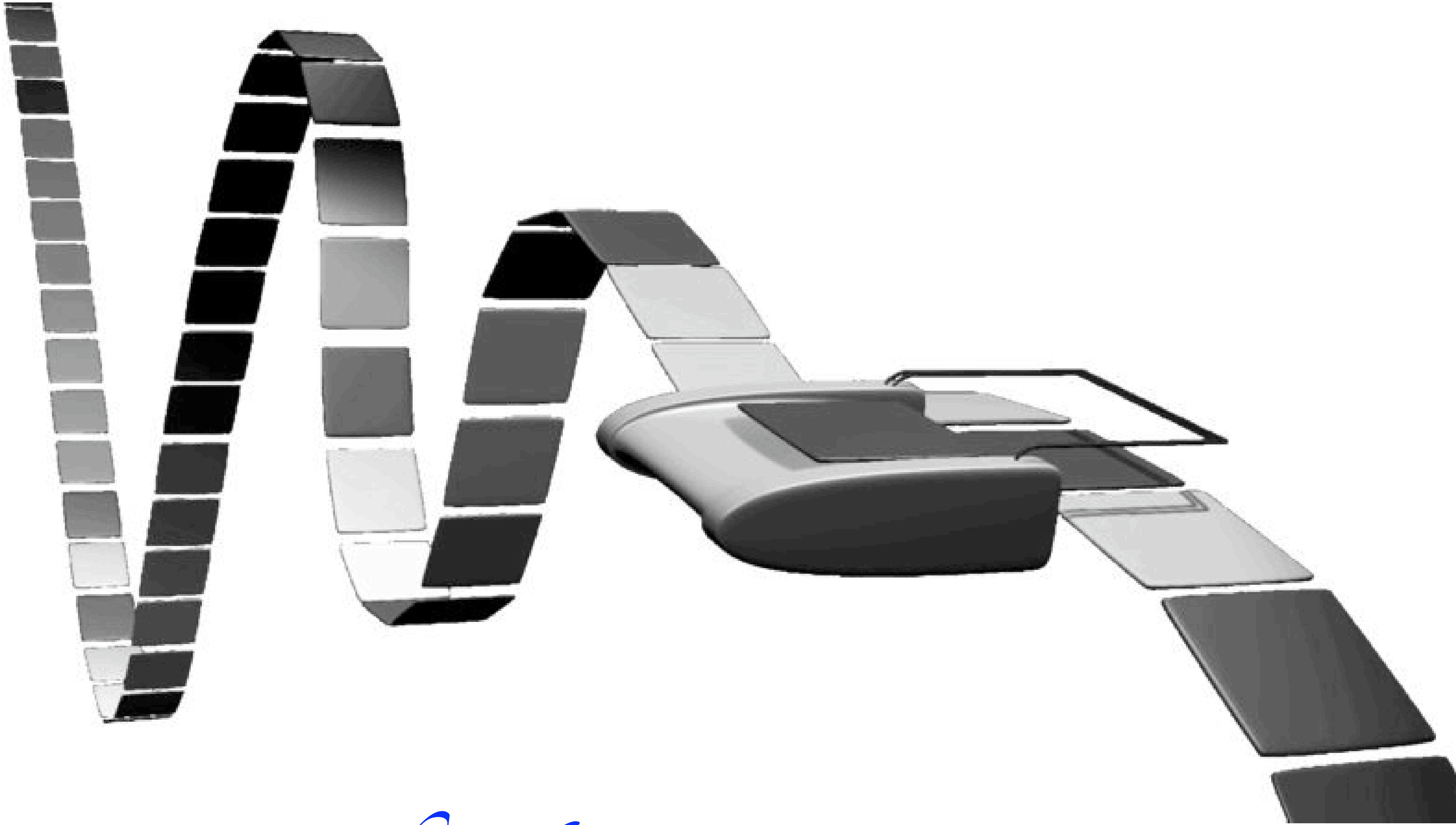
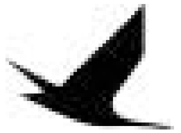


Gottfried's Dream

Redeneren en Rekenen

Henk Barendregt
Jan Willem Klop

Aristoteles Leibniz Boole Frege Russell Gödel
Church Turing de Bruijn en vele anderen



Symposium 'In vogelvlucht door de Logica'
24 april 2007
Radboud Universiteit Nijmegen

Symposium 'In vogelvlucht door de Logica'

Abstract lezing Henk Barendregt en Jan Willem Klop
Titel: Analyse van rekenen en redeneren

De reden dat duizenden jaren geleden de getallen waren uitgevonden en ook de meetkunde was de relevantie om bepaalde zaken te berekenen, zoals de oppervlakte van een bepaald stuk land. De Egyptenaren en later nog beter de Babyloniers waren redelijk goed in dat rekenen. Pas in de Griekse wiskunde is men gaan bewijzen. Nadat dit tot nieuwe inzichten en rekenmethoden had geleid kwam de vraag op of we het redeneren geheel konden vangen. Aristoteles heeft zich hiermee beziggehouden en 2300 jaar later heeft Frege diens werk voltooid, zoals in de volledigheidsstelling van Goedel (1930) bewezen wordt.

Op het moment dat vastlag wat de reikwijdte van het bewijzen was, kon een essentiële beperking ervan aangetoond worden. Zo liet Goedel in zijn onvolledigheidssteling (1931) zien dat een theorie die de elementaire rekenkunde bevat en geen inconsistenties kan afleiden (dwz een uitspraak kan nooit zowel bewezen als weerlegd worden) altijd onvolledig is: er is voor zo'n theorie een uitspraak die noch bewezen noch weerlegd kan worden.

Een paar jaar later werd in 1936 geheel vastgelegd wat de reikwijdte van het rekenen is. Dat werd gedaan door Church via lambda calculus en Turing via een machine model van het rekenen. Ook nu weer kon de conclusie gemaakt worden dat berekeningen maken zijn beperking heeft: er is geen computer (of wat op hetzelfde neer komt geen algoritme) die alle mogelijke wiskundige problemen oplost. Dit is een sterker resultaat dan de onvolledigheidsstelling van Goedel.

In de voordracht zal niet de nadruk komen te liggen op deze negatieve resultaten, maar meer op de positieve aspecten van de reflectieve kennis over berekeningen en over bewijzen. Klop zal spreken over het rekenen en Barendregt over het bewijzen.

ENGINES OF LOGIC

Mathematicians and the Origin of the Computer

MARTIN DAVIS

"A thoroughly enjoyable mix of biographical portraits and theoretical mathematics . . . full of well-honed anecdotes and telling detail."
—*Publishers Weekly*

Science

"Beautifully written. . . . [Martin Davis] argues that computers are in many ways the culmination of the glorious and powerful mathematical tradition we now call logic. . . . Davis's writing is elegant and fluid, and the mathematical development interspersed in the story is gentle and accessible. . . . The influence of logic on the development of computers and computation is undeniable and deep—and it could not have found a more eloquent and worthy chronicler."
—Christos H. Papadimitriou, *American Scientist*

"An elegant history of the search for the boundaries of logic and the machines that live within them. Blending mathematical details with biographical tidbits, Davis explains how we've come to understand what we do about the limits of computers and logical thought. . . . [His] book is a full-contact tour through an important branch of math and logic."
—Peter Wayner, *Wired*

Martin Davis, a world pioneer in the development of computer science, is also the author of *Computability and Unsolvability*, which has been called "one of the few real classics in computer science." A professor emeritus at New York University's Courant Institute of Mathematical Sciences, he is currently a visiting scholar at the University of California in Berkeley.

Originally published under the title
The Universal Computer:
The Road from Leibniz to Turing
Cover design by Martin Ogolter

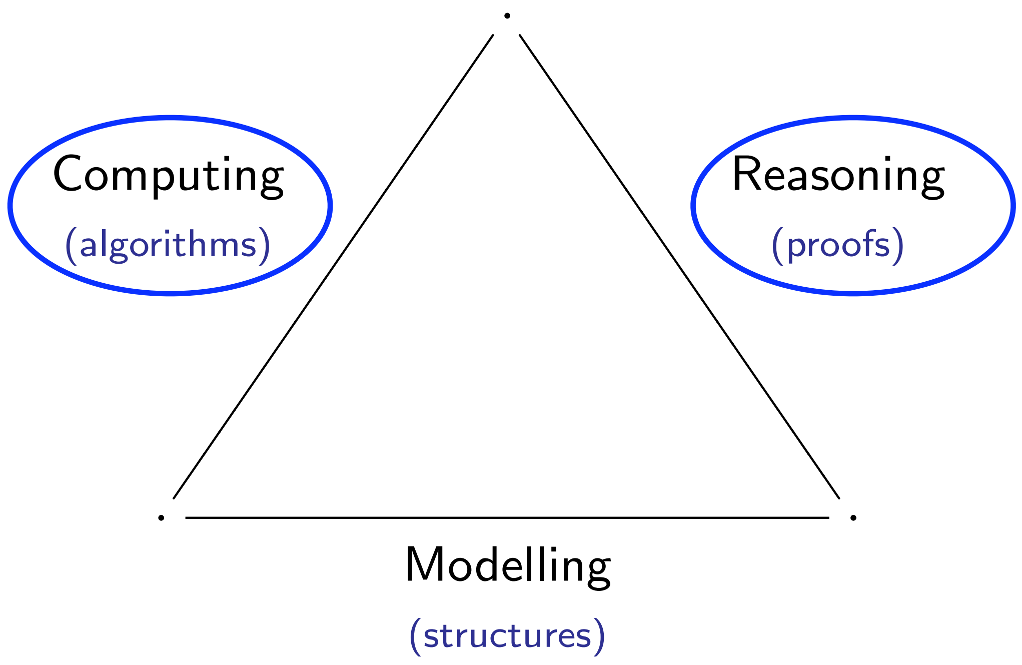
W. W. NORTON
NEW YORK • LONDON

ISBN 0-393-32229-7
51495
9 780393 322293
$14.95 USA $21.99 CAN.

www.wwnorton.com

1. Mathematics

Mathematical activity (stylised): modelling, computing, reasoning



Mathematics is usually done with *informal* rigour

refereeing playing an important role

1.1 Axioms and logic

Aristotle (384-322 BC)

De reden dat duizenden jaren geleden de getallen waren uitgevonden en ook de meetkunde was de relevantie om bepaalde zaken te berekenen, zoals de oppervlakte van een bepaald stuk land. De Egyptenaren en later nog beter de Babyloniers waren redelijk goed in dat rekenen. Pas in de Griekse wiskunde is men gaan bewijzen. Nadat dit tot nieuwe inzichten en rekenmethoden had geleid kwam de vraag op of we het redeneren geheel konden vangen. Aristoteles heeft zich hiermee beziggehouden en 2300 jaar later heeft Frege diens werk voltooid, zoals in de volledigheidsstelling van Goedel (1930) bewezen wordt.

- The axiomatic method

objects	properties
primitive	axioms
defined	derived

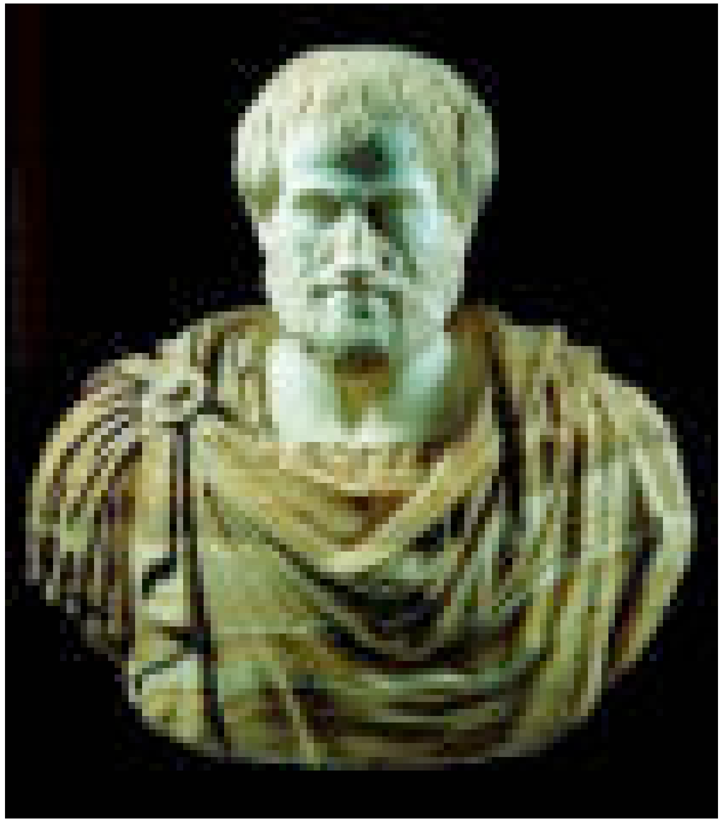
- The quest for logic: try to chart reasoning

  (finished by Frege [1879]; proved complete by Gödel [1930])

- Proof-checking vs theorem proving

’Αριστοτελησ

384 vC- 322 vC

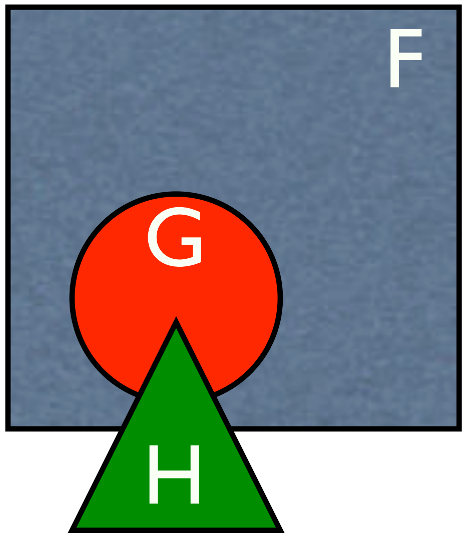


24 logische syllogismen, fragment van predikatenlogica

I. Barbara, Celarent, Darii, Ferio, Barbari, Celaront
II. Cesare, Camestres, Festino, Baroco, Cesaro, Camestros
III. Darapti, Disamis, Datisi, Felapton, Bocardo, Ferison
IV. Bamalip, Calemes, Dimatis, Fesapo, Fresison, Calemos

DIMARIS

sommige H zijn G
elke G is F
sommige F zijn H



BA

Trivium:
grammatica, rhetorica, logica

MA

Quadrivium:
rekenkunde, meetkunde, astronomie, harmonieleer

Gottfried Leibniz

1646 - 1716

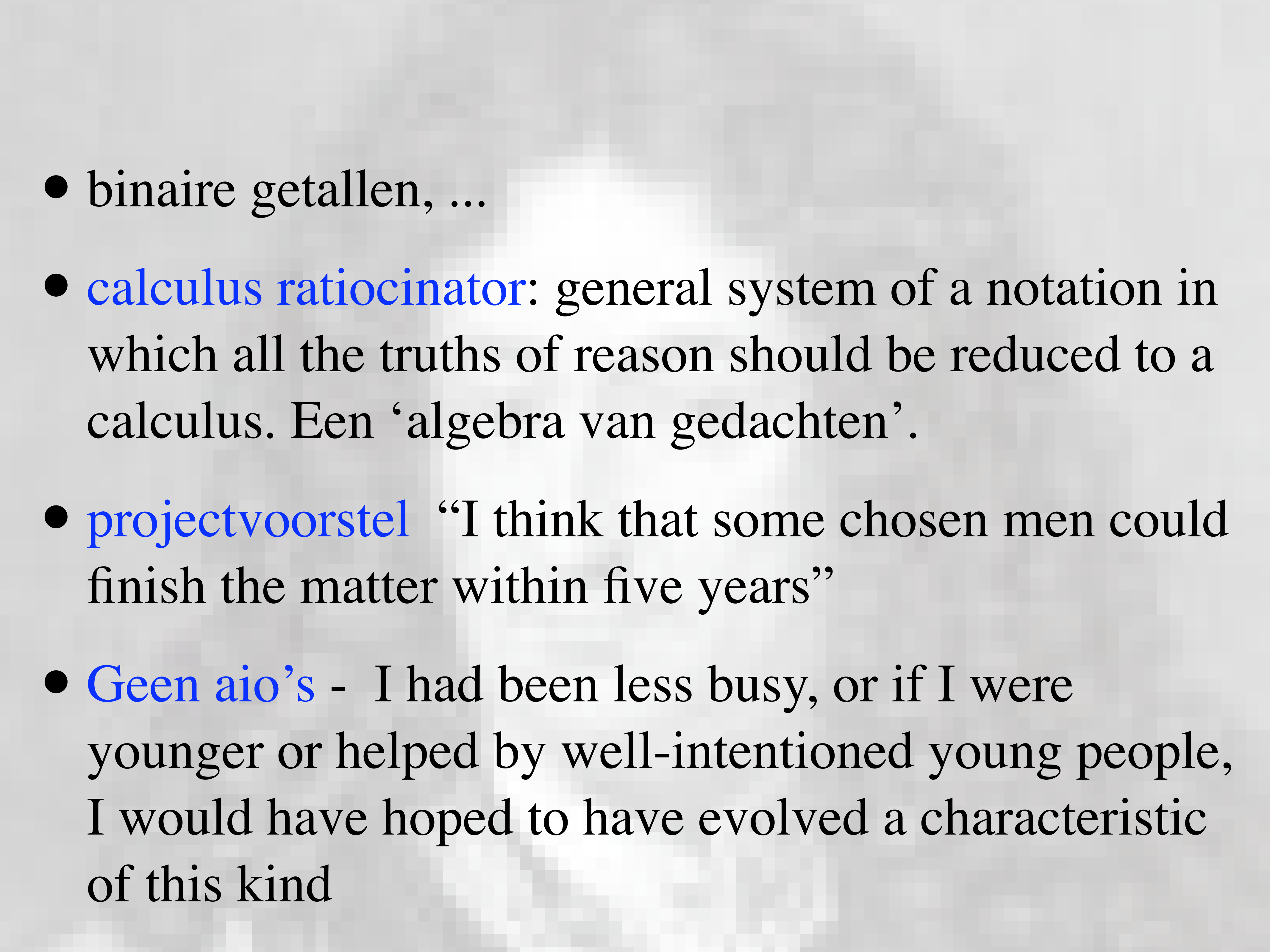
calculus ratiocinator

Leibniz equality:
in proof checker Coq

$$x = y$$

defined as

$$\forall P: P(x) \leftrightarrow P(y)$$



- binaire getallen, ...
- calculus ratiocinator: general system of a notation in which all the truths of reason should be reduced to a calculus. Een ‘algebra van gedachten’.
- projectvoorstel “I think that some chosen men could finish the matter within five years”
- Geen aio’s - I had been less busy, or if I were younger or helped by well-intentioned young people, I would have hoped to have evolved a characteristic of this kind

George Boole

1815 - 1864

1854 *An investigation into the Laws of Thought, on Which are founded the Mathematical Theories of Logic and Probabilities*.

One day in 1864 *he walked from his residence to the College, a distance of two miles, in the drenching rain, and lectured in wet clothes. The result was a feverish cold which soon fell upon his lungs and terminated his career ....*

What Macfarlane fails to say is that Boole's wife (Mary - niece of Sir George Everest, after whom the mountain is named) believed that a remedy should resemble the cause. She put Boole to bed and threw buckets of water over the bed since his illness had been caused by getting wet.

AN INVESTIGATION

OF

THE LAWS OF THOUGHT,

ON WHICH ARE FOUNDED

THE MATHEMATICAL THEORIES OF LOGIC
AND PROBABILITIES.

BY

GEORGE BOOLE, LL.D.

PROFESSOR OF MATHEMATICS IN QUEEN'S COLLEGE, CORK

LONDON:
WALTON AND MABERLEY,
UPPER GOWER-STREET, AND IVY-LANE, PATERNOSTER-ROW.
CAMBRIDGE: MACMILLAN AND CO.
1854.

