Papierherstellung im arabischen Raum; in Bagdad baute man zu dieser Zeit eine Papiermühle.

Al-Chwarizmi auf einer Karteikarte



778 - 840 n.Chr. Biehahn 20.7.50

um 778 - um 830
Mohammed ibn Musa. Arabischer Mathematiker. Führte durch Anwendung der Null das Dezimalsystem in seiner heutigen Form ein. Gebrauchte zuerst den Nullpunkt und gab jeder Ziffer den Wert ihrer Stellung. Verfaßte das älteste arabische Lehrbuch der Algebra.

Mohammed ibn Musa al Chwarismi, Perser aus Chwarism, dem heutigen Chiwa. Geburts- und Todesjahr nicht überliefert.
"Man weiß nur, daß er in der Bibliothek al-Mamuns (Kalif von 813-833) tätig war. Wahrscheinlich hat er seine Arbeiten begonnen mit der Aufstellung der astronomischen Tafeln, welche seinen Namen sofort berühmt machten." (3: S. 83)
Großer Astronom und Mathematiker. Verfaßte u.a. ein Werk über die Null und das "indische Rechnen", d.h. die Anwendung von Zahlzeichen nach dem Stellenwertsystem. Ihm wird das Hauptverdienst zugeschrieben an der Einführung der indischen Zahlen und des Stellenwertsystems in den Kulturkreis des Islam. Ferner ein Werk über angewandte Rechenkunst, das indische Arithmetik und griechische Geometrie vereinigt und in der Wissenschaftsgeschichte fälschlich als "Lehrbuch der Algebra" im heutigen Sinne dieses Wortes bezeichnet wird. (1) Die lateinische Übersetzung dieses Werkes blieb im Abendlande bis ins 16. Jh. das Grundwerk über Rechenkunst. Durch sie wurde die Bezeichnung "A l g e b r a " (nach arabisch al-dschabr", womit Muhammed ibn Musa eine bestimmte Operation beim Auflösen von Gleichungen bezeichnet) im Abend-
./.

Bevor es Computer, Datenbanken, Internet und Programme zur Textverarbeitung gab, war das Erstellen eines Nachschlagewerks und das dafür notwendige Zusammenführen von Information aus verschiedenen Quellen mühsame Handarbeit. Hier als Beispiel eine von ca. 34000 Karteikarten aus dem Jahr 1950 (es ist die erste von drei zum Stichwort *Al-Chwarizmi*, im Buch dann mit „Al Karismi" umschrieben) für die „Synchronoptische Weltgeschichte" von Arno und Anneliese Peters, an der 80 Personen anderthalb Jahrzehnte arbeiteten.

Die „Synchronoptische Weltgeschichte" erschien 1952 zweibändig im Atlasformat; ausklappbare Doppelseiten stellen zeitgleich (synchron-) auf einen Blick (-optisch) die Geschichte der menschlichen Zivilisation dar, unterteilt in Sparten wie Persönlichkeiten, Wirtschaft und Technik, Geistesleben und Kultur, Politik, Kriege und Revolutionen.

http://derdigitalepeters.de/das-programm/die-bausteine/

Zum Wort und Begriff „Informatik“

1968 war ein Schlüsseljahr

Wo wir schon Etymologie betreiben: Was ist mit „Informatik“?

- Karl Nickel (1924 – 2009) schreibt dazu 1971: *Wenn ich mich recht erinnere, war es während des von Prof. Lehmann (Dresden) einberufenen III. Internationalen Kolloquiums über aktuelle Probleme der Rechentechnik in Dresden vom 18.2. – 25.2.1968, daß dieser Name „erfunden" wurde. Während und außerhalb der Tagung wurden die verschiedensten Namen als Äquivalent für das englische „computer science" vorgeschlagen, wie etwa „Computer-Theorie" und „Komputor-Theorie", „Theorie der Informationsverarbeitung" („Informationstheorie" war schon für ein Spezialgebiet verbraucht), usf. Weil alle die vorgeschlagenen Namen nicht zweckmäßig erschienen (zu lang, nicht eindeutig genug,...), einigte man sich schließlich (wenn ich mich recht erinnere beim Frühstück am letzten Morgen der Tagung) auf „Informatik". Von einem Germanisten wurde ich darauf aufmerksam gemacht, daß diese Wortbildung falsch sei, korrekt müsse es „Informatorik" heißen. Inzwischen hat sich jedoch diese „falsche" Bezeichnung nicht nur im durchsprachigen Raum schon durchgesetzt.*

- Der deutsche Forschungsminister Gerhard Stoltenberg brachte dann den Begriff „Informatik“ als Name für das neue Lehr- und Forschungsgebiet bei einer gemeinsamen Konferenz von MIT und TU Berlin im Juli 1968 in die Öffentlichkeit. (Vorher sollte das Wort „Informatik“, aus Rücksicht auf die Namensrechte der Firma Standard Elektrik Lorenz daran, nicht offiziell benutzt werden.) Die Presse stürzte sich nach Stoltenbergs Ankündigung einer substantiellen staatlichen Finanzierung des neuen Gebietes darauf und machte so den Begriff bekannt.

Zum Wort und Begriff „Informatik“ (2)

Die vom deutschen Forschungsministerium im Juli 1968 herausgegebenen „Empfehlungen zur Ausbildung auf dem Gebiet der Datenverarbeitung“ werden mehrfach publiziert, oft schon direkt mit dem Titel „Studiengang INFORMATIK“:

...und seine Bedeutung

Der „Fachbeirat für Datenverarbeitung“, der das Bundesministerium für wissenschaftliche Forschung bei den Förderungsmaßnahmen auf dem Gebiet der Datenverarbeitung berät, hat Überlegungen über eine Verbesserung der akademischen Ausbildung auf dem Gebiet der Datenverarbeitung angestellt und Empfehlungen hierüber erarbeitet.

Diese Empfehlungen sind vom Bundesminister für wissenschaftliche Forschung dem Präsidenten der Ständigen Konferenz der Kultusminister der Länder der Bundesrepublik Deutschland, dem Vorsitzenden des Wissenschaftsrates und dem Präsidenten der Westdeutschen Rektorenkonferenz zugeleitet worden.

Die „Empfehlungen zur Ausbildung auf dem Gebiet der Datenverarbeitung“ haben folgenden Wortlaut:

(1) Die rasche technische Entwicklung auf dem Gebiet der Informationsverarbeitung macht an mehreren Universitäten und Technischen Hochschulen die Einrichtung eines Studienganges

Informatik

Hier nun der Name!

erforderlich.

(2) Dieser Studiengang sollte sich an der Ausbildung im Computer Science orientieren, wie sie sich in den letzten Jahren an den US-amerikanischen Hochschulen entwickelt hat. Er dient der Heranbildung von Akademikern für folgende Tätigkeiten:

a) In der Datenverarbeitungsindustrie: Entwicklung von Datenverarbeitungssystemen (logischer Entwurf, Entwurf von Programmiersystemen für Betrieb und Anwendung von Datenverarbeitungsanlagen).

b) Benutzer von Datenverarbeitungsanlagen (Rechenzentren in allen Bereichen der Industrie, Handel und Behörden): Pflege und Weiterentwicklung von Betriebssystemen, Beteiligung an System- und Einsatzplanungsaufgaben, Entwicklung benutzerspezifischer Anwendungsprogrammsysteme.

c) Forschung: Vorbereitungen zu eigenen Arbeiten an der Weiterentwicklung von Datenverarbeitungssystemen und von neuen Datenverarbeitungsverfahren sowie an der Erschließung neuer Anwendungsgebiete für Rechner.

Gedacht ist an einen Studiengang, der nach 9 Semestern mit einem akademischen Grad (z.B. Diplom-Informatiker) abgeschlossen sein soll, der im Niveau dem Diplom-Mathematiker bzw. Diplom-Ingenieur ent-

Zum Wort und Begriff „Informatik“ (3)

Textquelle: Internat. Elektron. Rundschau, 1968, Nr. 8, S. 211

spricht. Er umfaßt u. a. folgende Ausbildungsgebiete:

1. Mathematische Grundlagen, speziell Einführungen in
   a) Mengenlehre, algebraische Strukturen, Kombinatorik, Graphentheorie, mathematische Logik
   b) Analysis, Differentialgleichungen
   c) lineare Algebra
   d) numerische Mathematik
   e) Wahrscheinlichkeitsrechnung
2. Programmierung algorithmischer Prozesse
3. Datenverarbeitungssysteme, Organisation
4. Schaltwerkentwurf
5. Datenstrukturen und Datenorganisation
6. algorithmische Sprachen und ihre Übersetzer
7. Systemprogrammierung
8. Automatentheorie
9. Turingmaschinen und rekursive Funktionen
10. Heuristische Programmierung ·

Ergänzend dazu Lehrveranstaltungen über Statistik, Systemsimulation, Unternehmensforschung, Spieltheorie, Codierungs- und Informationstheorie, mathematische Optimierung, Algebra und Impulstechnik.

Im Anschluß an das Diplom sollte im Rahmen eines Aufbaustudiums auch die Möglichkeit zur Promotion bestehen.

(3) Die Verwirklichung dieser Studieneinrichtung sollte dadurch gefördert werden, daß die auf diesem Gebiet bereits tätigen Institute durch die Einrichtung neuer Lehrstühle verstärkt werden. Es scheint zweckmäßig, diese Lehrstühle in einem gegebenenfalls interfakultativen Institut zusammenzufassen.

(4) Diesem Institut sollte im Rahmen des Forschungsprogramms eine eigene Großrechenanlage zur Verfügung gestellt werden, evtl. mit der Auflage, damit auch die Funktion des Hochschulrechenzentrums zu übernehmen.

(5) Zur Förderung der Anwendungsmethoden auf den verschiedenen übrigen akademischen Disziplinen (Betriebswirtschaft, Medizin, Rechtswissenschaft usw.) sollen

a) von den Informatik-Lehrstühlen Lehrveranstaltungen zur Einführung in die Datenverarbeitung für Nichtinformatiker geboten werden.

b) Informatiker die Möglichkeit haben, im Rahmen ihrer Ausbildung in Wahlfächern Einführungen in die verschiedenen Anwendungsgebiete zu hören, die von den entsprechenden Fakultäten geboten werden.

c) Gemeinschaftsforschungsprojekte zwischen Informatik-Lehrstühlen und Lehrstühlen aus anderen Fakultäten gefördert werden. □

Zum Wort und Begriff „Informatik“ (4)

Kurt Mehlhorn (Jahrgang 1949) studierte von 1968 bis 1971 an der TU München und promovierte 1974 an der Cornell University. 1975 wurde er mit 26 Jahren Professor an der Universität des Saarlandes in Saarbrücken. 1990 wurde er Gründungsdirektor des Max-Planck-Instituts für Informatik; bis zu seiner Emeritierung im August 2019 leitete er dort die Arbeitsgruppe „Algorithmen und Komplexität“. Er erinnert sich, wie er als einer der Pioniere im Jahr 1968 zur Informatik kam:

„Mein Abitur machte ich 1968. Schon mit 14 war mir klar, dass ich Mathematik studieren würde. An meinem ersten Tag an der Universität erwähnte ein Kommilitone, dass es ein neues Studienfach ‚Informatik‘ gäbe und er vorschlagen würde, dass wir ‚Informatik I‘ hören sollten – was wir dann auch taten. Die Informatikvorlesungen waren roh und ungeschliffen, und der Schwierigkeitsgrad der einzelnen Vorlesungen schwankte stark. Einige waren trivial, andere sehr schwierig. Unser Dozent war F.L. Bauer, einer der Gründungsväter der deutschen Informatik. Er verdeutlichte uns, dass wir eine neue Welt betreten würden. Er konnte nicht beschreiben, wie diese Welt aussehen würde, aber sie würde ganz bestimmt wunderbar sein.

Ich war fasziniert. Ich lernte, einer Maschine das Lösen geistig anspruchsvoller Aufgaben beizubringen, wie etwa das Berechnen kürzester Wege. Kleine Probleme konnte ich manuell lösen, aber nachdem ich eine Maschine instruiert hatte, wie man dabei vorgeht, konnte sie sehr viel grössere Probleme lösen. Ich begriff, dass Computer die Denkkraft verstärken.“

F.L. Bauer wollte übrigens schon ein Jahr früher einen Informatik-Studiengang in München gründen. Da die Firma Standard Elektrik Lorenz (SEL) das Namensrechte an „Informatik“ hielt (für ein System, welches für das Versandhaus Quelle realisiert wurde), etablierte er 1967 zunächst einen Studienzweig „Informationsverarbeitung“ innerhalb der Mathematik.

Zum Wort und Begriff „Informatik“ (5)

VORLESUNGEN IN COMPUTERWISSENSCHAFTEN

Lehrveranstaltungen der Fachgruppe für Computer-Wissenschaften im SS 1971

Diese Lehrveranstaltungen umfassen:

- propädeutische Lehrveranstaltungen:

  90-180 V+U Einsatz von Rechenanlagen (Wirth)
  90-860 P Praktikum im Programmieren (Schild)

- Aufbaufächer:
  - Numerische Mathematik

    90-814 V+U Numerische Methoden I (Henrici)
  - Datenverarbeitung

    90-856 V+U Technik der Datenverarbeitung (Zehnder)
    90-887 V Theorie der Graphen (Läuchli)
    90-873 V Computer-Betriebssysteme (Engeli)
    90-868 V+U Automatic Information Organization and Retrieval (Salton)
  - Kolloquien und Seminare

    90-870 K Kolloquium über Computer-Wissenschaften
    90-869 K Kolloquium für Computer-Benützer
    90-878 S Seminar (Praktikum) AK Computer-Wissenschaften (Läuchli)
    80-876 S Seminar über Computer-Systeme (Wirth)

Grundsätzlich wird für sämtliche Aufbaufächer der Besuch der propädeutischen Lehrveranstaltungen vorausgesetzt (welche übrigens in jedem Semester (SS+WS) abgehalten werden). Dabei lernt der Anfänger im "Einsatz von Rechenanlagen" elementar programmieren, während das "Praktikum im Programmieren" daran anschliessend den Gebrauch der Programmelemente illustriert.

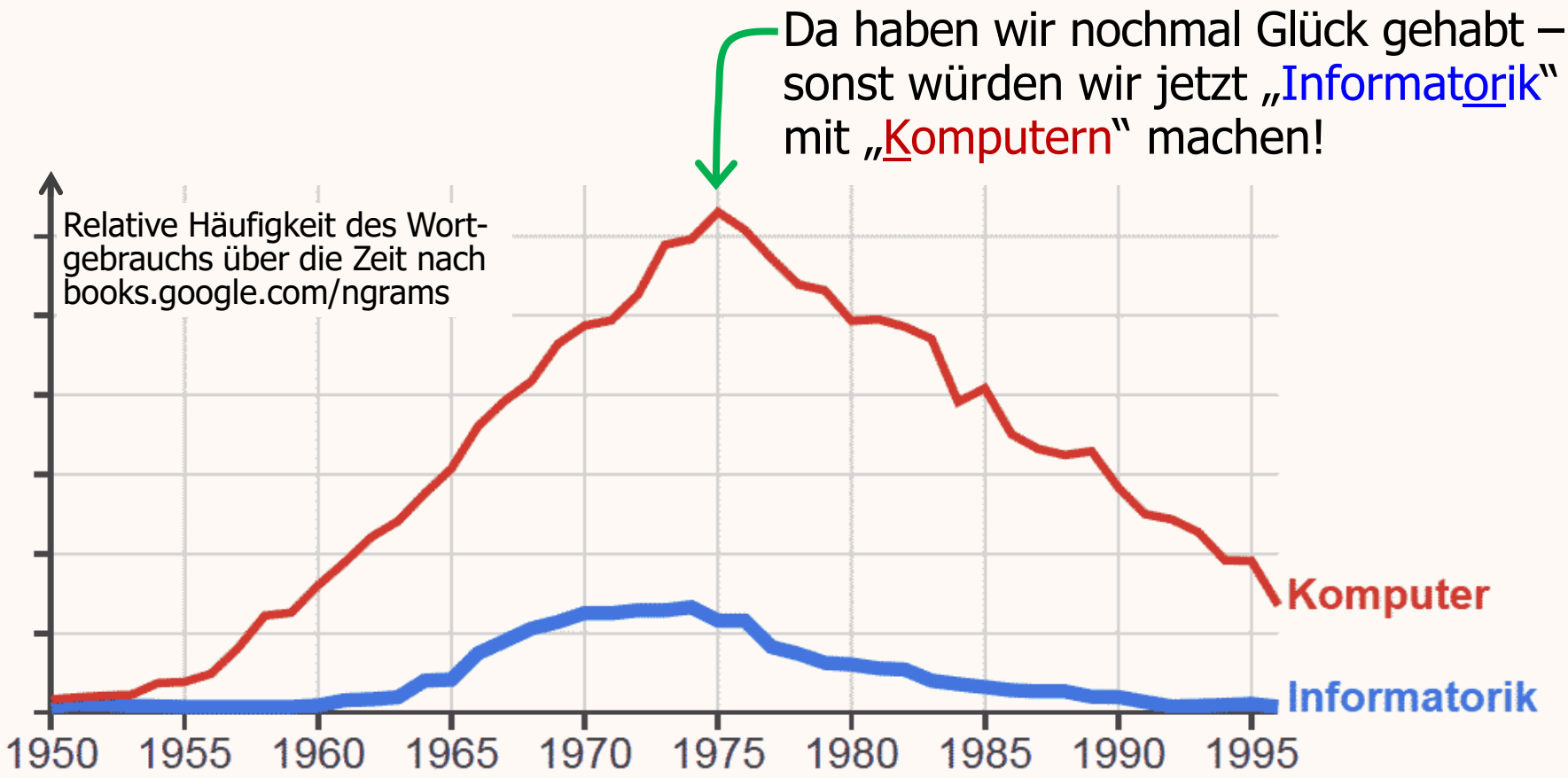
An der ETH Zürich wurde zunächst der Begriff „Computerwissenschaften“ verwendet; eine Gruppe von Professoren hielt Vorlesungen dazu

Zum Wort und Begriff „Informatik“ (6)

- An der ETH Zürich wurde die bereits 1968 unter dem Namen „Computerwissenschaften“ gegründete Fachgruppe im Jahr 1974 umgewandelt in das „Institut für Informatik“; die bisherige Bezeichnung der Fachgruppe treffe ohnehin nicht genau zu, da sich diese Gruppe weniger mit dem Computer als Objekt, sondern mit der Aktivität des Computing befasse.
- Das Wort „Informatik“ (als Verschmelzung von *Information* und *Automatik*) wurde von Karl Steinbuch (1917 – 2005) allerdings bereits 1957 in seinem Beitrag „Informatik: Automatische Informationsverarbeitung“ der SEG-Nachrichten (Standard Elektrik AG, Stuttgart) eingeführt.
- In Frankreich wurde der Begriff „informatique“ 1966 durch die Académie française legimitiert (nachdem dort bereits seit 1962 ein von Philippe Dreyfus gegründetes Unternehmen unter dem Namen „Société d’informatique appliqué“ firmierte) und so definiert: *«Science du traitement rationnel, notamment par machines automatiques, de l'information considérée comme le support des connaissances humaines et des communications dans les domaines techniques, économiques et sociaux».* Dort, und dann auch in der Schweiz, wird im Unterschied zu Deutschland (wo damit nur die Wissenschaft im Sinne von „computer science“ bezeichnet wird), „Informatik“ vor allem auch für die kommerzielle (elektronische) Datenverarbeitung und für EDV-Systeme gebraucht – eine Quelle häufiger Missverständnisse.

Informatorik? Komputer?

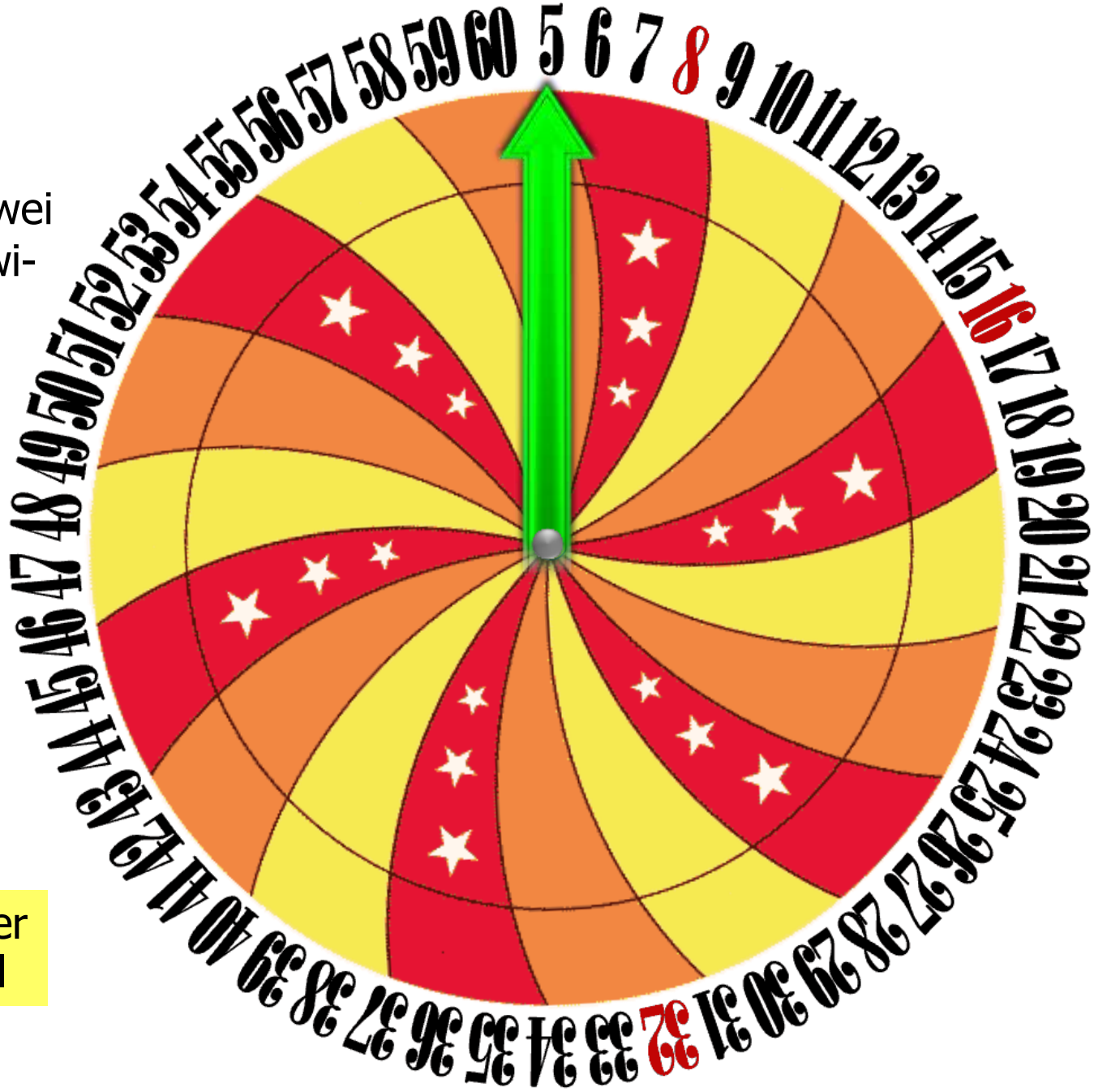
Computer Science is no more about computers than astronomy is about telescopes. -- Edsger Dijkstra



Genug (vorläufig) zur Historie – weiter geht es mit einem spannenden Algorithmus! →

Glücksrad

Wir ermitteln zwei Zufallszahlen zwischen 5 und 60



Multiplikation der Zahlen → Tafel

Einer der ältesten Algorithmen – das „altägyptische Multiplikationsverfahren“

Wird auch als abessinische oder russische Bauernmethode bezeichnet

„Wenn ihr nur duplieren und halbieren könnet, so könnet ihr das übrige ohne das Eins mal Eins multipliciren.“
Christian von Wolff, 1679 – 1754

„Der Multiplikand wird ständig verdoppelt, der Multiplikator (unter Wegwerfen des Restes) ständig halbiert; aufaddiert werden sogleich oder schlussendlich – ganz wie man will – diejenigen Vielfachen, bei denen in der Multiplikatorhalbierung ein Rest weggeworfen wurde.“

-- F. L. Bauer

Beispiel: 9 × 5 beziehungsweise 5 × 9:

9	5
~~18~~	~~2~~
36	1
45	

5	9
~~10~~	~~4~~
~~20~~	~~2~~
40	1
45	

- Verdoppeln (linke Spalte) bzw. ganzzahliges Halbieren (rechte Spalte)
- Zeilen streichen, bei denen rechts eine gerade Zahl steht
- Übrig gebliebene Zahlen der linken Spalte aufaddieren

Altägyptisches Multiplizieren

Although the method may seem bizarre to us, it actually is a very logical way to multiply if one does not have a complete number system. -- Dominic Olivastro

Nach C. Stanley Ogilvy [„Ethiopian multiplication“, Pentagon, Herbst 1950, S. 17-18]:

Ein Oberst unternahm einmal eine Expedition nach Äthiopien, und irgendwo im Landesinneren ergab sich die Gelegenheit, acht Ochsen zu kaufen.

“Wir versuchten dies”, so berichtete er, “auf dem ersten Markt, auf den wir stiessen, doch obwohl es dort Ochsen zu kaufen gab, wusste weder ihr Besitzer noch mein Führer, wie viel Maria-Theresien-Taler dazu ihren Besitzer wechseln sollten. Da keiner von ihnen rechnen konnte, stritten sie nur miteinander und kamen nicht von der Stelle. Schliesslich wurde der Priester des Ortes gerufen, da er der einzige war, der mit solchen Fragen fertigzuwerden imstande war. Der Priester erschien mit einem jungen Diener und ging daran, eine Reihe von Löchern in den Boden zu graben, jedes etwa in der Grösse einer Teetasse. Die Löcher waren in zwei parallelen Reihen angeordnet; mein Dolmetscher sagte, sie würden ‘Häuser’ genannt. Ihr Tun umfasste die ganze Mathematik, die in jener Gegend zur Abwicklung von Geschäften notwendig war, und die einzigen Voraussetzungen hierfür bestanden darin, zählen sowie mit zwei multiplizieren und dividieren zu können.

Der Diener des Priesters hatte eine mit Kieselsteinen gefüllte Tasche. In das erste Loch der ersten Reihe legte er acht Kiesel (für jeden Ochsen einen), und 11 in das erste Loch der zweiten Reihe, da jeder Ochse 11 Maria-Theresien-Taler kosten sollte. Mir wurde erklärt, dass die erste Reihe für die Multiplikation mit zwei verwendet würde: d.h. dass die doppelte Anzahl der Kiesel im ersten Loch in das zweite gelegt würde, und davon wieder die doppelte Anzahl in das dritte usw. Die zweite Reihe diene für die Division durch zwei: die halbierte Anzahl von Kieseln im ersten Loch würde in das zweite gelegt, und so weiter, bis im letzten Loch nur mehr ein Kiesel liege. Brüche würden vernachlässigt.

Dann würden die Löcher der Divisionsreihe geprüft, ob sie eine gerade oder ungerade Anzahl von Kieseln enthielten. Alle ‘Häuser’ mit einer geraden Anzahl würden als ‘böse’, alle mit eine ungeraden Anzahl als ‘gute’ bezeichnet. Aus allen ‘bösen Häusern’ würden die Kiesel herausgenommen und nicht gezählt. Die in den verbleibenden Löchern der Multiplikationsreihe enthaltenen Kiesel würden dann gezählt, und die Summe lieferte das Ergebnis.” Der Oberst staunte darüber, dass dieses Multiplikationssystem immer das richtige Ergebnis liefert, obwohl beim Halbieren einer ungeraden Zahl ein Fehler begangen wird.

Duplieren

- Verdoppeln (und auch halbieren) einer Zahl ist einfach, oder?
- Im Dualsystem geht es jedenfalls besonders einfach und schnell
  - Daher ist das für Computer keine Kunst
- Der Rechenmeister Adam Ries (1492 – 1559), der mit seinen auf deutsch verfassten Rechenbüchern wesentlich dazu beitrug, dass das römische Zahlensystem durch das praktischere indisch-arabische Stellenwertsystem mit den 10 Ziffern („Figuren", inklusive der 0) abgelöst wurde, beschreibt den Algorithmus dafür so:

Lehret wie du ein zahl zweyfaltigen solt [Rechenbuch, 1574]
Thu jhm also: Schreib die zahl vor dich / mach ein Linien darunter / heb an zu forderst / Duplir die erste Figur. Kompt ein zahl die du mit einer Figur schreiben magst / so setz die unden. Wo mit zweyen / schreib die erste / Die ander behalt im sinn. Darnach duplir die ander / und gib darzu / das du behalten hast / und schreib abermals die erste Figur / wo zwo vorhanden / und duplir fort bis zur letzten / die schreibe gantz auff.

Vorteile der altägyptischen Multiplikationsmethode

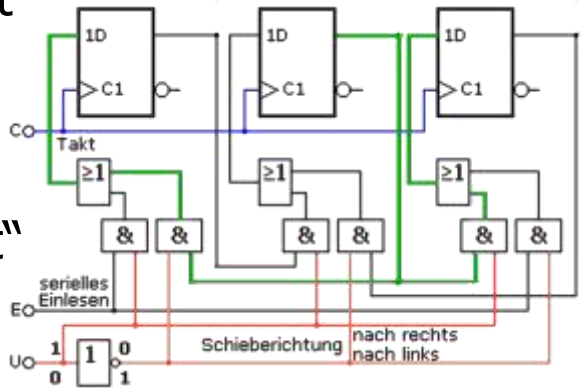
- Beschreibung ist kurz
- Einfach in der Anwendung
- Effizient für Computer im Dualsystem
  - Verdoppeln → „left shift“

    0 0 1 0 1
    0 1 0 1 0

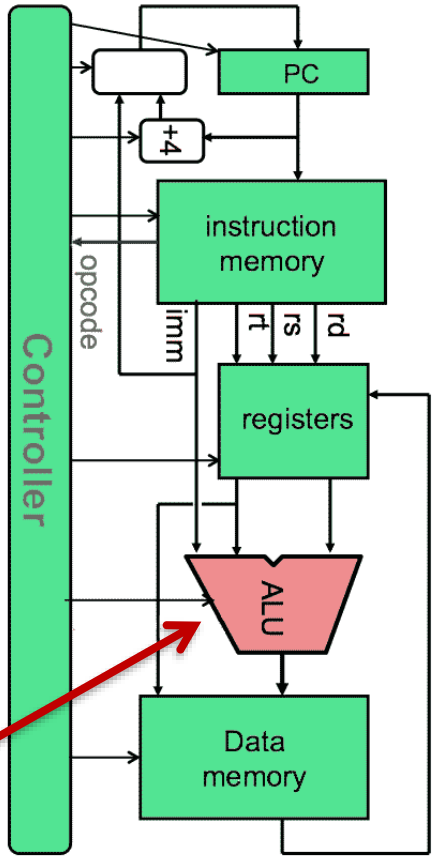
  - Halbieren → „right shift“

    („ganzzahlig“, d.h. abgerundet)

    0 0 1 0 1
    0 0 0 1 0



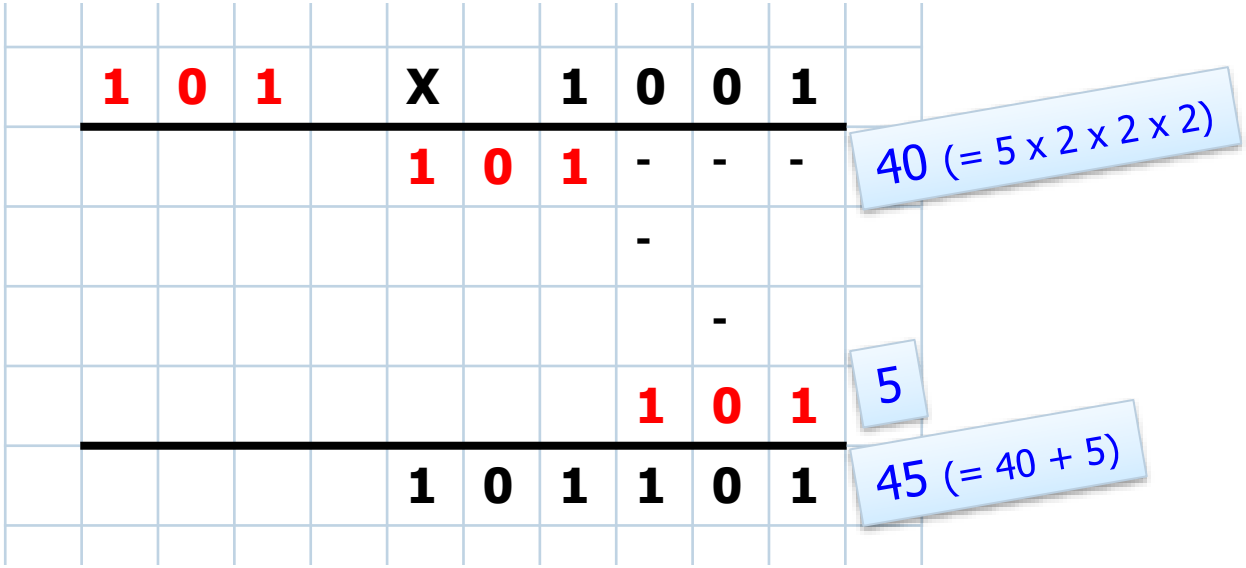
  - Algorithmus wird daher im Kern mancher CPUs verwendet, der „arithmetic logic unit“ (ALU)



CPU-Blockdiagramm

Analogie zur Multiplikation im Dualsystem

- 5 × 9 „schriftlich“ im Dualsystem multipliziert:

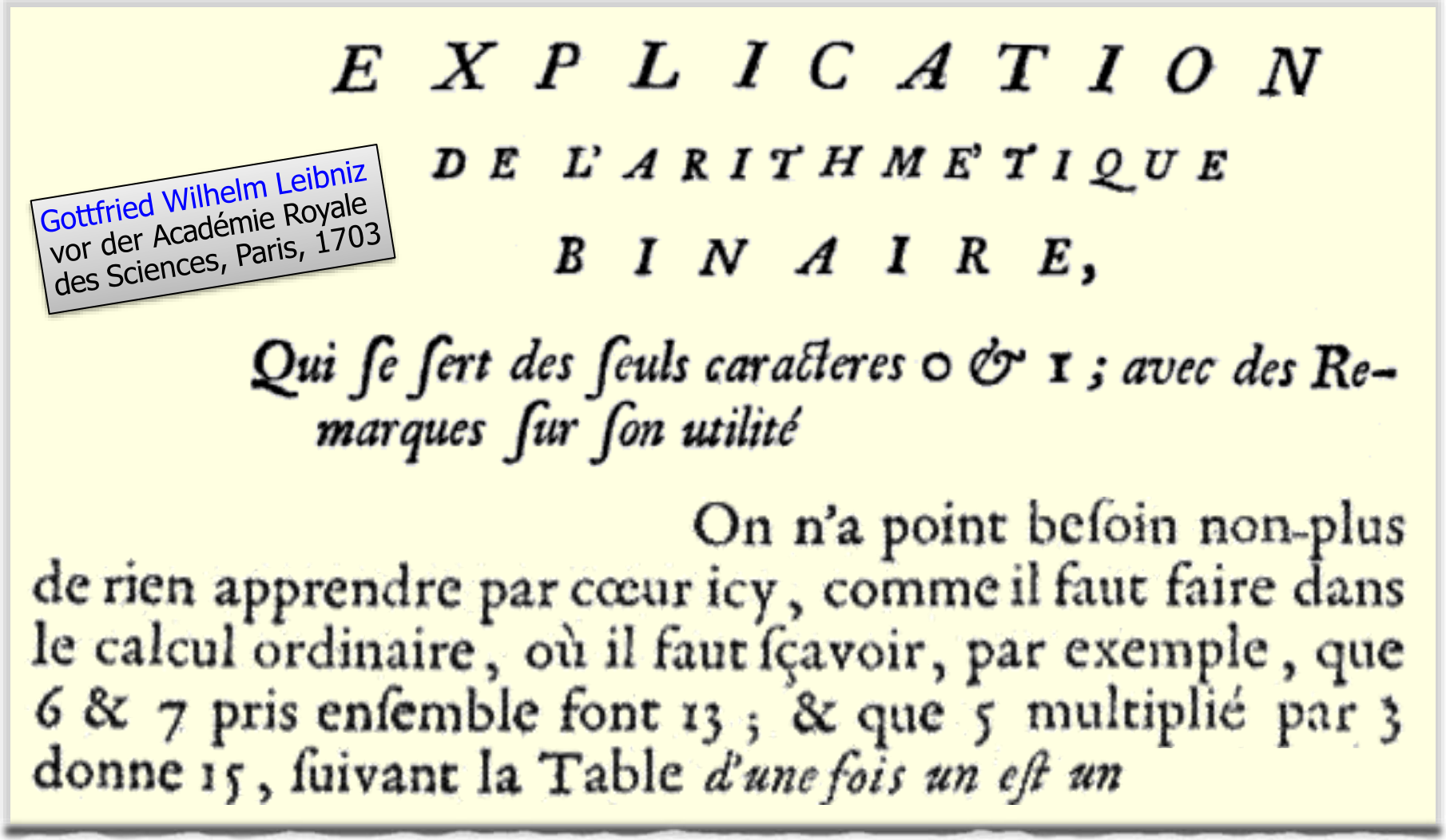


5	9
~~10~~	~~4~~
~~20~~	~~2~~
40	1
45	

- Man erkennt: Die Zweierpotenz-Vielfachen (einfach, zweifach, vierfach, achtfach etc.) des Multiplikanden 5 (≙101) werden entweder hinzugezählt oder nicht, je nachdem, ob die jeweilige Stelle des Multiplikators 9 (≙1001) eine 1 oder eine 0 in Dualdarstellung aufweist
- Bei der altägyptischen Multiplikation wird also implizit (aber eigentlich viel eleganter) eine „schriftliche Multiplikation“ im Dualsystem durchgeführt

Leibniz und das Dualsystem
(1646 – 1716)

Zu Leibniz eine längere historische Notiz



EXPLICATION
DE L'ARITHMÉTIQUE
BINAIRE,

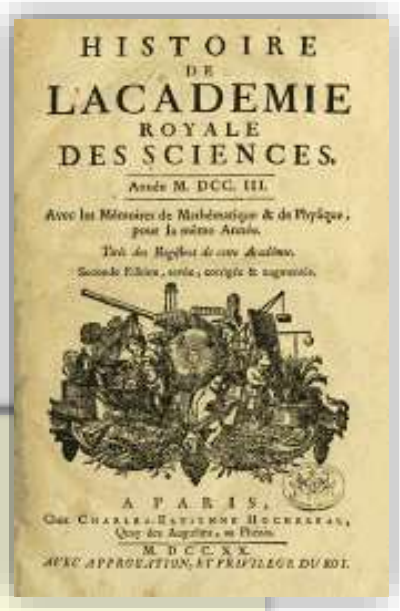
Qui ſe ſert des ſeuls caracteres 0 & 1 ; avec des Remarques ſur ſon utilité

On n'a point beſoin non-plus de rien apprendre par cœur icy, comme il faut faire dans le calcul ordinaire, où il faut ſçavoir, par exemple, que 6 & 7 pris enſemble font 13 ; & que 5 multiplié par 3 donne 15, ſuivant la Table *d'une fois un eſt un*

Gottfried Wilhelm Leibniz
vor der Académie Royale
des Sciences, Paris, 1703

Explication de l'Arithmétique Binaire

Die „Histoire de l'Académie royale des sciences" war eine der angesehensten wissenschaftlichen Zeitschriften des 17. und 18. Jahrhunderts. (Artikel von Leibniz: Seiten 85-89 der Ausgabe zum Jahr 1703, gedruckt 1705 in Paris).


HISTOIRE
DE
L'ACADEMIE
ROYALE
DES SCIENCES.
Année M. DCC. III.
A PARIS,
M. DCC. XX.
AVEC APPROBATION, ET PRIVILEGE DU ROY.

Car ici, c'eſt comme ſi on diſoit, par exemple, que 111 ou 7 eſt la ſomme de quatre, de deux & d'un.

100	4
10	2
1	1
111	7

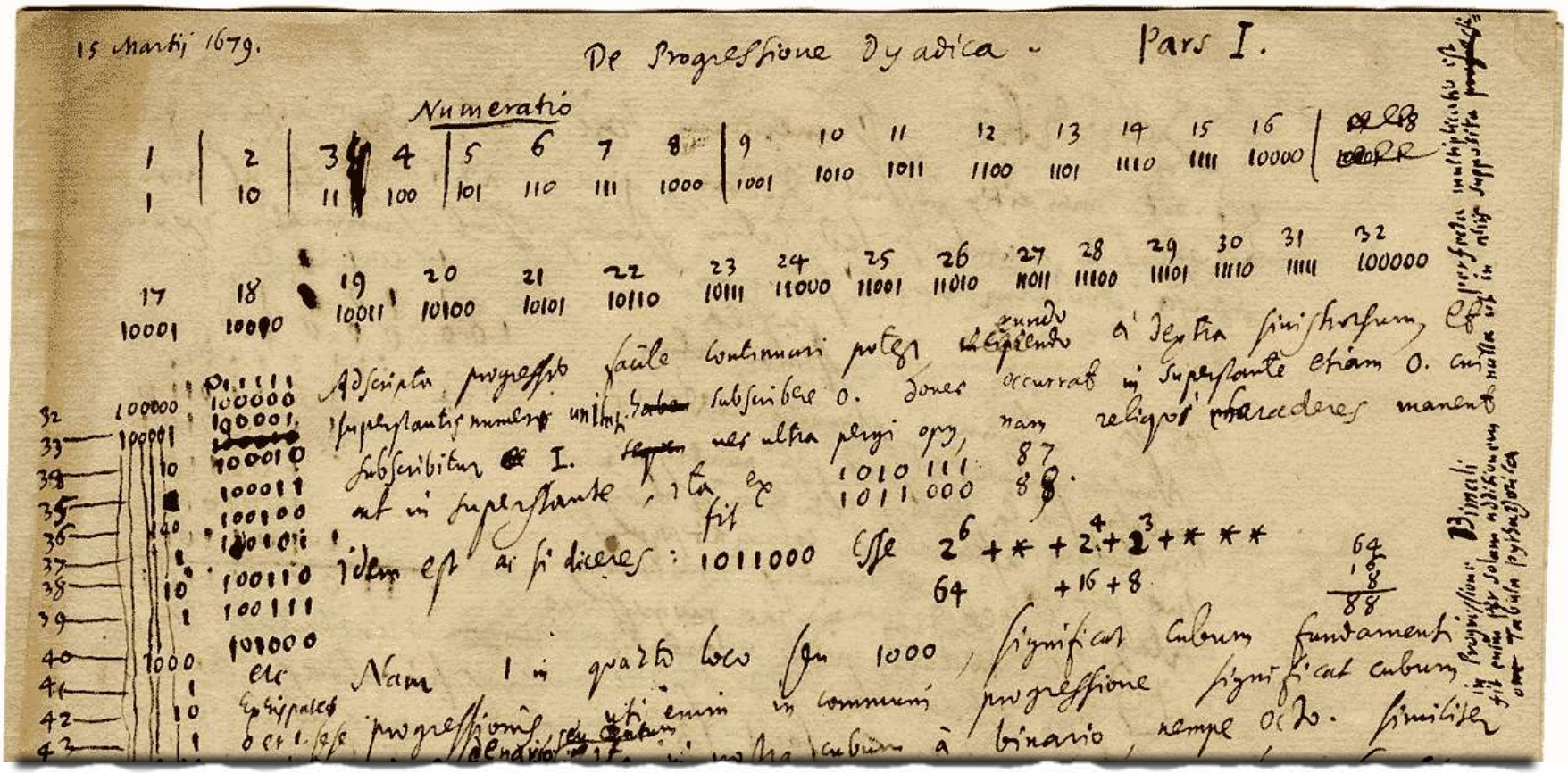
Et que 1101 ou 13 eſt la ſomme de huit, quatre & un. Cette propriété ſert aux Eſſayeurs pour peſer toutes ſortes de maſſes avec peu de poids, & pourroit ſervir dans les monnoyes pour donner pluſieurs valeurs avec peu de piéces.

1000	8
100	4
1	1
1101	13

Cette expreſſion des Nombres étant établie, ſert à faire très-facilement toutes ſortes d'opérations.

De Progressione Dyadica (Leibniz, 1679)

Leibniz erkannte schon 1679, dass im Dualsystem arithmetische Operationen viel einfacher durchgeführt werden können; er beschreibt die Grundrechenarten und gibt sich überzeugt, dass eines Tages eine „Machina Arithmeticae Dyadicae" dieses System nutzen würde.



De Progressione Dyadica – Transliteration & Übersetzung

15 Martii 1679 De Progressione Dyadica. Pars I.

Adscripta progressio facile continuari potest eundo a dextra sinistrorsum, et superstantis numeri unitati subscribere 0 donec occurat in superstante etiam 0 cui subscribitur 1 nec ultra ..., nam reliqui caracteres manent ut in superstante, ita ex 1010111 87
fit 1011000 88
idem est ai si diceres: 1011000 esse $2^6 + * + 2^4 + 2^3 + * * *$
64 +16 + 8

---

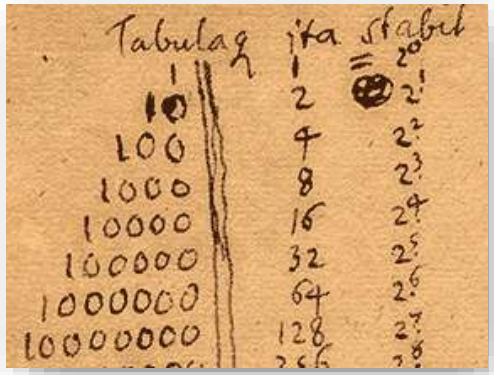
15. März 1679 Die dyadische Entwicklung. Teil I.

Nebenstehende Folge kann leicht fortgesetzt werden, wenn man, von rechts nach links gehend, unter die 1 der oberen Zahl jeweils 0 schreibt, bis bei der oberen auch 0 vorkommt, worunter man dann 1 schreibt; weiter braucht man nicht zu gehen, da die übrigen Ziffern gleichbleiben wie bei der oberen Zahl; aus 1010111 87
entsteht so 1011000 88.
Das ist dasselbe, als ob man sagte, 1011000 sei $2^6+*+2^4+2^3+ * * *$ ...

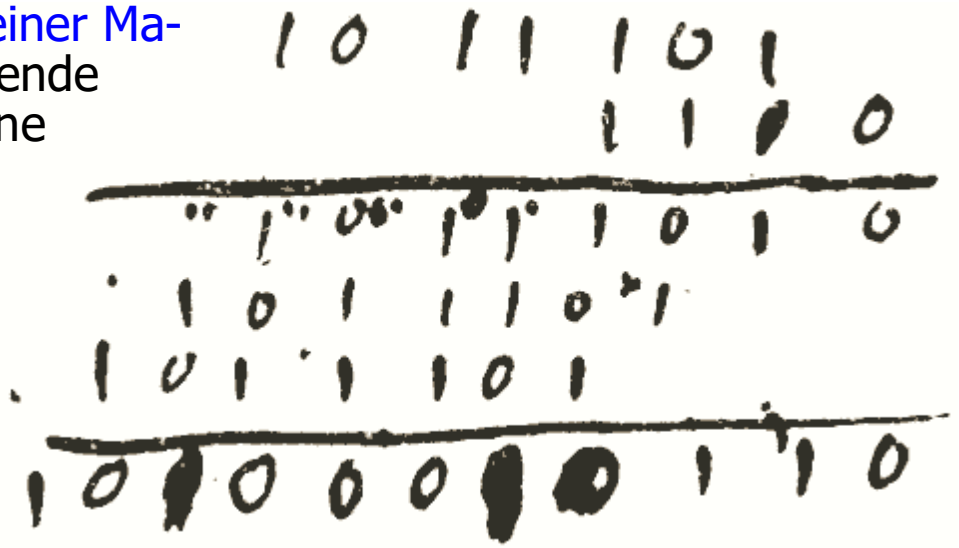
Wunderliche Vortheile der Dualzahlen

Leibniz versucht verschiedentlich, seine Zeitgenossen von den Vorteilen des Dualsystems zu überzeugen. Nachfolgend ein Ausschnitt eines Briefes von ihm an Rudolph August, Herzog von Braunschweig-Wolfenbüttel, vom 2. Januar 1697. Verbunden mit Neujahrsglückwünschen („... in beständiger Gesundheit alle selbst verlangende hohe Fürstliche Ersprießlichkeit zu gemeinem und Dero Lande besondern Besten, aus treuem Herzen anwünsche...“) und unterzeichnet mit „unterthänigster, treugehorsamster Gottfried Wilhelm Leibnitz“ schreibt er zu den Dualzahlen:

Das Schreiben wäre leicht, weilen man nur in gewisser Ordnung 0 und 1 aus dem Kopfe hinschreiben darf, also eben so geschwind und noch geschwinder, als wenn man etwas abschreibet. Eine Zahl, nach dieser Art geschrieben, wird nicht über viermal länger, als nach der gemeinen Weise. Es stecken aber, wie gedacht, noch so viel wunderbare und auch nützliche Observationen zur der Wissenschaft Vermehrung darin... ich sehe, daß sich aus dieser Schreibart der Zahlen wunderliche Vortheile ergeben werden, die hernach auch in der gemeinen Rechnung zu statten kommen würden, davon einsmals ein Mehreres erwähnet werden könnte.



Machina Arithmeticae Dyadicae (Leibniz, 1679)

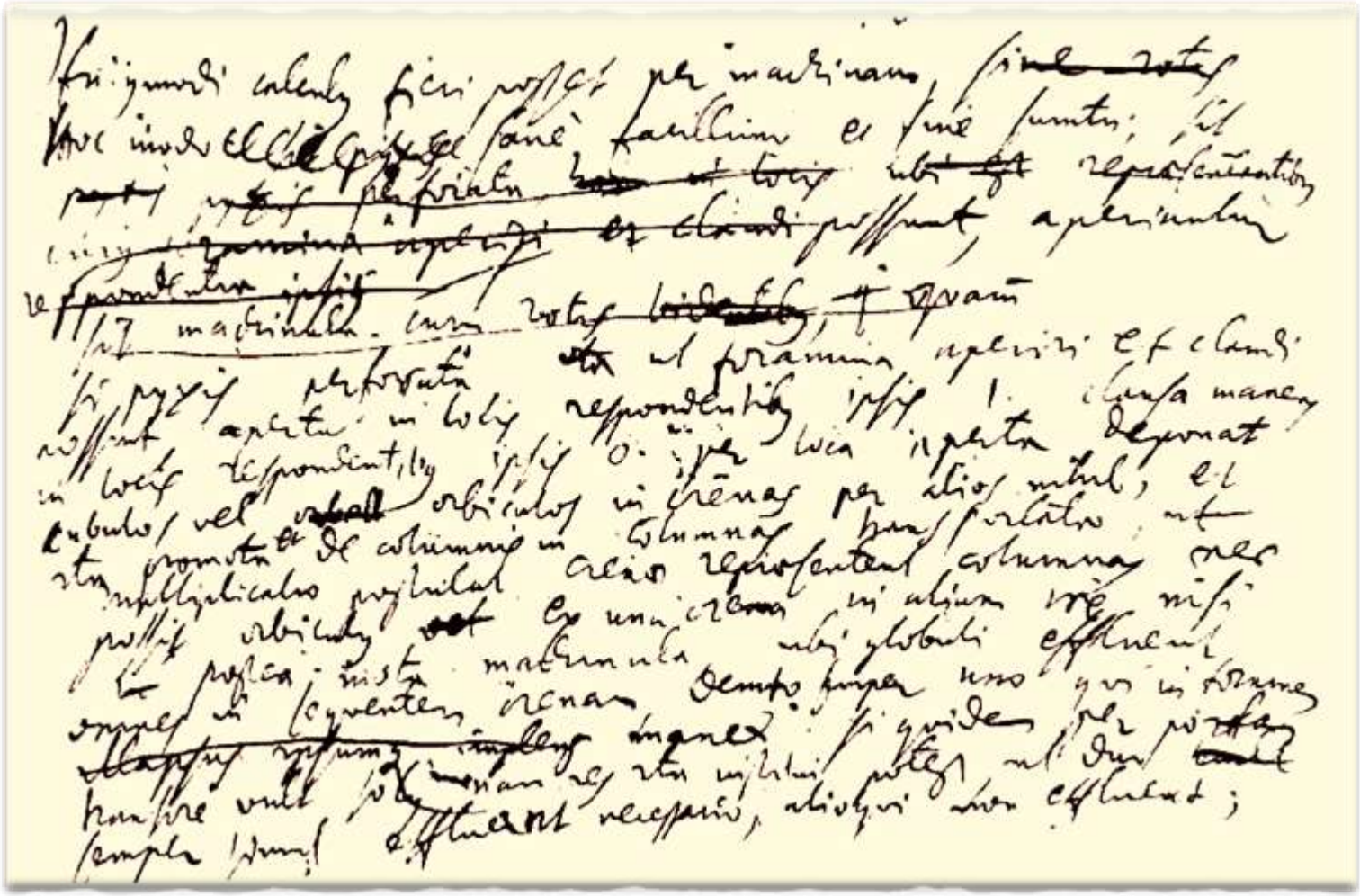


„Diese Art Kalkül könnte auch mit einer Maschine ausgeführt werden, auf folgende Weise sicherlich sehr leicht und ohne Aufwand: Eine Büchse soll so mit Löchern versehen sein, dass diese geöffnet und geschlossen werden können. Sie sei offen an den Stellen, die jeweils 1 entsprechen, und bleibe geschlossen an denen, die 0 entsprechen. Durch die offenen Stellen lasse sie kleine Würfel oder Kugeln in Rinnen fallen, durch die anderen nichts. Sie werde so bewegt und von Spalte zu Spalte verschoben, wie die Multiplikation es erfordert. Die Rinnen sollen die Spalten darstellen, und kein Kügelchen soll aus einer Rinne in eine andere gelangen können, es sei denn, nachdem die Maschine in Bewegung gesetzt ist. Dann fließen alle Kügelchen in die nächste Rinne, wobei immer eines weggenommen wird, welches in ein leeres Loch fällt, sofern es den Ausgang allein passieren will. Denn die Sache kann so eingerichtet werden, dass notwendig immer zwei zusammen herauskommen, sonst sollen sie nicht herauskommen."

Bei den letzten beiden Sätzen handelt es sich um die Beschreibung des „Zweierübertrags": Ein solcher muss stattfinden, wenn zwei oder mehr Kügelchen in einer Rinne sind. Dann müssen zunächst zwei Kügelchen austreten. Von diesen beiden fliesst das eine in „ein leeres Loch" und das andere in die nächste Rinne. Wenn nötig, muss dieser Vorgang mehrmals stattfinden; schliesslich befindet sich dann in jeder Rinne nur eine oder überhaupt keine Kugel. [Deutsche Übersetzung aus dem Lateinischen (siehe folgende Seite) nach Franz Xaver Wernz sowie Ludolf von Mackensen.]

Machina Arithmeticae Dyadicae (Leibniz, 1679)

Hier das zugehörige Manuskript auf Latein:

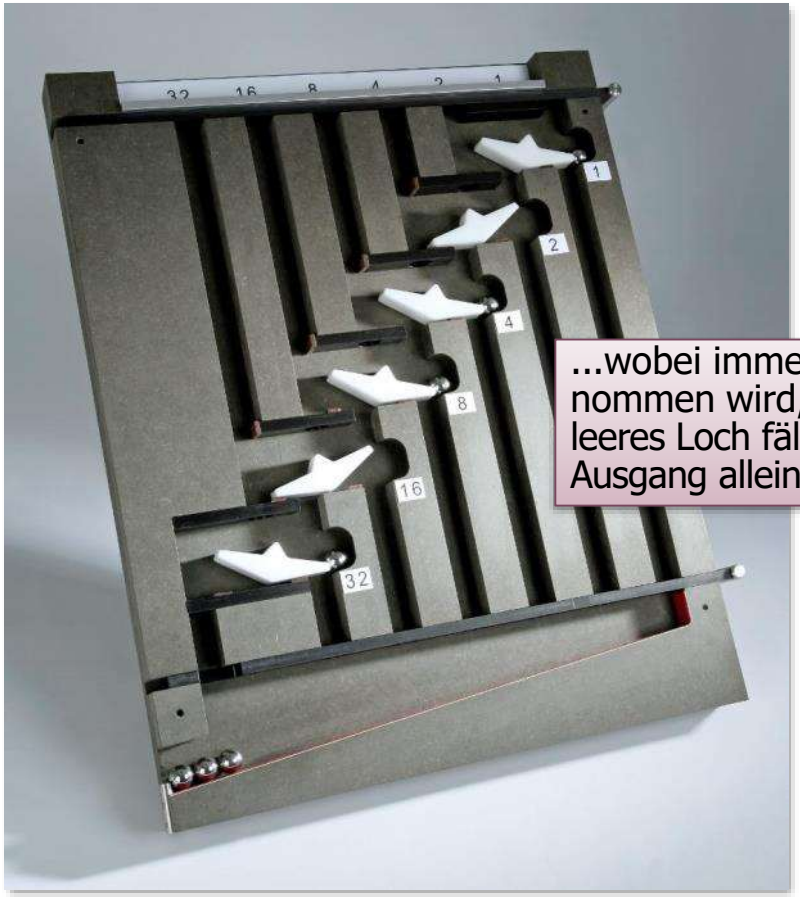


Es genügt nicht, Latein und Französisch zu können, um die Texte von Leibniz zu verstehen; bei den Manuskripten muss man auch seine Handschrift entziffern können!

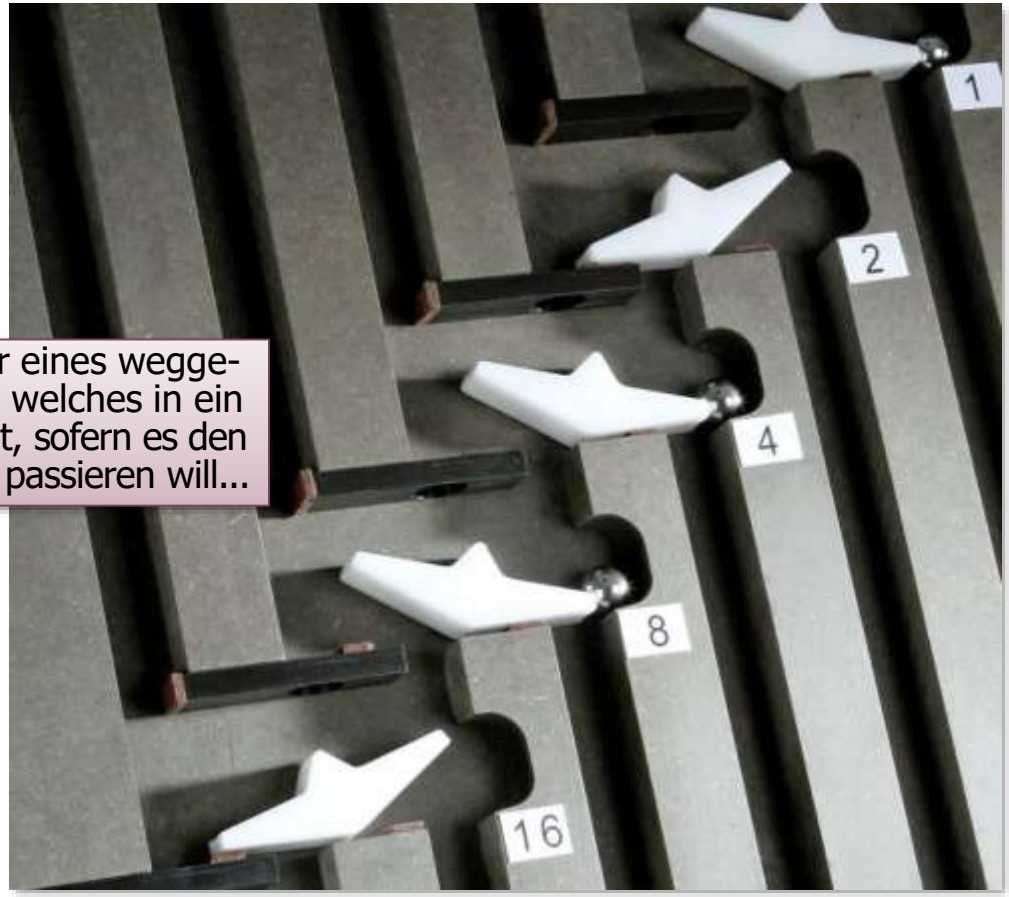
(Die 1. Zeile lautet: *„Huiusmodi calculus fieri posset per machinam sine rotis“.* Aber wieso ist „ohne Räder“ durchgestrichen? Eine zahnräderlose Rechenmaschine ist doch toll!)

„Leibniz strich viel durch. Meistens aber so, dass es noch lesbar blieb. Für alle Fälle.“ (Kathrin Zinkant)

Machina Arithmeticae Dyadicae (Modell 21. Jhd.)



...wobei immer eines weggenommen wird, welches in ein leeres Loch fällt, sofern es den Ausgang allein passieren will...



Das Herzstück der dyadischen Maschine, Addition mit Übertrag: Jede Wippe zeigt eine Binärzahl an: Eine Neigung nach links bedeutet 0, ist die rechte Seite unten, haben wir eine 1. Im ersten Fall wird eine Murmel, die von oben kommt, nach rechts abgelenkt und die Wippe umgestellt. Im zweiten Fall stellt die Murmel die Wippe ebenfalls um, rollt aber nach links zur nächsten Wippe.

Heinz Nixdorf MuseumsForum, Paderborn: http://blog.hnf.de/herr-leibniz-und-sein-dualzahlenrechner/

Machina Arithmeticae Dyadicae (Modell von 1972)

Duale 12-stellige Rechenvorrichtung nach Leibniz, 1972. Inventar-Nr. 79895, Deutsches Museum München.

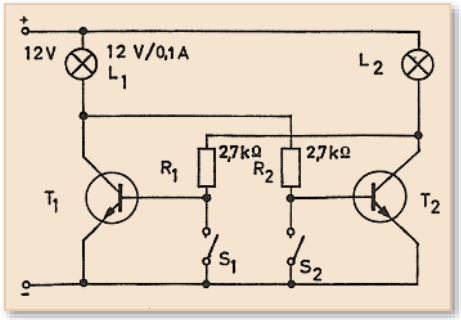
„Hier stellen wir den Maserati unter den Nachbildungen von Leibniz Rechenmaschinen vor. Hip, spektakulär, mit viel Charisma… ein wahrer Vertreter der wilden 1970er. Die Rekonstruktion wurde nach Angaben von Technikhistoriker Dr. Ludolf von Mackensen, bis 1975 am Forschungsinstitut des Deutschen Museums tätig, angefertigt.“

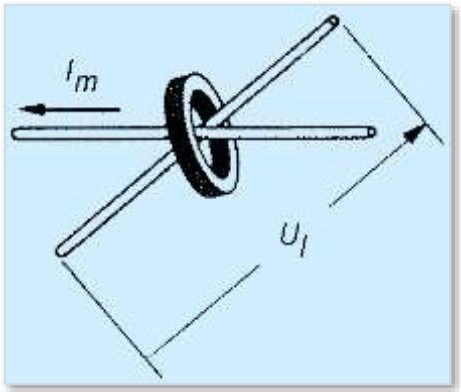
www.deutsches-museum.de/blog/blog-post/2015/04/10/depotfund-des-monats-1/

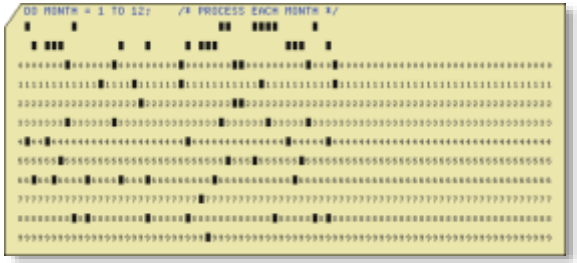
Dualsystem und digitale Rechentechnik

> Die Menschheit bleibt aber bei der Dezimaldarstellung, denn wir können uns eine siebenstellige Telefonnummer noch merken, aber nicht die entsprechende Binärdarstellung mit über 20 Bits.
>
> -- Jürg Nievergelt

Das Dualsystem ist geradezu prädestiniert für digitale Rechenmaschinen: Bei der Darstellung von Zahlen durch elektromagnetische Grössen bevorzugt man zwei leicht zu unterscheidende komplementäre Zustände wie Strom an / Strom aus oder Spannung deutlich über / unter einem Schwellwert, da auf diese Weise sehr fehlerresistente und einfache elektronische Schaltungen realisiert werden können. Bauelemente oder Schaltungen mit zwei stabilen Zuständen (z.B. Flip-Flop), die sich durch geeignet dimensionierte Impulse schnell in den jeweils anderen Zustand steuern lassen, bilden hierbei die Grundelemente. Aber auch schon rein elektromechanisch (mit Relais als ferngesteuerte Schalter) lassen sich Dualzahlen und Digitallogik implementieren. Auch die längerfristige Speicherung von Bits (als „Ziffern“ von Dualzahlen) ist relativ einfach mit diversen Technologien möglich, z.B. in der einen oder anderen Richtung magnetisierten Ringkernen (Ferritkernspeicher), Ladung in Kondensatoren oder Löchern in Lochkarten. Die Boolesche Algebra stellt in Form der Schaltalgebra zudem eine unmittelbar passende Methodik zur Realisierung der „logischen“ Steuerungen bereit.







Binäre Rechenmaschinen nach Leibniz

Friedrich L. Bauer beschreibt in seinem Buch „Historische Notizen zur Informatik“ die eher langsame Adoption des Binärsystems in der Rechnertechnik:

Die Idee des Rechnens im binären Zahlensystem bleibt nach Leibniz für 250 Jahre begraben [...]. Um 1930 bricht aber eine neue Zeit für das maschinelle Rechnen an. Um näher am binären Rechnen zu sein, verwendet Raymond Louis André Valtat Ziffernrädchen mit $2^3 = 8$ Positionen in einer französischen Patentschrift mit der Priorität 12.9.1931, er weist 1936 auf die Vorteile des binären Rechnens für den (mechanischen) Rechenwerksaufbau hin, gefolgt 1936 von Louis Couffignal in Frankreich und E. William Phillips in England. Letzterer führte ein mechanisches Modell zum Multiplizieren im Binärsystem vor und empfahl für Zahlentafeln das kompatible Oktalsystem.

Noch vor diesen entschied sich 1934 Konrad Zuse für die Verwendung des binären Zahlsystems (der früheste Beleg dafür ist eine Photographie des Versuchsaufbaus aus dem Jahr 1936) als einer natürlichen technischen Konsequenz der Verwendung von elektromagnetischen Relais, die zweier Zustände (angezogen, abgefallen) fähig sind. John von Neumann griff dann zusammen mit Herman Goldstine in dem Entwurf der ‚Princeton-Maschine‘, der 1946-48 als Bericht weite Verbreitung fand, das Binärsystem für eine elektronische Realisierung wieder auf.* Aber auch die in England von Alan Turing beeinflusste Entwicklung der Pilot ACE (James Hardy Wilkinson) war, auf Phillips Vorschlägen basierend, intern binär, extern oktal ausgelegt. Entsprechend waren auch alle anderen englischen Entwicklungen echt binär orientiert [...], während in den USA die von Howard Aiken [...] begonnenen Entwicklungen am Dezimalsystem klebten.

---

*) „In spite of the longstanding tradition of building digital machines in the decimal system, we feel strongly in favor of the binary system for our device.“

Gottfried Wilhelm Leibniz
(1646 – 1716)

Wenn ich nach Betrachtung der Wissenschaftsgeschichte einen Schutzpatron für die Kybernetik zu wählen hätte, so würde ich Leibniz wählen. – Norbert Wiener

Gottfried Wilhelm Leibniz, Philosoph, Mathematiker, Diplomat, politischer Berater, Jurist, Historiker, Sprachforscher und höfischer Intellektueller, wurde 1646 in Leipzig geboren. Er fühlte sich in der Schule weit unterfordert und brachte sich daher vieles selbst bei. Als Achtjähriger lernte er anhand der umfangreichen heimischen Bibliothek autodidaktisch die lateinische und die griechische Sprache. Im Alter von 20 Jahren hatte er die damals gängigen Hauptwerke der Mathematik, Philosophie, Theologie und Rechtswissenschaft gelesen.

Er studiert Jura und Philosophie in Leipzig und Jena und promoviert 1667 an der Nürnberger Universität. 1672 / 73 reist er in diplomatischer Mission nach Paris und arbeitet dort an seiner mechanischen Rechenmaschine mit Staffelwalze für die vier Grundrechenarten, die er (allerdings nicht ganz funktionstüchtig) der Royal Society in London vorstellt und so Mitglied dieser Gelehrtengesellschaft wurde. In dieser Zeit trifft er u.a. Huygens, Boyle und Newton. 1676 wird er Hofrat und Bibliothekar in Hannover.

Er befasst sich dann mit einer Vielzahl von Plänen und Projekten wie z.B. die Entwicklung der Infinitesimalrechnung und der Dualzahlen, die Entwässerung von Gruben mit Hilfe von Windkraft, der Perfektionierung seiner Rechenmaschine, der Entwurf einer Idealsprache und eines Logikkalküls, die Gründung einer Witwen- und Waisenkasse oder der Entwurf eines Unterseeboots. Er gilt als der historisch letzte Universalgelehrte.

An Gicht erkrankt, stirbt der „Königlich Preußische und Kurfürstlich Hannoversche Geheime Justitienrat" Leibniz 1716 in Hannover. Zu Lebzeiten erscheint nur ein kleiner Teil seiner Werke (u.a. die „Theodizee", eine Verteidigung Gottes gegen den Vorwurf, er sei ungerecht und grausam, mit der Folgerung, dass wir in der bestmöglichen Welt leben). Wichtige Manuskripte – meist lateinisch oder französisch geschrieben – werden erst nach seinem Tod publiziert, und noch immer ist Vieles seiner hohen Schaffenskraft unveröffentlicht.

Das Leibnizdenkmal in Leipzig

Leibniz wurde 1646 in Leipzig geboren und studierte an der dortigen Universität; Bürger der Stadt liessen 1883 ihm zu Ehren ein Denkmal errichten. Bei einem grossen Bombenangriff im zweiten Weltkrieg (rund 2000 Tonnen Spreng- und Brandbomben sowie Luftminen wurden am 4. Dezember 1943 aus Flugzeugen abgeworfen) wurden das Hauptgebäude der Universität

Das Leibnizdenkmal in Leipzig (2)

sowie 58 der 92 Universitätsinstitute getroffen und grösstenteils zerstört. Das Bild von 1948 zeigt das überlebende Leibnizdenkmal inmitten der Trümmer von Universitätsbauten in der Innenstadt. Ebenfalls unzerstört blieb die (im Bild nicht sichtbare) benachbarte Universitätskirche St. Pauli. Die sogenannte Paulinerkirche diente über Jahrhunderten vielen Professoren, Rektoren, Bürgermeistern sowie verdienten Leipziger Bürgern und ihrer Familien als Grabstätte. Die ehemalige Klosterkirche stammt aus dem Jahr 1240, 1545 wurde sie von Martin Luther als evangelische Universitätskirche geweiht. Sie erlitt – vom Krieg unzerstört – 1968 ein schweres Schicksal: Die DDR-Führung verfügte ihre Sprengung, und das Leibnizdenkmal wurde für viele Jahre, bis zum Neubau der Universität, eingelagert.

Direkt nach der Sprengung erfolgte die Schleifung der schätzungsweise 800 Gräber; die sterblichen Überreste wurden mehrheitlich in eine Bauschuttdeponie verkippt. Proteste gegen den Abriss führten zu Ermittlungen der DDR-Staatssicherheit und zu Verhaftungen. Interessant ist das Schicksal des Physikers Dietrich Koch, der mit seinem Bruder Eckhard und einigen anderen Personen einen zeitgesteuerten Auslösemechanismus für ein Protestplakat konstruierte, das sich beim internationalen Johann-Sebastian-Bach-Musikwettbewerb in der Leipziger Kongresshalle automatisch entrollte. Auf www.welt.de heisst es dazu: *Am 20. 6. 1968 sind es rund 1800 Zuhörer in der Kongresshalle der sächsischen Stadt. Als der letzte Redner des Abends weihevolle Worte spricht, brandet plötzlich anhaltender Beifall auf. Zuerst reagiert er geschmeichelt,*

Das Leibnizdenkmal in Leipzig (3)

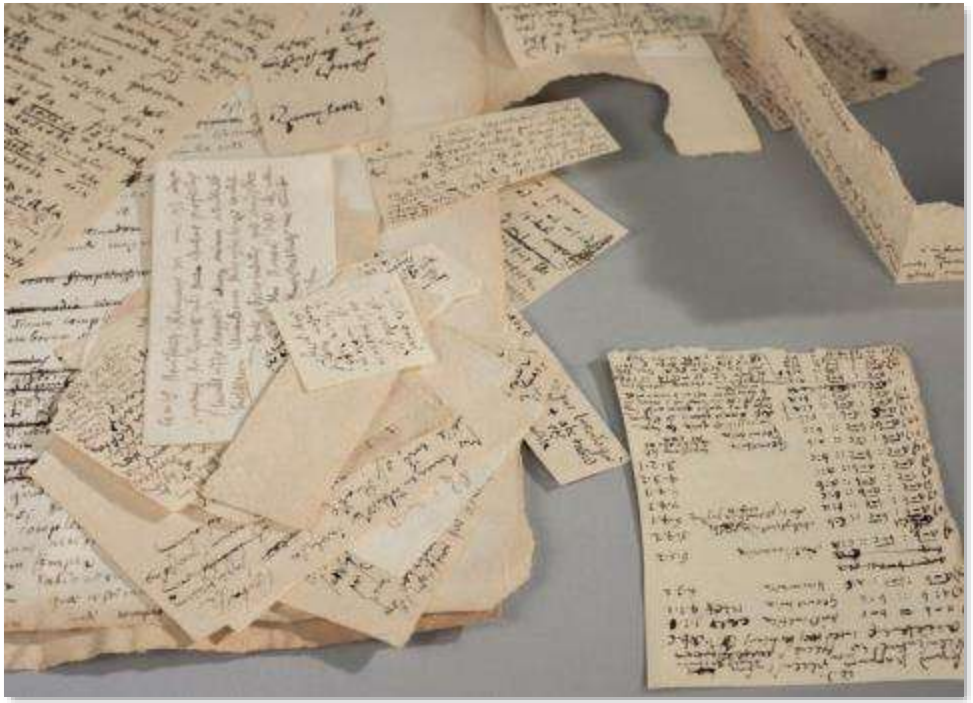
dann irritiert: Sollte seine Rede etwa so gut gewesen sein? Dann erkennt er den Grund für den unerwarteten Applaus: Während seiner Ansprache hat sich von der Decke ein großes gelbes Transparent entrollt. Darauf sind die Silhouette der Leipziger Universitätskirche und ein Kreuz zu sehen, daneben stehen die Worte: „Wir fordern Wiederaufbau."

