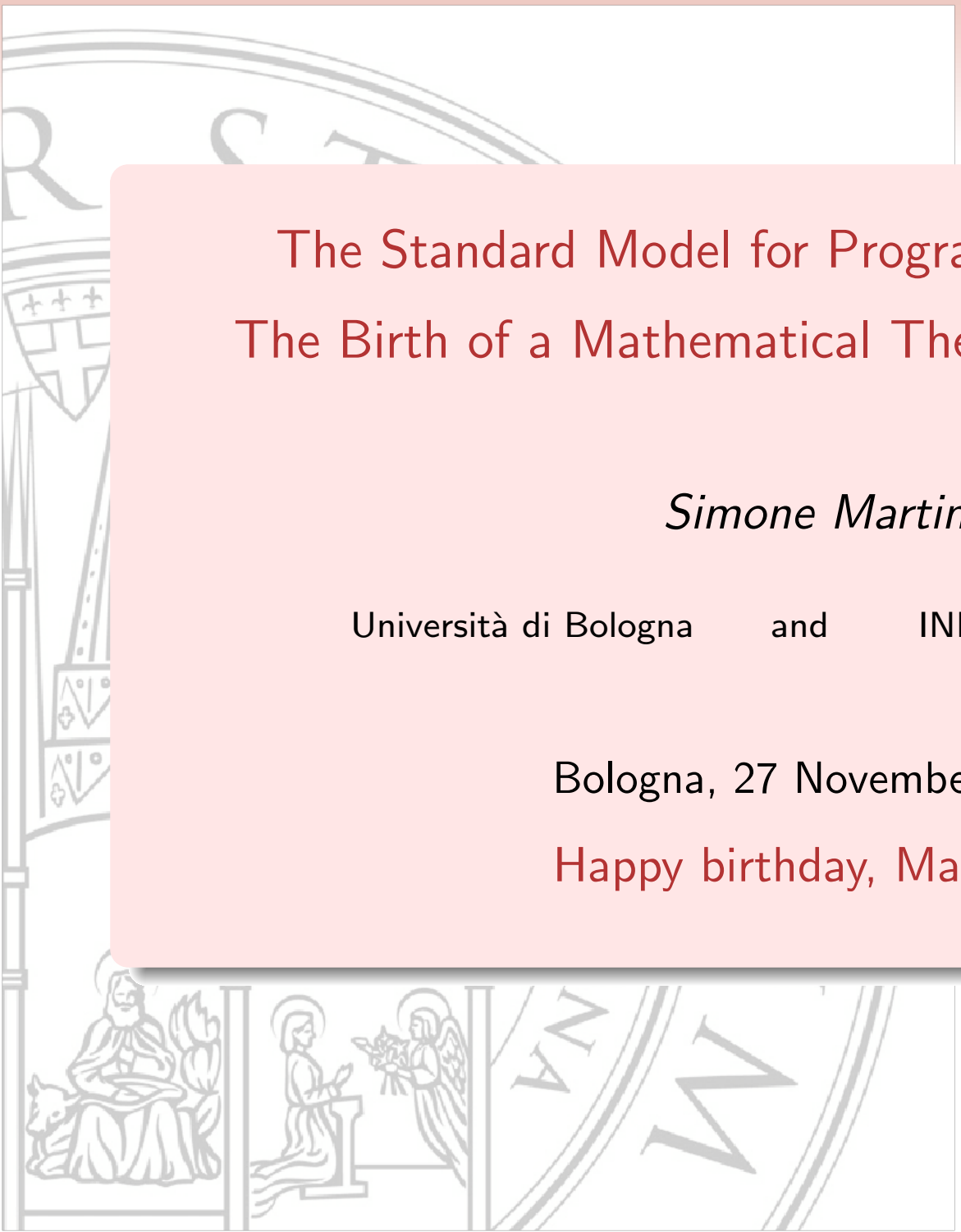


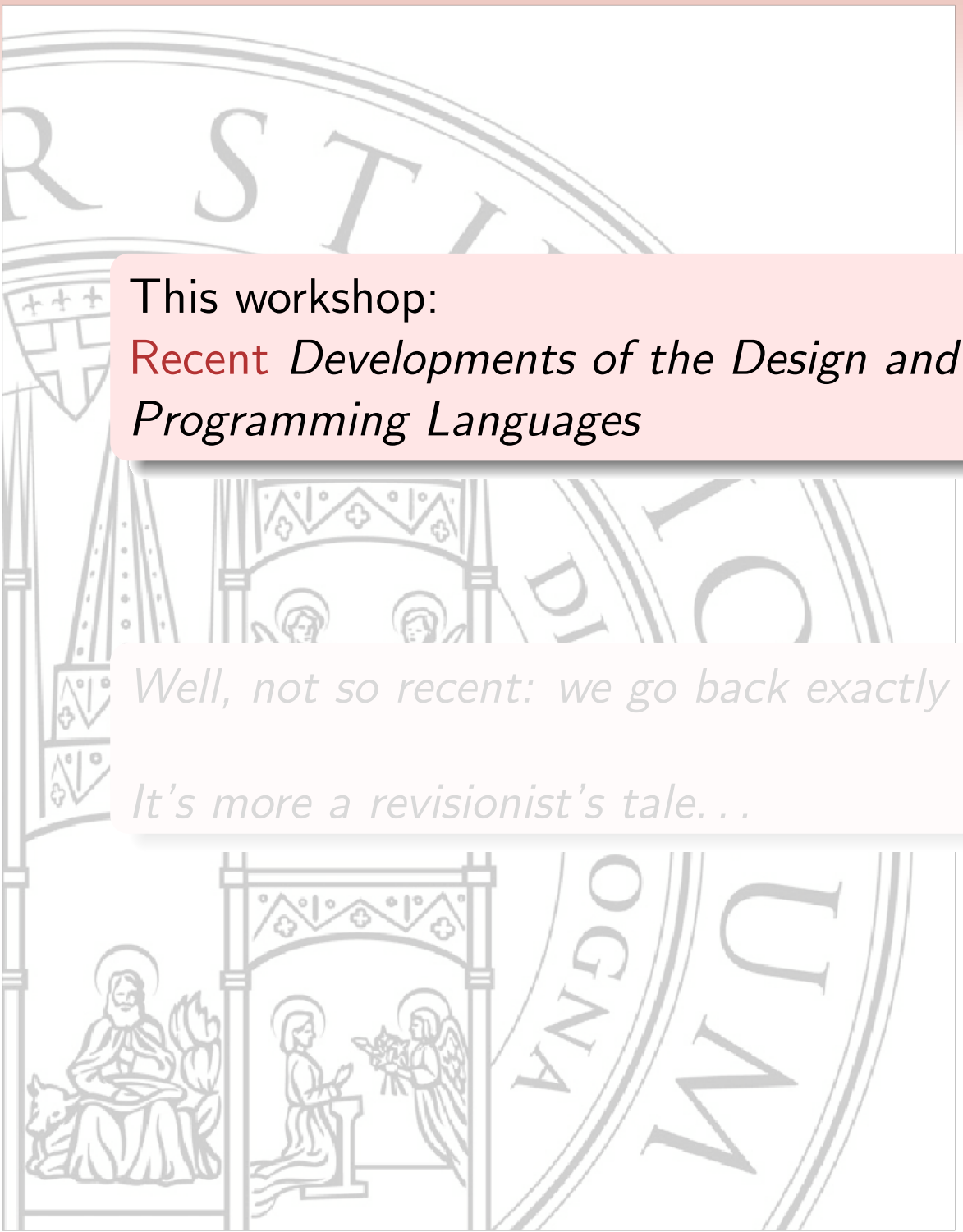
# The Standard Model for Programming Languages: The Birth of a Mathematical Theory of Computation

*Simone Martini*

Università di Bologna and INRIA Sophia-Antipolis

Bologna, 27 November 2020

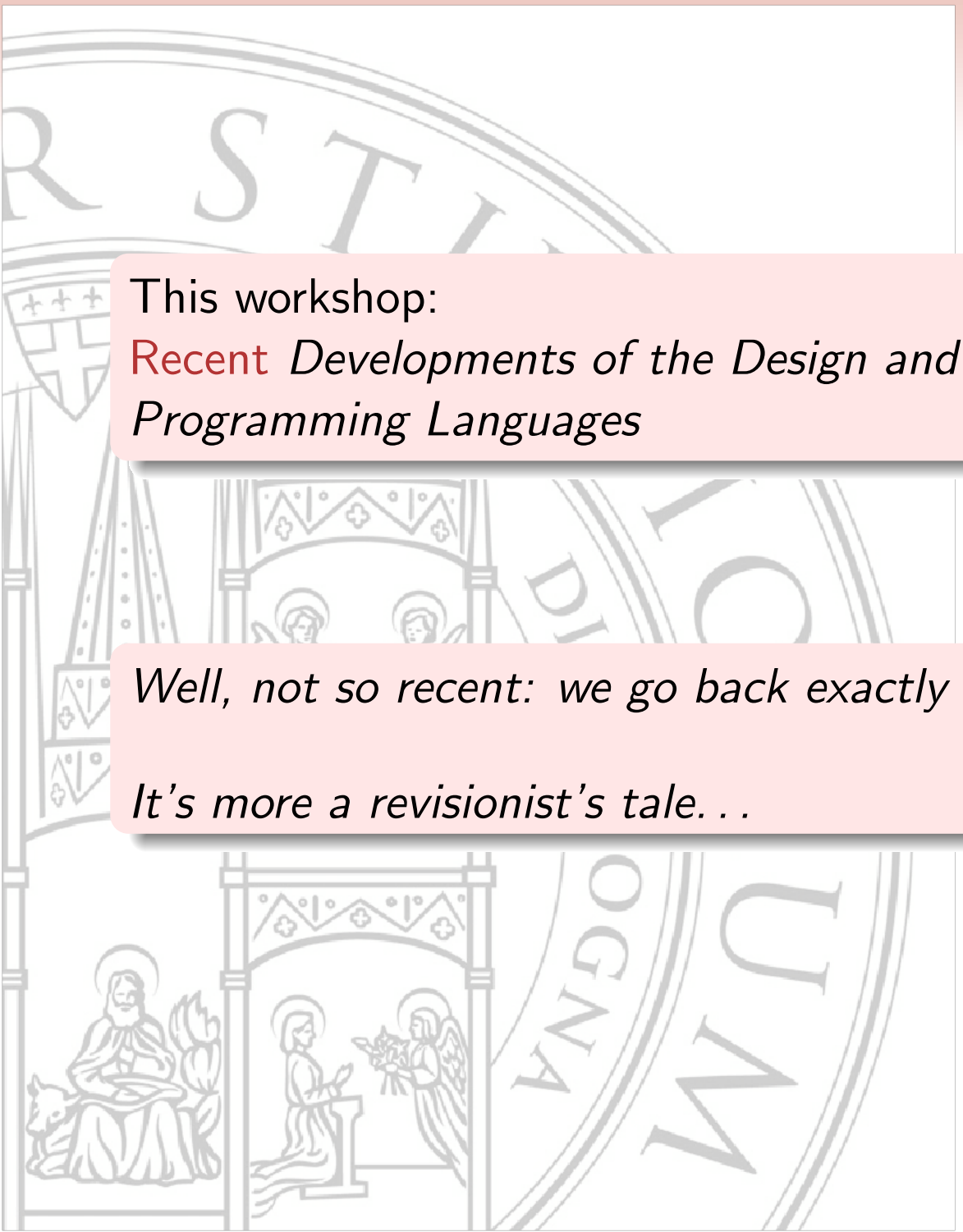
Happy birthday, Maurizio!