Propositielogica, zeer vluchtige impressie

wetten van De Morgan:

$$\vdash \neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$$

$$\vdash \neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$$

Jan Łukasiewicz bewees in 1930 dat het mogelijk is om de bovenstaande wetten af te leiden op basis van slechts drie axioma's

$A \to (B \to A)$ (straks met Herbrand)

$(A \to (B \to C)) \to ((A \to B) \to (A \to C))$ (type van combinator S)

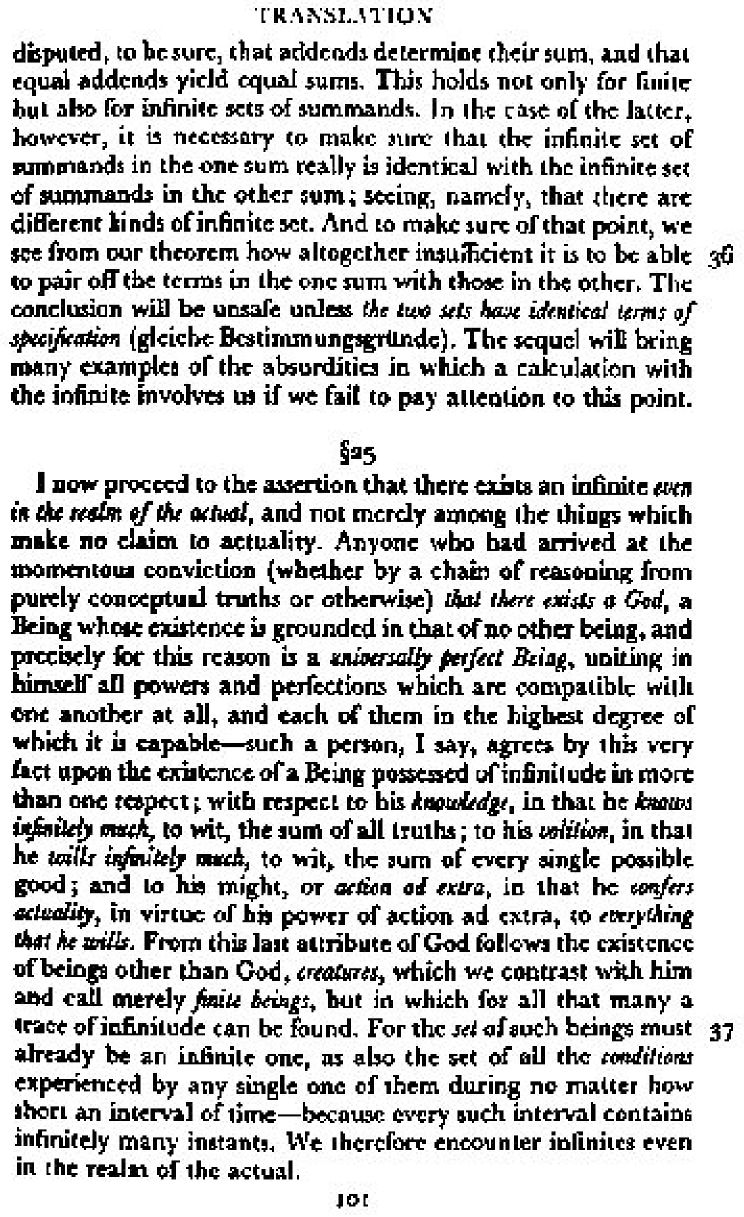
$(\neg A \to \neg B) \to (B \to A)$

in combinatie met de modus ponens: $A, A \to B \vdash B$
Bovendien geldt ook: $(A \to B) \leftrightarrow (\neg A \vee B)$

Bernhard Bolzano

1781 - 1848

Paradoxien des Unendlichen



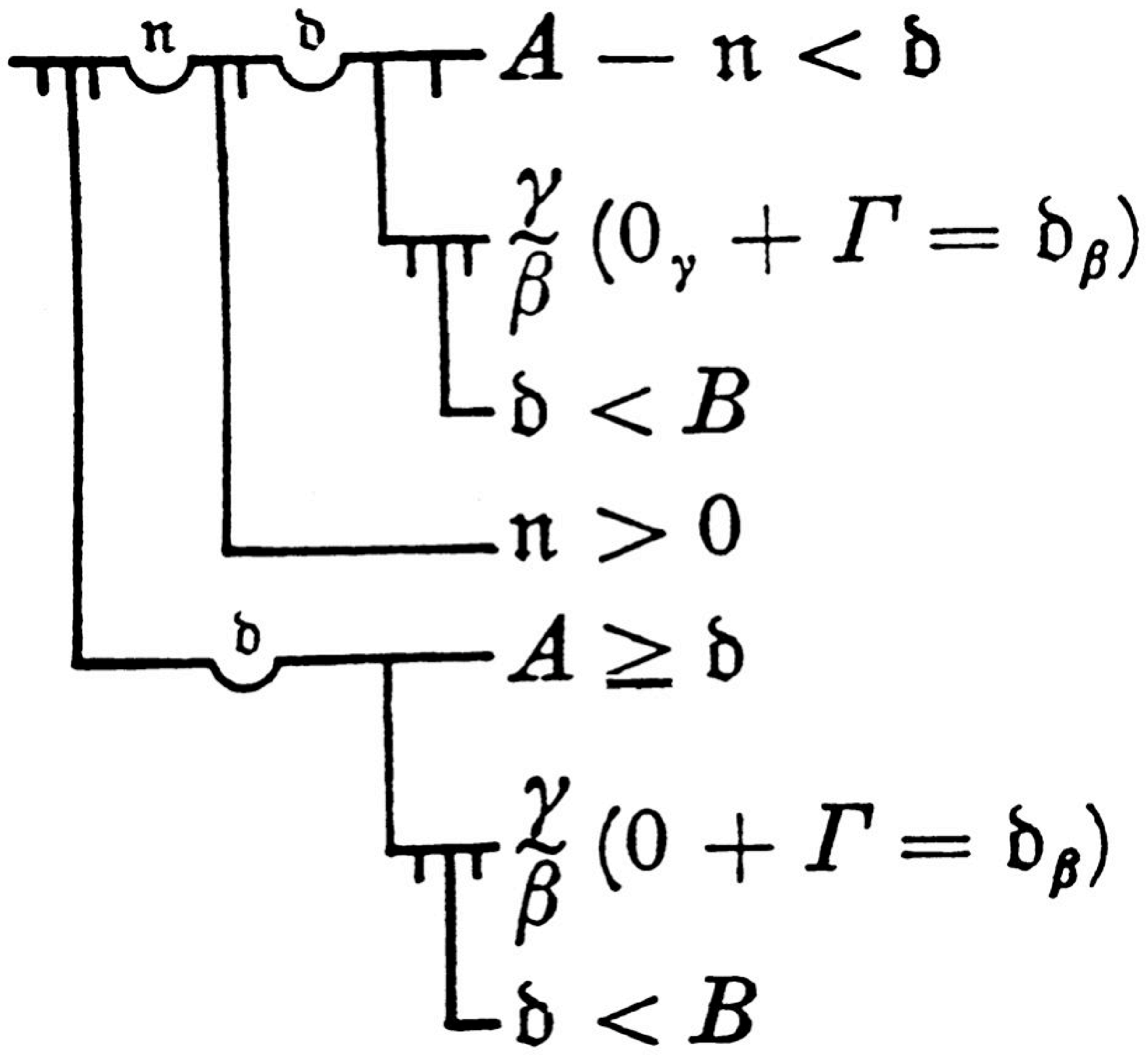
disputed, to be sure, that addends determine their sum, and that equal addends yield equal sums. This holds not only for finite but also for infinite sets of summands. In the case of the latter, however, it is necessary to make sure that the infinite set of summands in the one sum really is identical with the infinite set of summands in the other sum; seeing, namely, that there are different kinds of infinite set. And to make sure of that point, we see from our theorem how altogether insufficient it is to be able 36 to pair off the terms in the one sum with those in the other. The conclusion will be unsafe unless *the two sets have identical terms of specification* (gleiche Bestimmungsgründe). The sequel will bring many examples of the absurdities in which a calculation with the infinite involves us if we fail to pay attention to this point.

§25

I now proceed to the assertion that there exists an infinite *even in the realm of the actual*, and not merely among the things which make no claim to actuality. Anyone who had arrived at the momentous conviction (whether by a chain of reasoning from purely conceptual truths or otherwise) *that there exists a God*, a Being whose existence is grounded in that of no other being, and precisely for this reason is a *universally perfect Being*, uniting in himself all powers and perfections which are compatible with one another at all, and each of them in the highest degree of which it is capable—such a person, I say, agrees by this very fact upon the existence of a Being possessed of infinitude in more than one respect; with respect to his *knowledge*, in that he *knows infinitely much*, to wit, the sum of all truths; to his *volition*, in that he *wills infinitely much*, to wit, the sum of every single possible good; and to his might, or *action ad extra*, in that he *confers actuality*, in virtue of his power of action ad extra, *to everything that he wills*. From this last attribute of God follows the existence of beings other than God, *creatures*, which we contrast with him and call merely *finite beings*, but in which for all that many a trace of infinitude can be found. For the *set of* such beings must 37 already be an infinite one, as also the set of all the *conditions* experienced by any single one of them during no matter how short an interval of time—because every such interval contains infinitely many instants. We therefore encounter infinites even in the realm of the actual.

Gottlob Frege: Begriffschrift

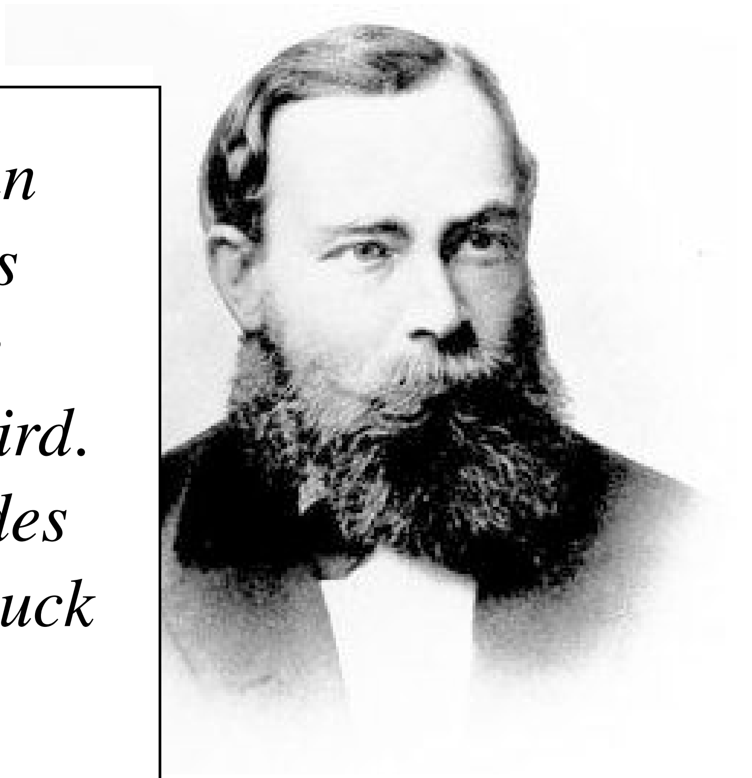
1848 - 1925


$\mathfrak{n}$
$\mathfrak{d}$
$A - \mathfrak{n} < \mathfrak{d}$
$\overset{\gamma}{\underset{\beta}{\sim}} (0_\gamma + \Gamma = \mathfrak{d}_\beta)$
$\mathfrak{d} < B$
$\mathfrak{n} > 0$
$\mathfrak{d}$
$A \geq \mathfrak{d}$
$\overset{\gamma}{\underset{\beta}{\sim}} (0 + \Gamma = \mathfrak{d}_\beta)$
$\mathfrak{d} < B$

2300 jaar later heeft Frege diens werk voltooid, zoals in de volledigheidsstelling van Goedel (1930) bewezen wordt.

Gottlob Frege: Begriffschrift

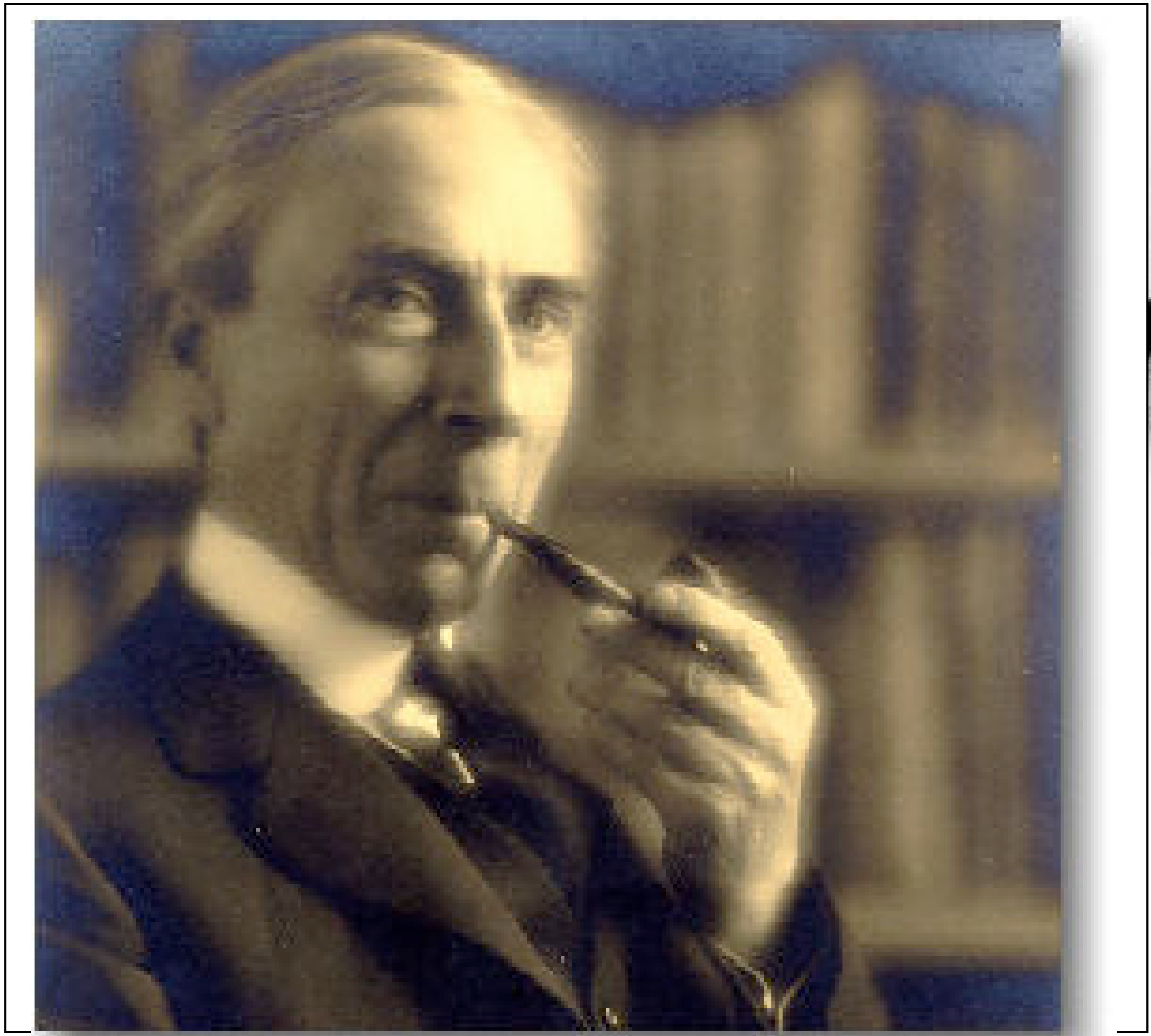
1848 - 1925

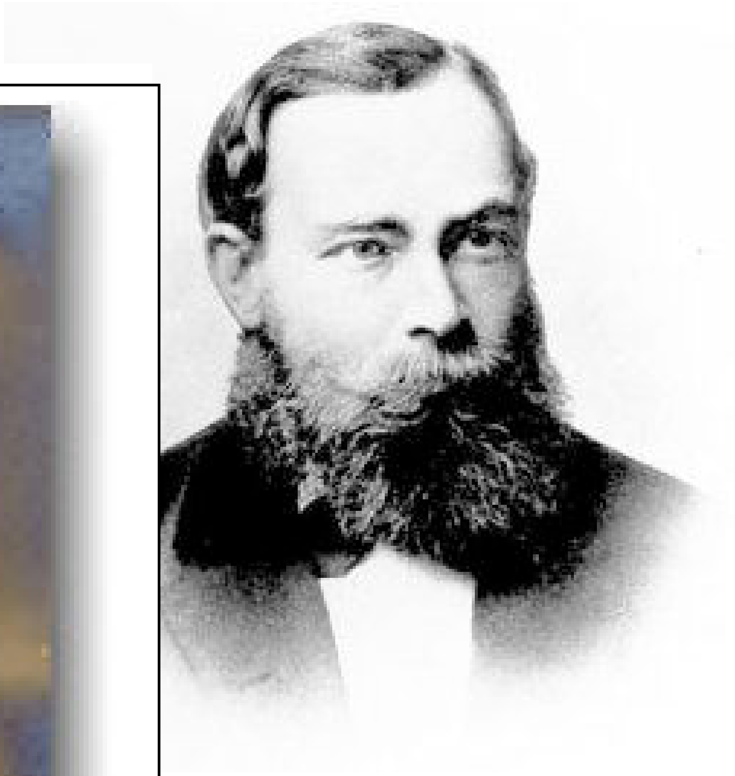


Einem wissenschaftlichen Schriftsteller kann kaum etwas Unerwünschteres begegnen, als dass ihm nach Vollendung einer Arbeit eine der Grundlagen seines Baues erschüttert wird. In diese Lage wurde ich durch einen Brief des Herrn Bertrand Russell versetzt, als der Druck dieses Bandes sich seinem Ende näherte.

Gottlob Frege: Begriffschrift

1848 - 1925





In plaats van Frege's notatie gebruiken we de moderne versie in de taal van de Predicatenlogica, hier informeel:

Peano's nine axioms, rephrased in contemporary notation, are:

1 1 is a natural number.
2 Every natural number is equal to itself (equality is reflexive).
3 For all natural numbers *a* and *b*, *a*=*b* if and only if *b*=*a* (equality is symmetric).
4 For all natural numbers *a*, *b*, and *c*, if *a*=*b* and *b*=*c* then *a*=*c* (equality is transitive).
5 If *a* = *b* and *b* is a natural number then *a* is a natural number.
6 If *a* is a natural number then S*a* is a natural number.
7 If *a* and *b* are natural numbers then *a*=*b* if and only if S*a* =S*b*.
8 If *a* is a natural number then S*a* is not equal to 1.
9 For every set *K*, if *1* is in *K* and for every natural number *x* in *K*, S*x* is also in *K*, then every natural number is in *K*. (It makes no difference here whether all elements of *K* are natural numbers.)

Peano's axioma's PA

1. $\forall x \forall y \forall z\, ((x+y)+z = x+(y+z))$
2. $\forall x \forall y\, (x+y = y+x)$
3. $\forall x \forall y \forall z\, ((x\cdot y)\cdot z = x\cdot(y\cdot z))$
4. $\forall x \forall y\, (x\cdot y = y\cdot x)$
5. $\forall x \forall y \forall z\, (x\cdot(y+z) = x\cdot y + x\cdot z)$
6. $\forall x\, ((x+0 = x) \wedge (x\cdot 0 = 0))$
7. $\forall x\, (x\cdot 1 = x)$
8. $\forall x \forall y \forall z\, ((x<y \wedge y<z) \Rightarrow x<z)$
9. $\forall x\, \neg(x<x)$
10. $\forall x \forall y\, (x<y \vee x=y \vee x>y)$
11. $\forall x \forall y \forall z\, (x<y \Rightarrow x+z<y+z)$
12. $\forall x \forall y \forall z\, (0<z \wedge x<y \Rightarrow x\cdot z < y\cdot z)$
13. $\forall x \forall y\, (x<y \Rightarrow \exists z(x+z=y))$
14. $0<1 \wedge \forall x\, (x>0 \Rightarrow x \geq 1)$
15. $\forall x\, (x \geq 0)$

To convert PA⁻ to PA, the **first-order induction schema** is added. This schema represents a countably infinite set of axioms. For each formula $\varphi(x, y_1, \ldots, y_k)$ in the language of Peano arithmetic, the **first-order induction axiom** for φ is the sentence

$$\forall \bar{y}(\phi(0,\bar{y}) \wedge \forall x(\phi(x,\bar{y}) \Rightarrow \phi(x+1,\bar{y})) \Rightarrow \forall x \phi(x,\bar{y}))$$

1.4 Ontology

Ontology: set theory, type theory

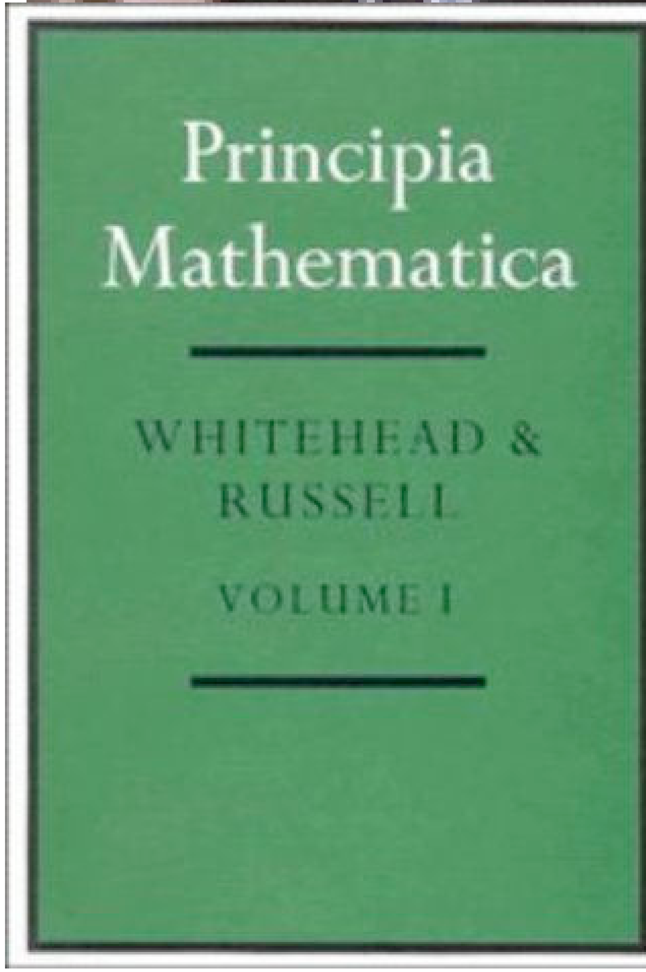
set theory	$\mathbb{N}$	Infinity
	$A \times B$	Cartesian Product
	$\{a{\in}A \mid P(a)\}$	Subset Selection
	$\{X \mid X \subseteq A\} = \mathcal{P}(A)$	Power Set
	$\{F(a) \mid a{\in}A\} = F``A$	Replacement

type theory inductively defined data types with their
recursively defined functions and closed under
function spaces $A{\rightarrow}B$ and dependent products $\Pi x{:}A.B_x$
developed by Russell, de Bruijn (for Automath),
extended by Scott, Martin-Löf, Girard, Huet and Coquand

Bertrand Russell

1872- 1970

Mathematics, rightly viewed, possesses not only truth, but supreme beauty – a beauty cold and austere, without appeal to any part of our weaker nature, without the gorgeous trappings of painting or music, yet sublimely pure, and capable of a stern perfection such as only the greatest art can show.