Die Protestgruppe wurde drei Jahre später von einem „inoffiziellen Mitarbeiter" (IM) der Staatssicherheit denunziert und Dietrich Koch (nachdem anderen Gruppenmitgliedern mittlerweile die Flucht in den Westen gelungen war) von der Stasi verhaftet. Trotz nächtlicher Verhöre, Psychopharmaka und Isolation liess sich er sich nicht zu einem Geständnis bewegen. Er wurde zu einer zweieinhalbjährigen Haft und einer anschliessenden unbegrenzten Einweisung in die Psychiatrie verurteilt. Das Urteil endet mit den Worten: „Um dem Wiederholen derartigen Verhaltens vorzubeugen und damit die Gesellschaft vor staatsfeindlichen Angriffen zu schützen [...], ist des weiteren nach Verbüssung der Freiheitsstrafe [...] die Einweisung des Angeklagten in eine psychiatrische Einrichtung [...] erforderlich." Es war ein politisches Fehlurteil, eine Rache der Stasi. 1995, nach der Wende, stellte ein Gutachten der sächsischen Untersuchungskommission zum Psychiatriemissbrauch fest: „Das medizinische Gutachten ist aus heutiger Sicht nicht vertretbar. Die Beurteilung durch Dr. Petermann war im Ergebnis methodisch ungenügend, inhaltlich falsch."

Die Universitätskirche von Leipzig wurde auch nach der Wende nicht wieder aufgebaut. Die überlebensgrosse Bronzestatue Leibnizens steht heute, um 90 Grad gedreht und mit Blick nach Osten, im Innenhof des Universitätsneubaus, dem Leibnizforum.

Wie Leibniz sich buchstäblich verzettelte

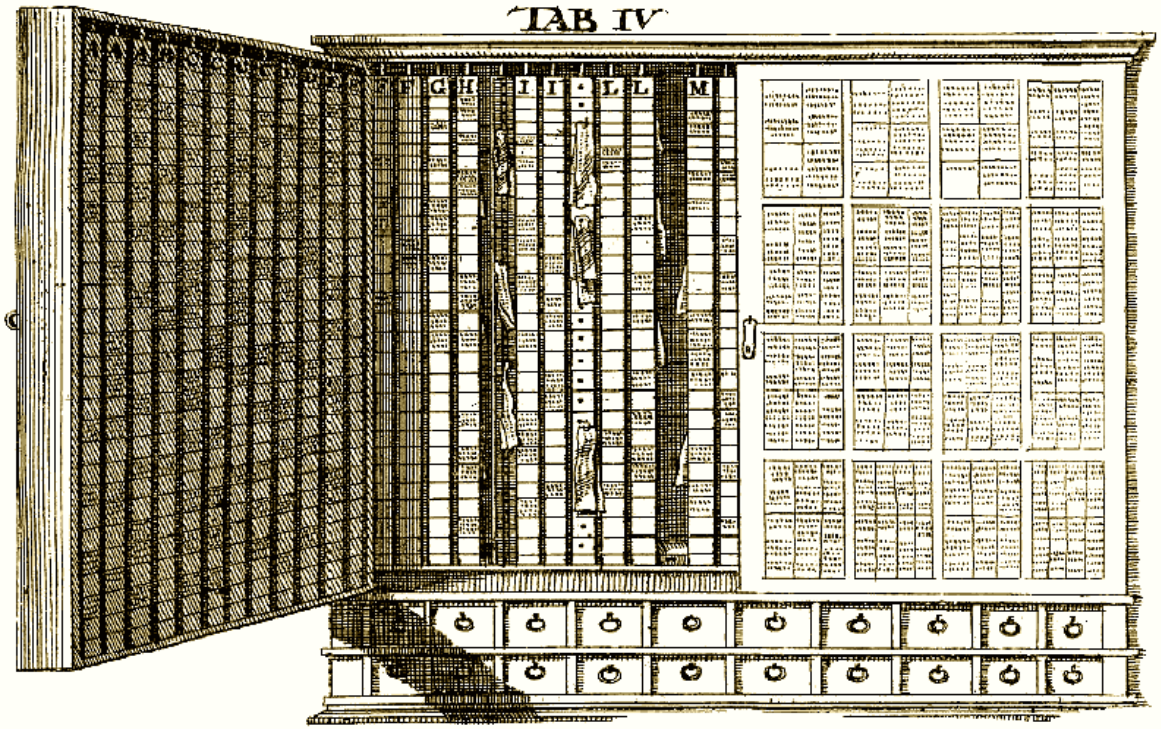
Aus einem Artikel der Süddeutschen Zeitung vom 2. 7. 2016 von Kathrin Zinkant:



www.hannover.de/var/storage/images/media/01-data-neu/bilder/redaktion-hannover.de/2015/2015_10/leibniz-schnipsel-artikel/12991820-2-ger-DE/Leibniz-Schnipsel-Artikel_image_full.jpg

Das Werk von Leibniz ist zu einem großen, wenn nicht größten Teil unbekannt. [...] Im Kern geht es um insgesamt 200 000 Seiten handbeschriebenes Papier. Darauf steht fast alles, was Leibniz nicht veröffentlich hat. Und das ist viel. Viele Gedanken, Berechnungen, Themen, von denen niemand weiß, was sie noch an Genialität zutage fördern. [...] Leibniz, dessen Schreibwut ungefähr so groß gewesen sein muss wie sein viel gerühmtes Genie, schrieb nicht einfach Blätter voll. Er übermalte angefangene Briefe, kritzelte auf Umschläge, beschriftete Schnipsel, strich großzügig ganze Absätze durch und machte sich Notizen auf Seitenrändern, selbst wenn das eigentliche Schriftstück mit der Notiz gar nichts zu tun hatte. In der Randspalte eines Textes über Reibung hielt Leibniz neben einigen Anmerkungen sogar einen Essensplan fest, in dem er sich unter anderem „zwei Würste" notiert. [...] [Eines der Probleme für die Forschung] besteht darin, dass Leibniz selbst Ordnung schaffen wollte. Und zwar mit der Schere. Er zerschnitt Seiten zu Schnipseln und sortierte sie thematisch auf kleine Stapel. Was für den Denker sicher nützlich war, ist für die Forschung ein Fluch, weil weder die Stapel noch existieren noch die Seiten, zu denen die Schnipsel einst gehörten.

Wie Leibniz versuchte, Ordnung zu halten



Leibniz besass einen Exzerpierschrank, um seine niedergeschriebenen Gedanken zu verwahren und zu sortieren. Dieser wurde vom Hamburger Gelehrten Vincent Placcius (1642 – 1699) in dessen Buch „De arte excerpendi – Vom Gelahrten Buchhalten" beschrieben und mit einem Kupferstich (s.o.) illustriert. „Nach dieser Invention ließ sich der Hannöverische Secretair Clacius einen gleichförmigen Schrank verfertigen. Nach dessen Tode kaufte ihn Herr von Leibniz", berichtet Christoph Gottlieb von Murr 1779. Der Schrank bot mit mehreren Türen eine grosse Flexibilität: Nach Stichworten sortiert, wurden Zettel mit kleinen Nägeln an den Zellen der Schranktüren aufgespiesst und konnten so immer wieder neu geordnet werden. Allerdings, so schreibt von Murr, „nach seinem Tode wurde alles untereinander geworfen, und diese Papiere sind jetzt *rudis indigestaque moles*." („Ein verworrenes rohes Gemenge" → Ovid, *Metamorphosen*)

Die unveröffentlichten Manuskripte von Leibniz

Das Entziffern und Transkribieren der Leibniz'schen Manuskripte ist recht mühsam. Der Technikhistoriker Ludolf von Mackensen schrieb dazu in seiner Dissertation von 1968:

„Ihr Verfasser war meist in Eile, getrieben von vielfältigen Ideen und Vorhaben, auf die er sich aufteilte, von unzähligen Briefwechseln und einer gewaltigen Arbeitslast bedrückt, die ihn, den Unverheirateten, nur wenig Schlaf finden ließ. Seine Augen waren kurzsichtig. Daher schrieb er meist recht klein und dicht. Es überwiegen bei ihm die französische oder lateinische Sprache gegenüber dem Deutschen. Gelegentlich benützt er Kürzel und variiert das Buchstabenbild beträchtlich, was die Graphologie genialischen Menschen zuschreibt. Viele Konzepte sind verschachtelt durch Ergänzungen, NB's, Randbemerkungen und gelegentlich auch mit Skizzen im Text versehen, die manchmal einen dürftigen, ja beinahe kindlichen Eindruck machen. Die sorgfältigeren Zeichnungen und Abschriften stammen von Schreibern, und wurden von Leibniz oft weiter ergänzt. Gelegentlich sind Worte zusammen, falsch, doppelt oder gar nicht geschrieben, besonders bei den eilig, auf das nächst-greifbare Papier geworfenen Gedanken. Groß- und Kleinschreibung, Interpunktion und Akzente verwendet er gemäß der Zeit uneinheitlich."

Leibniz' Rechenmaschinen

Den zeitgenössischen Gelehrten und gelehrten Zeitgenossen war Leibniz vor allem auch als Erfinder mechanischer Rechenmaschinen bekannt. Solche „Rechenautomaten", die analog zu den aus Zahnrädern und Stangen bestehenden Uhrwerken gefertigt wurden, waren im 17. Jahrhundert etwas prinzipiell Neues, auch wenn einerseits kunstvolle mechanische Automaten (etwa in Form von Musikautomaten oder sich bewegender Puppen) bereits bekannt waren und es andererseits Rechenhilfsmittel, also Gegenstände, die den Menschen beim Rechnen unterstützen, in anderer Form schon lange gab, etwa als Abakus, Kerbhölzer, Rechenbretter oder Rechenbänke mit Rechenpfennigen, Multiplikationstabellen, Rechenstäbe etc. Die Anwendung solcher Hilfsmittel erforderte allerdings spezielle Kenntnisse und Übung.

Die, soweit bekannt, erste solche Rechenmaschine wurde von Wilhelm Schickard (1592–1635), Astronom, Geodät und Orientalist in Württemberg, gebaut; sie ging jedoch in den Wirren des Dreißigjährigen Krieges verloren und wurde erst 1960 rekonstruiert. Die Maschine, von Schickard „Rechenuhr" genannt, beherrschte das Addieren und Subtrahieren von bis zu sechsstelligen Zahlen, einen Überlauf signalisierte sie durch das Läuten einer Glocke. In einem Brief an seinen Freund Johannes Kepler, dessen Mühe bei den langwierigen astronomischen Berechnungen von Planetenbahnen er mit seiner Erfindung erleichtern wollte, schrieb er 1623: „Ferner habe ich dasselbe, was Du rechnerisch gemacht hast, kürzlich auf me-

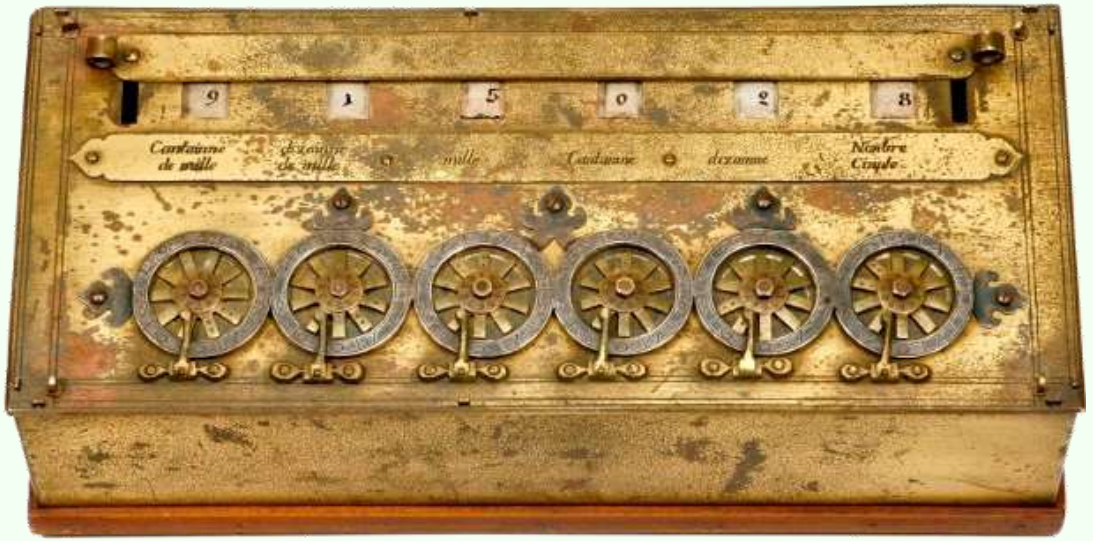
Nachbau der Rechenmaschine von Wilhelm Schickard (Foto: Wikipedia / Herbert Klaeren)

chanischem Wege versucht und eine aus elf vollständigen und sechs verstümmelten Rädchen bestehende Maschine konstruiert, welche gegebene Zahlen augenblicklich automatisch zusammenrechnet [...] Du würdest hell auflachen, wenn du da wärest und miterlebtest, wie sie, so oft es über einen Zehner oder Hunderter hinweggeht, die Stellen zur Linken ganz von selbst erhöht oder ihnen beim Subtrahieren etwas wegnimmt.“ („*Porro quod tu logistice, idem ego mechanice nuper tentavi, et machinam extruxi, undecim integris et sex mutilis rotulis constantem, quae datos numeros statim αὐτομάτως computet ... Rideres clare, si praesens cerneres, quomodosinistros denarium vel centenarium supergressos, sua sponte coacervet, aut inter subtrahendum ab eis aliquid suffuretur*“). Nachbauten aus dem Jahr 1960 und später funktionieren; jedoch ist nicht bekannt, ob die Originalkonstruktion zufriedenstellend arbeitete, insbesondere bei einem durchklappernden Zehnerübertrag, der für alle frühen Rechenmaschinenbauer eine grosse Herausforderung darstellte.

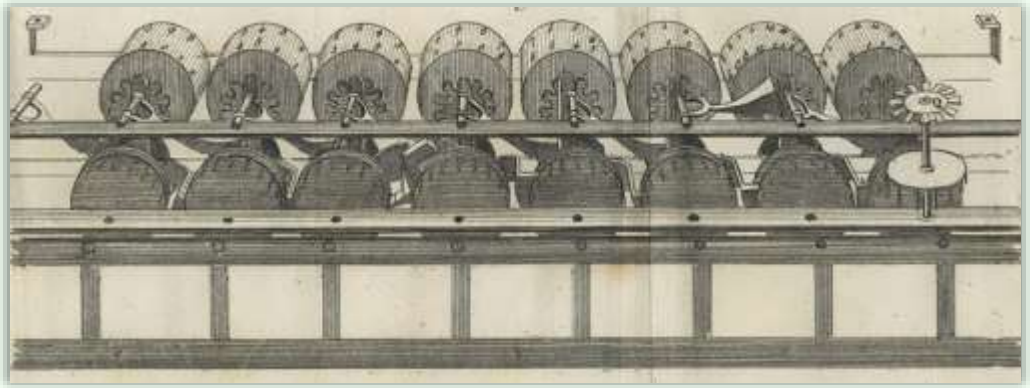
1645 führte der französische Mathematiker, Physiker und Philosoph Blaise Pascal (1623-1662) seine Rechenmaschine „Pascaline“ vor, die mit Zahnrädern und Sperrklinken funktioniert und addieren und (in späteren, verbesserten Modellen) subtrahieren konnte. Pascal begann mit der Arbeit an seiner Rechenmaschine 1642 im Alter von 19 Jahren; er konstruierte sie als Arbeitserleichterung für seinen Vater, einem Steuerbeamten. Leibniz liess sich durch die Pascaline motivieren und inspirieren.

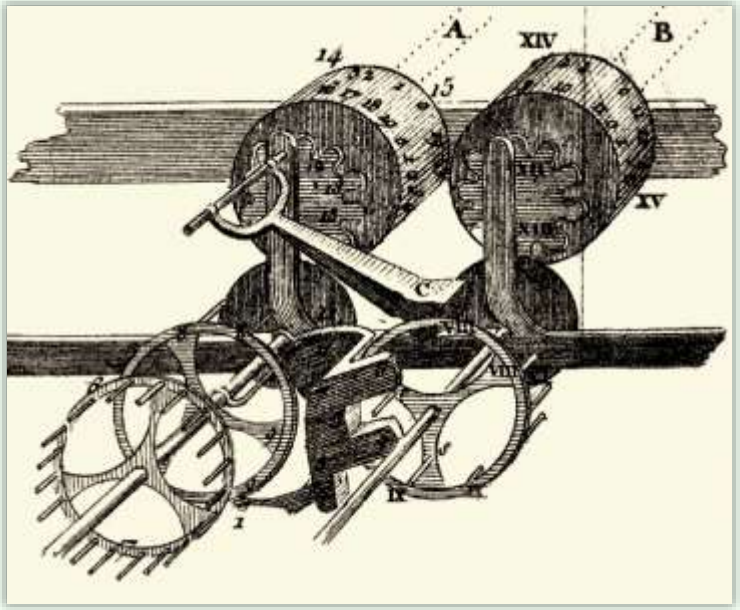
Blaise Pascal vor seiner Rechenmaschine. [Aus: Gallon, Machines et inventions approuvées par l'academie, Paris, 1735]

Leibniz strebte von vornherein eine Maschine an, die Berechnungen in allen vier Grundrechenarten, insbesondere also auch der Multiplikation, durchführt. Er schrieb dazu: „Als ich vor einigen Jahren zum ersten Mal ein Instrument sah, mit dessen Hilfe man seine eigenen Schritte ohne zu denken zählen kann, kam mir sogleich der Gedanke, es ließe sich die ganze Arithmetik durch eine ähnliche Art von Werkzeug fördern, in der nicht allein die bloße Zählung, sondern auch die Addition mit der Subtraktion und die Multiplikation mit der Division mit sicherem Erfolg von der entsprechend eingerichteten Maschine selbst leicht und bequem ausgeführt würde.“



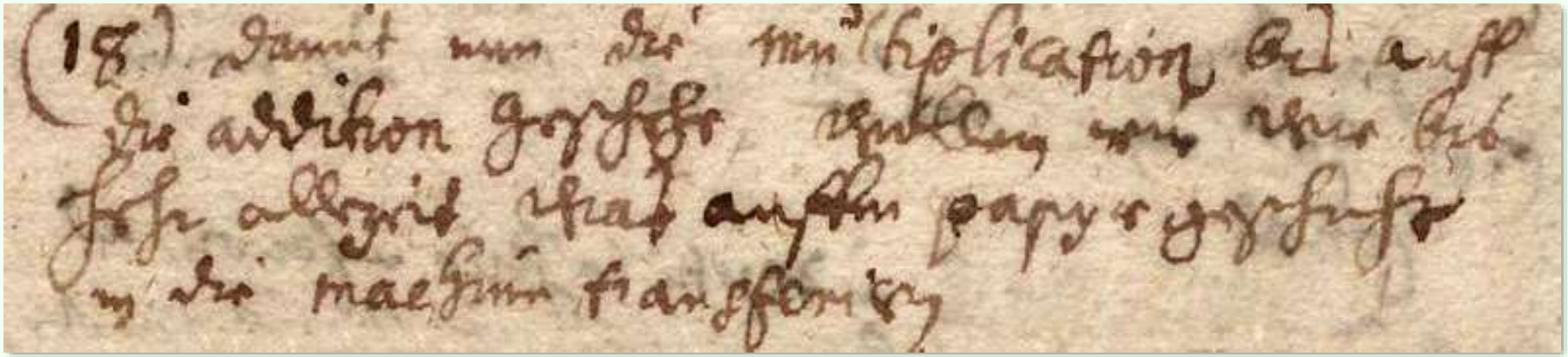
Rechenmaschine von Blaise Pascal zusammen mit zwei zeitgenössischen Funktionsskizzen zum Mechanismus des Zehnerübertrags





(„Quum aliquot abhinc annis instrumentum quoddam quod qui portat passus ipse suos ne cogitans quidem numerat, primum vidissem, — statim subiit animum cogitatio, posse toti arithmeticae simili machinamenti genere subveniri, ut non numeratio tantum, sed et additio cum subtractione, et multiplicatio cum divisione, homine successus securo ab ipsa machina recte disposita facile prompteque peragerentur.“)

Bereits als Vierundzwanzigjähriger verfasste Leibniz 1670 unter dem Titel „Instrumentum Arithmeticum“ eine Schrift, die erste Ideen zusammenfasst: Zum Addieren kann ein Zählwerk aus Rädern mit jeweils zehn Ziffern und einem Mechanismus für den Zehnerübertrag zwischen den Rädern von rechts nach links genutzt werden. Algorithmisch und bezüglich der Funktionsweise der Rechenmaschine ging Leibniz von der üblichen schriftlichen Rechenweise (im Dezimalsystem) aus; die Maschine sollte die elementaren Schritte dabei imitieren und soweit möglich automatisch vollziehen. Kurze Zeit später, vermutlich ebenfalls noch 1670, verfasste Leibniz (ausnahmsweise auf Deutsch) ein Manuskript „Instrumentum Panarithmeticon“ mit dem Untertitel „Lebendige Rechenbanck“, in dem das Übertragen der beim Rechnen auf Papier gehandhabten Operationen auf die Maschine zum Prinzip erhoben wird: „Damit nun die multiplication bis auff die addition geschehe, wollen wir wie bisher allezeit was auffm papyr geschehe, in die machina transferieren“:



Leibniz gibt später freimütig zu, dass das Additionswerk als Teil seiner Vier-Spezies-Maschine nicht originell ist: „Machina Additionis (Subtractionis) cum cistula Pascaliana in summa congruit. *(„Die Additions- und Subtraktionsmaschine stimmt mit dem Pascal'schen Rechenkästchen im Grossen und Ganzen überein.")*

Leibniz erlangte spätestens 1671 Kenntnis von der Pascaline, wovon sich ein Exemplar beim Königlichen Bibliothekar in Paris, dem Mathematiker Pierre de Carcavi, befand. Carcavi hatte in Frankreich eine zentrale Stellung als Vermittler und Ansprechpartner auswärtiger Wissenschaftler; er war auch Gründungsmitglied der Académie des Sciences. Leibniz erinnert sich 1685, wie er ihn (offenbar im Juni 1671) zur Pascaline und zu seinen eigenen Ideen für eine Rechenmaschine kontaktierte:

„Als ich somit erfuhr, dass eine derartige Maschine dort vorhanden sei, bat ich den hochwürdigen Carcavi in einem Brief um Aufklärung über die Arbeit, die sie leiste. Er antwortete, die Addition und Subtraktion würden direkt ausgeführt, das Übrige nur als Folge wiederholter Addition und Subtraktion mit einer Zusatzrechnung. Ich schrieb zurück, ich wage noch etwas Weiteres in Aussicht zu stellen, dass nämlich auch die Multiplikation ebenso wie die Addition in der Maschine mit grösster Schnelligkeit und Sicherheit bewerkstelligt würde. Jener erwiderte, dies würde nicht unerwünscht sein, und machte mir so Mut, mein Vorhaben an diesem Orte bei der angesehenen Königlichen Akademie darzustellen."

(„A quo cum didicissem Machinam eiusmodi hic extare, ab Amplissimo Carcavio per litteras petii explicationem effectus saltem quem praestaret. Qui respondit Additionem et Subtractionem recta, caetera per consequentias tantum additione et subtractione repetitis, calculo alio accedente — in ea perfici. Rescripsi me aliquid amplius promittere audere, ut scilicet muiliplicatio quoque non minus quam additio in machina summa celeritate ac certitudine perficiantur. Id ille ut non ingratum fore respondit, ita mihi ad institutum meum hoc loco apud illustrem Academiam Regiam exponendum animos fecit.")

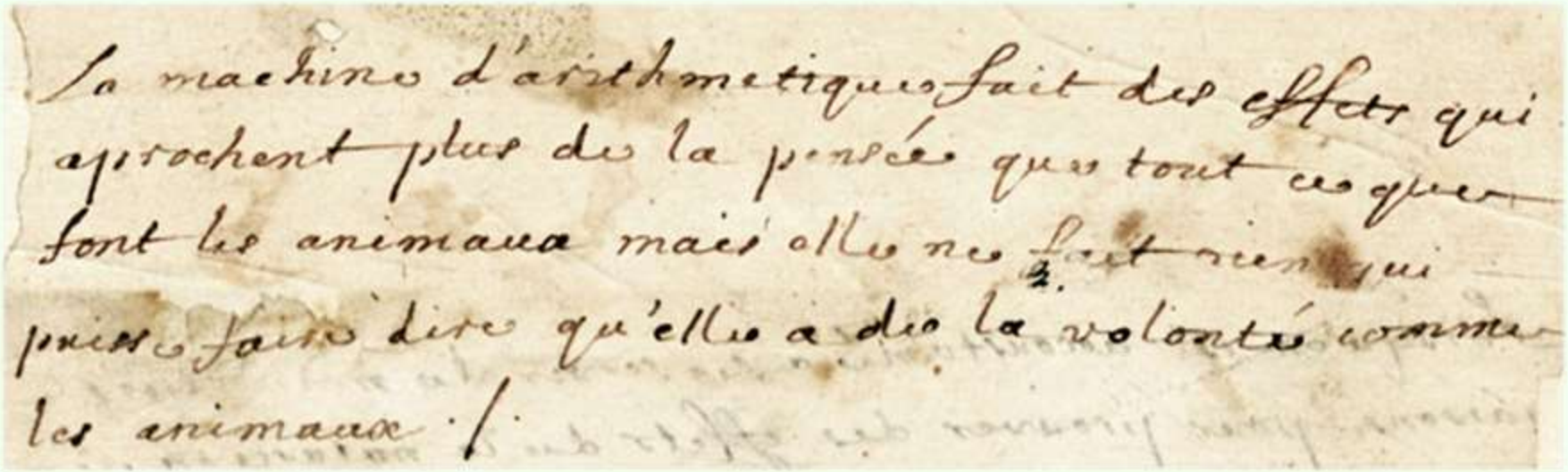
Einschub zum Pascal‘schen Rechenkasten, genannt „Pascaline“

Der französische Mathematiker, Physiker und Philosoph Blaise Pascal (1623 – 1662) wurde nur 39 Jahre alt. Es waren die langwierigen Rechnungen sein Vaters, königlicher Kommissar und oberster Steuereinnehmer für die Normandie, die Blaise Pascal zu einer Maschine inspirierten, die das Rechnen automatisieren sollte. 1645 übergibt der 22-jährige stolz Pierre Séguier, Kanzler von Frankreich unter König Ludwig XIV, ein Exemplar seiner Rechenmaschine. Im Widmungsbrief dazu heisst es: «Les longueurs et les difficultés des moyens ordinaires dont on se sert m’ayant fait penser à quelque secours plus prompt et plus facile, pour me soulager dans les grands calculs où j’ai été occupé depuis quelques années en plusieurs affaires qui dépendent des emplois dont il vous a plu honorer mon père pour le service de sa Majesté en la haute Normandie […].»

Pascal sieht seine Maschine primär als ein nützliches automatisiertes Rechenhilfsmittel. In seinem «Avis nécessaire à ceux qui auront curiosité de voir la Machine d’Arithmétique et de s’en servir» schreibt er dementsprechend: «Le plus ignorant y trouve autant d’avantage que le plus expérimenté : l’instrument supplée au défaut de l’ignorance ou du peu d’habitude, et, par des mouvements nécessaires, il fait lui seul, sans même l’intention de celui qui s’en sert, tous les abrégés possibles à la nature, et à toutes les fois que les nombres s’y trouvent disposés.»

Seine Schwester Gilberte Périer schreibt in ihrer Biographie über Blaise Pascal: «Ce fut en ce temps là à l’age de 19 ans qu’il inventa cette machine d’Arithmétique par laquelle on fait non seulement toutes sortes de supputations sans plumes & sans jetons, mais on les fait même sans scavoir aucune règle d’Arithmétique & avec une sûreté infaillible. Cet ouvrage a été considéré comme une chose nouvelle dans la nature d’avoir réduit en machine une science qui réside toute entière dans l’esprit & d’avoir trouvé le moyen d’en faire toutes les opérations avec une grande certitude sans avoir besoin de raisonnement.»

Der letzte Satz ist bemerkenswert: Man habe mit der Maschine also eine neue Wesensart in die Natur eingeführt, welche eine bisher rein geistige Fähigkeit in eine Maschine zwängt, deren Anwendung nun „gedankenlos“ erfolgen kann. Dies entspricht der Charakterisierung „ohne Arbeit des Gemüths“ von Leibniz. In den erst posthum herausgegebenen Pensées, in der diverse Notizen und Fragmente von Pascal (vor allem zur Begründung des christlichen Glaubens) zusammenfasst werden, findet sich eine weitere interessante Charakterisierung: «La machine d’arithmétique fait des effets qui approchent plus de la pensée que tout ce que font les animaux; mais elle ne fait rien qui puisse faire dire qu’elle a de la volonté, comme les animaux.» Eine solche Maschine approximiert also besser als alle Tiere das menschliche Denken, aber während man Tieren einen Willen zusprechen kann, hat eine Maschine in keiner Weise einen eigenen Willen. Dies erinnert an die moderne Diskussion, ob die künstliche Intelligenz nur Maschinen hervorbringt, die man als Werkzeuge und Instrumente betrachten kann, oder ob letztere (auch zum Nachteil des Menschen) einen eigenen Willen entwickeln können. Eine gewisse Form des Denkens kann man, so Pascal, den „Computern“ nicht absprechen, wohl aber einen inneren Antrieb und Motivation in Form eines eigenen Willens.

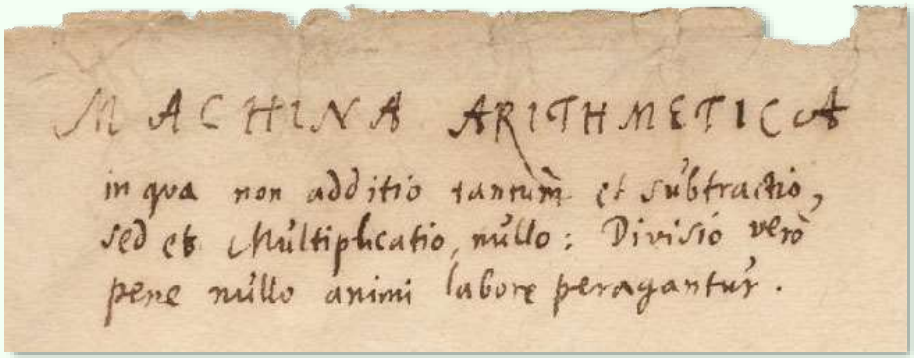


Die dritte Rechenmethode macht den Kopf überflüssig

Als Pascal 1645 seine Rechenmaschine der Öffentlichkeit vorstellt, spricht er von seinem „Einfall, eine dritte Methode zur Ausführung aller arithmetischen Operationen zu finden, die vollkommen neuartig ist und nichts mit den beiden üblichen Methoden der Feder und der Zählsteine zu tun hat.“ Bei den beiden traditionell gebräuchlichen Rechenmethoden (Abakus; Rechenstein) und (Calamus; Ziffernschrift) unterliegt das jeweils zu Grunde gelegte Zeichensystem festen Transformationsregeln, die es erlauben, komplexe Rechnungen auf einfache zu reduzieren, wie z.B. bei den Rechensteinen die Multiplikation und Division auf Addition und Subtraktion und diese weiter auf einfaches Zählen – und zwar nur von 1-5, welches durch bloßes Steineschieben realisiert wird. Der Stein bzw. der Stift müssen jedoch von Hand bewegt werden. Von einer Hand zumal, die geführt wird von einem Kopf, der die Transformationsregeln der Zeichen kennt, mit ihnen umzugehen weiß. Hier genau setzt die von Pascal entwickelte „troisième Méthode“ ein: Sie macht den Kopf überflüssig. Denn die Transformationsregeln sind in die Maschine selbst eingebaut. Letztlich ist die Maschine in ihrem mechanischen Teil nichts anderes als die Materialisierung der Transformationsregeln. Der menschlichen Hand bleibt nurmehr die Eingabe der Rechenaufgabe und die Aufgabe der Antriebsbewegung; eine Bewegung, der das Denken keinerlei Regeln mehr zu geben braucht, da dies bereits die Maschine leistet, die der Hand die einfache Drehbewegung der Kurbel vorschreibt, welche Drehbewegung sie mittels Stangen, Räder und Walzen zur Rechenoperation verwendet. Die Denkbewegung wird ersetzt durch eine mechanische Bewegung.

Soweit Stephan Meier-Oeser zu Pascals Idee, das Rechnen durch eine Maschine durchführen zu lassen (in „Die Entlastung von der Mühsamkeit des Denkens. Zeichentheoretische Bemerkungen zur Urgeschichte artifizieller Intelligenz im 17. Jahrhundert“, 1993).

Im Oktober 1671 berichtetet Leibniz in einem Brief Herzog Johann Friedrich in Hannover von seiner Idee einer „Lebendigen Rechenbank“; das Thema sollte ihn sein Leben lang nicht mehr loslassen: Insgesamt liess er im Laufe der Jahrzehnte vier Maschinen in Paris, Hannover, Helmstedt und Zeitz von den verschiedensten Mechanikern bauen, anscheinend allerdings ohne jemals eine über die Demonstration der prinzipiellen Funktion mit Beispielaufgaben hinausgehende robuste Funktionsfähigkeit, ohne dass öfter nachjustiert werden musste, zu erreichen. Sein Konzept war richtig, wie Nachbauten der heutigen Zeit beweisen; die erforderliche hohe Präzision der Zahnräder (teilweise mit Toleranzen von hundertstel Millimetern) und anderer Maschinenkomponenten (z.B. Abweichung der Orientierung einzelner Wellen von der exakten Position um höchstens 0.1 Grad) überforderte aber die Möglichkeiten der zeitgenössischen Feinmechaniker, die oft eher nach Augenmass und mit der Feile arbeiteten.



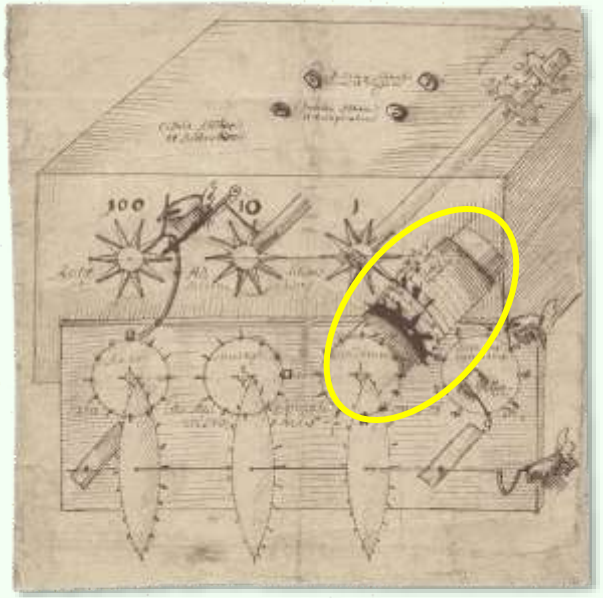
Machina Arithmetica in qua non additio tantum et subtractio, sed et Multiplicatio nullo, Divisio vero pene nullo animi labore peragantur. […bei der nicht nur Addition und Subtraktion, sondern auch die Multiplikation ohne – und die Division tatsächlich fast ohne – geistige Anstrengung ausgeführt werden.]

Ein erstes dreistelliges Holzmodell wird 1673 fertig. („Sie hat mich nicht wenig gekostet, ehe ich es so weit gebracht, dann das Model wohl 100 mahl verändert, und drey Viertheil Jahr daran gearbeitet worden“ schreibt Leibniz seinem Dienstherrn Herzog Johann Friedrich von Braunschweig.) Im gleichen Jahr stellte Leibniz seine Maschine den Mitgliedern der Royal Society in London vor und wurde damit Mitglied dieser berühmten Gelehrtengesellschaft. 1675 führte er ein in Messing gefertigtes Modell mit einem vierstelligem Eingabewerk und einer zwölfstelligen Ergebnisanzeige in der Académie des Sciences in Paris vor. In den folgenden Jahren initiierte er den Bau von zwei verbesserten Maschinen, die erst nach mühevollen Jahren fertiggestellt wurden; das letzte Modell ist heute als Original erhalten – es galt jahrzehntelang als verschollen und wurde erst 1876 in der

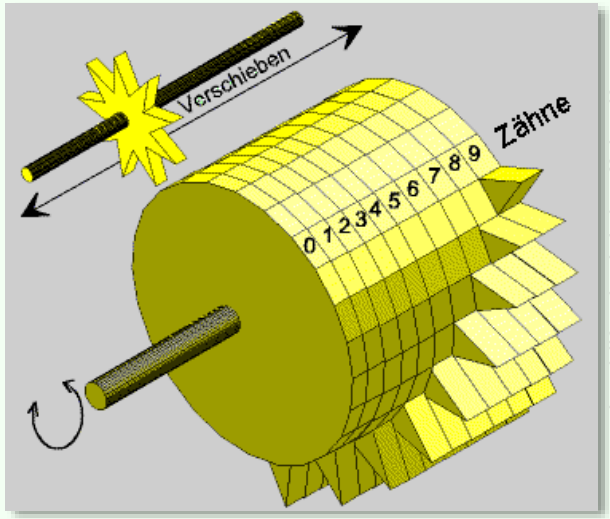
Modellkammer der Göttingen Universität wieder aufgefunden. Insgesamt steckte Leibniz ca. 24000 Taler seines Privatvermögens in die Entwicklung seiner Rechenmaschinen – für damalige Verhältnisse eine sehr hohe Summe, verdiente doch beispielsweise ein mit der Herstellung befasster Büchsenmacher nur gut 16 Taler im Jahr.

Das zentrale Funktionselement der Leibnizschen Maschinen, das eine schnelle Einstellung beliebiger Operanden sowie eine effiziente Multiplikation ermöglichte, ist die von ihm erfundene Staffelwalze – ein Zylinder mit neun Zahnrippen gestaffelter Länge auf einem Teil seines Umfanges. Je nach Position eines von dieser Staffelwalze angetriebenen, aber verschiebbar gelagerten, Zahnrades wird bei einer vollständigen Umdrehung der Staffelwalze dieses Zahnrad um null, einen, zwei,.... oder neun Zähne (das heisst Zehntel-Umdrehungen) weitergedreht. Solche Staffelwalzen fanden sich noch bis ins 20. Jahrhundert hinein in kommerziellen Rechenmaschinen.

Das letzte Modell verfügt über acht Eingabestellen und sechzehn Ausgabestellen. Eingegeben werden Zahlen über Einstellrädchen am Eingabewerk, das auf einem Schlitten gelagert ist und sich mit einer Tabulatorkurbel zwischen den Dekaden 1 und 8 gegenüber dem Rechenwerk verschieben lässt, wodurch auf einfache Weise Multiplikationen und Divisionen möglich sind: Bei einer Multiplikation mit 23 wird z.B. zunächst drei Mal hinter-



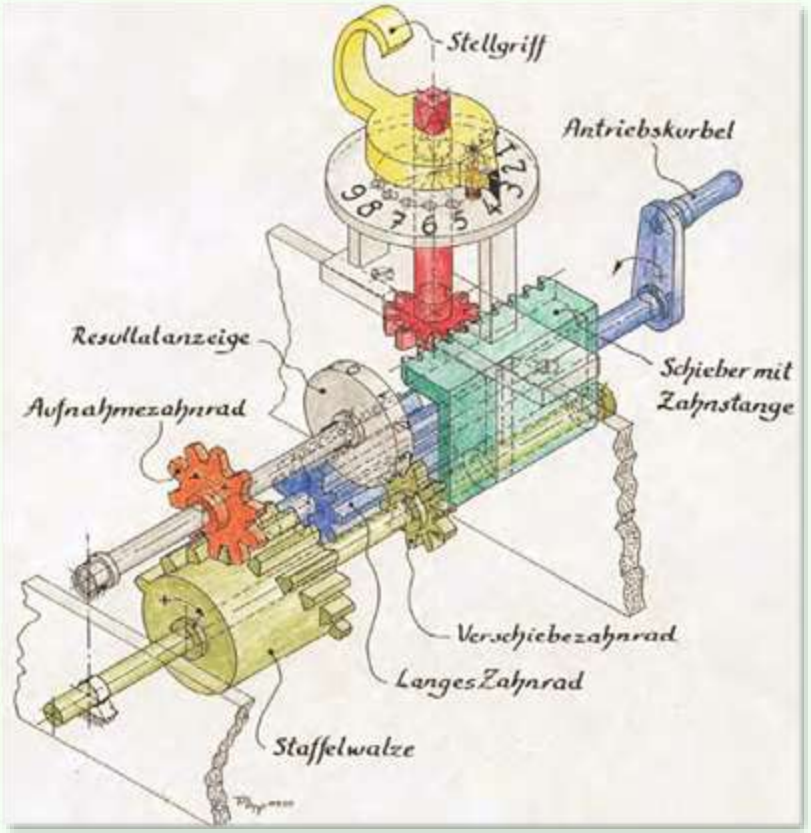
Skizze des 3-stelligen Holzmodells mit Staffelwalze („dentes variabilis“)



www.rechenhilfsmittel.de/staffelc.gif

Prinzip der Staffelwalze

einander durch je eine Kurbelumdrehung addiert, dann das gesamte Eingabewerk um eine Stelle verschoben und noch zwei Mal addiert. Im Allgemeinen entspricht die Anzahl der Additionen der Quersumme des Multiplikators; ein Umdrehungszähler mit Glocke spart einem das Mitzählen, wie oft man eine Stelle schon addiert hat.



Zeichnung: F.O. Kopp. Aus: Erwin Stein: Calculemus!

Das Hauptproblem stellte in mechanischer Hinsicht ein „durchklappernder“ indirekter Zehnerübertrag (wie etwa bei 999+1) dar. Die Fertigung und Justage der Räder für den Übertrag stellte besonders hohe Anforderungen; aus Briefen der Mechaniker an Leibniz ist zu entnehmen, dass damals über mehrere Jahre an dieser Sache gearbeitet wurde. Die Leibnizsche Maschine kann daher auch nur höchstens zwei Zehnerüberträge gleichzeitig ausführen; nicht abgeschlossene Zehnerüberträge werden aber durch besondere Scheiben an der hinteren Kante der Maschine angezeigt, sie müssen zum Abschluss der Rechenoperation dann in einem nachfolgenden Bedienvorgang berücksichtigt werden.

Ein kurzer Auszug aus dem Buch „Leibniz, Newton und die Erfindung der Zeit“ des Wissenschaftspublizisten Thomas de Padova (Piper, 2013):

Nach seiner Ankunft in Paris hat Leibniz etliche Salons aufgesucht, um die Hautevolee kennenzulernen, und mit Uhrmachern wie Handwerkern gesprochen, weil er den Bau einer Rechenmaschine voranbringen möchte. […]

Zuerst hatte Leibniz die Idee, die Zahlen Quantum für Quantum mithilfe einer Waage in die Maschine einzulesen. Ein anderer Entwurf sieht Zylinder vor, wie sie in Glockenspielen benutzt werden. Solche Zylinder sind mit Stiften bestückt, lassen Musikstücke erklingen und setzen Figuren in Bewegung. Die Anzahl der Stifte könnte auch für eine gewünschte Zahl stehen. Leibniz' Rechenautomat soll aber mit sehr großen Zahlen rechnen und würde daher viele Zylinder benötigen. In einem seiner Entwürfe sind es neun mal neun Zylinder in jeweils zwei Ausführungen, insgesamt also 162 Stück. Die Pariser Uhrmacher dürften ihm vor Augen geführt haben, dass dieses Modell nicht umsetzbar ist.

Neben der Stiftwalze sind weitere Informationsspeicher bekannt. Schon mittelalterliche Turmuhren, die mit einer Glocke verbunden waren, besaßen ein Stundenschlagwerk. Um die Uhrzeit über die jeweilige Zahl der Glockenschläge mitzuteilen, kerbten Uhrmacher das gewünschte Läutprogramm in eine rotierende Scheibe. Die automatisch ausgelösten Glockenschläge endeten genau dann, wenn der Sperrhebel in die dafür vorgesehene Kerbe fiel.

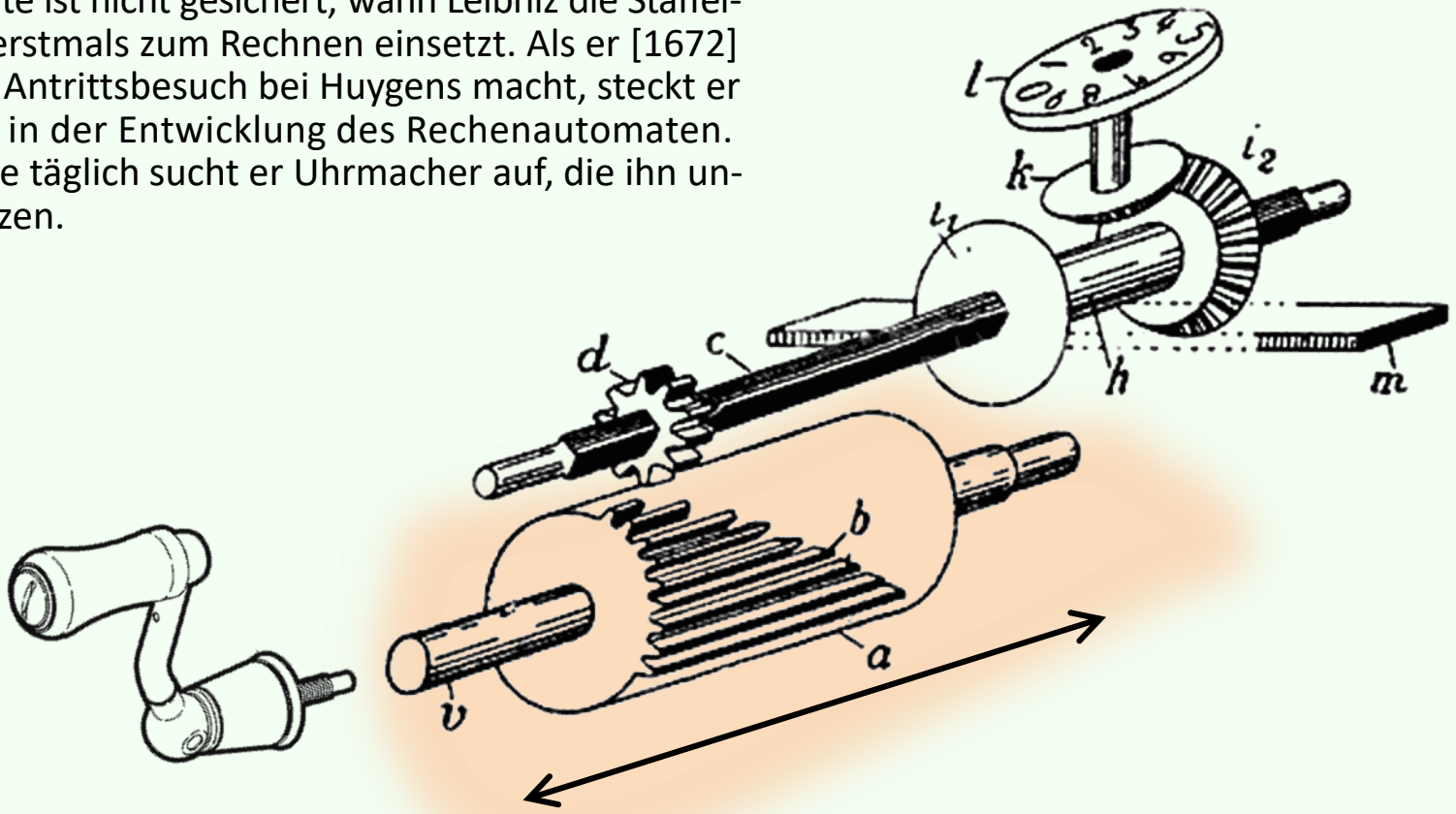
Im Gespräch mit den Pariser Handwerkern wird Leibniz klar, dass er sein Ziel am ehesten mit der herkömmlichen, in Uhren eingesetzten Technik erreichen kann, also mit Zahnrädern und Zahnstangen, Wellen und Handkurbel. Schließlich hat er eine ausgezeichnete Idee, wie sich alle Zahlen von Null bis Neun mit einem einzigen mechanischen Bauteil darstellen lassen:

Dreh- und Angelpunkt aller Barockmaschinen ist das Zahnrad. Stellen Sie sich ein Zahnrad mit neun Zähnen vor. Darauflegen Sie ein baugleiches Zahnrad, dem Sie einen Zahn wegnehmen. Dem nächsten Zahnrad, das Sie darüberlegen, fehlt ein weiterer Zahn, und so fort. Auf diese Weise entsteht aus den flachen Zahnrädern ein dreidimensionaler Zylinder mit ungleich langen Rippen, ähnlich einer Wendeltreppe: die Staffelwalze. Sie zählt zu Leibniz‘ bedeutendsten Erfindungen. Bis ins 20. Jahrhundert

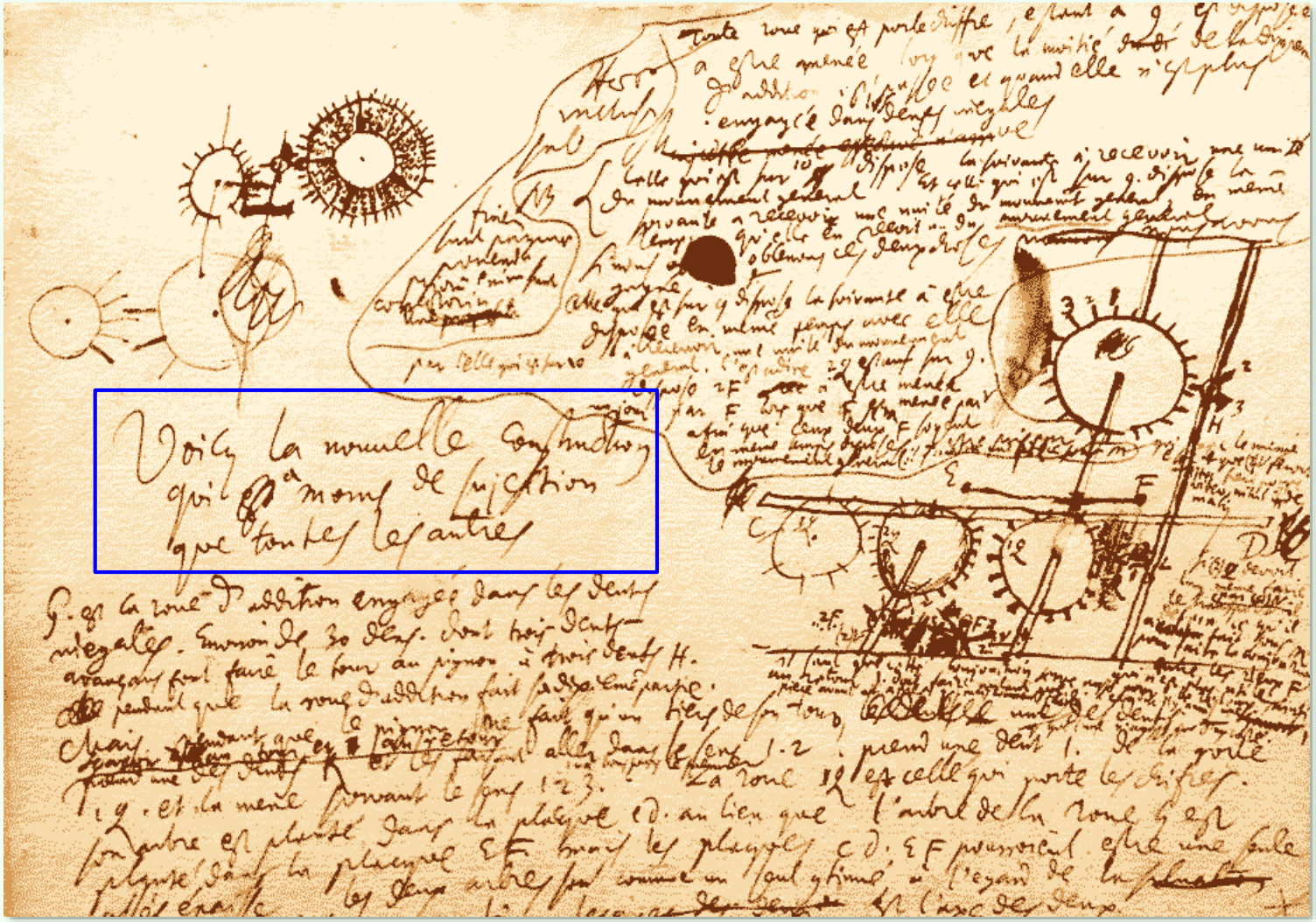
hinein bleibt sie neben dem Sprossenrad, das Leibniz ebenfalls benutzt, ein Herzstück mechanischer Rechenmaschinen.

Bei der Staffelwalze repräsentieren Zahnrippen die Zahlen. Welche dieser Zahlen abgegriffen wird, hängt davon ab, wie weit man die Walze in die Maschine hineinschiebt. Dreht sich zum Beispiel das Abgreifzahnrad nur über eine einzige Rippe, nämlich die längste, dann bewegt sich das damit verbundene Ziffernrad des Resultatwerks ebenfalls nur um einen Zahn weiter.

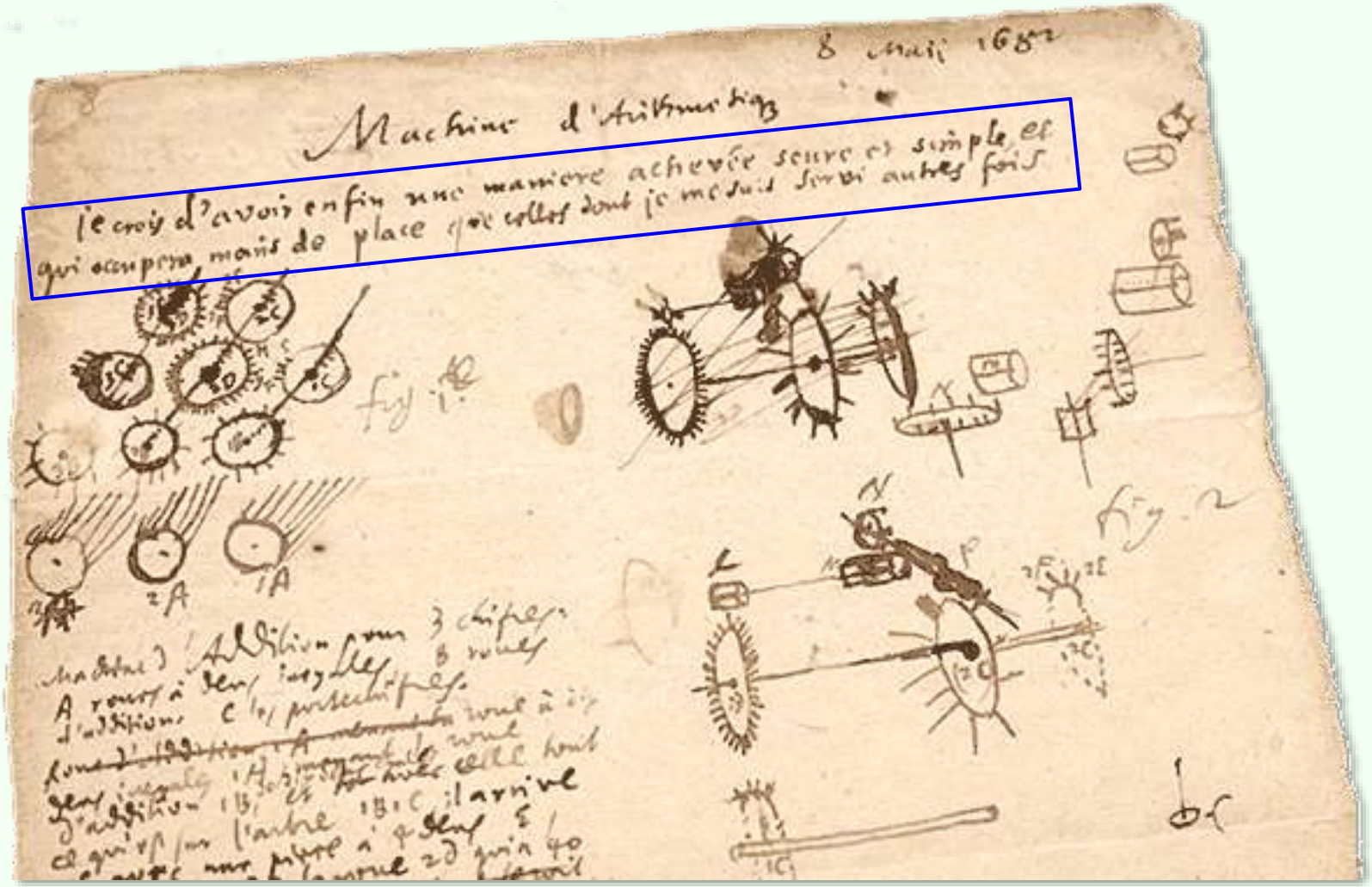
Bis heute ist nicht gesichert, wann Leibniz die Staffelwalze erstmals zum Rechnen einsetzt. Als er [1672] seinen Antrittsbesuch bei Huygens macht, steckt er mitten in der Entwicklung des Rechenautomaten. Beinahe täglich sucht er Uhrmacher auf, die ihn unterstützen.



Leibniz versucht immer wieder, seine Rechenmaschine zu verbessern. Er hält viele seiner Überlegungen skizzenhaft fest, wie z.B. hier: „Voicy la nouvelle construction qui a moins de friction que toutes les autres.“



http://blog.stephenwolfram.com/data/uploads/2013/05/16-large-a.png

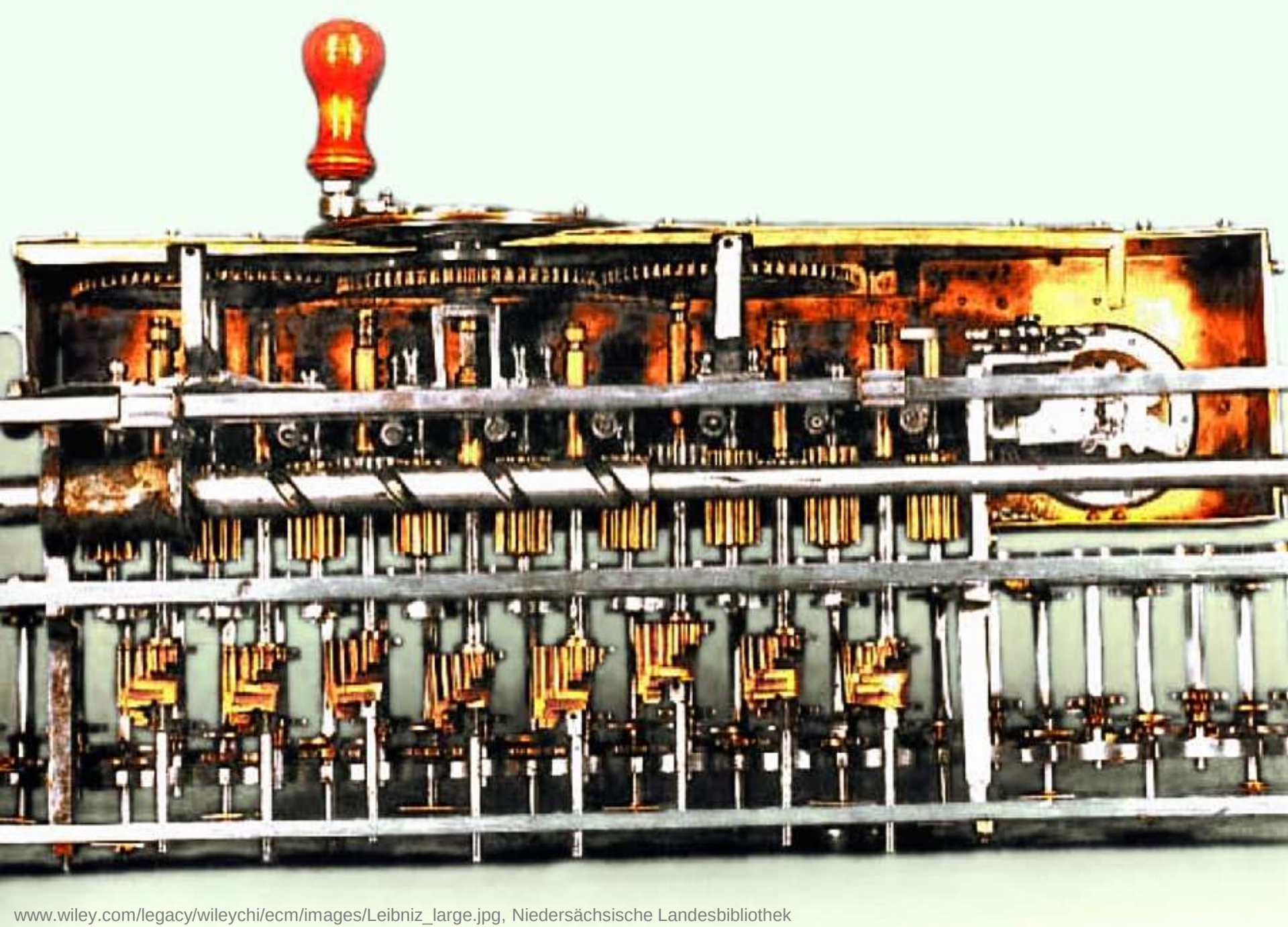


https://tubestatic.orf.at/static/images/site/tube/20161146/leibniz-faksimile_big.5539008.jpg

„Je crois d'avoir enfin une maniere achevée seure et simple, et qui occupera moins de place que celles dont je me suis servi autres fois", so Leibniz in dieser Notiz vom 8. Mai 1682 zu einer Konstruktionsverbesserung der Staffelwalzen mit axial verschiebbaren Zahnrädern und abnehmenden Zahnlängen für die Zahlen 1 bis 9. „Leibniz war ein Kopfmensch, und da er sehr klare Vorstellungen hatte, hielt er es für eine Kleinigkeit, seine kargen und ungenauen Skizzen in ein funktionierendes Gerät umzusetzen." [Klaus Badur]

Bild: Niedersächsische Landesbibliothek

Detailausschnitt (Umdrehungszählwerk) Bild: Niedersächsische Landesbibliothek



www.wiley.com/legacy/wileychi/ecm/images/Leibniz_large.jpg, Niedersächsische Landesbibliothek

Einige kurze Auszüge aus der lesenswerten Biographie „Der berühmte Herr Leibniz“ von Eike Christian Hirsch (C.H. Beck, Neuauflage 2016, pp. 525 - 527):

Noch ist die neue Maschine, an der seit drei Jahren gearbeitet wird, nicht weit gediehen. Mit Sorge und mit einigen Hoffnungen lässt Leibniz das kostbare Stück jetzt im Sommer des Jahres 1700 einpacken, damit es auf die Reise nach Helmstedt gehen kann. Es ist das neuere Exemplar, auf dem noch immer seine Hoffnungen ruhen. Angelegt ist es als eine grössere Maschine mit 16 Stellen im Ergebnis, und in ihr stecken einige neue Ideen. Sie ist zweieinhalb Ellen (78 cm) lang, aus Messing und Eisen und hat ein bewegliches Teil, den Schlitten. Nun soll alles weggegeben werden. Der Mechanicus Adam Scherp, ein Geselle für Turmuhren aus Linz, der wohl 1697 mit dem Neubau angefangen hatte, besass kein Talent dafür, arbeitete schludrig und ungenau und war auch nicht unter der Aufsicht des Sekretärs Balthasar Reimers zu Besserem anzuleiten gewesen. Mit dessen Tod war jetzt alles zum Erliegen gekommen.

Doch schon vor zwei Jahren (1698) hatte sich eine neue Hoffnung gezeigt, als sich ein wirklich kluger Kopf, Leibnizens Assistent Rudolf Christian Wagner, der Sache anzunehmen begann. Aber wie es mit guten Leuten ist, gerade (1700) war Wagner nach Helmstedt gegangen und sollte dort Professor der Mathematik werden. Jetzt aber, wie gesagt, wird diese Ruine unter des Erfinders Augen eingepackt, denn Wagner will sich ihrer in Helmstedt auf Dauer annehmen. Ja, er hat dort einen tüchtigen Mechaniker, einen ‹Opifex›, wie so jemand genannt wird, aufgetan, Levin Warnecke. So trennt Leibniz sich 1701 von dem Sorgenkind. In Gottes Namen, möge der Rechenmaschine in der Fremde eine bessere Zukunft beschieden sein! Die Werkstatt ist damit – unter neuer Leitung – nach Helmstedt verlegt. Und viele Jahre lang wird hier an der neuen Maschine gearbeitet werden. Es ist die, die heute in Hannover aufbewahrt wird, die einzig erhaltene. Weil Wagner so tüchtig ist und das Vertrauen von Leibniz hat, bekommt er auch noch die ältere nach Helmstedt geliefert, die eigentlich fertig sein sollte, und überprüft sie. Im Juli 1701 kann er nach Anweisung von Leibniz auf dieser älteren zu rechnen beginnen. Und er stellt immer wieder fest, sie ist weit besser gearbeitet, das neuere Modell aber ist besser erdacht, nur schrecklich angefertigt.

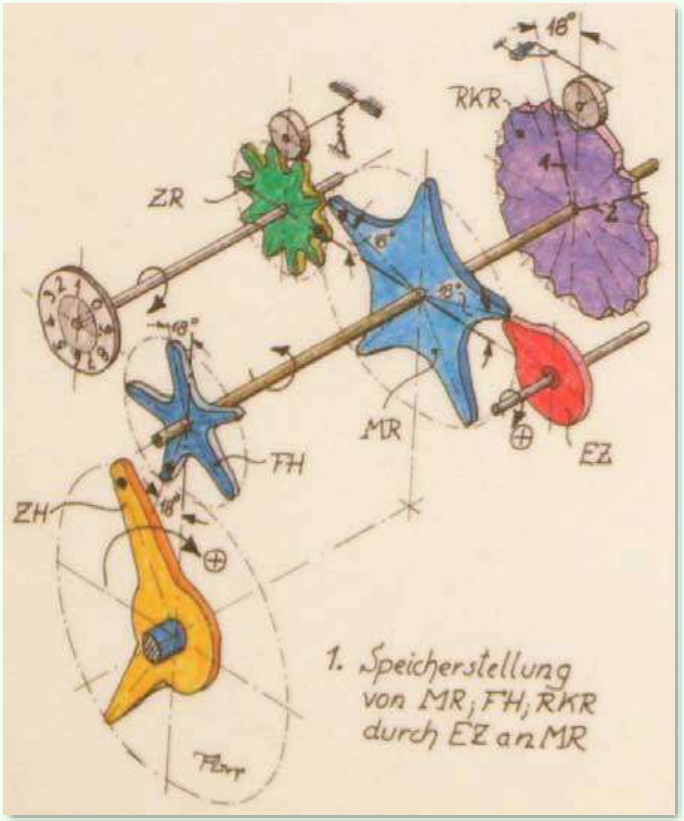
Auch der Opifex Levin Warnecke hat sich inzwischen die neue Maschine genau angesehen, er hat sie gut verstanden, aber er ist entsetzt und will die Kurbel, ein wichtiges Teil, das höchste Anforderungen zu erfüllen hat, lieber neu anfertigen. Leibniz lehnt das ab. Der Aufseher Professor Wagner stellt sich hinter seinen Techniker und klagt am 7. April 1701, diese jüngere Maschine sei so liederlich hergestellt, dass er «gewiss mit allen in diesen Dingen Kundigen beschwören möchte, dass diese Maschine bei einfacher Benutzung nicht einen Monat hätte überdauern können». Nichts an ihr ist im Lot und rechten Winkel, eine windigere Pfuscharbeit kaum denkbar. Der geplagte Levin Warnecke, der an seinem oft betrunkenen Bruder auch keine grosse Hilfe hatte, war genötigt, durch gesetzte Hammerschläge die Winkel zu richten, die Bohrungen mit Sorgfalt passend zu machen und ältere wieder zu ‹verbunzen›.

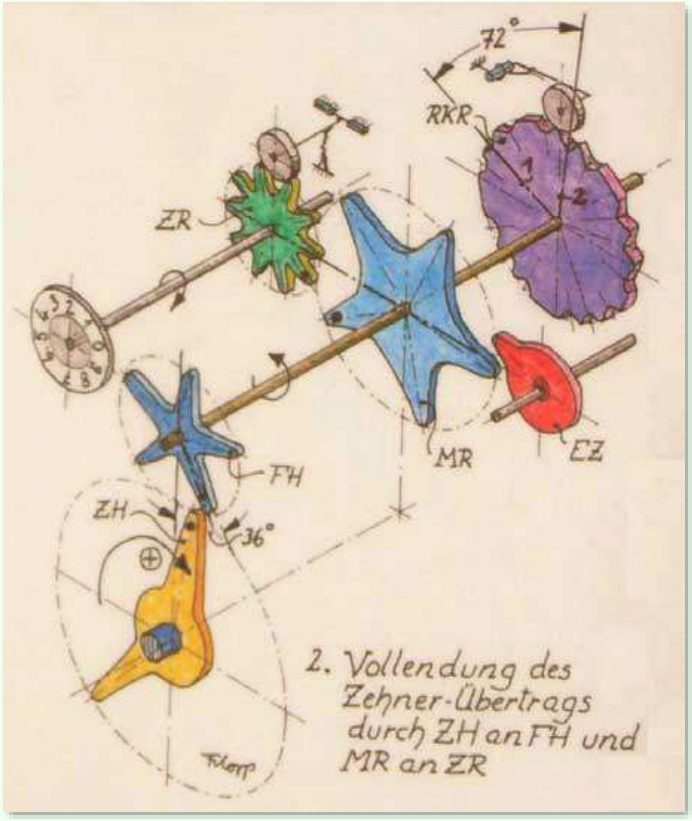
Nach drei Jahren konnte man mit ihr multiplizieren. Doch wenn man zum Dividieren die Kurbel rückwärts drehen wollte, hat es noch «durchaus nicht gehen wollen, sondern bald an diesem, bald an einem anderen gehangen und gefehlet», meldet Wagner. Meister Levin will immer noch lieber mit einem ganz neuen Modell von vorn anfangen und hätte mit seiner offenkundigen Geschicklichkeit und Sorgfalt gewiss etwas Besseres zustande gebracht. Im April 1704 klappt jedoch selbst bei diesem gestümperten Werk das Dividieren, nur die Zehnerübertragung will bei beiden Rechenarten nicht gut gehen. Das war schon immer die Schwachstelle, sogar beim Addieren wird allzu leicht aus 999 + 1 eine 910. Wagner meldet, das sei so, «weilen die einhorne in die fünfhorne gar zu knapp faßen». Gemeint sind Hebel mit einem Horn oder fünf Hörnern, die diese Zehnerübertragung bewerkstelligen sollten.