This workshop:
Recent *Developments of the Design and Implementation of Programming Languages*

*Well, not so recent: we go back exactly 60 years!*

*It's more a revisionist's tale...*



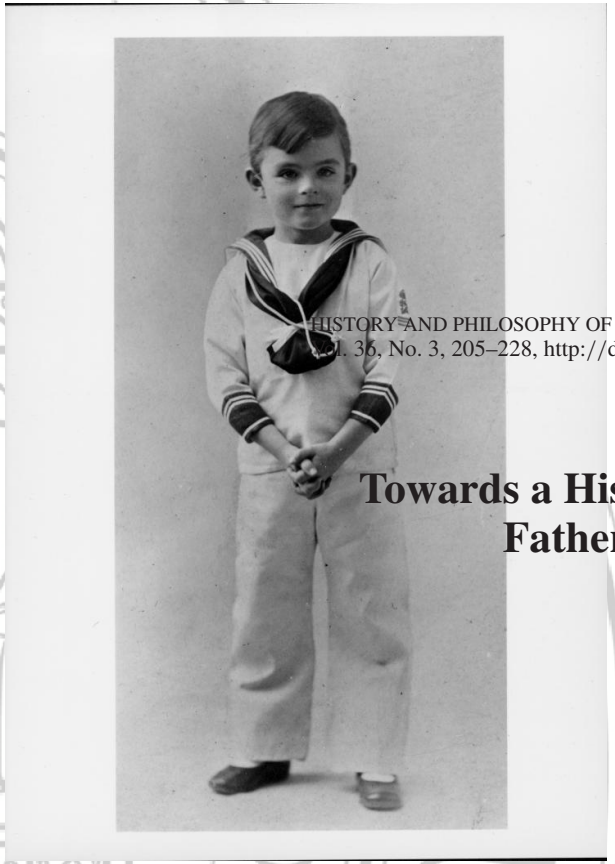
This workshop:
Recent *Developments of the Design and Implementation of Programming Languages*

*Well, not so recent: we go back exactly 60 years!*

*It's more a revisionist's tale...*

# Is Alan Turing the *father* of computer science?

# Is Alan Turing the *father* of computer science?



HISTORY AND PHILOSOPHY OF LOGIC, 2015
Vol. 36, No. 3, 205–228, http://dx.doi.org/10.1080/01445340.2015.1082050

Taylor & Francis
Taylor & Francis Group

**Towards a Historical Notion of 'Turing—the Father of Computer Science'**

# Is Alan Turing the *father* of computer science?

HISTORY
L. 36, No.
OF LOGIC, 2015
p://dx.doi.org/10.1080/01445340.2015.1082050
Taylor & Francis
Taylor & Francis Group

**T wards a Historical Notion of 'Turing—the Father of Computer Science'**

Viewpoints
DOI:10.1145/2629499

Edgar G. Daylight

Viewpoint

**A Turing Tale**

*Assessing the accuracy of popular descriptions of Alan Turing's influences and legacy.*

# Is Alan Turing the *father* of computer science?

HISTORY
L. 36, No.
OF LOGIC, 2015
p://dx.doi.org/10.1080/01445340.2015.1082050
Taylor & Francis
Taylor & Francis Group

**T wards a Historical Notion of 'T ring—the Father of Computer Scie e'**

Edgar G. Daylight

viewpoints

DOI:10.1145/2629499

**Viewpoint**

**A Turing Tale**

*Assessing the accuracy of popular descriptions of Alan Turing's influences and legacy.*

viewpoints

DOI:10.1145/2658985

**Viewpoint**

**Why Did Computer Science Make a Hero Out of Turing?**

*Comparing the legacy of Alan Turing in computer scien with that of Carl Friedrich Gauss in mathematics.*

Maarten Bullynck, Edgar

# Is Alan Turing the *father* of computer science?

viewpoints

DOI:10.1145/2542504 Thomas Haigh

**Historical Reflections**

**Actually, Turing Did Not Invent the Computer**

*Separating the origins of computer science and technology.*

viewpoints

DOI:10.1145/2629499 Edgar G. Daylight

**Viewpoint**

**A Turing Tale**

*Assessing the accuracy of popular descriptions of Alan Turing's influences and legacy.*

HISTORY OF LOGIC, 2015

http://dx.doi.org/10.1080/01445340.2015.1082050

Taylor & Francis
Taylor & Francis Group

**Towards a Historical Notion of 'Turing—the Father of Computer Science'**

viewpoints

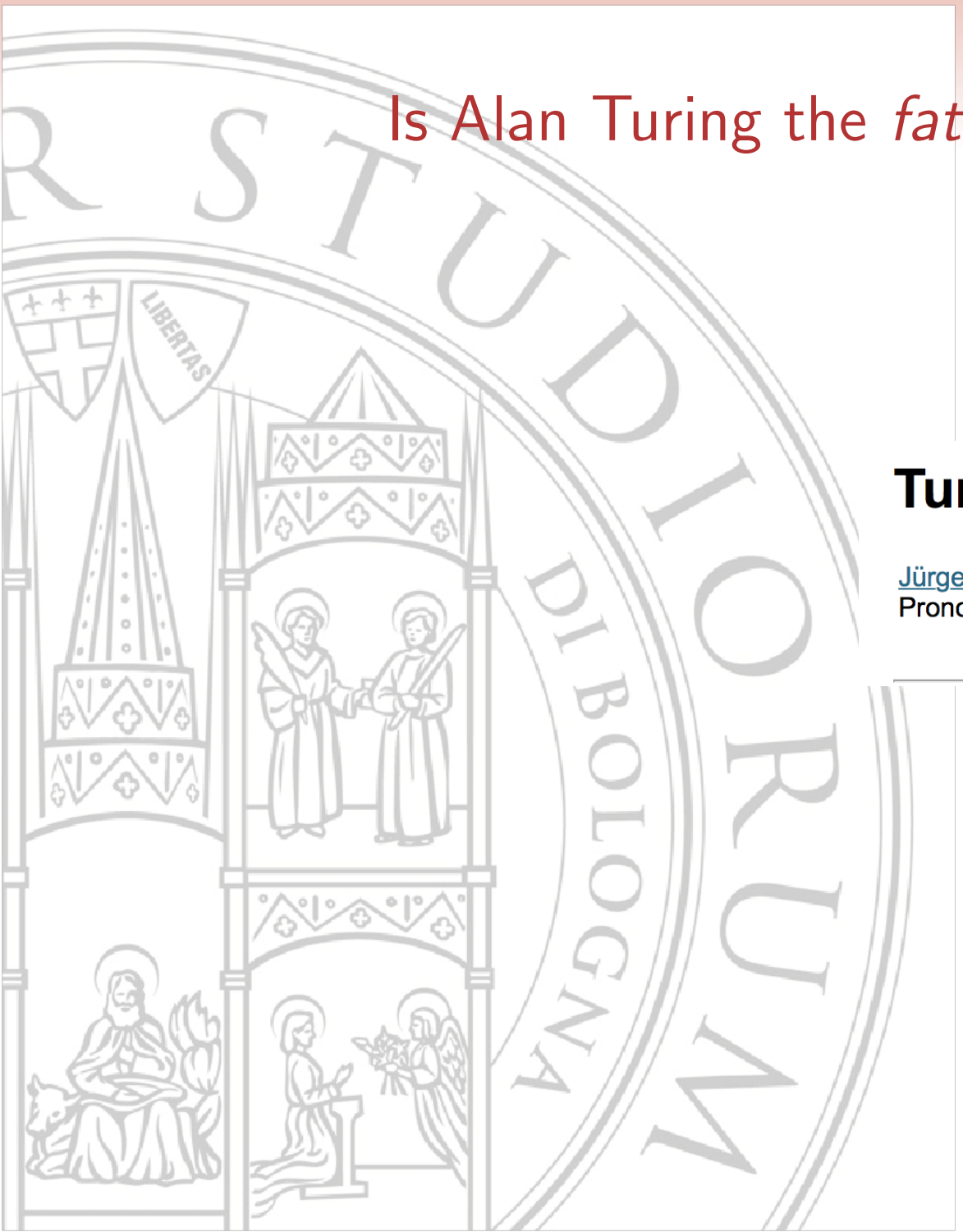
DOI:10.1145/2658985 Maarten Bullynck, Edgar

**Viewpoint**

**Why Did Computer Science Make a Hero Out of Turing?**

*Comparing the legacy of Alan Turing in computer scien with that of Carl Friedrich Gauss in mathematics*

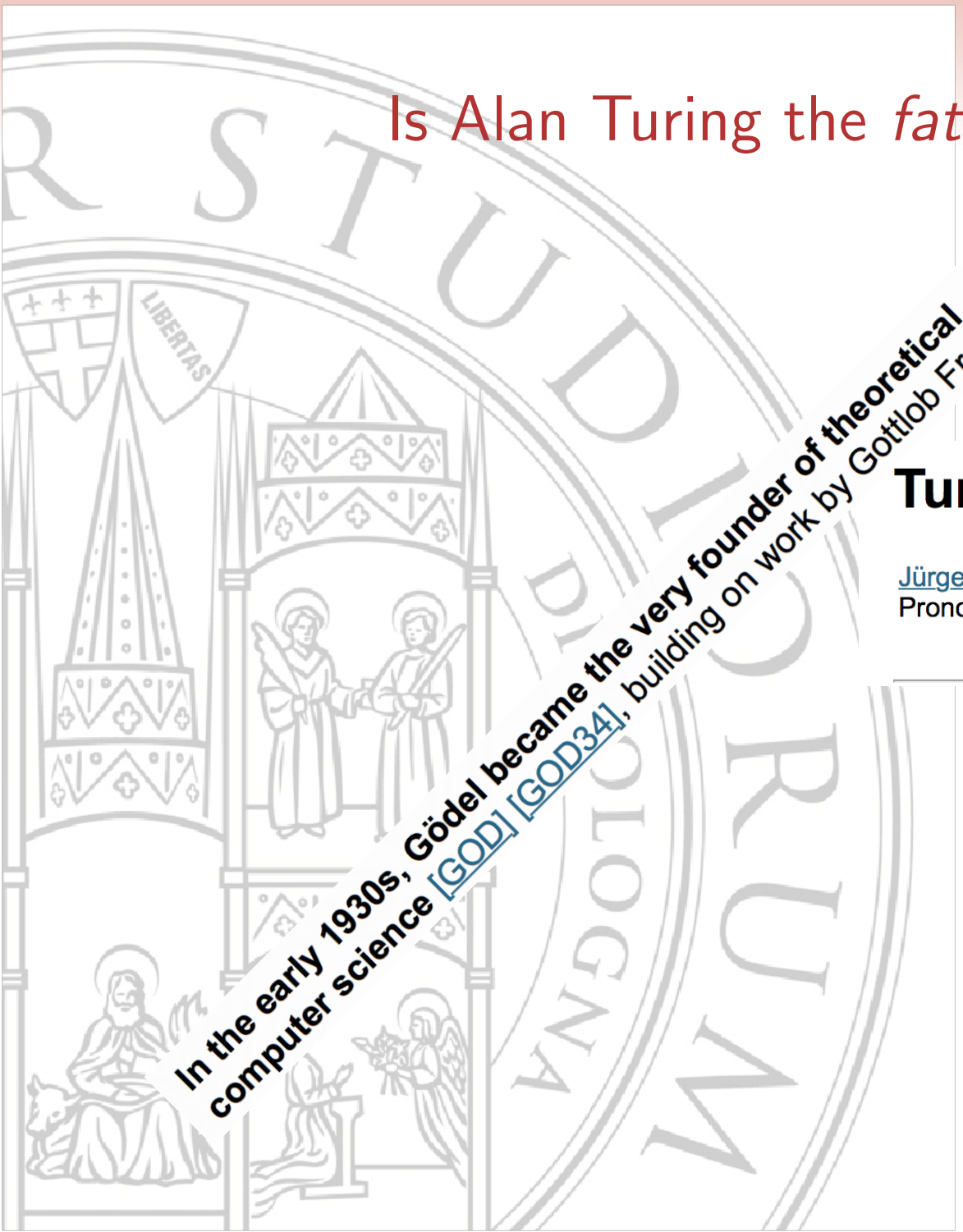
# Is Alan Turing the *father* of computer science?