Jacques Herbrand

1908-1931

logisch programmeren

↓

PROLOG

In 1931, he was awarded a Rockefeller fellowship that enabled him to study in Germany, first with John von Neumann in Berlin, then during June with Emil Artin in Hamburg, and finally, in July, with Emmy Noether.

$$\begin{array}{lcl}
x \Rightarrow y & \rightarrow & x \cdot y + x + 1 \\
x \vee y & \rightarrow & x \cdot y + x + y \\
\neg x & \rightarrow & x + 1 \\
x + 0 & \rightarrow & x \\
x + x & \rightarrow & 0 \\
x \cdot 0 & \rightarrow & 0 \\
x \cdot 1 & \rightarrow & x \\
x \cdot x & \rightarrow & x \\
x \cdot (y + z) & \rightarrow & x \cdot y + x \cdot z
\end{array}$$

As an example we exhibit the following reduction of the tautology $p \Rightarrow (q \Rightarrow p)$ to 1, where the most relevant associative/commutative steps are also stipulated.

$$\begin{array}{lcl}
p \Rightarrow (q \Rightarrow p) & \rightarrow & p(q \Rightarrow p) + p + 1 \\
 & \rightarrow & p(qp + q + 1) + p + 1 = p((qp + q) + 1) + p + 1 \\
 & \rightarrow & p(qp + q) + p1 + p + 1 \\
 & \rightarrow & pqp + pq + p1 + p + 1 \\
 & \rightarrow & pqp + pq + p + p + 1 = pqp + pq + (p + p) + 1 \\
 & \rightarrow & pqp + pq + 0 + 1 = pqp + (pq + 0) + 1 \\
 & \rightarrow & pqp + pq + 1 = (pp)q + pq + 1 \\
 & \rightarrow & pq + pq + 1 = (pq + pq) + 1 \\
 & \rightarrow & 0 + 1 = 1 + 0 \rightarrow 1
\end{array}$$

Mathematical Problems

Lecture delivered before the International Congress of Mathematicians at Paris in 1900

By Professor David Hilbert

Who of us would not be glad to lift the veil behind which the future lies hidden; to cast a glance at the next advances of our science and at the secrets of its development during future centuries? What particular goals will there be toward which the leading mathematical spirits of coming generations will strive? What new methods and new facts in the wide and rich field of mathematical thought will the new centuries disclose?

2. Applications of logic: metamathematics

Views on Mathematics

$\boxed{\text{“}\vdash A\text{” stands for “}A\text{ is provable”}}$

after Aristotle

$$\begin{tikzcd} \boxed{\text{Axioms}} \arrow[d, "\text{Reasoning}"] \\ \text{Mathematics} \end{tikzcd}$$

after Frege

$$\begin{tikzcd} \boxed{\text{Axioms}} \arrow[d, "\boxed{\text{Logic}}"] \\ \boxed{\text{Mathematics}} \end{tikzcd}$$

Gödel	(1931)	Mathematics is incomplete	$\nvdash G$ and $\nvdash \neg G$ for some G
			'p is a proof of A' is decidable
Turing	(1936)	Mathematics is undecidable	$\{A \mid\, \vdash A\}$ non-computable

Corollary. There are relatively short statements with very long proofs

Op het moment dat vastlag wat de reikwijdte van het bewijzen was, kon een essentiële beperking ervan aangetoond worden. Zo liet Goedel in zijn onvolledigheidssteling (1931) zien dat een theorie die de elementaire rekenkunde bevat en geen inconsistenties kan afleiden (dwz een uitspraak kan nooit zowel bewezen als weerlegd worden) altijd onvolledig is: er is voor zo'n theorie een uitspraak die noch bewezen noch weerlegd kan worden.

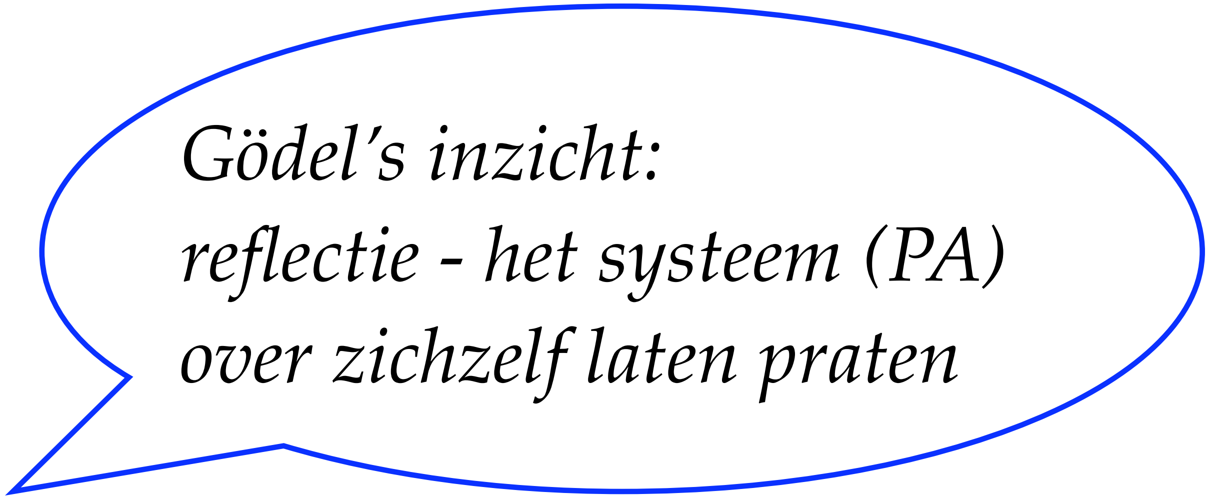
Kurt Gödel

1906- 1978

Either mathematics is too big for the human mind or the human mind is more than a machine.

3. Reflection

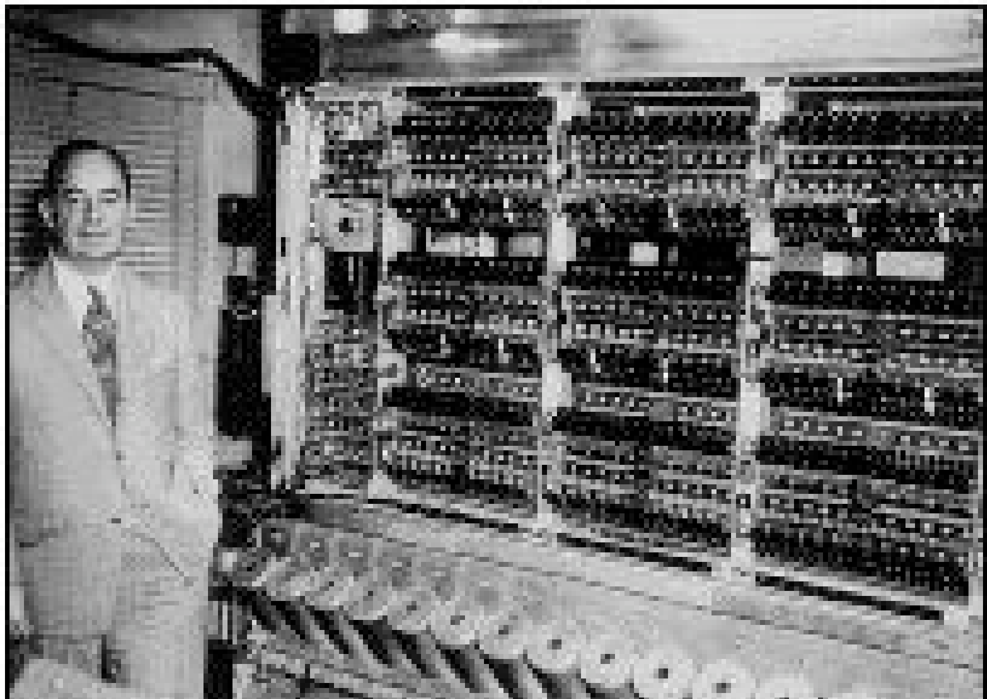
In arithmetic statements are about arithmetic.
Through coding arithmetic can be about arithmetic.
Proving statements by proving results about their codes is powerful.

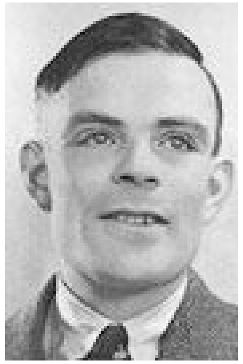


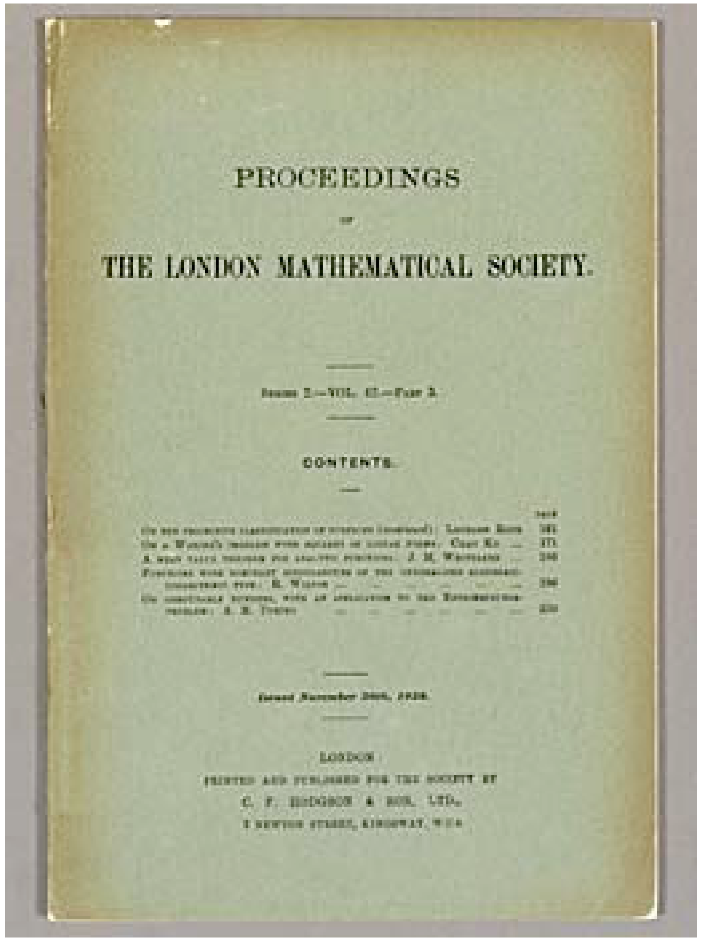
John von Neumann

1903 - 1957

game theory, computer architecture, cellular automata, set theory,
ordinaalgetallen







PROCEEDINGS

OF

THE LONDON MATHEMATICAL SOCIETY.

SERIES 2.—VOL. 42.—PART 3.

CONTENTS.

	PAGE
[illegible]	161
[illegible]	171
[illegible] J. M. Whittaker	210
[illegible] R. Wilson	226
On computable numbers, with an application to the Entscheidungsproblem: A. M. Turing	230

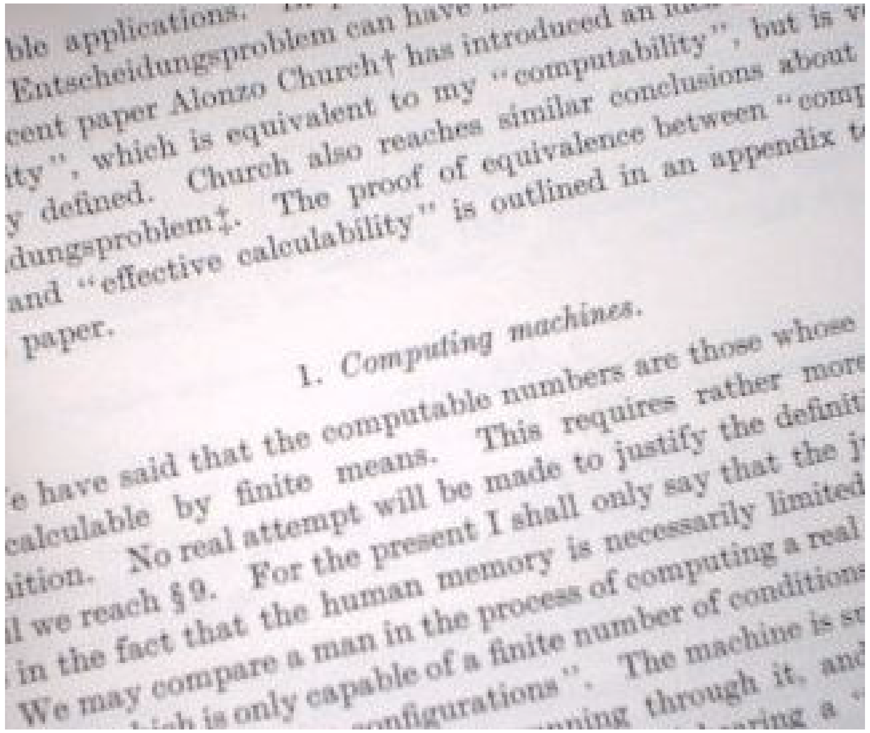
Issued November 30th, 1936.

LONDON

PRINTED AND PUBLISHED FOR THE SOCIETY BY

C. F. HODGSON & SON, LTD.,

2 NEWTON STREET, KINGSWAY, W.C.2



ble applications.

Entscheidungsproblem can have

cent paper Alonzo Church† has introduced an

ity", which is equivalent to my "computability", but is

y defined. Church also reaches similar conclusions about

dungsproblem‡. The proof of equivalence between "com

and "effective calculability" is outlined in an appendix t

paper.

1. *Computing machines.*

e have said that the computable numbers are those whose

calculable by finite means. This requires rather more

No real attempt will be made to justify the definit

ition. For the present I shall only say that the j

l we reach §9. For the present I shall only say that the j

in the fact that the human memory is necessarily limited

We may compare a man in the process of computing a real

hich is only capable of a finite number of condition

configurations". The machine is s

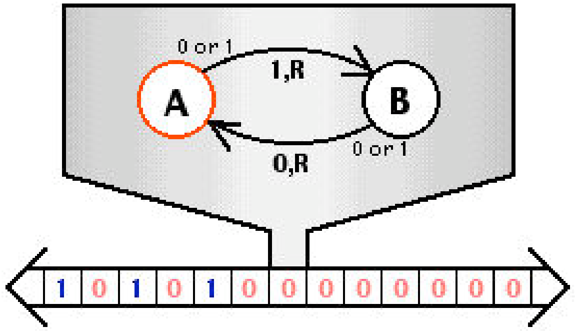
nning through it, and

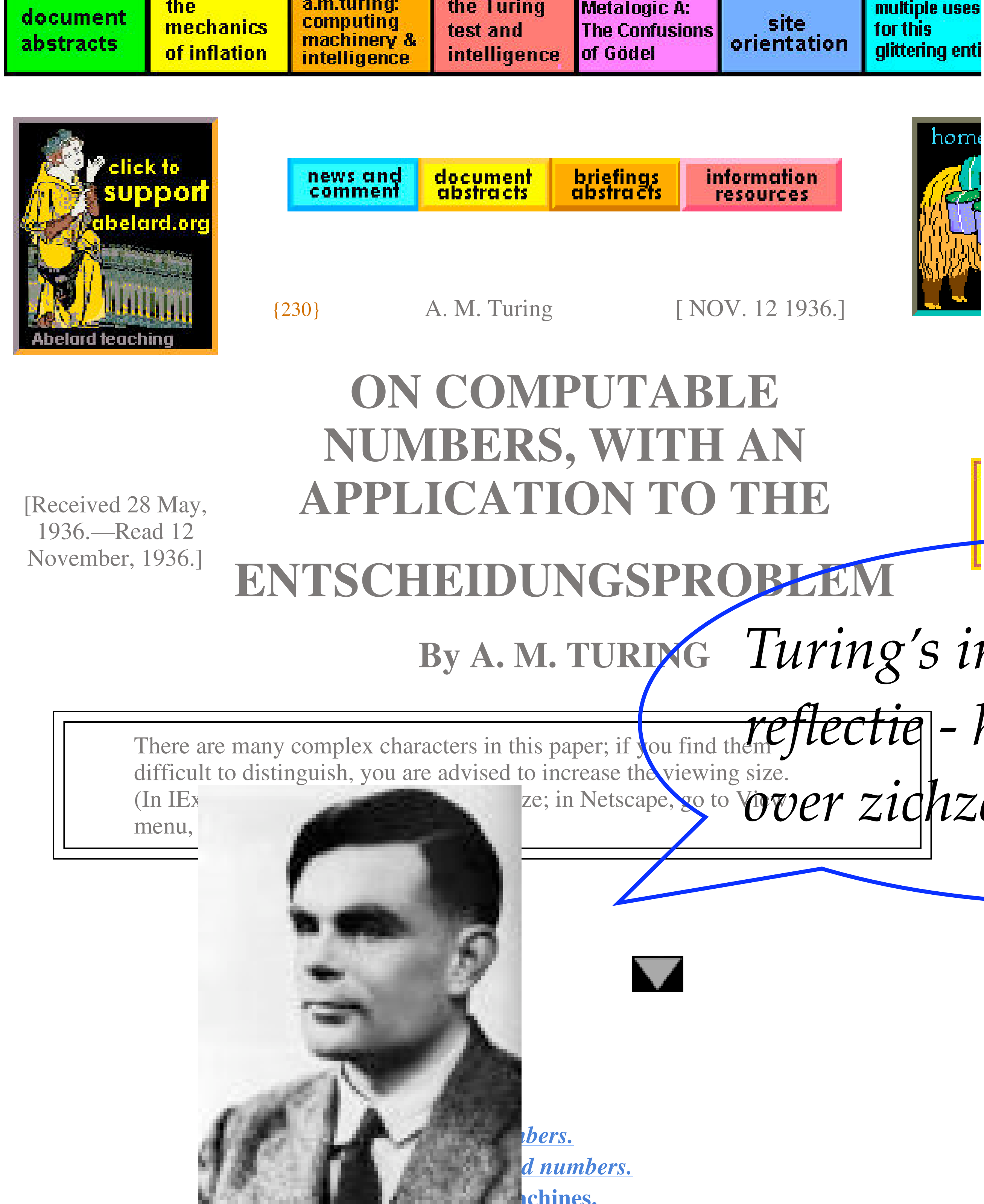
Alan Turing

1912 - 1945

promoveerde bij von Neumann in 1938

The enigma of intelligence




document abstracts
the mechanics of inflation
a.m.turing: computing machinery & intelligence
the Turing test and intelligence
Metalogic A: The Confusions of Gödel
site orientation
multiple uses for this glittering enti
click to support abelard.org
Abelard teaching
news and comment
document abstracts
briefings abstracts
information resources
home

{230} A. M. Turing [NOV. 12 1936.]

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

[Received 28 May, 1936.—Read 12 November, 1936.]

By A. M. TURING

There are many complex characters in this paper; if you find them difficult to distinguish, you are advised to increase the viewing size. (In IEx ze; in Netscape, go to View menu,

bers.
d numbers.
chines.

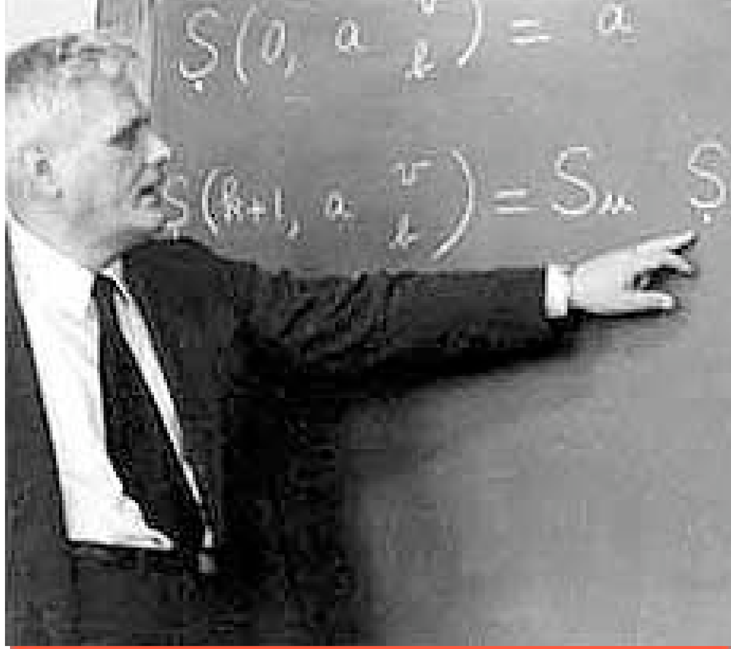
Turing's inzicht:
reflectie - het systeem (TM's) over zichzelf laten praten

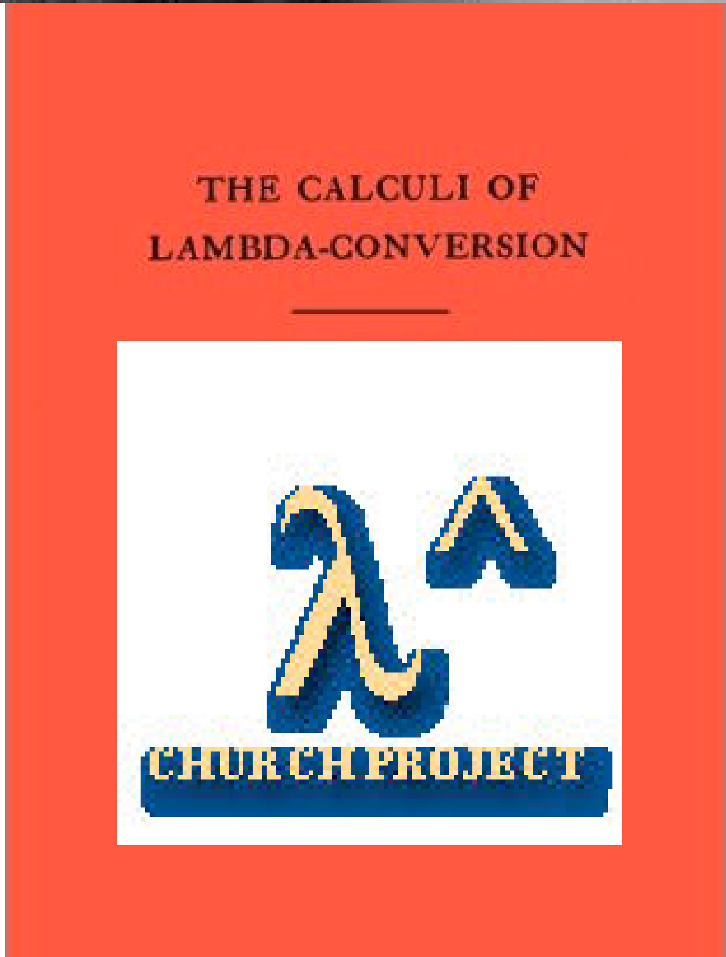
Een paar jaar later werd in 1936 geheel vastgelegd wat de reikwijdte van het rekenen is. Dat werd gedaan door Church via lambda calculus en Turing via een machine model van het rekenen. Ook nu weer kon de conclusie gemaakt worden dat berekeningen maken zijn beperking heeft: er is geen computer (of wat op hetzelfde neer komt geen algoritme) die alle mogelijke wiskundige problemen oplost. Dit is een sterker resultaat dan de onvolledigheidsstelling van Goedel.