Im April 1705 kommt Leibniz selbst nach Helmstedt und sieht nach den Mängeln, im Herbst gibt er Anweisungen zu den Einhörnern und den Fünfhörnern. Bald darauf sind Leibniz' Maschinen vermutlich besonders funktionstüchtig gewesen. Und es war nun ausgerechnet die jüngere, die von Leibniz zwar neu durchdachte, aber doch schlecht gearbeitete Maschine, die exakt funktionierte. Rudolf Christian Wagner konnte Leibniz am 22. Januar 1706 melden: «Auf der Maschine haben wir folgendes Exempel gantz wohl vor und Rückwarts gemacht 12 405 897 x 96 878 532 = 1 201 865 089 503 204 kommet zu product.» Bei dieser Rechnung war die volle Kapazität der Maschine gefordert und sie muss mit all ihren Teilen funktionsfähig gewesen sein. Eine unglaubliche Leistung, auch wenn sie nicht für immer aufrecht zu erhalten war. □

Der Dezimalübertrag

Das Problem des durchklappernden Übertrags wird so angegangen, dass dieser zwar in einer einzigen Kurbeldrehung durchgeführt wird, allerdings bei den einzelnen Dezimalstellen leicht zeitversetzt und in zwei Phasen: In der ersten Phase dreht der Einzahn (EZ, rot) auf der Welle der n-ten Ziffer beim Übergang von 9 auf 10 das fünfzähnige Muldenrad (MR, blau) um 18° in eine „Speicherstellung“. Hierdurch gerät das auf gleicher Welle liegende um 18° verdreht positionierte Fünfhorn (FH, dunkelblau) in den Flugkreis des darunter liegenden Zweihorns (ZH, gelb). In der zweiten Phase bewegt das Zweihorn ZH mit ausreichender Kraft (weil es auf der Welle der Staffelwalze sitzt) das Fünfhorn FH um 54°. Um den gleichen Winkel wird somit auch das auf gleicher Welle befindliche Muldenrad MR gedreht, welches das Aufnahmezahnrad (ZR, grün) der n+1-ten Stelle um einen Zahn weiterdreht, so dass das Resultatrad dieser Stelle die nächsthöhere Ziffer anzeigt. Der diffizile Mechanismus war zwar (ab etwa 1686) ingeniös erdacht, hat aber anscheinend nie sehr lange funktioniert.





Bilder und Text basierend auf den GWDG-Nachrichten 7/2009, Göttingen, S. 17-18

1685 macht Leibniz einige interessante Anmerkungen zum praktischen Nutzen seiner Maschine; dazu nachfolgend einige Auszüge des (ursprünglich lateinischen) Textes:

„Wenn man die Maschine noch bewundernswerter machen wollte, dann kann bewirkt werden, dass die Drehung der Räder und das Vorrücken der Multiplikationsmaschine von Operation zu Operation nicht von Menschenhand zu geschehen brauchte, indem man die Sache von vornherein so einrichtet, dass bei der Maschine alles von selbst läuft. Aber dies würde die Maschine kostspieliger und komplexer machen und ihre Brauchbarkeit wohl nicht erhöhen. [...]

Wie bedeutend die Anwendung dieser Maschine zukünftig sein wird, ist hinlänglich klar, denn die Beseitigung jeglichen Fehlers und aller oder doch fast aller Mühe aus der Zahlenrechnung ist von grossem Nutzen sowohl im Staate, wie in der Wissenschaft. [...] Da jetzt an die Rechenmaschine die letzte Hand angelegt ist, so darf man annehmen, dass sie allen erwünscht sein wird, die mit Berechnungen zu tun haben, wozu bekanntlich Finanzverwalter, Buchhalter, Kaufleute, Feldmesser, Geographen, Seeleute, Astronomen gehören, und was von Fachleuten sonst der Mathematik bedarf. [...]

So können nun ohne grosse Mühe die alten geometrischen und astronomischen Tafeln verbessert und neue aufgestellt werden. [...] Auch der Fleiss der Astronomen wird sicherlich durch keine Mühsal stärker auf die Probe gestellt, als die des Rechnens. Dies hindert sie an der Aufstellung und Verbesserung von Tafeln, an der Anlage von Ephemeriden, am Ausarbeiten von Hypothesen und am Zusammenführen ihrer Beobachtungen. Denn es ist ausgezeichneter Männer unwürdig, ihre Zeit mit sklavischer Rechenarbeit zu verlieren, die mit Anwendung der Maschine ohne Besorgnis jedem Beliebigen übertragen werden könnte.“

Der letzte Satz nochmal im Original auf Latein: „Indignum enim est excellentium virorum horas servili calculandi labore perire; qui Machina adhibita vilissimo cuique secure transcribi posset.“ In etwas freierer Übersetzung: „Es ist unwürdig, die Zeit von hervorragenden Leuten mit knechtischen Rechenarbeiten zu verschwenden, weil bei Einsatz einer Maschine auch der Einfältigste die Ergebnisse sicher hinschreiben kann.“

Rechnen mit der lebendigen Rechenbanck...

Von programmgesteuerten Rechenmaschinen, also Computern, ahnte Leibniz noch nichts. Aber dass das automatische Rechnen beim Militär, in der Wirtschaft und bei den Banken relevant werden wird und ein grosses Rationalisierungspotential besitzt, das war Leibniz schon klar. Er schreibt aber auch zu seiner eigenen Maschine: *Non est facta pro his qui olera aut pisculos vendunt.*

...welches dann vor Rechen-Cammern, Contoirs, Meß=Kunst, Fortification, Schiffart, ja ganze Mathesin und Mechanick, auch Commercien, undt Financen, einen unglaublichen nuzen haben, die Menschliche Arbeit darinn auff die Helffte mindern, auch so gar unnöthige Menge der dazu brauchenden Personen und viele Gagen ersparen kan. Zu geschweigen wie allerhand Tabulae in obgedachten Scientien, einmahl vor allemahl zu erleichterung Menschlicher arbeit, dergestalt durch dazu bestelte Leüte auszurechnen, da sonsten bekand, daß in den Tabulis Logarithmorum, so ein herrliches werck, unterschiedene dazu besoldete Personen über 20 Jahr zubracht.

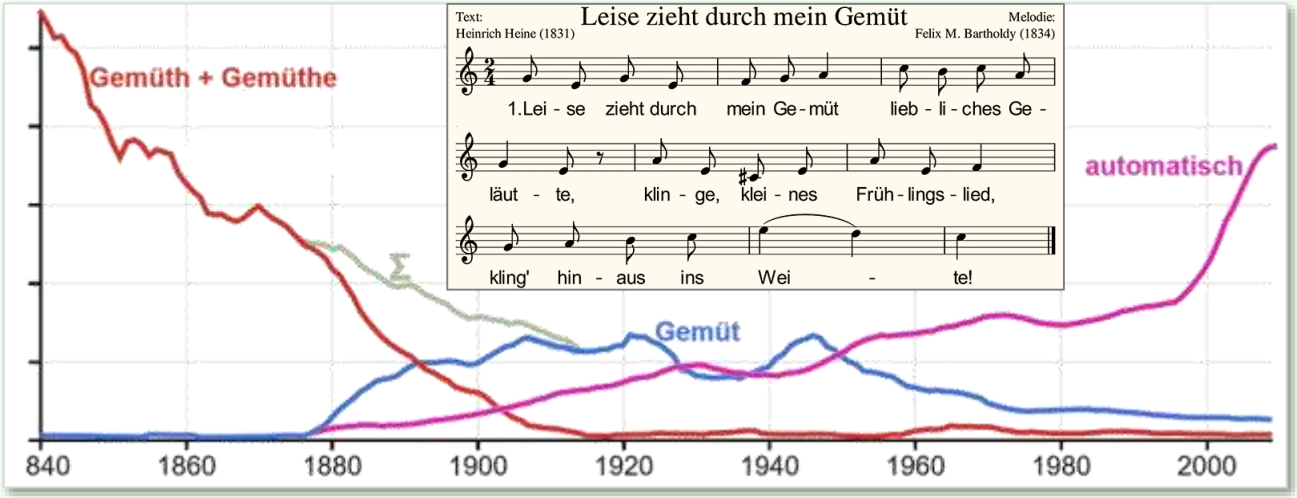
Ohne Arbeit des Gemüths

Eine der Subtilsten Inventionen so von Menschen gesehen worden, ist meine Machina Arithmetica so man in den beyden koniglichen Societäten zu London und Paris admiriret, da man doch nur die würckung in dem Schlechten Modell gesehen; wenn ich aber einmahl gelegenheit habe Handwercks leute zu halten, will ich deren etliche in Vollkommenheit vor großer Herren kammern und observatoria machen laßen, ein kind kann darauff die schwehrsten Exempel multipliciren und dividiren, und geschicht alles gleichsam in einem augenblick ohne arbeit des gemüths. Und große Zahlen werden eben sobald fertig als kleine. Ist treflich ganze tafeln auszurechnen, dienet aber sonderlich als ein ***Specimen der Menschlichen gemüthskraffte dadurch zu wege zu bringen daß eine Machina rechnen kan****, welches sonsten proprium hominis gehalten worden.*

Vom Gemüthe

Bei Leibniz ist oft vom „Gemüth“ und der menschlichen „Gemüthskraft“ die Rede; das Rechnen mit der Maschine solle z.B. ohne Arbeit und Mühe des Gemüths von sich gehen – wir würden heute eher von „geistiger Anstrengung“ sprechen oder lieber gleich sagen, dass der Vorgang „automatisch“ abläuft. Das Wort „automatisch“ ist allerdings erst ab dem späten 18. Jahrhundert gebräuchlich (gebildet entsprechend dem französischen „automatique“).

Bis zur Rechtschreibreform 1901 schrieb man noch „Gemüth“ oder „Gemüthe“, seither „Gemüt“. Der Begriff, der heute eher ungewöhnlich ist und oft nur noch in Redewendungen (z.B. „ein sonniges Gemüt haben“) auftaucht, bezeichnete die Gesamtheit der geistig-seelischen Kräfte sowie die inneren Empfindungen und Gedanken, was je nach Kontext auch durch Begriffe wie „Seele“, „Psyche“, „Geist“, „Wesensart“ oder „Denkvermögen“ umschrieben werden kann; letzterer erfasst am ehesten die hier von Leibniz intendierte Bedeutung, auch wenn sonst oft mit „Gemüt“ weniger das denkende Bewusstsein (*ratio*) als die „fühlende Seele“ (*anima*) gemeint ist.



Es gibt Dinge, die die Gemüter bewegen; die Gemüter können sich dann erregen oder erhitzen – aufgebrachte Gemüter sollte man besänftigen. Einige sollen ein zartes, kindliches Gemüt haben, andere ein Gemüt wie ein Veilchen oder sogar wie ein Fleischerhund. Es gibt einfache Gemüter und Gemütsmenschen, man hat einen Gemütszustand und befindet sich in einer Gemütslage. Man kann es sich gemütlich machen und sich etwas zu Gemüte führen; manches kann aber auch auf das Gemüt schlagen, wovon man gemütskrank werden kann.

Gänzlich ohne Arbeit des Gemüths und niemahlen fehlen:

Das Motiv für die mechanischen Rechenmaschinen war für Leibniz nicht so sehr die pure Beschleunigung des Rechnens (die Benutzung der Maschinen war ja auch noch relativ aufwändig), sondern lag eher philosophisch in der Entlastung von der Mühsamkeit des Denkens beziehungsweise in der Faszination der Automatisierung geistiger Tätigkeiten – „quo multiplicatio ac divisio fieri potest solo rotae cujusdam circumactu, nullo prorsus animi labore“ („wodurch Multiplikation und Division allein durch eine gewisse Drehbewegung, gänzlich ohne Geistesarbeit, stattfinden kann“). Er bemerkt allgemein, es sei unwürdig, die Zeit von hervorragenden Leuten mit knechtischen Rechenarbeiten zu verschwenden, weil mit dem Einsatz einer Maschine auch der Einfältigste die Ergebnisse sicher hinschreiben könne.

Später, nach Leibniz, kam (etwa bei Charles Babbage) noch ein weiteres wichtiges Motiv hinzu: Die Korrektheit bzw. deutlich geringere Fehlermöglichkeit beim mechanischen Rechnen gegenüber dem manuellen Rechnen. Jacob Leupold (1674–1727), ein in Leipzig wirkender „Mechanicus“, schreibt schon 1727 in seinem „Theatrum arithmetico-geometricum“, einer mehrbändigen enzyklopädischen Beschreibung des seinerzeitigen Standes der Ingenieurskunst und Technikwissenschaften: „Und obwohlen bey der Arithmetic vielen die Rechen-Machinen vor etwas überflüßiges und nicht allzunützliches scheinen dürfften, [...] so mögen selbige dargegen wohl erwegen, daß solche Machinen bey der Operation in Berechnung des Exempels niemahlen fehlen und folglich man wegen der gesuchten Zahl gewiß seyn kan, da man sonst immer, bis das Exempel probiret, in Zweiffel stehen muß, ob auch recht gerechnet.“ Generell bewundert Leupold die Rechenmaschine von Leibniz; er schreibt in seinem Traktat „Figur und kurze Beschreibung der curieusen Rechenmachine des Herrn von Leibnitz“: „Der Gebrauch ist demnach so bequem, sonderlich in der Multiplication und Division, und erfordert die Sache einerley Zeit, es mag auch die Zahl klein oder gross seyn, wenn sie nur nicht, wie schon gedacht, die Einrichtung der Machine übertrifft. Auch ist ganz klar, dass darzu kein Nachdenken erfordert werde, sondern mit Recht nur ein Kinderspiel zu nennen.“

Nett liest sich auch Leupolds etwas wehmütige Bemerkung zum unklaren zukünftigen Schicksal der Rechenmaschine von Leibniz, welcher erst wenige Jahre zuvor verstorben war: „... der Herr von Leibnitz so vielmahls mit denen Mechanicis unglücklich gewesen, daß solche Machine [...] niemahls nach seinen Angeben und Propos gerathen [...] Bald darauf aber ist der Herr von Leibniz verstorben, und weil die Erben kein Geld darzu mehr hergeben [...] ist das Werck nun gantz liegen blieben, wie weit es also damit kommen, kan eben nicht sagen."

Der fürstliche Hofprediger, Astronom und Mathematiker Gottfried Teuber aus Zeitz, der von Leibniz mit dem Bau eines neuen Modells der Maschine beauftragt war, schreibt ein Jahr nach dessen Tod in der Leipziger „Gelehrten Zeitung": „Die Machina Arithmetica, so allhier in Zeitz unter meiner Direction hat sollen verfertiget werden, ist curieux im Gebrauch. Alle Species Arithmeticae, können dadurch aufs geschwindeste und sicherte absolvirt werden. [...] Der Hr. von Leibnitz hat sehr viel Jahre mit grossem Kosten an dieser Machina arbeiten lassen; erst zu Hannover da sie aber nicht zu Stande gekommen: zum andern hier zu Zeitz, es ist aber eben so gangen, denn die Einrichtung eine grosse Accuratesse erfordert: die dritte wurde einem geschickte Mechanico allhier übergeben und committirte Er mir solche nach bestem Vermögen es möchte kosten, was es wolte, einrichten zu lassen: Allein dessen unverhoffter Todt hat gleich beym Anfange solche hinzulegen verursacht und ist immer Schade, daß eine so schöne Invention ins Stecken gerathen und wohl gäntzlich liegen bleiben muß."

Tatsächlich geriet die Leibniz'sche Rechenmaschine danach fast in Vergessenheit und blieb lange verschollen. Erst 1876 wurde sie auf dem Dachboden der Universität Göttingen wiederentdeckt. In den Jahren 1894 bis 1897 wurde sie auf Betreiben des Geodäten und Mathematikers Wilhelm Jordan durch den Rechenmaschinenfabrikanten Arthur Burkhardt in Glashütte in Sachsen untersucht, restauriert und ein Gutachten darüber erstellt. In diesem Gutachten von 1897 heisst es u.a.: „Nachdem der Verschlußtheil des Schaltwerks entfernt war, purzelten mir alle Stelltriebe

mit den Zahlenscheiben entgegen. Sie waren nur in dem Verschlußkasten und dem Rahmen gelagert. Ich versuchte sie sofort wieder an ihre Stellen zu bringen, was nur mit schwerer Mühe gelang [....] Traurig wackelnd, ohne jeden festen Halt, hingen die Schaltwalzen auf den Wellen, deren Vierecke [...] etwas zweifelhafter Natur waren.“

N.J. Lehmann erläutert 1967 in seiner Vorlesung „Maschinelle Rechentechnik“ die Von-Neumann-Architektur an der Kreidetafel

Nikolaus Joachim Lehmann (Bild: de.wikipedia.org)

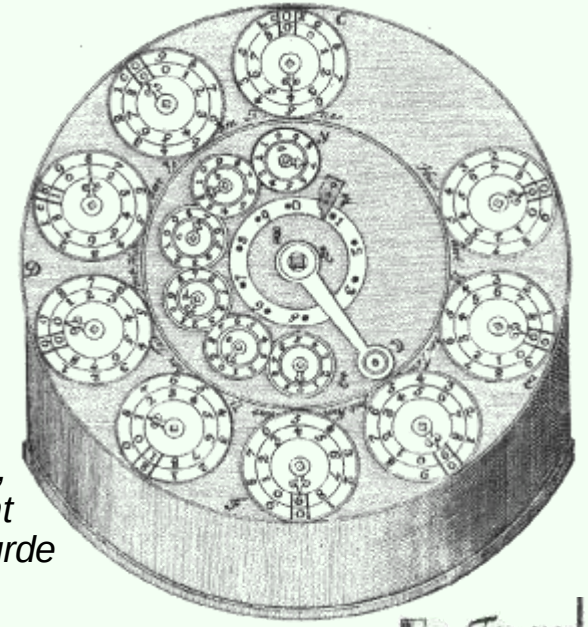
Joachim Lehmann (1921 – 1998, Informatikpionier der DDR, ab 1953 Professor an der TU Dresden), der Ende der 1980er-Jahre einen voll funktionsfähigen Nachbau der Leibniz-Maschine erstellte, merkt an, dass nicht nur die noch nicht ausreichend gute Fertigungstechnik der damaligen Zeit ursächlich für die Schwierigkeiten beim Bau der Maschine waren, sondern auch das fehlende Bewusstsein bei Leibniz für die Ingenieurskunst und die „mangelhafte Zusammenarbeit zwischen Erfinder und seinen Mechanikern. Hierzu ist der Briefwechsel zwischen Leibniz und seinen Werkmännern, die an seiner Maschine arbeiteten, sehr aufschlussreich. Zusammen mit den Manuskripten zum Gegenstand lassen sie die Gründe für die beim Aufbau ständig vorgekommenen Schwierigkeiten sehr deutlich erkennen. Infolge der oft nur brieflich und sogar über Mittelsmänner an seine Werkmänner geleiteten Anweisungen mussten diese eigentlich den gesamten Konstruktionsprozess allein auf sich gestellt bewältigen. Die seltenen (und nicht sehr klaren) Skizzen enthielten kaum Maßangaben und die Bearbeiter konnten daraus bestenfalls das Arbeitsprinzip entnehmen. Sie mussten danach selbst konstruieren, Materialfragen entscheiden und konnten erst dann fertigen. Von Leibniz wurde die Bedeutung der Ingenieurarbeit völlig unterschätzt.“

[N.J. Lehmann: Schickard und Leibniz als Erfinder von Rechenmaschinen, zweites Tübinger Schickard-Symposion, 1992]

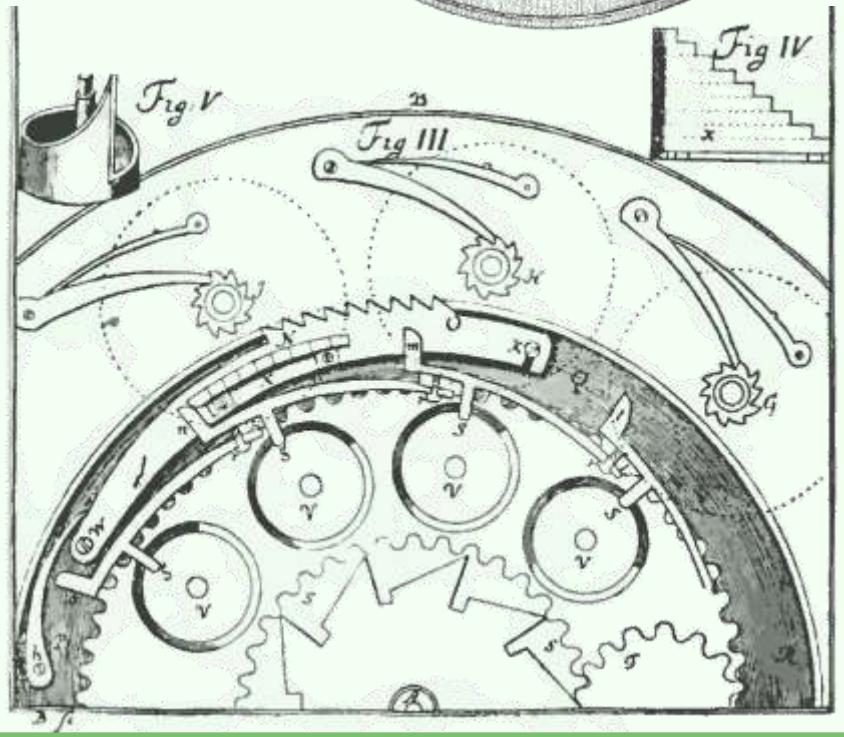
Mechanische Rechenmaschinen nach Leibniz

Nach Leibniz‘ Tod versuchten Zeitgenossen, aufbauend auf seiner Arbeit, ähnliche Rechenmaschinen zu bauen. So auch der oben erwähnte Mechanicus Jacob Leupold, der in Leipzig eine „Mechanische Fabrique“ führte und allerlei mechanische Apparate wie Luftpumpen, Dampfmaschinen, Feuerspritzen, Messinstrumente etc. erfand und fertigte. Er studierte Leibniz‘ Rechenmaschine genau und machte sich dann ans Werk, eine eigene „Leupoldische curieuse und ganz neue Rechen-Machine“ zu konstruieren:

„Nachdem ich vor mehr als zwantzig Jahren gelesen, daß man Rechen-Machinen erfunden, so ist mir gleichfals der Appetit hierzu ankommen, solche nicht allein zu sehen, sondern auch wohl selbst zu erfinden. […] habe zu inventiren angefangen, und nach und nach in die vier bis fünff Arthen heraus gebracht, die ich auch so weit ins Werck gerichtet, daß ich deren Effect unterschiedlichen Freunden zeigen konte. Wie nun aber bey der ersten sowohl als bey der andern und dritten Verfertigung mir alsdenn immer wieder etwas bessers und compendieusers eingefallen […].“



1727: Gesamtsicht und Detail der Maschine von Jacob Leupold, die jedoch nicht fertiggestellt wurde



Im Laufe des 18. Jahrhunderts konstruierten einige Gerätebauer, Mechaniker und Uhrmacher wie Johann Helfrich Müller (1746 – 1830), Anton Braun (1686 – 1728), Charles Stanhope (1753 – 1816) und Philipp Matthäus Hahn (1739–1790) „alltagstaugliche“ Vierspezies-Rechenmaschinen, die oft auf Ideen von Leibniz und Leupold beruhten. Allerdings waren diese Maschinen echte Meisterstücke und noch zu diffizil, als dass sie in einer Manufaktur durch Handwerksgesellen serienmässig hergestellt werden konnten.

Die „Rechnungmaschine“ von P. M. Hahn mit 11 Stellen, Durchmesser 26.5 cm, Höhe 11 cm; die Staffelwalzen stehen senkrecht im Kreis [Bild: Wikipedia]

Bekannt wurde vor allem der Württembergische „Mechaniker-Pfarrer“ Philipp Matthäus Hahn, der für seine Uhrmacherwerkstatt Brüder und Söhne anstellte und neben Rechenmaschinen auch aufsehenerregende „Himmelsmaschinen“ baute, die den Gang der Erde, der Sonne, der Planeten und der Sterne simulierten. Der Schriftsteller Heinrich Sander (1754 – 1782) berichtet von einer Besichtigung der Rechenmaschine: „Nachmittags war ich erst in Kornwestheim beim Herrn Pfarrer Hahn, seine Rechenmaschine und sein Sonnensystem zu besehen. Er hat eine artige kleine Frau, die ohne Prätension den Fremden alles zeigt, und die Kunstnamen wohl inne hat. Ins Innere des Kästchens läst er nicht sehen; es ist voller Räder. Oben sieht man nichts als emaillirte Zifferblätter. Man dreht eine Kurbel herum, wie an der Kaffeemühle. Die Frau machte Proben von allen 5. Rechnungsarten, wie ich sie ihr aufgab.“

Im August 1774 kam sogar der bekannte Zürcher Pfarrer und Autor Johann Caspar Lavater (1741 – 1801) zu Besuch zu Philipp Matthäus Hahn nach Württemberg und liess sich dessen mechanische Erfindungen zeigen. Vor allem von der Rechenmaschine war er äusserst beeindruckt, wusste er doch von den jahrelangen Mühen von Leibniz um eine praxistaugliche Maschine. Er schrieb in seinem Tagebuch: „Wir besahen die unbegreiflich wunderbare und unbegreiflich simple Rechnungsmaschine, [...] worauf alle erdenklichen Rechnungen gemacht werden können.“ Zu einem anderen Besuch notierte er: „Wir [...] besahen die astronomische Wunderuhr, an der er arbeitete. [...] Er zeigte uns [...] viel von seiner Rechnungsmaschine, die geht auf tausend Trillionen, worauf sich alle mögliche Fälle mit allen möglichen Brüchen berechnen laßen. [...] Ist Leibnitzens zehnjährlichen Bemühungen unmögliche Maschine ins Werk gesetzt und zehnmal vollständiger.“

Detail der Rechenmaschine von P. M. Hahn

www.weltderfertigung.de/archiv/jahrgang-2014/ausgabe-august-2014/758183a35d117b8a0/l001.jpg

Eine Art Serienfertigung von Rechenmaschinen fand erst ab ca. 1870 mit dem „Arithmomètre“ von Charles Xavier Thomas (1785–1870) aus Colmar statt; ca. 100 dieser Geräte wurden jährlich gefertigt. Dabei verbesserte und veränderte Thomas seine Maschine laufend. Tatsächlich erhielt Thomas sein Patent auf die Maschine bereits 1820, und die frühen Modelle waren vermutlich kaum besser für die Praxis geeignet als die oben erwähnten Maschinen des 18. Jahrhunderts. (In der Zeitschrift „English mechanic and world of science“ hiess es 1889 z.B.: „I have found that with hard wear the French machine soon gets out of order, owing, no doubt, to the parts being made of soft yellow brass and not being cut out of the solid.“) Allerdings nahm durch die fortschreitende Industrialisierung der Bedarf an „Rechenleistung“ durch menschliche Rechner in einem Masse zu, dass sich die Unterstützung durch Rechenmaschinen, selbst wenn diese unzulänglich waren, in ökonomischer Hinsicht „rechnete“. Sodann kam Thomas zugute, dass nach und nach eine Nutzergemeinde entstand, die laufend Hinweise zu Verbesserung seiner Maschinen lieferte.

Arithmomètre von Charles Xavier Thomas [Bild: www.arithmometre.org]

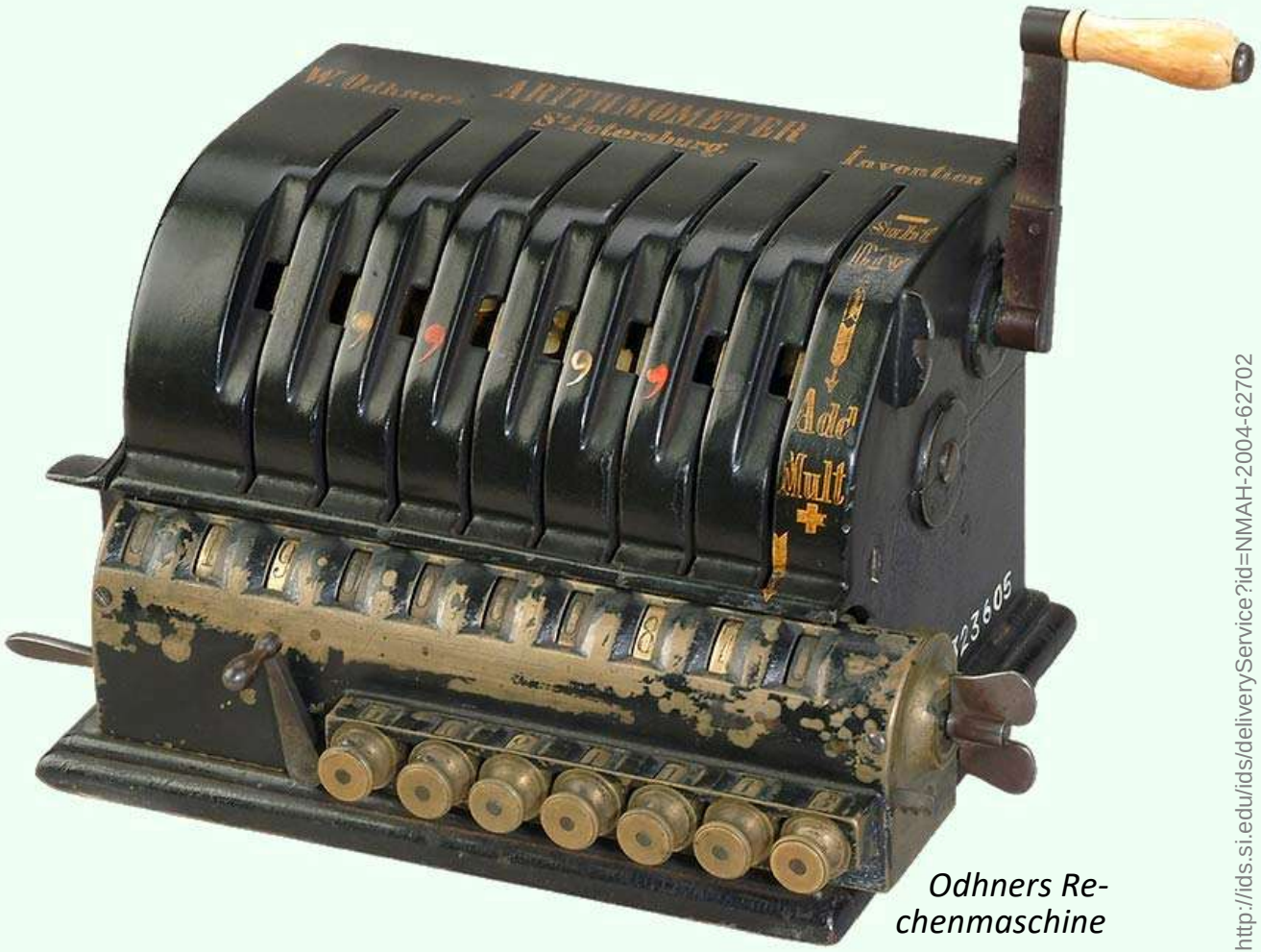
Wie die Brauchbarkeit der seinerzeitigen Rechenmaschinen und der Wunsch nach besseren Möglichkeiten 1872 eingeschätzt wurde, wird durch eine Notiz deutlich, die in „The Manufacturer and Builder“ [Vol. 11, p. 264] erschien (dabei entsprechen 5 $ im Jahr 1872 heute gut 100 $): „If a reliable calculating machine could be manufactured to retail at a low price, say five dollars, with which addition, subtraction, multiplication, and division could be done, it would no doubt find a ready-sale, as all go-ahead business men want such a time-saver. In all wholesale establishments; in banks, government offices, commercial houses, publishing companies, etc., an easily operated calculating machine, simple in construction, reliable in operation, not easily deranged, and substantially made, would be a great desideratum.

In looking over the models of calculating machines in the Patent Office last month, we saw none that answered the above description. Some of them were so complicated that it would take an engineer to run them, and a watch-maker to keep them in order, while others were evidently designed by men who knew nothing, practically, of the working of machinery; they having employed small wooden cams operating without friction rollers on sliding rods which would always stick fast; or, in other cases, strings were used to pull on wheels weighted on one side, and expected to return to their proper position by the action of gravitation, but which they never did, owing to friction, for which the inventor had made no allowance. Finally, every machine was out of order, and gave arithmetical results that would bankrupt the most successful business man in two turns of the handle."

Die Situation änderte sich bald darauf durch die Rechenmaschine des schwedischen Mechanikers Willgodt Odhner (1845 – 1905), der in St. Petersburg in der Maschinenfabrik von Ludvig Nobel (dem älteren Bruder von Alfred Nobel) angestellt war und dort Thomas-Arithmometer reparierte. Dies brachte ihn um 1871 auf die Idee einer besseren Rechenmaschine. In der St. Petersburger Zeitung erschien im September 1875 eine erste Notiz („Eine neue Rechenmaschine") zu seinen Bemühungen: „Wir hatten Gelegenheit, eine Rechenmaschine ganz neuer Konstruktion kennen zu lernen und uns von ihrer praktischen Anwendbarkeit zu überzeugen. Die uns von ihrem Erfinder vorgestellte Maschine ist ein elegant gearbeitetes Kästchen von dem Umfang eines kleinen Cigarrenkistchens. Auf den Mechanismus können wir nicht wohl eingehen, wir dürfen aber konstatiren, dass in unserer Gegenwart Exempel aus allen vier Species mit grosser Schnelligkeit und Genauigkeit ausgeführt wurden. Der Erfinder ist jetzt mit Herstellung einer Maschine beschäftigt, die alle möglichen Berechnungen in den Grenzen von 999 999 999 auszuführen im Stande sein wird."

Die Entwicklungsarbeit von Odhner dauerte zwar noch viele Jahre, zahlte sich aber letztendlich aus. Die Serienproduktion begann 1890. Im Vorwort zur Gebrauchsanweisung seiner Maschine schreibt Odhner: „Das Bedürfniss, nach einem einfachen, sicheren und billigen Apparat, mit Hülfe dessen unsere Zahlenrechnungen ausgeführt und controllirt werden können, ist ein längst

empfundenes. Viele, teils für allgemeine, teils für spezielle Rechnungszwecke eingerichtete Apparate wurden bereits hergestellt. Von allen diesen Maschinen jedoch fand nur eine, der von dem Elsässer Thomas 1820 construirte sogenannte „Arithmometer“, im praktischen Rechenwesen Anwendung. Indess blieb, der complicirten Construction und des hohen Preises wegen, seine Verbreitung eine sehr beschränkte.



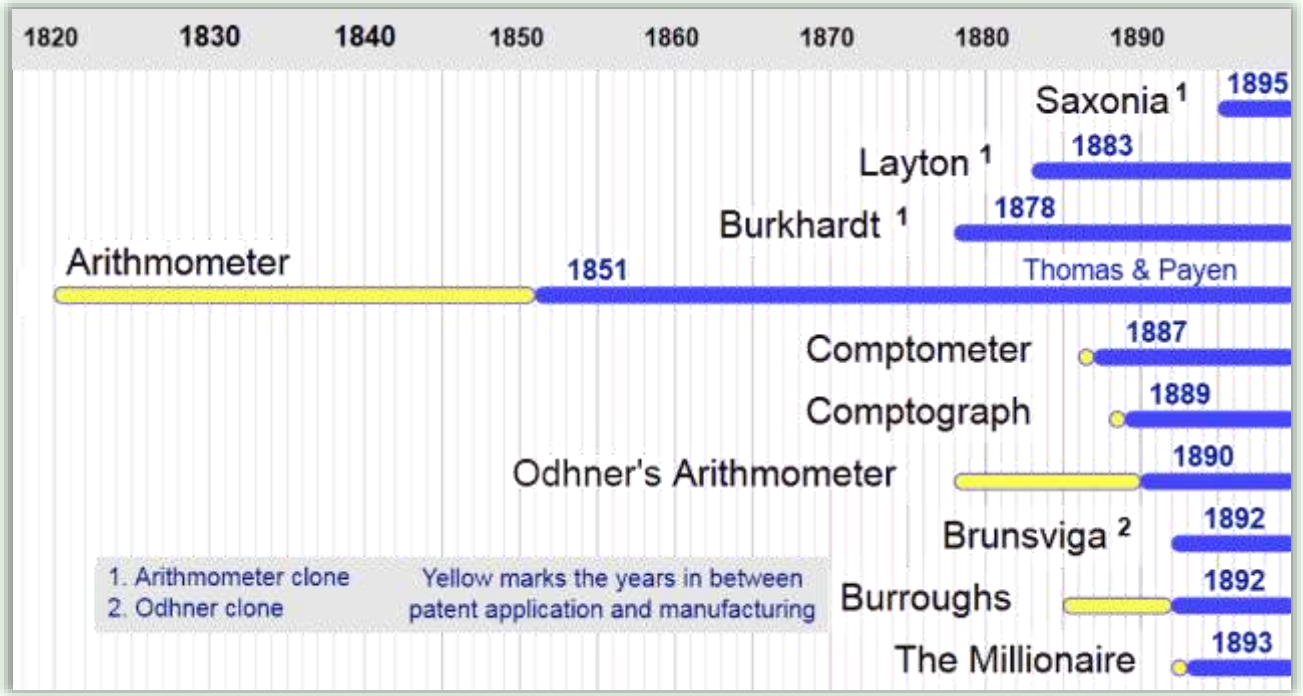
Odhners Rechenmaschine

http://ids.si.edu/ids/deliveryService?id=NMAH-2004-62702

Nach 15-jähriger Arbeit und stetigen Verbesserungen, gelang es mir endlich einen Apparat herzustellen, der seinen Vorgängern gegenüber sehr wesentliche Vorzüge besitzt, die mich hoffen lassen, dass er allen gerechten Forderungen genügen und mit der Zeit in keinem grösseren Geschäfte fehlen wird.“

Odhners Rechenmaschine wurde ein Erfolg. Etwa 23000 Exemplare wurden hergestellt, bis die Fabrik nach der Russischen Revolution 1918 schliessen musste. Die Maschine wurde auch an

andere Hersteller lizenziert und unter diversen Markennamen fabriziert sowie teilweise unabhängig weiterentwickelt. In Braunschweig wurde sie etwa unter der Bezeichnung „Brunsviga“ hergestellt. Die Maschinen galten als sehr robust und waren bis in die 1980er-Jahre in Gebrauch. Bis 1960 wurden weltweit ca. eine Million Exemplare gefertigt.



https://commons.wikimedia.org/wiki/File:DesktopMechanicalCalculators_inProduction_intheXIXCentury.svg

Die Abbildung zeigt die mechanischen Rechenmaschinen, die im 19. Jh. produziert wurden. Nachdem ein Markt dafür etabliert war, war es für andere Erfinder (mit weiteren Verbesserungen hinsichtlich „user experience“, Robustheit und Eignung für die Serienfertigung) leichter, zu reüssieren; ab etwa 1890 boomte das Rechenmaschinengeschäft regelrecht.

Die Zielgruppe der Hersteller waren grosse Wirtschaftsunternehmen, darunter vor allem Banken und Versicherungen. Wissenschaftler (in rechenintensiven Bereichen wie Astronomie, Meteorologie, Aerodynamik, Spektroskopie, Statistik etc.) nutzten die Maschinen auch, allerdings fehlten bei den mechanischen Geräten noch die für wissenschaftliche Zwecke wichtigen Winkelfunktionen und Operationen wie Potenzieren, Wurzelziehen etc. – daher waren als Rechenhilfsmittel noch bis zum Aufkommen der elektronischen Tisch- und Taschenrechnern in den 1970er–Jahren zusätzlich auch Tafeln für Logarithmen und Winkelfunktionen sowie Rechenstäbe in Gebrauch.

Leibniz' „Machina Deciphratoria“

Leibniz entwarf auch eine „Machina Deciphratoria“ zum Ver- und Entschlüsseln von Texten. Sie automatisiert die seinerzeit bereits bekannte Methode der polyalphabetischen Substitution, die allerdings für praktische Zwecke bei manueller Anwendung zu kompliziert und fehleranfällig war.

„Damit nahm er um reichlich 200 Jahre das Prinzip der Rotor-Schlüsselmaschine vorweg, nach dem die erste Generation der mechanischen Chiffriermaschinen (ab 1918) funktionierte. Wie die berühmte Enigma hatte Leibnizens Maschine eine Tastatur zur Eingabe von Buchstaben, die zugleich das Chiffrieralphabet weiterschaltete, und eine Anzeige für das Ergebnis. Allerdings war, der Zeit entsprechend, die Tastatur nicht von der Schreibmaschine inspiriert, sondern vom Klavier, und die Anzeige nicht elektrisch, sondern mechanisch. Nicht eine Elektronik wechselte für den jeweils nächsten Buchstaben in möglichst undurchschaubarer Weise die Verschlüsselung, sondern die Staffelwalze, die Leibniz bereits für seine Rechenmaschine verwendet hatte.“ [Nicholas Rescher]

Spektrum der Wissenschaft 9/16

Im Unterschied zu vielen seiner anderer Erfindungen sprach Leibniz mit fast niemandem über die Machina Deciphratoria; er hoffte auf potente Geldgeber bei Fürsten und Königen. Als ihm dann nach längeren Bemühungen 1688 endlich Audienz bei Kaiser Leopold I. in Wien gewährt wurde, bereitete

Leibniz das Treffen mit dem Monarchen eingehend vor; sein Nachlass enthält fünf Redemanuskripte unterschiedlicher Länge dazu. In ihnen preist er auch seine Verschlüsselungsmaschine an – z.B. so: „Damit ein potentat mit vielen ministris, in unterschiedlichen ziphern gleich correspondiren, und ohne einige muhe entweder die zipher die er schreiben will, und den verstand deßen so ihm in zipher zugeschickt wird gleichsam wie auff einem musicalischen instrument oder clavicordio greiffen könne, also dass es gleich mit berührung der clavir darstehe, und nur abcopiret werden dürffe.“

In einem weiteren Entwurf seiner Rede betont er neben der „usability“ noch einige andere Aspekte, wie z.B. die Sicherheit und die Portabilität seines Gerätes: „Ist eine kleine Machinula die leicht bey sich zu fuhren. Darauff kan ein großer herr viele fast unauflößliche Ciphern zugleich haben, und mit vielen Ministris correspondiren; weilen aber sowohl die stellung in Ziphern als das deciphriren mühsam, so bestehet die facilitat darinn, daß man die gegebene Ziphern oder buchstaben nur greiffen darff als wenn man auff einem clavicordio oder Instrument spielte, so kommen die begehrten augenblicklich herauß und stehen da; durffen denn nur abgeschrieben werden.“

Wie schon bei der Machina Arithmetica stellt auch die Machina Deciphratoria einen bedeutenden intellektuellen und praktischen Qualitätssprung dar: Zwar waren diverse Hilfsmittel (Chiffrierscheiben, Codebücher und -tabellen, Schablonen etc.) bei den seinerzeitigen Kryptographen in Gebrauch, doch war dies alles manuell anzuwenden. Erst Leibniz‘ Maschine

www.heise.de/ct/ausgabe/2015-4-Neues-zu-alten-Rechnern-und-1000-Jahre-Strassburger-Muenster-2520970.html

mechanisierte und automatisierte diesen Prozess, so dass er „kinderleicht“ und (zumindest prinzipiell) in hoher Geschwindigkeit angewendet werden konnte.

Anders als im zweiten Weltkrieg bei der Enigma und ähnlichen Chiffriermaschinen mit ihren computerisierten Gegenspielern „Turing-Bombe“ sowie „Colossus“ hätten seinerzeitige Codeknacker ohne Computer und bei den damals geringen Datenmengen wohl keine Chance gehabt. Dennoch wurde die Machina Deciphratoria zu Zeiten von Leibniz nicht gebaut – die damaligen Herrscher kommunizierten zwar mittels chiffrierter Nachrichten mit ihren Gesandten und Vertrauten, verliessen sich aber (oft zu Unrecht) auf die Kunst ihre Geheimkabinette und sahen daher in der Leibnizschen Maschine, die ein prinzipiell stärkeres Verschlüsselungsverfahren implementierte, keinen Zusatznutzen.

In fünfjähriger Arbeit wurden 2012 nach den alten Konstruktionszeichnungen zwei funktionierende Exemplare der Machina Deciphratoria von Klaus Badur und Gerald Rottstedt fertiggestellt.

Leibniz: So viele Einfälle!

„Il me vient quelques fois tant de pensées le matin dans une heure pendant que je suis encore au lit, que j'ay besoin d'employer toute la matinée et par fois toute la journée et au de là, pour les mettre distinctement par écrit."

„In Mathematicis und Mechanicis habe ich vermittelst artis combinatoriae einige dinge gefunden die in praxi vitae von nicht geringer importanz zu achten, und erstlich in Arithmeticis eine Machine, so ich eine lebendige Rechenbanck nenne, dieweil dadurch zuwege gebracht wird, daß alle zahlen sich selbst rechen, addiren subtrahiren multipliciren dividiren, ohne einige Mühe des Gemüths – nullo prorsus animi labore."

Leibniz's ingenious machines were not just developed for their quite obvious utility in their own right. They were integral parts of a larger philosophical program that implemented a deep-rooted and far ranging framework of thought, providing a vivid illustration of Leibniz's amazing capacity to give a concrete embodiment to his abstract reflections. -- Nicholas Rescher

Leibniz: So viele Erfindungen!

Verzeiht! Es ist ein groß Ergötzen,
sich in den Geist der Zeiten zu versetzen;
zu schauen, wie vor uns ein weiser Mann gedacht,
und wie wir's dann zuletzt so herrlich weit gebracht!

[Goethe, Faust, 1. Teil]

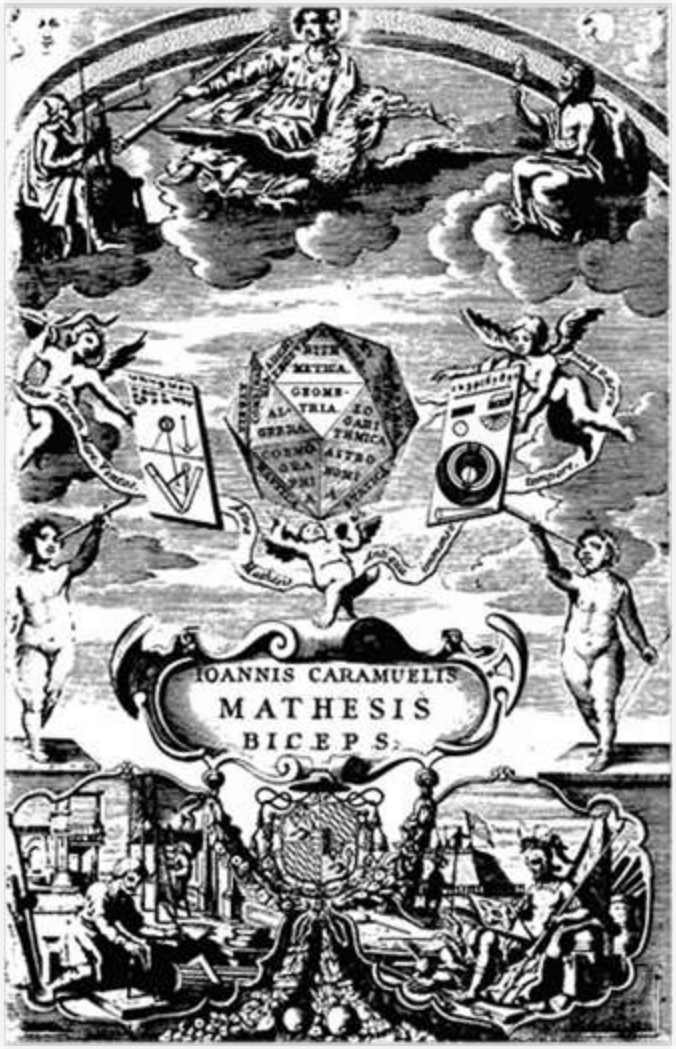
...meine Erfindung umfasst den Gebrauch der gesamten Vernunft, einen Richter für alle Streitfälle, einen Erklärer der Begriffe, eine Waage für die Wahrscheinlichkeiten, einen Kompass, der uns über den Ozean der Erfahrungen leitet, ein Inventar der Dinge, eine Tabelle der Gedanken, ein Mikroskop zum Erforschen der vorliegenden Dinge, ein Teleskop zum Erraten der fernen, einen generellen Calculus, eine unschädliche Magie, eine nicht-chimärische Kabbala, eine Schrift, die jedermann in seiner Sprache liest; und sogar eine Sprache, die man in nur wenigen Wochen erlernen kann und die bald in der ganzen Welt Geltung haben wird. (Leibniz im April 1679 in einem Brief an seinen Dienstherrn Herzog Johann Friedrich von Braunschweig mit der Bitte um eine Leibrente.)

Leibniz' Allegorie der Schöpfung

He was the first to emphasize the creative combinatorial potential of the 0 and 1 bit, and how everything can be built up from this one elemental choice, from these two elemental possibilities. So, perhaps not entirely seriously, I should propose changing the name of the unit of information from the bit to the leibniz! – Gregory Chaitin

Leibniz war so stolz auf seine Erfindung des Dualsystems, dass er eine Zeichnung für eine Medaille entwarf, auf der das Dualsystem dargestellt wurde: „Bild der Schöpfung, erdacht von Gottfried Wilhelm Leibniz im Jahre 1697" (Text im unteren Kreisabschnitt). In der Mitte befindet sich auf einer Tafel die Veranschaulichung des binären Zahlensystems. Links von der Tafel ist die Addition von zwei binären Zahlen und rechts von der Tafel die Multiplikation von zwei binären Zahlen dargestellt. Im Schriftzug oben steht „Omnibus ex nihilo ducendis sufficit unum" („Um alles aus dem Nichts herzuleiten genügt eins").

Dualzahlen aber auch schon vor Leibniz!



Bereits 1670, also noch vor Leibniz, erwähnt der gelehrte spanische Zisterziensermönch und spätere Bischof Juan Caramuel y Lobkowitz (Ioannis Caramuelis) in seinem Buch *Mathesis biceps* (Zweiköpfige Mathematik – zweiköpfig, weil neben der damals „alten“ Mathematik auch neue mathematische Aspekte behandelt werden) Zahlendarstellungen zu den Basen 2 bis 10, 12 sowie 60, wobei sich gewisse Basen durch eine besondere Natürlichkeit auszeichnen sollen (wie z.B. die 2 wegen den Oktaven der Musik oder die 3 aufgrund der heiligen Dreifaltigkeit); allerdings ohne auf die damit zusammenhängenden mathematischen Aspekte einzugehen.

ARTICVLVS I.

De Binariâ Arithmeticâ.

Num. III.

Vas Vnitates numerat: & posteà duos Binarios: & posteà duos Binariorum Binarios; & sic progreditur in infinitum. Ideò nominatur *Binaria*, quia per binas Vnitates, binos Binarios, binos Binariorum Binarios, &c. suas periodos absolvit. Puta, si sic procederet

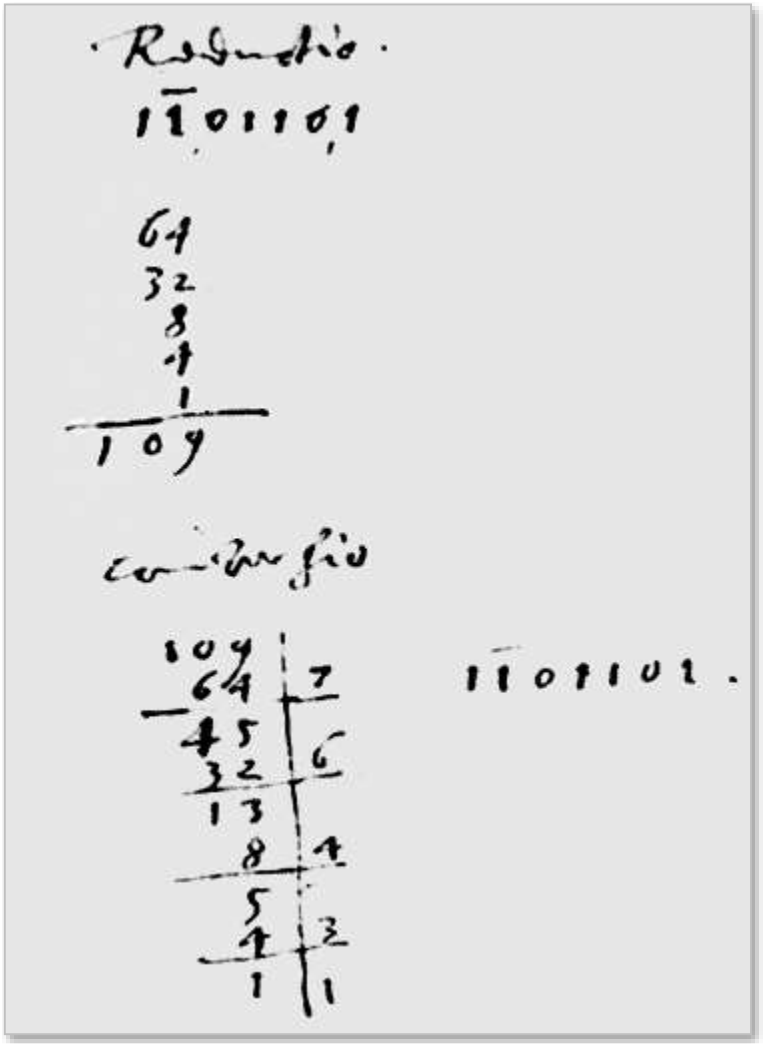
0	0
a	1
ao	2
aa	3
aoo	4
aoa	5
aao	6
aaa	7
aooo	8
aooa	9
aoao	10
aoaa	11
aaoo	12
aaoa	13
aaao	14
aaaa	15
aoooo	16

Dualzahlen aber auch schon vor Leibniz! (2)

Noch etwas früher, um 1605, befasste sich, wie aus unveröffentlichten Manuskripten hervorgeht, der englische Mathematiker, Physiker und Astronom Thomas Harriot (1560 – 1621) mit Dualzahlen. Angeregt dazu wurde er offenbar durch kombinatorischen Überlegungen, alle Teilmengen einer endlichen Menge in tabellarischer Weise darzustellen. Einen darüber hinausgehenden Nutzen scheint er in den Dualzahlen allerdings nicht erkannt zu haben.

Der nebenstehende Ausschnitt aus einem seiner Manuskripte zeigt die Umwandlung von der Dualdarstellung in die Dezimaldarstellung („Reductio“) und umgekehrt („Conversio“) am Beispiel 109 (dez.) = 1101101 (dual). An anderer Stelle im Manuskript finden sich Beispiele für die binäre Addition, Subtraktion und Multiplikation.

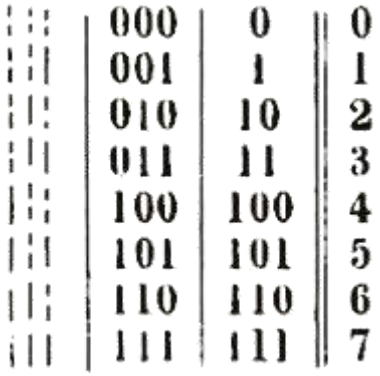
Durch systematische Kombination durchgezogener und gebrochener Linien konnten im chinesische Orakel I Ging bereits im 8. Jhd. v. Chr. 64 Hexagramme in „Binärdarstellung“ gebildet werden; die Analogie zu den Dualzahlen 0 bis 63 ist allerdings rein formal, irgend eine Form von Arithmetik war damit nicht verbunden.



Quelle: John W. Shirley: Binary numeration before Leibniz. American Journal of Physics 19.8 (1951): 452-454.

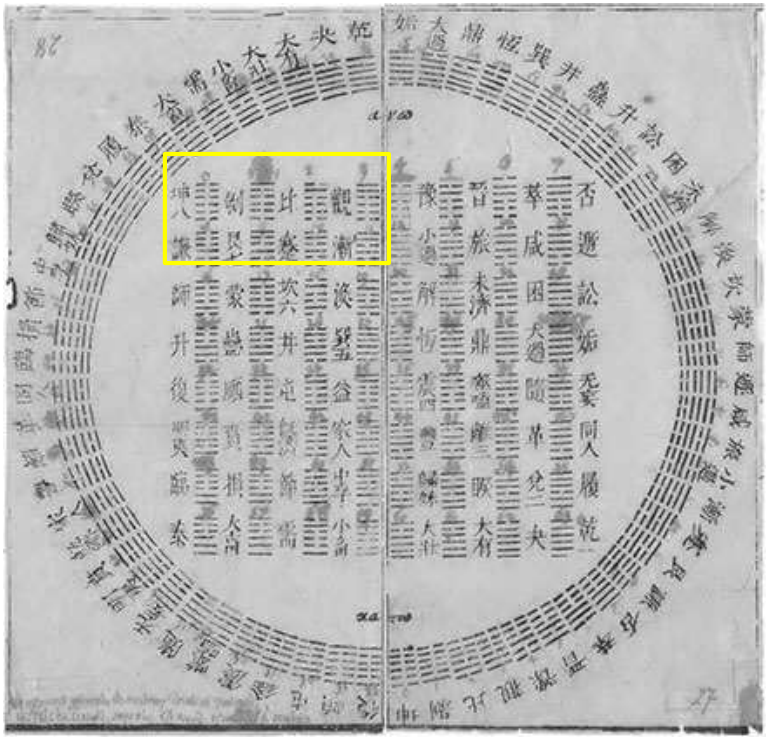
Das chinesische I Ging

Das I Ging (bzw. Yì Jīng), das „Buch der Wandlungen“, ist eine Sammlung von Strichzeichnungen und zugeordneten Sprüchen. Es ist der älteste der klassischen chinesischen Texte. Seine Entstehungsgeschichte wird bis in das 3. Jahrtausend v. Chr. zurückgeführt. Die älteste Schicht des Buches besteht aus 64 Gruppen von je sechs durchgehenden oder unterbrochenen Linien („Hexagramme“).



000	0	0
001	1	1
010	10	2
011	11	3
100	100	4
101	101	5
110	110	6
111	111	7

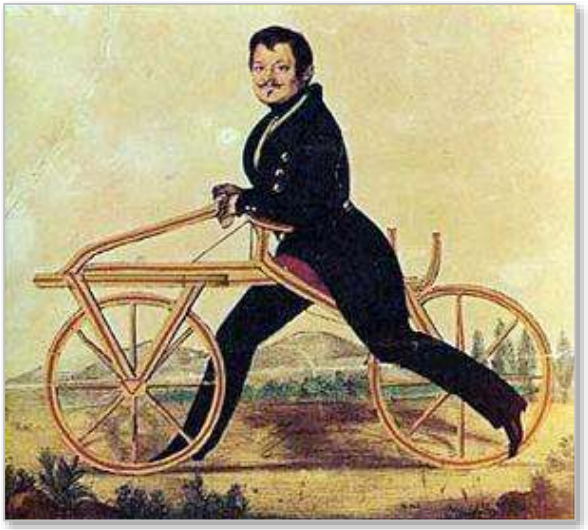
Leibniz erhielt 1701 obiges Dokument vom französischen Jesuitenpater Joachim Bouvet aus China und schloss daraus (fälschlicherweise) auf eine hochentwickelte altchinesische Mathematik.



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

Dualzahlen nach Leibniz: Karl Drais

Knapp 100 Jahre nach Leibniz‘ Tod beschäftigte sich kurioserweise Karl Drais (1785 – 1851), der Erfinder der Draisine und des Laufrads (einer frühen Form des heutigen Fahrrads), mit dem Dualsystem – nachdem es zwischenzeitlich fast vergessen wurde. Im *Badischen Magazin* veröffentlichte er 1813 zunächst eine kurze Vorankündigung eines geplanten Buches mit dem Titel „Dyadik“. Darin rechtfertigt



Badiſches Magazin.

Nr. 61. Dienſtag, den 16. März 1813.

er sein zukünftiges Buch so: „Das dyadische Rechensystem wurde schon vor mehr als hundert Jahren von dem großen Leibnitz aufgestellt, ist aber bis zur Stunde noch nicht genug gewürdigt worden. In der Aufstellung eines ganzen dyadischen Systems hingegen bin ich – nach der Behauptung des belesenen Herrn Professors Bürmann – der erste.“ Nachfolgend seine Anzeige im „Badischen Magazin“:

Dyadische Charakterik. *)

Auf die in einem öffentlichen Blatte erschienene gütige Aufforderung meines lieben verehrungswürdigen Lehrers, des Großherzogl. Badischen Herrn Directors und Professors Bürmann, zeige ich hierdurch an, daß ich zwar hoffe, mein bereits unternommenes Werk bald herausgeben zu können, und daß meine Absicht mit der des Herr Bürmann sehr übereinstimmt, indem wir uns beyde bestreben, zur Abkürzung und Uebersicht der Gedankenmittheilung etwas wesentliches beyzutragen. Herr Bürmann hat aber sein System zu einer allgemeinen Bezeichnung mit außerordentlichen Einsichten und Fleiß schon so weit verfolgt, als man es kaum einer ganzen gelehrten Gesellschaft hätte zumuthen können. Ich hingegen habe vor der Hand blos eine Vertheidigung des dyadischen Rechensystems nebst einigen Ideen zu einem durch mathematische Eintheilung vollkommen deutlichen, leicht zu übersehenden, sehr kurzen und durchaus wohllautenden Sprachsystem in die Bearbeitung genommen.

Mein Hauptsatz ist folgender:

Zur Bildung aller Charakterik *), sowohl für allgemeine Schrift- und Rede-Sprache **), als auch für allgemeine Messung und Zählung, ist das dyadische System das beste, weil es sich auf die einfachsten Grundsätze reducirt.

Der Titel meiner ersten kleinen Schrift über diesen Gegenstand wird seyn:

DYADIK,

oder

Aufstellung einer Charakterik, welche Alles durch zwey Zeichen ausdrückt.

*

Wenn einige Leser dieser Anzeige Lust haben, sich umständlicher von der Wahrheit meiner Sätze zu überzeugen, so bin ich schon vor der Herausgabe bereit, denselben das Wesentliche meiner Bearbeitung vorzulegen.

*) Durch dieses neue, aus dem Griechischen geformte Wort versteht Herr Director Bürmann — der Stifter desselben — die Bezeichnungskunde, nämlich die wissenschaftliche, die systematische Bezeichnung. Es muß demnach nicht Karakterik geschrieben, und noch weniger mit Charakteristik verwechselt werden, welches, ohne nähere Bestimmung, blos eine genaue Bezeichnung oder Unterscheidung bedeutet.

*) Wo nicht die Natur der Sache selbst eine Ausnahme vorschreibt, wie es z. B. in der Zeitrechnung, bey dem Uebergang von Jahren zu Tagen, oder von $\frac{1}{12}$ Jahr zu 16 Tagen, der Fall ist.

**) In jeder dyadischen Abstufung braucht man blos auf ein einziges charakterisches Unterscheidungszeichen Acht zu geben.

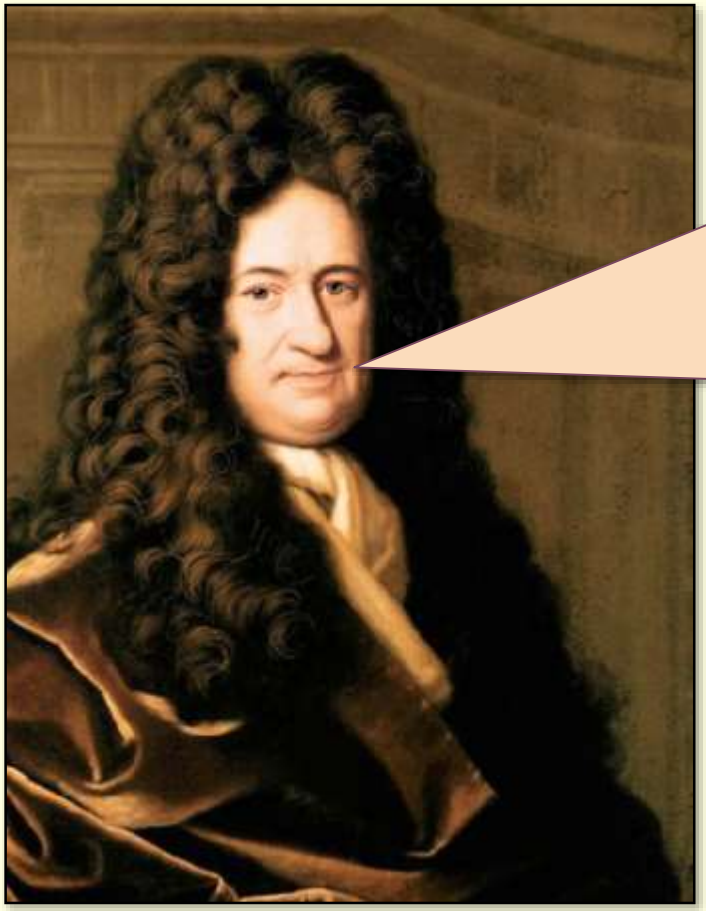
Dualzahlen nach Leibniz: Karl Drais (2)

Die angekündigte kleine Schrift von Karl Drais mit dem Titel „Dyadik“ erscheint tatsächlich 1814 und soll dem misslichen Umstand, dass „der grosse Leibnitz leider zu früh gestorben ist, um die Welt von dieser Wahrheit zu überzeugen“, abhelfen. Allerdings ist das 16 Seiten umfassende Traktat wenig ergiebig: Den „Beweis“ für den Vorteil des Dualsystems gegenüber dem Dezimalsystem sieht Drais vor allem darin, dass von Natur aus Verdoppelungen und Halbierungen 3 bis 4 Mal häufiger als Verzehnfachungen oder Einteilung in 10 gleiche Teile benötigt werden. Den rhetorischen Einwand, wir hätten 10 Finger, kontert er damit, dass wir 2 Arme, 2 Füsse, 2 Ohren und 2 Augen hätten und fährt fort: „Wie angenehm ist es nicht, wo es so üblich ist, dass eine Maas gerade 2 Bouteillen 4 Schoppen und 8 Gläser, ein Malter 8 Simmern, ein Simmern 16 Mäßel, ein Pfund 32 Loth, ein Loth 4 Quintchen, ein Conventionsthaler 2 schwere Gulden, ein Batzen 4 Kreuzer, ein Kreuzer 4 Pfennig, eine Carolin 4 grosse Thaler, eine Ducat 2 Kronenthaler, ein Sechsbätzner 2 Dreybätzner, ein Dreybätzner 2 Sechser und ein Sechser 2 Groschen hat.“

Drais hofft, dass ein einsichtsvoller Regent das dyadische System für die Staatsgeschäftsgänge einführt; die Staatsdiener würden es in wenigen Tagen erlernen. Das Erstellen neuer Tabellen mit dyadischen Logarithmen würde die Gelehrten nur rund ein Jahr beschäftigen. Er schliesst mit den Worten: „Sollte es mir durch diese kleine Abhandlung gelingen, zwey solche beschriebene Männer zur Ausführung meiner nützlichen Ansicht zu finden, so will ich recht gerne den Ruhm mit lhnen theilen, der dadurch für die Nachwelt entsteht.“

Drais war ein begnadeter Erfinder (er erfand u.a. auch noch die Tastenschreibmaschine und den Klavierrekorder), intellektuell (und erst recht in mathematischer Hinsicht) konnte er es mit Leibniz aber offensichtlich nicht aufnehmen.

Leibniz' Traum:
Die Automatisierung des logischen Denkens



„In Philosophia habe ich ein Mittel funden, dasjenige was Cartesius und andere per Algebram et Analysin in Arithmetica et Geometria gethan, in allen scientien zuwege zu bringen per Artem Combinatoriam welche Lullius und Kircherus zwar excolirt, bey weitem aber in solche deren intima nicht gesehen. Dadurch alle Notiones compositae der ganzen welt in wenig simplices als deren Alphabet reduciret, und aus solches alphabets combination wiederumb alle dinge, samt ihren theorematibus, und was nur von ihnen zu inventiren müglich, ordinata methodo, mit der zeit zu finden, ein weg gebahnet wird."

Welche invention als mater aller inventionen von mir vor das importanteste gehalten wird, ob sie gleich das ansehen noch zur zeit nicht haben mag.

Von der Universalschrift zum *Calculemus!*

Es ist nicht im mindesten überraschend, dass der geistige Impuls, der zur Entwicklung der mathematischen Logik geführt hat, gleichzeitig die ideelle oder tatsächliche Mechanisierung der Denkprozesse in Gang brachte. – Norbert Wiener

Gottfried Leibniz strebte danach, eine universelle Schrift zu entwickeln, in der jede Erkenntnis und alle menschlichen Gedanken klar ausgedrückt werden können, so dass mit einem darauf aufsetzenden Kalkül im Sinne einer symbolischen Logik (dem „calculus ratiocinator“) geistige Tätigkeiten, auch über die Mathematik hinaus, mechanisiert werden können. Angeregt zu dieser grandiosen Idee wurde er nicht nur durch die vom Theologen und Philosophen Raimundus Lullus bereits im 14. Jahrhundert ausgedachten „Ars Magna“ (in der durch mechanisches Kombinieren von Begriffen mittels einer „logischen Maschine“ auf jede (Glaubens)frage eine Antwort gefunden werden sollte), sondern vor allem durch die Arbeiten von René Descartes zur analytischen Geometrie (Algebraisierung und damit rechnerische Operationalisierung der synthetisch-idealistischen Geometrie nach Euklid) sowie nicht zuletzt auch durch seinen eigenen Differentialkalkül, der ma-

Bild: Österreichische Nationalbibliothek

Raimundus Lullus, „one of the most remarkable tragicomic figures of the Middle Ages“ (Martin Gardner) mit Sprechblase „Lux mea est ipse Dominus“ (Der Herr selbst ist mein Licht; Mi 7,8)

thematisch-physikalische Sachverhalte berechenbar machte. Die klassischen logikorientierten Arbeiten der griechischen Philosophen, wie sie etwa in Aristoteles' Werk beschrieben sind, sowie die darauf aufbauenden Bemühungen der oft theologisch motivierten mittelalterlichen Scholastik gehören natürlich ebenfalls zum geistigen Hintergrund von Leibniz' Idee.

Dieser Idee zugrunde lag einerseits die Annahme, dass die Elemente einer universellen Sprache kombinatorisch aus wenigen einfachen Begriffen gewonnen werden können (die begriffliche Konstruktionsmethode bezeichnete Leibniz daher als „ars combinatoria"), andererseits die Auffassung, dass das diskursive menschliche Denken nichts anderes als die Verknüpfung und Ersetzung von Zeichen ist *(„omnis Ratiocinatio nostra nihil aliud est quam characterum connexio et substitutio").* Letztere stellt, gemessen an den seinerzeitigen vorherrschenden theologischen Lehrmeinungen, eine radikale Vorstellung dar, sie ist jedoch heute grundlegend für die moderne Kognitionswissenschaft und die (sogenannte „starke") Theorie der künstlichen Intelligenz.

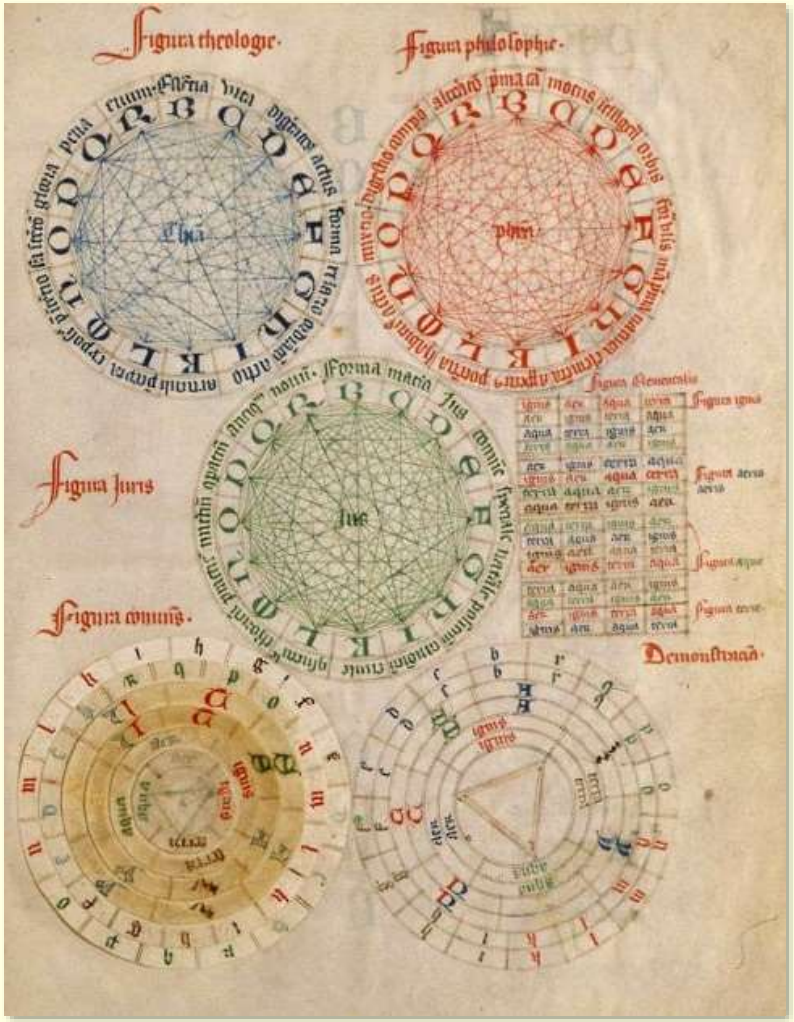


Abbildung aus der „Ars Magna" von Raimundus Lullus: Ringförmige, mit Liniennetzen gefüllte Diagramme, die zu Papiermaschinen aus drehbaren Kreisflächen führen – ein unvollkommener, aber geistesgeschichtlich folgenreicher, erster Versuch der schematischen Erzeugung aller wahren Aussagen.