**Turing Oversold**

Jürgen Schmidhuber (November 2020)
Pronounce: You_again Shmidhoobuh

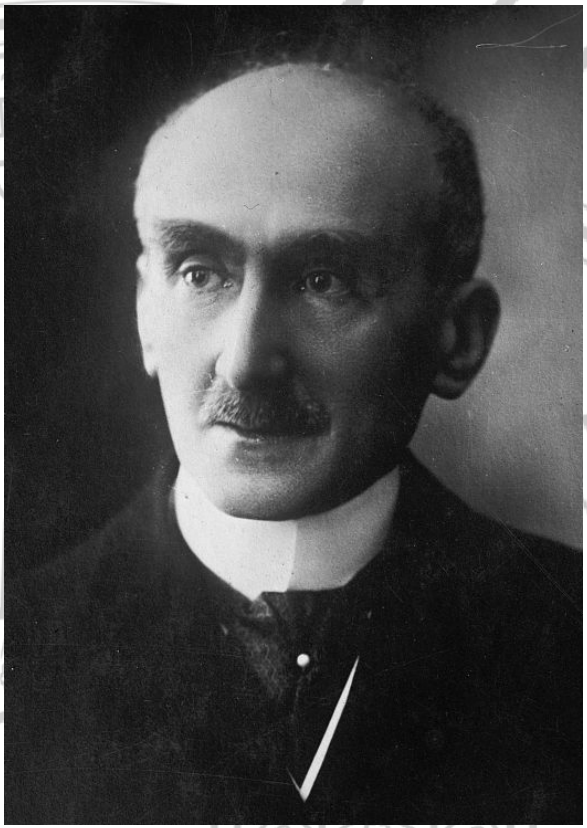
# Is Alan Turing the *father* of computer science?



**In the early 1930s, Gödel became the very founder of theoretical computer science** [GOD] [GOD34], building on work by Gottlob Frege

## Turing Oversold

Jürgen Schmidhuber (November 2020)
Pronounce: You_again Shmidhoobuh



## H. Bergson

the retrospective illusion of truth
*l'illusion rétrospective du vrai*

Par le seul fait de s'accomplir, la réalité projette derrière elle son ombre dans le passé indéfiniment lointain ; elle paraît ainsi avoir préexisté, sous forme de possible, à sa propre réalisation

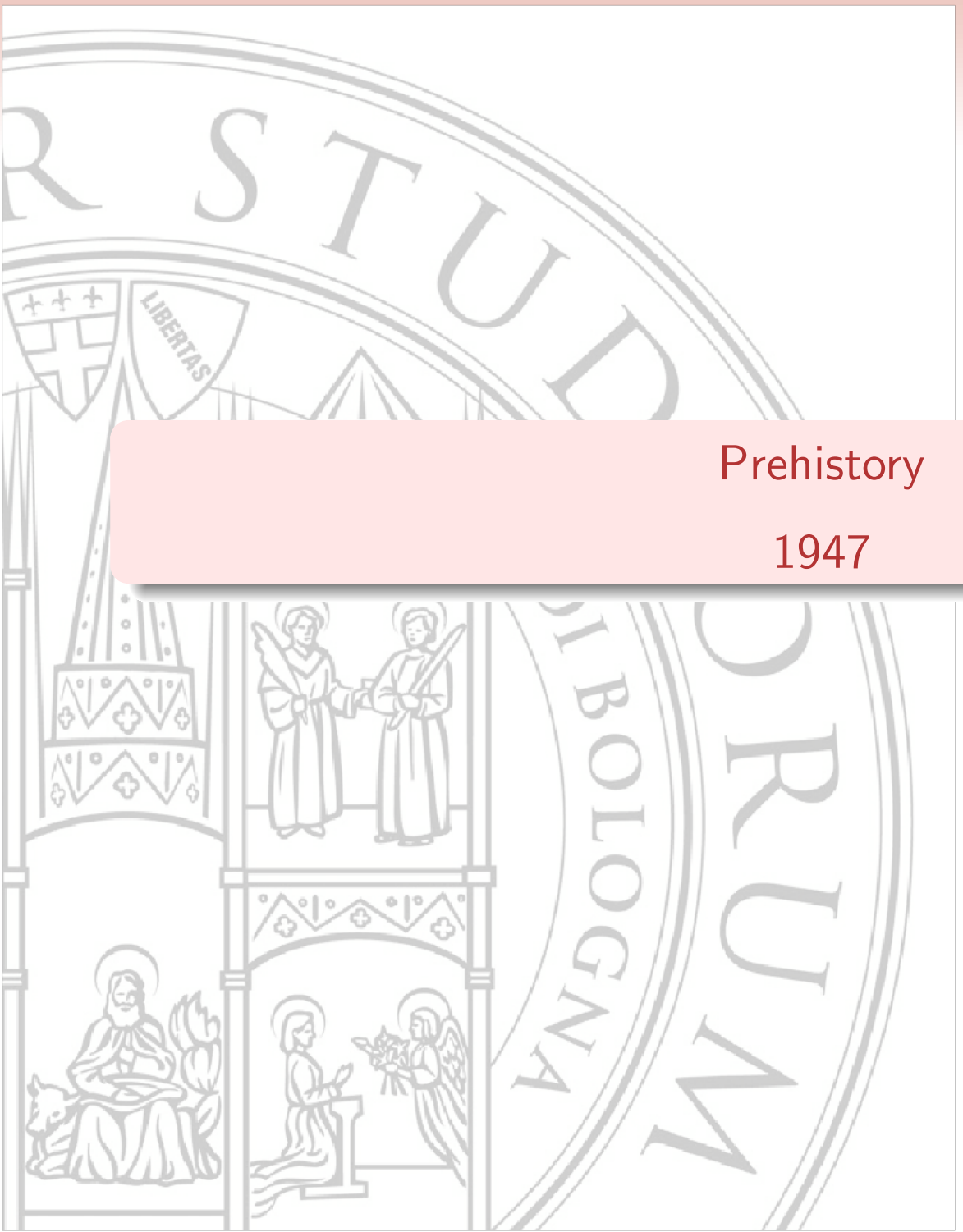
[H. Bergson, La pensée et le mouvant, 1934]

also: le mouvement rétrospectif/rétrograde du vrai

# Turing the father of CS?

- Little influence on actual computers
  EDVAC, and not Turing's ACE is the ancestor of Manchester Mark I
- The mathematical theory of computation
  is the result of an agenda of the late 50s

- Of course someone knew. . .
  von Neumann, Goldstine, Curry, Bernays, Gorn, . . .




## Prehistory

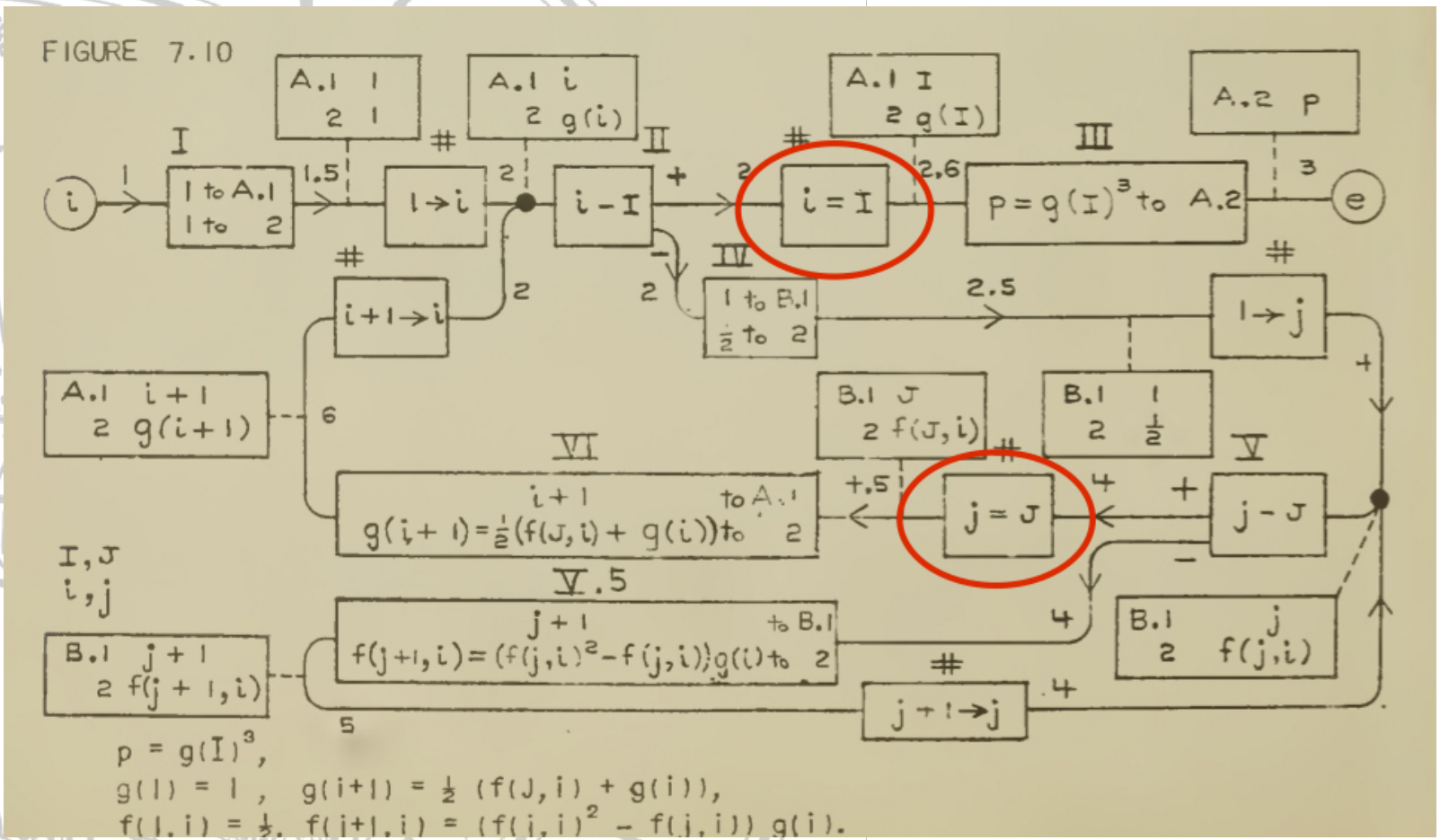
## 1947

# Goldstine and von Neumann

*[. . . ] coding [. . . ] has to be viewed as a logical problem and one that represents a new branch of formal logics.*

Hermann Goldstine and John von Neumann
Planning and Coding of problems for an Electronic Computing Instrument
Report on the mathematical and logical aspects of an electronic computing instrument,
Part II, Volume 1-3, April 1947. Institute of Advanced Studies.

# Boxes in flow diagrams



# Goldstine and von Neumann, 2

## Boxes in flow diagrams

- operation boxes
- substitution boxes
- assertion boxes

*The contents of an assertion box are one or more relations.*

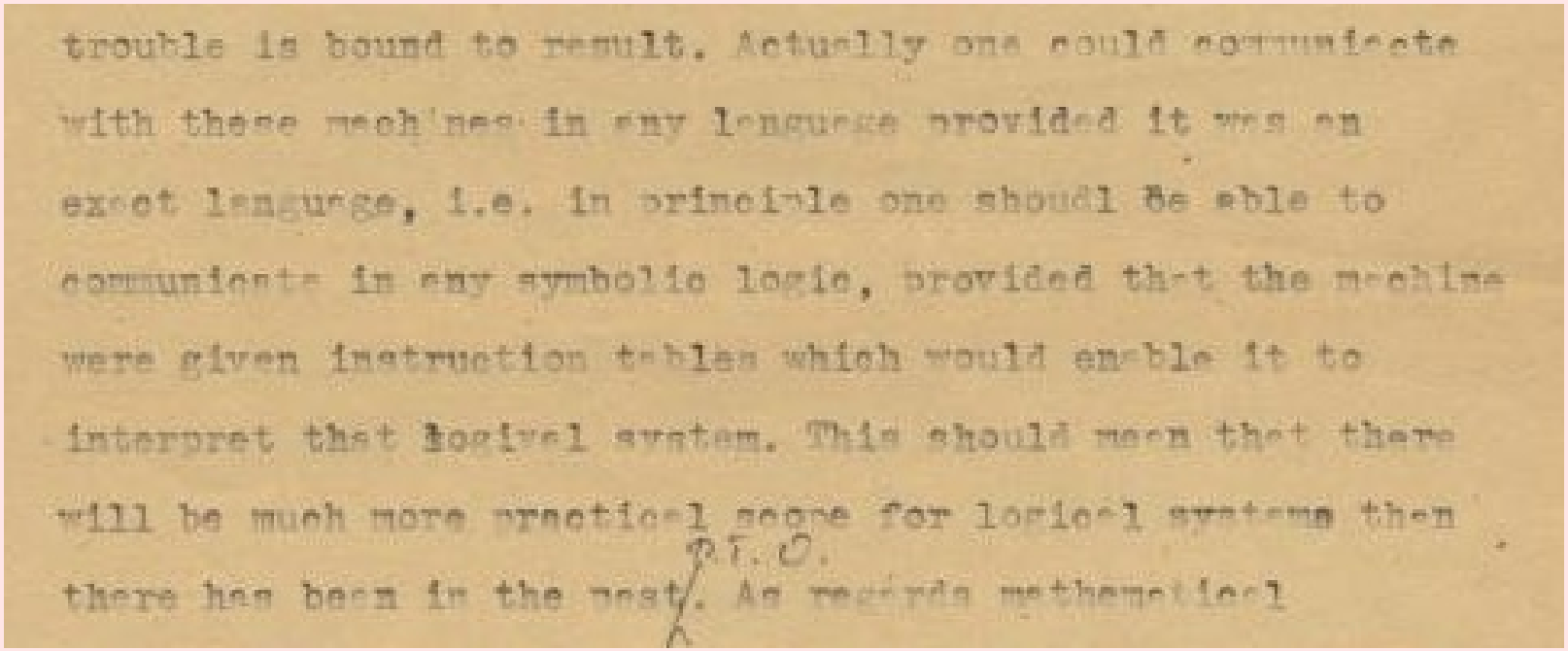
*An assertion box [. . . ] indicates only that certain relations are automatically fulfilled whenever* [the control reaches that point]

Free and bound variables, etc.