Alonzo Church

1903- 1995

At the time of his death, Church was widely regarded as the greatest living logician in the world

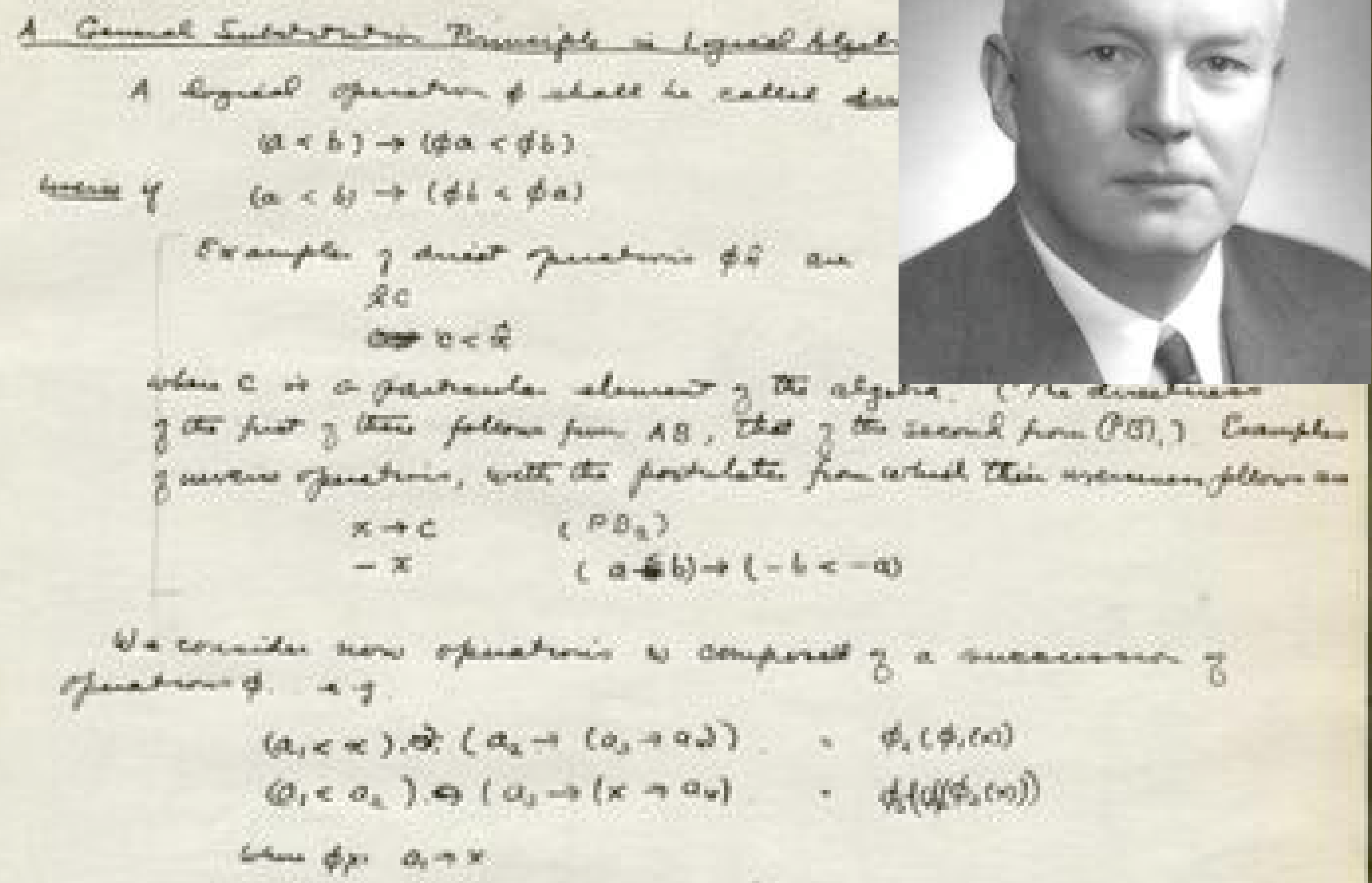



THE CALCULI OF

LAMBDA-CONVERSION

Haskell Curry

1900- 1982



A General Substitution Principle in Logical Algeb

A logical operation ϕ shall be called dir

$(a < b) \to (\phi a < \phi b)$

inverse if $(a < b) \to (\phi b < \phi a)$

Examples of direct operations ϕc are

xc

~~[illegible]~~ $c < x$

where c is a particular element of the algebra. (The directness of the first of these follows from AB, that of the second from $(PB)_1$.) Examples of inverse operations, with the postulate from which their inverseness follows are

$x \to c$ $\quad (PB_2)$

$-x$ $\quad (a < b) \to (-b < -a)$

We consider now operations compounded of a succession of operations ϕ. e.g.

$(a_1 < x) . \to (a_2 \to (a_3 \to a_4)) \quad = \quad \phi_4(\phi_1(x))$

$(a_1 < a_2) . \to (a_3 \to (x \to a_4)) \quad = \quad \phi_5(\phi_4(\phi_3(x)))$

where ϕ_3: $a_3 \to x$

Alfred Tarski

1901 - 1983

waarheid is ondefinieerbaar

The essence of mathematics lies in its freedom.

1.3 Computing

Skolem	[1922]	Primitive Computable Functions via primitive recursive schemes
Herbrand-Gödel	[1931]	Total Computable Functions via Term Rewrite Systems (TRSs)
Church-Turing	[1936]	Partial Computable Functions via λ-calculus and Turing Machines

Application: the von Neumann computer

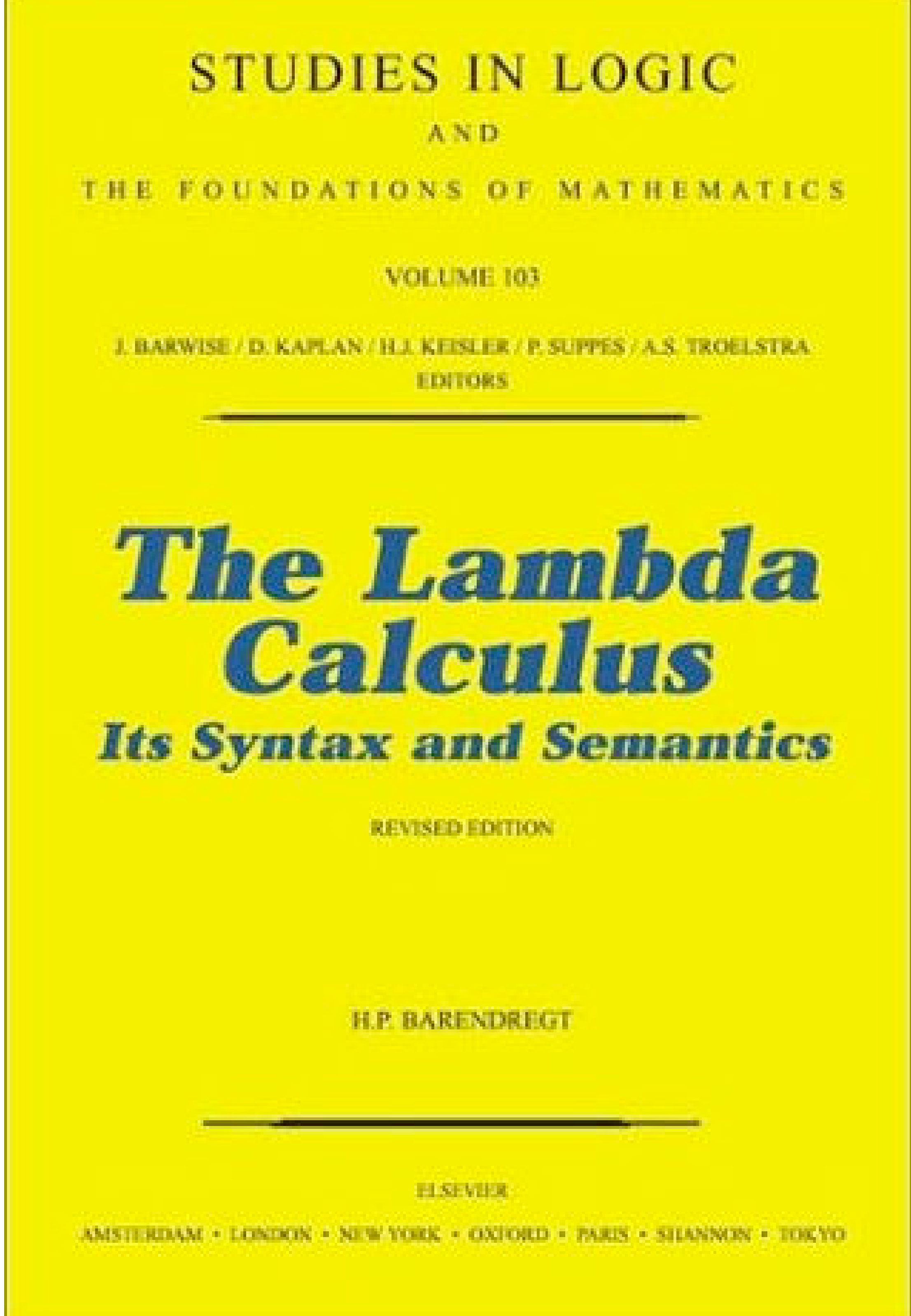
Simple computational model (Schönfinkel)

$$\begin{aligned} \mathsf{I}\ x &= x \\ \mathsf{K}\ x\ y &= x \\ \mathsf{S}\ x\ y\ z &= (x\ z)\ (y\ z) \end{aligned}$$

Lambda Calculus

$$(\lambda x.Z(x))Y \longrightarrow Z(Y)$$

Turing compleet



STUDIES IN LOGIC

AND

THE FOUNDATIONS OF MATHEMATICS

VOLUME 103

J. BARWISE / D. KAPLAN / H.J. KEISLER / P. SUPPES / A.S. TROELSTRA
EDITORS

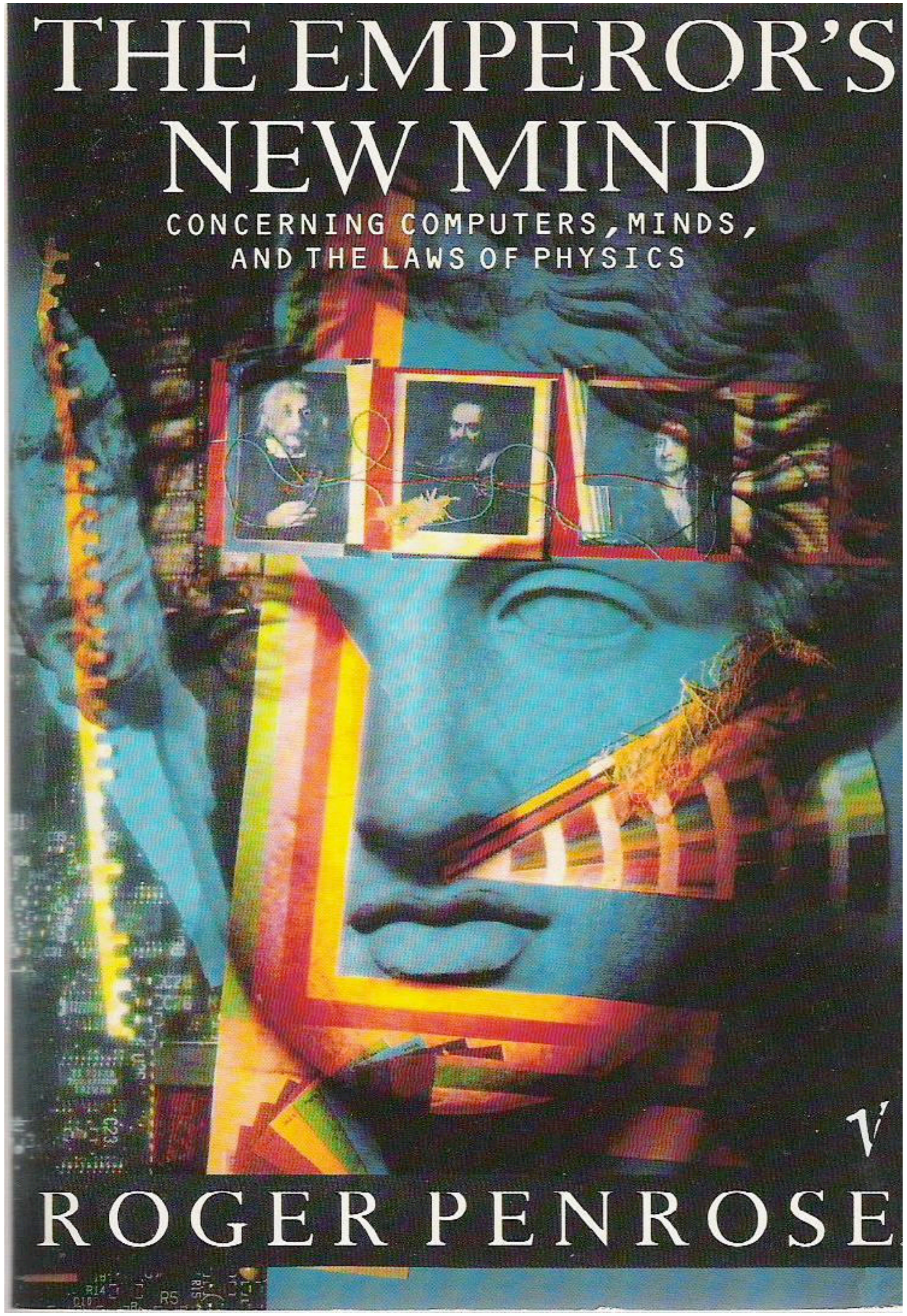
The Lambda Calculus
Its Syntax and Semantics

REVISED EDITION

H.P. BARENDREGT

ELSEVIER

AMSTERDAM • LONDON • NEW YORK • OXFORD • PARIS • SHANNON • TOKYO



THE EMPEROR'S NEW MIND

CONCERNING COMPUTERS, MINDS, AND THE LAWS OF PHYSICS

V

ROGER PENROSE

who are doing the 'knowing', while the algorithm just follows the rules we have told it to follow? Or are we ourselves merely following rules that we have been programmed to follow from the construction of our brains and from our environment? The issue is not really simply one of algorithms, but also a question of how one judges what is true and what is not true. These are central issues that we shall have to return to later. The question of mathematical truth (and its non-algorithmic nature) will be considered in Chapter 4. At least we should now have some feeling about the *meanings* of the terms 'algorithm' and 'computability', and an understanding of some of the related issues.

CHURCH'S LAMBDA CALCULUS

The concept of computability is a very important and beautiful mathematical idea. It is also a remarkably recent one – as things of such a fundamental nature go in mathematics – having been first put forward in the 1930s. It is an idea which cuts across *all* areas of mathematics (although it may well be true that most mathematicians do not, as yet, often worry themselves about computability questions). The power of the idea lies partly in the fact that some well-defined operations in mathematics are actually *not* computable (like the stopping, or otherwise, of a Turing machine; we shall see other examples in Chapter 4). For if there were no such non-computable things, the concept of computability would not have much mathematical interest. Mathematicians, after all, like puzzles. It can be an intriguing puzzle for them to decide, of some mathematical operation, whether or not it is computable. It is especially intriguing because the general solution of *that* puzzle is itself non-computable!

One thing should be made clear. Computability is a genuine 'absolute' mathematical concept. It is an abstract idea which lies quite beyond any particular realization in terms of the 'Turing machines' as I have described them. As I have remarked before, we do not need to attach any particular significance to the 'tapes' and 'internal states', etc., which characterize Turing's ingenious but particular approach. There are also other ways of expressing the idea of computability, historically the first of these being the

remarkable 'lambda calculus' of the American logician Alonzo Church, with the assistance of Stephen C. Kleene. Church's procedure was quite different, and distinctly more abstract from that of Turing. In fact, in the form that Church stated his ideas, there is rather little obvious connection between them and anything that one might call 'mechanical'. The key idea lying behind Church's procedure is, indeed, *abstract* in its very essence – a mathematical operation that Church actually referred to as 'abstraction'.

I feel that it is worth while to give a brief description of Church's scheme not only because it emphasizes that computability is a mathematical idea, independent of any particular concept of computing machine, but also because it illustrates the power of abstract ideas in mathematics. The reader who is not readily conversant with mathematical ideas, nor intrigued by such things for their own sake, may, at this stage, prefer to move on to the next chapter – and there would not be significant loss in the flow of argument. Nevertheless, I believe that such readers might benefit by bearing with me for a while longer, and thus witnessing some of the magical economy of Church's scheme (see Church 1941).

In this scheme one is concerned with a 'universe' of objects, denoted by say

$$a, b, c, d, \ldots, z, a', b', \ldots, z', a'', b'', \ldots, a''', \ldots, a'''', \ldots$$

each of which stands for a mathematical operation or *function*. (The reason for the primed letters is simply to allow an unlimited supply of symbols to denote such functions.) The 'arguments' of these functions – that is to say, the things on which these functions act – are other things of the same kind, i.e. also functions. Moreover, the result (or 'value') of one such function acting on another is to be again a function. (There is, indeed, a wonderful economy of concepts in Church's system.) Thus, when we write*

$$a = bc$$

* A more familiar form of notation would have been to write $a = b(c)$, say, but these particular parentheses are not really necessary and it is better to get used to their omission. To include them consistently would lead to rather cumbersome formulae such as $(f(p))(q)$ and $((f(p))(q))(r)$, instead of $(fp)q$ and $((fp)q)r$ respectively.

Hoe moet *e-mail: (λx.jwkxcs.vu.nl)@* gelezen worden? Met lambda calculus (λ-calculus).

De λ-calculus is de eenvoudigste programmeertaal die bestaat - de 'oerprogrammeertaal'. De woorden in die taal heten λ-termen. Die worden successievelijk getransformeerd totdat een λ-term ontstaat die een 'antwoord' voorstelt. Eén enkele regel of voorschrift, volstaat om deze transformaties uit te voeren, de beta-regel (β-regel). Het wonder van de λ-calculus is dat die β-regel zo simpel is, en dat er toch elke berekening of symboolmanipulatie, hoe ingewikkeld ook, mee kan worden uitgevoerd. In de λ-calculus wordt het begrip 'berekenen' of 'manipuleren met symbolen', met informatie, tot de kleinst denkbare bestanddelen teruggebracht. De werking van deze magische regel bestaat eenvoudig uit het invullen van een woord in een tweede woord op de daarvoor aangegeven plaatsen. In de volgende voorbeelden kleuren we het woord dat wordt ingevuld rood, en het woord waarin wordt ingevuld, blauw. De plaatsen waar wordt ingevuld zijn zwarte letters (officieel variabelen), en de variabele waarvoor moet worden ingevuld staat achter de λ.

Bijvoorbeeld. Vul voor elke letter z in het woord ziezo het woord aap in.
Het resultaat is dan aapieaapo . Deze herschrijfstap of β-reductiestap wordt in wiskundige notatie zo opgeschreven:

$$(\lambda z.\ \text{ziezo})\ \text{aap} \to \text{aapieaapo}$$

Een ander voorbeeld van een herschrijfstap is

$(\lambda a.\ aap)\ ziezo \to ziezoziezop$

In de taal van logica en wiskunde mogen woorden ook haakjes bevatten:
‘(‘ en ‘)’. Behalve deze symbolen wordt ook nog de punt ‘.’ gebruikt, zoals ook in de voorbeelden boven. We laten de kleuren verder weg.

Een woord (officieel, een λ-term) kan ook uit een enkele letter bestaan. Dus we hebben ook bijvoorbeeld de stappen

$(\lambda x.x)aap \to aap$
$(\lambda x.yxy)z \to yzy$

Een meer zinvol voorbeeld is het *verdubbelen* van woorden:

$(\lambda x.xx)aap \rightarrow aapaap$

Dus (λx.xx) is een woordverdubbelaar, elk woord waarop deze λ-term wordt 'toegepast', wordt in één stap verdubbeld. Een woord *uitwissen* kan ook:

$(\lambda a.goed)slecht \rightarrow goed$

Want goed bevat geen a!

λ-termen kunnen ook op zichzelf worden toegepast: *zelfapplicatie*. Voor de verdubbelaar levert dit een interessant verschijnsel op:

$(\lambda x.xx)(\lambda x.xx) \rightarrow (\lambda x.xx)(\lambda x.xx)$

Het is inmiddels wel in te zien dat *(λx.jwkxcs.vu.nl)@* → het emailadres van J.W. Klop.

De invul-instructie λx , die officieel lambda-abstractie heet, kan ook herhaald worden.