Blog.hnf.de/das-universum-des-ramon-llull/

Raimundus Lullus – der Erfinder der Denkmaschine des 13. Jahrhunderts

Auszug aus https://blog.hnf.de/das-universum-des-ramon-llull/:

Geboren wurde Lullus in Palma de Mallorca als Sohn eines Ritters vermutlich 1233. Der junge Ramon liebte Wein, Weib und Gesang, war Vater zweier Kinder und arbeitete im Königspalast als Prinzenerzieher und Hofmeister. Das änderte sich 1263. Lullus hatte ein religiöses Erlebnis, entsagte allen weltlichen Freuden und schloss sich dem Mönchsorden der Franziskaner an. In seiner Kammer studierte er sowohl christliche als auch islamische Theologie und Philosophie. Arabisch lernte er von einem maurischen Sklaven. 1271 oder 1272 schrieb er sein erstes Buch über die Logik des persischen Gelehrten Abu Hamid al-Ghazali. Von nun an führte Ramon Lull ein ruheloses Leben als Denker und Schriftsteller. Sein Output umfasst mehr als 260 Titel auf Arabisch, Lateinisch und Katalanisch, seiner Muttersprache. Im Alter erfuhr er Anerkennung der Professoren der Pariser Universität; zugleich packte ihn das missionarische Feuer.

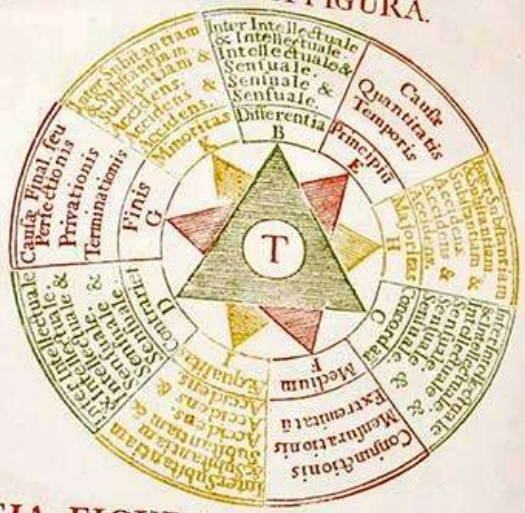
Raimundus Lullus (links) lernt Arabisch. (BLB, Breviculum ex artibus Raimundus Lulli electum.) "It is said that one day Lull struck the Moorish slave in the face after hearing him blaspheme the name of Christ. Soon thereafter the Moor retaliated by attacking Lull with a knife. Lull succeeded in disarming him and the slave was jailed. Expecting to be put to death, the Moor hanged himself with the rope that bound him." [Martin Gardner]

Ramon Llull war sicher der ungewöhnlichste Intellektuelle des Mittelalters. Schon früh nannte man ihn den „erleuchteten" doctor illuminatus. Er schrieb aber auch als Erster philosophische Texte in einer Volkssprache. Sein Hauptwerk ist die Ars magna, die Große Kunst. Das Markenzeichen der Ars magna sind die ringförmigen Diagramme, die zu Papiermaschinen aus drehbaren Kreisflächen führen. Die Grafiken sollten helfen, durch Vernunftschlüsse Fragen aus jeder Wissenschaft zu beantworten.

Das algorithmische Denken und die Freude an konstruktiven Lösungen machen Ramon Llull zu einem Gründervater der Informatik.

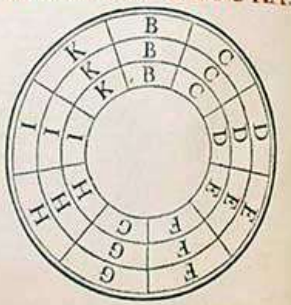
PRIMA FIGURA.

SECUNDA FIGURA.



TERTIA FIGURA

QUARTA FIGURA.



BEATI RAYMUNDI LULLI DOCTORIS ILLUMINATI ARS INVENTIVA VERITATIS SEU ARS INTELLECTIVA VERI.

QUÆ EST

INSTRUMENTUM INTELLECTIVÆ POTENTIÆ.

DEUS Potentissime & Benignissime, cujus Sapientiæ & Amoris non est numerus neque finis, ut te cognoscere sciamus & amare, incipit *Ars Inventiva.*

DE PROLOGO

1. ARs præsens descendit ab *Arte Demonstrativa;* & licèt existentia vel essentia hujus & illius sit una & eadem, tamen est earum Modus procedendi diversus: nam illa procedit per Terminos redactos in Literas, ista verò est contenta suis propriis Terminis seu Principiis, nec indiget aliâ notâ Literarum, sicut illa, ut illi, qui volunt evitare Alphabetum Demonstrativæ Artis, attingant Terminos seu Principia hujus Artis sub suis propriis significatis. Veruntamen oportet nos necessariò uti Literis pro Terminis in ista eadem Arte in Figuris, non autem in processu: nam aliter non posset fieri Inquisitio nec Inventio in duabus ultimis Figuris hujus Artis.

2. „Et cùm essentiæ & operationes rerum, „de quibus intenditur, sint majores & plu„res in se ipsis, quàm earum similitudines in „intellectu, & ipsæ similitudines majores & „ampliores in intellectu, quàm earum signi„ficata in verbis, est magna distantia significato-

Stephan Meier-Oeser schreibt dazu: „Das von älteren Entwürfen einer Universalsprache verfolgte Ziel der Erleichterung des wechselseitigen Verkehrs der Völker sei, wie Leibniz meint, noch der geringste Nutzen, den eine Universalcharakteristik haben würde, bildet sie als *scriptura rationalis* doch das mächtigste Instrument für die Erreichung des größten, was dem Menschen überhaupt widerfahren kann: Der Perfektionierung der Geistesfunktionen (perfectio functionum mentis). Es ist dies jedoch eine Vervollkommnung, die nicht am Ort dieser Geistesfunktionen selbst, der mens, ansetzt, sondern im externen Medium der Zeichen realisiert werden soll. Sie vollzieht sich daher als Exteriorisierung der Denkfunktionen, als Übertragung von Geistesfunktionen auf ein äußeres Zeichensystem, so dass die Vollkommenheit des Zeichensystems als Perfektionierung der Geistesfunktionen selbst erscheinen kann. [...] Entsprechend postuliert er, es müsste sich eine Art Alphabet der menschlichen Gedanken ersinnen und durch die Verknüpfung seiner Buchstaben und die Analyse der Worte, die sich aus ihnen zusammensetzen, alles andere entdecken und beurteilen lassen.“

In einem Brief an den Gelehrten Theodor Haak in England schrieb Leibniz: „Ich meine nämlich, eine Art Universalschrift ersinnen zu können, vermöge derer wir bei Dingen aller Art so rechnen und Beweise finden können wie in der Algebra und der Arithmetik“. *(„Ego enim scripturam quandam universalem excogitari posse arbitror, cuius ope calculare in omni genere rerum et demonstrationes invenire possimus perinde ac in Algebra et Arithmetica.“)* Es ging Leibniz also nicht nur um eine Beschreibung von Sachverhalten, sondern ganz bewusst sollte durch schematisches Rechnen die Entdeckung von noch Unbekanntem möglich sein. In einem Brief an den Abbé Jean Galloys [Gallois] bekräftigt Leibniz, dass seine Universalsprache nicht primär auf die (gesprochene) Sprache zielt, sondern auf die Gedanken und das Verständnis: „En voulant aller d’Angleterre en Hollande j’ay esté retenu quelque temps dans la Tamise par les vents contraires. En ce temps là ne sçachant que faire et n’ayant personne dans le vaisseau que des mariniers, je meditois sur les choses là, et surtout je songeois à mon vieux dessein d’un langue

ou écriture rationelle, dont le moindre effect seroit l'universalité et la communication de differentes nations. Son veritable usage seroit de peindre non pas la parole […] mais les pensées, et de parler à l'entendement plutost qu'aux yeux. Car si nous l'avions telle que je la conçois, nous pourrions raisonner en metaphysique et en morale à peu pres comme en Geometrie et en Analyse." Dass mit seinem Esperanto des Geistes auch gleich noch die nationalen Sprachbarrieren überwunden werden, stellt für Leibniz so gesehen also eher einen Nebeneffekt dar.

Im Jahr 1679 unternimmt Leibniz mehrere Ansätze, einen konkreten Logikkalkül (*characteristica universalis* bzw. *calculus universalis*) zu entwerfen – „eine Charakteristik der Vernunft, kraft derer Wahrheiten der Vernunft gewissermaßen durch einen Kalkül wie in der Arithmetik und in der Algebra, so in jedem anderen Bereich, soweit er der Schlussfolgerung unterworfen ist, erreichbar würde." In seinem kategorisch verfassten Aufsatz „Elementa Calculi" versucht Leibniz, den Begriffen in der Weise Zahlen zuzuordnen, dass begriffliche Unterordnung durch Teilbarkeit repräsentiert wird – in heutiger Terminologie formuliert, entwickelte er also erste Ansätze einer Ontologie in Gestalt eines mathematischen Begriffsverbands. Hieraus einige Zitate:

„Jedem Terminus soll eine charakteristische Zahl zugeordnet werden, die zum Rechnen verwenwendet würde, wie der Terminus selbst zum Schlussfolgern verwendet wird. *(Cuilibet Termino, assignetur suus numerus characteristicus, qui adhibeatur in calculando, ut terminus ipse adhibetur in ratiocinando.*)

Wenn der Begriff eines gegebenen Terminus in direkter Weise aus den Begriffen zweier oder mehrerer anderer Termini gebildet wird, dann soll die charakteristische Zahl des gegebenen Terminus durch Multiplikation der charakteristischen Zahlen derjenigen Termini, die den Begriff des gegebenen Terminus bilden, erzielt werden. Als Beispiel, da der Mensch ein vernunftbegabtes Lebewesen ist: Wenn mit der Zahl *a*, etwa 2, die Lebewesen charakterisiert sind, und Ver-

nunft mit der Zahl *r*, etwa 3, dann wird die Zahl *h* für Mensch gleich *a r* sein, das ist in diesem Beispiel 2 x 3 oder 6. (*Quando Termini dati conceptus componitur in casu recto ex conceptibus duorum pluriumve aliorum terminorum, tunc numerus termini dati characteristicus producatur ex terminorum termini dati conceptum componentium numeris characteristicis invicem multiplicatis. Verbi gratia quia Homo est Animal rationale hinc si sit Animalis numerus* a*, ut 2 Rationalis vero numerus* r *ut 3, erit numerus hominis seu* h *idem quod* a r *id est in hoc exemplo 2 x 3 seu 6.)"*

Gleichermassen würde Gold die Charakteristik 15 zukommen, wenn Metall den Wert 3 und die Eigenschaft „am schwersten" den Wert 5 hätten. Leibniz kommt dann auf die Beziehung zwischen Konzepten zu sprechen, insbesondere auf die Konzepthierarchie:

„Deshalb sage ich, Gold ist umfassender als Metall, da für den Begriff des Goldes mehr erforderlich ist als für den des Metalls, und es macht mehr Arbeit, Gold herzustellen als irgendein Metall. *(Itaque dico aurum majus metallo, quia plura requiruntur ad notionem auri quam metalli, et majus opus est aurum producere, quam metallum qualecunque.)*" Und weiter: „Mensch und Tier haben gemeinsam den Begriff des Lebewesens; Gold und Silber den des Metalls. (*Homo et brutum animalis conceptum habent communem. Aurum et Argentum metalli.*)" Mensch und Affe stehen aber jedenfalls nicht in einer hierarchischen Ordnung: „Sei zum Beispiel die charakteristische Zahl für den Menschen als 6 angenommen, für den Affen hingegen 10, dann enthält offenbar weder der Begriff des Affen den Begriff des Menschen noch umgekehrt, da weder 10 ganz durch 6 teilbar ist noch andersherum 6 durch 10. (*Exempli gratia, si Numerus characteristicus hominis fingatur esse 6, simiae vero 10 patet quod nec simiae notio contineat notionem hominis, nec contra haec illam, quia nec 10 dividi potest exacte per 6 nec contra 6 per 10.)"*

Eines der Manuskripte zu den „Elementa Characteristicae universalis" enthält am Ende eines beigefügten Notizblattes eine von Leibniz angefertigte Skizze. (Es ist aber unklar, wen die bärtige

Gestalt darstellen soll und was Leibniz zur der Skizze an dieser Stelle veranlasst hat. Vielleicht entstand sie ja nur als sinnloses Gekritzel beim Nachdenken...)

Wir kommen auf die Universalsprache und deren Zweck zurück. Mit ihr möchte Leibniz die Logik operationalisieren und letztlich automatisieren. Der Logik kommt nach Leibniz zwei Aufgaben zu, einerseits die der „Beurteilung“ von Aussagen („ars iudicandi“), d.h. den Nachweis ihrer deduktiven Abhängigkeit von anderen Aussagen, deren Wahrheit feststeht; andererseits die Herleitung („Erfindung“) von neuen Aussagen („ars inveniendi“) mittels formalem Schliessen aus bereits als wahr erkannten Aussagen. In heutiger Informatik-Terminologie: Formale Sprachen in der Funktion eines „Akzeptors“ bzw. eines „Generators“.

...nicht allein was fürgestellet zu beurtheilen, sondern auch was verborgen zu erfinden. -- Leibniz

In seinem Essay ohne Titel von 1677 schreibt Leibniz u.a. [Übersetzung von Herbert Herring]: „Nun geschah es aber, ich weiss nicht durch welches Schicksal, dass ich schon als Knabe auf diese Betrachtungen geführt wurde [...] Bei meinen eifrigen Bemühungen um dieses Problem gelangte ich dann mit innerer Notwendigkeit zu einer Betrachtung von erstaunlicher Tragweite: Es müsste sich, meinte ich, eine Art Alphabet der menschlichen Gedanken ausdenken und durch die Verknüpfung seiner Buchstaben und die Analyse der Wörter, die sich aus ihnen zusammensetzen, alles andere entdecken und beurteilen lassen. Dieser Einfall machte mir nun ganz ausserordentliche Freude, die allerdings nur kindlich war, da ich die Wichtigkeit der Sache damals noch nicht ausreichend begriff. Später aber kräftigte sich mit jedem weiteren Fortschritt meiner Erkenntnis in mir zugleich der Entschluss, einen Gegenstand von solcher Bedeutung weiter zu verfolgen. (*Factum est autem, nescio quo fato, ut ego adhuc puer, in has cogitationes inciderem [...]. Cui studio cum intentius incumberem, incidi necessario in hanc contemplationem admirandam, quod scilicet*

excogitari posset quoddam Alphabetum cogitationum humanarum, et quod literarum hujus Alphabeti combinatione et vocabulorum ex ipsis factorum analysi omnia quae ratione constant et inveniri et dijudicari possent. Hoc ego deprehenso mirifice exultavi, puerili quidem gaudio, nam tunc rei magnitudinem non satis capiebam. Sed postea quanto majorem in rerum cognitione progressum feci, eo magis confirmatus sum in consilio rem tantam prosequendi.)“

Leibniz sieht den Nutzen seiner Universalschrift aber auch darin, dass damit keine fehlerhaften Schlüsse mehr möglich sein sollten, denn diese würden sich als erkennbare Rechenfehler manifestieren: „Das ist zu erreichen: dass jeder Fehlschluss nichts anderes als ein Rechenfehler ist, [...] der durch die blossen Gesetze dieser philosophischen Grammatik leicht in Ordnung zu bringen ist. *(Id scilicet efficiendum est, ut omnis paralogismus nihil aliud sit quam error calculi, [...] ex ipsis grammaticae hujus philosophicae legibus facile revincendus.).“*

Auf nette Weise fasst Leibniz seine Vision in einer Beschreibung zweier Menschen guten Willens zusammen, die, in einen philosophischen Disput verwickelt und auf der Suche nach der Wahrheit, ihre Argumente in die formale Sprache übersetzen und dann nicht wie zwei Philosophen disputieren, sondern wie zwei Mathematiker sagen: „Calculemus“ – Rechnen wir es einfach aus! Wörtlich im Original auf Latein: *„Quo facto quando orientur controversiae, non magis disputatione opus erit inter duos philosophos, quam inter duos Computistas. Sufficiet enim calamos in manus sumere sedereque ad abacos, et sibi mutuo (accito si placet amico) dicere: calculemus.“* („Bei Streitfragen zwischen zwei Philosophen wird keine grössere Diskussion erforderlich sein als zwischen zwei Rechenmeistern oder Buchhaltern. Es wird nämlich genügen, Schreibzeug zur Hand zu nehmen, sich ans Rechengerät zu setzen und – wenn es beliebt, nach Hinzuziehen eines Vertrauten – zueinander zu sagen: *lasst uns rechnen!*“) Leibniz selbst scheint an diesem etwas skurrilen Szenario Gefallen gefunden zu haben, an anderer Stelle schrieb er nämlich: „Et si quelqu‘un doutoit de ce que j’aurois avancé, je luy dirois: contons, Monsieur, et ainsi prenant la plume et de l’encre, nous sortirions bientost d’affaire.“

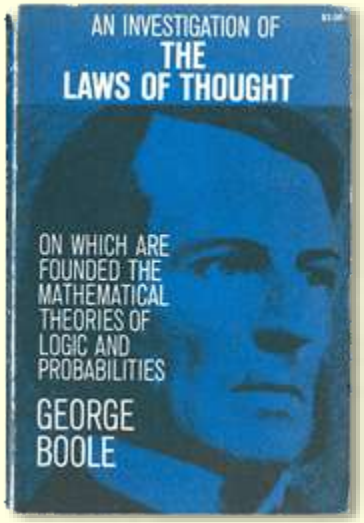
Dieses „lasst uns rechnen“ muss nun aber gar nicht mit Feder und Tinte geschehen, denn dank der rechnender Maschinen sollten Wahrheiten prinzipiell nun auch maschinell aufgefunden werden können – sind doch die von Leibniz konstruierte Rechenmaschine sowie seine symbolverarbeitende Chiffriermaschine nur Beispiele dafür, die „menschliche Gemüthskraft“, welche bisher als ein „proprium hominis“ gehalten wurde, durch „eine Machina zu wege zu bringen“. Eine veritable Maschine, die wahre Aussagen erzeugt oder bestätigt, wird von Leibniz nicht explizit postuliert; durch Phrasen wie „wie durch die Tätigkeit einer Maschine“ aber alludiert: „Diese allgemeine Algebra bewirkt, dass wir, selbst wenn wir das wollten, nicht irren können und dass die Wahrheit gleichsam als gemalt und wie durch die Tätigkeit einer Maschine auf das Papier gedruckt erfasst werden kann“; *(„Haec Algebra generalis praestat, Errare ne possumus quidem si velimus, et ut Veritas quasi picta, velut Machinae ope in charta expressa, deprehendatur“).*

Auch als Intelligenzverstärker solle sich, so Leibniz, seine Methode mit der Universalschrift eignen. Man könne sie nämlich „so gebrauchen, dass jeder auch nur mit mittelmässigem Verstand Begabte [...] die schwierigsten Dinge verstehen und die schönsten Wahrheiten [...] entdecken könnte.“ *(„...ita utendi, ut quisque mediocri licet ingenio praeditus [...], difficillima etiam intelligere, et pulcherrimas veritates [...] invenire possit.“)* Man kann sich natürlich fragen, ob ein „ausgelagerter Verstand“ einem mittelmässig Begabten wirklich nützt, und auch der französische Logiker Louis Couturat merkt in seinem Buch „La logique de Leibniz, d‘après des documents inédits“ an, dass mit Leibniz‘ Methode der Verstand eher nicht unterstützt, sondern ersetzt wird: „Mais elle est plus encore, à savoir l‘incarnation et le substitut de la raison : elle n‘aide pas seulement le raisonnement, elle le remplace“.

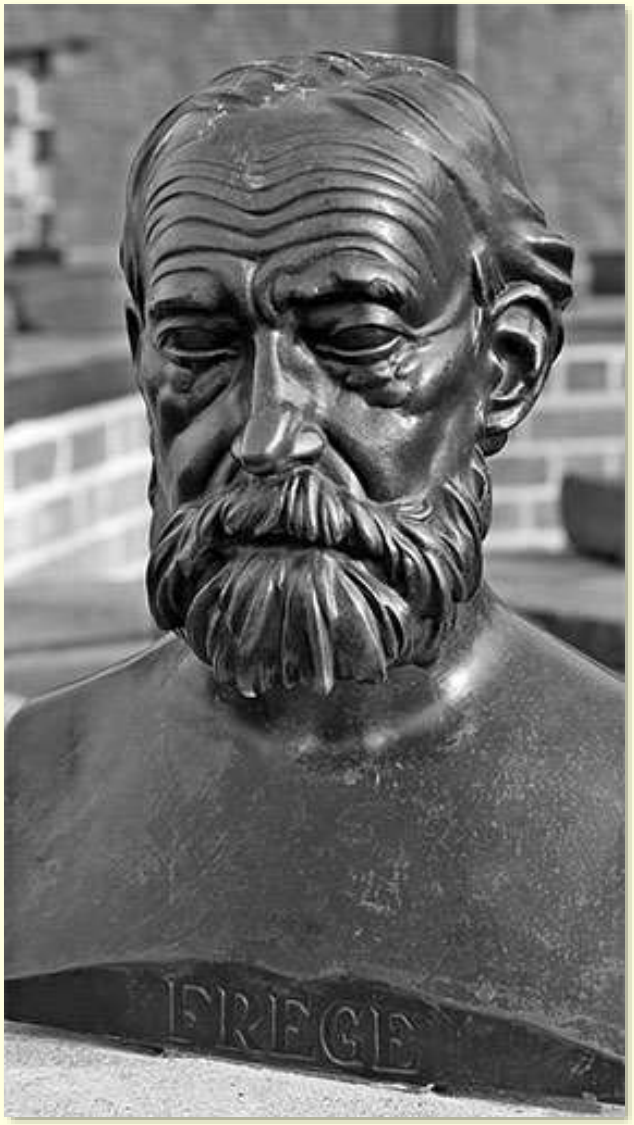
Zusammengefasst bestand Leibniz’ Idee in der Bildung eines vollständigen Systems von Begriffen als Grundlage einer schematischen (und idealerweise automatisierbaren) Methode in Form eines Kalküls, mit dem alle möglichen Urteile und Schlüsse gefunden werden können. Dem liegt

implizit die Überzeugung zugrunde, dass die algorithmische Methode der Mathematik auf den gesamten Bereich menschlicher Geistestätigkeit anwendbar sei und jedes in der Universalsprache als Aussage formulierte Problem mit „ja“ oder „nein“ entscheidbar sei. In den Jahrzehnten und Jahrhunderten nach Leibniz hat sich allerdings herausgestellt, dass diese Annahme problematisch ist und dass es dabei sogar prinzipielle Grenzen gibt. Dazu mussten aber zunächst Logik, Algorithmik und der Berechnungsbegriff besser verstanden und formalisiert werden.

Ein konkreter Logikkalkül wurde tatsächlich erst knapp zwei Jahrhunderte nach Leibniz von Augustus De Morgan und George Boole in Form der heute allseits bekannten Aussagenlogik und Booleschen Algebra entwickelt. Die Mächtigkeit dieses Formalismus ist allerdings gegenüber der Leibnizschen Vision noch relativ beschränkt.



“It is upon the foundation of this general principle, that I purpose to establish the Calculus of Logic, and that I claim for it a place among the acknowledged forms of Mathematical Analysis.”
-- George Boole



Gottlob Frege, Bronzebüste in Weimar von K.-H. Appelt

Mathematiker wie Gottlob Frege, Bertrand Russell, Giuseppe Peano und andere setzen in der zweiten Hälfte des 19. sowie zu Beginn des 20. Jahrhunderts das (gewissermassen von Leibniz begründete) Programm einer Formalisierung der Logik fort, indem weitere wichtige Aspekte wie Prädikatenlogik oder Semantik erforscht wurden.

Giuseppe Peano (1858 – 1932)

https://andreamacco.files.wordpress.com/2015/09/peano-math.jpg

Nicht allen Mathematikern gefielen seinerzeit die Bemühungen um die Formalisierung der Logik, da diese eine intuitionsfreie „automatische Mathematik“ zu implizieren schienen, bei der Beweise mechanisch ohne Sinn und Verstand geführt werden könnten. Henri Poincaré etwa verglich dies mit dem lustlosen mechanischen Anwenden von Schachregeln gegenüber den wirklich das Spiel beherrschenden wahren Schachexperten und bringt die Kritik 1908 in seinem Buch „Science et Méthode“ so auf den Punkt: „Pour démontrer un théorème, il n’est pas nécessaire ni même utile de savoir ce qu’il veut dire. On pourrait [...] imaginer une machine où l’on introduirait les axiomes par un bout pendant qu’on recueillerait les théorèmes à l’autre bout, comme cette machine légendaire de Chicago où les porcs entrent vivants et d’où ils sortent transformés en jambons et en saucisses. Pas plus que ces machines, le mathématicien n’a besoin de comprendre ce qu’il fait.“

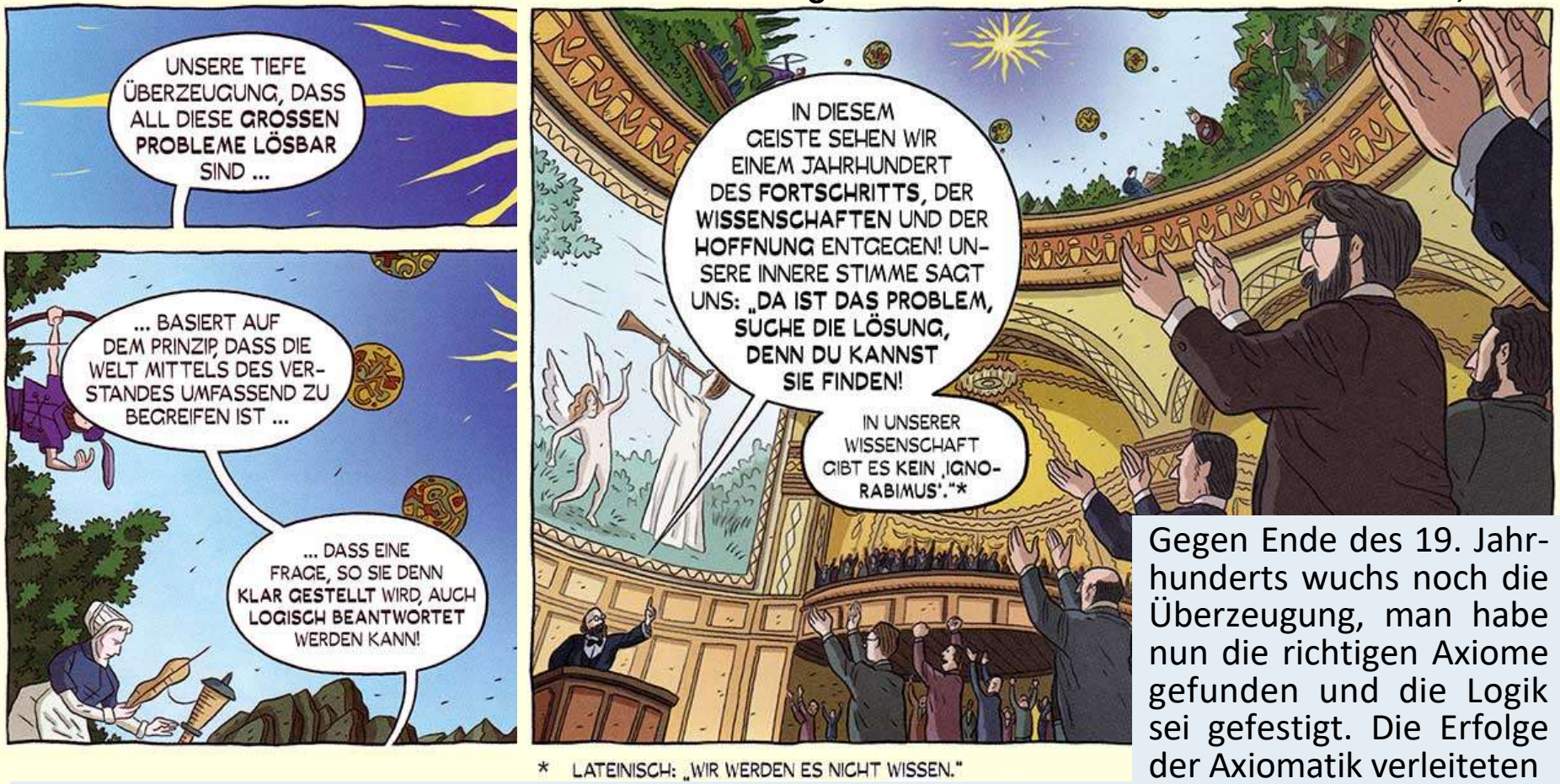
Dass jede mathematische Aussage im Prinzip formal beweisbar oder widerlegbar ist, dass also zumindest innerhalb der Mathematik und den durch sie formalisierbaren Wissensbereichen die

Leibniz'sche Idee einer kalkülbezogenen schematischen Verifikation möglich ist, war jedenfalls noch zu Beginn des 20. Jahrhunderts unbestritten. David Hilbert formulierte diese „Überzeugung, die jeder Mathematiker gewiss teilt" im Jahr 1900 in seiner programmatischen Rede „Mathematische Probleme" beim internationalen Mathematikerkongress in Paris so: *„Ich meine die Überzeugung, dass ein jedes bestimmte mathematische Problem einer strengen Erledigung notwendig fähig sein müsse, sei es, dass es gelingt, die Beantwortung der gestellten Frage zu geben, sei es, dass die Unmöglichkeit seiner Lösung und damit die Notwendigkeit des Misslingens aller Versuche dargetan wird."* (Hilbert nennt dies das „Axiom von der Lösbarkeit eines jeden Problems"; auf seinem Grabstein wird später stehen „Wir müssen wissen. Wir werden wissen.")

David Hilbert (1862–1943)

Roland Mechling kommentiert dies so: „Viel klarer kann man sein Glaubensbekenntnis zur unbedingten Entscheidbarkeit kaum formulieren! Folgerichtig ging Hilbert daran, die Leibniz'sche Idee von der vernünftigen Sprache oder Schrift in die Tat umzusetzen, indem er nämlich die Methode der algorithmischen Formulierung auf die gesamte mathematische Sprache ausdehnte: Sein Ziel war die Erstellung eines umfassenden formalen Regelwerks, in dem sich ein Mathematiker sicher bewegen kann. Ob ein Schluss von einer Aussage auf eine andere richtig ist, sollte sich völlig unabhängig vom Gehalt dieser Aussagen nur aufgrund von formalen Regeln begründen lassen." Hilbert begründet diese Vorgehensweise, fast in Opposition zu Poincaré und ganz im Geiste von Leibniz in seinem Buch „Die Grundlagen der Mathematik" so: „In meiner Theorie wird das inhaltliche Schliessen durch ein äusseres Handeln nach Regeln ersetzt; damit erreicht die axiomatische Methode diejenige Sicherheit und Vollendung, deren sie fähig ist und deren sie auch bedarf, wenn sie zum Grundmittel aller theoretischen Forschung werden soll."

Aus dem Comicbuch *Logicomix:*
David Hilbert beim internationalen Mathematikerkongress an der Pariser Sorbonne-Universität, 1900.

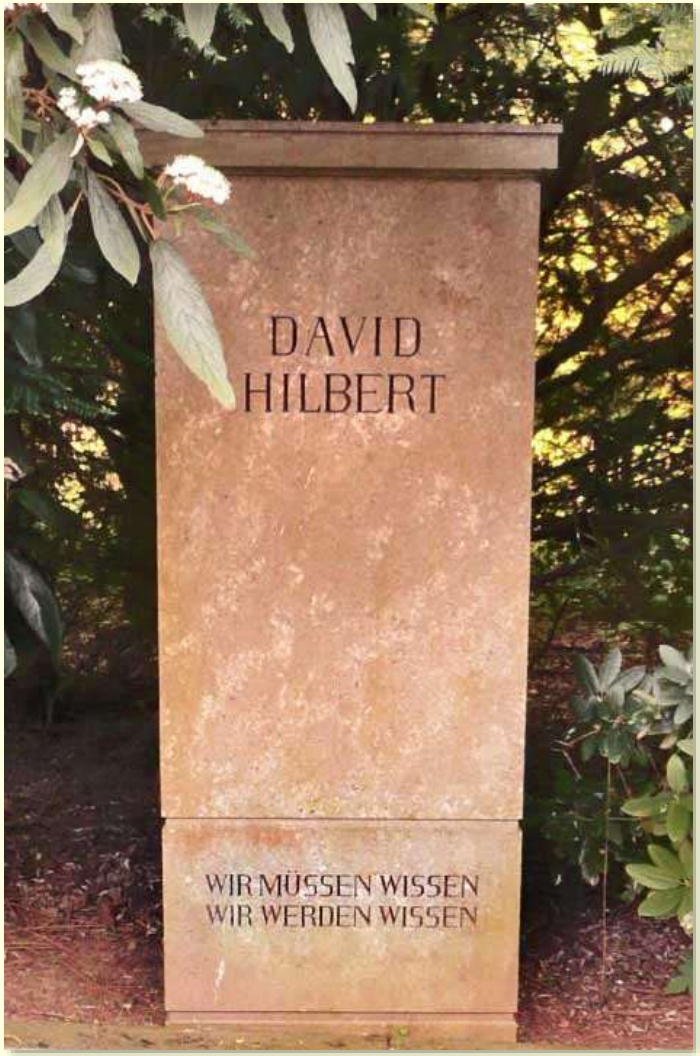


Gegen Ende des 19. Jahrhunderts wuchs noch die Überzeugung, man habe nun die richtigen Axiome gefunden und die Logik sei gefestigt. Die Erfolge der Axiomatik verleiteten David Hilbert und andere zu der Überzeugung, alles in der Mathematik sei logisch einwandfrei begründbar, jede Aussage könne man beweisen oder widerlegen. Hilbert stellte auf dem zweiten internationalen Mathematikerkongress im Jahr 1900 in seinem auf Deutsch gehaltenen Vortrag eine Sammlung von 23 seinerzeit offener mathematischer Fragen vor, die er in diesem Sinne allesamt für lösbar hielt. Aufgrund von Hilberts Prominenz übte die Vorstellung der Problemsammlung einen wesentlichen Einfluss auf die Entwicklung der Mathematik im 20. Jahrhundert aus. Einige der Probleme sind allerdings noch immer offen, andere stellten sich als unentscheidbar heraus.

Passend dazu veröffentlichen zwischen 1910 und 1913 Bertrand Russell und Alfred North Whitehead ihr dreibändiges Werk „Principia Mathematica“, in dem versucht wird, die gesamte klassische Mathematik in einem Logikkalkül zu entwickeln und mit diesem alle mathematischen Wahrheiten aus einem wohldefinierten Satz von Axiomen und Schlussregeln herzuleiten. Einige prinzipielle Fragen blieben offen, und David Hilbert schlug um 1920 ein mathematisches Forschungsprogramm vor, dessen Ziel darin bestand, einen Kalkül mit einfachen Axiomen zu finden, mit dem für jeden mathematischen Satz beweisbar ist, ob er wahr oder falsch ist (Entscheidbarkeit), und mit dem alle wahren Sätze aus dem Axiomensystem ableitbar sind (Vollständigkeit).

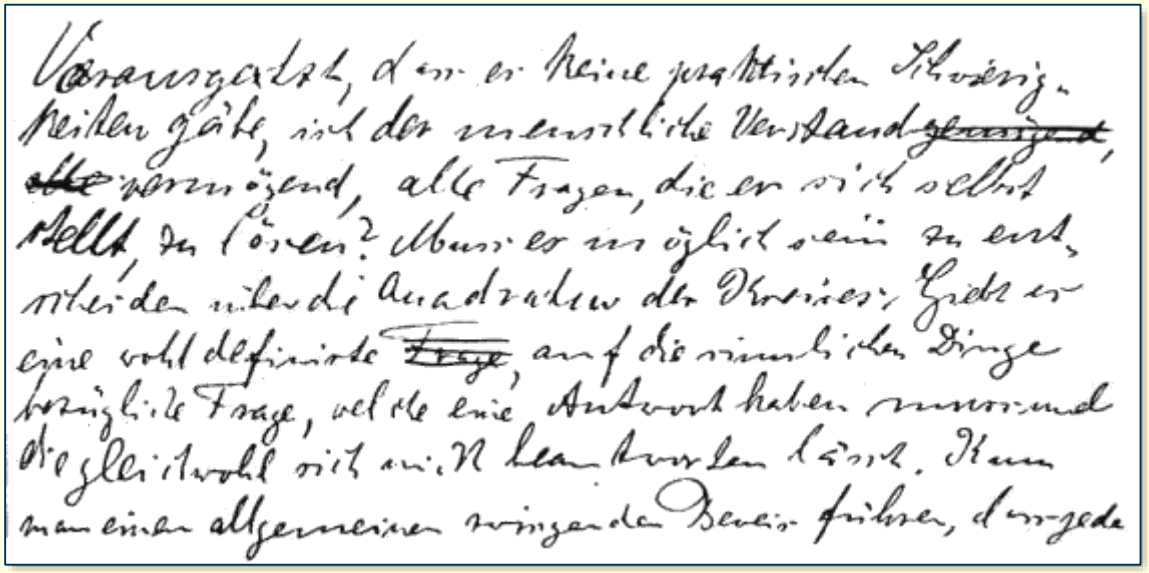
In Hilberts Nachlass, der in der Göttinger Universitätsbibliothek aufbewahrt wird, finden sich Notizbucheinträge, die zwischen 1888 und 1890 entstanden sein dürften, und aus denen hervorgeht, dass Hilbert schon damals die Frage beschäftigte, ob alle klar formulierten mathematischen Probleme „beantwortet“ werden können. Er artikuliert dies so:

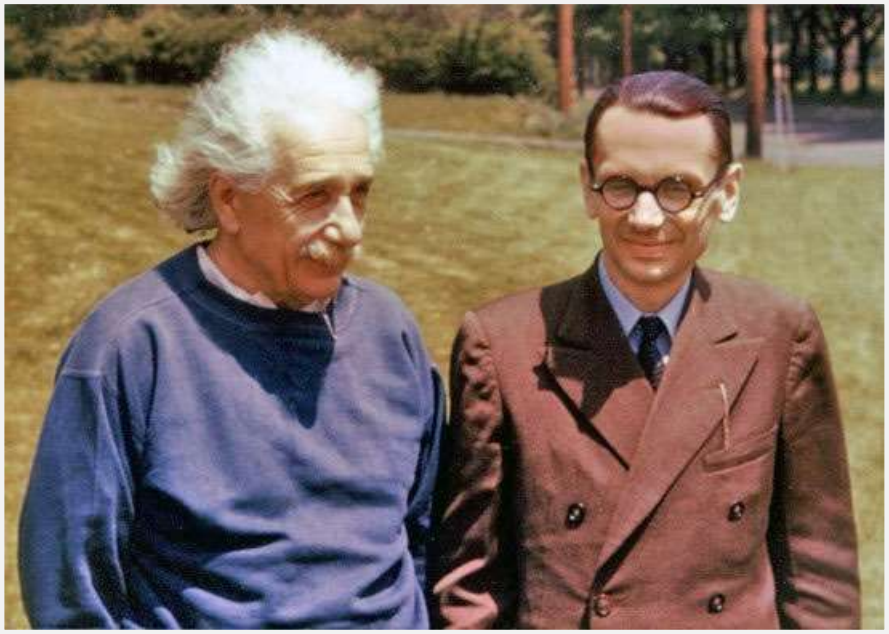
„Vorausgesetzt, dass es keine praktischen Schwierigkeiten gäbe, ist der menschliche Verstand vermögend, alle Fragen, die er sich selbst stellt, zu lösen? Muss es möglich



Wenn ich nach einem tausendjährigen Schlaf erwachte, wäre meine erste Frage: „Ist die Riemann-Hypothese bewiesen?“

sein zu entscheiden über die Quadratur des Kreises; giebt es eine wohl definirte, auf die sinnlichen Dinge bezügliche Frage, welche eine Antwort haben muss und die gleichwohl sich nicht beantworten lässt? Kann man einen allgemeinen zwingenden Beweis führen, dass jede Erkenntniss möglich sein muss?“

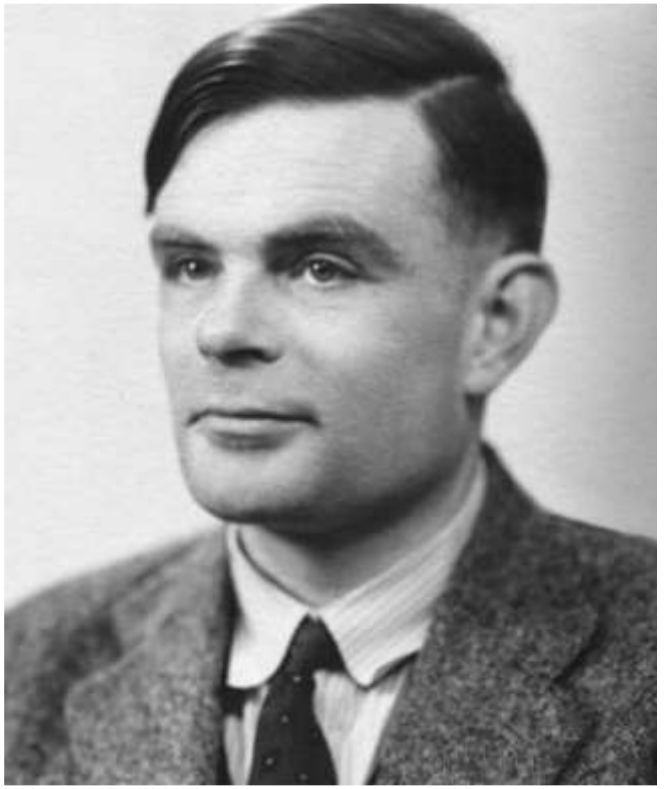




Der Schock geschah dann 1931, als Kurt Gödel (1906 – 1978) in einer epochalen Arbeit („Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme“) zeigt, dass in jedem (hinreichend starken) formalen System ein Satz existiert, der innerhalb dieses Systems weder beweisbar noch widerlegbar ist. Dies gelang ihm, indem er alle mathematischen Aussagen effektiv „durchnummerierte“ (was heute als „Gödelisierung“ bezeichnet wird) und analog zu Cantors bekanntem Diagonalver-

Gödel (re.) mit seinem Freund Albert Einstein, fotografiert von Oskar Morgenstern, ca. 1948

fahren einen Widerspruch durch Selbstbezug konstruierte. 1936 wurde dann die Unentscheidbarkeit des von Hilbert so genannten „Entscheidungsproblemes“ von Alan Turing (und auf etwas andere Art unabhängig auch von Alonzo Church) gezeigt. Turing führte in seiner berühmten Veröffentlichung „On Computable Numbers, with an Application to the ‘Entscheidungsproblem’“, in der er auch die nach ihm benannte abstrakte Turing-Maschine konstruierte, das Entscheidungsproblem auf das Halteproblem dieser Maschinen zurück und zeigte, dass die Frage, ob eine solche Maschine anhält, unentscheidbar ist. Mit den Erkenntnissen von Gödel, Turing und Church schien Leibniz‘ grosser Calculemus-Traum von der „Streitschlichtung und Entscheidung durch einfaches Nachrechnen“ ausgeträumt. (ETH-Professor Erwin Engeler merkt an: „Zum Glück, meine ich, und nicht überraschend zudem. Denn Mathematik ist ein zentrales Kulturgut, und wer würde schon glauben, dass Kultur programmierbar sei. Es sei denn ein Ideologe.“)



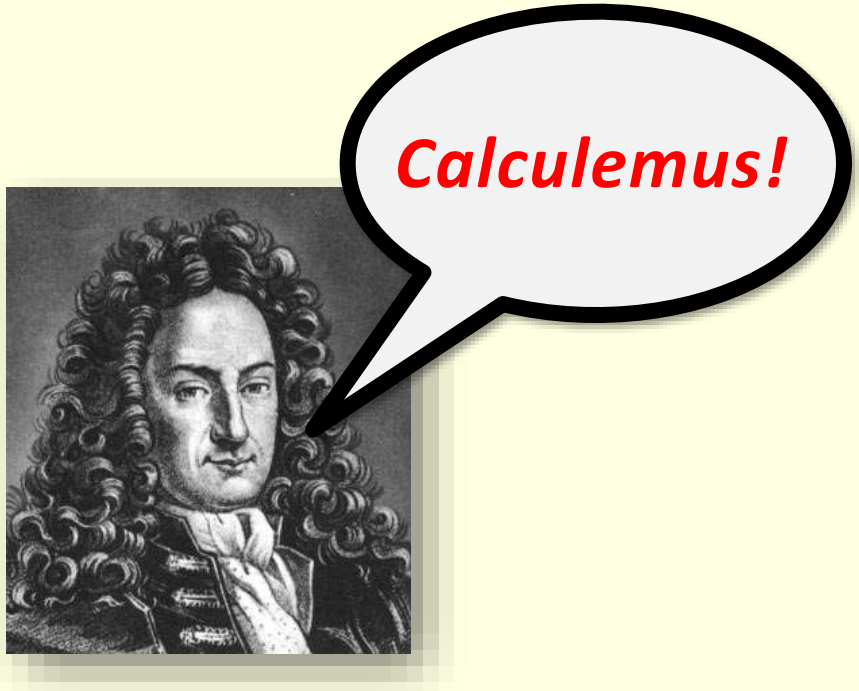
A. M. Turing *(1912 – 1954)*

So einschneidend und fundamental diese negativen Ergebnisse auch sind – sie bedeuten allerdings nicht, dass das automatische Beweisen der Korrektheit mathematischer Sätze (und auch der Korrektheit von Programmen!) im Allgemeinen unmöglich ist. Der Prädikatenkalkül ist nämlich „semi-entscheidbar“ – das heisst, wenn eine Aussage wahr ist, dann kann dies auch in

endlich vielen Schritten nachgewiesen werden. (Sollte die Aussage hingegen falsch sein, dann gibt es zwei Möglichkeiten: Entweder ein Nachweis der Inkorrektheit gelingt im konkreten Fall trotzdem, oder das automatische Verifikationsprogramm läuft ohne je zu terminieren weiter und man weiss es eben nicht: Kommt vielleicht doch noch irgendwann ein positives Ergebnis oder sollte man lieber aufgeben, weil man ja nicht ewig warten kann?)

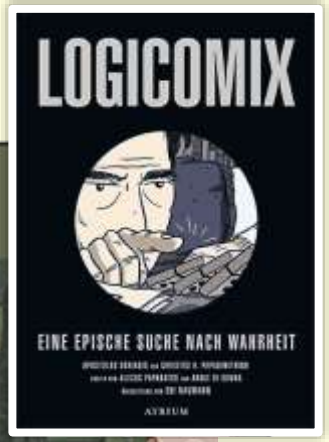
Mit dem Aufkommen elektronischer Digitalrechner wurden ab den 1950er-Jahren Computerprogramme zum Beweisen mathematischer Sätze entwickelt, die die Regeln der Kalküle automatisch (und möglichst zielführend) anwenden. Der historisch erste computergenerierte Beweis einer mathematischen Aussage soll übrigens der Satz gewesen sein, dass die Summe zweier gerader Zahlen wieder eine gerade Zahl ist. Die Theorie und Praxis automatischer Theorembeweiser und Deduktionssysteme (sowie automatischer Programmverifikation als spezielle Anwendung hiervon) ist heute ein wichtiges Teilgebiet der künstlichen Intelligenz.

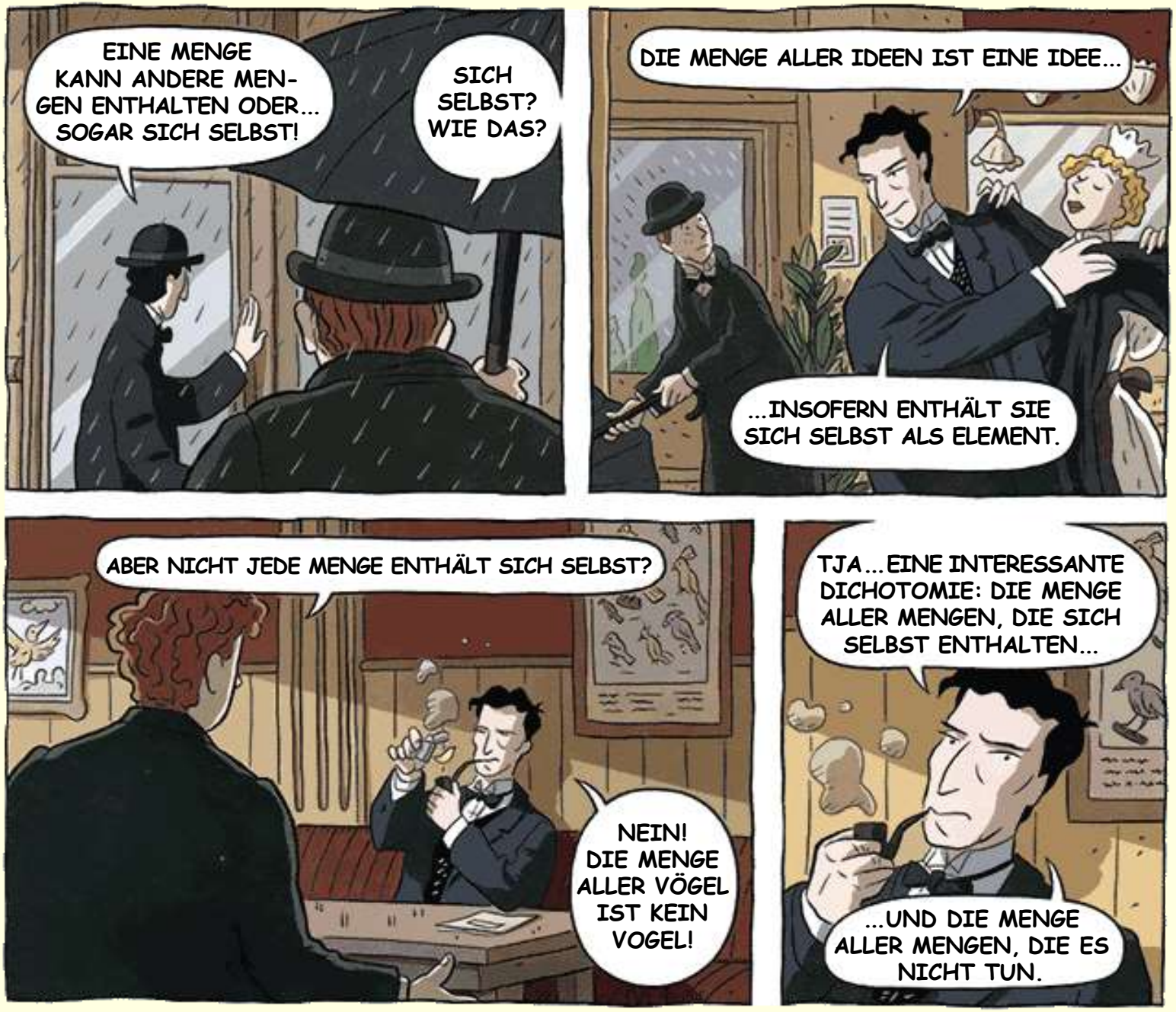
Leibniz‘ über 300 Jahre alte Calculemus-Vision ist in ihrem umfassenden Anspruch so zwar nicht ganz wahr geworden – aber sein „Forschungsprogramm“ dazu wurde von späteren Generationen aufgegriffen und hat viele bedeutende Erkenntnisse hervorgebracht sowie neue Teildisziplinen in der Wissenschaft begründet. Leibniz, der Schutzpatron der Informatik, war also nicht nur mit seinen Rechenmaschinen und dem Dualsystem, sondern auch mit der Idee einer Formalisierung sowie Mechanisierung menschlicher Intelligenzleistung seiner Zeit weit voraus und einer ganz spannenden Sache auf der Spur!



Lesenswert ist in diesem Kontext die „graphic novel" Logicomix, die auch auf Deutsch erschienen ist.

Sie berichtet durch den Erzähler Bertrand Russell vom Bemühen, die Logik Ende des 19. und in der 1. Hälfte des 20. Jhd. zu formalisieren. Unter anderen treten auf: Georg Cantor, Ludwig Wittgenstein, Alfred North Whitehead, David Hilbert, Gottlob Frege, Kurt Gödel, Henri Poincaré und Christos Papadimitrio. (Griechische Originalausgabe 2008; 2010 auf Deutsch.)





Als Beispiel hier die Einführung der Russellschen Antinomie $R := \{x \mid x \notin x\}$ im Comic.

Antinomie: Die Richtigkeit der Aussage hat ihre eigene Falschheit zur Folge und umgekehrt. In formalen Systemen führt dies zu Inkonsistenzen und damit zur völligen Unbrauchbarkeit der zugrunde liegenden Axiome und Schlussregeln.

„Die Menge aller Mengen, die sich nicht selbst als Element enthalten“ stellt eine logische Antinomie dar, die unabhängig von konkreten Mengenaxiomen gilt. Daher hat sie besonders kräftig erschüttert und schlagartig das Ende der naiven Mengenlehre herbeigeführt.

Es gibt eine Reihe populärer Varianten der Russellschen Antinomie. Am bekanntesten ist das Barbier-Paradoxon (*man kann einen Barbier als jemanden definieren, der genau diejenigen rasiert, die sich nicht selbst rasieren; die Frage ist: rasiert der Barbier sich selbst?*), mit dem Russell selbst 1918 seinen Gedankengang veranschaulichte.

Der Barbier ist eine Frau

Hätte der Barbier auch selbst lösen können, indem er sich einen Vollbart wachsen lässt – dazu braucht es keine Frau. -- H. Danisch.

Die türkischen Barbiere in meiner Straße rasieren sich gegenseitig. -- N.N. Anon.

Johanna Hartmann: *Barbiere und sonstiger Zwist in der Mathematik um 1900. Easy – wie man in der Anerkennung gesellschaftlicher Realitäten mathematische Probleme löst.* In: Bulletin – Texte 32, Humboldt-Universität Berlin, ISSN 0947-6822, Sep. 2006, www.gender.hu-berlin.de/de/publikationen/gender-bulletin-broschueren/bulletin-texte/texte-32/texte32pkt2.pdf (Auszug)

[...] Die Frage, der ich in dieser Arbeit nachgehen möchte, ist die, ob die „Krise" der Mathematik als Krise von Männlichkeit gelesen werden kann. [...] Ein Blick, der nicht für die mathematischen Ausdrucksweisen geschult ist, kann Phänomene als auffällig wahrnehmen und ins Zentrum der Untersuchung rücken, die einer mathematischen Betrachtung möglicherweise nicht ins Auge fallen. [...]

Männlichkeit und Objektivität werden gleichgesetzt in der Darstellung eines mathematischen Wissens, das unabhängig von Produktionsformen entsteht in einer Welt, in der sich mathematische Gesetzmäßigkeiten dem Wissenschaftler zeigen, der sie nicht produziert, sondern entdeckt. [...] Die Mathematik kann in der „Grundlagenkrise" ihren Anspruch nach einer aus sich selbst erklärenden Wahrheit, nach Universalität, Zeitlosigkeit und Übertragbarkeit nicht mehr aufrecht erhalten. Das männliche Erkenntnissubjekt verliert seine Position eines objektiven Erkennens der Wahrheit und ist insofern in der Krise. [...]

Die Erschütterung der Mathematik in ihren Grundfesten droht die unangezweifelte, unangreifbare, unsichtbare und überlegene Position von Männlichkeit in Frage zu stellen. All dies verdeutlicht sich im Beispiel des ungelösten Problems des Barbiers in der Erläuterung der mengentheoretischen Antinomie. Die Lösung dafür ist simpel, aber dennoch (bisher) nicht innerhalb des Rahmens mathematischer Lösungsvorschläge möglich. Auch wenn es sich bei dem Beispiel mit dem Barbier lediglich um ein Bild handelt [...], so ist dieses von Mathematikern gewählte Bild doch ein äußerst treffendes, das die Begrenztheit der Lösungsvorschläge [...] auf Männlichkeit als Lösung verdeutlicht. Für die Lösung muss der Schritt aus der gegenseitigen Legitimierung und Stabilisierung der Konstrukte von Mathematik und Männlichkeit gemacht werden. Der Barbier ist eine Frau. Easy.

Leibniz' Traum: Automatisiertes Denken als Vorbote künstlicher Intelligenz

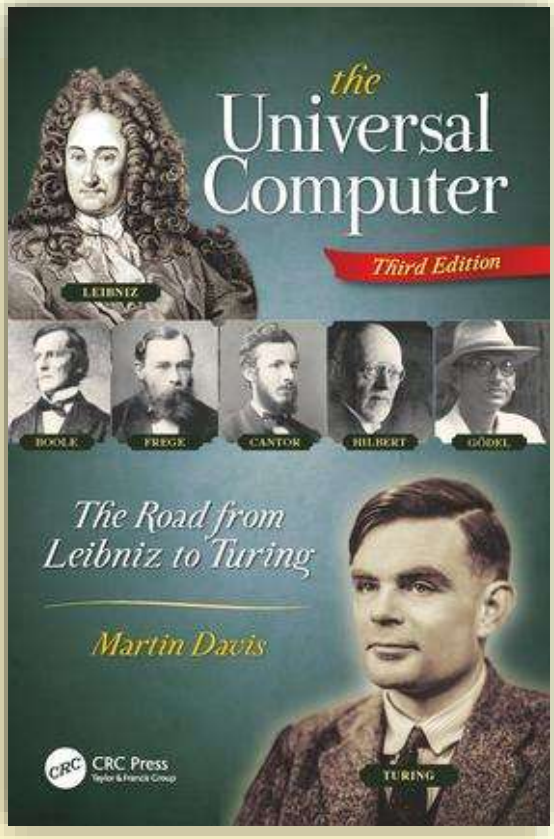
Witold Marciszewski schreibt in seinem Essay „Leibniz's Idea of Automated Reasoning Compared with Modern AI" u.a.:

While Hilbert's contention was concerned with mathematics alone, Leibniz believed that all scientific and philosophical problems can be definitely solved in a foreseeable time. In this respect, he was confident like Descartes. However, while Descartes attributed the power of reasoning to the mind alone, and discounted linguistic devices, Leibniz extended that power to the mind-imitating machines equipped with a suitable symbolic system.

Leibniz believed that his arithmetical machine is just the beginning of a development that should result in logical machines to match humans in the ability of reasoning. And that, in principle, there are in science and philosophy no unsolvable problems, either for men of for logical machines, but in practice (he presumably thought) the machines should act better (as carefully equipped for their cognitive tasks).

Leibniz's stress on the sensibility and palpability of characters used in reasoning resembles the formalistic point of Hilbert, and his belief expressed in the famous *Calculemus* is to the effect that every demonstration can be performed in a finite number of steps.

Leibniz' Traum: Ein Buch zu seiner langen Geschichte



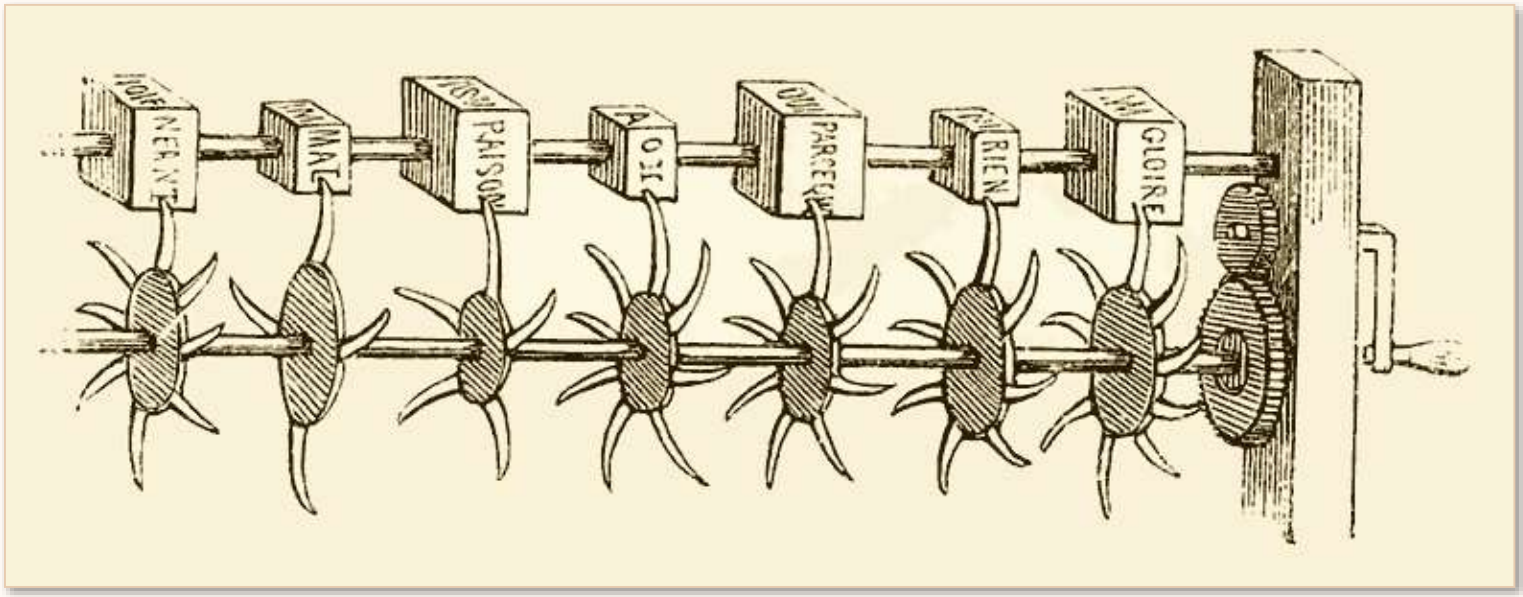
Written by Martin Davis, respected logician and researcher in the theory of computation, *The Universal Computer: The Road from Leibniz to Turing* (3rd ed., 2018) explores the lives, ideas, and discoveries of seven remarkable mathematicians.

"The story is told in seven chapters, each devoted to a character whose work profoundly influenced further developments. It begins in the seventeenth century with Gottfried Wilhelm Leibniz and his symbolic infinitesimal calculus, his work on mechanical computing machines, and his "wonderful idea" of a symbolic language whose alphabet would represent concepts to which formalized rules of deduction could be applied in a mechanical fashion to verify the validity of arguments. Then we move to George Boole and his algebra of truth values, and to Gottlob Frege and the system of first-order logic. In the next three chapters, we reach the dawn of the modern era in foundations of mathematics: Georg Cantor's set theory and diagonal arguments, and David Hilbert's program and its collapse under the weight of Kurt Gödel's incompleteness theorems. Finally, in Chapter 7, we meet the young Alan Turing, and we follow his life and the ideas that led directly to the modern understanding of mechanical computation and its limits.

Davis mixes anecdotes from the professional and private lives of the great thinkers with lucid explanations of their central ideas. The history of the ideas is interesting on its own, but it becomes quite fascinating when combined with insights into the personal lives of the philosophers and mathematicians who have made that history. It is hard not to notice how special those characters were, and how much of their pioneering work was made outside, and often against, the mainstream." [Roman Kossak, The Mathematical Intelligencer, June 2019, 41: 78-79]

Jonathan Swift (1667 – 1745), 21 Jahre jünger als Leibniz, parodiert in seinem Roman „Gullivers Reisen“ (1726) die Ideen zahlreicher zeitgenössischer Wissenschaftler, darunter auch die Bemühungen von Lullus und Leibniz, auf mechanische Weise zu Wissen zu gelangen. Im dritten Teil seines satirischen Romans (nachdem im ersten Teil Gulliver bei den Liliputanern war, im zweiten das Land der Riesen besuchte) reist Gulliver zur Akademie von Lagado. Er entdeckt eine Vielzahl von Absurditäten; unter anderem jemanden, der versucht, Sonnenlicht aus Gurken zu extrahieren, oder einen Architekten, der ein Haus „top down“, also vom Dach her nach unten, bauen will, oder einen Professor, der meint, Frauen sollten nach ihrer Schönheit und nach ihrer Geschicklichkeit in der Bekleidung besteuert werden.

Dann trifft Gulliver einen Gelehrten, der die spekulativen Wissenschaften durch praktische und mechanische Operationen zu verbessern trachtet: „Er schmeichelte sich mit dem Gedanken, dass eine höhere und edlere Idee noch nie aus dem Gehirn eines Menschen entsprungen sei. Ein jeder wisse, wieviel Mühe die gewöhnliche Erlernung der Künste und Wissenschaften bei den Menschen erfordere; er sei überzeugt, durch seine Erfindung werde die ungebildetste Person bei mässigen Kosten und bei geringer körperlicher Anstrengung Bücher über Philosophie, Poesie, Mathematik und Theologie ohne die geringste Hilfe des Genies oder von Studien schreiben können. Er führte mich an einen Rahmen, wo alle seine Schüler in Reihen aufgestellt waren. Der Rahmen enthielt zwanzig Quadratfuss und befand sich in der Mitte des Zimmers. Die Oberfläche bestand aus einzelnen Holzstücken von der Form eines Würfels, von denen jedoch einzelne grösser als andere waren. Sie waren sämtlich durch leichte Drähte miteinander verbunden. Diese Holzstücke waren an jeder Fläche mit Papier überklebt, auf dem alle Worte [...] ohne alle Ordnung aufgeschrieben waren. Der Professor bat mich, achtzugeben, da er seine Maschine in Bewegung setzen wolle. Jeder Zögling nahm auf seinen Befehl einen eisernen Griff zur Hand, von denen vierzig am Rande befestigt waren. Durch eine plötzliche Wendung wurde die ganze



Die Wortkombinationsmaschine der Akademie von Lagado, Details des Getriebes. [Voyages de Gulliver dans des contrées lointaines par Swift, Edition illustrée par Grandville, Paris, 1838]

Anordnung verändert. Dann befahl er sechzehn Knaben, die verschiedenen Zeilen langsam zu lesen, und wenn sie drei oder vier Worte herausgefunden hatten, die einen Satz bilden konnten, diktierten sie diese vier anderen Knaben, welche sie niederschrieben. [...]

Der Professor zeigte mir mehrere Folianten, die auf diese Weise aus abgebrochenen Sätzen gebildet waren und die er zusammenstellen wollte. Aus diesem reichen Material werde er einen vollständigen Inbegriff aller Wissenschaften und Künste bilden. [...] Er gab mir die Versicherung, diese Erfindung habe schon von Jugend auf alle seine Gedanken in Anspruch genommen; er habe seinen Rahmen so eingerichtet, dass er den ganzen Sprachreichtum umfasse. [...] Ich bezeigte dieser ausgezeichneten Person meinen demütigsten Dank für seine grosse Güte, mir die ganze Erfindung mitzuteilen, und [...] sagte ihm: Obgleich es in Europa bei Gelehrten Gewohnheit sei, sich einander die Erfindungen zu stehlen, wodurch sie den Vorteil hätten, dass wenigstens ein Streit über das Eigentum stattfinde, so werde ich doch mit allem Eifer darauf hinwirken, dass er, ohne irgendeinen Nebenbuhler, die Ehre an seiner Erfindung ausschliesslich erlange.“ □

Kurze Leseprobe aus: **Umberto Eco: Die Insel des vorigen Tages** (1994) [L'Isola del Giorno Prima]

Deshalb hatten sie ein großes Rad konstruiert, das sie [...] aufrecht auf den Dorfplatz gestellt hatten. Es bestand aus sechs konzentrischen Kreisen, die sich jeder für sich drehen ließen. Der erste war in vierundzwanzig Felder geteilt, der zweite in sechsunddreißig, der dritte in achtundvierzig, der vierte in sechzig, der fünfte in zweiundsiebzig und der sechste in vierundachtzig. In den Feldern standen geschrieben [...] Tätigkeiten wie Gehen, Kommen oder auch Sterben, Leidenschaften wie Hassen, Lieben oder auch Frieren, dazu Modalitäten wie gut und schlecht, traurig oder fröhlich und schließlich Orts- und Zeitangaben wie zu Hause oder nach einem Monat.

Wenn man diese Räder nun drehte, erhielt man Geschichten wie «Gestern ging er nach Hause und traf seinen Feind, der Schmerzen litt, und brachte ihm Hilfe» oder «Er sah ein Tier mit sieben Köpfen und tötete es». Die Insulaner behaupteten, dass man mit dieser Maschine siebenhundertzweiundzwanzig Millionen Millionen verschiedene Geschichten schreiben oder denken könne, und das sei genug, um dem Leben eines jeden von ihnen in allen kommenden Jahrhunderten einen Sinn zu geben. [...]

Pater Emanuele sagte, er sei gerade im Begriff, den Besuchern seine Aristotelische Maschine zu zeigen, woraufhin er sie alle drei in einen Raum führte, in dem sich das sonderbarste Möbel befand, das man sich vorstellen kann. [...] Der untere Teil bestand aus einer Kommode, in deren Vorderseite einundachtzig Schubladen eingelassen waren – neun waagrechte Reihen auf neun senkrechte, jede Reihe oben und an der Seite, wie bei einem Schachbrett, beschriftet mit einem Buchstaben in der Abfolge BCDEFGHIK. Oben auf der Kommode stand links ein Lesepult, auf dem ein großes Buch lag, eine aufgeschlagene Handschrift mit kolorierten Initialen. Rechts neben dem Pult befanden sich drei ineinandergesteckte zylinderförmige Walzen von abnehmender Länge und zunehmendem Umfang (wobei die kürzeste die geräumigste war, so dass sie die beiden längeren in sich aufnehmen konnte), die mit einer rechts angebrachten Kurbel so gedreht werden konnten, dass sie sich aufgrund des Trägheitseffektes mit unterschiedlicher Geschwindigkeit je nach ihrer Schwere ineinander drehten. Jede Walze trug am linken Ende die gleichen neun Buchstaben eingraviert, die auch die Schubladen bezeichneten. Es genügte, die Kurbel einmal zu drehen, und die Walzen setzten sich unabhängig voneinander in Bewegung, und wenn sie wieder zum Stillstand gekommen waren, konnte man Triaden von zufällig zusammengestellten Buchstaben lesen, CBD, KFE oder BGH. Pater Emanuele begann das Prinzip zu erklären, das seine Maschine beherrschte. [...] □

Leibniz-Trivia

Es gibt viele Menschenrechte, dazu gehört aber nicht das Recht, dass man nicht verspottet werden darf. – Ernst Horst

1966, zum 250. Todesjahr von Leibniz, schrieb „Der Spiegel" in der Rubrik «Philosophie» u.a.: „Mit 13 Jahren schrieb er ein Pfingstgedicht in 300 lateinischen Hexametern, mit 14 bezog er die Universität, mit 16 bestand er das Baccalaureus-Examen, mit 20 habilitierte er sich, mit 21 war er Hofjurist in Mainz. In Paris als Scheidungsanwalt tätig, begann er erst als 26-jähriger, sich mit der Mathematik zu befassen. Unabhängig von Newton fand er den Infinitesimalkalkül. Sein geradezu gefräßiger Forschungseifer beschäftigte sich mit der Konstruktion von Türschlössern, der Entzifferung von Geheimschriften, der Entwicklung von U-Booten wie mit Gottesbeweisen und der Erfindung einer «künstlichen» Sprache. Preußens Soldatenkönig Friedrich Wilhelm I. hielt ihn für einen Kerl, der nicht einmal zum Schildwachestehen nütze sei. Die meisten Bürger Hannovers halten Leibniz für den Erfinder der gleichnamigen Kekse."

Der bekannte Butterkeks (mit 14 Zähnen in der Länge und 10 Zähnen in der Breite) der Firma Bahlsen aus Hannover wurde 1891 tatsächlich nach Gottfried Wilhelm Leibniz benannt. („Was ißt die Menschheit unterwegs? Na selbstverständlich Leibniz Cakes!" lautete seinerzeit ein Reklameslogan.) Allerdings hatte **Leibniz selbst nichts damit zu tun und hat ihn schon gar nicht „erfunden". (Leibniz hatte seinerzeit zwar nach einem haltbaren Produkt zur Verpflegung von Soldaten gesucht, dafür aber Zwieback favorisiert.) Das Wort „Keks" wurde übrigens nach dem Englischen „cakes" für „Kuchen" gebildet. Der Duden schrieb 1915 dazu: „Diese Eindeutschung des engl. cake ist annehmbar, aber es muß in der Einzahl Kek gesagt werden, nicht Keks." (In der Schweiz bzw. im Alemannischen hat sich „Keks" nicht durchgesetzt, hier ist dafür das Wort „Guetzli" oder auch „Biscuits" gebräuchlich, in süddeutschen Dialekten auch die Varianten „Gutsle" oder „Guatzl").**

Leibniz-Trivia (2) Leibniz-Uni geht Bahlsen auf den Keks

134 **Wirtschaft: Nachrichten**

Wem gehört Leibniz?

Der Bahlsen-Konzern in Hannover sieht seine Markenrechte verletzt und zieht im Namensstreit gegen die Universität Hannover vor Gericht. Die nach Gottfried Wilhelm Leibniz (1646-1716) benannte Hochschule hatte sich die Webdomain "Leibniz-Shop" in allen möglichen Schreibweisen gesichert und den Namen beim Patentamt angemeldet.

Die Bahlsen-Gruppe mit einem Jahresumsatz von rund 552 Mio. Euro sieht eine Verwechslungsgefahr mit ihrer Marke Leibniz, die sie schon 1897 hatte schützen lassen. Die Universität wollte nicht Stellung nehmen.

ARD Teletext, 15.09.2017

Leibniz Universität Hannover

Leibniz – Gelehrter oder Gebäck?

Kann man einen Philosophen der Aufklärung und ein Traditionsgebäck verwechseln? Ja, sagt der Hersteller der Leibniz-Kekse - und klagt gegen die Leibniz-Universität in Hannover. Wer jedoch im Onlineshop der Uni nach den Schlagworten „Kekse“ oder „Gebäck“ sucht, erhält lediglich die Meldung: „Zu dem angegebenen Suchbegriff konnten keine Artikel gefunden werden.“ Angeboten werden stattdessen T-Shirts, USB-Sticks oder Schreibgeräte mit dem Hochschullogo.

www.spiegel.de

Leibniz-Trivia (3) Leibniz-Uni geht Bahlsen auf den Keks

...Forderung der Firma Bahlsen, dass sie aufgrund ihrer Marke für Kekse o.ä. allein den Namen Leibniz in einer Domain und einem Internetshop beanspruchen dürfe, und zwar unabhängig vom angebotenen Produktsortiment.