## Lecture on Automatic Computing Engine

London Mathematical Soc., 20 Feb 1947. Typewritten notes, in Turing Archive, AMT/C/32

## High-level languages



trouble is bound to result. Actually one could communicate
with these machines in any language provided it was an
exact language, i.e. in principle one should be able to
communicate in any symbolic logic, provided that the machine
were given instruction tables which would enable it to
interpret that logical system. This should mean that there
will be much more practical scope for logical systems than
there has been in the past. As regards mathematical

P.T.O.

# Turing

## Lecture on Automatic Computing Engine

London Mathematical Soc., 20 Feb 1947. Typewritten notes, in Turing Archive, AMT/C/32

## High-level languages

In principle one should be able to communicate [with these machines] in any symbolic logic [...].

This would mean that there will be much more practical scope for logical systems than there has been in the past.

# Turing, again: 1949

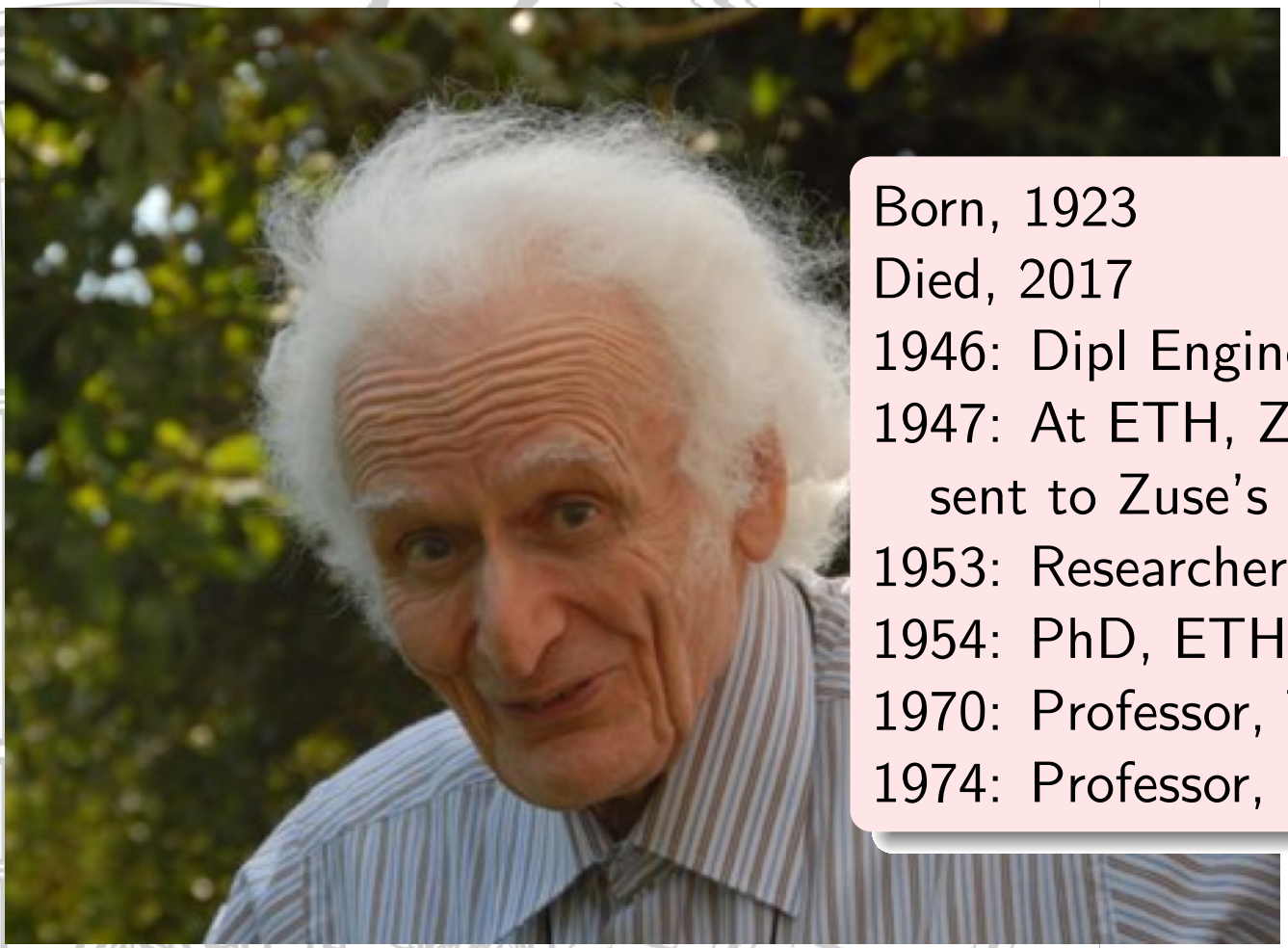
*The programmer should make assertions about the various states that the machine can reach.*

*The checker has to verify that* [these assertions] *agree with the claims that are made for the routine as a whole.*

*Finally the checker has to verify that the process comes to an end.*

*A.M. Turing. Checking a large routine. Paper read on 24 June 1949 at the inaugural conference of the EDSAC computer at the Mathematical Laboratory, Cambridge.*
*Discussed by Morris and Jones, Annals of the History of Computing, Vol. 6, Apr. 1984.*

# Corrado Böhm



Born, 1923
Died, 2017
1946: Dipl Engineering, Lausanne
1947: At ETH, Zürich
sent to Zuse's lab to evaluate the Z4
1953: Researcher at IAC, Rome
1954: PhD, ETH Zürich (Stiefel, Bernays)
1970: Professor, Turin
1974: Professor, Rome La Sapienza

# Böhm: Universal machines

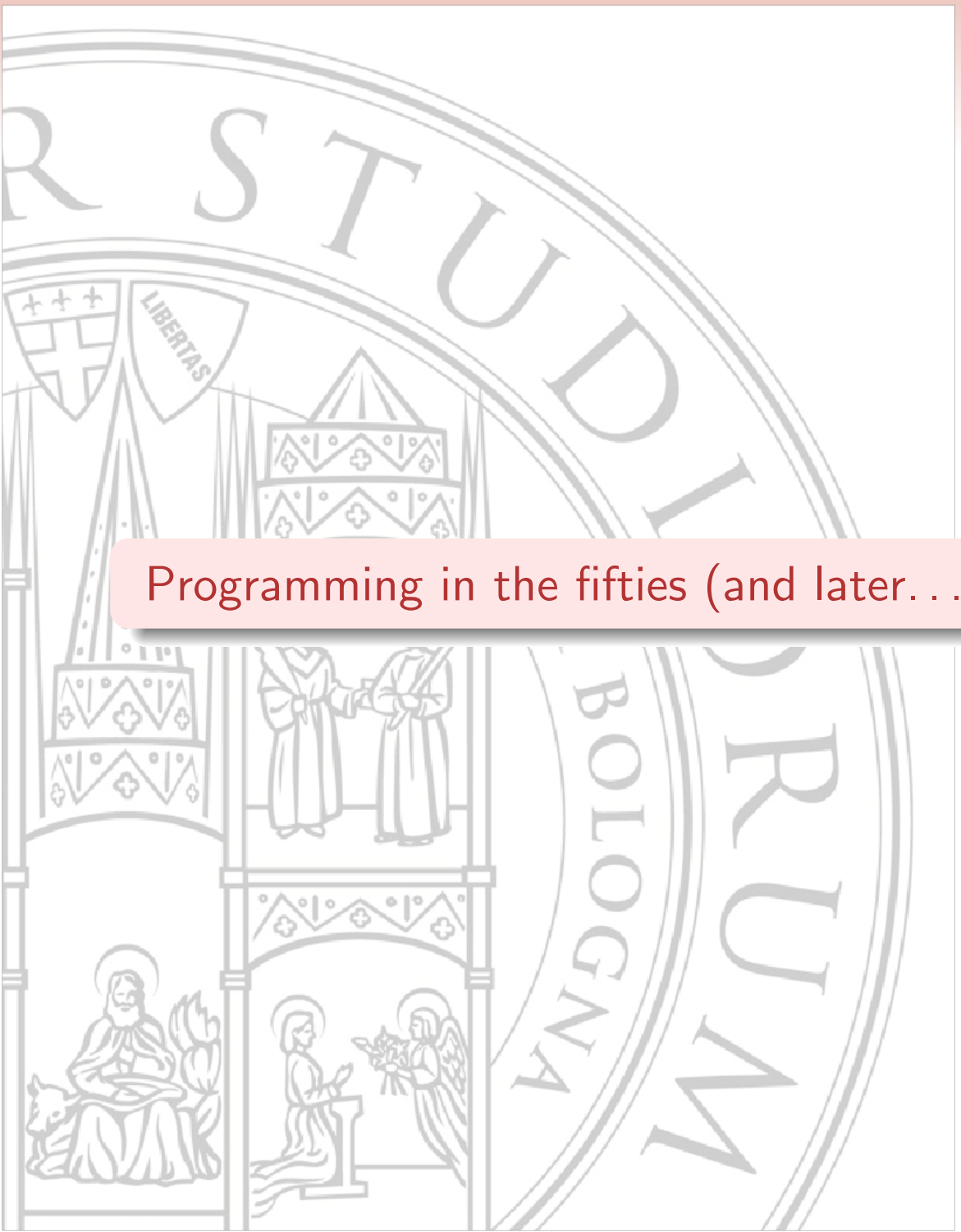
## Calculatrices digitales. Du déchiffrage de formules logico-mathématiques par la machine même dans la conception du programme.

Mémoire de Corrado Böhm (à Roma) (*).

---

Nous voulons admettre — ce qui est assez plausible — que les calculatrices les plus évoluées sont universelles, au sens spécifié par M. Turing.