λb((λa.abba)c)d

Rekenen met λ-termen.

maal: M = λfgx.f(gx)
plus: P = λfgxy.fx(gxy)
1 = λfx.fx
2 = λfx.f(fx)
3 = λfx.f(f(fx))
4 = λfx.f(f(f(fx)))

$$2 \text{ maal } 3 = M23 \rightarrow$$
$$(\lambda gx.2(gx))3 \rightarrow$$
$$\lambda x.2(3x) \rightarrow$$
$$\lambda xx'.3x(3xx') \rightarrow$$
$$\lambda xx'.(\lambda x'.x(x(xx')))(3xx') \rightarrow$$
$$\lambda xx'.x(x(x(3xx'))) \rightarrow$$
$$\lambda xx'.x(x(x((\lambda x'.x(x(xx')))x'))) \rightarrow$$
$$\lambda xx'.x(x(x(x(x(xx'))))) = 6$$

$$
\begin{aligned}
&3 \text{ maal } 2 = M32 \rightarrow \\
&\quad (\lambda gx.3(gx))2 \rightarrow \\
&\quad \lambda x.3(2x) \rightarrow \\
&\quad \lambda xx'.2x(2x(2xx')) \rightarrow \\
&\quad \lambda xx'.(\lambda x'.x(xx'))(2x(2xx')) \rightarrow \\
&\quad \lambda xx'.x(x(2x(2xx'))) \rightarrow \\
&\quad \lambda xx'.x(x((\lambda x'.x(xx'))(2xx'))) \rightarrow \\
&\quad \lambda xx'.x(x(x(x(2xx')))) \rightarrow \\
&\quad \lambda xx'.x(x(x(x((\lambda x'.x(xx'))x')))) \rightarrow \\
&\quad \lambda xx'.x(x(x(x(x(xx'))))) = 6
\end{aligned}
$$

$$
\begin{aligned}
&2 \text{ plus } 3 = P23 \rightarrow \\
&(\lambda gxy.2x(gxy))3 \rightarrow \\
&\lambda xy.2x(3xy) \rightarrow \\
&\lambda xy.(\lambda x'.x(xx'))(3xy) \rightarrow \\
&\lambda xy.x(x(3xy)) \rightarrow \\
&\lambda xy.x(x((\lambda x'.x(x(xx')))y)) \rightarrow \\
&\lambda xy.x(x(x(x(xy)))) = 5
\end{aligned}
$$

3 tot de macht 2 $= 3^2 = 23$

$(\lambda fx.f(fx))3 \rightarrow$

$\lambda x.3(3x) = \lambda x.(\lambda fx'.f(f(fx')))(3x) \rightarrow$

$\lambda xx'.3x(3x(3xx')) \rightarrow$

$\lambda xx'.(\lambda x'.x(x(xx')))(3x(3xx')) \rightarrow$

$\lambda xx'.x(x(x(3x(3xx')))) \rightarrow$

$\lambda xx'.x(x(x((\lambda x'.x(x(xx')))(3xx')))) \rightarrow$

$\lambda xx'.x(x(x(x(x(x(3xx')))))) \rightarrow$

$\lambda xx'.x(x(x(x(x(x((\lambda x'.x(x(xx')))x')))))) \rightarrow$

$\lambda xx'.x(x(x(x(x(x(x(x(xx')))))))) = 9$

$(\lambda ab.b(aab))(\lambda ab.b(aab))\ M \rightarrow$

$(\lambda b.b((\lambda ab.b(aab))(\lambda ab.b(aab))b))M \rightarrow$

$M((\lambda ab.b(aab))(\lambda ab.b(aab))M)$

$\blacksquare \rightarrow\ \rightarrow M(\ \blacksquare\ )$

$\blacksquare\ =\ M(\ \blacksquare\ )$

$\blacksquare$ *is fixed point van M*

Een eindige λ-term is een programma.

Met Y kunnen we oneindige cyclische woorden maken, bv. aapaapaap....

$X = \mathrm{aap}X$

$X = (\lambda x.\mathrm{aap}x)X$

$X = Y_T(\lambda x.\mathrm{aap}x)$

$N = \lambda ab.b(aab)$

$NN\lambda x.\mathrm{aap}x \rightarrow$

$(\lambda b.b(NNb))\lambda x.\mathrm{aap}x \rightarrow$

$(\lambda x.\mathrm{aap}x)(NN\lambda x.\mathrm{aap}x) \rightarrow$

$\mathrm{aap}(NN\lambda x.\mathrm{aap}x) \rightarrow$

$\mathrm{aap}((\lambda b.b(NNb))\lambda x.\mathrm{aap}x) \rightarrow$

$\mathrm{aap}((\lambda x.\mathrm{aap}x)(NN\lambda x.\mathrm{aap}x)) \rightarrow$

$\mathrm{aap}(\mathrm{aap}(NN\lambda x.\mathrm{aap}x)) \rightarrow$

$\mathrm{aap}(\mathrm{aap}((\lambda b.b(NNb))\lambda x.\mathrm{aap}x)) \rightarrow$

$\mathrm{aap}(\mathrm{aap}((\lambda x.\mathrm{aap}x)(NN\lambda x.\mathrm{aap}x))) \rightarrow$

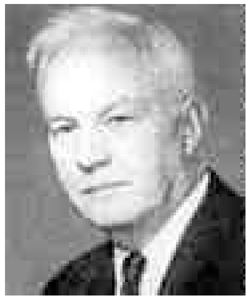
$\mathrm{aap}(\mathrm{aap}(\mathrm{aap}(NN\lambda x.\mathrm{aap}x))) \rightarrow$

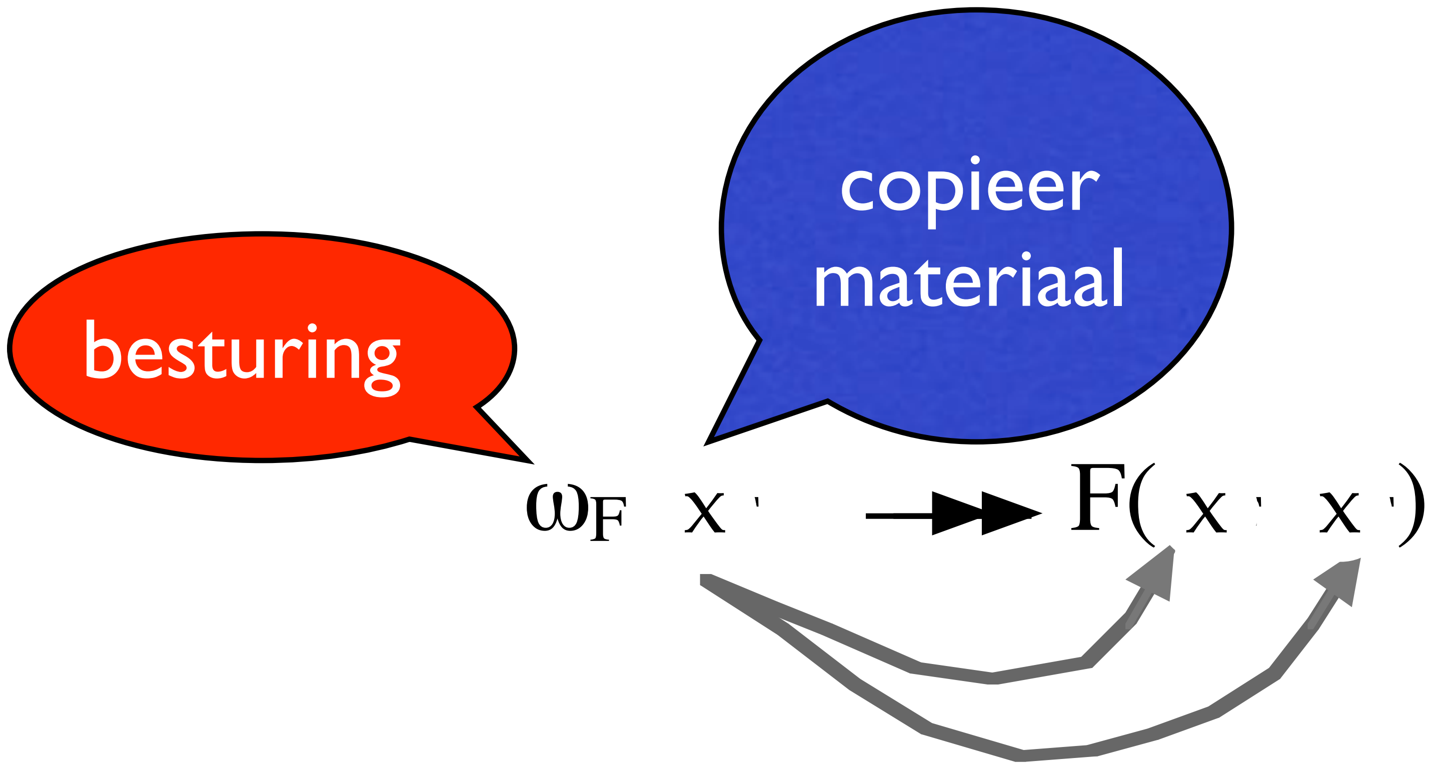
$\mathrm{aap}(\mathrm{aap}(\mathrm{aap}((\lambda b.b(NNb))\lambda x.\mathrm{aap}x))) \rightarrow$

$\mathrm{aap}(\mathrm{aap}(\mathrm{aap}((\lambda x.\mathrm{aap}x)(NN\lambda x.\mathrm{aap}x)))) \rightarrow$

$\mathrm{aap}(\mathrm{aap}(\mathrm{aap}(\mathrm{aap}(NN\lambda x.\mathrm{aap}x)))) \rightarrow$

How to make a fixed point combinator

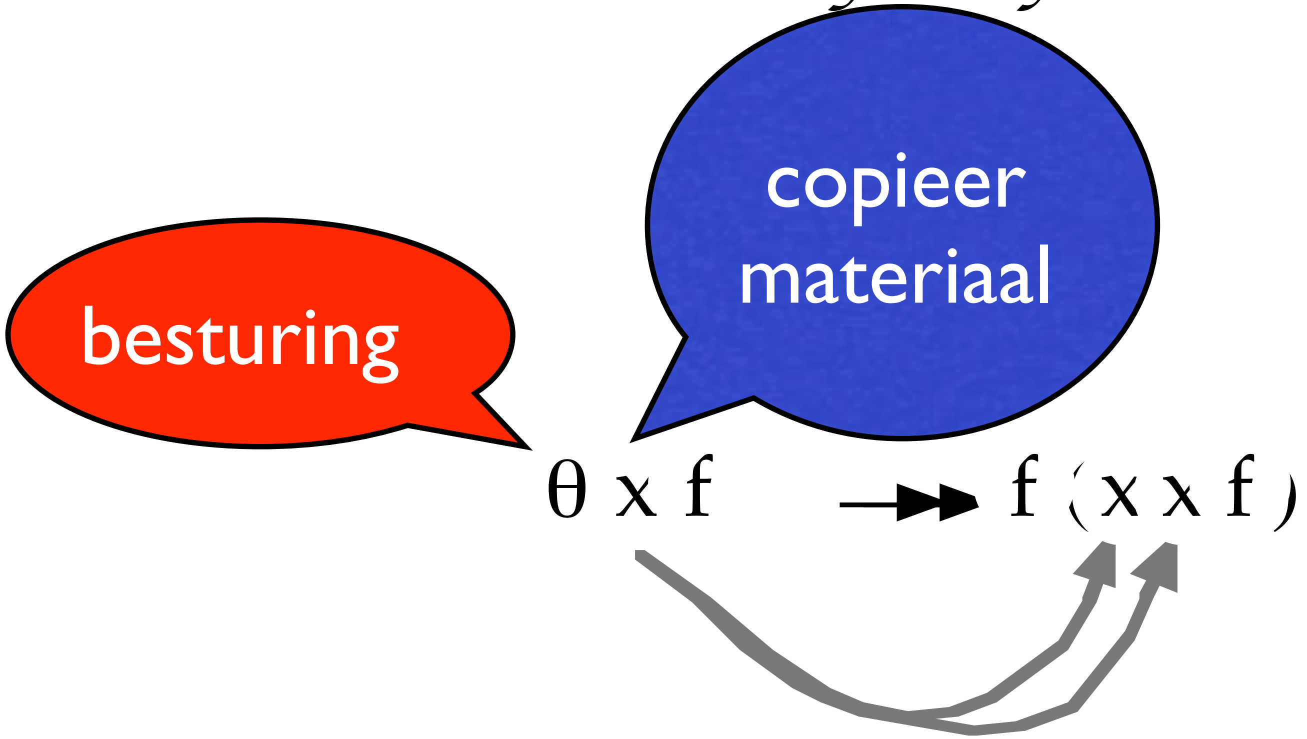




$$\omega_F\ x \twoheadrightarrow F(x\ x)$$

$$Y = \lambda f.\ \omega_f\omega_f \equiv \lambda f.\ (\lambda x.f(xx))(\lambda x.f(xx))$$

How to make a better fixed point combinator



$$\theta\ x\ f \twoheadrightarrow f\,(x\ x\ f)$$

$$\theta \equiv \lambda xf.\ f(xxf)$$

$$Y_T \equiv (\lambda xf.\ f(xxf))(\lambda xf.\ f(xxf))$$

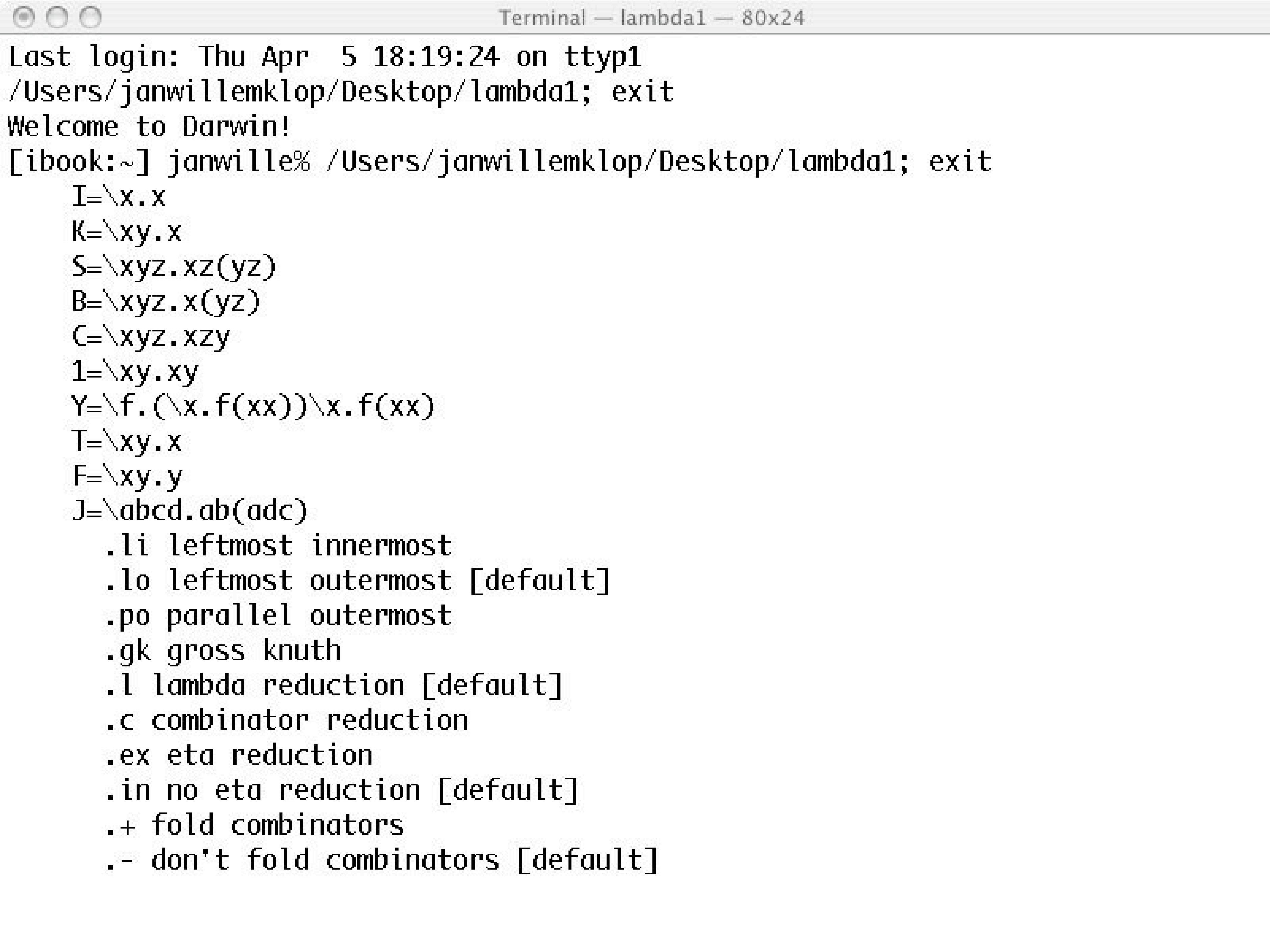
1.3 Computing

Skolem	[1922]	Primitive Computable Functions via primitive recursive schemes
Herbrand-Gödel	[1931]	Total Computable Functions via Term Rewrite Systems (TRSs)
Church-Turing	[1936]	Partial Computable Functions via λ-calculus and Turing Machines

Application: the von Neumann computer

Simple computational model (Schönfinkel)

$$\begin{aligned} \mathsf{I}\ x &= x \\ \mathsf{K}\ x\ y &= x \\ \mathsf{S}\ x\ y\ z &= (x\ z)\ (y\ z) \end{aligned}$$



```
Last login: Thu Apr  5 18:19:24 on ttyp1
/Users/janwillemklop/Desktop/lambda1; exit
Welcome to Darwin!
[ibook:~] janwille% /Users/janwillemklop/Desktop/lambda1; exit
    I=\x.x
    K=\xy.x
    S=\xyz.xz(yz)
    B=\xyz.x(yz)
    C=\xyz.xzy
    1=\xy.xy
    Y=\f.(\x.f(xx))\x.f(xx)
    T=\xy.x
    F=\xy.y
    J=\abcd.ab(adc)
      .li leftmost innermost
      .lo leftmost outermost [default]
      .po parallel outermost
      .gk gross knuth
      .l lambda reduction [default]
      .c combinator reduction
      .ex eta reduction
      .in no eta reduction [default]
      .+ fold combinators
      .- don't fold combinators [default]
```

SII(SII)

I(SII)(I(SII))

SII(I(SII))

I(I(SII))(I(I(SII)))

I(SII)(I(I(SII)))

SII(I(I(SII)))

I(I(I(SII)))(I(I(I(SII))))

I(I(SII))(I(I(I(SII))))

I(SII)(I(I(I(SII))))

SII(I(I(I(SII))))

I(I(I(I(SII))))(I(I(I(I(SII)))))

I(I(I(SII)))(I(I(I(I(SII)))))

I(I(SII))(I(I(I(I(SII)))))

I(SII)(I(I(I(I(SII)))))

SII(I(I(I(I(SII)))))

I(I(I(I(I(SII)))))(I(I(I(I(I(SII))))))

I(I(I(I(SII))))(I(I(I(I(I(SII))))))

I(I(I(SII)))(I(I(I(I(I(SII))))))

I(I(SII))(I(I(I(I(I(SII))))))

I(SII)(I(I(I(I(I(SII))))))

SII(I(I(I(I(I(SII))))))

I(I(I(I(I(I(SII))))))(I(I(I(I(I(I(SII)))))))

I(I(I(I(I(SII)))))(I(I(I(I(I(I(SII)))))))

I(I(I(I(SII))))(I(I(I(I(I(I(SII)))))))

I(I(I(SII)))(I(I(I(I(I(I(SII)))))))

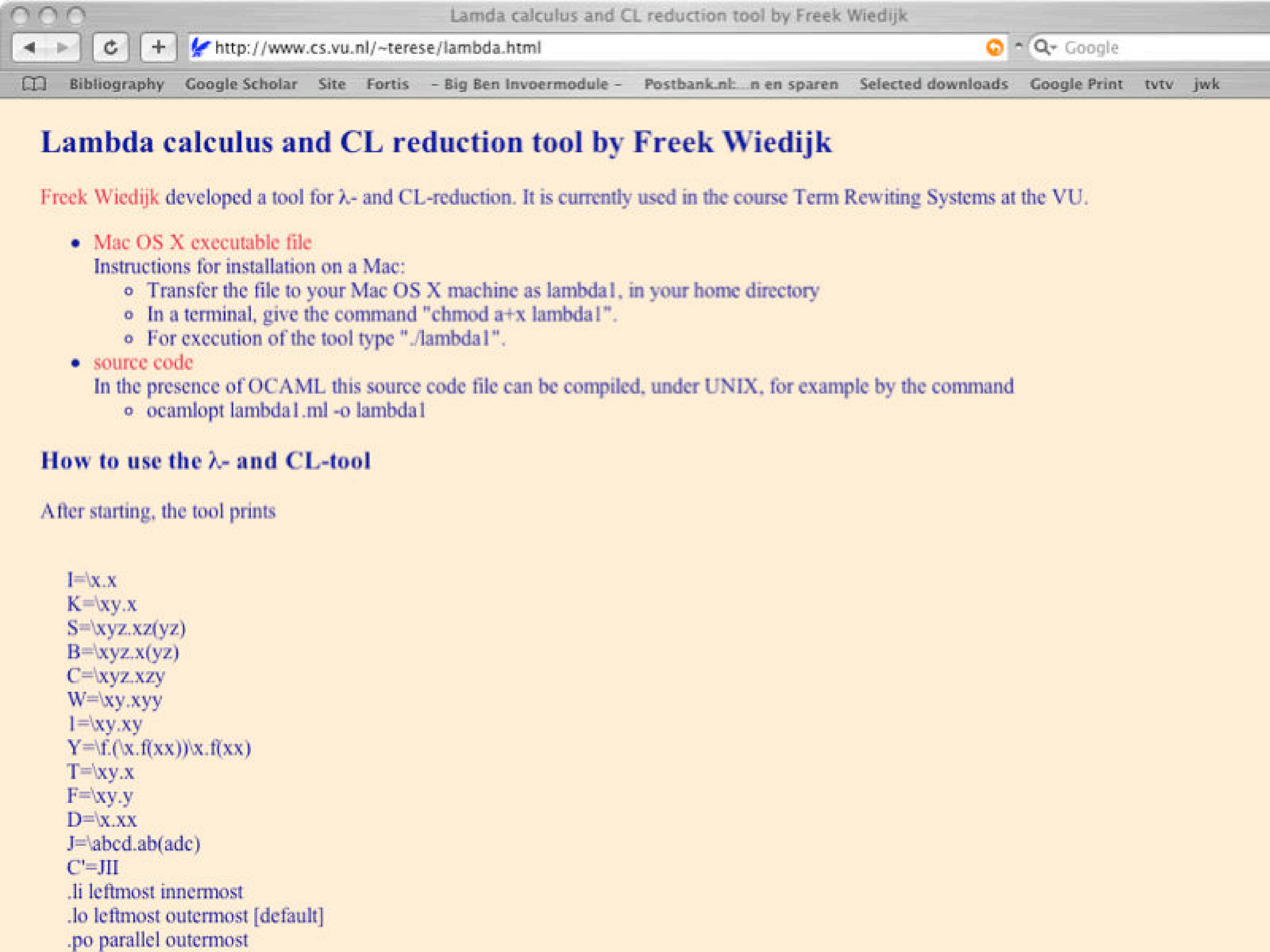
I(I(SII))(I(I(I(I(I(I(SII)))))))