...überzeugt uns die Vorstellung, die Figur oder das Wort Leibniz sei automatisch in der Wahrnehmung der Öffentlichkeit mit einem Keks assoziiert, nicht. Sie würde auch dem Leben und Wirken von Leibniz als Gelehrtem nicht gerecht.

Aus dem „Statement der Hochschulleitung der Leibniz Universität Hannover zur Klage der Firma Bahlsen gegen die Universität" vom 19. September 2017

Leibniz (am 27. Nov. 1711, im Alter vom 65)

„...hoffen, meine implicationes fündeten in futuri ihr voll entffaltung“

...versprach ich einen *abacus universalis*, mit dem man könnt den *spiritus entlasten*. dieweil hab ich, seit dazumahlen in paris ich die gerätheschaft ersann, noch immer nicht den rechten opifex automatarius gefunden, deßen räderwerke den winndungen der ratio paribus wärn: Meyn mechanicus Adam erweiset sich, trotz laurus nobilis, als entteuyscherung monumentalis. Meine *mathe= matica dualis*, seit jahro 1679 conceptionirt, war bereyts quasi auffgegeben, so schwër gerietet alleyn die Machina in decimalis. Und doch konnt ich underweysen, in unsrer «Explication de l'arith= métique binaire», editum 1703 in den Mémoires de l'Académie royale, *daz auch der Chinëse die dyadica kennt* (ich hätt die explicatio lieber teutsch geschriebn, alleyn der auslender hätt's nicht können verstünden). Kunde erreichtt mich von der Missio Societate Iesu in Peching, mit der ich pflege regelmessige corres= pondentia, dass der keyserlüche monarch Fo-hi, eyn gar großer Liebhaber der Rëchten Kunst, dyadisch componiret. *Einz - null*,... Obschohn ich fürderhin *hoffete*, *daz meine dyadica wird eynes tages practick werden*, in arithmeticis et mechanicis. Meine kreffte schwinden hinfort und ob man wohl sagen möchtt es wär vergebens den Stall zu schließen, wenn die pferde geraubet, so will ich dennoch *inständigst hoffen, meine implicationes fün= deten in futuri ihr voll entffaltung.* Leibniz

Ende der historischen Notiz zu Leibniz

Fragen zur altägyptischen Multiplikationsmethode

- Funktioniert das immer?
  - Was heisst „immer“? (z.B. negative Zahlen?)
  - Wenn nicht: wann?
- Warum funktioniert es?
  - Wie beweist man die Korrektheit?
- Ist das besser als die „Schulmethode“?
  - Was heisst „gut“?
  - Lässt sich Güte linear anordnen?
- Wie kommuniziert man das Verfahren unmissverständlich?
  - In welcher Notation / Sprache?

Kernaspekte der Informatik!

Wieso funktioniert die Methode?

- Beobachtung bei **3 × 18**:

a	b
~~3~~	~~18~~
6	9
~~12~~	~~4~~
~~24~~	~~2~~
48	1
54	

Die erste Zeile wird sowieso gestrichen

Gleiches Ergebnis, als hätte man 6 × 9 (statt 3 × 18) berechnet

a	b
6	9
~~12~~	~~4~~
~~24~~	~~2~~
48	1
54	

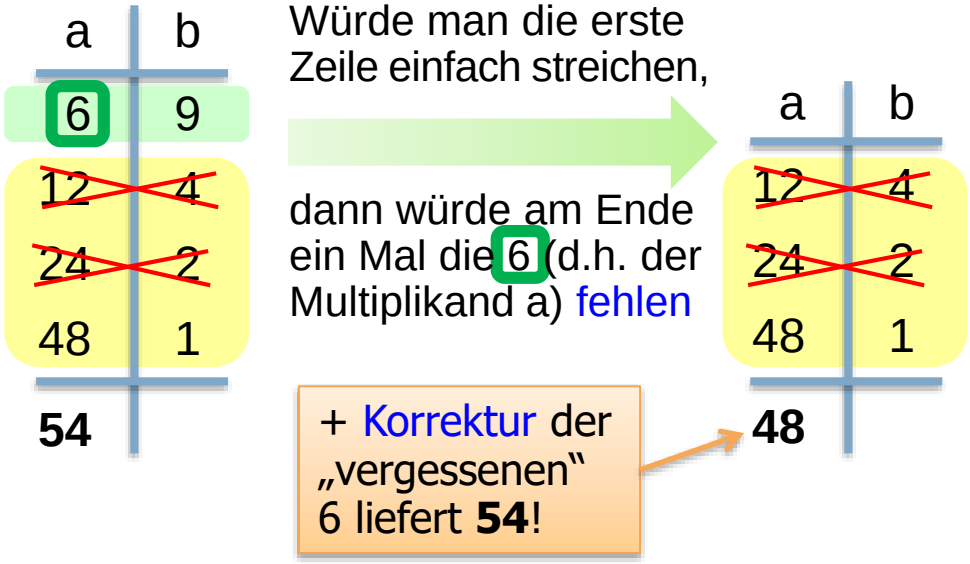
- Also: a × b wird auf 2a × b/2 zurückgeführt, falls b gerade
  - a × b = 2a × b/2 ist offenbar richtig
  - Bei geradem b darf man die erste Zeile also einfach streichen → Tabelle wird kürzer!
- Aber bei ungeradem b?

Denkübung: Was geschieht, wenn b eine Zweierpotenz ist?

Denkübung: Man wandle Multiplikand und Multiplikator in das Dualsystem um (z.B. 3 → 11 und 18 → 10010) und multipliziere „schriftlich“: Welchen Bezug zur altägyptischen Multiplikationsmethode erkennt man?

Ungerade Multiplikatoren

- Beispiel $a \times b = \mathbf{6 \times 9}$:



- Also: Wenn der Multiplikand b ungerade ist, muss man den Multiplikator **a** zum Ergebnis (des in der 2. Zeile spezifizierten Multiplikationsproblems) hinzuaddieren

- Rechtfertigung:

$$a \times b = \mathbf{a} + (2a \times (b-1)/2)$$

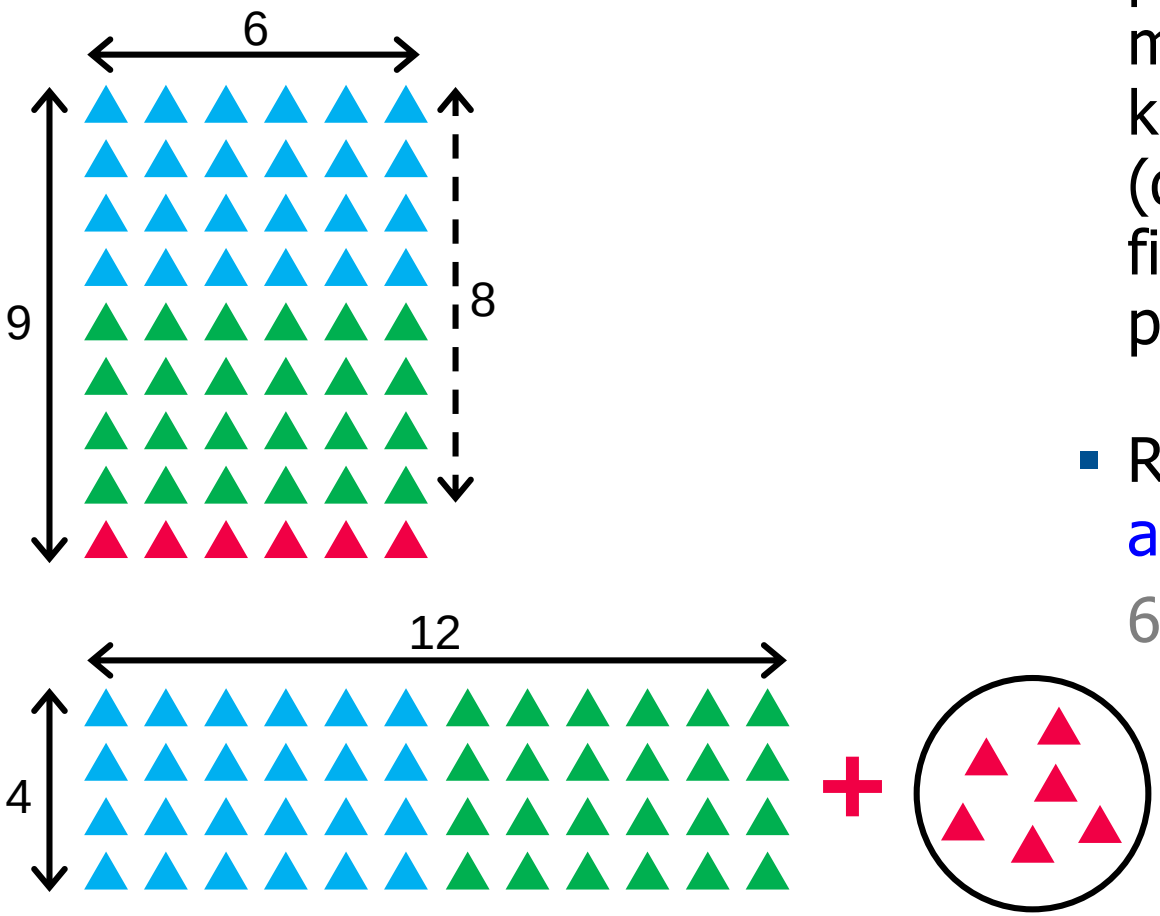
$$6 \times 9 = 6 + (12 \times \underbrace{(9-1)/2}_{4})$$

Wir haben also **6 × 9** durch **12 × 4** „approximiert“

Bzw. durch **6 × 8** (indem 9 zu 8 „abgerundet“ wurde)

Ungerade Multiplikatoren

- Beispiel $a \times b = \mathbf{6 \times 9}$:



- Also: Wenn der Multiplikand b ungerade ist, muss man den Multiplikator **a** zum Ergebnis (des in der 2. Zeile spezifizierten Multiplikationsproblems) hinzuaddieren

- Rechtfertigung:
  $a \times b = \mathbf{a} + (2a \times (b-1)/2)$
  $6 \times 9 = 6 + (12 \times (9-1)/2)$

Problemreduktion

- Es gilt $a \times b = \begin{cases} 2a \times \frac{b}{2} & \textit{falls b gerade} \\ a + \left(2a \times \frac{b-1}{2}\right) & \textit{falls b ungerade} \end{cases}$

- Was nützt diese leicht zu beweisende Eigenschaft?

➔ Wenn Duplizieren und Halbieren triviale Operationen sind, dann kann die Multiplikation zweier Zahlen auf ein „einfacheres" Multiplikationsproblem zurückgeführt werden

- 12 × 4 ist „einfacher" als 6 × 9, da mit 4 zu multiplizieren einfacher ist als mit 9
- „Einfacher" heisst, näher bei der 1: Mit 1 zu multiplizieren, ist ganz einfach („trivial")!

Endlose Reduktion?

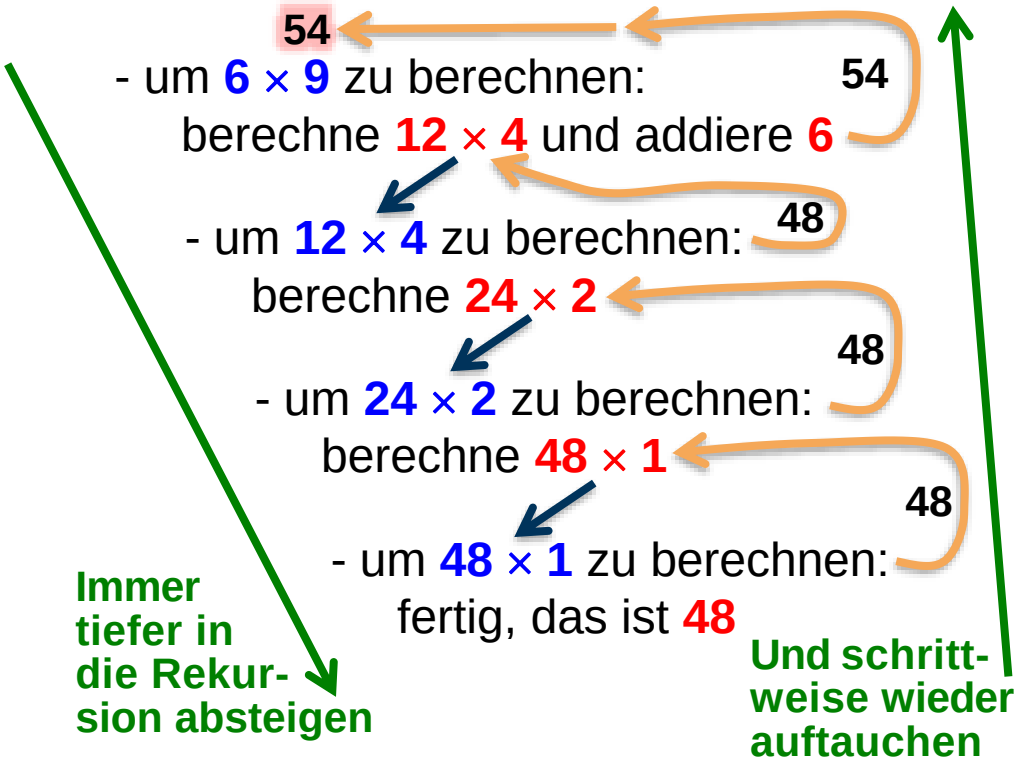
- Wir müssen aber noch festlegen, wann man mit der Problemreduktion am Ende ist
  - Eine endlose Reduktion ist schliesslich nicht praktikabel
  - Wir hören auf, wenn das Problem (hoffentlich) trivial geworden ist
- Hier: wenn b = 1 ist
  - Denkübung: Wieso nicht bei b = 0 (wäre doch noch einfacher!)?
- Wir schreiben daher: $a \times b = \begin{cases} a & \textit{falls } b = 1 \\ 2a \times \frac{b}{2} & \textit{falls } b \textit{ gerade} \\ a + \left(2a \times \frac{b-1}{2}\right) & \textit{sonst} \end{cases}$

Rekursion

To understand recursion, one must first understand recursion

- Wir haben ein Problem auf eine einfachere „Instanz“ des gleichen Problems zurückgeführt („Rekursion“)

- Im Beispiel sieht das etwa so aus:



Die Multiplikation könnte als rekursive Funktion f daher so definiert werden:

$$f(a,b) = \begin{cases} a & \text{, falls } b = 1 \\ f(2a, b/2) & \text{, falls } b \text{ gerade} \\ a + f\left(2a, \frac{b-1}{2}\right) & \text{, sonst} \end{cases}$$

Denkübung: Mathematisch exakte Rechtfertigung für Korrektheit?
(Und für welchen Definitionsbereich?)

Der Algorithmus in Java

$$f(a,b) = \begin{cases} a & \text{, falls } b = 1 \\ f(2a, b/2) & \text{, falls } b \text{ gerade} \\ a + f\left(2a, \frac{b-1}{2}\right) & \text{, sonst} \end{cases}$$

Das interessiert uns jetzt nicht

„f“: Name der Methode (≙ Funktion in C / C++)

Das Ergebnis und die beiden Parameter sind ganze Zahlen

Zugehörige schliessende Klammer „}“ am Ende

```
static int f(int a, int b){
  System.out.println(a + "  " + b);
  if (b == 1)   return a;
  if (b%2 == 0) return f(a+a, b/2);
     else return a + f(a+a, (b-1)/2);
}
```

Jedes Mal, wenn f „aufgerufen“ wird, werden die Parameter ausgegeben

Modulus-Operator % liefert den Rest bei der ganzzahligen Division

Denkübung: Kann man das Schlüsselwort „else“ hier auch weglassen?

Wir nennen die Methode der Kürze wegen hier „f“; eigentlich sollte man aussagekräftigere Bezeichnungen wie z.B. „multAegypt“ wählen

Ein ganzes Java-„Programm“

In Java ist immer alles in Klassen eingepackt; unsere Klasse nennen wir „Mult“

```
class Mult {
  public static void main(String args[]) {
  int i = 5, j = 9;
  System.out.println
   (i + " mal " + j + " ist " + f(i,j));
  }

  static int f(int a, int b){
    System.out.println(a + "  " + b);
    if (b == 1)   return a;
    if (b%2 == 0) return f(a+a, b/2);
          else return a + f(a+a, b/2);
  }
}
```

Zwei Variablen für ganzahlige Werte (hier 5 bzw. 9)

Das Ende der Methode „main“

Aufruf der Methode „f“

Die Methode „f“ von vorhin

Hier steht jetzt aber b/2 statt (b-1)/2: wieso ist das korrekt?

Das Ende der Klasse

Wir brauchen noch ein „Hauptprogramm“, das die Methode f aufruft und das Ergebnis ausgibt

- die Methode, die alles startet, heisst „main“
- um die anderen Dinge in dieser Zeile („public“ etc.) kümmern wir uns jetzt nicht; man schreibe das zunächst immer exakt so hin

Das Ergebnis:

```
 5  9
10  4
20  2
40  1
5 mal 9 ist 45
```

Illegale Eingaben? Terminierung?

- Was würde geschehen, wenn wir diese Zeile vergessen hätten?


```
static int f(int a, int b){
  System.out.println(a + "  " + b);
  if (b == 1)   return a;
  if (b%2 == 0) return f(a+a, b/2);
        else return a + f(a+a, b/2);
}
```

- Oder: Woher wissen wir, dass stets b schliesslich 1 wird?
- Und was geschieht z.B. bei Aufrufen wie f(1,0) oder f(0,5)?
- Oder bei negativen Zahlen?

Ein Experiment: f(1,0)

1	0
2	0
4	0
8	0
16	0
32	0
64	0
128	0
256	0
512	0
1024	0
2048	0

verdoppeln

halbieren

4096	0
8192	0
1048576	0
2097152	0
4194304	0
8388608	0

Klar! 1 mal 0 ist das selbe wie 2 mal 0 ist das selbe wie 4 mal 0 ist das selbe wie …

16777216	0
33554432	0
67108864	0
134217728	0
268435456	0
536870912	0
1073741824	0
-2147483648	0

Das ist eine „prominente" Zweierpotenz

Ach ja, wir haben ja schon in „Informatik I" gelernt, dass man bei einem Überlauf aus dem Wertebereich negative Zahlen erhalten kann und es dabei meist keine Fehlermeldung gibt!

...auf „if (b == 1)" kann man da lange warten...

Ein Experiment: f(1,0)

1	0	4096	0	16777216	0
2	0	8192	0	33554432	0
4	0	16384	0	67108864	0
8	0	32768	0	134217728	0
16	0	65536	0	268435456	0
32	0	131072	0	536870912	0
64	0	262144	0	1073741824	0
128	0	524288	0	-2147483648	0
256	0	1048576	0	0	0
512	0	2097152	0	0	0
1024	0	4194304	0		
2048	0	8388608	0		

Fixpunkt erreicht?

Es geht also weiter!

java.lang.StackOverflowError

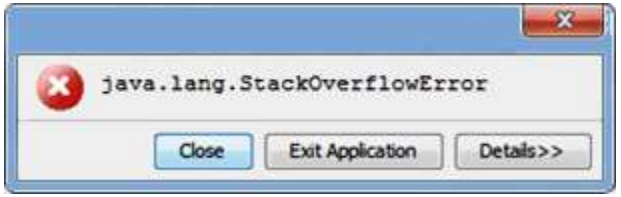
verdoppeln

halbieren

...auf „if (b == 1)“ kann man da lange warten...

Nullmol Null es Null, bliev Null!
(En d'r Kayjass Nummer Null)

java.lang.StackOverflowError



Eine freundliche Erklärung auf Französisch aus https://**fr.wikipedia.org**/wiki/D%C3%A9passement_de_pile:

En informatique, un dépassement de pile ou débordement de pile (en anglais, stack overflow) est un bug causé par un processus qui, lors de l'écriture dans une pile, écrit à l'extérieur de l'espace alloué à la pile, écrasant ainsi des informations nécessaires au processus.

L'expression *dépassement de pile* peut s'appliquer à toutes les piles. Cependant, lorsque l'on parle de dépassement de pile, on fait habituellement référence à la pile d'exécution.

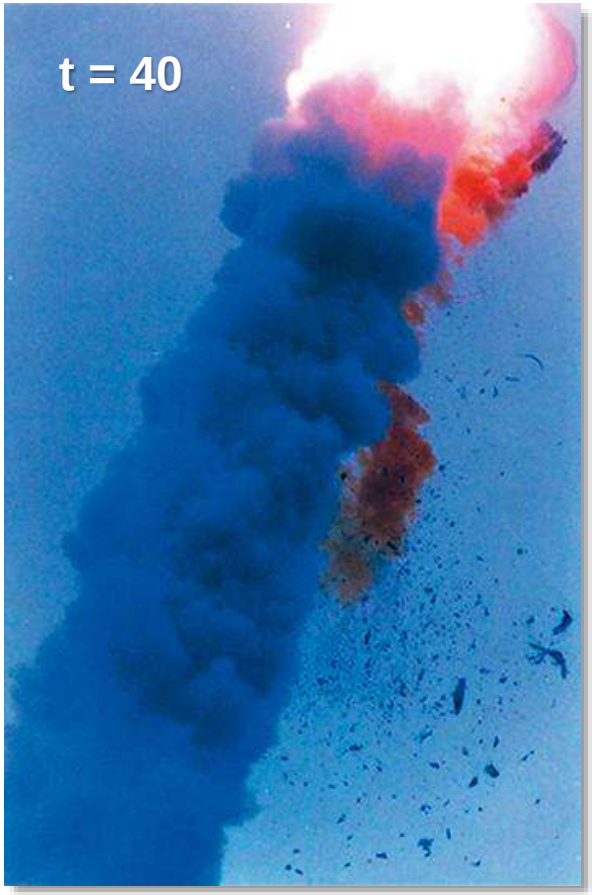
Dans tous les langages de programmation, la pile d'exécution contient une quantité limitée de mémoire, habituellement déterminée au début du programme. La taille de la pile d'exécution dépend de nombreux facteurs, incluant le langage de programmation, l'architecture du processeur, l'utilisation du traitement multithread et de la quantité de mémoire vive disponible. Lorsque trop d'informations sont enregistrées dans la pile d'exécution, la pile déborde et écrase des zones de programme à l'extérieur de la pile. Il en résulte généralement une interruption du programme.

La cause la plus fréquente des dépassements de pile est une récursivité trop profonde ou infinie.

4. Juni 1996: „An overflow occurred“

Historische Notiz

Am 4. Juni 1996 misslang der erste Flug einer Ariane 5-Rakete. Sie explodierte nur 40 Sekunden nach dem Start in einer Höhe von ca. 3700 m.



An overflow occurred...

Fri, 2 Aug 1996 – The causes of the Ariane 5 rocket crash (summary):

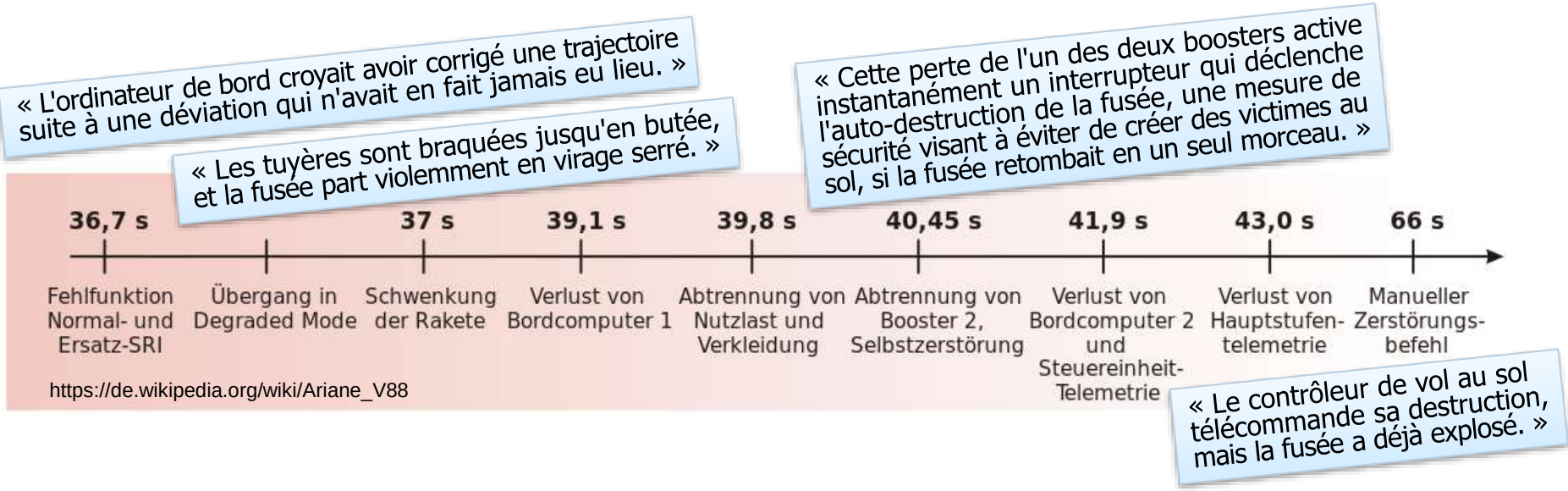
- An overflow occurred in the Inertial Reference System (SRI) computer when converting a 64-bit floating point to 16-bit signed integer value.
  - SRI: computes position, orientation, and velocity without external references
- There was no error handler for that specific overflow. The default handler shut down the SRI unit.
- The standby SRI unit had previously shut itself down for the same reason. The hot SRI and the standby were running the same software.
- The shutdown caused the SRI to output a memory dump on the bus. The main computer interpreted the memory dump as flight data, causing such a violent trajectory "correction" that the rocket disintegrated.
- The program that failed was a pre-flight program, and should not have been running during the flight.

Quelle: www.esrin.esa.it/htdocs/tidc/Press/Press96/ariane5rep.html

An overflow occurred...

Nothing is as expensive as making mistakes. – E.W. Dijkstra

- "The investigation team concluded that the designers of the computer system put in protections against hardware faults but did not take into account software faults."
- Das gesamte Ariane-Programm wurde so anderthalb Jahre verzögert; dies verursachte Kosten von ca. 10 Milliarden €.



Vgl. dazu: J. Jezequel, B. Meyer: "Design by Contract: The Lessons of Ariane", Computer, Jan. 1997, 129-130; sowie: https://de.wikipedia.org/wiki/Ariane_V88

« La récupération des deux systèmes de guidage inertiel parmi les débris de la fusée, et l'analyse des informations encore présentes dans la mémoire des appareils a permis de retracer avec précision les dernières secondes du vol. »

An overflow occurred...

Paolo Ferri von der European Space Agency (ESA) war seinerzeit der "Spacecraft Operations Manager" für die mit Nutzlast, die Cluster-Satellitenflotte. Er war im Kontrollzentrum dabei und erinnert sich:

I remember Heike Schweitzer, the secretary of our Flight Director, standing behind the back row, gazing fascinated at the pictures on the wall screen.

Suddenly, Heike's eyes widened and her expression changed. I turned to the screen: a big fireball was there, in place of the previous image of Ariane climbing to the clouds. My first thought was: well, this is the separation of the solid boosters. At the same time, in the headphones I heard two voices: the voice of the Kourou launch control centre speaker stating the usual: "trajectoire normale, tous les paramètres normaux", and, on a different loop, the voice of Nicolas Bobrinsky, the ESOC Ground Operations Manager, announcing to all the ground stations: "Launcher exploded. Loss of mission". (Probably all these events happened within less than a second. I still wonder how Nicolas could react so quickly to what he was seeing.) My next thought at this stage was: "What a stupid joke to make about the separation of the solid boosters!". I was still refusing to accept the evidence. Then the picture on the screen changed, the camera was zooming in on the burning and smoking remains of Ariane and its payload falling down to the coast of Guyana. It was at this stage that I first thought: "that piece of burning hardware must be one of the Cluster spacecraft". And I sat back in my chair, feeling totally empty. I turned again to the back row: Heike was still staring at the screen, tears in her eyes. […] We had foreseen and trained for all sorts of launcher problems, low injection altitude, wrong separation attitude, even the catastrophic case of having to acquire for tumbling spacecraft. In all cases we had found solutions, designed contingency procedures, practised them. Of course there had been no point in simulating a launcher explosion: there was no possible reaction for this case. […]

Heike was not the only one with tears in her eyes on that day. Several of my colleagues were surreptitiously wiping their eyes, and many of them were crying openly. When you have invested more than 5 years of your life in a project, it is not easy to see it literally go up in smoke 38 seconds after launch.

Aus: Madeleine Schäfer: How to Survive in Space! ESA, 1997

An overflow occurred...

Der Programmcode:

- Berechnung der Ariane-Vertikalbeschleunigung (ist OK: hat Überlaufschutz):

```
L_M_BV_32 := TBD.T_ENTIER_32S ((1.0/C_M_LSB_BV)
                * G_M_INFO_DERIVE(T_ALG.E_BV));
if L_M_BV_32 > 32767 then P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
      P_M_DERIVE(T_ALG.E_BV) :=
                UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M_BV_32));
end if;
```

Programm in der Sprache „Ada"

Konvertierung auf 16-Bit-Integer

- Und der Horizontalbeschleunigung (ohne Überlaufschutz!):

```
P_M_DERIVE(T_ALG.E_BH) :=
    UC_16S_EN_16NS (TDB.T_ENTIER_16S ((1.0/C_M_LSB_BH)
      * G_M_INFO_DERIVE(T_ALG.E_BH)));
```

Konvertierung auf 16-Bit-Integer

Nachfolgende Reparatur: E_BH *(biais horizontal)* als 32-Bit- statt 16-Bit-Integer

Modul enthält sieben Variablen, deren Berechnung evtl. einen Operandenfehler auslösen kann. Vier der Variablen wurden durch eine Ausnahmebehandlung geschützt. Da die maximale Auslastung des SRI-Computers laut den technischen Vorgaben 80% nicht überschreiten durfte, wurde von den Projektpartnern beschlossen, aus Effizienzgründen die restlichen drei Variablen inklusive E_BH ungeschützt zu lassen. Analysen hatten ergeben, dass sie entweder begrenzte Auswirkungen hatten oder ein grosser Sicherheitsspielraum vorhanden war. Dies stellte sich im Falle von E_BH als falsch heraus.

An overflow occurred...

Der Programmcode:

The official Ariane 5 accident report notes that software was assumed to be correct until it was shown to be faulty. The opposite assumption is more realistic. -- Nancy G. Leveson

Der sodann „reparierte" Code zur Berechnung der Horizontalbeschleunigung; nun mit Überlaufschutz, analog zur Vertikalbeschleunigung:

```
L_M_BH_32 := TBD.T_ENTIER_16S((1.0 / C_M_LSB_BH)
                 * G_M_INFO_DERIVE(T_ALG.E_BH));
if L_M_BH_32 > 32767 then P_M_DERIVE(T_ALG.E_BH) := 16#7FFF#;
elseif L_M_BH_32 < -32768 then P_M_DERIVE(T_ALG.E_BH) := 16#8000#;
else
       P_M_DERIVE(T_ALG.E_BH) :=
                 UC_16S_EN_16NS(TBD.T_ENTIER_16S(L_M_BH_32));
end if;
```

An overflow occurred...

https://en.wikipedia.org/wiki/Cluster_%28spacecraft%29

The Ariane 5 reused the inertial reference platform from the Ariane 4, but the Ariane 5's flight path differed considerably from the previous models. Specifically, the Ariane 5's greater horizontal acceleration caused the computers in both the back-up and primary platforms to crash and emit diagnostic data misinterpreted by the autopilot as spurious position and velocity data. Pre-flight tests had never been performed on the inertial platform under simulated Ariane 5 flight conditions so the error was not discovered before launch. During the investigation, a simulated Ariane 5 flight was conducted on another inertial platform. It failed in exactly the same way as the actual flight units.

The greater horizontal acceleration caused a data conversion from a 64-bit floating point number to a 16-bit signed integer value to overflow and cause a hardware exception. Efficiency considerations had omitted range checks for this particular variable, though conversions of other variables in the code were protected. The exception halted the reference platforms, resulting in the destruction of the flight.

Although a software error was identified as the direct cause, this was considered to be made possible by system design failures and management issues:

a) On the basis of those calculations the main computer commanded the booster nozzles, and somewhat later the main engine nozzle also, to make a large correction for an attitude deviation that had not occurred.

b) A rapid change of attitude occurred, which caused the launcher to disintegrate at 39 seconds after H_0 due to aerodynamic forces.

c) Ariane 5's inertial reference system is essentially the same as a system presently flying on Ariane 4. The part of the software that caused the interruption in the inertial system computers is used before launch to align the inertial reference system and, in Ariane 4, also to enable a rapid realignment of the system in case of a late hold in the countdown. This realignment function, which does not serve any purpose on Ariane 5, was nevertheless retained for commonality reasons and allowed, as in Ariane 4, to operate for approximately 40 seconds from lift-off.

An overflow occurred...

https://en.wikipedia.org/wiki/Cluster_%28spacecraft%29

d) During design of the software of the inertial reference system used for Ariane 4 and Ariane 5, a decision was taken that it was not necessary to protect the inertial system computer from being made inoperative by an excessive value of the variable related to the horizontal velocity, a protection provided for several other variables of the alignment software. When taking this design decision, it was not analysed or fully understood which values this particular variable might assume when the alignment software was allowed to operate after lift-off.

e) In Ariane 4 flights using the same type of inertial reference system there had been no such failure because the trajectory during the first 40 seconds of flight is such that the particular variable related to horizontal velocity cannot reach, with an adequate operational margin, a value beyond the limit present in the software.

f) Ariane 5 has a high initial acceleration and trajectory, which leads to a build-up of horizontal velocity five times more rapid than for Ariane 4. The higher horizontal velocity of Ariane 5 generated, within the 40-second timeframe, the excessive value that caused the inertial system computers to cease operation.

g) The purpose of the review process, which involves all major partners in the Ariane 5 programme, is to validate design decisions and to obtain flight qualification. In this process, the limitations of the alignment software were not fully analysed and the possible implications of allowing it to continue to function during flight were not realised.

h) The specification of the inertial reference system and the tests performed at equipment level did not specifically include the Ariane 5 trajectory data. Consequently the realignment function was not tested under simulated Ariane 5 flight conditions, and the design error was not discovered.

i) It would have been technically feasible to include almost the entire inertial reference system in the overall system simulations which were performed. For a number of reasons it was decided to use the simulated output of the inertial reference system, not the system itself or its detailed simulation. Had the system been included, the failure could have been detected.

j) Post-flight simulations have been carried out on a computer with software of the inertial reference system and with a simulated environment, including the actual trajectory data from the Ariane 501 flight. These simulations have faithfully reproduced the chain of events leading to the failure of the inertial reference systems.

L'explosion était inéluctable

https://fr.wikipedia.org/wiki/Vol_501_d'Ariane_5

Le bug informatique qui a causé la mise hors service des systèmes de guidage à centrales inertielles a eu lieu durant la procédure d'étalonnage de l'appareil. Dans un usage normal, cet étalonnage mesure de très faibles valeurs lorsque la fusée est immobile, et il n'est donc pas protégé contre des valeurs élevées, telles que l'on peut obtenir lorsque ces mesures sont effectuées durant le vol. Le choix de laisser l'appareil en mode calibrage après le décollage date du début du programme Ariane, plus de dix ans avant l'incident. Il est motivé par le fait que sur les premières fusées Ariane, en cas de retardement du décollage, il était nécessaire de relancer la procédure de calibrage, qui durait plus de 45 minutes.

Tout tenait à une seule petite variable : celle allouée à l'accélération horizontale. En effet, l'accélération horizontale maximum produite par Ariane 4 donnait une valeur décimale d'environ 64. La valeur d'accélération horizontale de la fusée étant traitée dans un registre mémoire à 8 bits, cela donne en base binaire 2^8 = 256 valeurs disponibles, un nombre suffisant pour coder la valeur 64, qui donne en binaire 1000000 et ne nécessite que 7 bits. Mais Ariane 5 était bien plus puissante et brutale : son accélération pouvait atteindre la valeur 300, qui donne 100101100 en binaire et nécessite un registre à 9 bits. Ainsi, la variable codée sur 8 bits a connu un dépassement de capacité, puisque son emplacement mémoire n'était pas assez grand pour accepter une valeur aussi importante. Il aurait fallu la coder sur un bit de plus, donc 9 bits, ce qui aurait permis de stocker une valeur limite de 2^9-1 = 511, alors suffisante pour coder la valeur 300. De ce dépassement de capacité a résulté une valeur absurde dans la variable, ne correspondant pas à la réalité. Par effet domino, le logiciel décida de l'autodestruction de la fusée à partir de cette donnée erronée.

Le système de navigation, utilisé depuis longtemps sur Ariane 4, était réputé fiable et le Centre national d'études spatiales a tout simplement demandé à ne pas effectuer les simulations de vol pour ces appareils, ce qui devait ainsi lui permettre d'économiser 800 000 francs sur le coût des préparatifs avant-lancement. Réalisées en laboratoire après la catastrophe, ces simulations ont justement permis de vérifier que l'explosion était inéluctable.

Ende der historischen Notiz

Was geschah eigentlich genau bei f(1,0)?

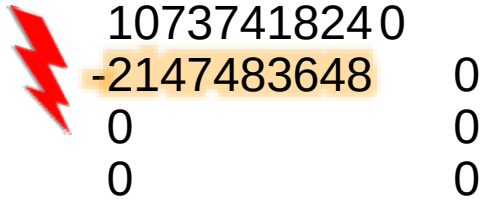
1	0	4096	0	16777216	0
2	0	8192	0	33554432	0
4	0	16384	0	67108864	0
8	0	32768	0	134217728	0
16	0	65536	0	268435456	0
32	0	131072	0	536870912	0
64	0	262144	0	1073741824	0
128	0	524288	0	-2147483648	0
256	0	1048576	0	0	0
512	0	2097152	0	0	0
1024	0	4194304	0	…	
2048	0	8388608	0	java.lang.StackOverflowError	

verdoppeln

halbieren

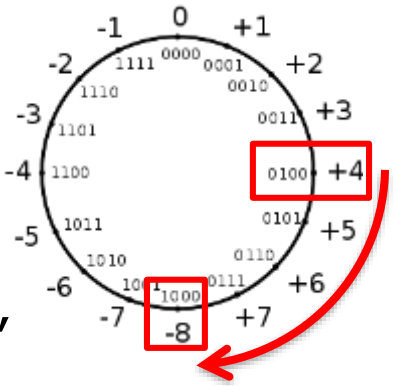
Arithmetischer Überlauf

- Eigentlich trat in f (beim Multiplizieren mit b = 0) schon lange vor dem Stack-Überlauf ein anderer Überlauf auf:



- Die Sprachspezifikation von Java erläutert das Phänomen:

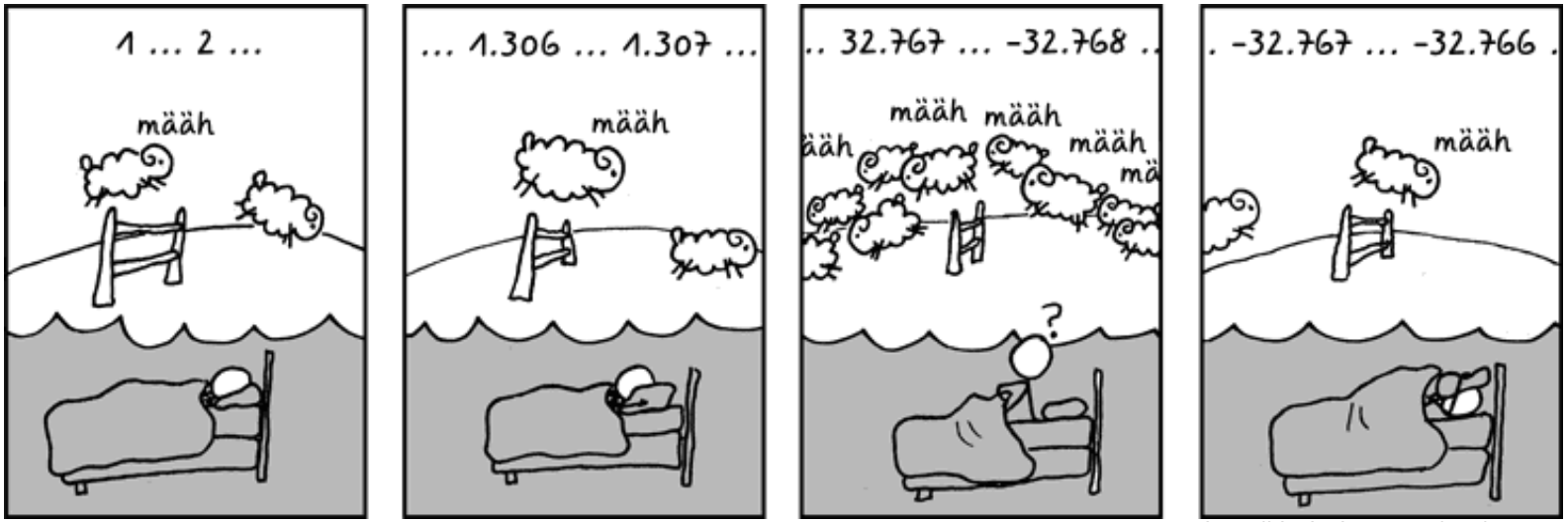
  "If an integer multiplication overflows, then the result is the low-order bits of the mathematical product as represented in some sufficiently large two's-complement format. As a result, if overflow occurs, then the sign of the result may not be the same as the sign of the mathematical product of the two operand values."



- Hier kam es also zu einem „arithmetischen" Überlauf
  - Durch die Multiplikation mit 2 entstand eine Zahl, die grösser ist als der bei int darstellbare Bereich (-2147483648 bis 2147483647), man läuft im Zahlenring in den anschliessenden negativen Bereich hinein
  - Dies könnte auch bei einem Eingabewert b ≠ 0 passieren!

Arithmetischer Überlauf (2)

- Diesen Fehlertypus würde man gerne abfangen, aber:
  - "Despite the fact that overflow, underflow, or loss of information may occur, evaluation of a multiplication operator never throws a run-time exception."
- Man muss also selbst dafür sorgen, dass das Produkt aus a und b nicht grösser als 2147483647 wird
  - Denkübung: wie?



https://xkcde.dapete.net/571/

Korrektheit von f(a,b) = a × b

- Wir wollen zeigen, dass die Funktion

$$f(a,b) = \begin{cases} a & \text{, falls } b = 1 \\ f(2a, b/2) & \text{, falls } b \text{ gerade} \\ a + f\left(2a, \frac{b-1}{2}\right) & \text{, sonst} \end{cases}$$

für alle a, b ∈ $\mathbb{N}^+$ das Produkt von a und b berechnet.

So, wie wir es in der Schule kennengelernt haben, z.B.: 5 x 7 = $\underbrace{7 + 7 + 7 + 7 + 7}_{\text{fünf Mal die 7}}$

Wir beschränken also ganz bewusst den Definitionsbereich der Funktion (keine 0, keine negativen Zahlen, keine rationalen Zahlen bzw. Gleitpunktzahlen...)

Korrektheit des Java-Programms?

- Damit wäre auch gezeigt, dass das Programmstück

```
static int f(int a, int b){
  if (b == 1)    return a;
  if (b%2 == 0) return f(2*a, b/2);
           else return a + f(2*a, (b-1)/2);
}
```

$$f(a,b) = \begin{cases} a & \text{, falls } b = 1 \\ f(2a, b/2) & \text{, falls } b \text{ gerade} \\ a + f\left(2a, \frac{b-1}{2}\right) & \text{, sonst} \end{cases}$$

das erwartet tut, also „a mal b“ berechnet.

- **Einspruch, Euer Ehren!**
  - Woher wissen wir eigentlich, dass das Programm genau der oben definierten Funktion entspricht? (→ *Semantik*)
  - Wie ist eigentlich das „sonst“ zu interpretieren? Als Alternative zu „gerade“ (also: „ungerade“) oder auch zu b = 1?
  - Müsste nicht ein „else“ zwischen den zwei if-Zeilen stehen?
  - Was geschieht bei einem Überlauf?

Korrektheit von f induktiv

$$f(a,b) = \begin{cases} a & \text{, falls } b = 1 \\ f(2a, b/2) & \text{, falls } b \text{ gerade} \\ a + f\left(2a, \frac{b-1}{2}\right) & \text{, sonst} \end{cases}$$

- Behauptung: $\forall$ a, b $\in \mathbb{N}^+$: **f(a,b) = a × b**

Zum Java-Programm sagen wir hier noch gar nichts

- Beweis induktiv über b:
  - b = 1:

    1 + 1 + . . . + 1 (a Mal): Man *zählt* bis a

    $\forall$ a $\in \mathbb{N}^+$: f(a,1) = a = a × 1 (gilt offensichtlich nach der Def. von f, *Fall 1*)

  - b = n+1, mit der Induktionsannahme $\forall$ a $\in \mathbb{N}^+$, $\forall$ b $\in$ {1,...,n}: f(a,b) = a×b:

    a) Sei b gerade: Es gilt
    f(a,b) = f(2a,b/2) [nach Definition, *Fall 2*]
    = 2a × b/2 [wg. Induktionsannahme, ! da b/2 $\in$ {1,...,n}]
    = a × b.

    b) Sei b ungerade (und ≠1): Es gilt
    f(a,b) = a + f(2a, (b-1)/2) [nach Definition, *Fall 3*]
    = a + 2a × (b-1)/2 [wg. Induktionsannahme da (b-1)/2 $\in$ {1,...,n}]
    = a + a × (b-1)
    = a × b. □

Aber wieso eigentlich?

Zwei Denkübungen: 1) Darf a im Beweis auch negativ sein?
2) Hätte man im Beweis nicht auch den Fall b = 0 mit einschliessen können?

Das Induktionsprinzip

Historische Notiz

- Die vollständige Induktion ist in der Informatik generell ein wichtiges Beweisprinzip – im verallgemeinerten Sinn können mit der „strukturellen Induktion“ Aussagen über rekursiv definierte Mengen (z.B. Programm- oder Datenstrukturen) oder iterative Methoden bewiesen werden

- *Successive Induction* by Augustus De Morgan (1806 – 1871)
  In: *The Penny Cyclopedia of the Society for the Diffusion of Useful Knowledge*, 1838

INDUCTION (Mathematics). The method of induction, in the sense in which the word is used in natural philosophy, is not known in pure mathematics. There certainly are instances in which a general proposition is proved by a collection of the demonstrations of different cases, which may remind the investigator of the inductive process, or the collection of the general from the particular. Such instances however must not be taken as permanent, for it usually happens that a general demonstration is discovered as soon as attention is turned to the subject.

There is however one particular method of proceeding which is extremely common in mathematical reasoning, and to which we propose to give the name of *successive induction*. It has the main character of induction in physics, because it is really the collection of a general truth from a demonstration which implies the examination of every particular case; but it differs from the process of physics inasmuch as each case depends upon one which precedes. Substituting however demonstration for observation, the mathematical process bears an analogy to the experimental one, which, in our opinion, is a sufficient justification of the term ‘successive induction.’

Allerdings wurde das Beweisprinzip schon im 17. Jh., nur nicht unter diesem Namen, von Blaise Pascal und Jakob Bernoulli verwendet. Richard Dedekind führt dann 1888 in einer ersten Version seiner berühmten Schrift „Was sind und was sollen die Zahlen?“ den Begriff „*vollständige* Induktion“ ein.

Induktion: Erläuterung von Bertrand Russell

Der britischer Philosoph, Mathematiker und Logiker Bertrand Russell (1872 – 1970) schreibt in seiner „Einführung in die mathematische Philosophie“ (1919) zur Induktion folgendes:

Die Verwendung der mathematischen Induktion bei Beweisen war in der Vergangenheit eine Art Mysterium. Man konnte scheinbar keinen ernsthaften Zweifel an der Gültigkeit dieser Beweismethode hegen. Aber niemand wusste, warum sie gültig war. Manche glaubten, sie sei ein Spezialfall der Induktion im logischen Sinn. Poincaré hielt sie für ein Prinzip von grösster Wichtigkeit, durch das man eine unendliche Zahl von Syllogismen in ein einziges Argument zusammenziehen könne. Wir wissen heute, dass all diese Betrachtungen irrtümlich sind. Die mathematische Induktion ist eine Definition und kein Prinzip. Es gibt gewisse Zahlen, für die sie gilt, und andere [Russel spricht hier die unendlichen Kardinalzahlen Cantors an], *für die sie nicht gilt. Wir definieren die „natürlichen Zahlen" als diejenigen, auf die man die mathematische Induktion bei Beweisen anwenden kann, d.h. als diejenigen, die alle induktiven Eigenschaften besitzen. Daraus folgt, dass solche Beweise auf die natürlichen Zahlen angewandt werden können. Dies ist nicht irgendeine mysteriöse Intuition oder irgendein Axiom oder Prinzip. Es folgt vielmehr einfach aus dem Satz selbst. Wenn „Vierfüssler" definiert sind als Tiere mit vier Füssen, so folgt daraus, dass Tiere, die vier Füsse haben, Vierfüssler sind. Ganz ähnlich liegt der Fall der Zahlen, die der mathematischen Induktion genügen.*

Russel schreibt ferner: *Die verallgemeinerte Theorie der Induktion verdankt man Frege. Sie wurden schon i.J. 1879 in seiner „Begriffsschrift" veröffentlicht. Trotz des grossen Wertes dieser Arbeit war ich meines Erachtens der erste Mensch, der sie überhaupt gelesen hat – und das über 20 Jahre nach ihrer Veröffentlichung.*

Die „Begriffsschrift“ von Gottlob Frege

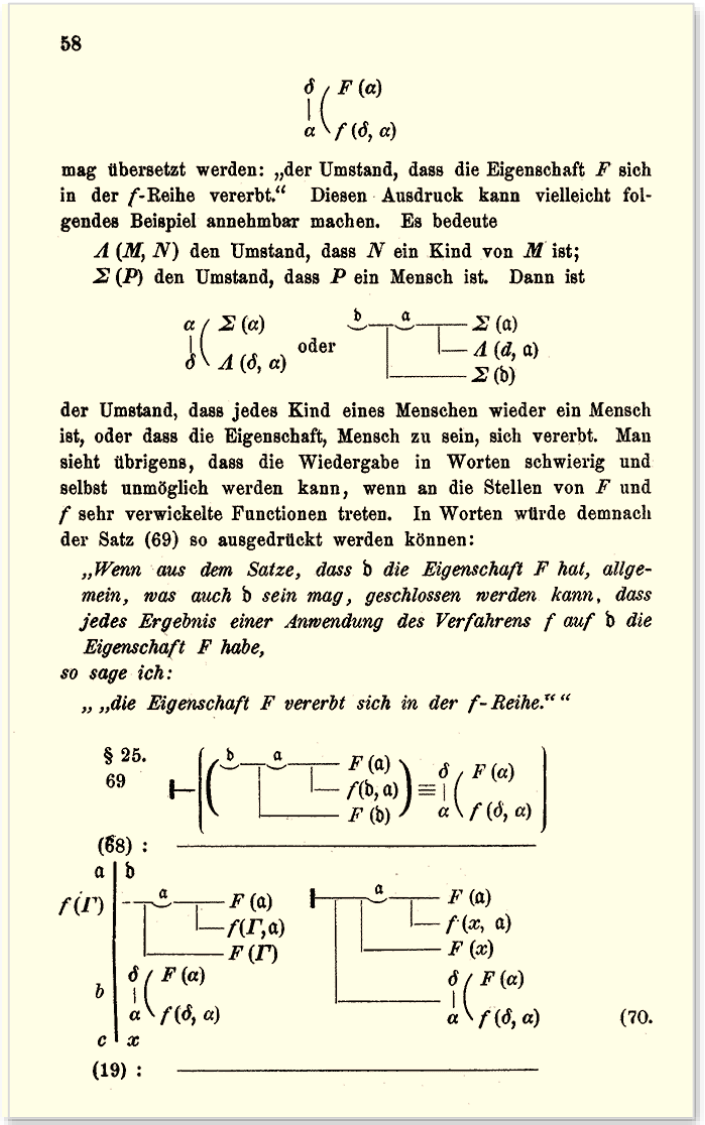
Die von Russell erwähnte „Begriffsschrift“ ist ein schmales, nur etwa 80 Seiten umfassendes Buch des Jenaer Mathematikers und Philosophen Gottlob Frege (1848 – 1925) zur Logik. Es wurde 1879 mit dem Untertitel *Eine der arithmetischen nachgebildete Formelsprache des reinen Denkens* veröffentlicht und gilt allgemein als die wichtigste Veröffentlichung im Bereich der Logik seit Aristoteles’ Organon. Frege gelang in diesem Buch zum ersten Mal eine Formalisierung der klassischen Prädikatenlogik und damit die erste Formalisierung einer Logik, in der sich ein grosser Teil der Mathematik ausdrücken liess.

§ 1. Die in der allgemeinen Grössenlehre gebräuchlichen Zeichen zerfallen in zwei Arten. Die erstere umfasst die Buchstaben, von denen jeder entweder eine unbestimmt gelassene Zahl oder eine unbestimmt gelassene Function vertritt. Diese Unbestimmtheit macht es möglich die Buchstaben zum Ausdrucke der Allgemeingiltigkeit von Sätzen zu verwenden wie in

$$(a + b)\,c = ac + bc.$$

Die andere Art umfasst solche Zeichen wie +, —, √, 0, 1, 2, von denen jedes seine eigenthümliche Bedeutung hat.

Diesen Grundgedanken der Unterscheidung zweier Arten von Zeichen, der in der Grössenlehre leider nicht rein durchgeführt ist*), *nehme ich auf, um ihn für das umfassendere Gebiet des reinen Denkens überhaupt nutzbar zu machen.*



58

$$\begin{matrix}\delta \\ | \\ \alpha\end{matrix}\Big(\begin{matrix}F(\alpha) \\ f(\delta, \alpha)\end{matrix}$$

mag übersetzt werden: „der Umstand, dass die Eigenschaft F sich in der f-Reihe vererbt.“ Diesen Ausdruck kann vielleicht folgendes Beispiel annehmbar machen. Es bedeute

$\Lambda(M, N)$ den Umstand, dass N ein Kind von M ist;
$\Sigma(P)$ den Umstand, dass P ein Mensch ist. Dann ist

$$\begin{matrix}\alpha \\ | \\ \delta\end{matrix}\Big(\begin{matrix}\Sigma(\alpha) \\ \Lambda(\delta, \alpha)\end{matrix} \quad \text{oder} \quad \begin{matrix}\mathfrak{b} & \mathfrak{a} & \\ \smile & \smile & \\ & & \Sigma(\mathfrak{a}) \\ & & \Lambda(\mathfrak{d}, \mathfrak{a}) \\ & & \Sigma(\mathfrak{b})\end{matrix}$$

der Umstand, dass jedes Kind eines Menschen wieder ein Mensch ist, oder dass die Eigenschaft, Mensch zu sein, sich vererbt. Man sieht übrigens, dass die Wiedergabe in Worten schwierig und selbst unmöglich werden kann, wenn an die Stellen von F und f sehr verwickelte Functionen treten. In Worten würde demnach der Satz (69) so ausgedrückt werden können:

„Wenn aus dem Satze, dass $\mathfrak{b}$ *die Eigenschaft* F *hat, allgemein, was auch* $\mathfrak{b}$ *sein mag, geschlossen werden kann, dass jedes Ergebnis einer Anwendung des Verfahrens* f *auf* $\mathfrak{b}$ *die Eigenschaft* F *habe,*

so sage ich:

„ „die Eigenschaft F *vererbt sich in der* f*-Reihe.“ “*

$$\S\ 25.\quad 69 \qquad \vdash\Big(\begin{matrix}\mathfrak{b} & \mathfrak{a} & \\ \smile & \smile & \\ & & F(\mathfrak{a}) \\ & & f(\mathfrak{b}, \mathfrak{a}) \\ & & F(\mathfrak{b})\end{matrix} \equiv \begin{matrix}\delta \\ | \\ \alpha\end{matrix}\Big(\begin{matrix}F(\alpha) \\ f(\delta, \alpha)\end{matrix}\Big)$$

(68) :

$$f(\Gamma)\Big|\begin{matrix}\mathfrak{a} & \mathfrak{b} \\ \smile & \smile \\ & F(\mathfrak{a}) \\ & f(\Gamma, \mathfrak{a}) \\ & F(\Gamma)\end{matrix} \qquad \begin{matrix}\mathfrak{a} & \\ \smile & \\ & F(\mathfrak{a}) \\ & f(x, \mathfrak{a}) \\ & F(x)\end{matrix}$$

$$b\Big|\begin{matrix}\delta \\ | \\ \alpha\end{matrix}\Big(\begin{matrix}F(\alpha) \\ f(\delta, \alpha)\end{matrix} \qquad \begin{matrix}\delta \\ | \\ \alpha\end{matrix}\Big(\begin{matrix}F(\alpha) \\ f(\delta, \alpha)\end{matrix}$$

$$c\Big|\ x \qquad (70.$$

(19) :

Russell trifft Frege (im Comic)

Hier und auf einigen der folgenden Seiten ein paar Auszüge aus der bereits früher erwähnten „graphic novel“ Logicomix.

Frege-Steckbrief im CIA intelligence aggregator:

Born: 8-Nov-1848. Birthplace: Wismar, Mecklenburg-Schwerin, Germany. Died: 26-Jul-1925. Location of death: Bad Kleinen, Germany. Cause of death: Illness. Remains: Buried, Wismar Cemetery. Gender: Male. Religion: Lutheran. Race or Ethnicity: White. Body height: 5 feet 5 inches. Sexual orientation: Straight. Occupation: Mathematician, Philosopher. Nationality: Germany. Military service: Prussian Army (1876).

Gottlob Frege: Formelsprache des reinen Denkens

G. Frege um 1880 [Wikipedia]

Gottlob Frege verwendete eine eigens von ihm geschaffene graphische, zweidimensionale Darstellung für Ausdrücke der Aussagen- und Prädikatenlogik; sein Kalkül führte erstmals den Allquantor sowie mehrstellige Prädikate ein. Wie erst später die *polnische Notation* kommt schon die Begriffsschriftnotation ohne Klammern aus.

Trotz ihrer epochalen Bedeutung ist die Begriffsschrift nicht Freges Hauptwerk. Ihr folgten 1884 *Die Grundlagen der Arithmetik* sowie 1893 und 1903 die beiden Bände der *Grundgesetze der Arithmetik*.

Freges vorrangiges Ziel war es, die Mathematik als Teil der Logik auszuweisen, also zu zeigen, dass alle mathematischen Sätze aus wenigen rein logischen Axiomen abgeleitet werden können. Auch wenn Freges eigenwilliger Schreibweise kein grosser Erfolg beschieden war, fusst nahezu jede Arbeit in der modernen Logik wenigstens mittelbar auf den Grundgedanken der Begriffsschrift. Da die Logik ferner Hilfs- und Grundlagendisziplin u.a. der Mathematik, Linguistik und Informatik ist, sind die indirekten Auswirkungen von Freges Werk kaum zu überschauen.

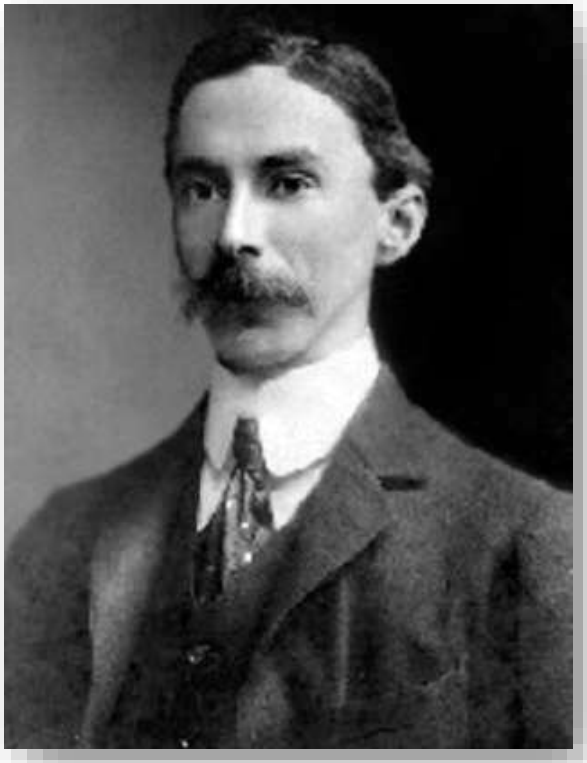
Die Mathematisierung der Logik enthielt jedoch einen Widerspruch (die sogenannte Russellsche Antinomie), wie Frege in einem berühmt gewordenen Brief von Bertrand Russell aus dem Jahr 1902 erfahren musste. Frege sah sein Lebenswerk gescheitert und zog sich resigniert

Der Briefwechsel zwischen Russell und Frege

von der Logik zurück. 1903 gestand Frege im Nachwort seiner *Grundgesetze der Arithmetik* ein, dass durch Russell die „Grundlagen seines Baues erschüttert" worden seien. In den Folgejahren verfiel Frege in eine Depression, die sich unter anderem darin äusserte, dass er keine grösseren Arbeiten mehr publizierte. [Obige 4 Absätze nach Wikipedia]

Die entsprechende Passage in Russels Brief an den „sehr geehrten Herrn Collegen" Frege lautet:

Nur in einem Punkte ist mir eine Schwierigkeit begegnet. Sie behaupten (S. 17) es könne auch die Funktion das unbestimmte Element bilden. Dies habe ich früher geglaubt, jedoch jetzt scheint mir diese Ansicht zweifelhaft, wegen des folgenden Widerspruchs: Sei w das Prädicat, ein Prädicat zu sein, welches von sich selbst nicht prädicirt werden kann. Kann man w von sich selbst prädiciren? Aus jeder Antwort folgt das Gegentheil. Deshalb muss man schliessen, dass w kein Prädicat ist. Ebenso giebt es keine Klasse (als Ganzes) derjenigen Klassen, die als Ganze sich selber nicht angehören. Daraus schliesse ich, dass unter gewissen Umständen eine definierbare Menge kein Ganzes bildet.

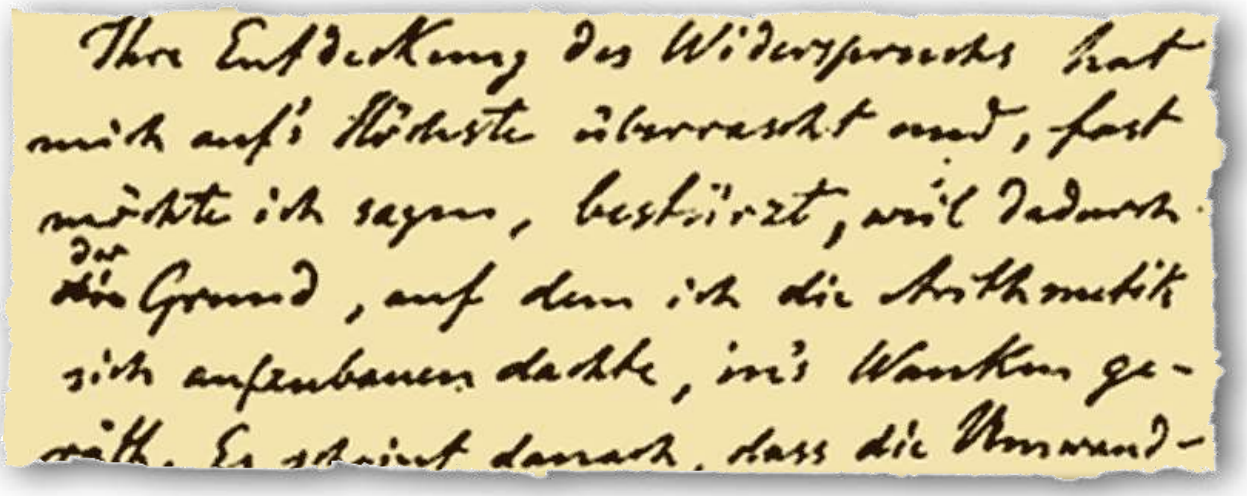


Bertrand Russell 1907 [Wikipedia]

Unmittelbar darauf antwortet Frege an Russell: *[...] Ihre Entdeckung des Widerspruchs hat mich auf's Höchste überrascht und, fast möchte ich sagen, bestürzt, weil dadurch der Grund,*

Versinkt die Grundlage der Arithmetik?

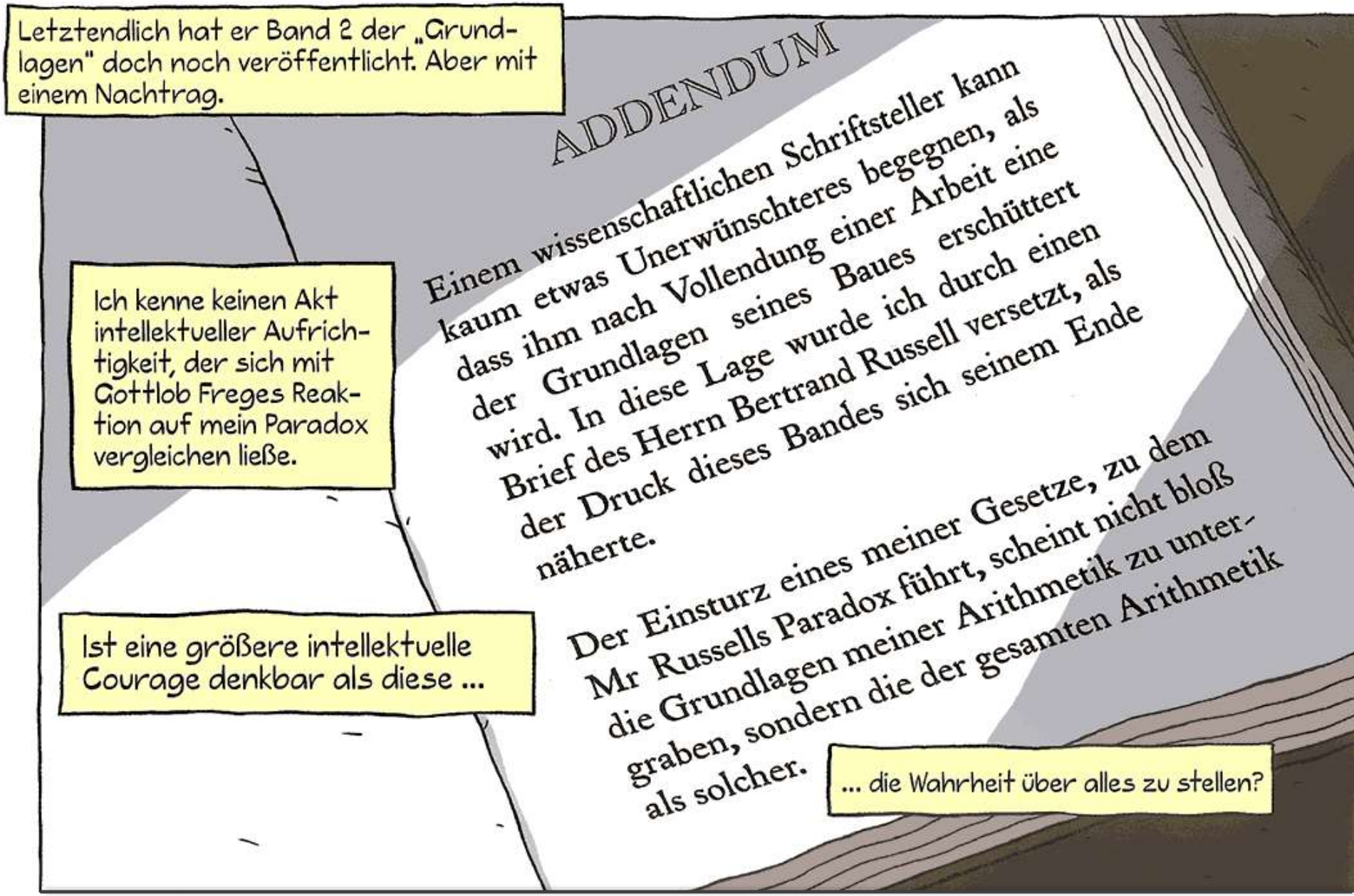
auf dem ich die Arithmetik sich aufzubauen dachte, in's Wanken geräth. [...] Ich muss noch weiter über die Sache nachdenken. Sie ist um so ernster, als mit dem Wegfall meines Gesetzes V nicht nur die Grundlage meiner Arithmetik, sondern die einzig mögliche Grundlage der Arithmetik überhaupt zu versinken scheint. [...]



Bertrand Russell schrieb über Frege 1962 an den französischen Logiker (sowie persönlichen Sekretär und Leibwächter von Leo Trotzki) Jean van Heijenoort: *As I think about acts of integrity and grace, I realise that there is nothing in my knowledge to compare with Frege's dedication to truth. His entire life's work was on the verge of completion, much of his work had been ignored to the benefit of men infinitely less capable, his second volume was about to be published, and upon finding that his fundamental assumption was in error, he responded with intellectual pleasure clearly submerging any feelings of personal disappointment.*

Aus dem weiter oben erwähnten Comicbuch *Logicomix.*

Bertrand Russell, der Erzähler in Logicomix


Letztendlich hat er Band 2 der „Grundlagen" doch noch veröffentlicht. Aber mit einem Nachtrag.
ADDENDUM
Einem wissenschaftlichen Schriftsteller kann kaum etwas Unerwünschteres begegnen, als dass ihm nach Vollendung einer Arbeit eine der Grundlagen seines Baues erschüttert wird. In diese Lage wurde ich durch einen Brief des Herrn Bertrand Russell versetzt, als der Druck dieses Bandes sich seinem Ende näherte.
Der Einsturz eines meiner Gesetze, zu dem Mr Russells Paradox führt, scheint nicht bloß die Grundlagen meiner Arithmetik zu untergraben, sondern die der gesamten Arithmetik als solcher.
Ich kenne keinen Akt intellektueller Aufrichtigkeit, der sich mit Gottlob Freges Reaktion auf mein Paradox vergleichen ließe.
Ist eine größere intellektuelle Courage denkbar als diese ...
... die Wahrheit über alles zu stellen?

„Über die wissenschaftliche Berechtigung einer Begriffsschrift“

G. Frege, Zeitschrift für Philosophie und philosophische Kritik, 81 (1882), 48 – 56

In den abstracteren Theilen der Wissenschaft macht sich immer auf’s Neue der Mangel eines Mittels fühlbar, Mißverständnisse bei Andern und zugleich Fehler im eignen Denken zu vermeiden. Beide haben ihre Ursache in der Unvollkommenheit der Sprache. Denn der sinnlichen Zeichen bedürfen wir nun einmal zum Denken. [...]

Deshalb verachte niemand die Zeichen! Von ihrer zweckmäßigen Wahl hängt nicht wenig ab. [...] Wir würden uns ohne Zeichen auch schwerlich zum begrifflichen Denken erheben. Indem wir nämlich verschiedenen aber ähnlichen Dingen dasselbe Zeichen geben, bezeichnen wir eigentlich nicht mehr das einzelne Ding, sondern das ihnen Gemeinsame, den Begriff. Und diesen gewinnen wir erst dadurch, daß wir ihn bezeichnen; denn da er an sich unanschaulich ist, bedarf er eines anschaulichen Vertreters, um uns erscheinen zu können. So erschließt uns das Sinnliche die Welt des Unsinnlichen. [...]

Die Sprache kann in dieser Hinsicht mit der Hand verglichen werden, die uns trotz ihrer Fähigkeit, sich den verschiedensten Aufgaben anzupassen, nicht genügt. Wir schaffen uns künstliche Hände, Werkzeuge für besondere Zwecke, die so genau arbeiten, wie die Hand es nicht vermöchte. Und wodurch wird diese Genauigkeit möglich? Durch eben die Starrheit, die Unveränderlichkeit der Theile, deren Mangel die Hand so vielseitig geschickt macht. So genügt auch die Wortsprache nicht. Wir bedürfen eines Ganzen von Zeichen, aus dem jede Vieldeutigkeit verbannt ist, dessen strenger logischer Form der Inhalt nicht entschlüpfen kann. [...]