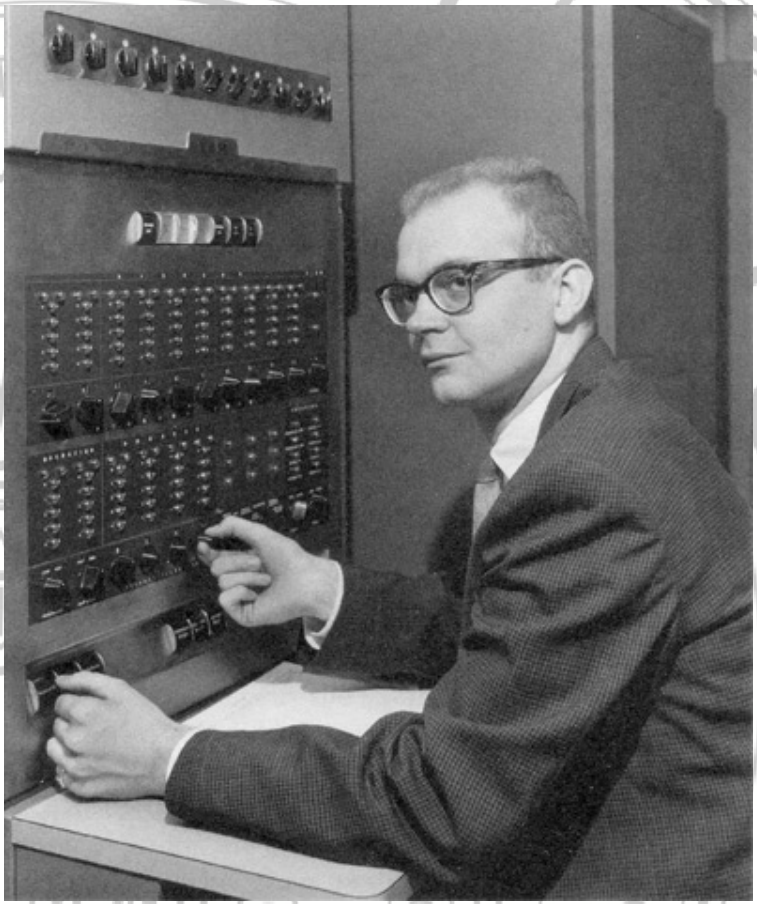
(*) *Ricercatore* à l'*Istituto Naz. per le Applicazioni del Calcolo*. Thèse présentée à l'Ecole Polytechnique Fédérale, Zurich, pour l'obtention du grade de Docteur ès Sciences mathématiques. Rapporteur : Prof. Dr. E. Stiefel ; corapporteur : Prof. Dr. P. Bernays (1952).



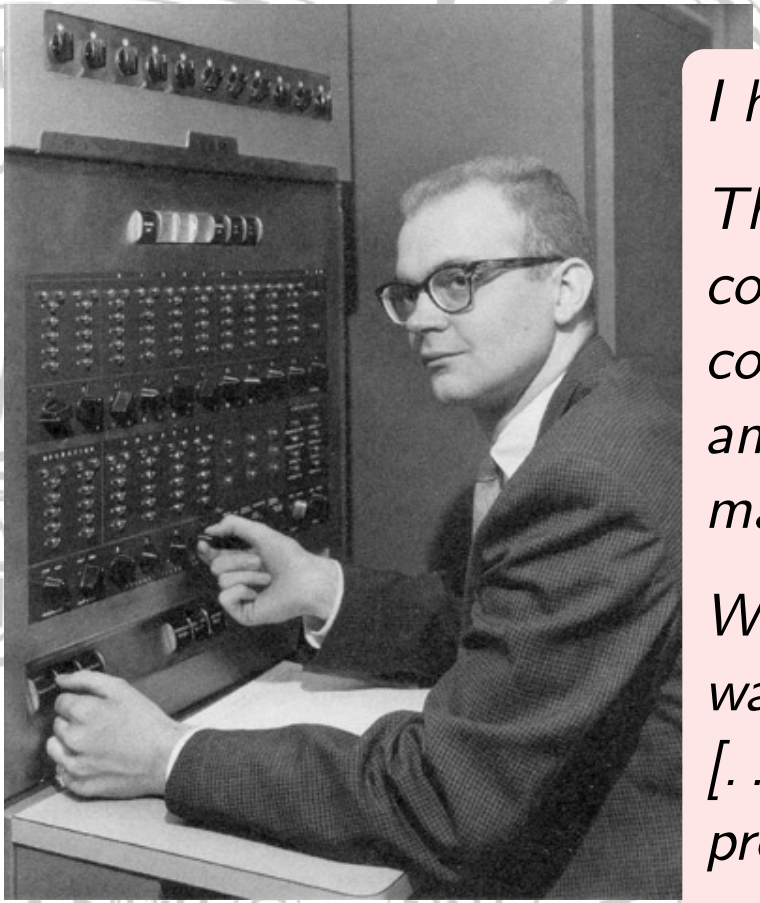

# However,

Programming in the fifties (and later...) was a different story...

# Knuth's recollection, circa 1962



# Knuth's recollection, circa 1962



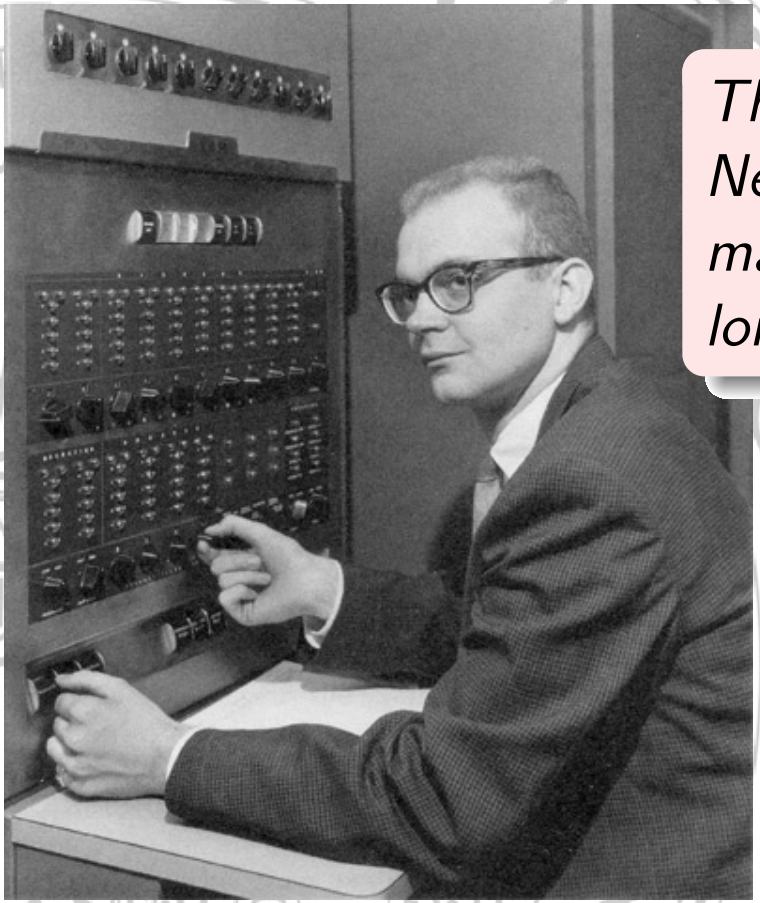
*I had never heard of "computer science"*

*The accepted methodology for program construction was [. . . ]: People would write code and make test runs, then find bugs and make patches, then find more bugs and make more patches, and so on.*

*We never realized that there might be a way to construct a rigorous proof of validity [. . . ] even though I was doing nothing but proofs when I was in a classroom*

*[D.K. Knuth, Robert W. Floyd, in memoriam. ACM SIGACT News 2003]*

# Knuth's recollection, circa 1962



*The early treatises of Goldstine and von Neumann, which provided a glimpse of mathematical program development, had long been forgotten.*

# Donald E. Knuth


Born, 1938
Bachelor and Master of science:
Physics, Mathematics, 1960
PhD Mathematics, 1963
Stanford University, since 1968
*The Art of Computer Programming*: 1968 - today
Turing Award, 1974 (he was 36. . . )

Sorry Mau:

when you'll get the Turing Award, you'll be at least 60. . .

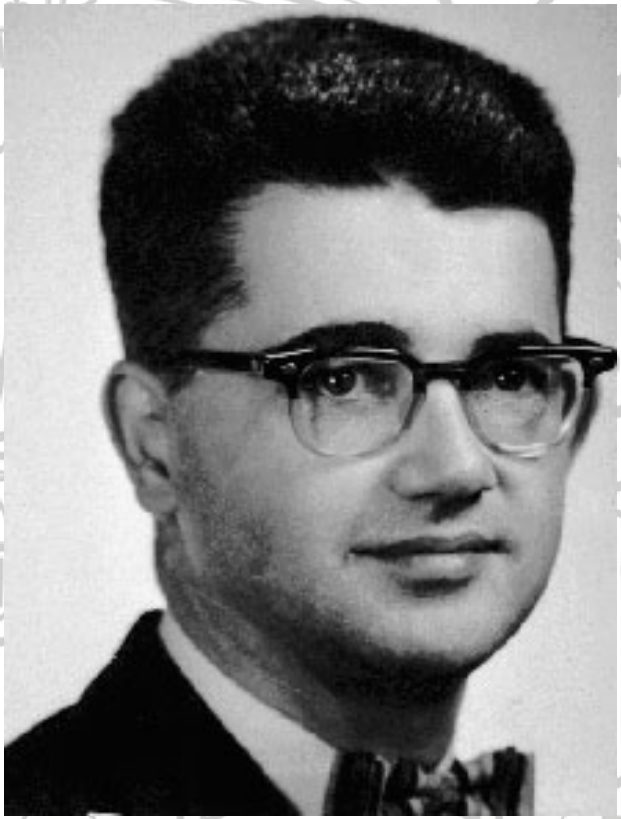
# Donald E. Knuth


Born, 1938
Bachelor and Master of science:
Physics, Mathematics, 1960
PhD Mathematics, 1963
Stanford University, since 1968
*The Art of Computer Programming*: 1968 - today
Turing Award, 1974 (he was 36. . . )

**Sorry Mau:**

when you'll get the Turing Award, you'll be at least 60. . .

# A Mathematical Theory of Computation

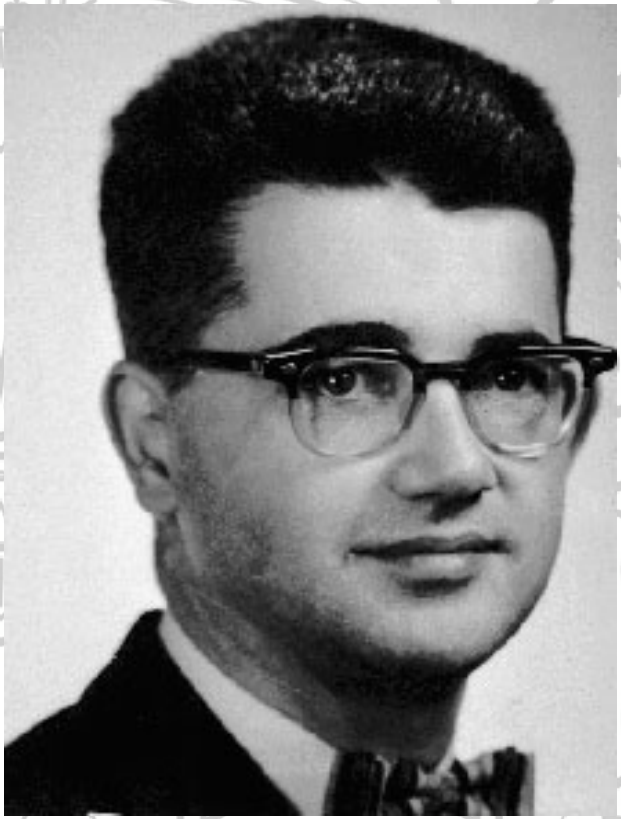


*It is reasonable to hope that the relationship between computation and mathematical logic will be as fruitful in the next century as that between analysis and physics in the last.*

John McCarthy, MIT 1961; Stanford 1963

From the conclusion of the final version of the paper (1963): A Basis for a Mathematical Theory of Computation. 1961: the Western Joint Computer Conference; 1962: IBM symposium in Blaricum, Netherlands; 1963: in *Computer Programming and Formal Systems*, North Holland.

# A Mathematical Theory of Computation



*It is reasonable to hope that the relationship between computation and mathematical logic will be as fruitful in the next century as that between analysis and physics in the last.*

John McCarthy, MIT 1961; Stanford 1963

From the conclusion of the final version of the paper (1963): A Basis for a Mathematical Theory of Computation. 1961: the Western Joint Computer Conference; 1962: IBM symposium in Blaricum, Netherlands; 1963: in *Computer Programming and Formal Systems*, North Holland.