I(SII)(I(I(I(I(I(I(SII)))))))

SII(I(I(I(I(I(I(SII)))))))

I(I(I(I(I(I(I(SII)))))))(I(I(I(I(I(I(I(SII))))))))

```
A=SSS
AAA
AAA
SSSAA
SA(SA)A
AA(SAA)
SSSA(SAA)
SA(SA)(SAA)
A(SAA)(SA(SAA))
SSS(SAA)(SA(SAA))
S(SAA)(S(SAA))(SA(SAA))
SAA(SA(SAA))(S(SAA)(SA(SAA)))
A(SA(SAA))(A(SA(SAA)))(S(SAA)(SA(SAA)))
SSS(SA(SAA))(A(SA(SAA)))(S(SAA)(SA(SAA)))
S(SA(SAA))(S(SA(SAA)))(A(SA(SAA)))(S(SAA)(SA(SAA)))
SA(SAA)(A(SA(SAA)))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
A(A(SA(SAA)))(SAA(A(SA(SAA))))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
SSS(A(SA(SAA)))(SAA(A(SA(SAA))))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
S(A(SA(SAA)))(S(A(SA(SAA))))(SAA(A(SA(SAA))))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
A(SA(SAA))(SAA(A(SA(SAA))))(S(A(SA(SAA)))(SAA(A(SA(SAA)))))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
SSS(SA(SAA))(SAA(A(SA(SAA))))(S(A(SA(SAA)))(SAA(A(SA(SAA)))))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
S(SA(SAA))(S(SA(SAA)))(SAA(A(SA(SAA))))(S(A(SA(SAA)))(SAA(A(SA(SAA)))))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
SA(SAA)(SAA(A(SA(SAA))))(S(SA(SAA))(SAA(A(SA(SAA)))))(S(A(SA(SAA)))(SAA(A(SA(SAA)))))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
A(SAA(A(SA(SAA))))(SAA(SAA(A(SA(SAA)))))(S(SA(SAA))(SAA(A(SA(SAA)))))(S(A(SA(SAA)))(SAA(A(SA(SAA)))))(S(SA(SAA))(A(SA(SAA))))(S(SAA)(SA(SAA)))
```


Lamda calculus and CL reduction tool by Freek Wiedijk
http://www.cs.vu.nl/~terese/lambda.html
Google
Bibliography
Google Scholar
Site
Fortis
- Big Ben Invoermodule -
Postbank.nl:...n en sparen
Selected downloads
Google Print
tvtv
jwk

Lambda calculus and CL reduction tool by Freek Wiedijk

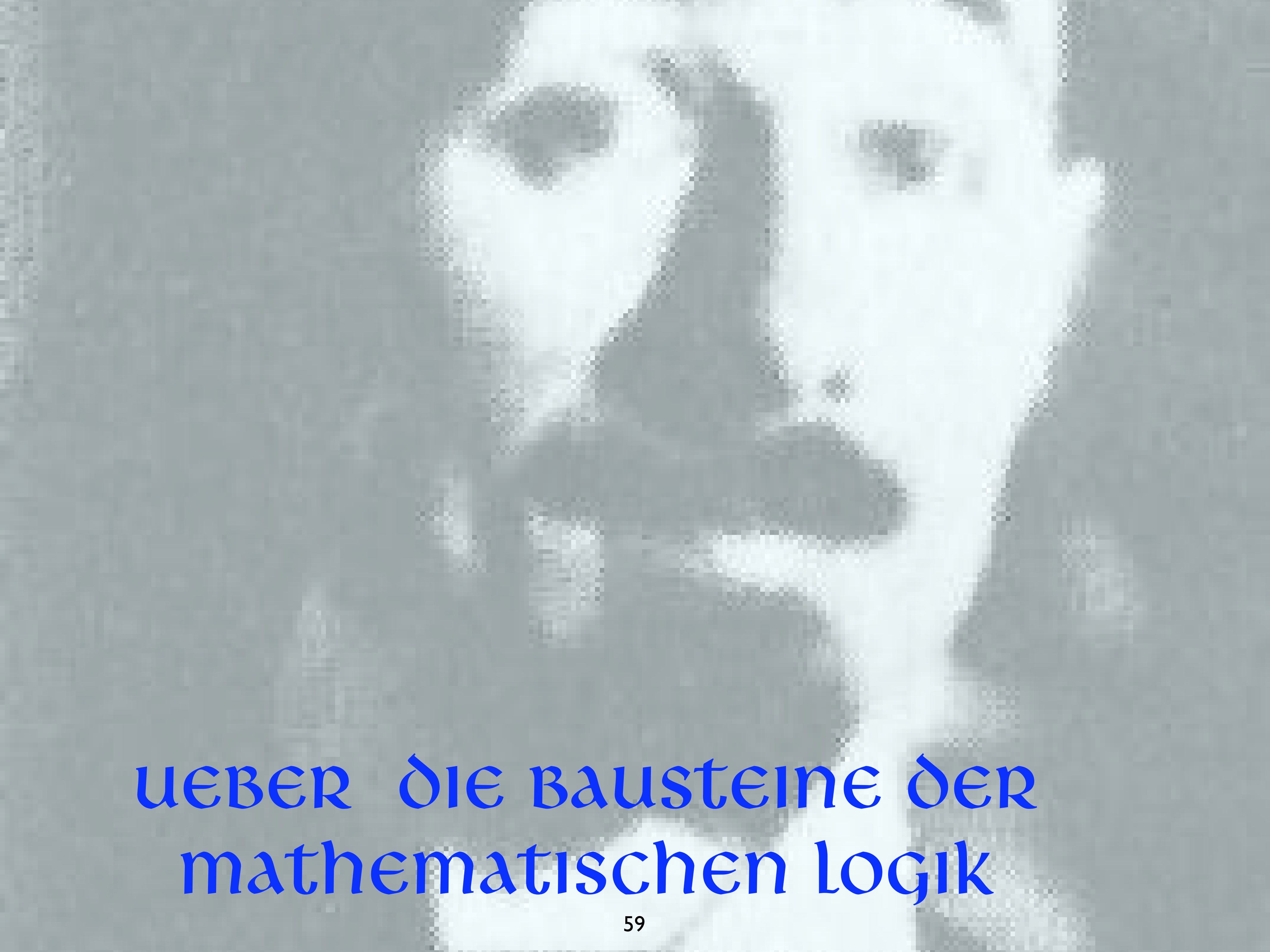
Freek Wiedijk developed a tool for λ- and CL-reduction. It is currently used in the course Term Rewiting Systems at the VU.

- Mac OS X executable file
  Instructions for installation on a Mac:
  - Transfer the file to your Mac OS X machine as lambda1, in your home directory
  - In a terminal, give the command "chmod a+x lambda1".
  - For execution of the tool type "./lambda1".
- source code
  In the presence of OCAML this source code file can be compiled, under UNIX, for example by the command
  - ocamlopt lambda1.ml -o lambda1

How to use the λ- and CL-tool

After starting, the tool prints

```
I=\x.x
K=\xy.x
S=\xyz.xz(yz)
B=\xyz.x(yz)
C=\xyz.xzy
W=\xy.xyy
1=\xy.xy
Y=\f.(\x.f(xx))\x.f(xx)
T=\xy.x
F=\xy.y
D=\x.xx
J=\abcd.ab(adc)
C'=JII
.li leftmost innermost
.lo leftmost outermost [default]
.po parallel outermost
```



Ueber die Bausteine der mathematischen Logik

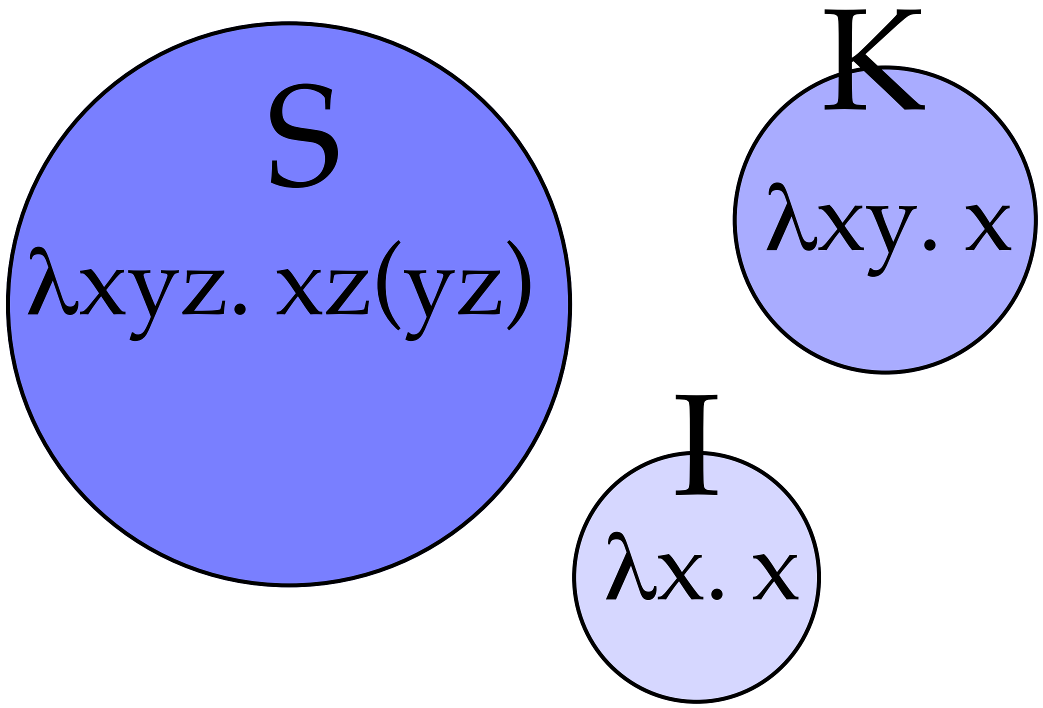
Combinatory Logic

$\mathrm{I}x \longrightarrow x$

$\mathrm{K}xy \longrightarrow x$

$\mathrm{S}xyz \longrightarrow xz(yz)$

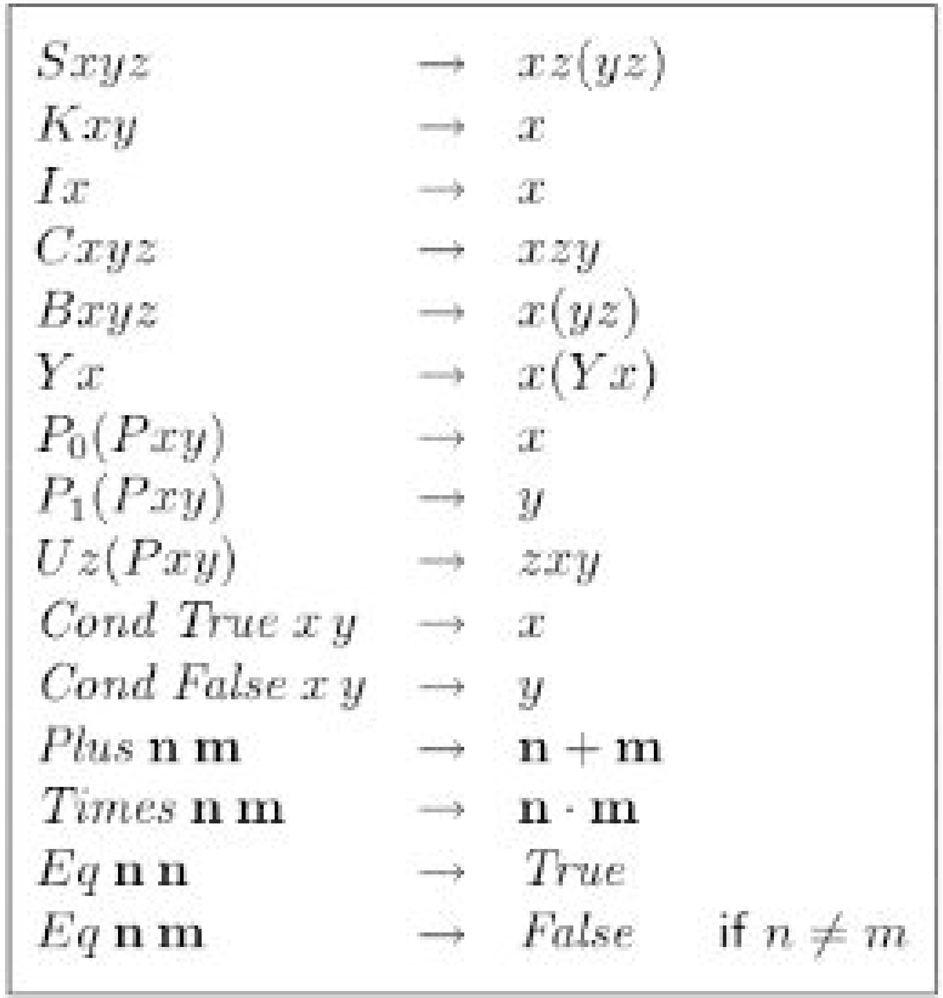
Turing compleet



1924. "Über die Bausteine der mathematischen Logik"

Moses Schönfinkel

SKIM



$Sxyz$	$\rightarrow$	$xz(yz)$	
Kxy	$\rightarrow$	x	
Ix	$\rightarrow$	x	
$Cxyz$	$\rightarrow$	xzy	
$Bxyz$	$\rightarrow$	$x(yz)$	
Yx	$\rightarrow$	$x(Yx)$	
$P_0(Pxy)$	$\rightarrow$	x	
$P_1(Pxy)$	$\rightarrow$	y	
$Uz(Pxy)$	$\rightarrow$	zxy	
$\mathit{Cond}\ \mathit{True}\ x\ y$	$\rightarrow$	x	
$\mathit{Cond}\ \mathit{False}\ x\ y$	$\rightarrow$	y	
$\mathit{Plus}\ \mathbf{n}\ \mathbf{m}$	$\rightarrow$	$\mathbf{n} + \mathbf{m}$	
$\mathit{Times}\ \mathbf{n}\ \mathbf{m}$	$\rightarrow$	$\mathbf{n} \cdot \mathbf{m}$	
$\mathit{Eq}\ \mathbf{n}\ \mathbf{n}$	$\rightarrow$	True	
$\mathit{Eq}\ \mathbf{n}\ \mathbf{m}$	$\rightarrow$	False	if $n \neq m$

voorloper van Miranda

valorisatie!

1.2 (Propositional) Logic

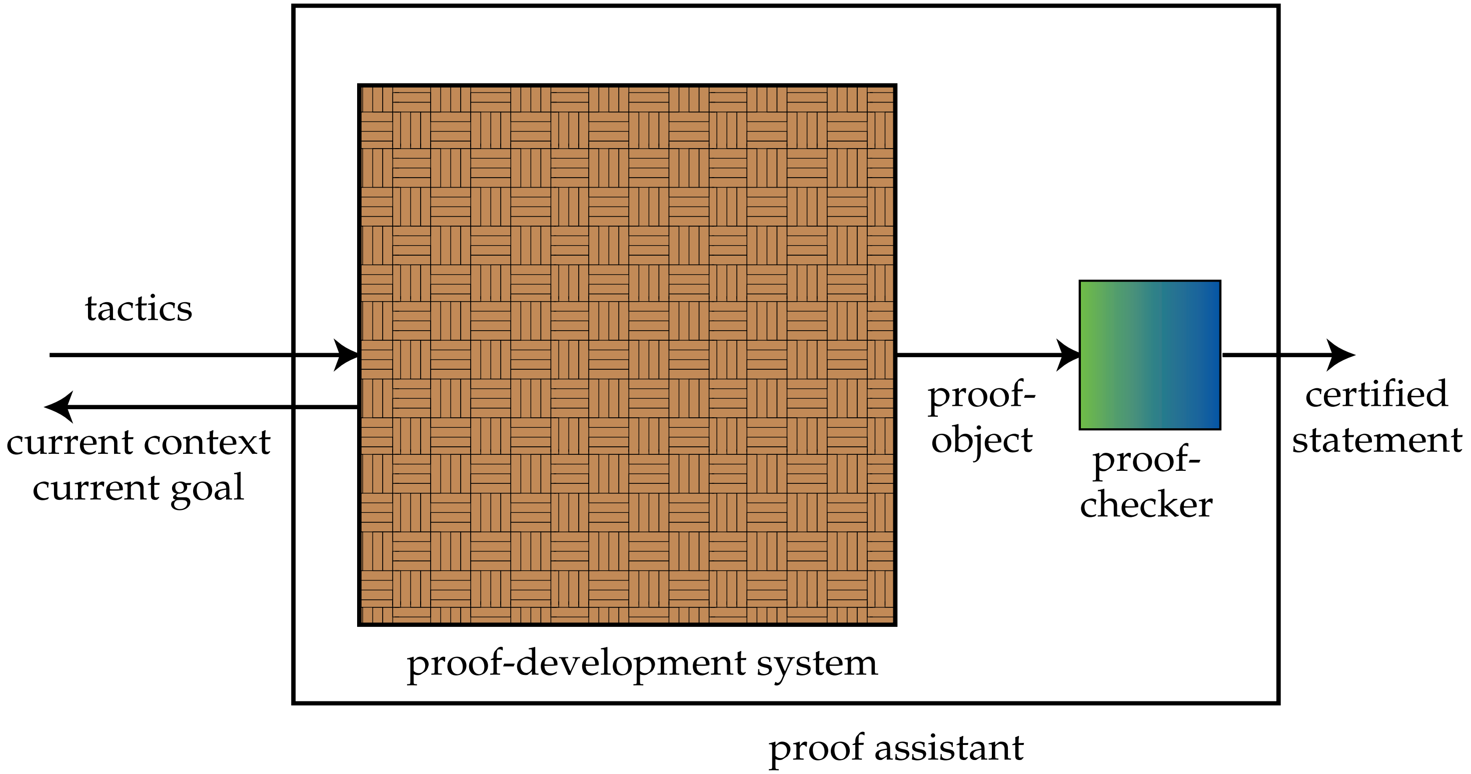
Introduction Rules	Elimination Rules
$\dfrac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash (\lambda x{:}A.M) : (A \to B)}$	$\dfrac{\Gamma \vdash F : (A \to B) \quad \Gamma \vdash p : A}{\Gamma \vdash (Fp) : B}$
$\dfrac{\Gamma \vdash p : A \quad \Gamma \vdash q : B}{\Gamma \vdash \langle p, q \rangle : (A \mathbin{\&} B)}$	$\dfrac{\Gamma \vdash z : (A \mathbin{\&} B)}{\Gamma \vdash z.1 : A} \quad \dfrac{\Gamma \vdash z : (A \mathbin{\&} B)}{\Gamma \vdash z.2 : B}$
$\dfrac{\Gamma \vdash p : A}{\Gamma \vdash (\mathrm{in}_1\ p) : (A \vee B)} \quad \dfrac{\Gamma \vdash p : B}{\Gamma \vdash (\mathrm{in}_2\ p) : (A \vee B)}$	$\dfrac{\Gamma \vdash p : (A \vee B) \quad \Gamma, x{:}A \vdash q : C \quad \Gamma, y{:}B \vdash r : C}{\Gamma \vdash ([\lambda x{:}A.q, \lambda y{:}B.r]p) : C}$
Absurdum Rule	Classical Negation
$\dfrac{\Gamma \vdash p : \bot}{\Gamma \vdash (\mathrm{abs}_A\ p) : A}$	$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A} \quad \neg A {:=} (A \to \bot)$

Classical/Intuitionistic Predicate Logic Natural Deduction Style (Gentzen)
Blue Proofs as λ-terms
Curry-Howard-de Bruijn-Martin-Löf isomorphism between proofs and λ-terms

$$\begin{aligned}
[\![A]\!] &= \{p \mid p \text{ is a proof of } A\} \\
[\![A \to B]\!] &= [\![A]\!] \to [\![B]\!] \\
[\![A \mathbin{\&} B]\!] &= [\![A]\!] \times [\![B]\!] \\
[\![A \vee B]\!] &= [\![A]\!] \cup [\![B]\!] \\
[\![\bot]\!] &= \emptyset
\end{aligned}$$

4. Proof-checking

Assistance



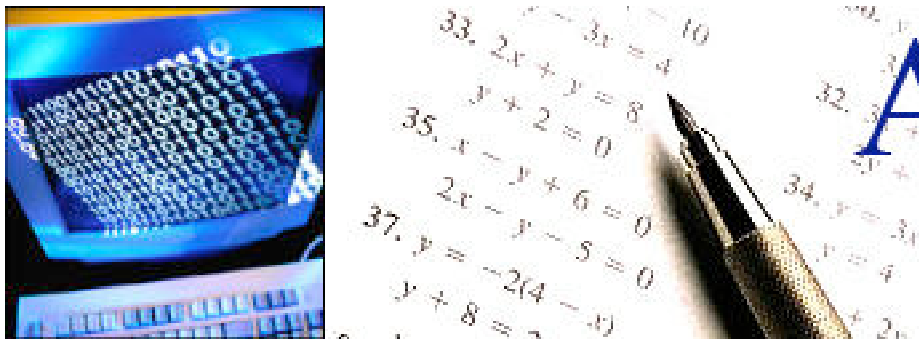
Reliability? The de Bruijn criterion: have a small checker.