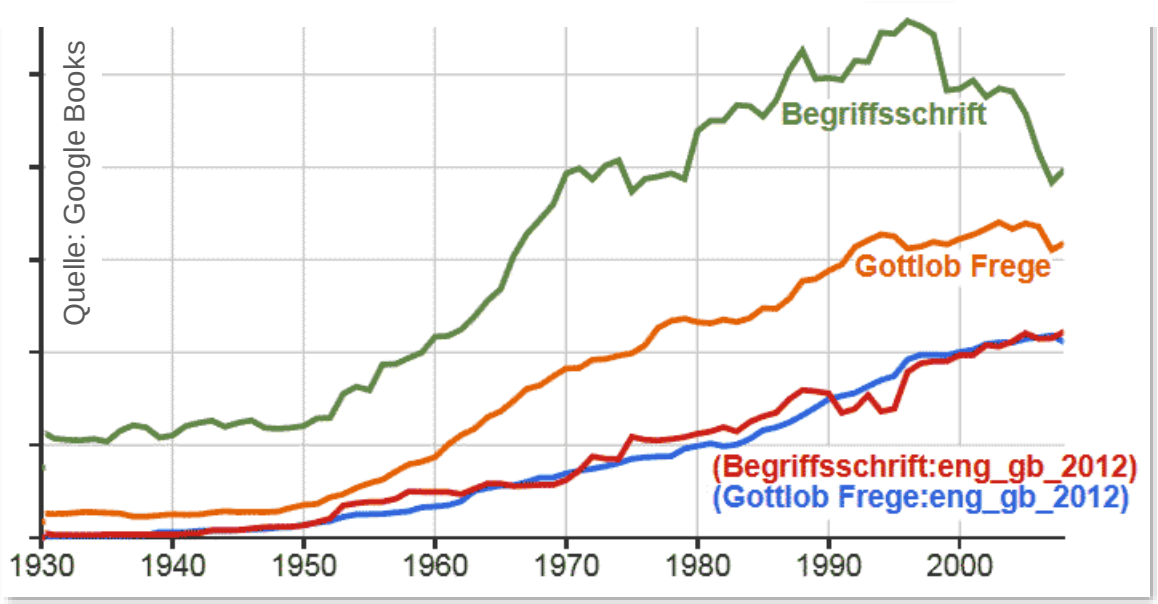
Von einer solchen [wahren Begriffsschrift] möchte ich Folgendes verlangen. Sie muß für die logischen Beziehungen einfache Ausdrucksweisen haben, die, an Zahl auf das Nothwendige beschränkt, leicht und sicher zu beherrschen sind. Diese Formen müssen geeignet seyn, sich mit einem Inhalte auf das Innigste zu verbinden. Dabei muß solche Kürze erstrebt werden, daß die zweifache Ausdehnung der Schreibfläche für die Uebersichtlichkeit der Darstellung gut ausgenutzt werden kann.

Ich habe nun versucht die mathematische Formelsprache durch Zeichen für die logischen Verhältnisse zu ergänzen, sodaß daraus zunächst für das Gebiet der Mathematik eine Begriffsschrift hervorgehe, wie ich sie als wünschenswerth dargestellt habe. Die Verwendung meiner Zeichen auf andern Gebieten wird dadurch nicht ausgeschossen. [...] Möchten deshalb auch Philosophen der Sache einige Beachtung schenken!

Begriffsschrift als Begriff

> Ich befinde mich in einem unglücklichen Cirkel: Bevor man der Begriffsschrift Beachtung schenkt, verlangt man deren Leistungen zu sehen und diese kann ich wieder nicht zeigen, ohne die Bekanntschaft mit ihr vorauszusetzen. So scheint mein im Anfange erwähntes Buch kaum auf Leser rechnen zu dürfen. [Frege 1882 an Anton Marty]

Allgemein bekannt werden Frege und seine Begriffsschrift erst 1936 durch eine kommentierte Bibliographie zur symbolischen Logik von Alonzo Church (Princeton), einem der Begründer der theoretischen Informatik, im „Journal of Symbolic Logic". Church führt über 2000 Werke auf, nennt aber als einflussreichste Personen Boole, DeMorgan, Russell, Brouwer, Gödel, Zermelo und den bis dahin weitgehend unbekannten Frege. Erst ab ca. 1950 tauchen Frege und seine Begriffsschrift als Begriffe in nennenswerter Zahl in der Literatur auf – hier bzgl. der deutschsprachigen und der in Grossbritannien erschienen Bücher entsprechend Google Books. Siebzig Jahre lang wurde dem aus heutiger Sicht epochalen Werk also kaum Beachtung geschenkt – wie schrieb schon Bertrand Russell: *Trotz des grossen Wertes dieser Arbeit war ich meines Erachtens der erste Mensch, der sie überhaupt gelesen hat – und das über 20 Jahre nach ihrer Veröffentlichung.*



BEGRIFFSSCHRIFT,

EINE DER ARITHMETISCHEN NACHGEBILDETE

FORMELSPRACHE

DES REINEN DENKENS.

VON

DR. GOTTLOB FREGE.

PRIVATDOCENTEN DER MATHEMATIK AN DER UNIVERSITÄT JENA.

HALLE A/S.

VERLAG VON LOUIS NEBERT.

1879.

Nett komponiert Matthias Wille im Einleitungskapitel seines Buches „Gottlob Frege“ diverse Zitate zu Freges Begriffsschrift zu einem schillernden Mosaik:

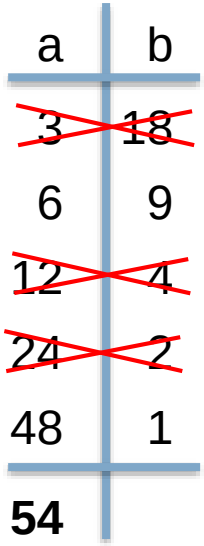
Selten wurde in der Wissenschaft der Wandel vom Traditionellen zum Modernen derart präzise und exakt durch eine einzige Studie bestimmt wie in ihrem Fall und noch seltener blieb eben diese Schrift wirkungsgeschichtlich so vollkommen belanglos. Historiographisch vielleicht ein einmaliges Phänomen.

Als der bahnbrechende Wert der Schrift endlich offiziell erkannt wird, ist sie schon nicht mehr zu bekommen. Eingestampft mangels Nachfrage. Da ist ihr Autor bereits seit mehr als einem Jahrzehnt tot, diesem nunmehr größten Logiker seit Aristoteles, vielleicht der größte überhaupt, von dem man im selben Ton zu sprechen hat wie von einem Kant. Mann und Werk sind im philosophischen Olymp angekommen. Aus dem verlegerischen Misserfolg von einst wird Mitte des 20. Jahrhunderts ein begehrtes Sammlerstück, eine bibliophile Rarität. Obgleich erst ein Menschenalter jung, ist sie seltener als so manche Inkunabel, ihr ungeheurer Wert Folge der vormaligen Wertlosigkeit. Eine paradoxe Biographie für eine logische Monographie. Antiquarisch so gut wie überhaupt nicht gehandelt, befinden sich die meisten der wenigen bekannten Exemplare wohlgehütet in den „Rare Books“-Abteilungen renommierter Universitäts- oder Nationalbibliotheken. [...]

In der gesamten Geschichte der Logik gibt es lediglich eine einzige andere Schrift, die mit ihr verglichen werden kann, die *Ersten Analytiken* des Aristoteles. Dazwischen liegen sagenhafte 2200 Jahre. Vielleicht ist sie die bedeutsamste Einzelschrift der Logik, die jemals verfasst wurde, in jedem Fall repräsentiert sie das wichtigste Kapitel in der Geschichte der modernen Logik und ihr Erscheinungsjahr, dieses Epochenjahr erster Ordnung, das bedeutungsvollste Datum in der gesamten Logikhistorie. Diese Geburtsurkunde der modernen Mathematikphilosophie enthält den fundamentalsten technischen Einzelfortschritt, der jemals in der Logik stattfand. Durch ihre kategorial neuen Gedanken vollzog sie eine kopernikanische Revolution der Disziplin, ein philosophiehistorischer Epochenwechsel ersten Ranges, vergleichbar mit jenem Descartes’. Erst durch sie wird diese uralte Wissenschaft zu einem wahrhaft großen Gebiet, zu einer strengen Wissenschaft als solcher. Es ist die Rede von einer umfangskleinen, aber inhaltlich überaus schwergewichtigen Abhandlung, es ist die Rede von einem brillanten, einem epischen Werk, es ist die Rede von der *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens...*

Altägyptische Multiplikation anders implementiert

- Beispiel 3×18:



Aufgrund des fortwährenden Halbierens von b

$\log_2 x \approx 3.322 \log_{10} x \approx 1.4427 \ln x$

1) Zeilen streichen, bei denen rechts eine gerade Zahl steht
2) Übrig gebliebene Zahlen der linken Spalte aufaddieren

Dieses Aufaddieren realisieren wir jetzt aber auf eine andere Art, indem wir explizit schritthaltend „akkumulieren“

Altägyptische Multiplikation iterativ

While-Schleife statt Rekursion

```
while (b > 0) {
    if ungerade(b) {
        z = z+a;
        b = b-1;
    }
    b = b/2;
    a = 2*a;
}
return z;
}
```

z akkumuliert die Werte der linken Spalte nicht gestrichener Zeilen

Akkumulator für das Ergebnis

b halbieren, a verdoppeln

a	b
~~3~~	~~18~~
6	9
~~12~~	~~4~~
~~24~~	~~2~~
48	1
54	

a akkumulieren, wenn b ungerade

Altägyptische Multiplikation iterativ

While-Schleife statt Rekursion

```
static int f(int i, int j) {
    int a = i;
    int b = j;
    int z = 0;
    while (b > 0) {
        if ungerade(b) {
            z = z+a;
            b = b-1;
        }
        b = b/2;
        a = 2*a;
    }
    return z;
}
```

Methode `ungerade(b)` kann als einfache Übung implementiert werden

i und j werden in der Methode f nicht verändert

Hier gilt offenbar $a \times b + z = i \times j$

So eine logische Aussage über Werte von Variablen nennt man „Zusicherung" („assertion")

z akkumuliert die Werte der linken Spalte nicht gestrichener Zeilen

Falls hier $a \times b + z = i \times j$

dann auch hier

Hier ist b gerade

Falls hier $a \times b + z = i \times j$

dann auch hier

Hier sollte b = 0 sein

Akkumulator für das Ergebnis

Altägyptische Multiplikation iterativ

While-Schleife statt Rekursion

```
static int f(int i, int j) {
    int a = i;
    int b = j;
    int z = 0;
    while (b > 0) {
        if ungerade(b) {
            z = z+a;
            b = b-1;
        }
        b = b/2;
        a = 2*a;
    }
    return z;
}
```

i und j werden in der Methode f nicht verändert

Hier gilt offenbar $a \times b + z = i \times j$

So eine logische Aussage über Werte von Variablen nennt man „Zusicherung" („assertion")

z akkumuliert die Werte der linken Spalte nicht gestrichener Zeilen

~~Falls~~ hier $a \times b + z = i \times j$

dann auch hier

Hier ist b gerade

Falls hier $a \times b + z = i \times j$

Transitivität der Implikation! („Kettenschluss")

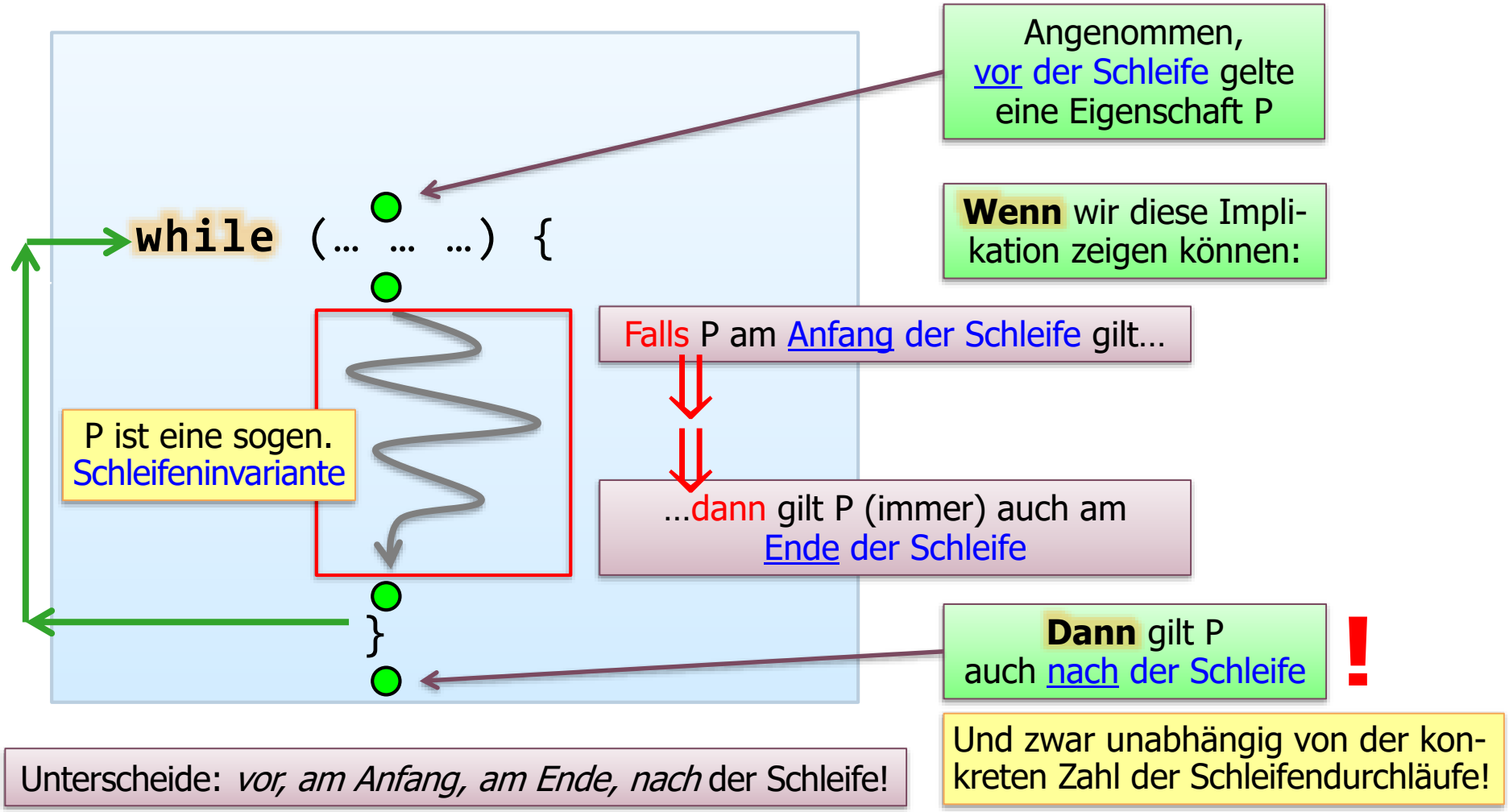
~~dann~~ auch hier

(Modus Ponens)

Hier sollte b = 0 sein

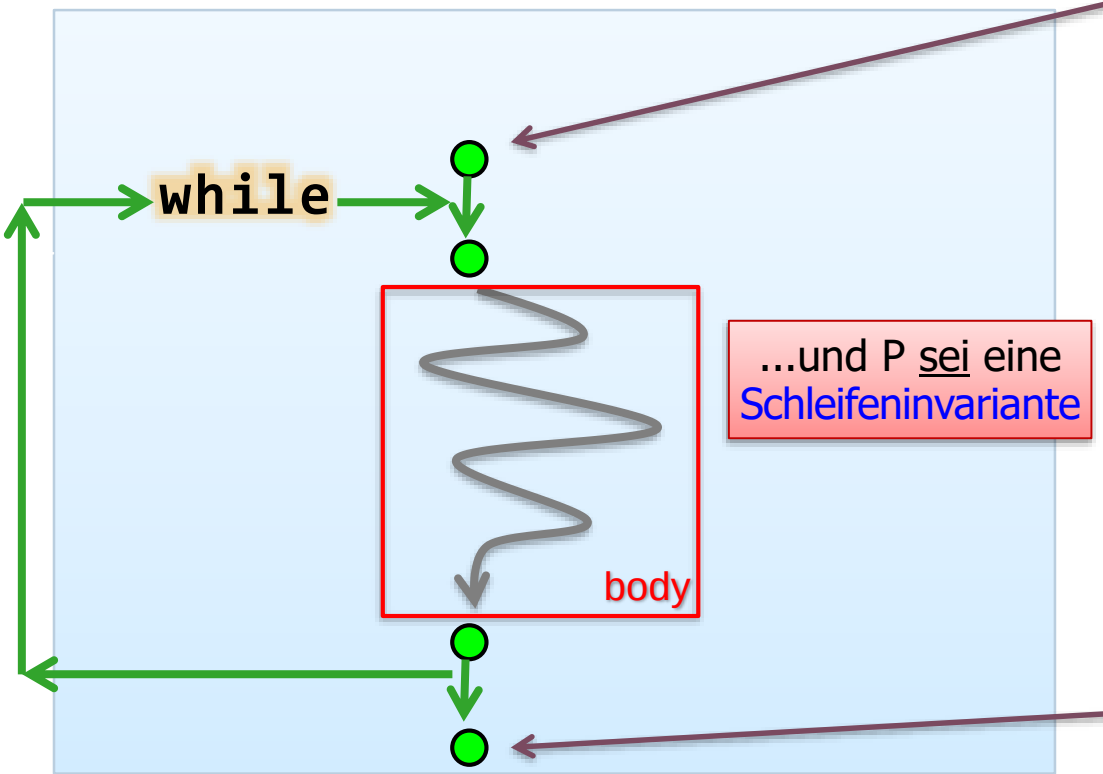
Akkumulator für das Ergebnis

Schleifeninvariante



Beweisregel für while-Schleifen

Denkübung: Wieso gilt eigentlich diese Beweis-regel? (Ist es eine „natur-notwendige“ Tatsache?)



Angenommen, vor der Schleife gelte eine Eigenschaft P...

...und P sei eine Schleifeninvariante

Dann gilt P auch nach der Schleife

Diese Regel sieht, im sogenannten „Hoare-Kalkül“, formal so aus (Aussage über dem Strich impliziert die darunter)

$$\frac{\{C \wedge P\}\ \text{body}\ \{P\}}{\{P\}\ \textbf{while}\ (C)\ \text{body}\ \{\neg C \wedge P\}}$$

! Und zwar unabhängig von der kon-kreten Zahl der Schleifendurchläufe!

Altägyptische Multiplikation iterativ (Invariante)

```
static int f(int i, int j) {
    int a = i;
    int b = j;
    int z = 0;
    while (b > 0) {
        if ungerade(b) {
            z = z+a;
            b = b-1;
        }
        b = b/2;
        a = 2*a;
    }
    return z;
}
```

i und j werden in der Methode f nicht verändert

Hier gilt offenbar $a \times b + z = i \times j$

Falls hier $a \times b + z = i \times j$

z akkumuliert die Werte der linken Spalte nicht gestrichener Zeilen

Dass $a \times b + z = i \times j$ eine Schleifeninvariante ist, hatten wir schon eingesehen → obige Beweisregel ist anwendbar

Hier ist b gerade

dann auch hier

Hier sollte b = 0 sein

Hier gilt also $b = 0$ und $a \times b + z = i \times j$

(Letzteres vermöge Beweisregel)

Altägyptische Multiplikation iterativ (Invariante)

i und j werden in der Methode f nicht verändert

```
static int f(int i, int j) {
    int a = i;
    int b = j;
    int z = 0;
    while (b > 0) {
        if ungerade(b) {
            z = z+a;
            b = b-1;
        }
        b = b/2;
        a = 2*a;
    }
    return z;
}
```

Eingabe i, j

z akkumuliert die Werte der linken Spalte nicht gestrichener Zeilen

Hier ist b gerade

Hier sollte b = 0 sein

Ausgabe z

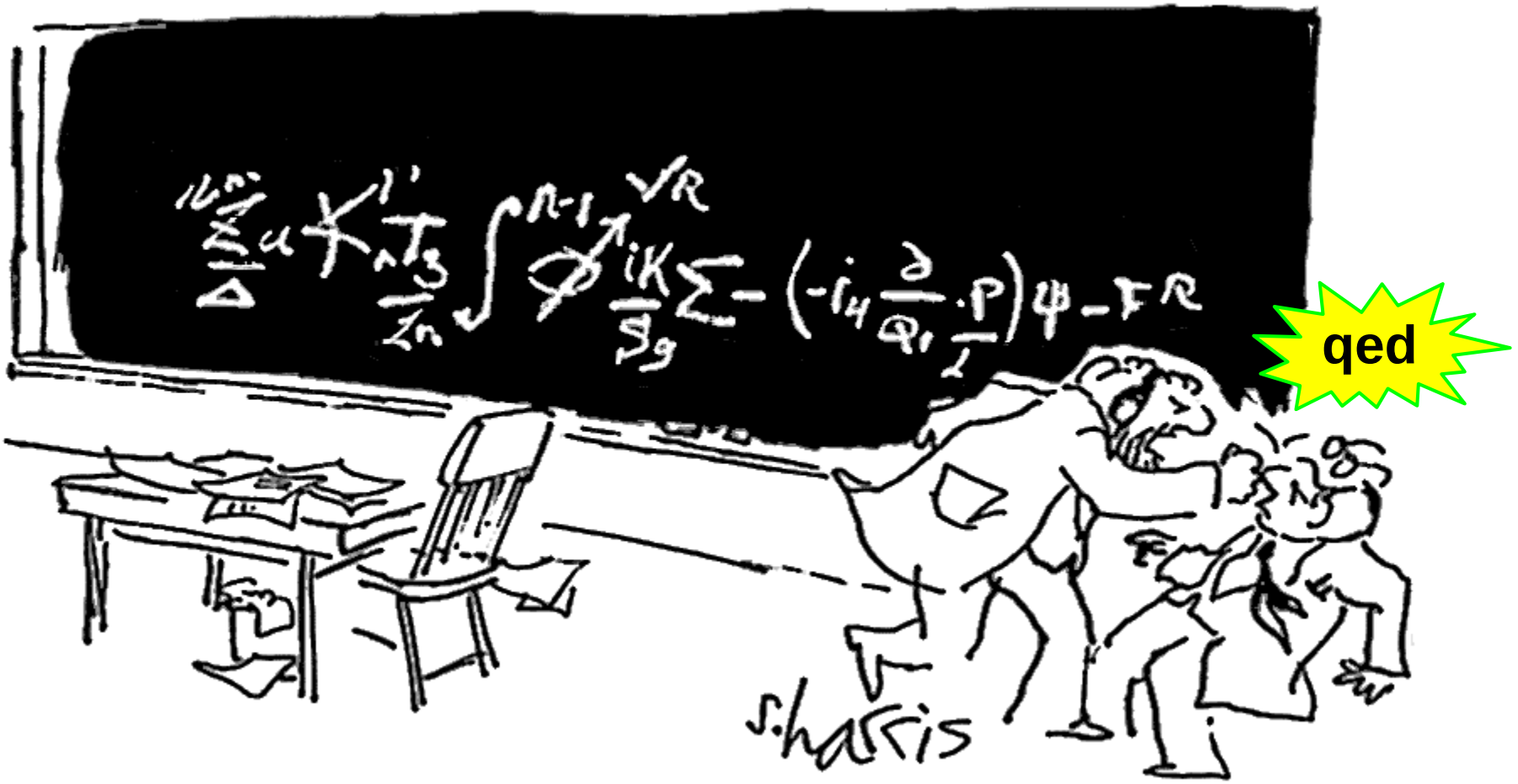
An allen Punkten ○ gilt die Zusicherung **a × b + z = i × j**, insbesondere vor „return" (mit b = 0)

Was zu beweisen war

Also gilt dort **z = i × j**

Also wird das Produkt der Eingabeparameter zurückgeliefert!

"Schlagkräftiger Beweis? Geb' ich Dir!"



https://pbs.twimg.com/media/D0qAmK8W0AEuPmV.jpg

Logische Programmverifikation mit Invarianten

```
static int f(int i, int j) {
    int a = i;
    int b = j;
    int z = 0;
    while (b > 0) {
        if ungerade(b) {
            z = z+a;
            b = b-1;
        }
        b = b/2;
        a = 2*a;
    }
    return z;
}
```

Eingabe i, j

z akkumuliert die Werte der linken Spalte nicht gestrichener Zeilen

Hier ist b gerade

Hier sollte b = 0 sein

Ausgabe

z = i x j

Wir können beweisen, dass f das Produkt zweier Zahlen i, j berechnet, selbst wenn wir

- die zugrundeliegende „Idee" des Algorithmus nicht kennen;
- mit „reverse engineering" die Idee nicht ermitteln können.

Aber sollte dann nicht auch eine Maschine ganz mechanisch prüfen können, ob die Invariante bei jedem Schritt erhalten bleibt?

→ Aha, „automatische" Programmverifikation!

Logische Programmverifikation mit Invarianten

```
static int f(int i, int j) {
    int a = i;
    int b = j;
    int z = 0;

    return z;
}
```

Eingabe
i, j

Black Magic Box

(Bewahrt die Invariante
a x b + z = i x j und
setzt am Ende **b** auf **0**)

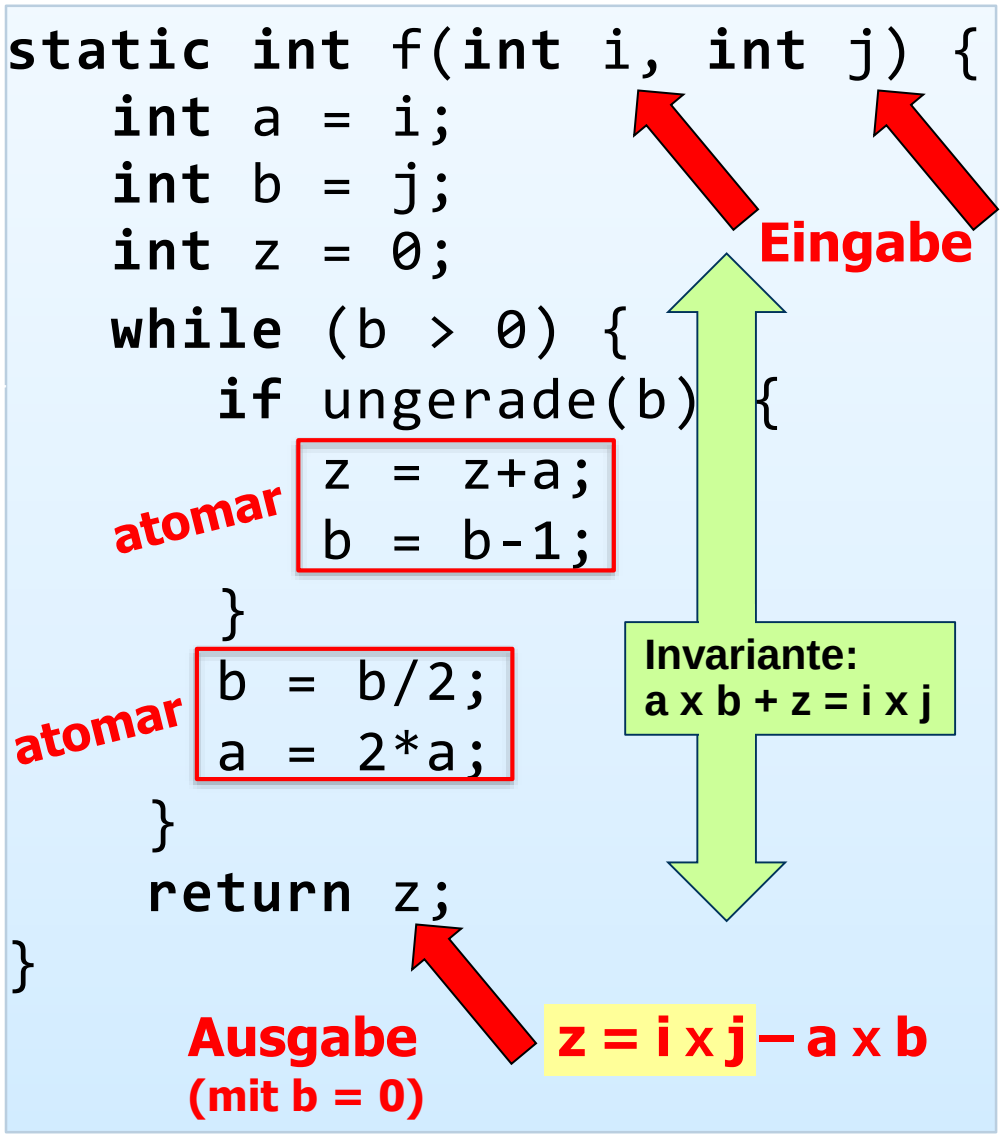
Ausgabe
(da b = 0)

z = i x j

→ Aha, „automatische“ Programmverifikation!

Beware of bugs in the above code; I have only proved it correct, not tried it. -- Donald Knuth

Invarianten sind mächtige Beweishilfsmittel



- Der Ausdruck $a \times b + z = i \times j$ ist eine sogenannte Invariante
- Zwar ändern sich die Werte von a, b und z im Programm, aber $a \times b + z$ bleibt „im Wesentlichen“ unverändert:
  - „Kurzzeitig“ wird die Invariante durch eine Anweisung zerstört, aber dann gleich darauf wieder repariert
  - Sieht man solche Aktionsfolgen als „atomar“ an, bleibt der Wert in der Tat invariant
  - Insbesondere garantiert die Schleife die Invariante („Schleifeninvariante“), sie wirkt dort wie der Induktionsschritt beim Induktionsprinzip

Invarianten sind mächtige Beweishilfsmittel

```
static int f(int i, int j) {
    int a = i;
    int b = j;
    int z = 0;
    while (b > 0) {
        if ungerade(b) {
            z = z+a;
            b = b-1;
        }
        b = b/2;
        a = 2*a;
    }
    return z;
}
```

atomar

?

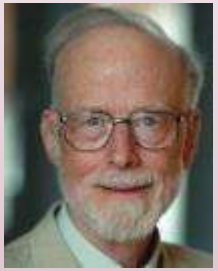
Eingabe

Invariante:
a x b + z = i x j

Ausgabe
(mit b = 0)

z = i x j – a x b

Im allgemeinen Sinn wird eine derartige Programmverifikation durch den „Hoare-Kalkül“ ermöglicht, entwickelt vom britischen Informatiker C.A.R. („Tony“) Hoare; dieser erfand 1960 auch den Quicksort-Algorithmus sowie 1978 die formale Sprache CSP; 1980 erhielt er den Turing Award. (Zum Hoare-Kalkül gleich mehr.)



Denkübungen:

- Gilt die Invariante auch, wenn b=b/2; a=2*a wegfällt? *Bleibt das Programm dann korrekt??*
- Ist der Beweis auch für negative i bzw. j korrekt?
- Und wenn man „while (b != 0)“ im Programm schreibt?

Schleifeninvarianten

Der Kolumnist interessiert sich bei der Beschäftigung mit news weniger für das, was sich ändert, als vielmehr für das, was bleibt, das Grundsätzliche. – Stefan Betschon

„Gilt eine Schleifeninvariante P *während* der Schleife?“

Oh je – was soll denn „während“ heissen bzw. bedeuten?

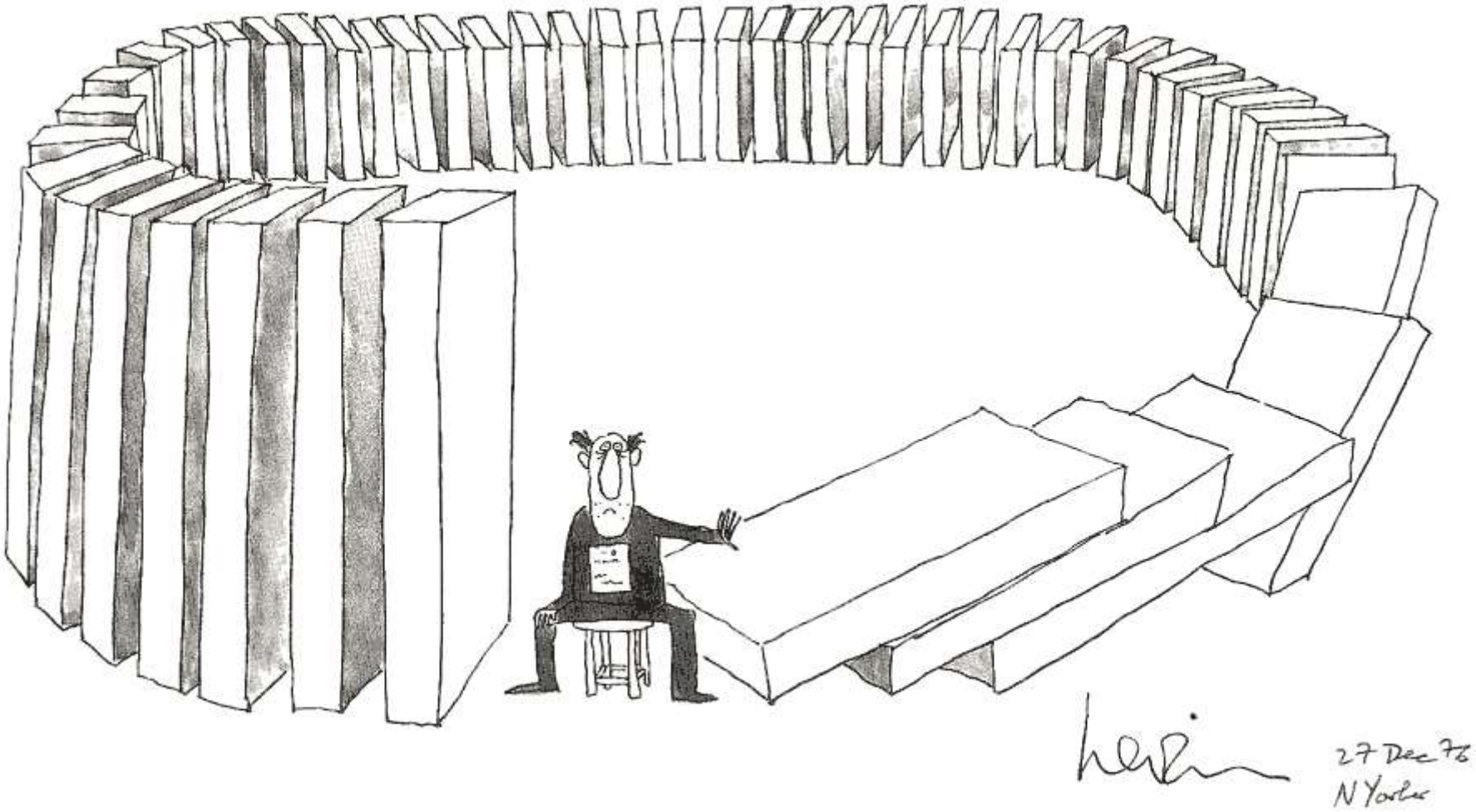
Wir bewegen uns hier ja im Bereich der Logik, also der Mathematik. Mathematische Objekte (wie hier die Programmtexte) sind aber statisch – es geschieht nichts bei ihnen oder in ihnen, es vergeht keine Zeit etc.

Vielleicht ist ja gemeint: „An jeder Stelle im Schleifenkörper“? Dann wäre die Antwort allerdings „nein“ – eine Schleifeninvariante kann ja „kurzzeitig“ ungültig werden, wenn sie bald danach, innerhalb des Schleifenkörpers, wieder „repariert“ wird.

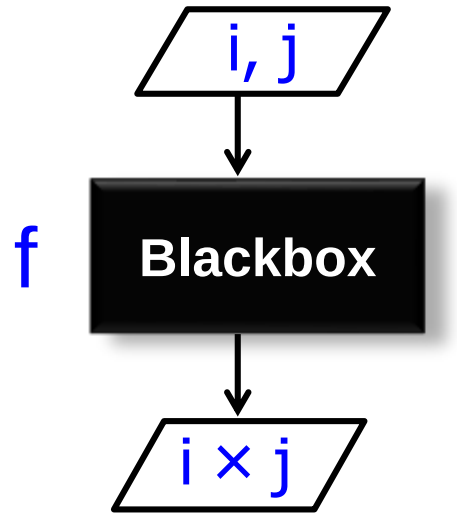
Aber vielleicht ist ja gemeint: „Wird die Eigenschaft P durch die Schleife erhalten, d.h., handelt es sich bei P um eine Schleifeninvariante?“ Das müsste man dann im konkreten Fall prüfen bzw. beweisen.

Als Denkübung zum Verständnis von Schleifeninvarianten: Kann eine falsche Aussage (z.B. 5 > 8) im Prinzip als Schleifeninvariante fungieren?

Schleifeninvarianten und Induktion sind mächtige Hilfsmittel



Programmspezifikation zur Programmverifikation



Als Vertrag zwischen Auftraggeber und Lieferant: „*Es soll das Produkt der beiden Eingabewerte berechnet werden*“

- **Verifikation**: Die „funktionale Korrektheit“ der Implementierung **relativ zur Spezifikation** (hier: $f(i,j) = i \times j$) logisch-formal zeigen
  → Programmtexte als mathematische Objekte!
  → Kalkül aus Beweisregeln, Axiomen, Aussagen- / Prädikatenlogik,…

- Oft wird ein Vertrag zusätzlich noch einige nichtfunktionale Eigenschaften spezifizieren, z.B. die gewünschte Effizienz

Hoare-Kalkül – ein kurzer Einblick
(Hoare: *An axiomatic basis for computer programming,* 1969)

- „Hoare-Tripel" von der Art $\{x \geq 0\}\ \ x = x + 1\ \ \{x > 0\}$.
- Allgemeiner: Seien *P, Q* und *R* „Zusicherungen" (engl: „assertions") in Form prädikatenlogischer Ausdrücke und seien **S** und **T** Anweisungsfolgen.

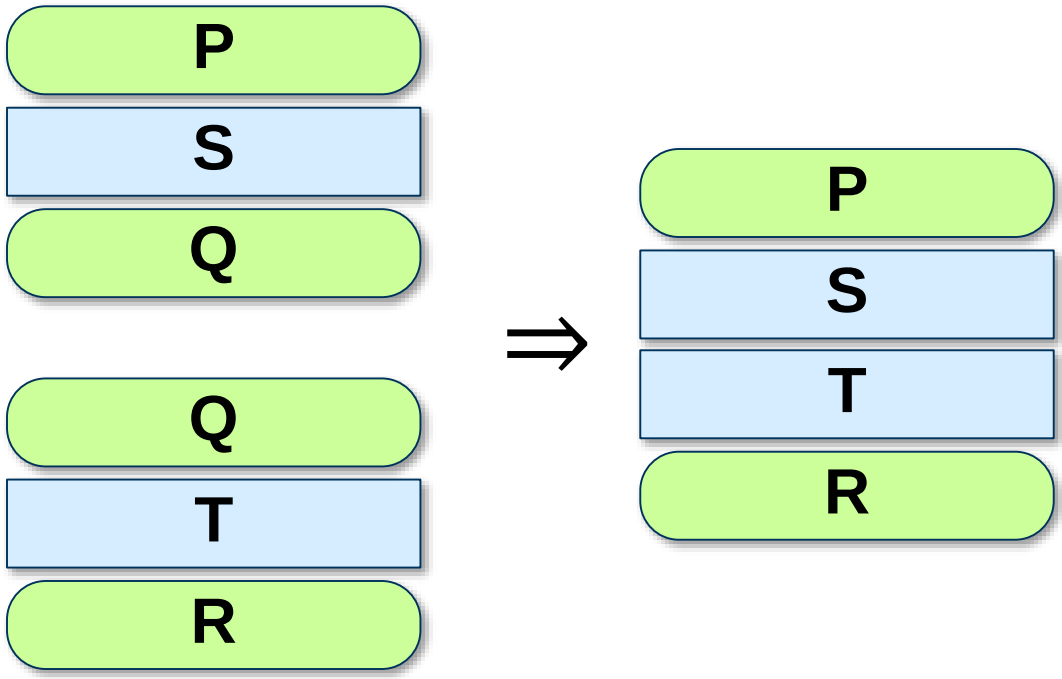
Wenn vor der Anweisung „x = x+1" gilt, dass x ≥ 0 ist, dann darf man davon ausgehen, dass danach sicherlich x > 0 gilt.

$$\frac{\{P\}\ \mathbf{S}\ \{Q\} \qquad \{Q\}\ \mathbf{T}\ \{R\}}{\{P\}\ \mathbf{S;T}\ \{R\}}$$

- Dann besagt z.B. obige Beweisregel, dass wenn die beiden über dem Strich stehenden Aussagen bewiesen worden sind, dann auch die unter dem Strich stehende Aussage als bewiesen angesehen wird – konkret: wenn **S** in einem Zustand gestartet wird, in dem die Vorbedingung *P* gilt, und nach Ausführung von **S** die Nachbedingung *Q* gilt, und wenn Analoges für **T** bzgl. *Q* und *R* gilt, dann ist garantiert, dass dies für die „Komposition" (also die Hintereinanderausführung) **S;T** bezüglich *P* und *R* gilt.

Hoare-Kalkül – ein kurzer Einblick: Komposition

- Veranschaulichen kann man diese Kompositionsregel so:

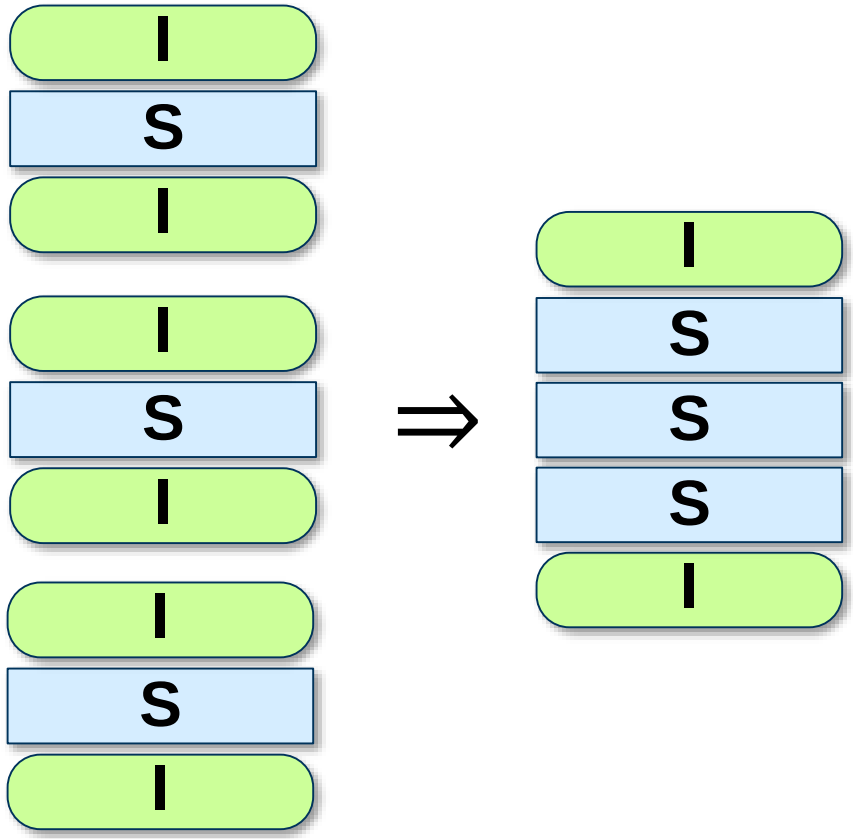


Für tiefere Einblicke: Man finde heraus (z.B. via Wikipedia), was „Modus Barbara“ als Teil der Syllogistik bedeutet

- Die Regel ermöglicht es also, aus den Aussagen zu einzelnen Programmteilen Aussagen über zusammengesetzte Programmteile zu gewinnen.

Hoare-Kalkül – ein kurzer Einblick: Invariante

- Wenn man eine Zusicherung I hat, die vor sowie nach einer Anweisungsfolge S gleichermassen gilt (also „invariant“ ist), dann kann S offenbar beliebig oft hintereinander ausgeführt werden, so dass I am Ende noch immer gilt.
- Dies führt zum Konzept der „Schleifeninvarianten“.



Hoare-Kalkül – ein kurzer Einblick: Schleife

- Entsprechend besagt folgende Beweisregel für Schleifen

$$\frac{\{C \wedge I\}\ \text{body}\ \{I\}}{\{I\}\ \textbf{while}\ (C)\ \text{body}\ \{\neg\, C \wedge I\}}\ ,$$

Endliche Formel macht eine Aussage über unendlich viele Fälle (beliebig viele Iterationen)

dass wenn der Schleifenkörper die Zusicherung I invariant lässt, dass man dann schliessen darf: Gilt I direkt vor der Schleife, dann auch nach Verlassen der Schleife; ausserdem ist dann die Schleifenbedingung C in negierter Form erfüllt.

- Schleifenbedingung C sollte aber keine Methode mit Nebeneffekten sein
- Und: Das Finden geeigneter Schleifeninvarianten ist nicht immer einfach!

- Obige Beweisregeln haben wir (eher informell) bei der Verifikation des Multiplikationsprogramms angewendet.
- Es gibt noch einige weitere Beweisregeln; wir erwähnen beispielhaft nachfolgend noch das Konstrukt für die Zuweisung.

Hoare-Kalkül – ein kurzer Einblick: Zuweisung

- Für die Zuweisung x = t; gilt folgende Regel (als Axiom!):

$$\{P_{[x \leftarrow t]}\}\ \mathbf{x = t;}\ \{P\}.$$

Nicht umgekehrt, wie man naiverweise zunächst meinen könnte!

- Dabei bezeichnet $\{P_{[x \leftarrow t]}\}$ diejenige Formel, die aus *P* (rechts) hervorgeht, indem jedes Auftreten der Variablen x durch den Ausdruck t ersetzt wird.

Bsp: Wenn y+z den Wert 1 hat, so ist nach dieser Zuweisung x gleich 1:

$\{y+z=1\}$ **x = y + z ;** $\{x=1\}$

- Das Zuweisungsaxiom besagt, dass jede Aussage, die vorher (!) für die rechte Seite der Zuweisung galt, nun für die Variable gilt.

- Als Anwendungsbeispiel dazu folgendes Hoare-Tripel:

$$\{a \times b - a + (z + a) = i \times j\}\ \mathbf{z = z + a;}\ \{a \times b - a + z = i \times j\}$$

Es wird etwas später, beim Thema „Proof-Carrying Code", verwendet, um den Erhalt der Invariante bei der Anweisungsfolge `z = z+a; b = b-1` formal nachzuweisen.

Hoare-Kalkül – ein kurzer Einblick: Konsequenz

- Gelegentlich ist es notwendig, eine Nachbedingung abzuschwächen oder eine Vorbedingung zu verstärken, um bei einer Anweisungskette die Passgenauigkeit zu erreichen.
- Formal kann dies mit der Konsequenzregel erzielt werden →

$$\frac{\{P'\}\ \mathbf{S}\ \{Q'\}}{\{P\}\ \mathbf{S}\ \{Q\}} \quad \text{falls } P \Rightarrow P' \text{ und } Q' \Rightarrow Q$$

- Beispiel: Benötigt man nach einer Zuweisung „**x = 0;**" die Zusicherung „$x \geq 0$", so kann man ausgehend vom Zuweisungsaxiom $\{0 = 0\}$ **x = 0;** $\{x = 0\}$ die Nachbedingung zu „$x \geq 0$" abschwächen (sowie die etwas sonderbare Vorbedingung durch „*wahr*" bzw. „*true*" ersetzen).
- Die im Kalkül möglichen Formeln „$\{true\}$**;** P" (d.h. mit der leeren Anweisung) sowie „$\{true\}$ **S** $\{false\}$" führen zusammen mit der Konsequenzregel zu tiefsinnigen Aspekten der Logik: Unvollständigkeit (Gödel) bzw. Halteproblem (Turing) – worauf wir hier jedoch nicht eingehen können.

Zusammengenommen definieren alle Regeln für alle Programmkonstrukte die (axiomatische) Semantik einer Programmiersprache.

Historische Notiz

An Axiomatic Basis for Computer Programming

C. A. R. Hoare
The Queen's University of Belfast, Northern Ireland*

In this paper an attempt is made to explore the logical foundations of computer programming by use of techniques which were first applied in the study of geometry and have later been extended to other branches of mathematics. This involves the elucidation of sets of axioms and rules of inference which can be used in proofs of the properties of computer programs. Examples are given of such axioms and rules, and a formal proof of a simple theorem is displayed. Finally, it is argued that important advantages, both theoretical and practical, may follow from a pursuance of these topics.

Sir Charles Antony Richard Hoare (mit Ehefrau Jill vor dem Buckingham-Palast) zeigt die Medaille, die er anlässlich des Ritterschlags durch Königin Elisabeth II. am 7. März 2000 erhalten hat.

Hoare, C. A. R. (1969). An axiomatic basis for computer programming. *Communications of the ACM*, 12(10), 576-580.

Stichwort „Kalkül“

Der eben grob skizzierte Hoare-Kalkül gibt Anlass, auf den Begriff „Kalkül“ selbst einzugehen: Es handelt sich dabei um ein System von Regeln, mit denen sich aus gegebenen Formeln oder Aussagen weitere ableiten lassen.

Das Wort leitet sich vom lateinischen *calculus* ab (Rechenstein, ursprünglich Kiesel- bzw. Kalksteinchen, *calculi*; später auch Berechnung), vgl. auch engl. *chalk*, beides aus lat. *calx* („Kies, Kalkstein“). Im 17. Jh. wird *calculus* in die deutsche Kaufmannssprache entlehnt und vielfach zu „Calcul“ gekürzt. Dieses nimmt entsprechend frz. *calcul* (Rechnung) im 18. Jh. frz. Aussprache und danach die Schreibweise „Kalkül“ an. Während z.B. „predicate calculus“ auf deutsch mit „Prädikatenkalkül“ zu übersetzen ist, bezeichnet in der Mathematik das engl. „calculus“ im generellen Sinne (oft als implizite Abkürzung von z.B. „differential calculus“) die Analysis bzw. Differential- und Integralrechnung; das frz. „calcul“ würde man hingegen eher mit „Rechnen“ (z.B. im Sinne der Arithmetik) bzw. „Rechnung“ übersetzen.

Formal besteht ein Kalkül zunächst aus einer Menge von Grundzeichen (dem Alphabet), aus denen Formeln nach festgelegten syntaktischen Mustern (den Formationsregeln) gebildet werden können. Präziser verfügt ein Kalkül über eine Menge von ausge-

calcŭlārĭus, *ii*, m. *(calculus)*, qui concerne un compte : MODEST. *Dig. 50, 8, 8.*

calcŭlātĭo, *ōnis*, f. *(calculus)*, ¶ 1 la pierre, [calculs dans la vessie] C.-AUR. *Tard. 5, 4, 60* ¶ 2 calcul, compte : CASSIOD. *Ep. 1, 10.*

calcŭlātŏr, *ōris*, m. *(calculo)*, calculateur : MART. *10, 62, 4* ‖ celui qui dresse, qui tient les comptes : DIG. *38, 1, 5* ‖ qui enseigne à compter : ISID. *Orig. 1, 3, 1.*

CALCULATOR

calcŭlātōrĭus, *a, um*, qui sert à compter : SCHOL. JUV. *7, 73.*

calcŭlātrix, *ĭcis*, f., celle qui calcule, qui enseigne à compter : BED. *Mund. Const. 1, p. 403.*

calcŭlensis, *e (calculus)*, relatif aux graviers, aux cailloux : *calculense purpurarum genus* PLIN. *9, 131*, sorte de pourpre.

1 **calcŭlo**, *āre*, tr., calculer, supputer : PRUD. *Peri. 2, 131* ‖ évaluer, estimer : SID. *Ep. 7, 9.*

2 **calcŭlo**, *ōnis*, m., calculateur : AUG. *Ord. 2, 12.*

calcŭlōsus, *a*, [illegible] (*s*), caillouteux : PLIN. *34*, [illegible] gravelle ou la pi[illegible] CELS. *7, 26.*

calcŭlus, *i*, m. (dimin. de *calx* 2), petite pierre ¶ 1 caillou : *conjectis in os calculis* CIC. *de Or. 1, 261,*

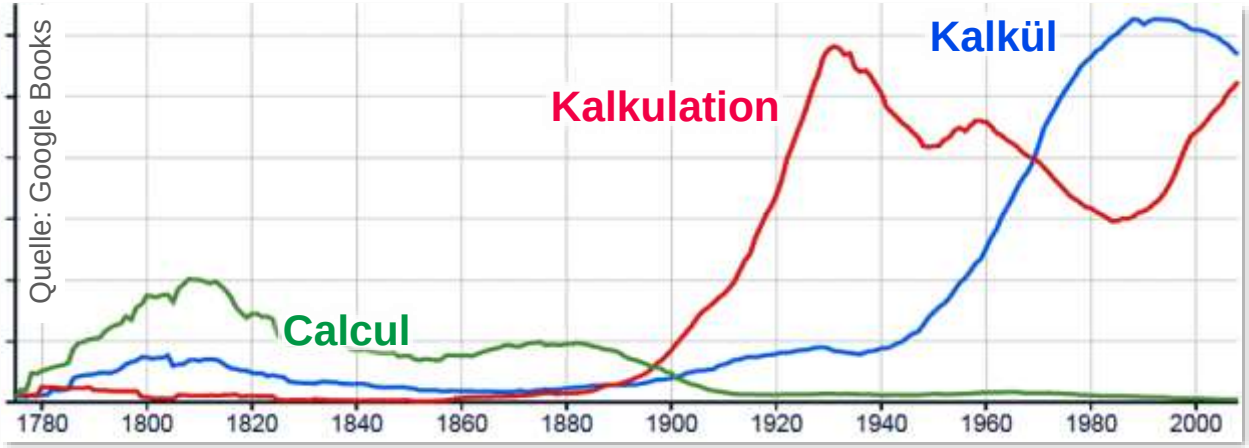
Dictionnaire latin-français, Félix Gaffiot, 1934

Stichwort „Kalkül“ (2)

zeichneten Formeln, den Axiomen, sowie einer Menge ausgezeichneter endlicher Folgen von Formeln $\varphi_1, \ldots, \varphi_n, \varphi$, den Deduktionsregeln (oder Beweisregeln, Ableitungsregeln bzw. Schlussregeln), wobei die Formeln $\varphi_1, \ldots, \varphi_n$ als die Prämissen und die Formel φ als die Konklusion der Deduktionsregel bezeichnet werden.

Ein Beweis einer Formel φ besteht dann aus einer endlichen Folge von Formeln $\psi_1, \ldots, \psi_m$ mit $\psi_m = \varphi$, wobei jede Formel ψ_i ein Axiom oder die Konklusion einer Deduktionsregel ist, deren Prämissen weiter vorne im Beweis vorkommen.

Die philosophischen Wurzeln des Kalkülbegriffs führt man bis auf die Syllogistik von Aristoteles zurück („*ein Syllogismos ist also ein Argument, in welchem sich, wenn etwas gesetzt wurde, etwas anderes als das Gesetzte mit Notwendigkeit durch das Gesetzte ergibt*“.) Wie dieses sich mit und ab Leibniz („*diese Art Kalkül könnte auch mit ei-einer Maschine ausgeführt werden*“) weiterentwickelte, wurde bereits weiter vorne im Abschnitt *Leibniz' Universalschrift: Calculemus!* dargestellt.



Proof-Carrying Code

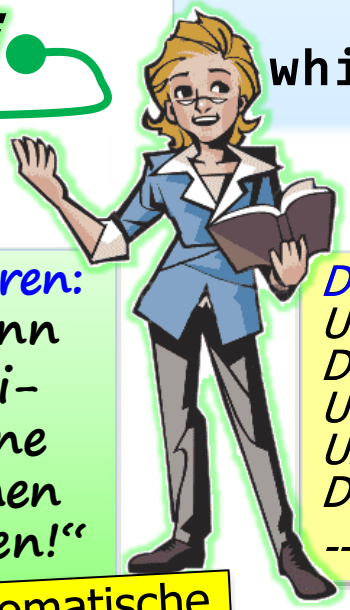
```
static int f(int i, int j){
    int a = i; // a = i
    int b = j; // b = j (Axiome)
    int z = 0;

    while (b != 0) {
```

```
b = j; // b = j
```

bedeutet

Ich kann garantieren:
„Hier haben dann b und j den gleichen Wert – ohne weitere Annahmen treffen zu müssen!“



Der Philosoph, der tritt herein
Und beweist Euch, es müsst so sein:
Das Erst wär so, das Zweite so,
Und drum das Dritt und Vierte so;
Und wenn das Erst und Zweit nicht wär,
Das Dritt und Viert wär nimmermehr.
-- J. W. Goethe (Faust I, Mephisto)

Axiomatische Semantik

Proof-Carrying Code

```
b = j;
```

bedeutet

```
// b = j
```

Axiomatische Semantik der Zuweisung

z vor der Zuweisung → $z + a = \varphi$

```
z = z+a;
```

z danach → $z = \varphi$

Tagesbefehl 317 an Maschinisten:
GEHE ZU *j*; GREIFE IN SEINEN SCHLUND UND HOLE SEINEN WERT HERAUS; DANN GEHE NACH LINKS ZU *b* UND STOPFE IHM DIESEN WERT INS MAUL!

Operationelle Semantik

```
static int f(int i, int j){
   int a = i; // a = i
   int b = j; // b = j (Axiome)
   int z = 0;

   while (b != 0) {

      if ungerade(b) {

         z = z+a;

         b = b-1;

      }

      b = b/2;

      a = 2*a;

   }

   return z;
}
```

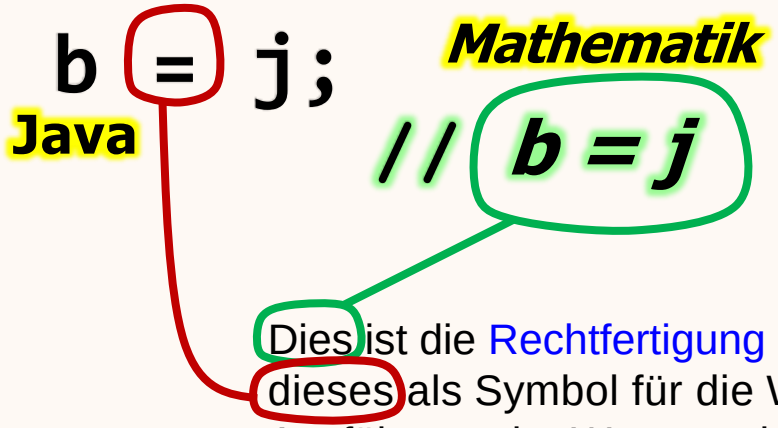
Крокодил

Computermaschinist als Saufnase und Mathematikerin als taillierte Blondine in einer Karikatur des sowjetischen Satiremagazins Крокодил („Krokodil“) vom Februar 1976 (Nr. 4).

Die Publikationen im „Krokodil“ hatten einen sehr aggressiven Charakter mit zahlreichen abstossenden, erniedrigenden und sogar offen antisemitischen Karikaturen. Entsprechend wirkt auch dieses Bild (jedenfalls aus heutiger Sicht) stark stereotypisch oder sogar sexistisch. Tatsache ist allerdings, dass das Computer-Personal seinerzeit wirklich oft weisse Ingenieurskittel trug und ein Minirock mit Stiefeln in dieser Zeit (auch im Westen) nicht unüblich war.

Typisch für Computerkarikaturen der 1970er-Jahre allgemein sind raumfüllende Maschinen mit grossen Schaltern und analogen Anzeigeinstrumenten.

Wertzuweisungssymbol



Dies ist die Rechtfertigung dafür, dass man in vielen Programmiersprachen dieses als Symbol für die Wertzuweisung einer Variablen schreibt: *Nach* Ausführung der Wertzuweisung gilt dann tatsächlich die Gleichheit!

Generell bereitet das „Gleichheitszeichen“ im Kontext der Zuweisung vielen Programmier-anfängern etwas Mühe. Niklaus Wirth meinte dazu einmal: „A notorious example for a bad idea was the choice of the equal sign to denote assignment. It goes back to Fortran in 1957 and has blindly been copied by armies of language designers. Why is it a bad idea? Because it overthrows a century old tradition to let ‘=’ denote a comparison for equality, a predicate which is either true or false. But Fortran made it to mean assignment, the enforcing of equality. In this case, the operands are on unequal footing: The left operand (a variable) is to be made equal to the right operand (an expression). x = y does not mean the same thing as y = x.”
Nicht nur Fortran, Java oder C, sondern auch viele anderen Sprachen verwenden dafür das „=“-Zeichen. Dagegen wird z.B. bei Ada, ALGOL, Pascal oder Simula die Zeichenkombination „**:=**“ verwendet; bei APL oder Smalltalk war „x ← y“ üblich, bei Forth schrieb man dafür „y x !“, bei BASIC „LET X = Y“ und beim geschwätzigen COBOL heisst es „MOVE y TO x“.

Wertzuweisungssymbol als Hackerangriff

Dass in Java, C und einigen anderen Programmiersprachen das mathematische Gleichheitszeichen für die Zuweisung verwendet wird (und dafür dann „==“ für den Test auf Wertgleichheit), wurde auch schon ausgenutzt um zu versuchen, einen Hintereingang („Backdoor“) in den Open-Source Linux-Quellcode einzuschleusen:

Jemand verschaffte sich, getarnt unter dem Namen des Kernentwicklers „David Miller“, illegal Zugang zum Code-Repository-System, wo der Linux-Quellcode aufbewahrt wird, und schleuste folgendes Codestück in die Routine ein, die den wait4-Systembefehl implementiert:

```
if ((options == (__WCLONE|__WALL)) && (current->uid = 0))
retval = -EINVAL;
```

Es sieht harmlos aus: Dass beim Superuser (mit uid = 0) gelegentlich Dinge anders ablaufen sollen als bei normalen Nutzern, ist klar, und der Betriebssystemkern enthält viele solche Situationen. Nur ein geübtes Auge wird aber entdecken, dass hier mit „uid = 0“ kein Test erfolgt, sondern damit der Variablen „uid“ der Wert 0 zugewiesen wird. Die aktuelle Nutzerkennung wird so auf 0 gesetzt, was bei Linux die Bezeichnung für den Superuser mit Root-Rechten ist, der vollen Zugriff auf das gesamte System hat. Jemand, der in einem besonderen Kontext den wait4-Systembefehl mit den richtigen Optionen WCLONE bzw. WALL nutzt, erhält also Superuser-Privilegien und vollen Zugriff auf das System; aufgrund der Shortcut-Evaluation von „&&“ geschieht sonst nichts.

Der bösartige Systempatch wurde noch rechtzeitig erkannt; wer hinter dem Angriff stand, ist bislang unbekannt.

Proof-Carrying Code

- Der vollständige Beweis ist in den Code hineingewoben
- Jeder Schritt ist begründet durch eine *Beweisregel* bzw. ein *Axiom* der Sprachsemantik (oder durch klassische *Inferenzen* und *Umformung* der Ausdrücke und Formeln)
- Man kann einen Beweisausdruck als Zusicherung („assertion") dazu ansehen, dass an der jew. Stelle die entsprechende Eigenschaft gilt
- Das Überprüfen, ob der mitgelieferte Beweis korrekt ist („proof checking"), sollte einfach (wenn auch nicht sehr spassig) sein und (wenn die Umformungen in elementare Schritte aufgelöst sind) automatisch erfolgen können

```
static int f(int i, int j){
    int a = i; // a = i
    int b = j; // b = j (Axiome)
    int z = 0; // z = 0
// a × b + z = i × j  (folgt aus obigem)
    while (b != 0) {
// a × b + z = i × j  (zwei "Vorgänger"!)
        if ungerade(b) {
// a × b − a + (z + a) = i × j
            z = z+a;
// a x b − a + z = i x j      (Zuweisung)
// a x (b − 1) + z = i x j
            b = b-1;
// a × b + z = i × j          (Zuweisung)
 b gerade}
// 2a × b/2 + z = i × j
        b = b/2;
// 2a × b + z = i × j         (Zuweisung)
        a = 2*a;
// a × b + z = i × j          (Zuweisung)
    }
// a × b + z = i × j & b=0    (while)
// z = i × j                  (Vereinfachung)
    return z;
}
```

Proof-Carrying Code

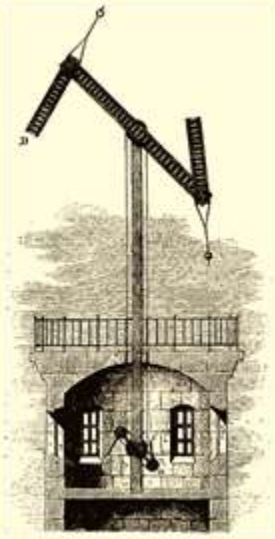
Beweis ist
in den Code hineingewoben

proof checking
automatisch

Stichwort „Semantik“

Welche Wirkung hat ein Programmkonstrukt (z.B. eine Zuweisung, ein if-Statement etc.) genau, das heisst, was ist seine Bedeutung? Irgendwie lernt man dies, manchmal auch durch „trial and error“, und kann es an Andere weitervermitteln – aber wird das auch in komplizierteren Fällen (z.B. bei Nebeneffekten in Ausdrücken) von Allen gleich verstanden? Vor allem für die Entwickler von Compilern ist es wichtig, dass die intendierte Bedeutung aller Konstrukte unmissverständlich festgeschrieben ist, damit sie dies genau so im Compiler implementieren können.

Typischerweise bestehen sowohl formale als auch natürliche Sprachen aus elementaren bedeutungstragenden Elementen, die entsprechend einer Syntax zu komplexeren Konstrukten zusammengefügt werden können. Die Bedeutung zusammengesetzter Konstrukte (und somit z.B. auch ganzer Java-Programme) sollte sich in natürlicher Weise als „Komposition“ aus der Bedeutung der einzelnen Teile ergeben. Gottlob Frege hatte sich damit intensiv auseinandergesetzt („Frege-Prinzip“), er schrieb z.B. in einem Brief an den Logiker Philip Jourdain (1879–1919): „Die Möglichkeit für uns, Sätze zu verstehen, die wir noch nie gehört haben, beruht offenbar darauf, dass wir den Sinn eines Satzes aufbauen aus Teilen, die den Wörtern entsprechen.“



Die Wissenschaft von der Bedeutung von Zeichen ist die Semantik (vom griech. σημα = *Zeichen* bzw. σημαντικός = *bezeichnend* und σημαίνειν = *zum Zeichen gehörig*; vgl. auch *Semaphor*, wörtlich „Zeichenträger“, = *optischer Telegraph* bzw. *Flaggenzeichenalphabet* bei der Marine).

Stichwort „Semantik“ (2)

In die Sprachwissenschaft eingeführt wurde dieser Begriff 1897 vom französischen Philologen Michel Bréal (1832–1915) in seinem Werk *Essai de Sémantique*. Er schreibt dort (zu Beginn des zweiten Buchteils): „Nous nous proposons d’examiner pour quelles causes les mots, une fois créés et pourvus d’un certain sens, sont amenés à le resserrer, à l’étendre, à le transporter d’un ordre d’idées à un autre, à l’élever ou à l’abaisser en dignité, bref à le changer. C’est cette seconde partie qui constitue proprement la *Sémantique* ou science des significations.“ Bréal war der Ansicht, dass nicht das Wort an sich den Sinn einer sprachlichen Äusserung ausmacht, sondern dass dieser sich aus dem grammatikalischen Kontext ergibt, in den das Wort eingebettet ist („une valeur qui ne leur appartient pas en propre, mais qui résulte de la position qu’ils occupent dans la phrase“).

Der formalen Semantik kommt die Aufgabe zu, mit logisch-mathematischen Methoden die Bedeutung formaler, künstlicher Sprachen exakt zu bestimmen und zu interpretieren. Neben Sprachen innerhalb der Logik (wie Aussagenlogik oder Prädikatenlogik) ist die Festlegung der Semantik einer Programmiersprache (die parallel zur Definition ihrer Syntax erfolgt) ein Anwendungsfall davon. Dabei besteht das Ziel darin, die Bedeutung bzw. das Verhalten eines Programms in einer eigenen formalen Sprache syntaktisch (!) so darzustellen, so dass sich mittels Regeln eines Kalküls relevante Aussagen über das Programm (wie z.B. Korrektheit oder die Konformität zu einer ebenfalls formal vorliegenden Spezifikation) beweisen lassen.

Stichwort „Semantik“ (3)

Zu diesem Zweck wurden verschiedene Arten von Semantikkalkülen entwickelt. Dazu gehören u.a. die operationelle Semantik, bei der die möglichen Ausführungsschritte durch eine Zustandsübergangsfunktion beschrieben werden, sowie die oben (mit dem Hoare-Kalkül) beispielhaft ausgeführte axiomatische Semantik.

Bei der operationellen Semantik stellt man sich vor, dass die einzelnen Programmkonstrukte direkt, also ohne dass sie durch einen Compiler übersetzt werden, in einer Folge einzelner Schritte von einer (idealisierten, abstrakten) Maschine ausgeführt (also „interpretiert“) werden und gibt dazu jeweils exakt an, wie sich bei jedem Schritt der Zustand dieser Maschine (also eigentlich der durch die Belegung der Variablen definierte Programmzustand) verändert. Bei abstrakten Maschinen verzichtet man soweit möglich auf (für diesen Zweck) unnötige Details realer Maschinen wie Register oder Speicheradressen; man legt also ein möglichst einfaches, idealisiertes (und mathematisch beschreibbares) Modell eines Berechnungsablaufs zugrunde.

Die axiomatische Semantik beruht auf der Festlegung der logischen Eigenschaft eines jeden Programmfragments, welche unmittelbar zur Bedeutungszuschreibung des Fragments genutzt wird. Die Bedeutung eines Programmkonstrukts wird also mit dem zugehörigen logischen Axiom (bzw. den daraus ableitbaren Zusicherungen im Kalkül) identifiziert und muss nicht weiter interpretiert werden.

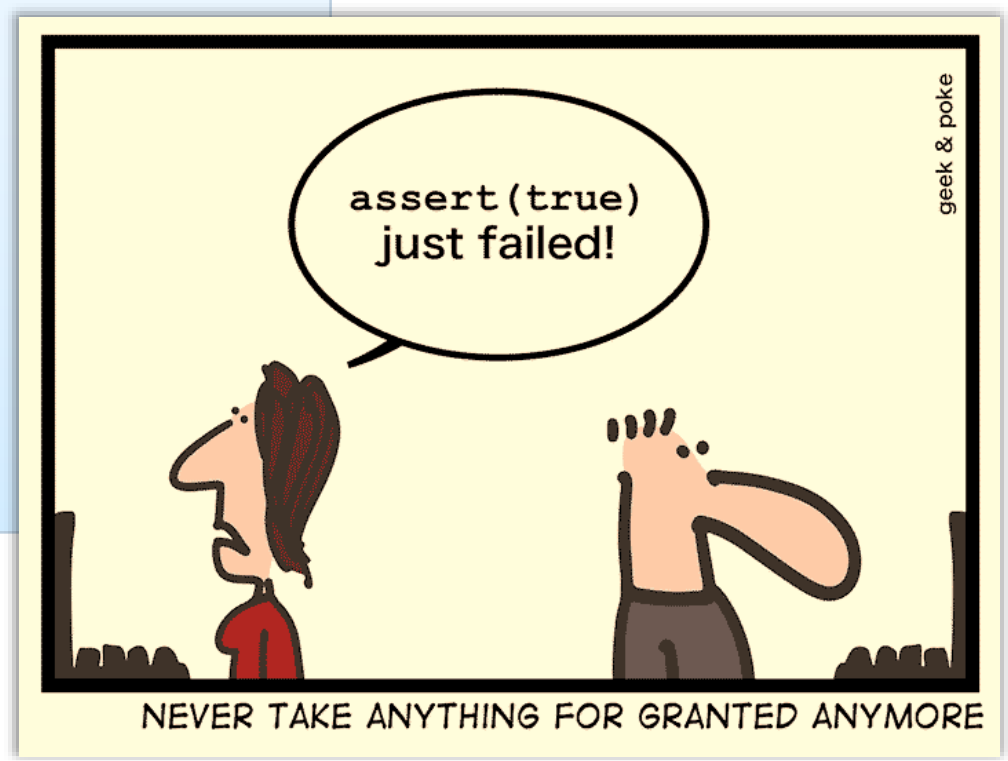
Ein Übungsbeispiel

Wenn ein Logicus Regeln ohne Exempel gibt, es eben sey als wenn man mit blossen Worthen wollte Fechten lehren. -- G.W. Leibniz.

- Man verifiziere analog folgenden Multiplikationsalgorithmus:

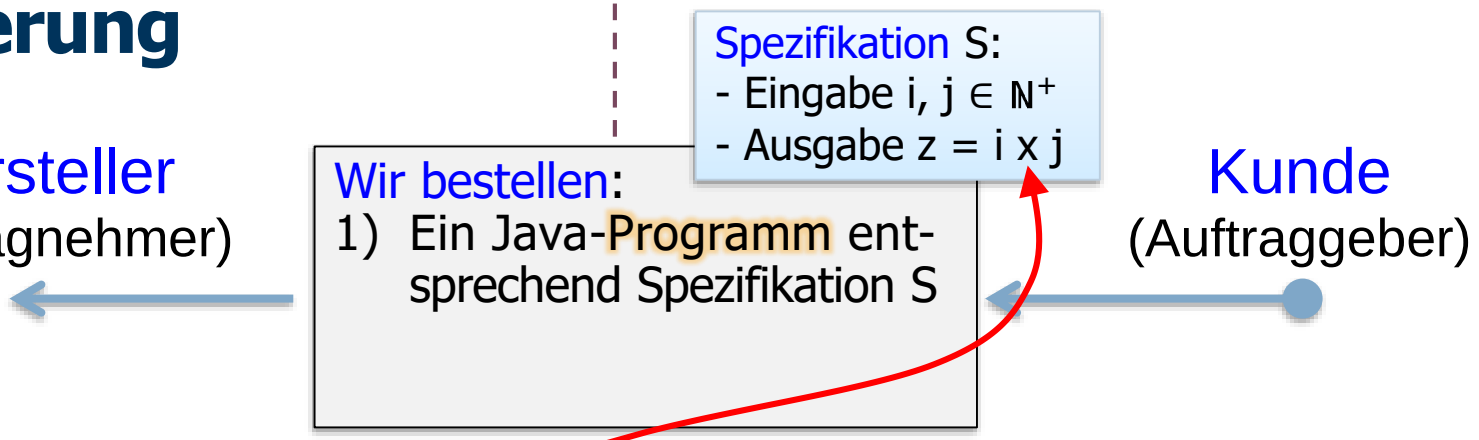
```
static int f(int i, int j) {
// assert i ≥ 0, j ≥ 0;
   int u = i;
   int z = 0;
   while (u > 0) {
      z = z + j;
      u = u - 1;
   }
   return z;
}
```

Tipp: Man betrachte die Werte von **z**, **u×j** und **i×j** an diversen Stellen im Programm



- Nebenbei: Wie gut ist dieses Multiplikationsverfahren relativ zur altägyptischen Methode?