# John McCarthy

Born, 1927
Died, 2011

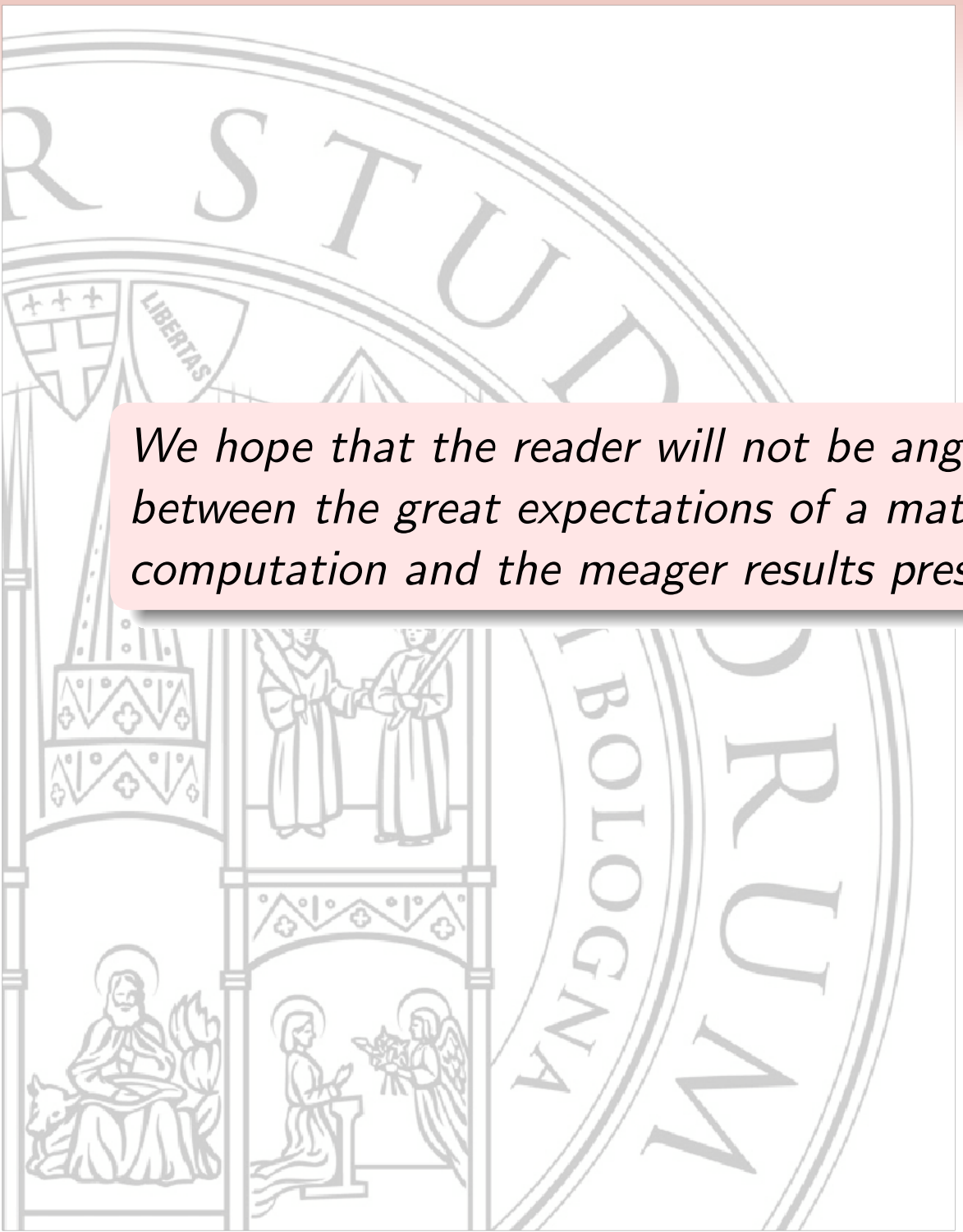
Bachelor of science: Mathematics, 1942
PhD Mathematics, 1951
MIT, 1956–1962
Stanford University, since 1968
Turing Award, 1971

Artificial intelligence
Lisp

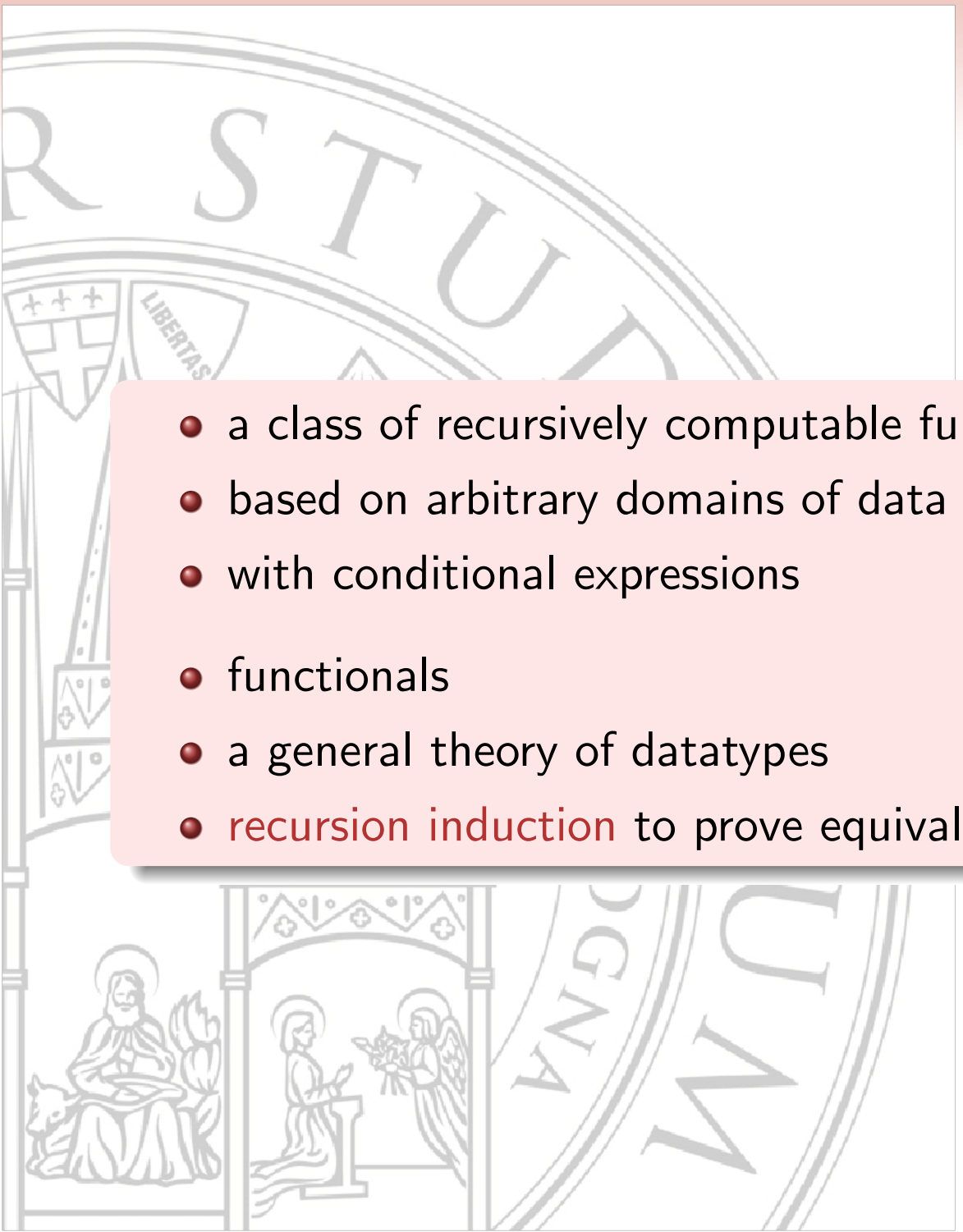
## A basis for a Mathematical Theory of Computation

Expected *practical* results:

1. To develop a universal programming language
2. To define a theory of the equivalence of computation processes
3. To represent algorithms by symbolic expressions [...]
4. To represent computers as well as computations in a formalism that permits a treatment of the relation between a computation and the computer that carries out the computation
5. To give a quantitative theory of computation.



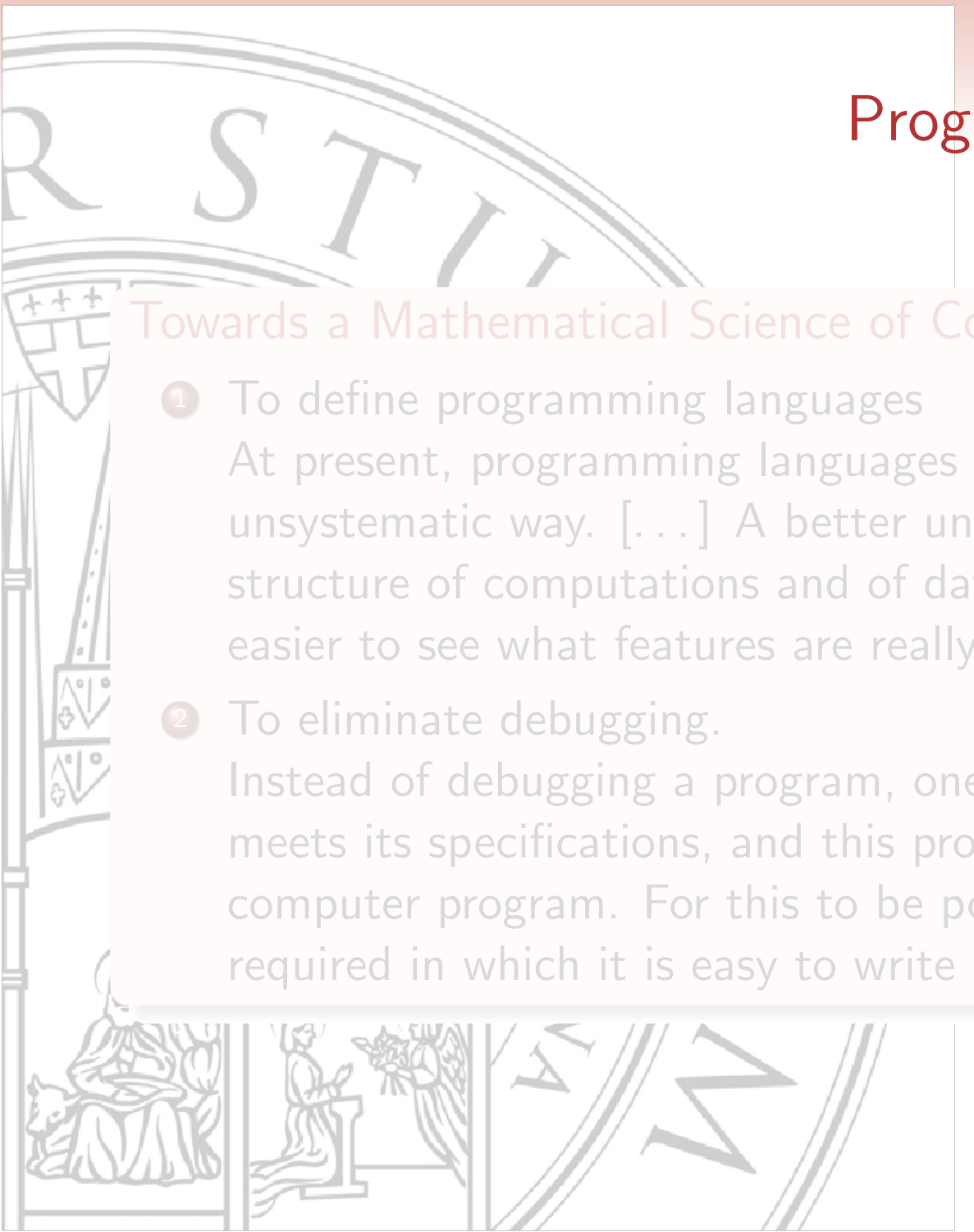
*We hope that the reader will not be angry about the contrast between the great expectations of a mathematical theory of computation and the meager results presented in this paper.*