Dick de Bruijn

1918

Institute in Nijmegen and the Formal Methods section of Eindhoven University of Technology. Started by prof. H. Barendregt, in cooperation with Rob Nederpelt, this archive project was launched to digitize valuable historical articles and other documentation concerning the Automath project.

Initiated by prof. N.G. de Bruijn, the project Automath (1967 until the early 80's) aimed at designing a language for expressing complete mathematical theories in such a way that a computer can verify the correctness. This project can be seen as the predecessor of type theoretical proof assistants such as the well known Nuprl and Coq.



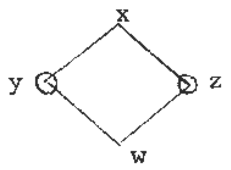
A note on weak diamond properties.

1. Introduction. Let S be a set with a binary relation >. We assume it to satisfy $x > x$ for all $x \in S$. We are interested in establishing a property CR (named after its relevance for the Church-Rosser theorem of lambda calculus, cf. [1]). We say that $x \sim y$ if $x > y$ or $y > x$. We say that $x >^* y$ if there is a finite sequence $x_1, \ldots, x_n$ with $x = x_1 > x_2 > \ldots > x_n = y$, and also if $x = y$. We say that (S,>) satisfies CR if for any sequence $x_1, \ldots, x_n$ with

$$x_1 \sim x_2 \sim \ldots \sim x_n$$

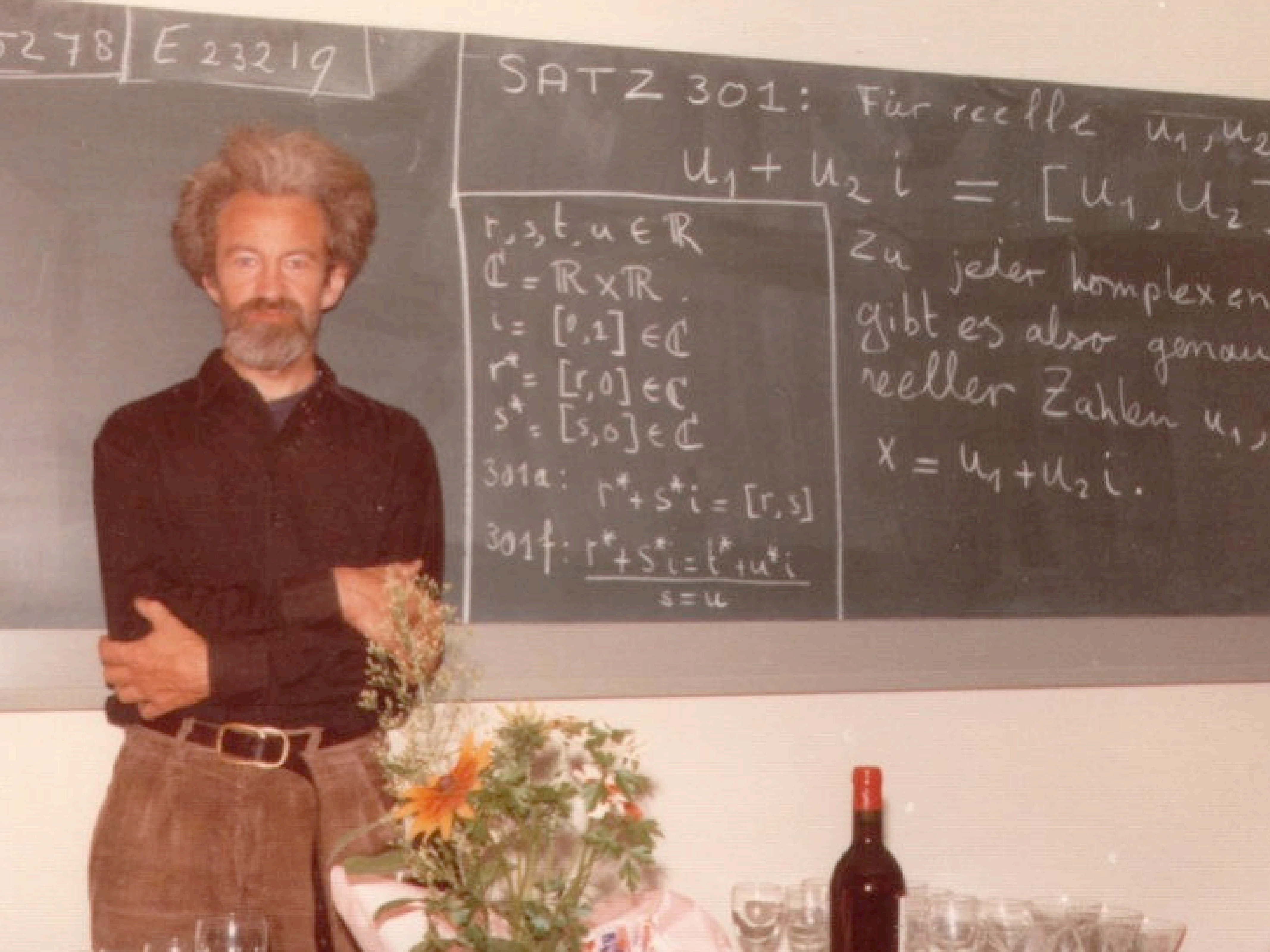
there exist an element $x \in S$ with both $x_1 >^* z$ and $x_n >^* z$.

It is usual to say that (S,>) has the diamond property (DP) if for all x,y,z with $x > y$, $x > z$ there exists a w with $y > w$, $z > w$. This is depicted in the following diagram:



where $x > y$ is indicated by a line from x downwards to y, etc. The little circles around y and z illustrate the logical situation: the diagram $y \overset{x}{\wedge} z$ can be closed by $\underset{w}{y \vee z}$.

It is not hard to show that DP implies CR. A simple way to present a proof is by counting "inversions" in sequences like $x_1 > x_2 < x_3 < x_4 >$



5278 E 23219

SATZ 301: Für reelle u_1, u_2

$u_1 + u_2 i = [u_1, u_2$

Zu jeder komplexen

gibt es also genau

reeller Zahlen $u_1,$

$x = u_1 + u_2 i.$

$r, s, t, u \in \mathbb{R}$

$\mathbb{C} = \mathbb{R} \times \mathbb{R}$

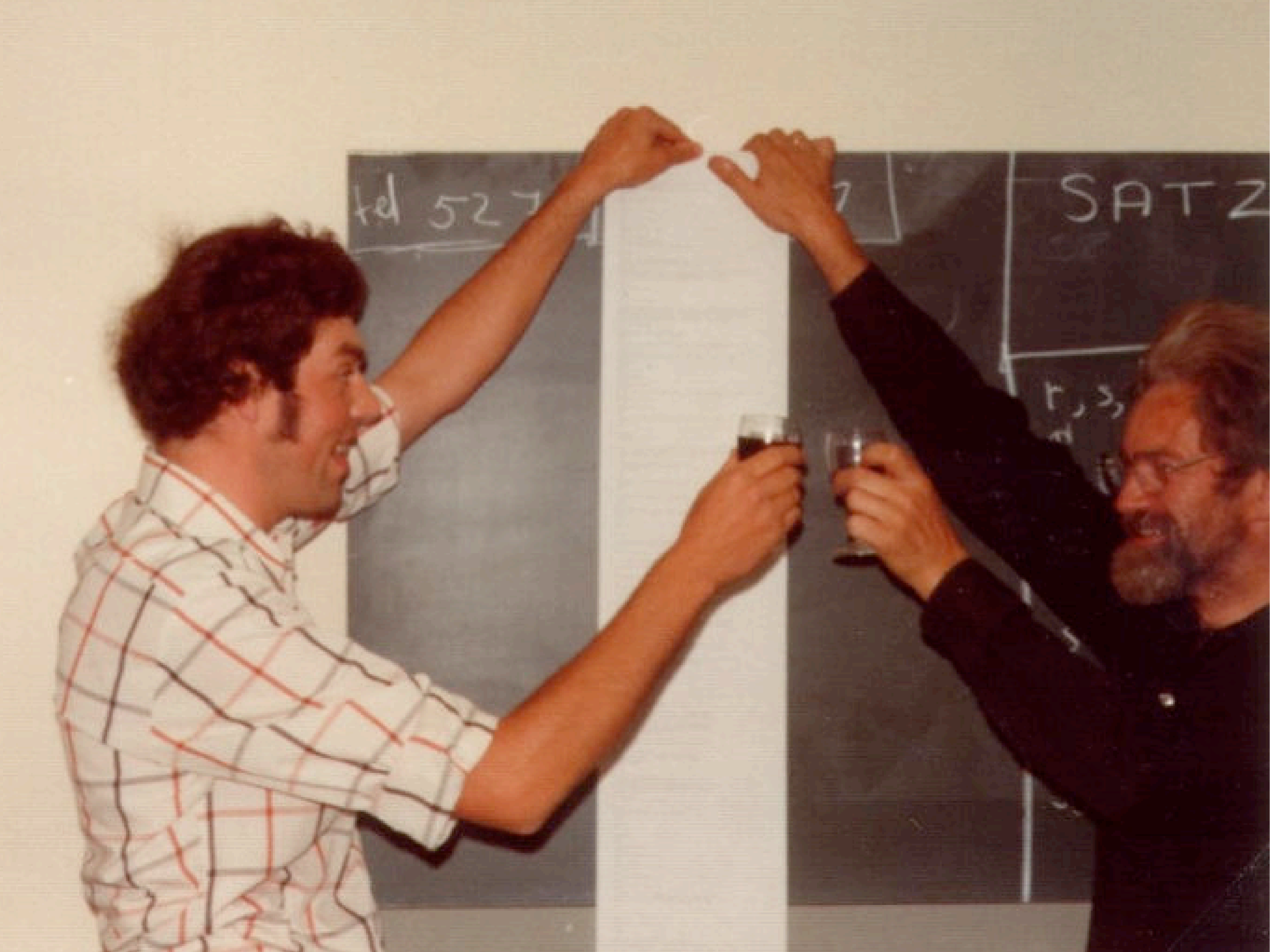
$i = [0, 1] \in \mathbb{C}$

$r^* = [r, 0] \in \mathbb{C}$

$s^* = [s, 0] \in \mathbb{C}$

301a: $r^* + s^* i = [r, s]$

301f: $\dfrac{r^* + s^* i = t^* + u^* i}{s = u}$


fd 527
SATZ
r, s,

L.E.J. (Bertus) Brouwer

1881-1966

intuitionisme

$\cancel{A \vee \neg A}$ *tertium non datur*

$\cancel{\neg\neg A \leftrightarrow A}$

3, 5, 7/8, ... *rationaal*

e, π, $\sqrt{2}$ *irrationaal*

zijn er α en β, irrationaal,

zodat α^β rationaal is?

~~$(\sqrt{2})^{\sqrt{2}}$ is rationaal of irrationaal~~

JA

JA

$\alpha = \beta = \sqrt{2}$:

α^β rationaal

$\alpha = (\sqrt{2})^{\sqrt{2}}$ en $\beta = \sqrt{2}$:

$\alpha^\beta = ((\sqrt{2})^{\sqrt{2}})^{\sqrt{2}} = 2$

NB

Lambda calculus en Combinatorische Logica (CL) zijn niet het einde van het verhaal, de ultieme rekensystemen, ook al zijn ze Turing compleet. Ze zijn inherent sequentieel, en kunnen geen parallelle operaties definieren, zoals parallel or:

por(x,T) = T
por(T,x) = T,
por(F,F) = F

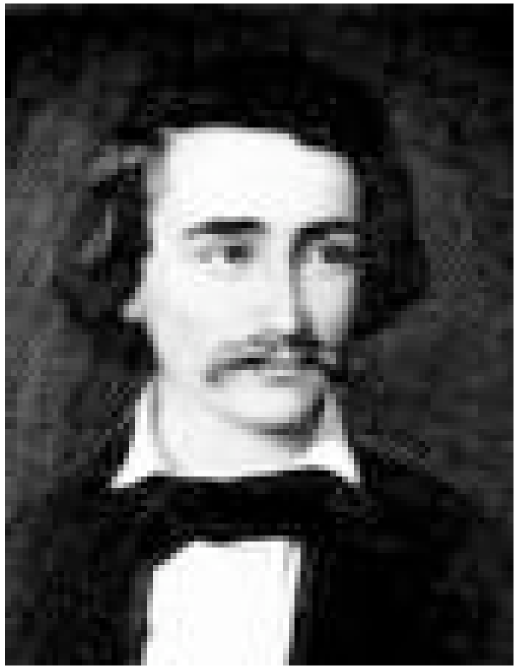
Grassmann 1861, Dedekind 1888

$A(x, 0) \to x$

$A(x, S(y)) \to S(A(x, y))$

$M(x, 0) \to 0$

$M(x, S(y)) \to A(M(x, y), x)$



data

0, S(0), S(S0)),...

$M(S(S(0)), S(S(0))) \rightarrow$

$A(M(S(S(0)), S(0)), S(S(0))) \rightarrow$

$S(\ A(M(S(S(0)), S(0)),S(0))\ ) \rightarrow$

$S(\ S(\ A(M(S(S(0)), S(0)),0)\ )\ ) \rightarrow$

$S(\ S(\ M(S(S(0)), S(0))\ )\ ) \rightarrow$

$S(S(A(M(S(S(0)),0), S(S(0))))) \rightarrow$

$S(S(A(0,S(S(0))))) \rightarrow$

$S(S(S(A(0,S(0))))) \rightarrow$

$S(S(S(S(A(0,0))))) \rightarrow$

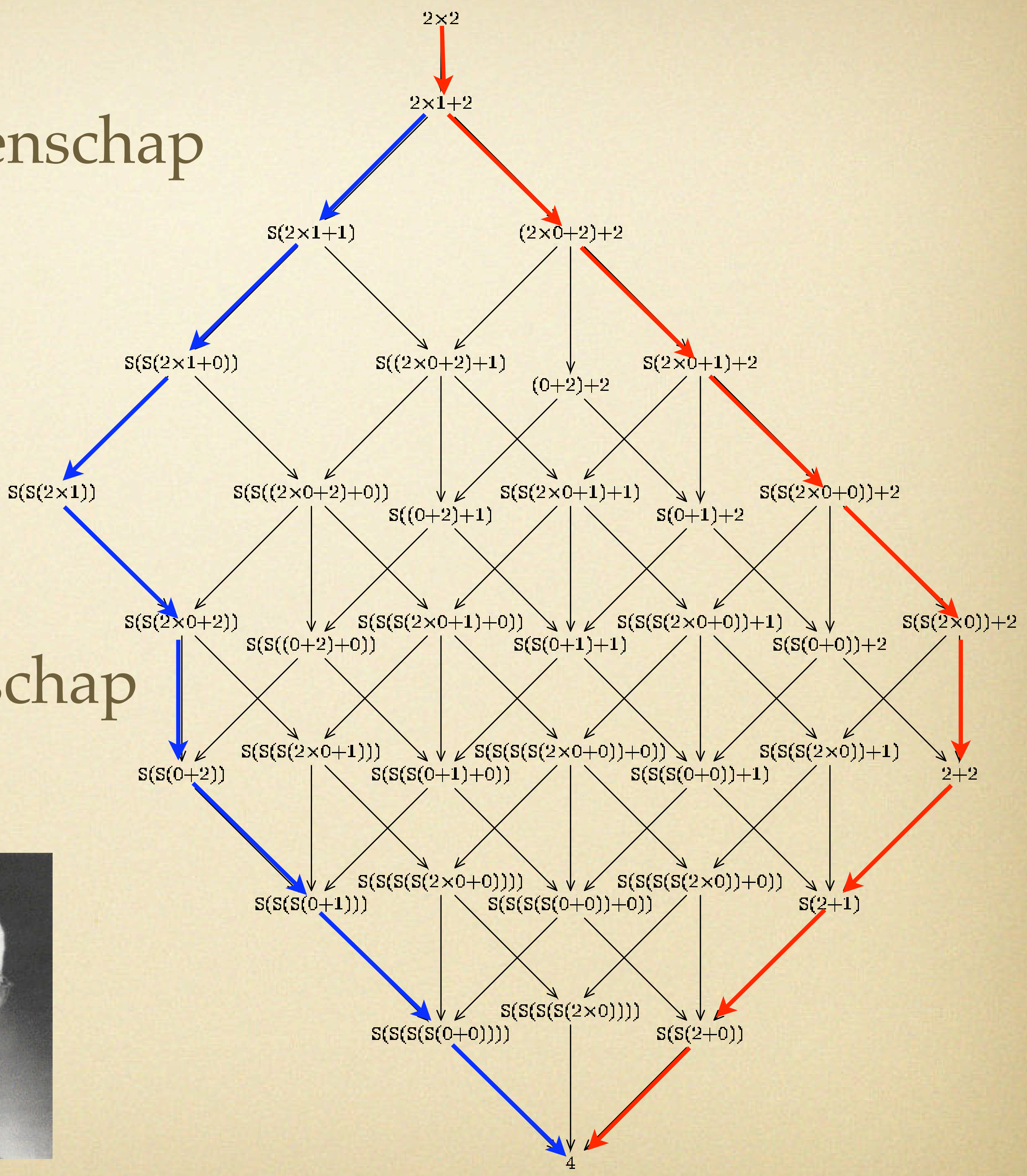
$S(S(S(S(0))))$.

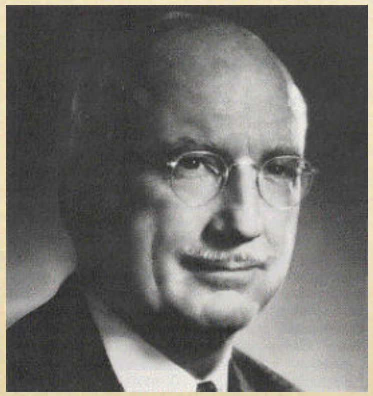
1. two times two

$$\begin{tikzcd}
 & & & 2\times2 \arrow[d] & & & & \\
 & & & 2\times1+2 \arrow[dl] \arrow[dr] & & & & \\
 & & S(2\times1+1) \arrow[dl] \arrow[dr] & & (2\times0+2)+2 \arrow[dl] \arrow[d] \arrow[dr] & & & \\
 & S(S(2\times1+0)) \arrow[dl] \arrow[dr] & & S((2\times0+2)+1) \arrow[dl] \arrow[d] \arrow[dr] & (0+2)+2 \arrow[dl] \arrow[dr] & S(2\times0+1)+2 \arrow[dl] \arrow[d] \arrow[dr] & & \\
S(S(2\times1)) \arrow[dr] & & S(S((2\times0+2)+0)) \arrow[dl] \arrow[d] \arrow[dr] & S((0+2)+1) \arrow[dl] \arrow[dr] & S(S(2\times0+1)+1) \arrow[dl] \arrow[d] \arrow[dr] & S(0+1)+2 \arrow[dl] \arrow[dr] & S(S(2\times0+0))+2 \arrow[dl] \arrow[d] \arrow[dr] & \\
 & S(S(2\times0+2)) \arrow[d] \arrow[dr] & S(S((0+2)+0)) \arrow[dl] \arrow[dr] & S(S(S(2\times0+1)+0)) \arrow[dl] \arrow[d] \arrow[dr] & S(S(0+1)+1) \arrow[dl] \arrow[dr] & S(S(S(2\times0+0))+1) \arrow[dl] \arrow[d] \arrow[dr] & S(S(0+0))+2 \arrow[dl] \arrow[dr] & S(S(2\times0))+2 \arrow[dl] \arrow[d] \\
 & S(S(0+2)) \arrow[dr] & S(S(S(2\times0+1))) \arrow[d] \arrow[dr] & S(S(S(0+1)+0)) \arrow[dl] \arrow[dr] & S(S(S(S(2\times0+0))+0)) \arrow[dl] \arrow[d] \arrow[dr] & S(S(S(0+0))+1) \arrow[dl] \arrow[dr] & S(S(S(2\times0))+1) \arrow[dl] \arrow[d] & 2+2 \arrow[dl] \\
 & & S(S(S(0+1))) \arrow[dr] & S(S(S(S(2\times0+0)))) \arrow[d] \arrow[dr] & S(S(S(S(0+0))+0)) \arrow[dl] \arrow[dr] & S(S(S(S(2\times0))+0)) \arrow[dl] \arrow[d] & S(2+1) \arrow[dl] & \\
 & & & S(S(S(S(0+0)))) \arrow[dr] & S(S(S(S(2\times0)))) \arrow[d] & S(S(2+0)) \arrow[dl] & & \\
 & & & & 4 & & &
\end{tikzcd}$$