Validierung



Rückfrage: Was soll denn das genau bedeuten?

Naja: j+j+j...+j; halt i-Mal "j" addiert

Geht es etwas präziser, d.h. mathematischer?

Folgendes (induktiv über i definiert):

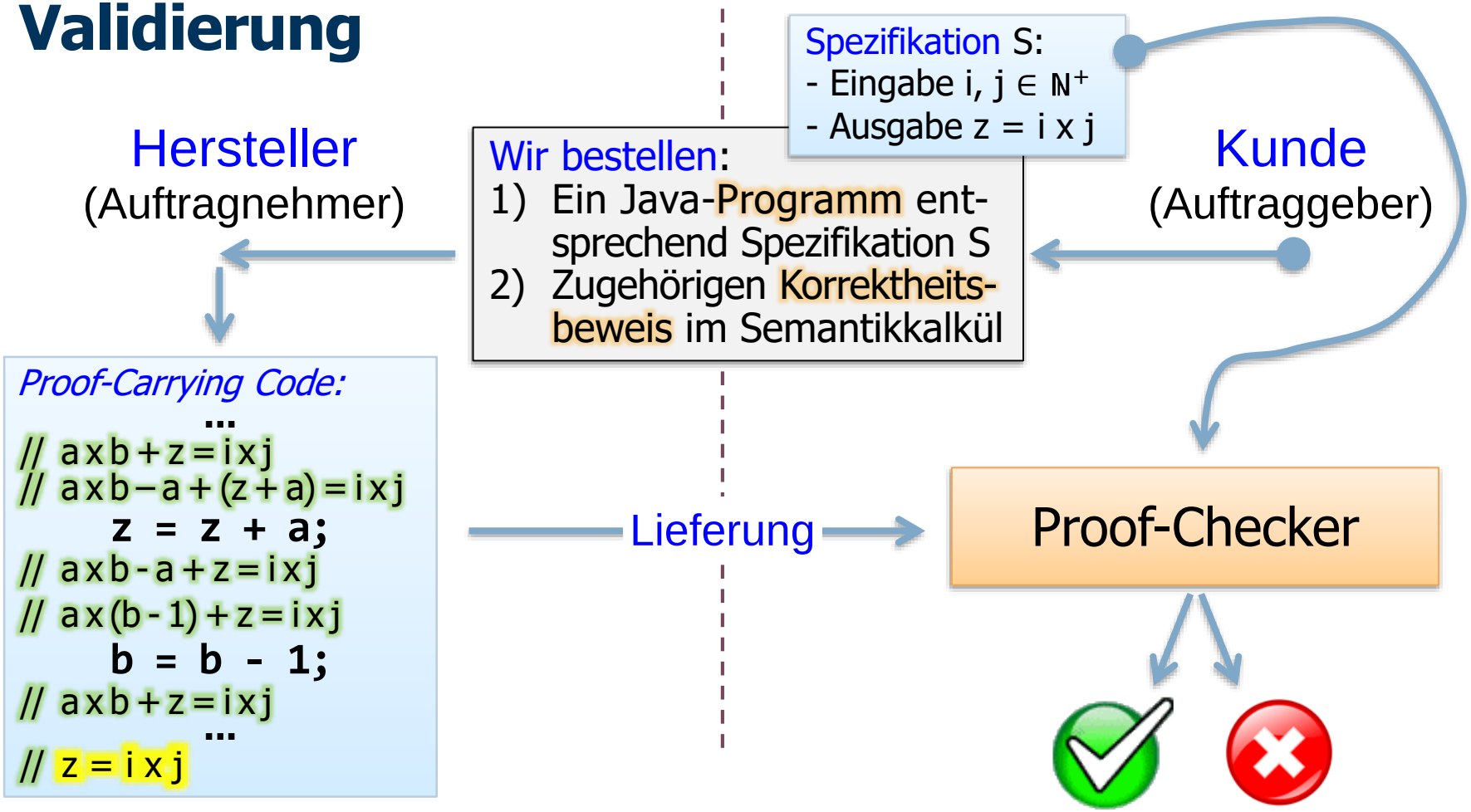
$$1 \times j := j$$

$$(i+1) \times j := i \times j + j$$

Alles klar?

glasklar!

Validierung



- Klappt die Beweis-Validierung, dann ist die Implementierung konform zur Spezifikation (die die geforderten Eigenschaften aufführt)
- Dazu prüft ein (idealerweise automatischer) Proof-Checker jeden Beweisschritt auf logische Korrektheit entsprechend dem Kalkül

Validierung?

Spezifikation S:
- Eingabe i, j ∈ $\mathbb{N}^+$
- Ausgabe z = i x j

Hersteller
(Auftragnehmer)

Wir bestellen:
1) Ein Java-Programm entsprechend Spezifikation S
2) Zugehörigen Korrektheitsbeweis im Semantikkalkül

Kunde
(Auftraggeber)

```
…f(int i, int j){
   int z = 6;
   i = 2;
   j = 3;
// z = i × j
   return (z);
}
```

Lieferung

Proof-Checker

„...sind für die Quality Assurance die Hausjuristen zuständig, die in *end user license agreements* wortreich darlegen, dass die Software halt einfach ist, wie sie ist, dass die Käufer nicht berechtigt sind, irgendwelche Ansprüche zu stellen.“ – Stefan Betschon

Validierung??

Spezifikation S:
- Eingabe i, j ∈ $\mathbb{N}^+$
- Ausgabe z = i x j

Hersteller
(Auftragnehmer)

Wir bestellen:
1) Ein Java-Programm entsprechend Spezifikation S
2) Zugehörigen Korrektheitsbeweis im Semantikkalkül

Kunde
(Auftraggeber)

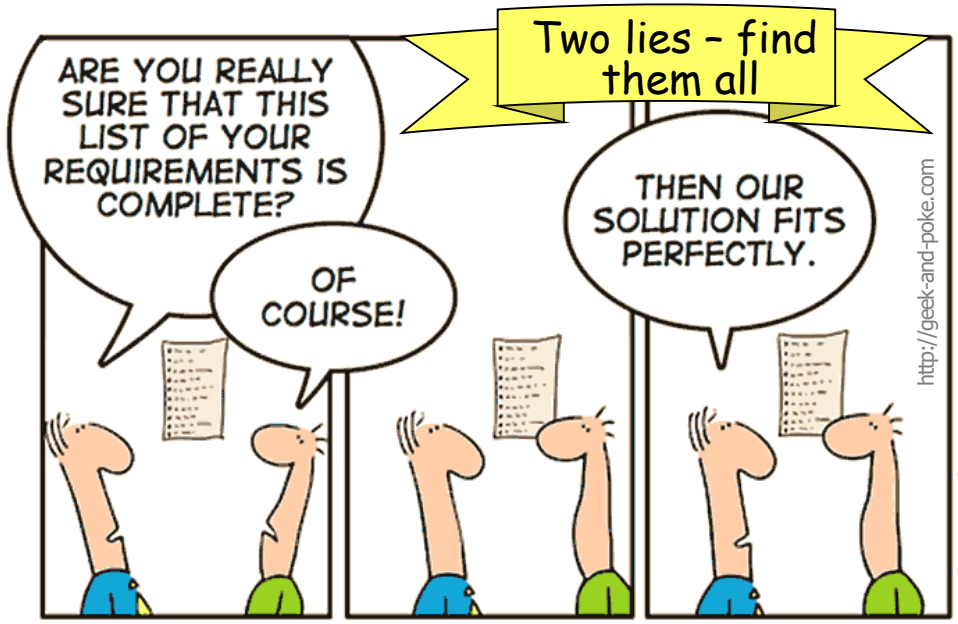
```
…f(int i, int j){
 int a = i;
 int b = j;
 int z = 0;
 while (b > 0){
  if ungerade(b){
   z = z+a;
   b = b-1;
  }
 }
 return z;
}
```

Hier wurde b = b/2; a = 2*a im Schleifenkörper vergessen → Endlosschleife!

Invariante / Beweis gelten aber doch weiterhin, oder?

← Der Hersteller liefert uns diesmal das...

(Dies illustriert den Unterschied zwischen der sog. „partiellen Korrektheit" und „totalen Korrektheit"; nur letztere garantiert Terminierung!)



Binärarithmetik beim altägyptischen Multiplikationsverfahren

Der altägyptische Multiplikationsalgorithmus rechnet implizit mit einer Dualdarstellung von b; diese kann wie folgt explizit gemacht werden:

iterativ (in umgekehrter Bitziffernreihenfolge!)

```
while (b > 0)
{ if ungerade(b)
  {  out("1");
     z = z+a;
     b = b-1;
     b = b/2;
  }
  else out("0");
  b = b/2;
  a = 2*a;
}
```

← Wieso das?

rekursiv

```
static void f(int b)
{ if (b == 1) { out("1"); return; }
  if (b%2==0) { f(b/2); out("0"); }
      else {f((b-1)/2); out("1"); }
}
```

Hierbei steht „out“ als Abkürzung für „System.out.println“

Bitoperationen beim altägyptischen Multiplikationsverfahren

Anstatt Integer-Variablen explizit mit 2 zu multiplizieren oder durch 2 zu dividieren, kann in Java, C oder C++ auch mit den Operatoren zur bitweisen Verschiebung und der Möglichkeit, einzelne Bits auf „0" bzw. „1" zu testen, gearbeitet werden. In der Sprache C sieht das dann z.B. so aus (wobei „b & 1" testet, ob das rechteste Bit von b auf „1" gesetzt ist):

```
int f(int a, int b) {
  int z; z = 0;
  while (b > 0) {
    if (b & 1)  // (bitweise UND)
       z = z + a;
    a = a << 1; // (Verdoppeln)
    b = b >> 1; // (Halbieren ohne Rest)
  }
  return z;
}
```

Bei Java statt dessen:
`if ((b & 1) == 1)`

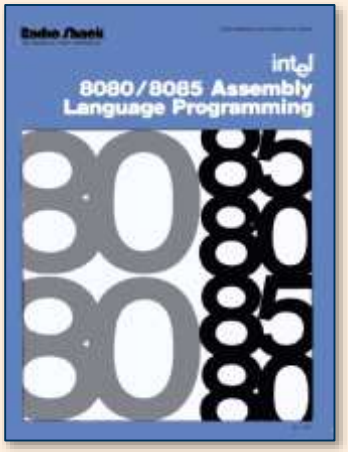
Insbesondere auf Assembler- und Maschineninstruktionsebene ist das bitweise Verschieben von Registerinhalten sehr effizient

"Software Multiply" in Assembly Code

"Back in the 1970's, we had to use this technique on the original Intel PC (8080) processor. Just looking back at my ancient 8080 Assembler book (copyright 1976, by Intel), there's a chapter on Software Multiply and Divide. If you wrote any significant code on the 8080, and needed to multiply, you would have this code snippet in your program. Here's the quote from the book:

Using shift operations provides faster multiplication. Shifting a byte left one bit is equivalent to multiplying by 2, and shifting a byte right one bit is equivalent to dividing by 2. The following process will produce the correct 2-byte result of multiplying a one byte multiplicand by a one byte multiplier:

a) Test the least significant bit of the multiplier. If zero, go to step *b*. If one, add the multiplicand to the most significant byte of the result.
b) Shift the entire two-byte result right one position.
c) Repeat steps *a* and *b* until all 8 bits of the multiplier have been tested.



Explanation: step *a* is checking to see if the multiplier is odd. If so, add the multiplicand to the result (at the HIGH end), and shift it right in step *b* (so later calculations get added in further left than earlier calculations, by specific powers of two).

Below is the 8080 Assembly code, again from the Intel 8080 book. This code snippet multiplies C times D, with the result going into a 2-byte register pair referred to as B (but it's really B and C; 8080 registers were 1 byte, but could be referred to in pairs).

It's pretty clever how C is used initially to hold the multiplier, and with each iteration, the multiplier gets shifted out, while the result gets shifted in. So half way through the algorithm, the right 4 bits of C holds the high order 4 bits of the multiplier, while the left 4 bits of C holds the low order 4 bits of the result.

Hard to imagine that we had to do all this in software, just to multiply two numbers.

"Software Multiply" in Assembly Code (2)

```
MULT:   MVI     B, 0        ; Initialize the most significant byte of result
        MVI     E, 9        ; Bit counter
MULT0:  MOV     A, C        ; Rotate least significant bit
        RAR                 ; of multiplier to carry and
        MOV     C, A        ; shift low-order byte of result
        DCR     E
        JZ      DONE        ; Exit if complete
        MOV     A, B
        JNC     MULT1
        ADD     D           ; Add multiplicand to high-order
                            ; byte of result if bit was one
MULT1:  RAR                 ; Carry = 0 here; shift high-order byte of result
        MOV     B, A
        JMP     MULT0
DONE:
```

MVI R,# Move immediate to register
MOV R,S Move register S to reg. R
RAR Rotate A right through carry
DCR R Decrement register
JZ Jump if zero flag set
JNC Jump if carry flag set
ADD R Add register to A
JMP Unconditional jump

An analogous procedure is used to divide an unsigned 16 bit number by an unsigned 8 bit number. Here, the process involves subtraction rather than addition, and rotate-left instructions instead of rotate-right instructions.

To put this in perspective, to multiply 2 bytes, the 10 or 11 instructions between MULT0 and DONE would get executed 8 times, plus 2 set up instructions at the beginning, so 82 to 90 instructions (at maybe 4 or 5 cycles per instruction) to multiply 2 bytes. You're talking about noticeable fractions of a second just for one multiply! (At 2 MHz – the max speed of the 8080 – if you are lucky, you might be able to perform 5000 byte-multiplies (and nothing else)). Byte-multiplies can multiply an integer less than 256 times another integer less than 256. So TINY whole numbers!"

[www.reddit.com/r/math/comments/zfhre/ethiopian_multiplication_a_method_of_multiplying]

“Software Multiply” in Assembly Code (3)

Beispiel: Multiplikation von hexadezimal 2A × 3C = 9D8

	MULTIPLIER	MULTIPLICAND	HIGH ORDER BYTE OF RESULT	LOW ORDER BYTE OF RESULT
Start	00111100	00101010	00000000	00000000
Step 1	a----------			
	b		00000000	00000000
Step 2	a----------			
	b		00000000	00000000
Step 3	a----------		00101010	00000000
	b		00010101	00000000
Step 4	a----------		00111111	00000000
	b		00011111	10000000
Step 5	a----------		01001001	10000000
	b		00100100	11000000
Step 6	a----------		01001110	11000000
	b		00100111	01100000
Step 7	a----------			
	b		00010011	10110000
Step 8	a----------			
	b		00001001	11011000

Der Intel 8080 (eingeführt 1974) gilt als erster „richtiger“ Mikroprozessor: 8-Bit-CPU, 2MHz-Taktrate, 48 Befehle, 7 8-Bit-Register, 16k-Adressraum, 6µm-Technologie, 6000 Transistoren, 40-Pin-Gehäuse.

Erste Hobby- und Homecomputer wie IMSAI und Altair 8800 nutzten diesen Prozessor, bald danach produzierten auch andere Hersteller ähnliche Mikroprozessoren (Zilog Z80, 6502 von MOS Technology, Motorola 68000 etc.)

1) Test multiplier 0-bit; it is 0, so shift 16 bit result right one bit.
2) Test multiplier 1-bit; it is 0, so shift 16 bit result right one bit.
3) Test multiplier 2-bit; it is 1, so add 2A to high order byte of result and shift 16 bit result right one bit.
4) Test multiplier 3-bit; it is 1, so add 2A to high order byte of result and shift 16 bit result right one bit.
…

Effizienz des Multiplikationsalgorithmus

- Frage: Wie lange dauert eine Multiplikation von a und b?

```
static int f(int a, int b){
  if (b == 1)❶ return a;
  if (b%2 == 0) return f(2*a, b/2);
          ❷ else return a + f(2*a, b/2);
}                              ❺   ❸   ❹
```

- Effizienzmass: Zahl ausgeführter elementarer Operationen
  - Test auf „1" bzw. „gerade", Verdoppeln, Halbieren, Addition
  - Kommen offenbar *pro Aufruf* insgesamt nicht mehr als 5 Mal vor

---

- Aber: Handelt es sich bei „b%2 == 0" nicht um *zwei* Operationen statt einer einzigen?
- Antwort: So wie der Test auf „gerade" hier realisiert ist, werden für „b%2 == 0" tatsächlich mehr Maschineninstruktionen ausgeführt als z.B. für „b == 1"; wir wollen aber von der konkreten Realisierung abstrahieren – auf Maschineninstruktionsebene lässt sich ein Test auf „gerade" jedenfalls einfach und effizient realisieren, indem man testet, ob das rechteste Bit der Speicherzelle von b eine 0 oder eine 1 ist.

Effizienzabschätzung

- Genauere Überlegungen müssten die unterschiedlichen „Kosten" der verschiedenen Operationen berücksichtigen
  - Verdoppeln ist „billiger" als Addieren etc.
  - Grosse Zahlen sind „teurer" als kleine Zahlen
  - Vereinfachend setzen wir aber alle Operationen als gleich „teuer" an

Kosten ≈ Zeit

- Wesentliches Kriterium: Anzahl der rekursiven Aufrufe
  - Dies ist nur von b abhängig, nicht von a

```
if (b == 1)   return a;
if (b%2 == 0) return f(2*a, b/2);
     else return a + f(2*a, b/2);
```

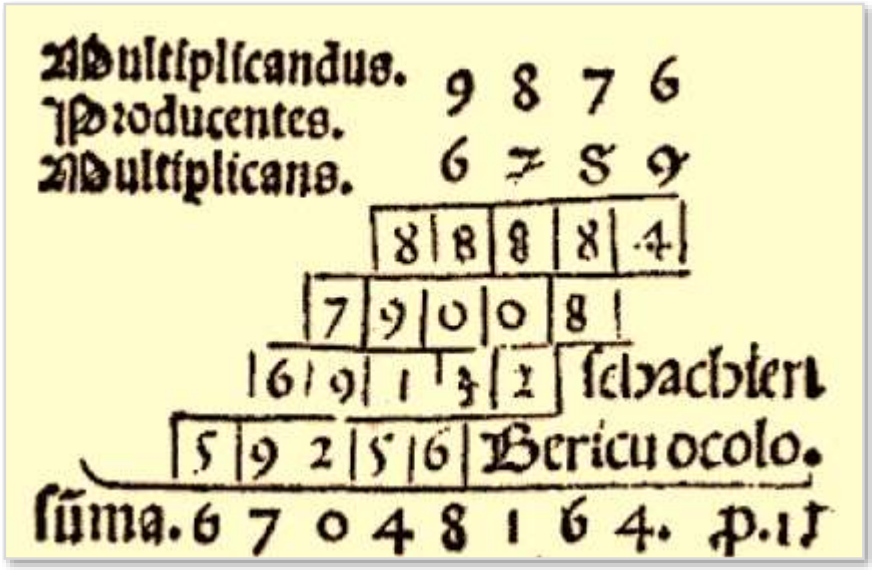
- Also: wie viele rekursive Aufrufe gibt es?
  - Wie oft kann man b halbieren, bis der Wert 1 herauskommt?
    (durch das Abrunden beim Halbieren ist dies konservativ geschätzt)
  - $b / (2^x) = 1 \Rightarrow x = \log_2 b$
- Es werden also nicht mehr als $5 \log_2 b$ Operationen benötigt
  - Analog bei iterativer Lösung: Nicht mehr als $\log_2 b$ Schleifendurchläufe

Aufwands- / Qualitätsvergleich

- Aus der Schule bekannt: „schriftliche“ Multiplikation
  - Wieso lernt man eigentlich nicht das altägyptische Prinzip stattdessen?

- Diese ist jedenfalls auch ein altbekanntes Rechenschema:

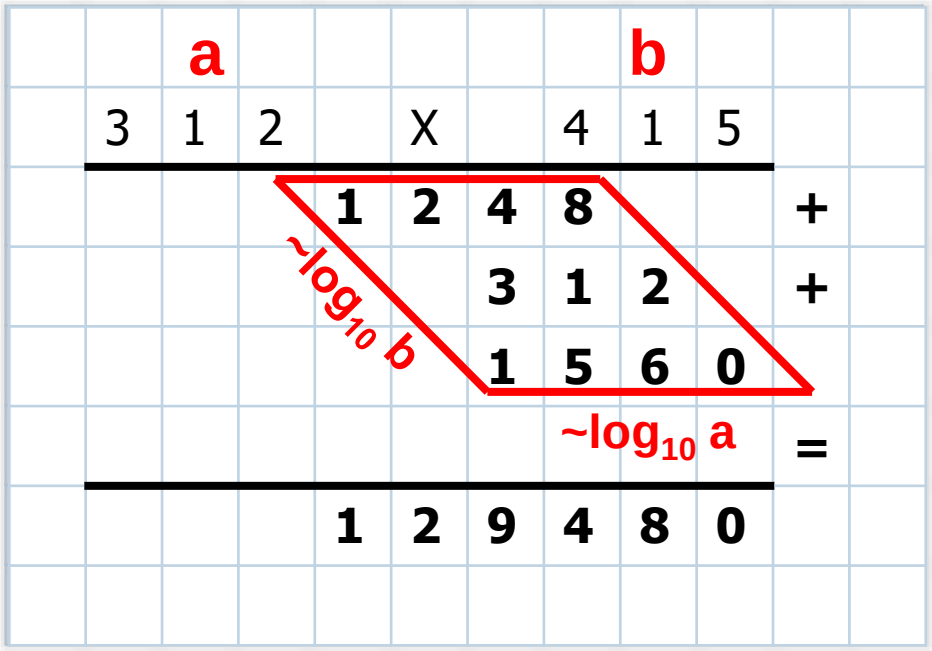
Rechnen ist nicht das Ausführen mathematischer Operationen nach mathematischen Methoden, sondern das mechanische Manipulieren von Symbolen nach vorgegebenen Regeln. -- Dirk Siefkes



Aufwands- / Qualitätsvergleich (2)

- Aus der Schule bekannt: „schriftliche“ Multiplikation
  - Wieso lernt man eigentlich nicht das altägyptische Prinzip stattdessen?

- Wie gut ist diese „Schulmethode“?
  - Wie viele (vergleichbar teure) Elementaroperationen?



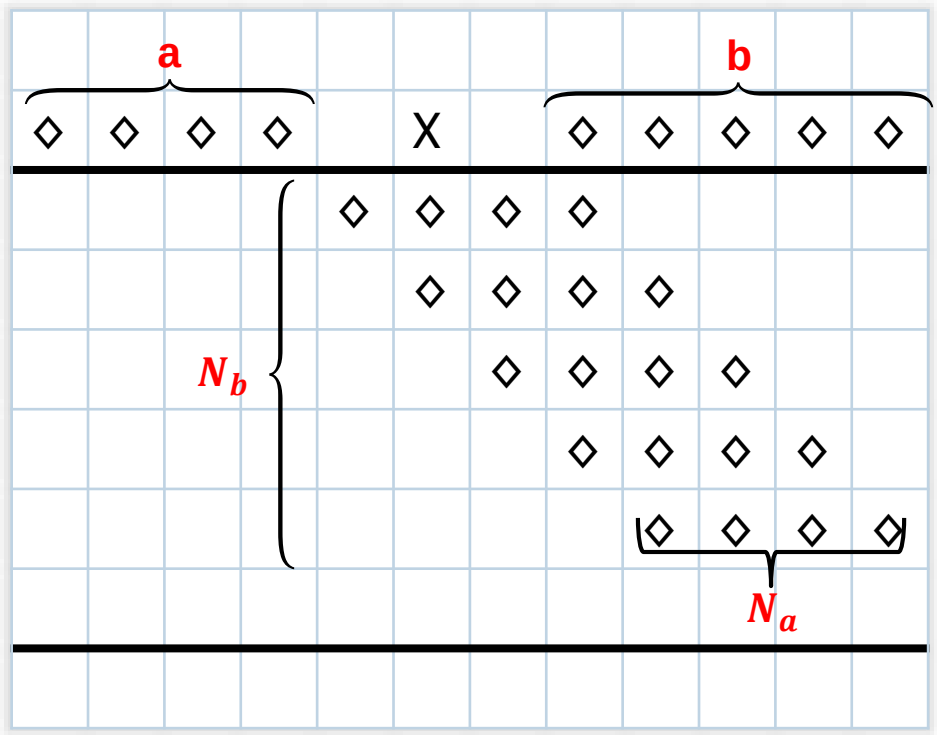
Wir erinnern uns: Man multipliziert den Multiplikator „312“ nacheinander mit jeder Ziffer des Multiplikanden und schreibt die jeweiligen Resultate richtig verschoben untereinander; anschliessend addiert man spaltenweise Ziffer für Ziffer alle erhaltenen Zwischenresultate.

Jede einzelne Ziffer ist das Resultat von maximal 2 Elementaroperationen (kleines Einmaleins und evtl. Übertrag)

Für die Spaltensummen wird jede Ziffer ein 2. oder 3. Mal genutzt

Wie gut ist $3\ (\log_{10} a)\ (\log_{10} b)$ im Vergleich zu $5 \log_2 b$?

Aufwands- / Qualitätsvergleich (3)



$$N_a = 1 + \lfloor \log_{10} a \rfloor$$

$$N_b = 1 + \lfloor \log_{10} b \rfloor$$

≈ 0.3 $\log_2$ b

→ $N_a \times N_b$ „elementare" Multiplikationen (*kleines Einmaleins kann man auswendig!*) und etwa doppelt so viele Additionen von Ziffern, also insges. ca. 3 ($\log_{10}$ a) ($\log_{10}$ b) elementare Operationen

Wann ist 5 $\log_2$ b besser als 3 ($\log_{10}$ a) ($\log_{10}$ b)?

- Die multiplikativen Konstanten 5 bzw. 3 sind nicht exakt, das ist aber nicht so relevant
- Hängt wesentlich von a ab: 3 ($\log_{10}$ a) ($\log_{10}$ b) kann (bei festem b) mit grösserem a über alle Grenzen wachsen!

Wie schnell kann man überhaupt multiplizieren?

- Wenn man prinzipielle Überlegungen anstellt, sollte man die Länge der Faktoren berücksichtigen und elementarste Operationen verwenden.
- Sinnvoll ist es dann, Binärzahlen zu betrachten.
- Lange vermutete man, dass man in jedem Fall quadratischen Aufwand in der Länge n der Operanden (also deren Bitzahl) benötigt.
- Dann aber zeigte der 23-jährige Student Anatoli Karatsuba (1937-2008), dass die Grössenordnung $n^{\log 3}$ genügt (log zur Basis 2 mit log 3 = 1.58496…). Das zugrundeliegende allgemeine Prinzip wurde später „Divide and Conquer" genannt.
- Veröffentlicht wurde dies 1962: Карацуба А.А., Офман Ю.П. Умножение многозначных чисел на автоматах. Докл. АН СССР. 1962. Т. 145, № 2. С. 293-294. [A.A. Karatsuba, Yu. P. Ofman: Multiplikation mehrstelliger Zahlen mit Automaten. Akademie der Wissenschaften der UdSSR, 1962, 145(2), 293-294.]
- 1971 wurde von Schönhage und Strassen gezeigt, dass sogar O(n (log n) log (log n)) genügt, wenn man Prinzipien der schnellen Fouriertransformation anwendet.

www.ccas.ru/personal/karatsuba/karatzuba.jpg

Kleines Einmaleins kann man auswendig, oder?

Lern das ein mal eins mit fleis,
so wirstu aller rechnung weis.

[Johann Fischer: Ein kurtz Rechenbüchlein für anfahende Schüler gemacht. Ca. 1580]

Lern auszwendig das Einmal ein/ So würt Dir all Rechnung gemein.

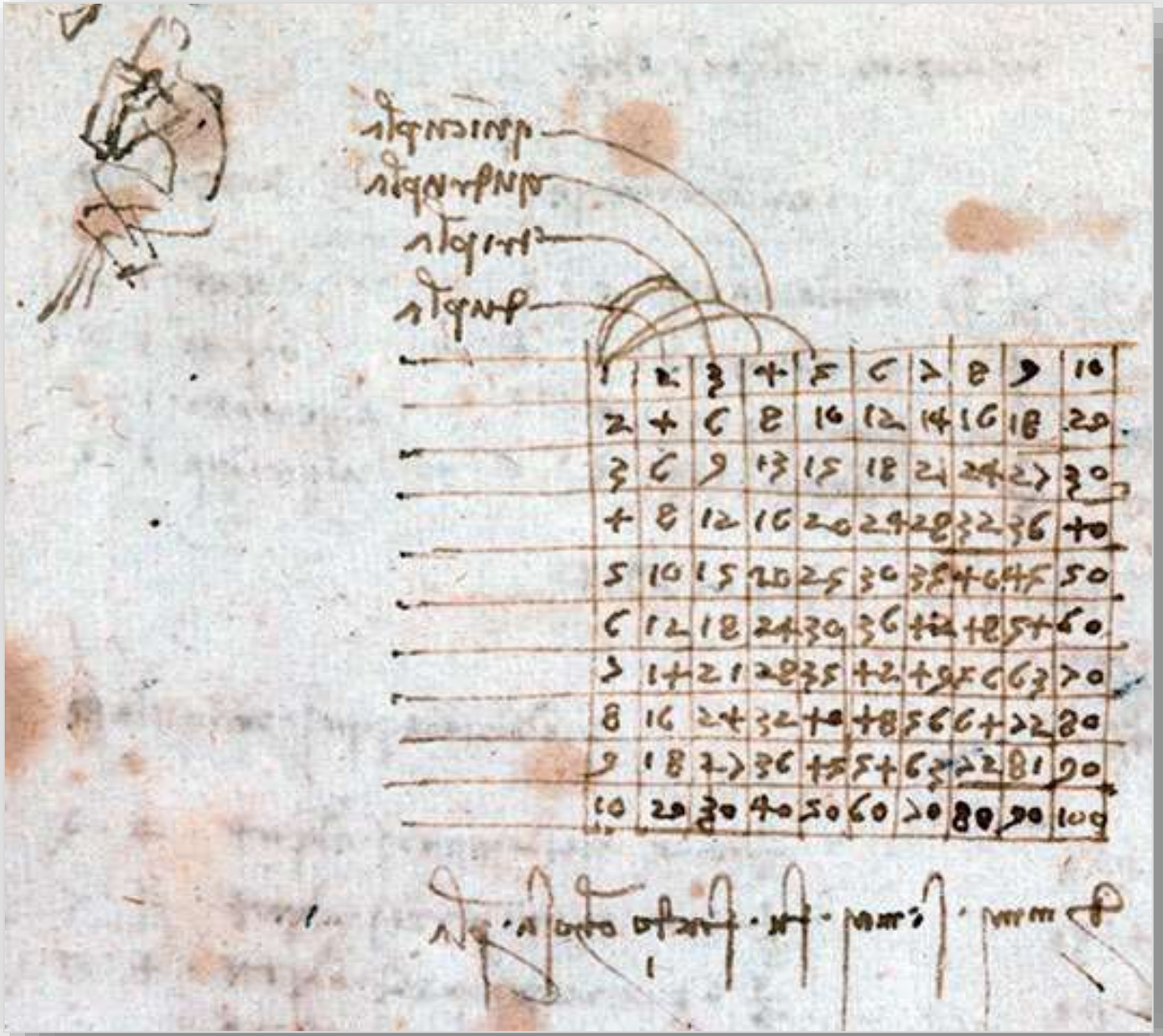
„Zum Ersten soltu wissen…“
Jacob Köbel: Ein newü Rechenpüchlein. Oppenheim, 1522. ETH-Bibl. Rar 2355

XXXV.

Zum Erſtē ſoltu wiſſen/das die zale um der erſten vnd oberſten linien/die mit den büchſtaben a vnd c gezeichet iſt/auch die zal in der neben linien/gegen der lincken handt/mit den Büchſtaben a vnd b vermerckt/annfäng vnd gebererin ſeind aller anderer zalen/die in dem Pythagoriſchen Tiſch geſchrieben ſteend. Dañ die zale 4/6/8/wer dē geborē vō der zal 2. Vnd 6/9/ 12/ 15 2c. entſpringen von der zal 3 2c. Vnnd alle anderñ zale/haben der maß/von den zalen/in den obenangezeygten linien a b vnd a c/ iren vrſprungk/anefangk/vnd fundament.

Eigentlich müsste es „Tafel“ heissen, nicht „Tisch“ – es handelt sich um eine Fehlinterpretation des lateinischen Wortes „tabula“!

Leonardo da Vincis (1452 – 1519) Multiplikationstafel



www.bl.uk/manuscripts/Viewer.aspx?ref=arundel_ms_263_f001r

"Codex Arundel" (British Library): Notebook of Leonardo da Vinci. A collection of papers written in Italian, in his characteristic left-handed mirror-writing, covering a broad range of topics in science and art, as well as personal notes. The core of the notebook is a collection of materials that Leonardo describes as 'a collection without order, drawn from many papers, which I have copied here, hoping to arrange them later each in its place according to the subjects of which they treat'."

ε × ζ = λε

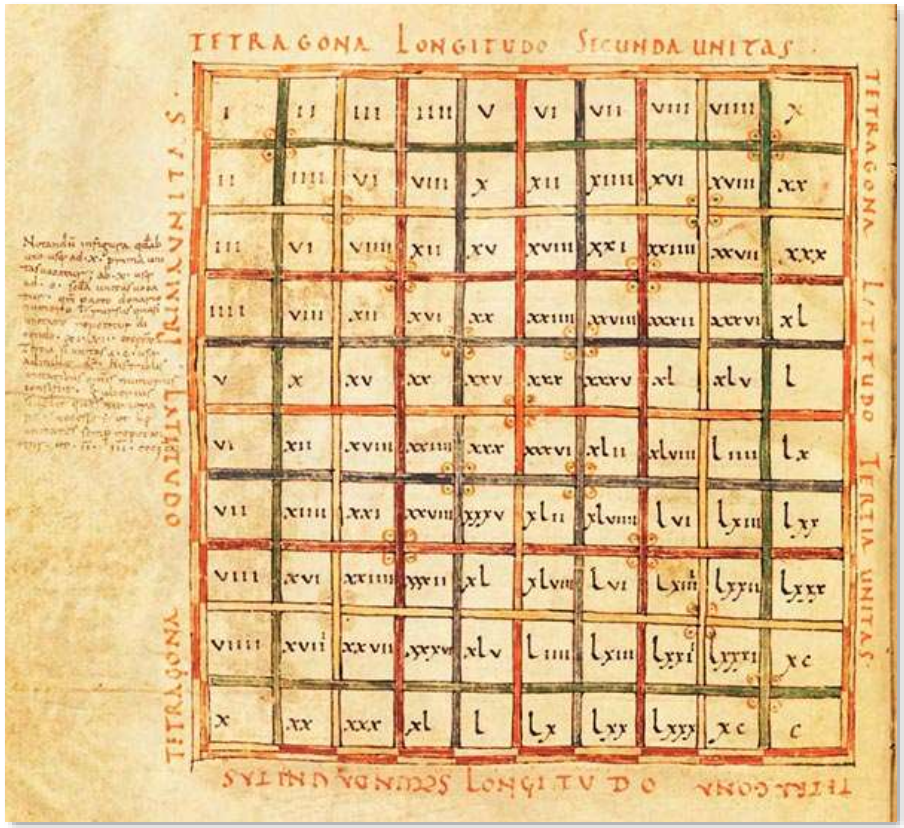
Epsilon × Zeta = Lambda-Epsilon

www.lib.umich.edu/writing-graeco-roman-egypt/multiplication.html

Griechische Multiplikationstafel (2. – 3. Jh., Karanis, Ägypten). Die Multiplikationstafel illustriert, wie die Griechen Buchstaben zur Darstellung von Zahlen nutzten. In der ersten vollständigen Zeile steht ε ς λ (*Epsilon, Sigma, Lambda*), was „5, 6, 30“ bedeutet. In der zweiten Zeile steht ε ζ λε (*Epsilon, Zeta, Lambda-Epsilon*) mit der Bedeutung „5, 7, 35“. Die ersten beiden Zahlen ergeben als Produkt die dritte Zahl; dieses Schema wurde im Papyrus für Zahlen bis in die Tausende fortgeführt.

VI × VII = XLII

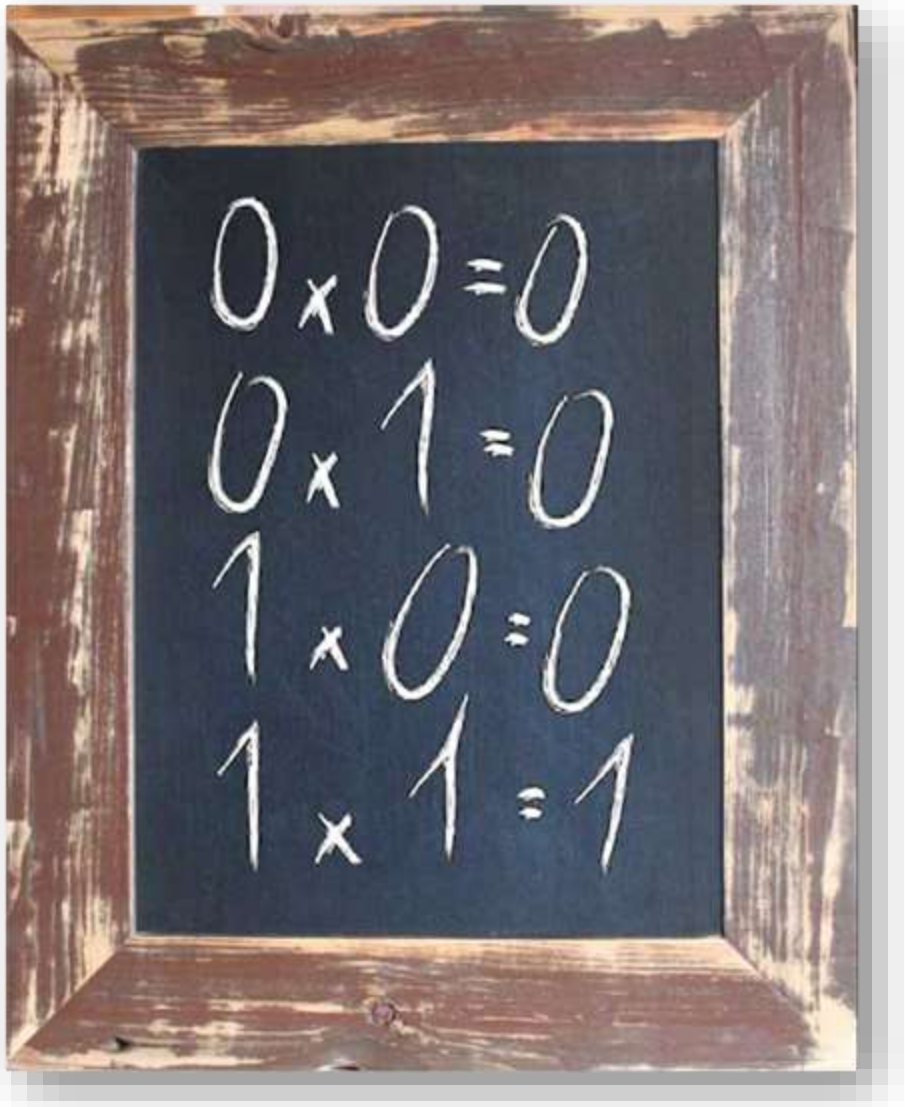
sex × septies = quadraginta duo



www.mechrech.info/exhibit/eEinmal/Einmaleins.html

Einmaleins-Tafel mit römischen Ziffern in einer Handschrift Ende des zehnten oder Anfang des elften Jh.

Das ganz kleine Einmaleins im Binärsystem



	0	1
0	0	0
1	0	1

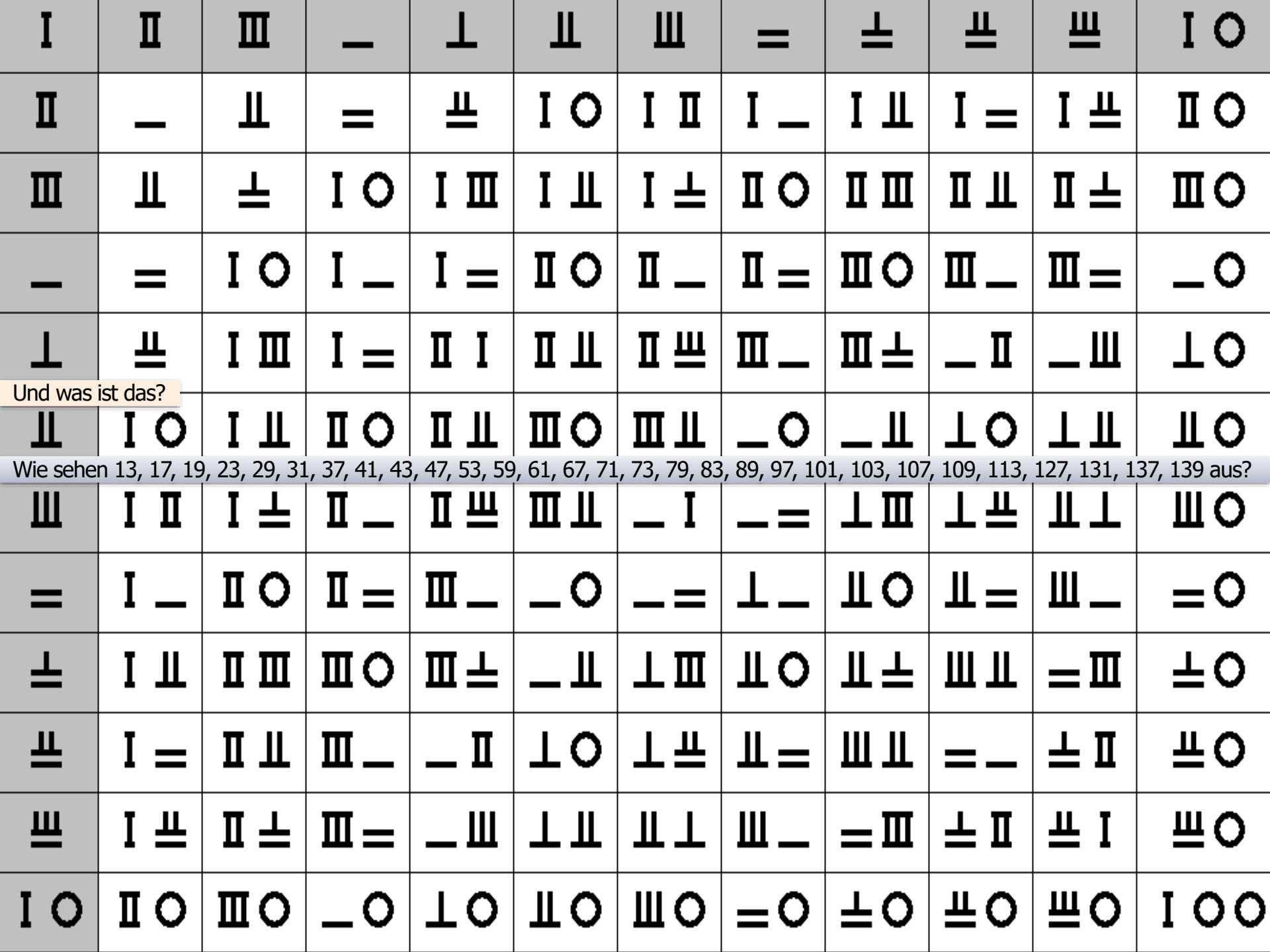
Wenn, wie üblich, Null-zeile / -spalte wegfallen:

	1
1	1

Minimal sogar so:

1

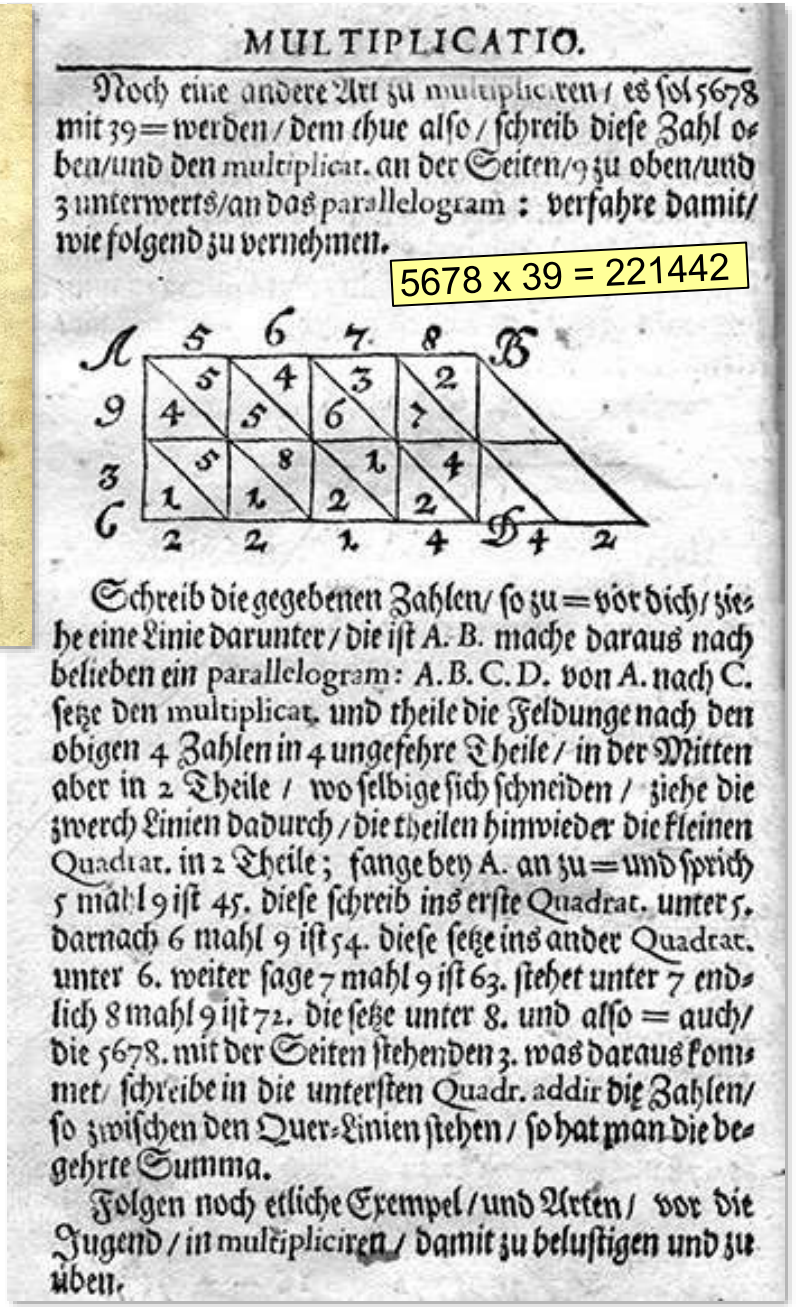
„On n'a point besoin non-plus de rien apprendre par cœur icy" -- Leibniz


Und was ist das?
Wie sehen 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139 aus?

Multiplicatio

Andreas Georgius Schütz(e): Arithmetischer Wegweiser / Zu der so genanten Italiänischen Practic: Samt deroselben kurtze und gründliche Erklärung / in allerhand vorfallenden Handels Regeln solvirt. Der Schul-Jugend zum besten in Teutsch und Schwedischer Sprache.

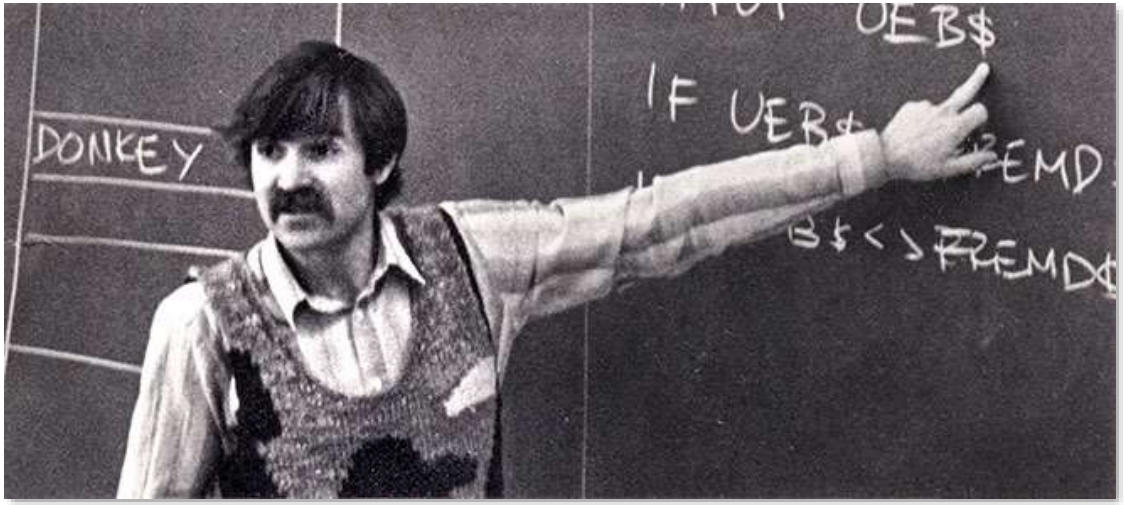
Arithmetiſcher
Wegweiſer/
Zu der
ſo genanten
Italiäniſchen
PRACTIC
Samt deroſelben kurtze und gründliche Erklärung/ in allerhand vorfallenden Handels Regeln ſolvirt/
Der Schul-Jugend zum beſten in Teutſch und Schwediſcher Sprache
heraus gegeben durch
ANDREAM GEORG. SCHÜTZEN,
Philo Mathemat. beſtalten Schreib- Rechenmeiſter und Buchhalter der Teutſchen Schulen in Stockholm.
Gedruckt in der von J. K. M. priv. Burchardiſchen Druckerey/ durch J. J. Genath. Fact.
Im Jahr deſſen Tripli + 741 iſt Radix cubica 18.


MULTIPLICATIO.
Noch eine andere Art zu multipliciren/ es ſol 5678 mit 39 = werden/ dem thue alſo/ ſchreib dieſe Zahl oben/ und den multiplicat. an der Seiten/ 9 zu oben/ und 3 unterwerts/ an das parallelogram: verfahre damit/ wie folgend zu vernehmen.
5678 x 39 = 221442
Schreib die gegebenen Zahlen/ ſo zu = vor dich/ ziehe eine Linie darunter/ die iſt A. B. mache daraus nach belieben ein parallelogram: A. B. C. D. von A. nach C. ſetze den multiplicat. und theile die Feldunge nach den obigen 4 Zahlen in 4 ungefehre Theile/ in der Mitten aber in 2 Theile/ wo ſelbige ſich ſchneiden/ ziehe die zwerch Linien dadurch/ die theilen hinwieder die kleinen Quadrat. in 2 Theile; fange bey A. an zu = und ſprich 5 mahl 9 iſt 45. dieſe ſchreib ins erſte Quadrat. unter 5. darnach 6 mahl 9 iſt 54. dieſe ſetze ins ander Quadrat. unter 6. weiter ſage 7 mahl 9 iſt 63. ſtehet unter 7 endlich 8 mahl 9 iſt 72. die ſetze unter 8. und alſo = auch/ die 5678. mit der Seiten ſtehenden 3. was daraus kommet/ ſchreibe in die unterſten Quadr. addir die Zahlen/ ſo zwiſchen den Quer-Linien ſtehen/ ſo hat man die begehrte Summa.
Folgen noch etliche Exempel/ und Arten/ vor die Jugend/ in multipliciren/ damit zu beluſtigen und zu üben.

In diesem Buch wird nochmal ein anderer Multiplikationsalgorithmus angegeben. Der Autor, Mitglied der 1690 gegründeten „Kunst-Rechnungs lieb- und übenden Societät“ in Hamburg (nur wer mindestens quadratische und kubische Gleichungen, „dabei aber auch die vornehmsten und nötigsten Fundamenta Euclidea nebst sattem Verstande numerorum irrationalium et binomiorum“ beherrschte, konnte dort Mitglied werden), war in Stockholm als Schreib- und Rechenmeister tätig. Stockholm hatte bis in die frühe Neuzeit eine mehrheitlich deutschstämmige Bevölkerung; der Handel über die Ostsee wurde seit dem 14. Jahrhundert von der Deutschen Hanse beherrscht. Mit den Kaufleuten wanderte auch die Praxis des Ziffernrechnens und der Buchhaltung im 16. Jahrhundert nach Skandinavien. Zum Erscheinungsjahr des Rechenbuches vermerkt die Titelseite: „Im Jahr dessen Tripli + 741 ist radix cubica 18“.

Kosten ≈ Zeit?

„Jede Minute kostet 33 Franken im Rechenzentrum der ICS-Corporation in Zürich. Das steht auf Schildern, welche der Schichtleiter Martin Kern überall anbringen lässt. Damit seine Operatoren ständig vor Augen haben, warum die Computer Tag und Nacht laufen müssen.“ (Limmat-Verlag, 1977)



Emil Zopfi *(geb. 1943 in Wald ZH, Kindheit in Gibswil) studierte, nach einer Berufslehre bei Zellweger AG in Uster zum Fernmelde- und Elektroapparatemonteur, Elektrotechnik am Technikum Winterthur und arbeitete als Programmierer und Systemingenieur am Institut für Physikalische Chemie der ETH Zürich, bei Siemens in Karlsruhe und bei IBM in Zürich.*

Kosten ≈ Zeit?

„Jede Minute kostet…" heisst der Titel der Taschenbuchausgabe in Deutschland. Die 33 Franken musste man weglassen, nicht nur, weil Computerleistung in der Zwischenzeit billiger geworden ist, sondern weil in Deutschland „Franken" keine Währungseinheiten sondern Menschen der Region „Franken" im Nordwesten Bayerns sind.

Emil Zopfi:
Nachwort nach 25 Jahren

Der Schriftsteller Otto F. Walter bezeichnete in einer Fernsehrezension das vorliegende Buch als «Frühwarnsystem», als «ferner junger Verwandter jenes Homo Faber, mit dem Max Frisch 1959 die Frage nach der Verantwortung des Technikers für unsere Gesellschaft warnend gestellt hat». «Jede Minute kostet 33 Franken» erschien am 1. Mai 1977 in einer Zeit, als Computer noch grosse und teure Anlagen waren, nur wenigen Spezialisten zugänglich und verständlich.

Es ist die alte Welt der elektronischen Datenverarbeitung, die ich in diesem Roman gestaltete, weitgehend aus eigener Erfahrung und im Bewusstsein, dass sich hier eine Technologie und Kultur entwickelt, die in Zukunft die Gesellschaft radikal verändern wird. Ich wollte von dem Unbekannten erzählen, das ich kannte und von dem ich überzeugt war, dass es schon bald das Leben und den Alltag vieler bestimmen werde. Die Technologie war noch so fremd, dass der Limmat Verlag das Wort «Computer» im Titel vermeiden wollte, da es zu wenig geläufig sei. Das Buch begründete auch das literarische Programm des Verlags.

Nach einem Vierteljahrhundert sind Computer nun aller Welt bekannt, viele Menschen nutzen sie privat und im Beruf. Die ökonomischen und politischen Folgen sind tiefgreifend, die weltweite Vernetzung der Wirtschaft, oft «Globalisierung» genannt, wäre ohne Computertechnologie undenkbar. In diesem Sinne ist die Bezeichnung «Frühwarnsystem» sicher treffend. www.zopfi.ch/0e/Minute.html

Elementaroperationen

- Wir haben mit der altägyptischen Multiplikationsmethode die Multiplikation auf die Addition zurückgeführt

a	b
6	9
~~12~~	~~4~~
~~24~~	~~2~~
48	1
54	

- Elementaroperationen waren dabei
  - Halbieren
  - Verdoppeln
  - Test auf gerade / ungerade (bzw. =1)

- Könnte man auf diese Elementaroperationen evtl. verzichten und mit noch weniger Grundrechenkenntnissen auskommen?
  - Ja: Diese lassen sich durch sich selbst rekursiv definieren und benötigen sonst nur noch das Inkrementieren (+1) und Dekrementieren (-1)

→ nächste Seite

Die drei Operationen als rekursive Funktionen:

```
static boolean gerade(int x){
  if (x == 0) return true;
  return !gerade(x-1);
}
```

Negation

Der Wertebereich sei $\mathbb{N}^0$

Datentyp aus den beiden Werten `true` und `false`

```
static int verdopple(int x){
  if (x == 0) return 0;
  return 2 + verdopple(x-1);
}
```

```
static int halbiere(int x){
  if (x == 0) return 0;
  if (x == 1) return 0;
  return 1 + halbiere(x-2);
}
```

Was ergibt Halbieren ungerader Zahlen?

Statt -2 (bzw. +2) könnten wir sogar -1-1 bzw. +1+1 schreiben. Man braucht also eigentlich nur eine Maschine, die inkrementieren und dekrementieren kann (sonst keine Arithmetik) und würde z.B. schreiben:

```
return incr(halbiere(decr(decr(x))))
```

Ein funktionales Programm

- Die Multiplikationsfunktion f sieht dann so aus:

Keine Schleifen

```
static int f(int a, int b){
  if (b == 0) return 0; // Rekursionsende bei b = 0
  if gerade(b) return f(verdopple(a),halbiere(b));
  else return add(a, f(verdopple(a),halbiere(b)));
}
```

Keine Wertzuweisung an Variablen

Die Methode `add` rekursiv formuliert als Übungsaufgabe

- Das ganze erscheint etwas gekünstelt und unnötig komplex
  - Es soll hier auch nur das Prinzip illustrieren
  - Algorithmus implementiert im „funktionalen“ Stil!

Funktionale Programmiersprachen

- Man kommt tatsächlich im Prinzip stets ohne explizite Zuweisungen und ohne Schleifen aus!
  - Man könnte gewissermassen jedes Programm auch einfach in Form einer „mathematischen Funktion“ hinschreiben

- Konsequent umgesetzt ist dies bei funktionalen Programmiersprachen (z.B. Lisp): im Gegensatz zu „imperativen Programmiersprachen“ (wie z.B. Java) enthalten diese
  - Keine Variablen (d.h. „Speicherzellen“), die im Programmablauf unterschiedliche Werte zugewiesen bekommen können
  - Keine Schleifen – wären sowieso nutzlos, wenn es keine Wertzuweisungen gibt!

Ist Dekrementieren elementar?

Omnibus ex nihilo ducendis sufficit unum. -- G.F. Leibniz.

- Kann man einer Maschine, die nur Vorwärtszählen („incr“) kann, das Rückwärtszählen „per Programm“ beibringen?
- Ja, wir benutzen dazu eine rekursive Hilfsmethode h:

```
static int h(int x, int y){
   if (x == incr(y)) return y;
   else return h(x, incr(y));
}
```

Bei $y \geq x$ terminiert die Methode nicht; das ist für unseren Zweck aber irrelevant

- Ein Aufruf h(5, 4) liefert offenbar 4
- Allgemein: h(u, u-1) liefert u-1
- Der Aufruf h(5, 3) führt zu h(5, 4) in der Rekursion
- Also liefert **h(u, 0)** schliesslich h(u, u-1) = **u-1** (für u > 0)

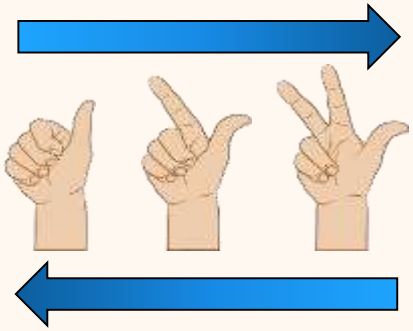
Ist Dekrementieren elementar? (2)

- Wegen h(u,0) = u-1 lässt sich decr (für z ∈ $\mathbb{N}^0$) so definieren:

```
static int decr(int z){
   if (z == 0) return 0;
   return h(z,0);
}
```

Da wir im Bereich der natürlichen Zahlen bleiben wollen (und nicht ins Negative rutschen wollen), ist decr(0) als 0 definiert.

- Rückwärtszählen lässt sich durch Vorwärts-zählen „simulieren" bzw. implementieren!
  - Erstaunlich?



- → Fragestellung der theoretischen Informatik: Was muss ein Computer („in Hardware") mindestens können, damit er (mit geeigneter Software) als „Universalrechner" funktioniert?

Langsame Multiplikation beschleunigen?

- Wir nehmen nochmal einen frischen Anlauf bei der Multiplikation, für die nach Definition gilt: $n \times a = \underbrace{a + a + \ldots + a}_{\text{n-mal}} = \underbrace{(a + a + \ldots)}_{\text{(n-1)-mal}} + a = (n-1) \times a + a$ (für $n \geq 1$)

- Dies kann man direkt in eine rekursive Methode umsetzen:

```
static int f(int n, int a){
  if (n == 1) return a;
  return f(n-1,a) + a;
}
```

```
if (n%2 == 0) // Abkz.
    return f(n/2,a*2);
```

- Aber können wir die Sache nicht abkürzen, indem wir n/2 mit dem Doppelten von a multiplizieren, falls n eine gerade Zahl ist?
  - Denkübung: (1) Wieviel bringt diese Abkürzung?
  - (2) Wir fordern n≥1, aber darf a negativ oder 0 sein?

Verallgemeinerung der altägypt. Multiplikation (1)

Das Prinzip der altägyptischen Multiplikation $a \times b$ kann man auch so verstehen:

- Die Zahl b lässt sich als $b = 2c + k$ mit $k \in \{0, 1\}$ schreiben
- Dann ist $a \times b = a \times (2c + k) = 2(a \times c) + a \times k$
  - wobei c nur noch etwa halb so gross wie b ist und
  - k bestimmt, ob a hinzuaddiert wird (k = 1) oder nicht (k = 0)
- Das „kleinere Problem" $a \times c$ wird dann rekursiv gelöst

→ „Fortgesetztes Verdoppeln und gelegentliches Addieren"

```
a = 2 * a;
```

```
if ... z = z + a;
```

(„double and add")

Verallgemeinerung der altägypt. Multiplikation (2)

- Ganz analog kann nun auch die Potenz a^b berechnet werden:

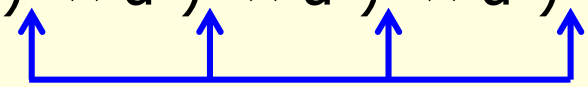
  $a^b = a^{2c+k} = (a^c)^2 \times a^k$ [mit a^k entweder 1 oder a, da $k \in \{0, 1\}$]

- Dies kann man auch ein bisschen anders wie folgt darstellen:
- Angenommen, wir wollen a^b mit b = 13 berechnen, also a^{13}. Gemäss dem Horner-Schema kann die Dualdarstellung $1101 = \mathbf{1} \times 2^3 + \mathbf{1} \times 2^2 + \mathbf{0} \times 2^1 + \mathbf{1} \times 2^0$ von 13 so ausgewertet werden:

  $13 = ((((0) \times 2 + \mathbf{1}) \times 2 + \mathbf{1}) \times 2 + \mathbf{0}) \times 2 + \mathbf{1}$

- Es ist also $a^{13} = a^{((((0) \times 2 + \mathbf{1}) \times 2 + \mathbf{1}) \times 2 + \mathbf{0}) \times 2 + \mathbf{1}}$, und unter Anwendung der Regeln für die Potenzrechnung ergibt sich:

  $a^{13} = ((((a^0)^2 \times a^{\mathbf{1}})^2 \times a^{\mathbf{1}})^2 \times a^{\mathbf{0}})^2 \times a^{\mathbf{1}}$ [mit $a^0 = 1$ und $a^1 = a$]



→ „Fortgesetztes Quadrieren und gelegentliches Multiplizieren“

Entsprechendes Rekursionsprinzip:

$$x^n = \begin{cases} (x^2)^{n/2}, & \text{falls n gerade} \\ x(x^2)^{(n-1)/2}, & \text{falls n ungerade} \end{cases}$$

Verallgemeinerung der altägypt. Multiplikation (3)

- Allgemein gilt (für $b \geq 1$): $a^b = (a^{\lfloor b/2 \rfloor})^2 \times a^{(b \bmod 2)} = (a^2)^{\lfloor b/2 \rfloor} \times a^{(b \bmod 2)}$. Statt b Multiplikationen bei „naiver" Berechnung von a^b benötigt man nur ca. $\log_2 b$ Halbierungen und max. $2 \log_2 b$ Multiplikationen – eine gewaltige Verbesserung!
- Schnelle binäre Exponentiation („square & multiply"): Die Idee kann man nun benutzen, um „schnell" Potenzen mit grossen ganzzahligen Exponenten zu berechnen: Der Exponent wird schrittweise halbiert und die Basis quadriert, dabei werden die Potenzen mit ungeraden Exponenten „aufmultipliziert". (Beim Potenzieren in einem *Restklassenring* bildet man nach jeder Rechenoperation gleich den *Rest*, um so zu verhindern, dass die berechneten Zahlen zu gross werden.) Die schnelle *modulare* Exponentiation (bzw. „diskrete Exponentialfunktion") ist für die Kryptographie zentral; die Umkehrfunktionen (diskreter Logarithmus; Wurzel) können i.Allg. nicht „schnell" berechnet werden (→ „Einwegfunktion").
- Das Verfahren wird zum Beispiel bei der RSA-Verschlüsselung zur Berechnung von $a^b \bmod M$ angewendet; in Chipkarten spielt dabei zur Authentifizierung der Karte der intern gespeicherte geheime Schlüssel b die Rolle des Exponenten. Hacker könnten durch genaues Messen der Zeit („timing attack") und der Energie („power analysis"), die ein Schleifendurchlauf braucht, so allerdings herausfinden, ob der „if-Zweig" bei „if ungerade (b)" ausgeführt wird oder nicht, und damit darauf schliessen, ob die jeweilige Bitstelle des geheimen Schlüssels eine 0 oder eine 1 ist; auf diese Art würde schliesslich schrittweise der ganze Geheimschlüssel verraten! Erläuterung dazu („Seitenkanalangriff") folgt gleich.

Von der Multiplikation zur Exponentiation a^b

```
// Schnelle Exponentiation
static int f(int a, int b)
{ int z = 0;
  while (b > 0)
  { if ungerade(b)
      z = z + a;
    b = b / 2;
    a = a + a;
  }
  return z;
}
```

1; // neutr. // Element

z * a; // mult. // statt add.

a * a; // quadrieren // statt verdoppeln

Der Algorithmus ist auch unter „square and multiply“ bekannt.

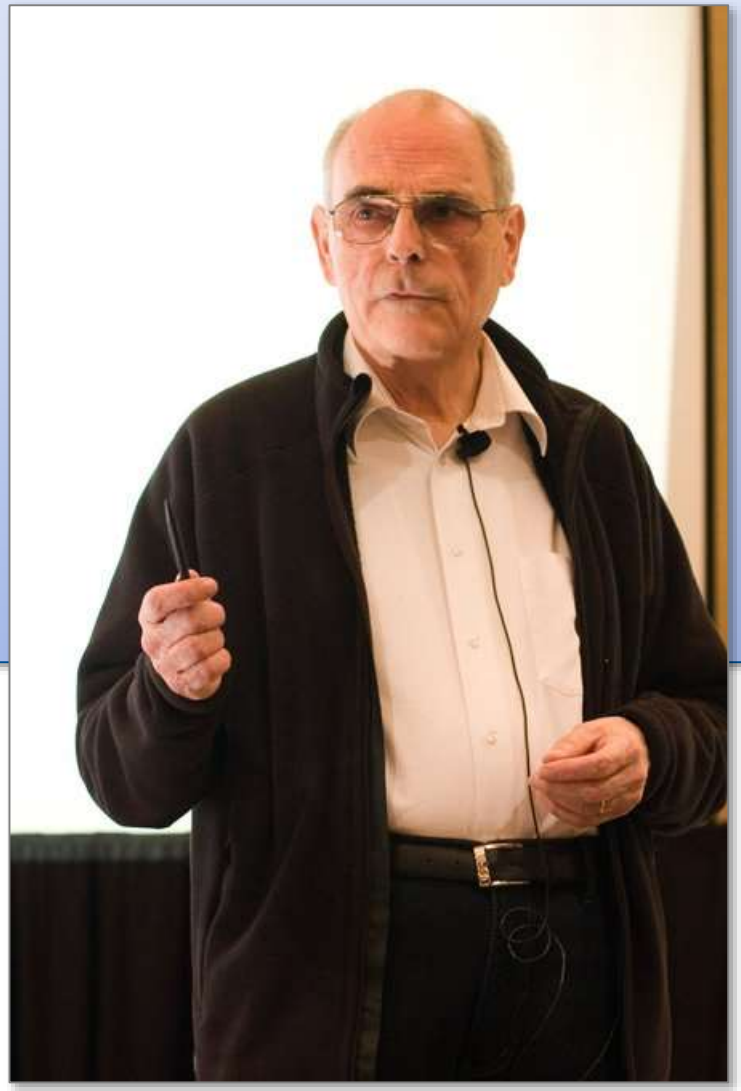
Eine häufige Anwendung ist das Potenzieren modulo m.

Bei a^b mod m wird bei jedem Quadrieren und Multiplizieren gleich mod m gerechnet, damit die Zwischenergebnisse nicht zu gross werden; hierfür existieren wiederum spezielle Algorithmen, die effizient bzgl. Zeit und Platz sind.

- Es wird jetzt die Potenz a^b (statt dem Produkt $a \times b$) berechnet
  - Formaler Korrektheitsbeweis völlig analog zur altägyptischen Multiplikation!
- Der Algorithmus ist für die Kryptographie hochrelevant, dort wird a^b (mod m) oft für sehr grosse Parameter (> 1000 Bits) benötigt; erst mit so grossen Zahlen werden die Verfahren sicher
  - → Effizienz wichtig, weitere Beschleunigung evtl. durch spezielle Hardware

„Mein Interesse gilt nun aber viel grösseren Zahlen, sagen wir tausendstelligen. Allein schon das Zählen bis zu einer solchen Zahl ist völlig unmöglich und wird es, unabhängig von zukünftigen Technologien, wohl auch bleiben. Erinnern wir uns daran, dass die Anzahl der Atome im Weltall auf weniger als 10^{80} geschätzt wird und dass 10^{80} eine 80-stellige Zahl ist, so sehen wir: 1000-stellige Zahlen kommen in der Natur als Anzahlen (disjunkter Objekte) nicht vor, und die Mathematik dieser Zahlen ist keine Naturwissenschaft. Solche Zahlen codieren vielmehr mathematische Strukturen wie die Ringe $\boldsymbol{Z}_n$ und andere (z.B. geheime) Informationen. Ihre Mathematik ist Strukturwissenschaft."

-- Volker Strassen



www.ics.uci.edu/~eppstein/pix/strassen/StrassenTalk4-m.jpg

Volker Strassen, 2009

Schnelle („binäre“) Multiplikation und Exponentiation nochmal anders betrachtet:

$$a \times b \overset{\text{Def.}}{=} \underbrace{a + \underbrace{a + a + \ldots + a}_{(b-1)\text{-Mal}}}_{b\text{-Mal}}$$

- Die Multiplikation kann entsprechend so für a, b ∈ $\mathbb{N}^+$ rekursiv berechnet werden:

$$f(a,b) = \begin{cases} a & \text{falls } b = 1 \\ \xleftarrow{\text{optionaler Booster}} & \boldsymbol{f(2a, b/2) \text{ falls } b \text{ gerade}} \\ a + f(a, b-1) & \text{sonst} \end{cases}$$

Der Trick der altägyptischen Multiplikation!