# Contents

- a class of recursively computable functions
- based on arbitrary domains of data and operations on them
- with conditional expressions

- functionals
- a general theory of datatypes
- recursion induction to prove equivalences



# Program correctness

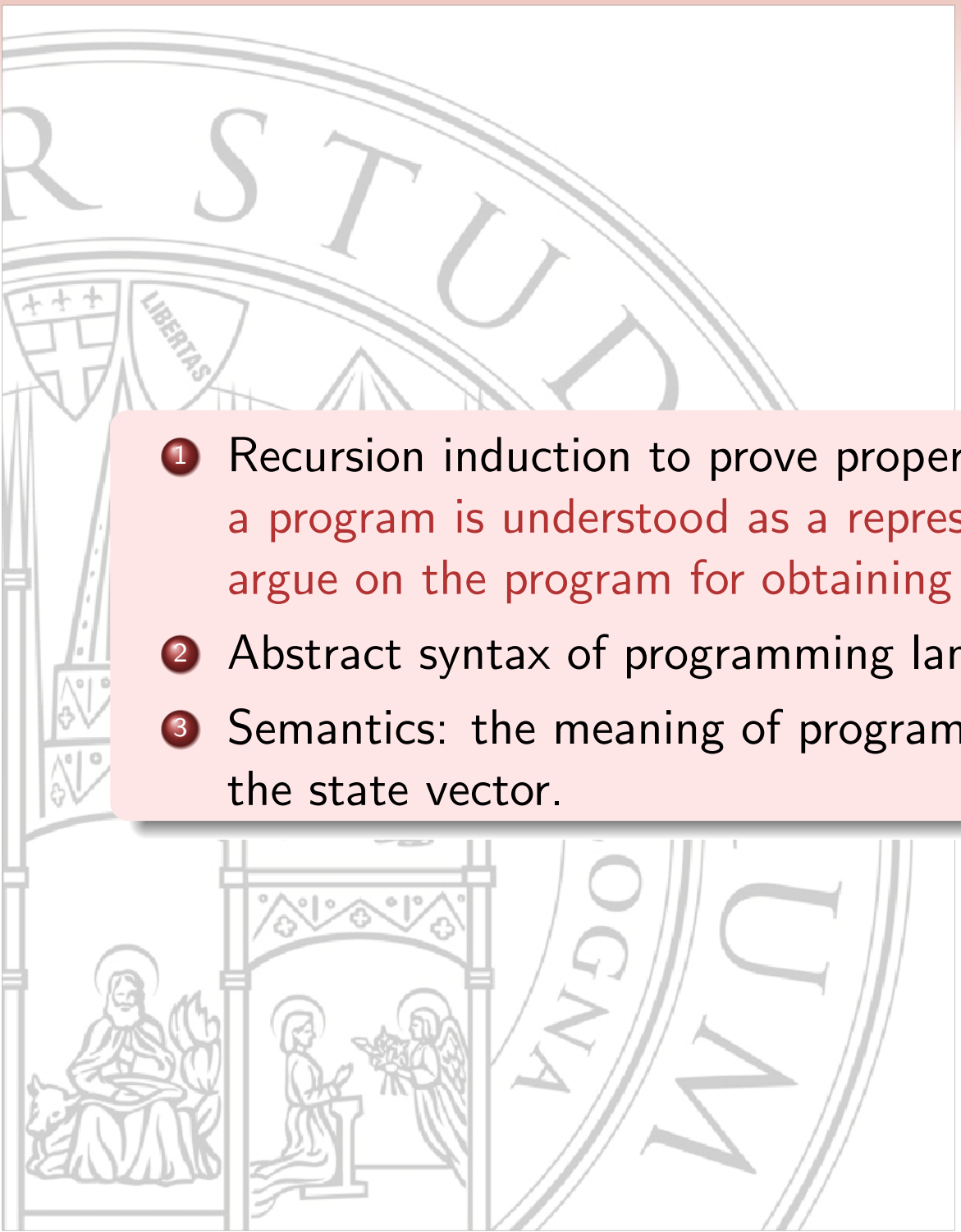
## Towards a Mathematical Science of Computation, IFIP 1962

1. To define programming languages
   At present, programming languages are constructed in a very unsystematic way. [...] A better understanding of the structure of computations and of data spaces will make it easier to see what features are really desirable.
2. To eliminate debugging.
   Instead of debugging a program, one should prove that it meets its specifications, and this proof should be checked by a computer program. For this to be possible, formal systems are required in which it is easy to write proofs.

# Program correctness

## Towards a Mathematical Science of Computation, IFIP 1962

1. To define programming languages
   At present, programming languages are constructed in a very unsystematic way. [...] A better understanding of the structure of computations and of data spaces will make it easier to see what features are really desirable.
2. To eliminate debugging.
   Instead of debugging a program, one should prove that it meets its specifications, and this proof should be checked by a computer program. For this to be possible, formal systems are required in which it is easy to write proofs.



# Contents

1. Recursion induction to prove properties of Algol programs:
   a program is understood as a representative for its meaning:
   argue on the program for obtaining results on its model
2. Abstract syntax of programming languages
3. Semantics: the meaning of program is defined by its effect on the state vector.

# R. Floyd

*An adequate basis for formal definitions of the meanings of programs [. . . ] in such a way that a rigorous standard is established for proofs about computer programs*

*Based on ideas of Perlis and Gorn*

*That semantics of a programming language may be defined independently of all processors [. . . ] appear[s] to be new,*

*although McCarthy has done similar work for programming languages based on evaluation of recursive functions.*

*Robert W. Floyd. Assigning meaning to programs. Mathematical Aspects of Computer Science, AMS 1967.*

# R. Floyd

*An adequate basis for formal definitions of the meanings of programs [. . . ] in such a way that a rigorous standard is established for proofs about computer programs*

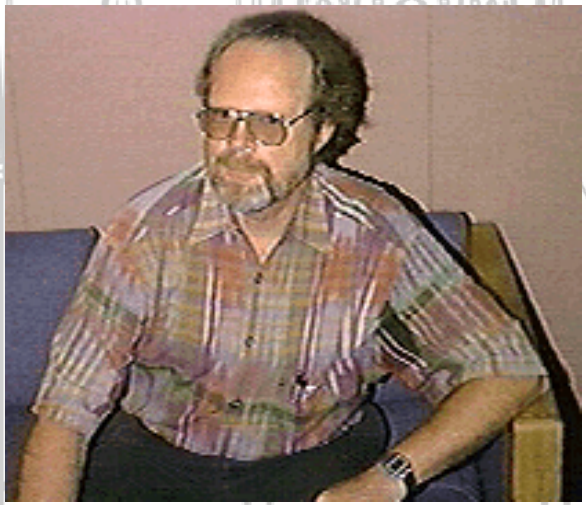
*Based on ideas of Perlis and Gorn*

*That semantics of a programming language may be defined independently of all processors [. . . ] appear[s] to be new,*

*although McCarthy has done similar work for programming languages based on evaluation of recursive functions.*

*Robert W. Floyd. Assigning meaning to programs. Mathematical Aspects of Computer Science, AMS 1967.*

# Robert W. Floyd



Born, 1936
Died, 2001

Bachelor in liberal arts: 1953 (he was 17)
Bachelor in physics: 1958
No PhD
Computer programmer
Carnegie Mellon University, 1965–1968
Stanford University, since 1968
Turing Award, 1978

Grammars
Programming language semantics

Our giant founding fathers:

- Alan Turing
- John von Neumann
- Corrado Böhm
- John McCarthy
- Bob Floyd
- Tony Hoare
- Peter Landin
- Christopher Strachey
- and the many many others...

*There is a common theme in their (very different!) approaches...*

Our giant founding fathers:

- Alan Turing
- John von Neumann
- Corrado Böhm
- John McCarthy
- Bob Floyd
- Tony Hoare
- Peter Landin
- Christopher Strachey
- and the many many others...

*There is a common theme in their (very different!) approaches...*

# The standard model

*True arithmetic and (in principle) unbounded resources*

Proof methods applied directly to programs without first transforming them into their semantic equivalent [McCarthy 1962]

# The standard model

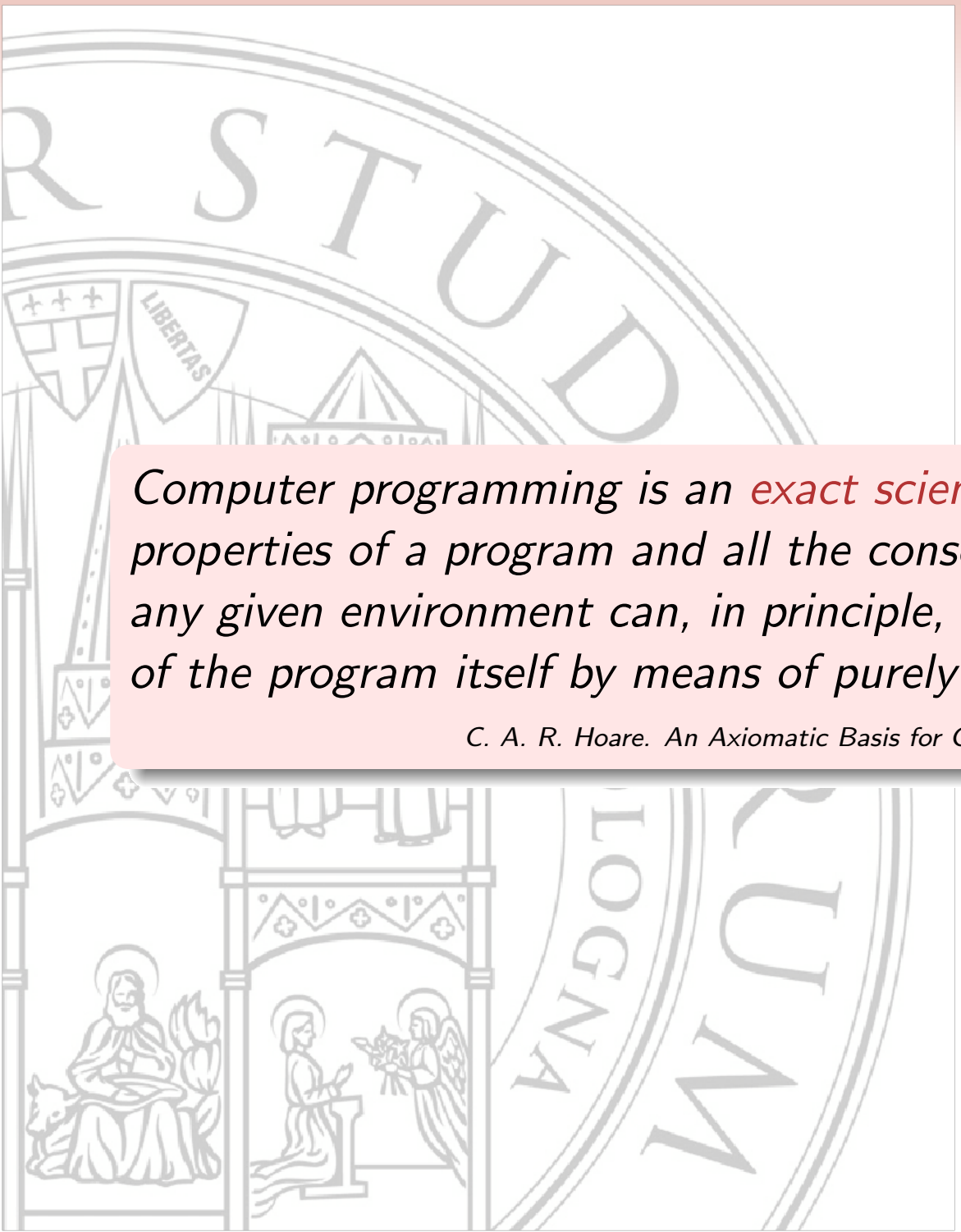
*True arithmetic and (in principle) unbounded resources*

Proof methods applied directly to programs without first transforming them into their semantic equivalent [McCarthy 1962]

# The standard model

Three features:

- compositionality
- extensionality
- referential transparency

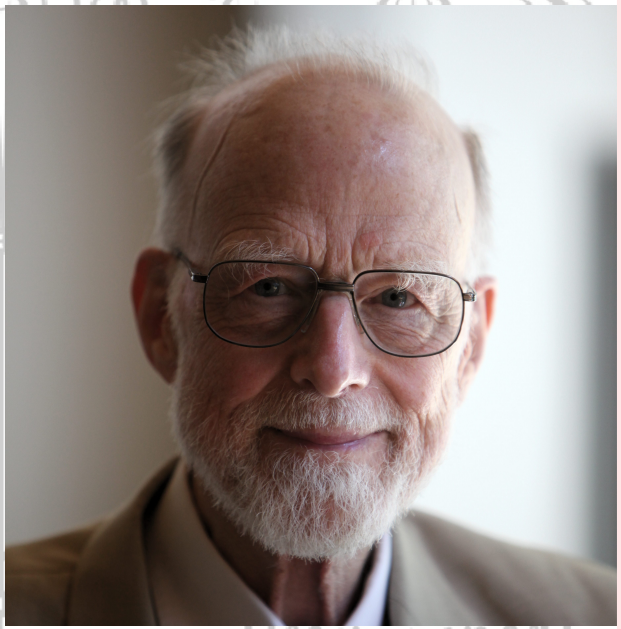


# C.A.R. Hoare

*Computer programming is an exact science in that all the properties of a program and all the consequences of executing it in any given environment can, in principle, be found out from the text of the program itself by means of purely deductive reasoning.*

*C. A. R. Hoare. An Axiomatic Basis for Computer Programming. CACM 12(10), 1969.*

# Charles Anthony R. (Tony) Hoare

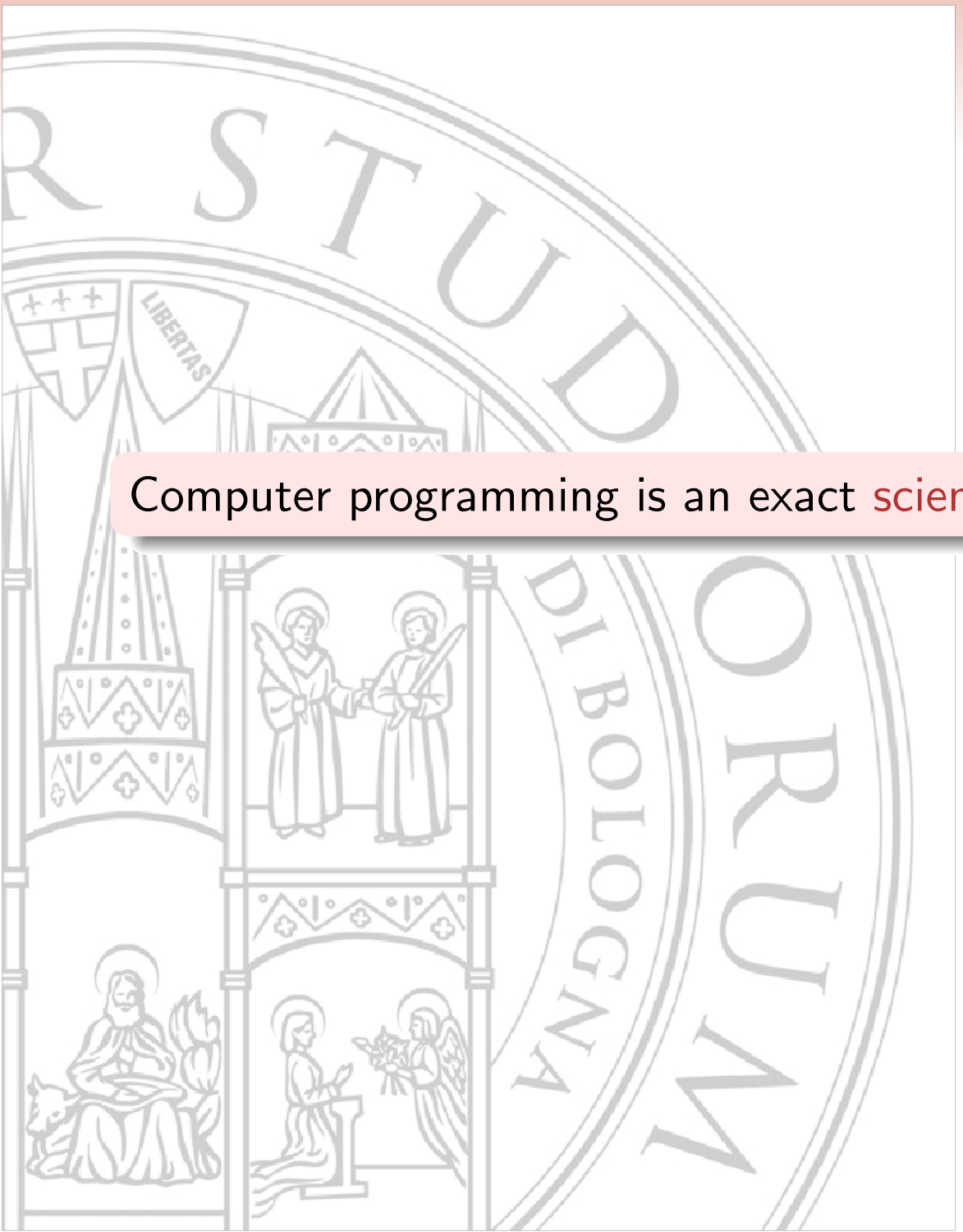


Born, 1934

Classics and philosophy: 1956
Statistics, Computer programmer: 1956-1968

Queen's University of Belfast, 1968–1977
Oxford University, since 1977
Turing Award, 1980
FRS

Algorithms (QuickSort), concurrency
Definition and design of programming languages




Computer programming is an exact science. . .

# Resistances

*Most scientists thought that using a computer was simply programming — that it didn't involve any deep scientific thought and that anyone could learn to program. So why have a degree? They thought computers were vocational vs. scientific in nature.*

*[Conte, Computerworld magazines, 1999]*

# Computer Science Dpts

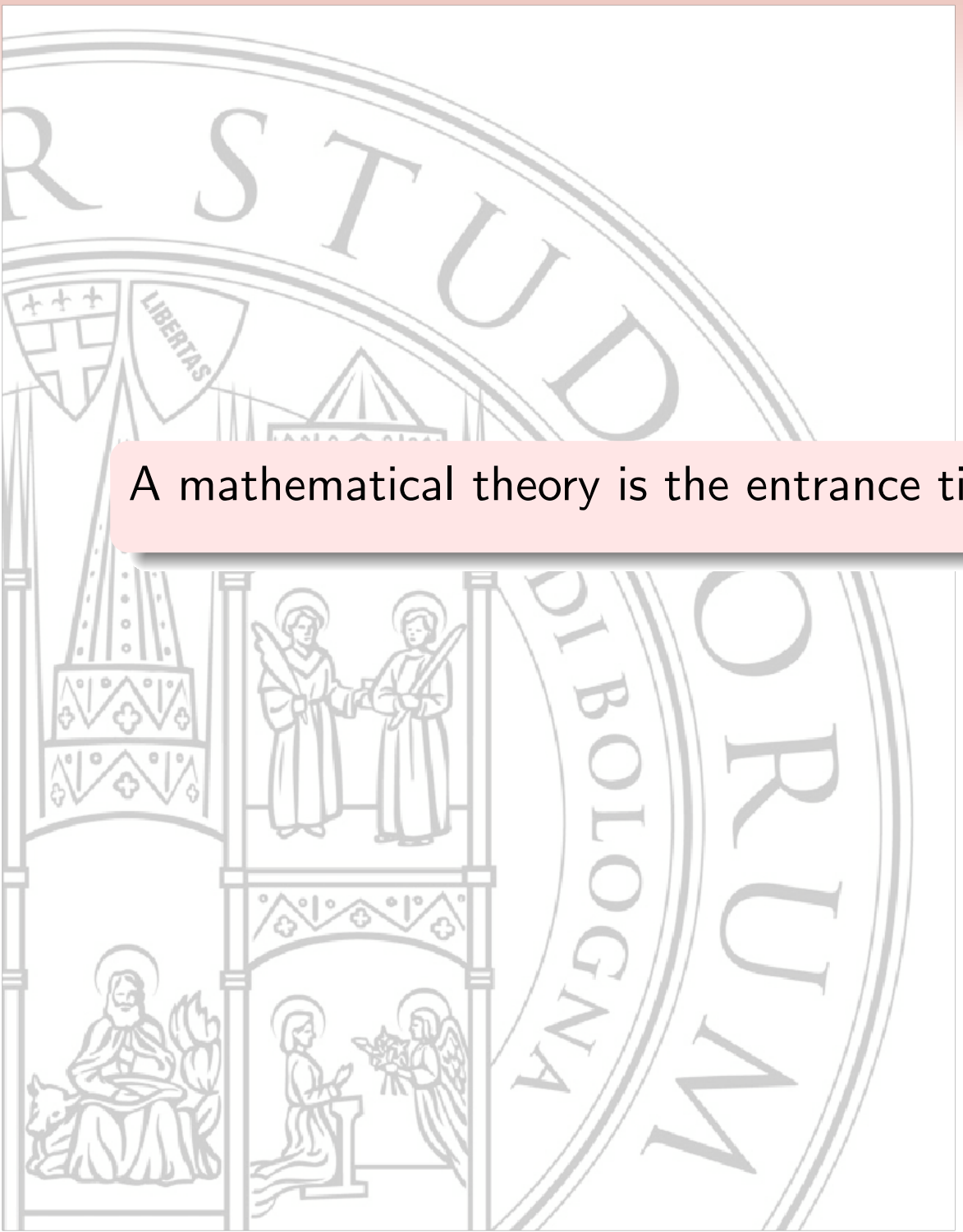
1962 Purdue University (West Lafayette, IN): first dpt of CS; Samuel D. Conte (Perlis: 1951-1956@computation center)

1965 Stanford University (Palo Alto, CA); George Forsythe (Herriot, McCarthy, Feigenbaum, Wirth, Knuth(later)) Since 1961 it was a "division" of Math Dpt.

1965 Carnegie Mellon University (Pittsburg, PA); Alan J. Perlis (Allen, Simon)

1965 First PhD *given by a CS Dpt*: Richard Wexelblat @ University of Pennsylvania (ENIAC!)

1971 Yale (New Haven, CT); Perlis




A mathematical theory is the entrance ticket to science

# The grand view

## Structural engineering

- mathematical physics laws
- empirical knowledge

to understand, predict, and calculate the stability, strength and rigidity of structures for buildings.

*McCarthy:*

*the relationship between computation and mathematical logic will be as fruitful as that between analysis and physics.*

# C.A.R. Hoare

*When the correctness of a program, its compiler, and the hardware of the computer have all been established with mathematical certainty, it will be possible to place great reliance on the results of the program, and predict their properties with a confidence limited only by the reliability of the electronics.*

*C. A. R. Hoare. An Axiomatic Basis for Computer Programming. CACM 12(10), 1969.*

# Hierarchy of machines

- All levels are of the same (abstract) nature
- All levels could be subject (at least conceptually) to the same analysis.
- A formally proved chain of compilers:
  a proof that a model of the hight level program satisfies a condition,
  transfers to a proof that a model of the low level program satisfies a certain condition (automatically obtained from the other)
- No concrete, iron, workmanship is involved.

# The standard model

*True arithmetic and (in principle) unbounded resources*

The standard model is to PL
what movement without friction is to mechanics

# The standard model

*True arithmetic and (in principle) unbounded resources*

The standard model is to PL
what movement without friction is to mechanics

# The standard model

## Ontological interpretation

Contrary to some mechanistic interpretation (Piccinini, but also some Dijkstra):

- a program *denotes*
- a program talks about the world
- a program acts on the object of the world

And then an ethical responsability. . .

# The standard model

## Ontological interpretation

Contrary to some mechanistic interpretation (Piccinini, but also some Dijkstra):

- a program *denotes*
- a program talks about the world
- a program acts on the object of the world

## And then an ethical responsability. . .

Happy birthday, Maurizio!