CR confluent,
Church-Rosser eigenschap

UN unieke normaalvormeigenschap





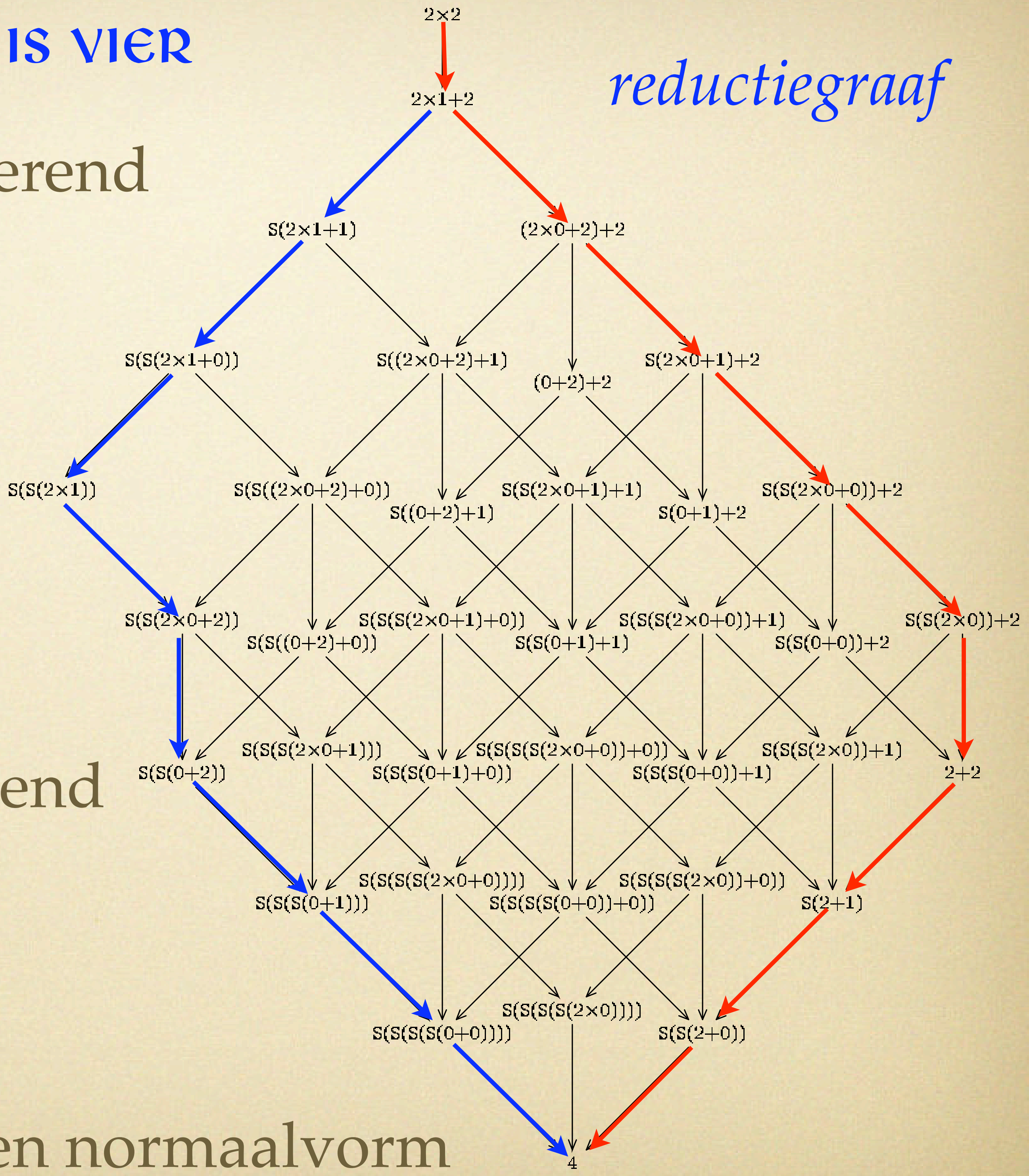
1. twee maal twee is vier

reductiegraaf

WN zwak normaliserend

$\Uparrow$

SN sterk normaliserend



S(S(S(S(0)))), een normaalvorm

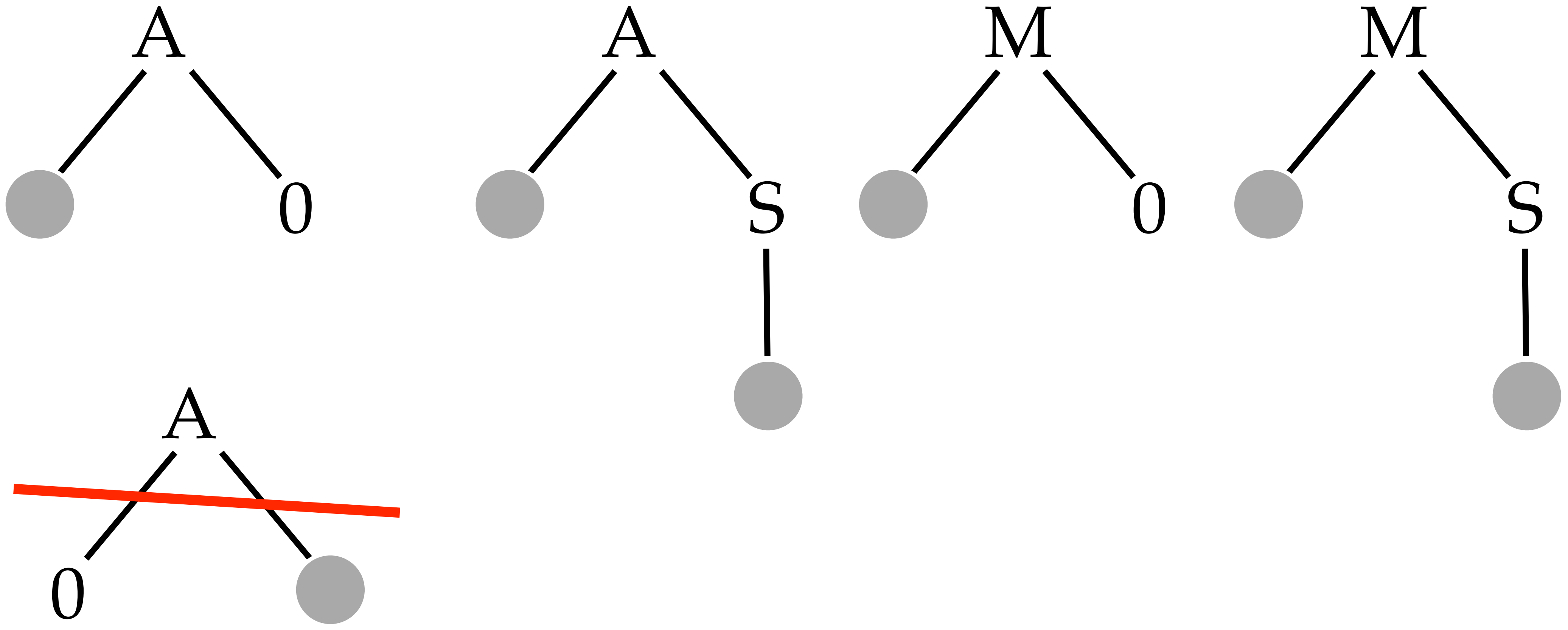
$A(x, 0) \rightarrow x$

$A(x, S(y)) \rightarrow S(A(x, y))$

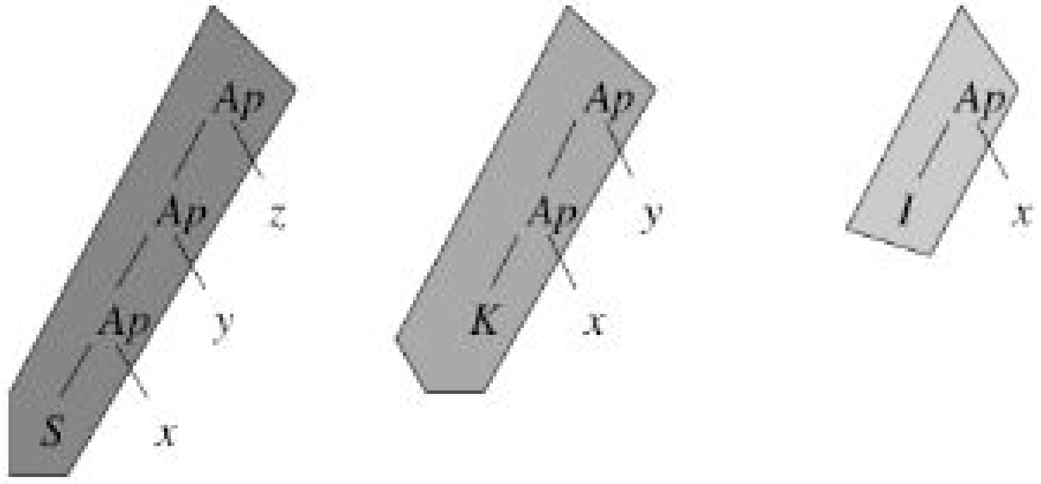
$M(x, 0) \rightarrow 0$

$M(x, S(y)) \rightarrow A(M(x, y), x)$

regels links-lineair



Combinatory Logic



$$Ix \longrightarrow x$$

$$Kxy \longrightarrow x$$

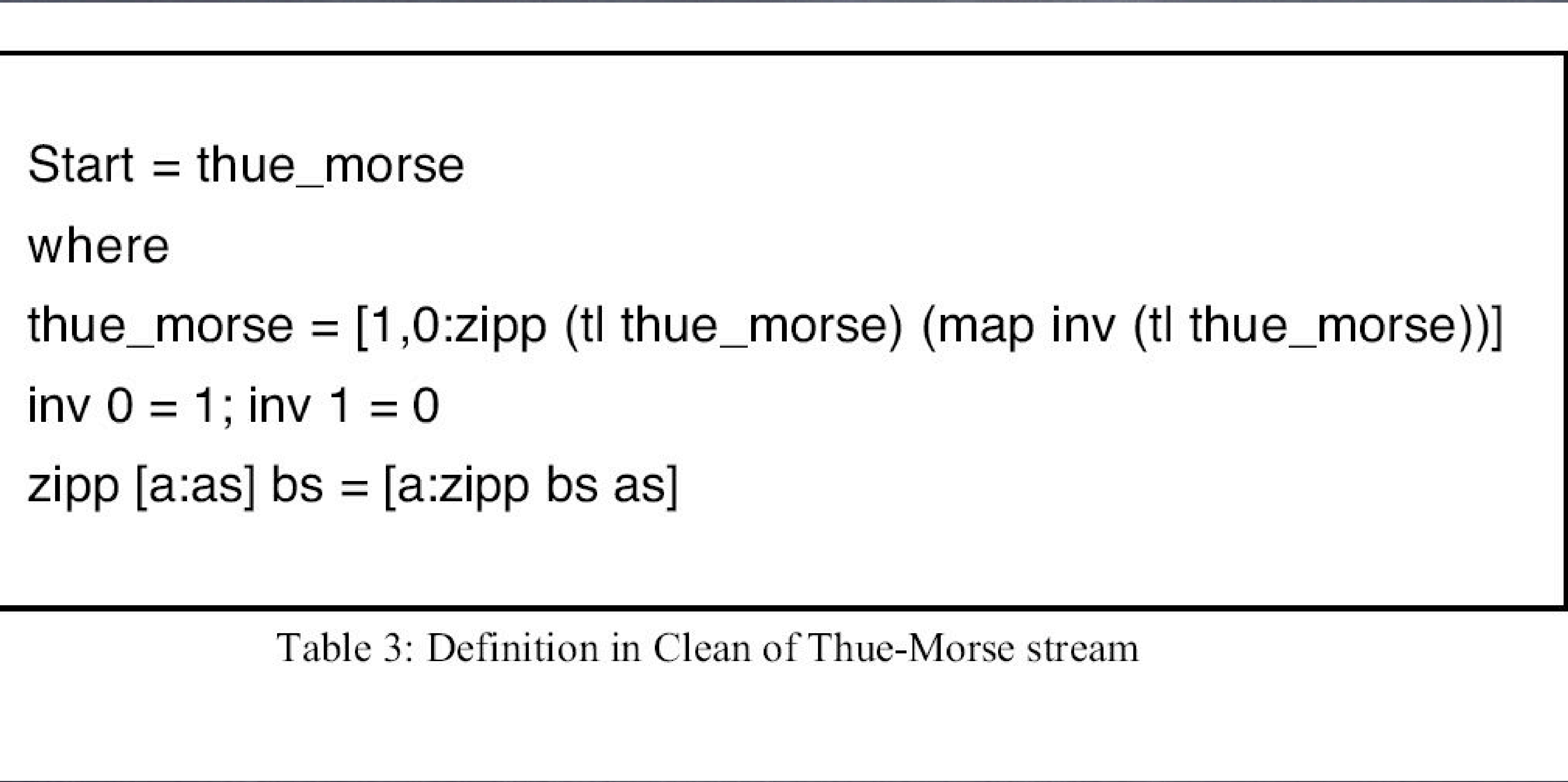
$$Sxyz \longrightarrow xz(yz)$$

orthogonaal, dus confluent

$$\begin{aligned}
&\mathit{ones} \to 1 : \mathit{ones}\\
&\mathit{zeros} \to 0 : \mathit{zeros}\\
&\mathit{alt} \to 0 : 1 : \mathit{alt}\\
&\mathit{zip}(n{:}x,\ y) \to n : \mathit{zip}(y,\ x)
\end{aligned}$$

zip(zeros, *ones)* and *alt* rewrite in infinitely many steps to the same infinite normal form

$$
\begin{aligned}
\mathit{filter}(x : y, 0, m) &\rightarrow 0 : \mathit{filter}(y, m, m) \\
\mathit{filter}(x : y, s(n), m) &\rightarrow x : \mathit{filter}(y, n, m) \\
\mathit{sieve}(0 : y) &\rightarrow \mathit{sieve}(y) \\
\mathit{sieve}(s(n) : y) &\rightarrow s(n) : \mathit{sieve}(\mathit{filter}(y, n, n)) \\
\mathit{nats}(n) &\rightarrow n : \mathit{nats}(s(n)) \\
\mathit{primes} &\rightarrow \mathit{sieve}(\mathit{nats}(s(s(0))))
\end{aligned}
$$



```
Start = thue_morse
where
thue_morse = [1,0:zipp (tl thue_morse) (map inv (tl thue_morse))]
inv 0 = 1; inv 1 = 0
zipp [a:as] bs = [a:zipp bs as]
```

Table 3: Definition in Clean of Thue-Morse stream

4.1 Proof-checking: state of the art

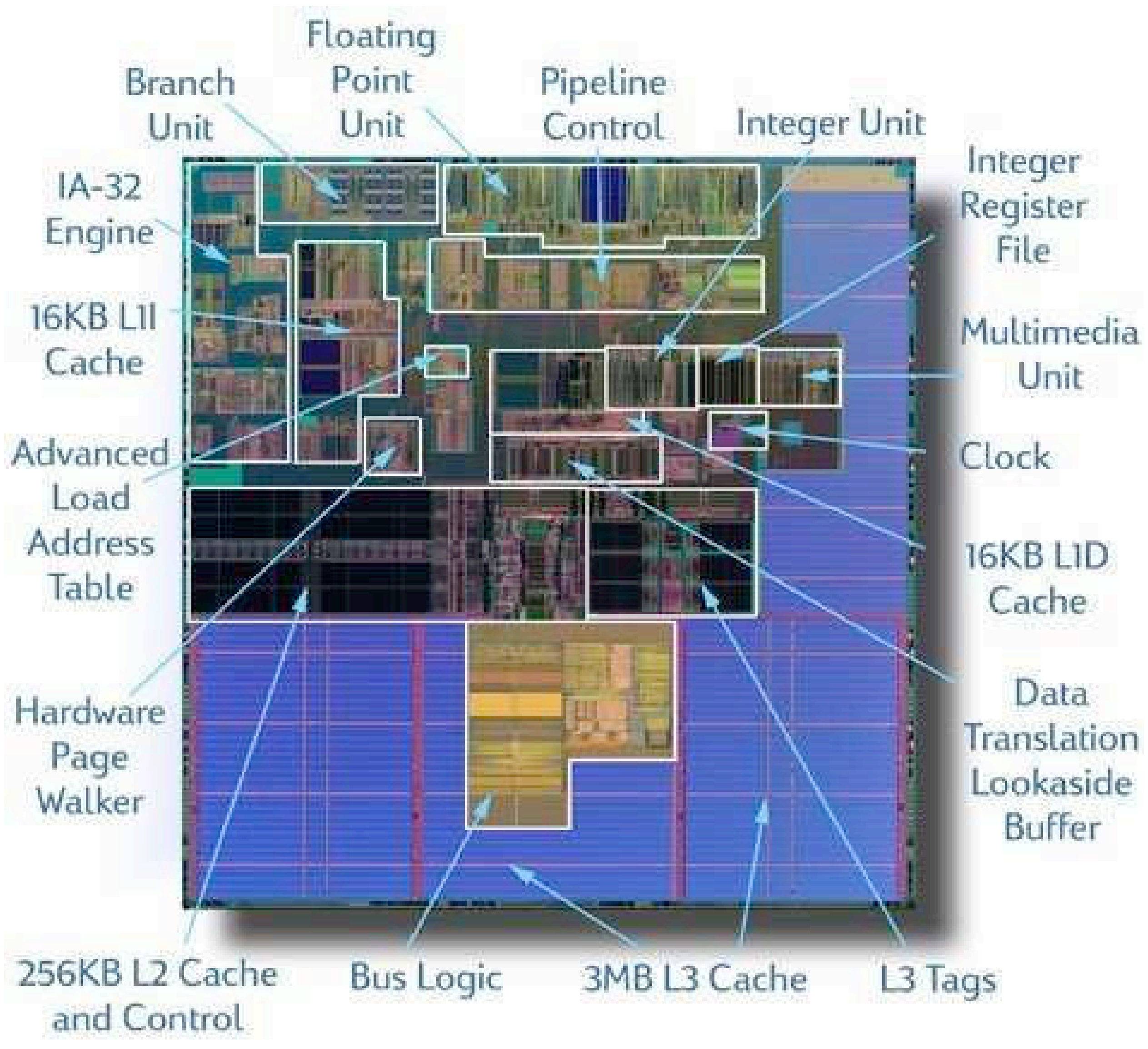
Mathematical developments

Fundamental Theorem of Algebra	Geuvers, Wiedijk, Zwanenburg, Pollack and Niqui	Coq
Fundamental Teorem of Calculus	Cruz-Filipe	Coq
Correctness Buchberger's algorithm	Théry	Coq
Primality of $9026258083384996860449366072142307801963$	Oostdijk, Caprotti	Coq
Correctness of Fast Fourier Transform	Capretta	Coq
Book "Continuous lattices"	?	Mizar
Quadratic Reciprocity	Harrison	Hol-light
Prime Number Theorem	Avigad	Isabelle-Hol
Four Colour Theorem	Gonthier	Coq
Jordan Curve Theorem	Hales	Hol
Primality of >100 digit numbers	Grégoire, Théry, Werner	Coq

Applications

Protocol verification for embedded software

4.2 Industrial application

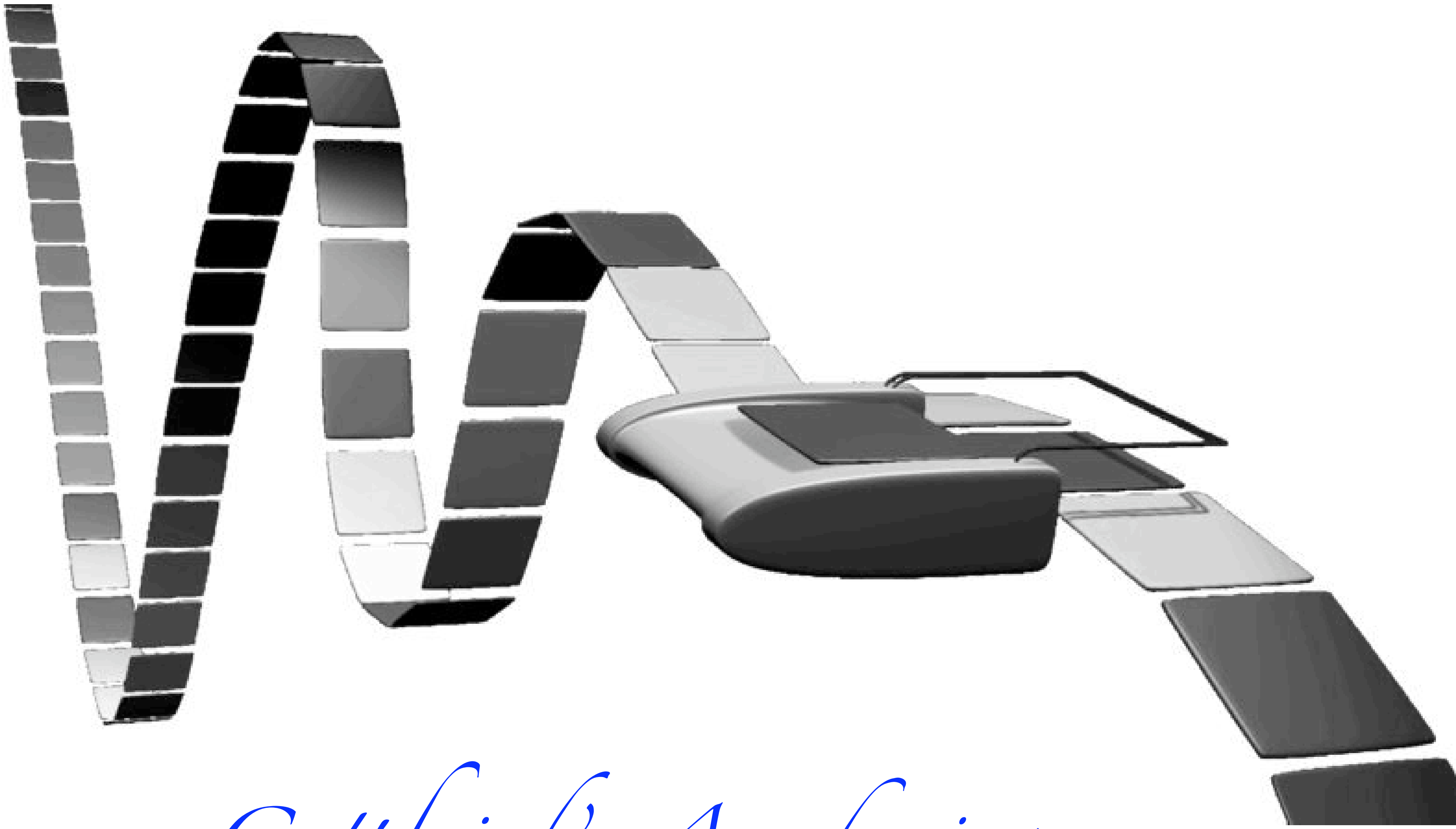


Intel® Itanium® 2 microprocessor

In de voordracht zal niet de nadruk komen te liggen op deze negatieve resultaten, maar meer op de positieve aspecten van de reflectieve kennis over berekeningen en over bewijzen. Klop zal spreken over het rekenen en Barendregt over het bewijzen.

summing up: positief of negatief?

- binaire getallen
- propositielogica
- predicatenlogica
- universele abstracte computer
- echte computer
- prolog
- miranda, lisp, scheme, haskell
- proof checkers
- geverifieerde software
- ...



Gottfried's Awakening