- Die Exponentiation in analoger Weise so:

Statt b zu dekrementieren, reduziert der Booster b durch Halbieren viel schneller

$$f(a,b) = \begin{cases} a & \text{falls } b = 1 \\ \xleftarrow{\text{optionaler Booster}} & \boldsymbol{f(a^2, b/2) \text{ falls } b \text{ gerade}} \\ a \times f(a, b-1) & \text{sonst} \end{cases}$$

Denn $a^b = (a^2)^{b/2}$

Schnelle Exponentiation in Ruby

- Die Programmiersprache Ruby vereinigt viele bewährte Konzepte und Paradigmen anderer Programmiersprachen; sie ist vor allem bei der Programmierung von Server-Anwendungen und (im Sinne einer Skriptsprache) bei der Administration von Serveraufgaben beliebt. Der Algorithmus zur schnellen binären Exponentiation sieht in Ruby z.B. so aus:

```
def expo(a, b)
  z = 1
  while b.nonzero?
    if b[0].nonzero?
      z *= a
      b -= 1
    end
    a *= a
    b /= 2
  end
  return z
end
```

Beispiel 3^{10}: a = 3; b = 10
z := 1 (vor der Schleife)

1. Iteration
b = 10 → b ist gerade (rechtes Bit b[0] ist 0)
a := $a^2 = 3^2 = 9$
b := b / 2 = 5

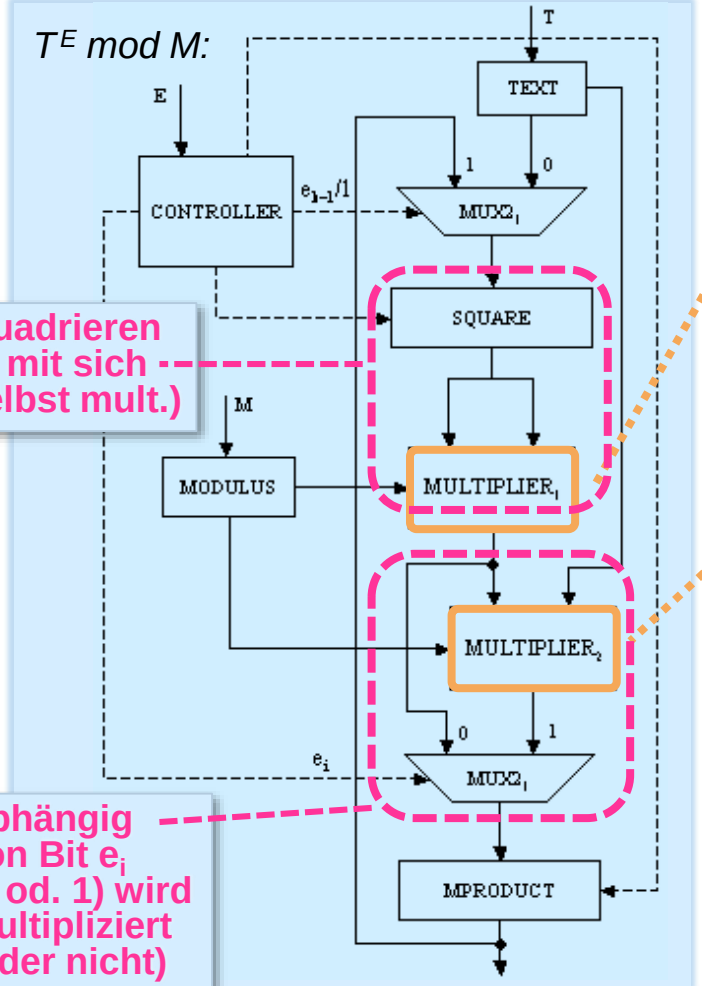
2. Iteration
b = 5 → b ist ungerade
z := z * a = 1 * a = 1 * 9 = 9
b := b - 1 = 4
a := $a^2 = 9^2 = 81$
b := b / 2 = 2

3. Iteration
b = 2 → b ist gerade
a := $a^2 = 81^2 = 6561$
b := b / 2 = 1

4. Iteration
b = 1 → b ist ungerade
z := z * a = 9 * 6561 = 59049 (Resultat)
b := b - 1 = 0 (→ Schleife wird verlassen)

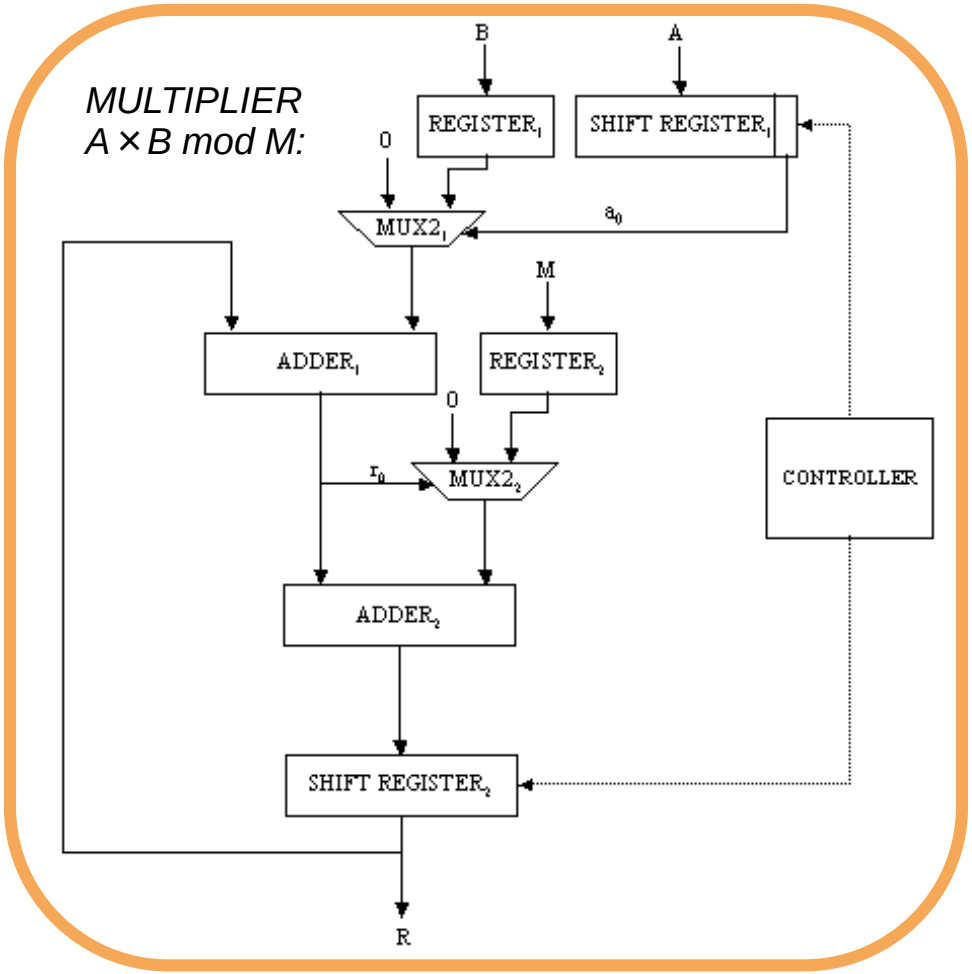
Schnelle Exponentiation in Hardware

- Z.B. als Koprozessor oder als digitale Schaltung in Chipkarten
  - *Cryptographic acceleration engine* mit Wortlängen von mehreren 1000 Bits



Quadrieren (= mit sich selbst mult.)

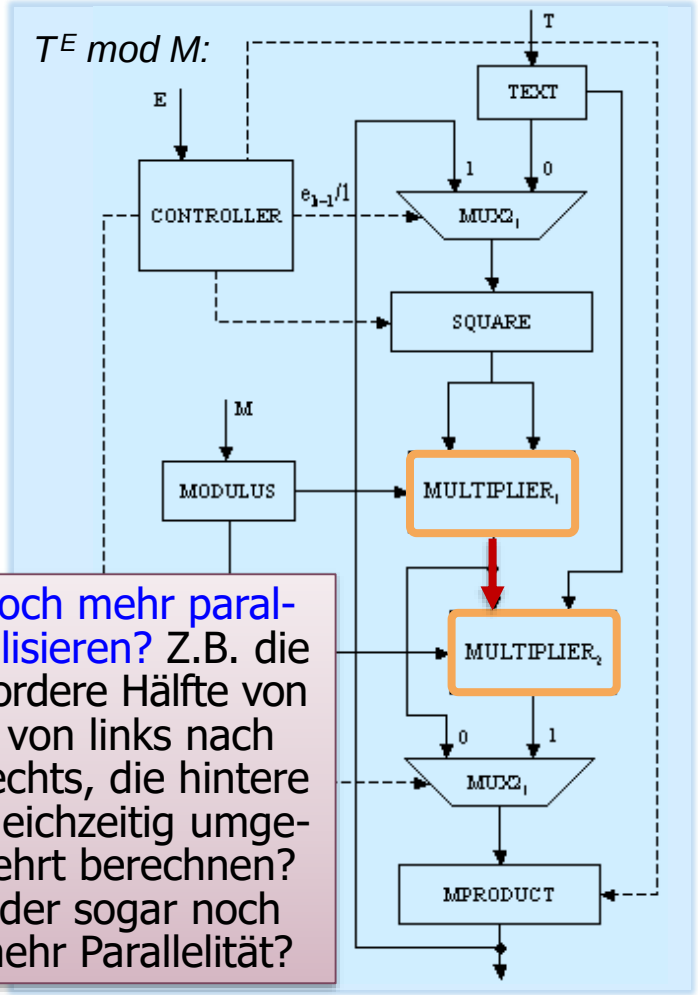
Abhängig von Bit e_i (0 od. 1) wird multipliziert (oder nicht)



N. Nedjah: Software/Hardware Co-Design of Efficient and Secure Cryptographic Hardware, j-jucs, 2005, Vol 11 No 1, 66-82

Schnelle Exponentiation in Hardware

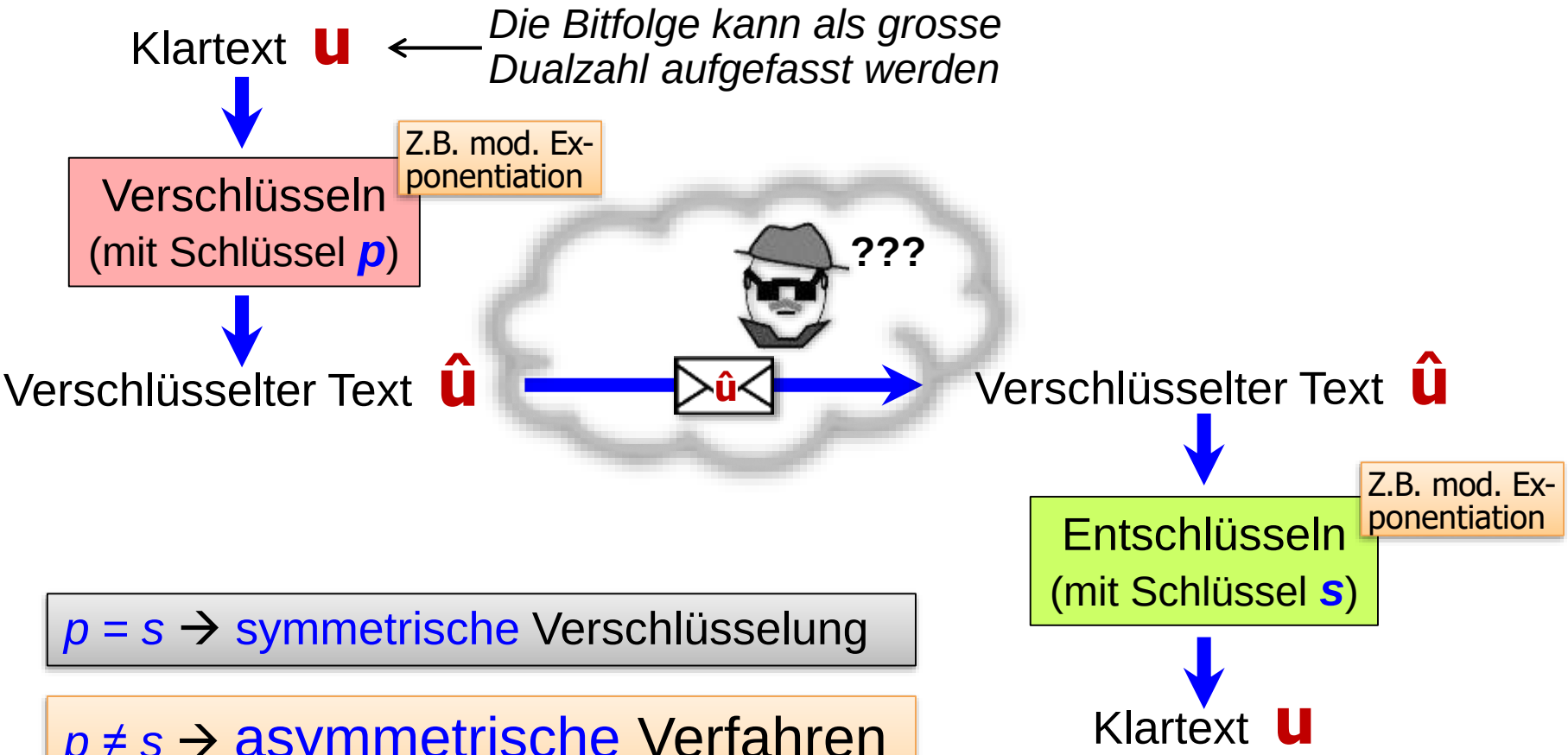
- Z.B. als Koprozessor oder als digitale Schaltung in Chipkarten
  - *Cryptographic acceleration engine* mit Wortlängen von mehreren 1000 Bits



Noch mehr paral-lelisieren? Z.B. die vordere Hälfte von B von links nach rechts, die hintere gleichzeitig umge-kehrt berechnen? Oder sogar noch mehr Parallelität?

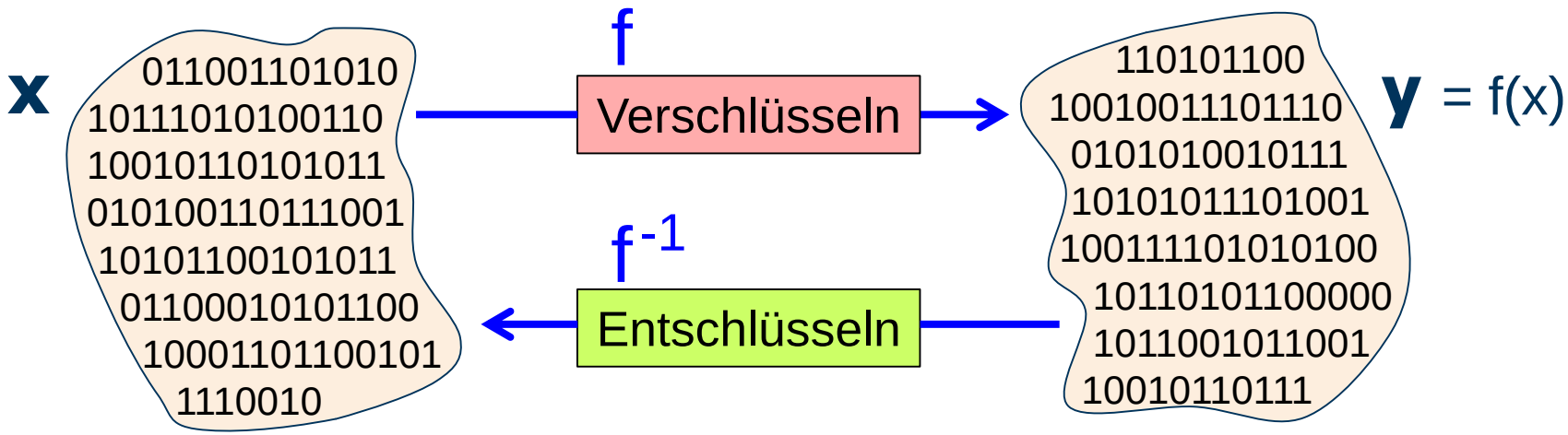
- Die Schaltung enthält einen Signalpfad von Multiplier 1 (Quadrierung) zu Multiplier 2.
- Dies bedeutet eine Datenabhängigkeit, so dass die beiden Multiplizierer nicht parallel arbeiten können.
- Der weiter vorne gezeigte Algorithmus „most significant bit" enthält tatsächlich im Schleifenkörper diese Abhängigkeit:
  - `z = (z*z)% M;`
    `if (…) z = (z*a)% M;`
- Bei „unserer" Variante hingegen könnten die beiden Multiplikationen gleichzeitig in einem einzigen Takt durchgeführt werden:
  - `if (…) z = (z*a) % M;`
    `…`
    `a = (a*a)% M;`

Verschlüsseln in der Kryptographie



Verschlüsseln als Bijektion auf $\{0,1\}^n$

- f bijektiv auf $\{0,1\}^n$
  - Für n >> 1000



- Geeignet: $\mathbf{f(x) = x^p \bmod m;\ f^{-1}(y) = y^s \bmod m}$
  - Mit zueinander passenden *p*, *s* (und *m* in der Grössenordnung 2^n)
  - Notwendig: Effiziente Exponentiation für sehr grosse Operanden
  - Klartext *x* lässt sich nicht (in ausreichend effizienter Weise) nur aus dem verschlüsseltem Text x^p und *p* ermitteln; d.h., die Funktion $x^p \bmod m$ lässt sich nicht einfach umkehren – Kenntnis von *s* ist dafür notwendig

Verschlüsseln mit dem RSA-Verfahren

Rivest, Shamir, Adleman, 1978

- Verschlüsseln von Nachrichten nach dem Public-Key-Prinzip

Sicherheit beruht darauf, dass für die Primfaktorzerlegung grosser Zahlen sowie die Umkehrfunktionen der Exponentiation in $\mathbb{Z}_m$ keine effizienten Algorithmen bekannt sind (hier nicht weiter ausgeführt)

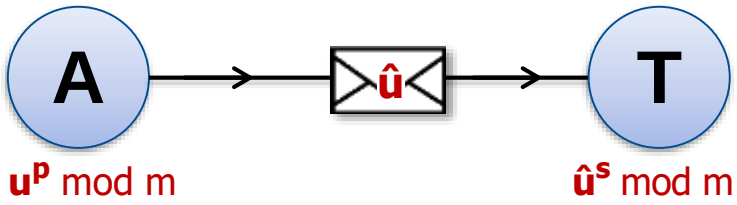
- Für jeden Teilnehmer T gibt es ein Schlüsselpaar (p, s)
  p – public key für T (kennt jeder!)
  s – secret key von T (kennt nur T; z.B. in einer Chipkarte von T enthalten)

p, q und m werden passend zueinander (ausgehend von grossen zufälligen Primzahlen) so bestimmt, dass die unten angegebenen Operationen Ver- / Entschlüsseln invers zueinander sind.

- Verschlüsseln einer Nachricht u für T: $\hat{u} = u^p \bmod m$

Mod. Exponentiation!

- „Offenes" Verschicken der verschlüsselten Nachricht û an T
  - Denn niemand kann û (ohne Kenntnis von s) entziffern (hier ohne Beweis)

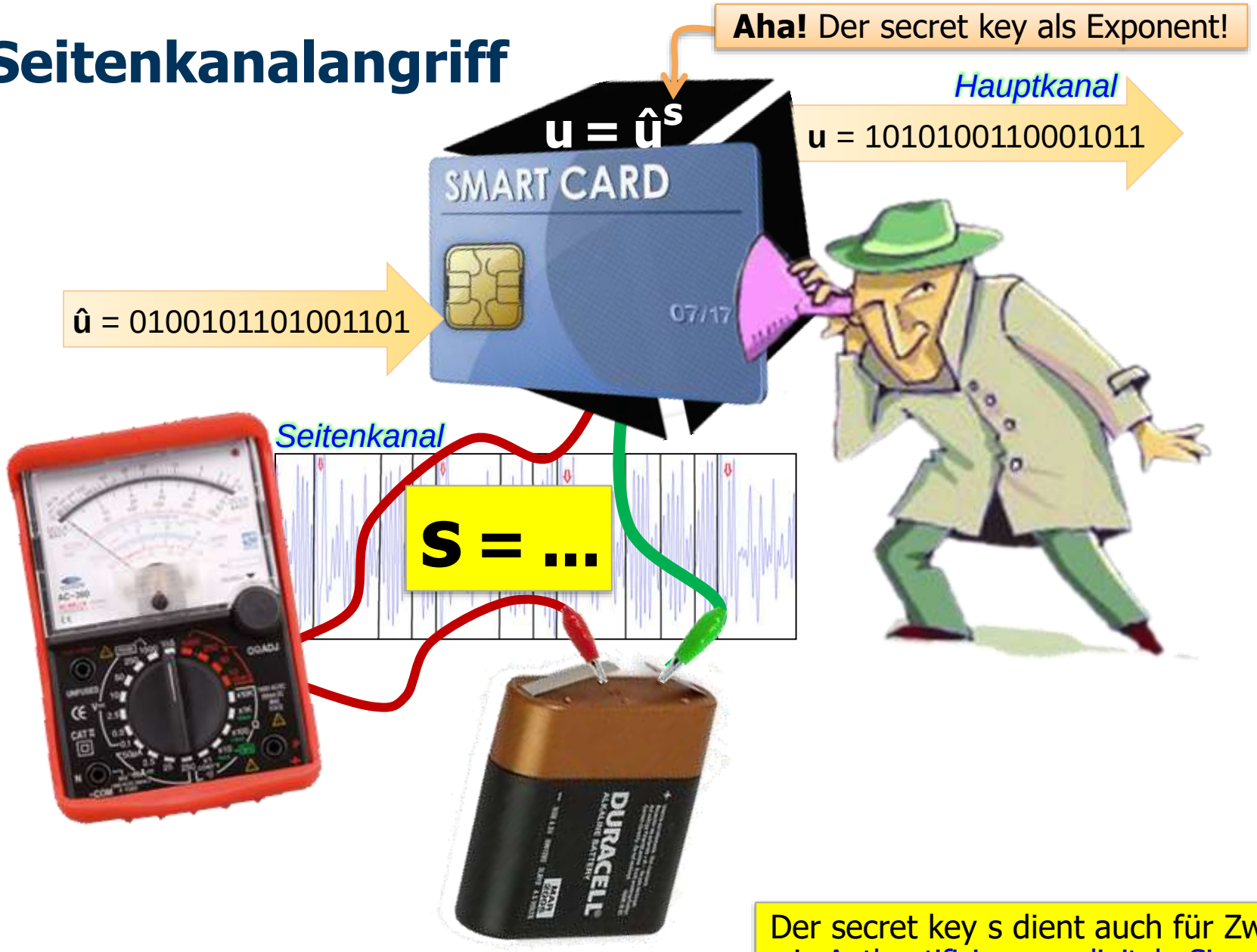


- Entschlüsseln von û durch T analog: $u = \hat{u}^s \bmod m$

Mod. Exponentiation!

Aha! Der secret key als Exponent!

Seitenkanalangriff



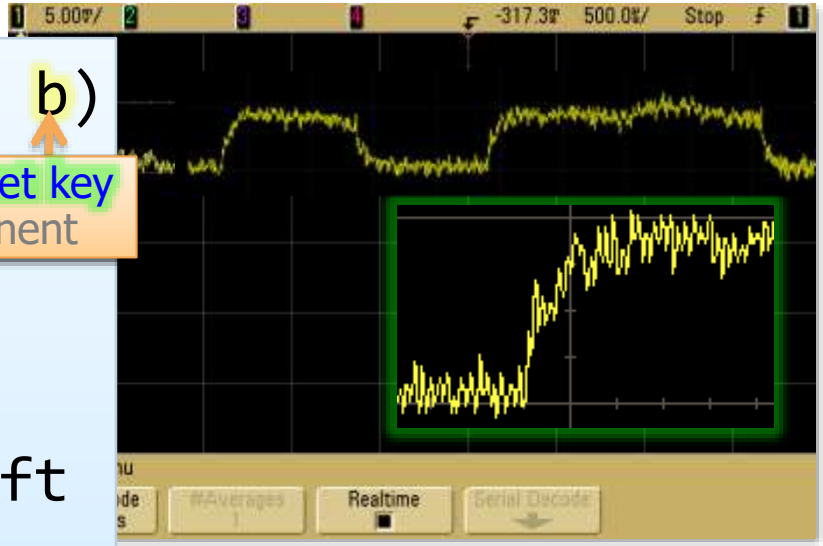
Der secret key s dient auch für Zwecke wie Authentifizierung, digitale Signatur,...

Seitenkanalangriff auf Chipkarten bei a^b

Unser Programm für a^b

```
static int f(int a, int b)
{ int z = 1;
  while (b > 0)
  { if ((b & 1) == 1)
    // ist re. Bit 1?
M       z = z * a;
    b = b / 2; // r-shift
S   a = a * a;
  }
  return z; // = a^b
}
```

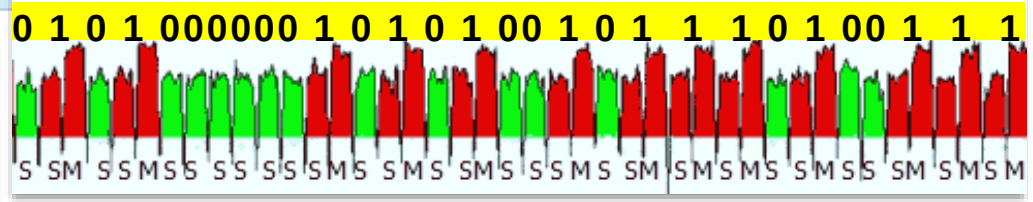
Der secret key als Exponent



http://en.wikipedia.org/wiki/Side-channel_attack

Wir messen den elektr. Strom über die Zeit (Oszilloskop)

S = Square (grün), SM = zusätzlich Multiply (rot)

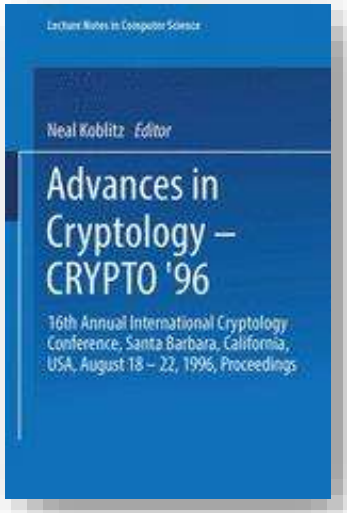


http://m.eet.com/media/1109840/Fig1.jpg

L'attaque par la simple mesure de la consommation était bien connue avant sa publication par Paul Kocher. […] Nous avions effectué deux campagnes de mesure, l'une en 1989, l'autre en 1992. Nos directeurs étaient tellement effrayés qu'ils détruisirent les documents en nous interdisant formellement d'en parler. [J-J Quisquater: Comment la crypto fut introduite dans la carte à puce, 2012]

Es wird nicht das Verfahren selbst angegriffen; es werden hingegen Schwachstellen einer Implementierung ausgenutzt, so dass der secret key der „black box" offenbar wird.

Paul C. Kocher: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: *Annual International Cryptology Conference* — CRYPTO '96. Springer, LNCS 1109, 1996, pp. 104-113



Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems

Paul C. Kocher

Cryptography Research, Inc.
607 Market Street, 5th Floor, San Francisco, CA 94105, USA.
E-mail: paul@cryptography.com.

Abstract. By carefully measuring the amount of time required to perform private key operations, attackers may be able to find fixed Diffie-Hellman exponents, factor RSA keys, and break other cryptosystems. Against a vulnerable system, the attack is computationally inexpensive and often requires only known ciphertext. Actual systems are potentially at risk, including cryptographic tokens, network-based cryptosystems, and other applications where attackers can make reasonably accurate timing measurements. Techniques for preventing the attack for RSA and Diffie-Hellman are presented. Some cryptosystems will need to be revised to protect against the attack, and new protocols and algorithms may need to incorporate measures to prevent timing attacks.

Keywords: timing attack, cryptanalysis, RSA, Diffie-Hellman, DSS.

1 Introduction

Cryptosystems often take slightly different amounts of time to process different inputs. Reasons include performance optimizations to bypass unnecessary operations, branching and conditional statements, RAM cache hits, processor instructions (such as multiplication and division) that run in non-fixed time, and a wide variety of other causes. Performance characteristics typically depend on both the encryption key and the input data (e.g., plaintext or ciphertext). While

Angriffe auf Chipkarten und Sicherheitshardware

Matus Nemec, Marek Sys, Petr Svenda, Dusan Klinec and Vashek Matyas: The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In: *24th ACM Conference on Computer and Communications Security* — CCS'2017, 1631-1648.

Immer wieder werden Methoden entdeckt, mit denen die Sicherheit von Chipkarten und Sicherheitshardware in Geräten (z.B. zur Verschlüsselung vertraulicher Kommunikation, bei WLAN-Routern, aber auch bei elektronische Ausweisen etc.) kompromittiert wird – wobei die betroffenen Devices evtl. seit Jahren millionenfach verbreitet sind und nicht einfach ausgetauscht werden können. Hier ein Beispiel aus dem Jahr 2017, mit einem kurzen Textauszug:

A newly discovered vulnerability in generation of RSA keys used by a software library adopted in cryptographic smartcards, security tokens and other secure hardware chips manufactured by Infineon Technologies AG allows for a practical factorization attack, in which the attacker computes the private part of an RSA key. The attack is feasible for commonly used key lengths, including 1024 and 2048 bits, and affects chips manufactured as early as 2012, that are now commonplace. The vulnerable chips are pervasive and not necessarily sold directly by Infineon Technologies AG, as the chips can be embedded inside devices of other manufacturers.

Our attack is not based on any weakness in a random bit generator or any additional side-channel information. Instead, the attack utilizes the specific structure of the primes as generated by Manufacturer's on-chip cryptographic library. We had access neither to the RSALib's source code nor to the object code (since it is stored only in the secure on-chip memory and is not extractable), and the whole analysis was performed solely using RSA keys generated and exported from the Manufacturer's cards and tokens.

Chipkarten und Sicherheit

Historische Notiz

„In ihrer Frühzeit in den 1980er Jahren galt die Chipkarte als unangreifbar und genoss absolutes Vertrauen – so sehr, dass man dem Publikum eine Menge Geld auf einem Silbertablett anbot. Anlässlich einer großen Ausstellung in Brüssel um das Jahr 1985 herum erwog ein Direktor von Philips, einige zehntausend Chipkarten an die Besucher zu verteilen mit dem Versprechen, dem ersten, der die vierstellige Geheimzahl seiner Karte finde, eine Million Dollar auszuzahlen. Da die Chance, die richtige Zahl durch Zufall zu erraten, 1 zu 10 000 beträgt, wäre dieses Ereignis praktisch mit Sicherheit eingetreten, aus rein statistischen Gründen. Mit der Qualität der Karte hätte das nichts zu tun gehabt. Das trug einer von uns (Quisquater) dem Direktor vor – und der glaubte es nicht!

Er ließ erst von seinem Vorhaben ab, als ich ihm zeigte, dass man tatsächlich den Kode der Karte knacken kann. Das gelingt mit einem Verfahren, das wir zusammen mit Louis Guillou, einem der Pioniere auf dem Gebiet der Chipkartensicherheit, gefunden hatten. Man stecke die Karte in ein geeignet präpariertes Lesegerät, gebe probeweise eine erste Ziffer ein, wenn der Chip die PIN anfordert, messe mit einem Oszilloskop die Zeit, die der Chip zur Verarbeitung dieser Information benötigt – und unterbreche genau im richtigen Moment die Stromzufuhr, bevor der Chip seinen Fehlversuchszähler um eins hochsetzt. Damals brauchte der Chip nämlich für die korrekte Ziffer weniger Rechenzeit als für eine falsche. Hatte man so durch Probieren die erste Ziffer gefunden, bestimmte man die folgenden Ziffern nacheinander auf dieselbe Weise. Die Büchse der Pandora war geöffnet.“ *[Jean-Jacques Quisquater, Jean-Louis Desvignes: Wie sicher ist die Chipkarte? Spektrum der Wissenschaft, März 2017, 56-64.]*

https://i.skyrock.net/0395/19590395/pics/2785430140_1.jpg

Modulare Exponentiation in der Kryptographie – Ein Angriff auf den Secret Key per Radio

Stealing Keys from PCs using a Radio:
Cheap Electromagnetic Attacks on Windowed Exponentiation

(extended version)

Daniel Genkin

Lev Pachmanov
Tel Aviv University
levp@post.tau.ac.il

Itamar Pipman

Eran Tromer
Tel Aviv University
tromer@tau.ac.il

ed February 27, 2015
dated March 2, 2015

We managed to use a plain consumer-grade radio receiver to acquire the desired signal ... After appropriate tuning, all that remained was to record the radio's headphone jack output, and digitally process the signal.

...whenever the decryption routine encounters particular bit patterns in the secret key, intermediate values occur with a special structure that causes observable fluctuations in the electromagnetic field. Through suitable signal processing and cryptanalysis, the bit patterns and eventually the whole secret key are recovered.

We demonstrate the attacks' feasibility by extracting keys from GnuPG, in a few seconds, using a nonintrusive measurement of electromagnetic emanations from laptop computers.

Die Gedanken sind „frei“...

Die Teilnehmer an Haynes Experimenten bekamen zwei Zahlen gezeigt, und jeder sollte entscheiden, ob er diese addieren oder voneinander abziehen wollte. Eine Computersoftware analysierte die Aktivitätsmuster, die während des Entscheidungsprozesses in einer Region des Stirnhirns, dem präfrontalem Kortex, auftraten. Je nach geplanter Rechenoperation sahen diese etwas unterschiedlich aus. Hatte der Computer einmal gelernt, welches Muster mit welcher Entscheidung einherging, konnte er mit einer Genauigkeit von 70 Prozent erkennen, ob die Versuchsperson eine Addition oder eine Subtraktion plante.

Bild der Wissenschaft, „Einblick in unsere Gedanken“, 28. 4. 2015

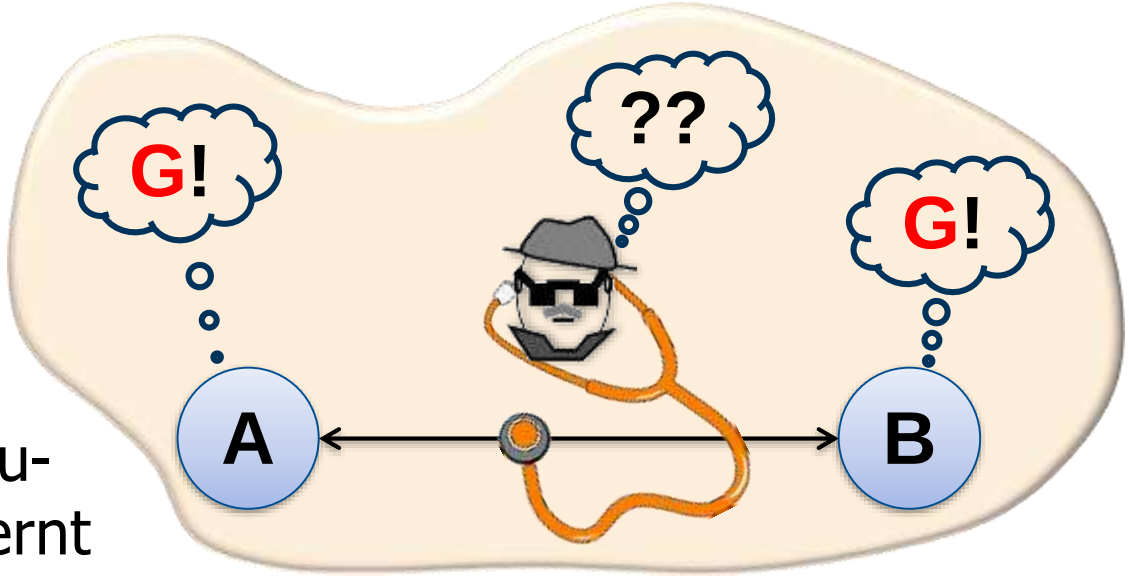
Modulare Exponentiation in der Kryptographie

- RSA wird z.B. im Open-Source-Toolkit „OpenSSL“ benutzt, das die Kommunikation wichtiger Internet-Komponenten (Bowser, Web-Server, TCP/IP-Server) absichert.
  - RSA-Schlüssel sollten mindestens 2000, besser 3000 Bit lang sein
- Digitale Unterschriften und Authentizitätsnachweise („ist die Chip-karte wirklich echt?“) werden bei RSA ebenfalls mit der modularen Exponentiation realisiert.
- Neben RSA beruhen auch andere Kryptoverfahren (z.B. ElGamal, Fiat-Shamir) wesentlich auf der modularen Exponentiation.
- Das Verfahren von Diffie-Hellman, um über einen offenen Kanal ein gemeinsames Geheimnis (z.B. Geheimcode) zu etablieren, nutzt ebenfalls die modulare Exponentiation. ➔

- Nützliche Eigenschaft: $f(x) = c^x \bmod p$ ist effizient berechenbar, die bei-den Umkehrfunktionen (diskreter Logarithmus / k-te Wurzel) jedoch nicht

Ein gemeinsames Geheimnis in einer offenen Welt?

Ziel: A und B sollen sich über einen unsicheren Kanal auf ein gemeinsames "Geheimnis" G (z.B. ein Passwort) einigen, ohne dass ein mitlauschender Angreifer es lernt



It might seem intuitively obvious that if two people have never had the opportunity to prearrange an encryption method, then they will be unable to communicate securely over an insecure channel. While this might seem intuitively obvious, I believe it is false. I believe that it is possible for two people to communicate securely without having made any prior arrangements that are not completely public.

Aus einem (seinerzeit abgelehnten) Projektvorschlag von Ralph Merkle im Rahmen des Computer-Security-Kurses in Berkeley 1974

Ein Gleichnis

- Geheimnis ausdenken; in Kiste packen; Vorhängeschloss in Zürich kaufen, Kiste damit verschliessen und absenden:

Gotthard-räuber

Aber geht das auch softwaretechnisch?

Lugano

- Kollege in Lugano kauft dort ein Schloss; Kiste doppelt verschlossen retour nach ZH

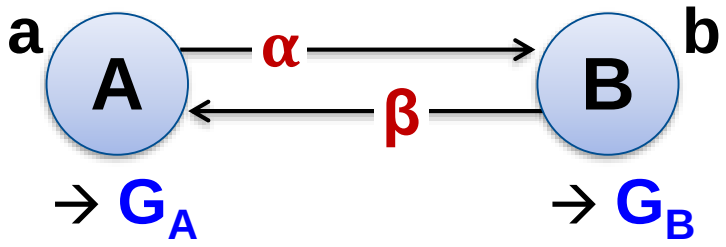
ZH

- In Zürich eigenes Schloss entfernen; Kiste nur mit Luganer Schloss erneut nach Lugano, dort öffnen und Geheimnis lesen!

Hinkt das Gleichnis?

- Könnten die Gotthardräuber nicht meine Kiste abfangen, ein eigenes Schloss dranmachen und mir zurückschicken?
- Ich in Zürich wäre dann der Meinung, die doppelt verschlossene Kiste stamme aus Lugano.
- Am Ende hätte ich dann ein gemeinsames Geheimnis mit den Gotthardräubern statt mit meinem Luganeser Kollegen!
- Symmetrisch dazu könnten die Gotthardräuber ein (anderes) gemeinsames Geheimnis mit dem Luganeser Kollegen vereinbaren.
- Sie könnten dann jede Folgenachricht entschlüsseln und mit dem anderen Schlüssel (= Geheimnis) neu verschlüsselt weiterleiten.
- Ja: Das wäre ein *aktiver* Angriff (anstelle des reinen Lesens der Nachricht als *passive* Angriffsform) eines sogenannten „Man in the Middle".
- Um solche Mittelsleute in der Leitung zu erkennen, gibt es allerdings Gegenmassnahmen, siehe z.B. https://en.wikipedia.org/wiki/Interlock_protocol
- Das, was bei Nachrichten als Bitfolgen einfach machbar ist, wäre mit einer materiellen Kiste kaum durchführbar – hier hinkt das Gleichnis leider!

Der Diffie-Hellman-Algorithmus



1. A wählt eine Zufallszahl a
2. A berechnet $\alpha = f(a)$ und sendet α an B

1′. B wählt eine Zufallszahl b
2′. B berechnet $\beta = f(b)$ und sendet β an A

Nach Empfang der Nachrichten mit α bzw. β:

3. A berechnet $G_A = \beta^a \bmod p$

3′. B berechnet $G_B = \alpha^b \bmod p$

Im folgenden: Nutzung von

$f(x) = c^x \bmod p$

Mit festen c, p ($1 < c < p$), wobei p eine grosse Primzahl ist (und weitere Eigenschaften haben soll, auf die wir hier nicht eingehen).

Bem.: a und b ($> 1, < p-1$) sind nur lokal bekannt und bleiben geheim, sie fungieren jeweils als secret key; α und β spielen die Rolle eines jeweiligen public keys.

Behauptung: $G_A = G_B$
(gemeinsames Geheimnis!)

$G_A \stackrel{!}{=} G_B$ → Das gemeinsame Geheimnis

Zu zeigen: (G_A =) $\beta^a \bmod p = \alpha^b \bmod p$ (= G_B)

$f(x) = c^x \bmod p$

- Also: $(c^b \bmod p)^a \bmod p \stackrel{?}{=} (c^a \bmod p)^b \bmod p$
- Vereinfachung: „mod p" überall weglassen... (Wieso darf man das?)
- Also zu zeigen: $(c^b)^a = (c^a)^b$
- Aufgrund der Rechenregeln für Potenzen gilt aber: $(c^b)^a = c^{ba} = c^{ab} = (c^a)^b$ □

Was fängt man mit einem gemeinsamen Geheimnis **G** an? Man kann es z.B. als Schlüssel für effiziente symmetrische Verschlüsselungsverfahren nutzen.

- Einzusehen bliebe noch, dass G aus Kenntnis von α, β, c und p von einem Mitlauscher (in effizienter Weise) nicht ermittelt werden kann – die Sicherheit beruht i.W. darauf, dass diskrete Logarithmen nicht effizient berechnet werden können.
- Obwohl das Verfahren so einfach erscheint, kann man bei der Implementierung viel falsch machen, so dass Angriffe möglich werden. Interessierte mögen studieren: *Raymond, J. F. and Stiglic, A. (2000). Security issues in the Diffie-Hellman key agreement protocol. IEEE Transactions on Information Theory, 22, 1-17.*

Ein Beispiel zur Diffie-Hellman-Methode

Modulare Arithmetik

In Konstanz gibt es einen Studenten, der sich ein T-Shirt angefertigt hat, das vorne so beschriftet ist: **3 x 5 = 1**. *Kommt er auf einen zu, so möchte man in Erregung geraten ob dieser Provokation. Aber wie beim Doppler-Effekt beruhigt sich die Pulsfrequenz, sobald der junge Mann vorübergeht, denn auf der Rückseite erkennt man die Weisheit des Hemdes:* **in Z_7**.

-- Volker Strassen

Das Beispiel [Wikipedia] dient zur Veranschaulichung und benutzt deshalb sehr kleine Zahlen. In der Praxis werden dagegen Zahlen mit hunderten von Stellen verwendet.

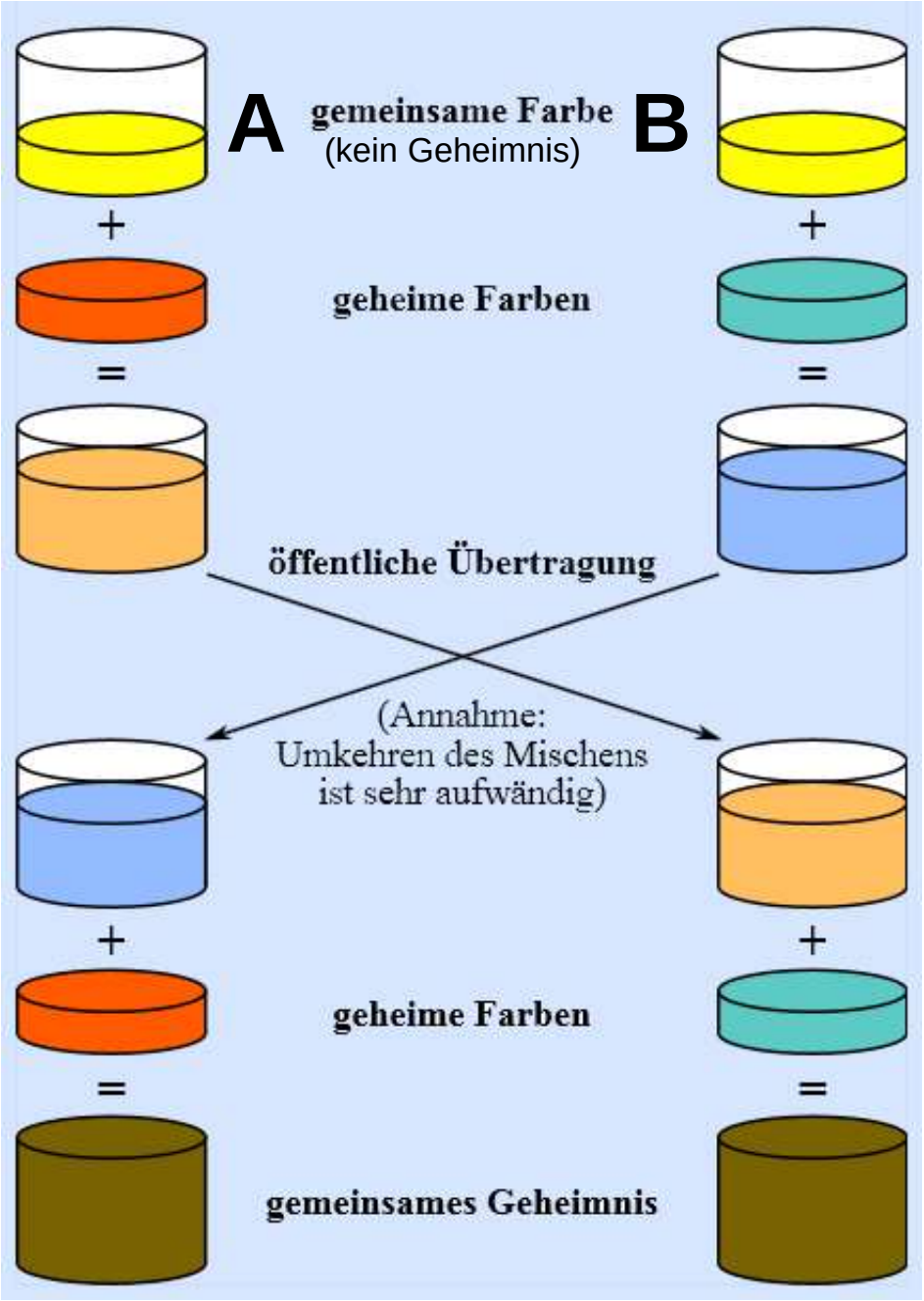
1. Die beiden Systemparameter p und c seien mit $p = 13$ und $c = 2$ fest vorgegeben.
2. Alice wählt die geheime Zufallszahl $a = 5$ und Bob $b = 8$.
3. Nun berechnet Alice $\alpha = c^a \bmod p = 2^5 \bmod 13 = 6$ und sendet α an Bob. Bob berechnet $\beta = c^b \bmod p = 2^8 \bmod 13 = 9$ und sendet β an Alice.
4. Alice berechnet $G_A = \beta^a \bmod p = 9^5 \bmod 13 = 3$. Bob berechnet $G_B = \alpha^b \bmod p = 6^8 \bmod 13 = 3$.
5. Beide erhalten das gleiche Ergebnis $G = G_A = G_B = 3$.

Eine Lauscherin Eve kann zwar die Zahlen 13, 2, 6 und 9 in Erfahrung bringen bzw. mithören, das eigentliche gemeinsame Geheimnis G = 3 von Alice und Bob bleibt ihr aber verborgen. G = 3 kann als Schlüssel für die nachfolgende Kommunikation verwendet werden.

Mit Hilfe abgefangener Nachrichten könnte Eve immerhin die zwei Gleichungen $6 = 2^a \bmod 13$ sowie $9 = 2^b \bmod 13$ aufstellen. Daraus kann sie beispielsweise durch Ausprobieren die beiden geheimen Zahlen a = 5 und b = 8 bestimmen. Den vereinbarten Schlüssel G von Alice und Bob kann sie dann mit $G = c^{ab} \bmod p$ ausrechnen. Wenn jedoch die Primzahl p gross genug gewählt wird und c ein Generator der Gruppe Z_p ist (im obigen Beispiel ist 2 ein Generator der Gruppe Z_{13}), ist es für Eve zu aufwändig, alle Zahlen zwischen 1 und $p - 1$ durchzuprobieren, die als Resultat der modularen Potenz $c^a \bmod p$ in Frage kommen.

Noch ein Gleichnis zur Diffie-Hellman-Methode

- Hier in Form von Farbenmischen
- Ziel: A und B sollen sich über einen unsicheren Kanal auf ein gemeinsames "Geheimnis" (hier: eine Farbe) einigen, ohne dass ein Angreifer es erfährt



Bildquelle: Wikipedia

Historische Notiz

Der 1976 entdeckte Algorithmus wurde 1977 patentiert

United States Patent [19]

Hellman et al.

[11] 4,200,770

[45] Apr. 29, 1980

[54] **CRYPTOGRAPHIC APPARATUS AND METHOD**

[75] Inventors: **Martin E. Hellman**, Stanford; **Bailey W. Diffie**, Berkeley; **Ralph C. Merkle**, Palo Alto, all of Calif.

[73] Assignee: **Stanford University**, Palo Alto, Calif.

[21] Appl. No.: **830,754**

[22] Filed: **Sep. 6, 1977**

[51] **Int. Cl.**[2] **H04L 9/04**

[52] **U.S. Cl.** ..... **178/22**; 340/149 R; 375/2; 455/26

[58] **Field of Search** ..... 178/22; 340/149 R

[56] **References Cited**

PUBLICATIONS

"New Directions in Cryptography", Diffie et al., *IEEE Transactions on Information Theory*, vol. IT-22, No. 6, Nov. 1976.

Diffie & Hellman, Multi-User Cryptographic Techniques", *AFIPS Conference Proceedings*, vol. 45, pp. 109–112, Jun. 8, 1976.

Primary Examiner—Howard A. Birmiel

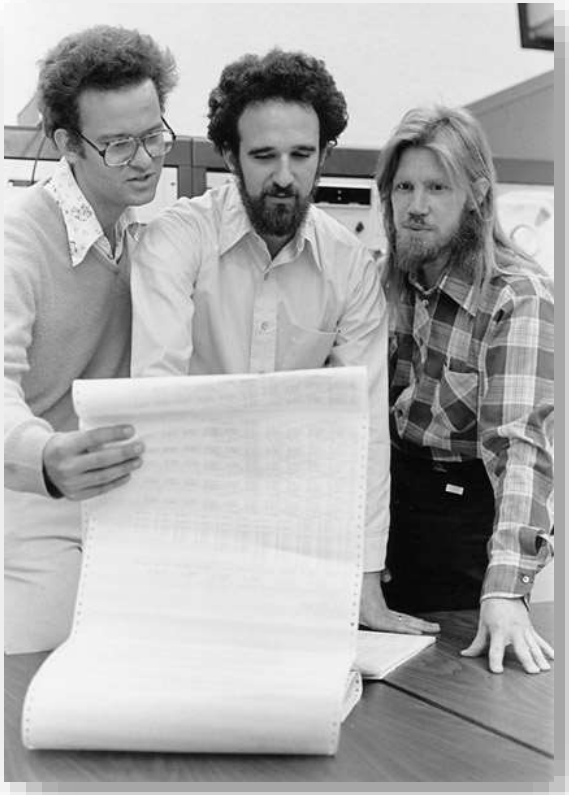
Attorney, Agent, or Firm—Flehr, Hohbach, Test

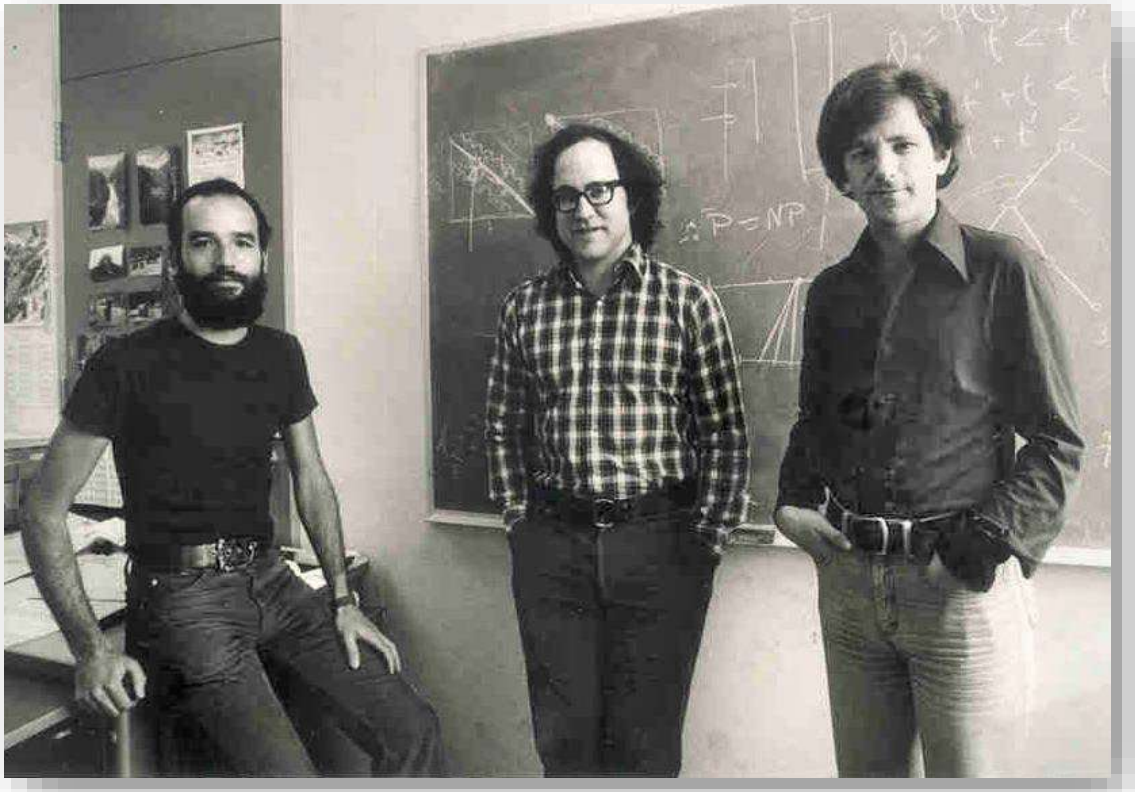
[57] **ABSTRACT**

A cryptographic system transmits a computationally secure cryptogram over an insecure communication channel without prearrangement of a cipher key. A secure cipher key is generated by the conversers from transformations of exchanged transformed signals. The conversers each possess a secret signal and exchange an initial transformation of the secret signal with the other converser. The received transformation of the other converser's secret signal is again transformed with the receiving converser's secret signal to generate a secure cipher key. The transformations use non-secret operations that are easily performed but extremely difficult to invert. It is infeasible for an eavesdropper to invert the initial transformation to obtain either conversers' secret signal, or duplicate the latter transformation to obtain the secure cipher key.

8 Claims, 6 Drawing Figures

Merkle, Hellman, Diffie; Shamir, Rivest, Adleman (1977)





Diffie und Hellman erhielten 2015 den Turing Award; Shamir, Rivest und Adleman bereits 2002

Die frühere Entdeckung der Public-Key-Kryptographie durch den britischen Geheimdienst

Von Boris Gröndahl, Telepolis, 20. Jan. 1998 (Auszug)

Auf einer schmucklosen Textseite präsentierte die britische Regierungsbehörde Communications-Electronics Security Group (CESG) [eine Unterorganisation des GCHQ] Mitte Dezember 1997 bescheiden einen Bericht ihres pensionierten [und kurz zuvor verstorbenen] Mitarbeiters James Ellis.

Der Inhalt des Papiers ist eine wissenschaftsgeschichtliche Sensation: Ellis erzählt darin, wie er 1970 die Public-Key-Kryptographie (PKC) entdeckte. Diese heutzutage praktisch allen wichtigen Verschlüsselungstechniken zugrundeliegende Theorie war bisher den Mathematikern Whitfield Diffie, Martin Hellman und Ralph Merkle zugeschrieben worden, die sie 1976 veröffentlichten. Auch der wichtigste Algorithmus der PKC, mit dem Ron Rivest, Adi Shamir, und Leonard Adleman 1977 an die Öffentlichkeit traten, und der heute unter dem Kürzel RSA ein Begriff ist, wurde laut Ellis von der britischen Krypto-Agentur Jahre vorher entwickelt.

Ellis schrieb den Bericht nach Aussagen der CESG bereits 1987. Seine Veröffentlichung hat er nicht mehr erlebt: Er starb am 25. November 1997. [...] In Ellis' Worten: „Können wir eine sicher verschlüsselte Nachricht erstellen, die der vorgesehene Empfänger lesen kann, ohne dass zuvor ein geheimer Austausch des Schlüssels stattfinden muss? Diese Frage fiel mir eines Nachts im Bett ein, und der Beweis der theoretischen Möglichkeit war eine Sache weniger Minuten." Ellis publizierte seinen nächtlichen Geistesblitz 1970 in dem internen Zirkular der CESG. Clifford Cocks, ein Kollege von Ellis, veröffentlichte die erste Implementation ebenfalls CESG-intern 1973: Cocks fand, wie Ellis darlegt, einen Spezialfall des RSA-Algorithmus. Und Malcolm Williamson (1950 – 2015), ein weiterer CESG-Agent, fand das später als Diffie-Hellman-Algorithmus bekannt gewordene Verfahren ebenfalls lange bevor es 1976 öffentlich vorgestellt wurde.

Keine Veröffentlichung bedeutet für die britischen Krypto-Forscher nicht nur kein Ruhm, sondern auch keine Ansprüche auf das Patent. Die CESG erklärte, sie habe die Patentierung seinerzeit prüfen lassen, doch die Patentierung mathematischer Formeln sei [...] nicht möglich gewesen.

James Ellis: **The Story of Non-Secret Encryption**
(1987; öffentlich gemacht durch CESG / GCHQ am 16. Dez. 1997; Auszüge)

[…] It was obvious to everyone, including me, that no secure communication was possible without secret key, some other secret knowledge, or at least some way in which the recipient was in a different position from an interceptor. After all, if they were in identical situations how could one possibly be able to receive what the other could not? Thus there was no incentive to look for something so clearly impossible.

The event which changed this view […] The reason was not far to seek. The difference between this and conventional encryption is that in this case the recipient takes part in the encryption process. Without this the original concept is still true. So the idea was born. Secure communication was, at least, theoretically possible if the recipient took part in the encipherment. […]

The proof of the theoretical possibility took only a few minutes. We had an existence theorem. The unthinkable was actually possible. The only remaining question was "Can it be made practicable?" […] Because of the weakness of my number theory, practical implementations were left to others. The first workable idea was put forward […] by Clifford Cocks. This is essentially the RSA Algorithm.

[...] reaffirming that the credit belongs to Malcolm Williamson. […] The method was published […] by Diffie and Hellman. This was identical to Williamson's version […]. This was the start of public awareness of this type of cryptography and subsequent rediscovery of the Non-Secret Encryption techniques I have described.

Copy No. 33

COMMUNICATIONS-ELECTRONICS SECURITY GROUP

C
E
S
G

Research Report No. 3006

THE POSSIBILITY OF SECURE NON-SECRET DIGITAL ENCRYPTION

SECRET

SECRET

C.E.S.G. REPORT NO.3006

THE POSSIBILITY OF SECURE NON-SECRET DIGITAL ENCRYPTION

J. H. Ellis

Summary

This report considers the problem of achieving secure transmission of digital information in the circumstances where there is no information initially possessed in common by the two legitimate communicators which is not also known to the interceptor. It demonstrates, by means of a model having the required properties that a theoretical solution exists, but does not establish that a practical system can be devised.

Case No. 305 refers

Date of approval for issue:
January 1970

SECRET

SECRET

Introduction

1. It is generally regarded as self-evident, that, in order to prevent an interceptor from understanding a message which is intelligible to the authorised recipient, it is necessary to have some initial information known to the sender and to the recipient but kept secret from the interceptor. This information can take many forms, such as the method of encipherment itself, the construction of a cipher machine, a key setting or a one-time tape. All these methods require that there is a route by which this secret information can be sent without fear of interception. Only then can the cipher text be sent safely in a non-secret manner, and large quantities of cipher text of high security thus tend to need the parallel transmission of smaller, but still substantial quantities of secret information.

2. This report demonstrates that this secret information is not theoretically necessary and that, in principle, secure messages can be sent even though the method of encipherment and all transmissions between the authorised communicators are known to the interceptor. This is what is meant by "non-secret encryption". It must be emphasised

When Cocks first explained his work on public-key cryptography to Williamson, Williamson really didn't believe it and tried to prove that Cocks had made a mistake and that public-key cryptography did not really exist. Remarkably enough, Williamson failed to find a mistake, instead he found… [Song Y. Yan]

THOUGHTS ON CHEAPER NON-SECRET ENCRYPTION [Auszüge!]

M J Williamson, 10 August 1976 [http://cryptocellar.org/cesg/cheapnse.pdf]

This note is mainly a much-delayed response to some calculations made on the speed and cost of the equipment to implement non-secret encryption. Modifications can be made to the method I suggested in ["Non Secret Encryption Using a finite field", 21 January 1974, M J Williamson] to make it cheaper and faster. […] The method I am putting forward […] is thus: both link ends (and any interceptor) know a primitive element x of a finite field F and are working with elements of the field as polynomials modulo the primitive polynomial of x and coefficients modulo a prime p. Let there be pq elements in F, then x has period pq-1.

a. The two link ends generate random numbers a,b in the range 1 to p^q-1;
b. they calculate x^a and x^b respectively;
c. they calculate $(x^b)^a = (x^a)^b$;
d. both link ends know this x^{ab} and it is difficult for the interceptor to recover it so either link end can use it as additive key to send a message.

Es handelt sich i.W. um das Diffie-Hellmann-Prinzip

[…] Almost all the work involved in the procedure is in the calculation of y^a in the field. […] Now y^a can be built up by a sequence of squarings and multiplications by y. […] There is a type of FFT precisely suited to the convolution of two modulo p sequences where, as here, p is a Fermat prime. […] I feel that non-secret encryption could be implemented with current technology but that work still needs to be done on the theory of the method. In open literature there is a certain amount of work published on the complexity of functions […] and I regret that I do not have at present the necessary background knowledge to understand ["Computational Complexity over Finite Fields", Volker Strassen, Siam J. Comput. 5(2), June 1976, 324-331].

Volker Strassen war von 1968 bis 1988 Prof. für Mathematik an der Universität Zürich, vgl. nächste Slide.

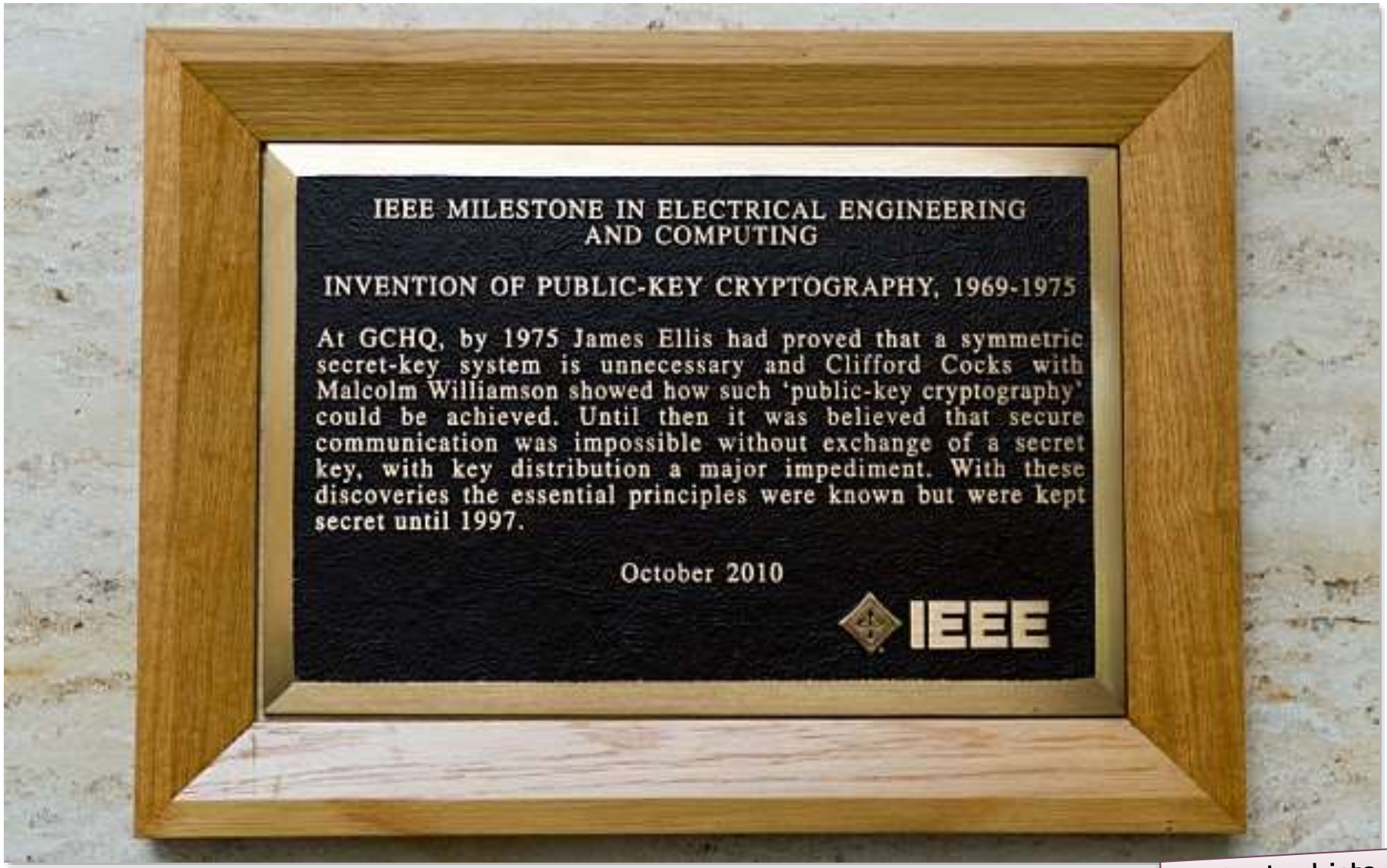
Zum oben erwähnten Volker Strassen einige Informationen:

Volker Strassen (Jg. 1936) begann sein Studium 1955 zunächst mit Musik und Philosophie in Köln, kurz darauf wechselte er zu Mathematik, Philosophie und Physik an die Universität Freiburg i. Br. Ab 1957 studierte er Physik und Mathematik an der Ludwig-Maximilians-Universität München; ab 1958 in Göttingen. Dort Studienabschluss 1961 sowie bereits 1962 Promotion mit einer Arbeit über Informationstheorie („Meßfehler und Information“). 1962 bis 1968 war er mit kurzer Unterbrechung in Berkeley; 1968 wurde er an die Universität Zürich berufen. Von 1988 bis zu seiner Emeritierung 1998 war er schliesslich Professor in Konstanz.

Strassen ist einer der Wegbereiter der Komplexitätstheorie. Der berühmt gewordene *Strassen-Algorithmus* (1969 veröffentlicht unter dem Titel „Gaussian Elimination is not optimal“) realisiert die Matrizenmultiplikation (asymptotisch) effizienter als das Standardverfahren, welches kubischen Aufwand verursacht. Er entwickelte auch Verfahren zur schnellen Multiplikation grosser Zahlen sowie effiziente probabilistische Primzahltests. Seine Arbeiten aus dem Bereich der algorithmischen Zahlentheorie spielen bei der modernen Kryptographie eine wichtige Rolle. Daneben war Volker Strassen auch in mehreren anderen Teilgebieten der Mathematik in kreativer Weise tätig.

Volker Strassen 1979 am mathematischen Forschungsinstitut Oberwolfach bei einer Tagung zu Komplexitätstheorie.

Die James Ellis gewidmete Gedenktafel beim GCHQ (“Government Communications Headquarters”)



http://i.telegraph.co.uk/multimedia/archive/03593/james_ellis4_3593310b.jpg

Ende der historischen Notiz

How the NSA Can Break Trillions of Encrypted Web and VPN Connections
Ars Technica (Oct 15, 2015), Dan Goodin

Researchers at the universities of Michigan and Pennsylvania warn a serious flaw in the way the Diffie-Hellman cryptographic key exchange is implemented is allowing the U.S. National Security Agency (NSA) to break and eavesdrop on trillions of encrypted connections. For commonly used 1024-bit keys, it would take about a year and millions of dollars to crack just one of the extremely large prime numbers that form the starting point of a Diffie-Hellman negotiation. However, the researchers found only a few primes are commonly used, putting the price well within NSA's $11-billion annual budget for "groundbreaking cryptanalytic capabilities." "Breaking a single, common 1024-bit prime would allow NSA to passively decrypt connections to two-thirds of VPNs and a quarter of all SSH servers globally. Breaking a second 1024-bit prime would allow passive eavesdropping on connections to nearly 20% of the top million HTTPS websites. In other words, a one-time investment in massive computation would make it possible to eavesdrop on trillions of encrypted connections," say researchers J. Alex Halderman and Nadia Heninger. "While the documents make it clear that NSA uses other attack techniques, like software and hardware 'implants' to break crypto on specific targets, these don't explain the ability to passively eavesdrop on virtual private network traffic at a large scale." In addition, they say the technique makes it possible for other countries, including adversaries of the U.S., to decrypt communications on a massive scale.

http://arstechnica.co.uk/security/2015/10/how-the-nsa-can-break-trillions-of-encrypted-web-and-vpn-connections/

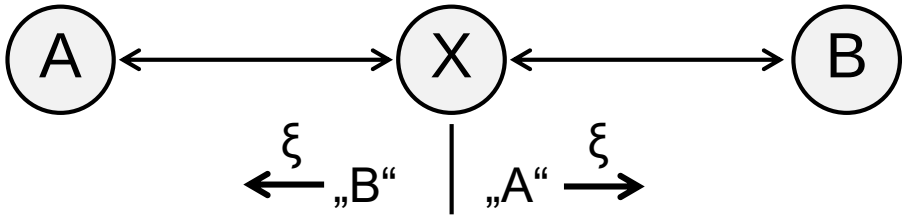
Kryptographie und Informationssicherheit

„Sicherheit“ ist ein grosses Gebiet, wir haben nur wenige Aspekte gestreift – folgendes wäre u.a. (!) noch interessant:

- Symmetrische Verschlüsselungsverfahren
  - Hierbei wird zum Ver- und Entschlüsseln der gleiche Schlüssel benutzt
  - Beispiele: AES, IDEA, DES
  - Vorteil gegenüber asymmetrischen Verfahren wie RSA: Viel schneller
  - Nachteil: (Verschlüsselungs)schlüssel muss geheim gehalten werden
- Verschlüsseln mit One-Time-Pads
  - „Perfektes“ symmetrisches Kryptosystem (nicht zu brechen!)
  - Wähle echt zufällige Sequenz von Schlüsselbits als Schlüssel
  - *Verschlüsselung:* Schlüsseltext = Klartext XOR Schlüsselbitsequenz
  - *Entschlüsselung:* Klartext = Schlüsseltext XOR Schlüsselbitsequenz
  - Begründung: (a XOR b) XOR b = a (für alle Bitbelegungen von a, b)
  - Nachteil: Länge des Schlüssels

Kryptographie und Informationssicherheit (2)

- Man-In-The-Middle-Angriffe
  - X imitiert die Identität des jeweils anderen – z.B. ein Problem beim Diffie-Hellman-Verfahren!



- Einwegfunktionen
  - y = f(x) einfach aus x berechenbar, aber x = $f^{-1}(y)$ ist extrem schwierig (d.h. ineffizient) aus y zu ermitteln
  - Bilden Grundlage vieler Kryptoverfahren
  - Beispiel: $f(x) = c^x \bmod p$
- Digitale Signatur
- Kryptographisch gesicherte Zertifikate

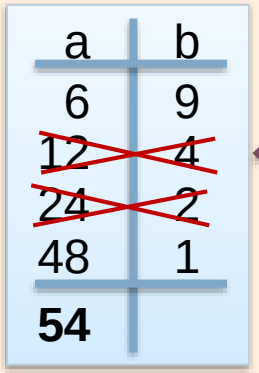
Übrigens: RSA, Diffie-Hellman und alle anderen asymmetrischen Verfahren werden unsicher, falls es zu erheblichen Fortschritten bei der Realisierung von Quantencomputern käme!

Auf all' das können wir in *dieser* Vorlesung aber leider nicht eingehen!

Resümee des Kapitels

[Aus der kurzen Wiederholung der wesentlichen Themen der Vorwoche jeweils zu Beginn einer Lektion]

- Algorithmus „altägyptische Multiplikation“
  - Verdoppeln und Halbieren ohne Rest
- Rekursion: Reduktion auf eine einfachere Instanz des gleichen Problems
  - Algorithmus als rekursives Java-Programm
- Korrektheitsnachweis $\forall\, a,b \in \mathbb{N}^+$: **f(a,b) = a × b** mit vollst. Induktion (über b)
- Fehlerhaftes Programmverhalten
  - Z.B. Stack-Überlauf bei Eingabe b = 0
- Multiplikationsalgorithmus iterativ (d.h., mit while-Schleife)
- Formale Verifikation der Programmkorrektheit mit math. Kalkül
  - Zusicherungen (insbesondere Schleifeninvarianten) zum Programmzustand



Mathematische Formalisierung

$$f(a,b) = \begin{cases} a & \text{, falls } b = 1 \\ f(2a, b/2) & \text{, falls } b \text{ gerade} \\ a + f\left(2a, \frac{b-1}{2}\right) & \text{sonst} \end{cases}$$

Resümee des Kapitels (2)

- Logikkalkül zur Programmsemantik
  - Korrektheitsbeweis mit dem Code verwoben
  - Verifikation durch Proof-Checker

```
// z+a = φ
  z = z + a;
// z = φ
```

- Effizienz des altägypt. Multiplikationsalgorithmus a × b
  - Zählen von Elementaroperationen → ~ 5 $\log_2$ b
  - Vergleich mit der Schulmethode (Nutzung des kleinen Einmaleins)
- „Elementare" Berechnungsoperationen
  - *gerade*, *halbiere*, *verdopple*,... rekursiv implementierbar
- Funktionales Programmieren
  - Keine Variablen und Zuweisungen; keine Schleifen
- Kryptographie
  - Modulare Exponentiation, RSA-Verschlüsselungsmethode
  - Diffie-Hellman-Methode für gemeinsames Geheimnis
  - Seitenkanalangriffe bei Chipkarten