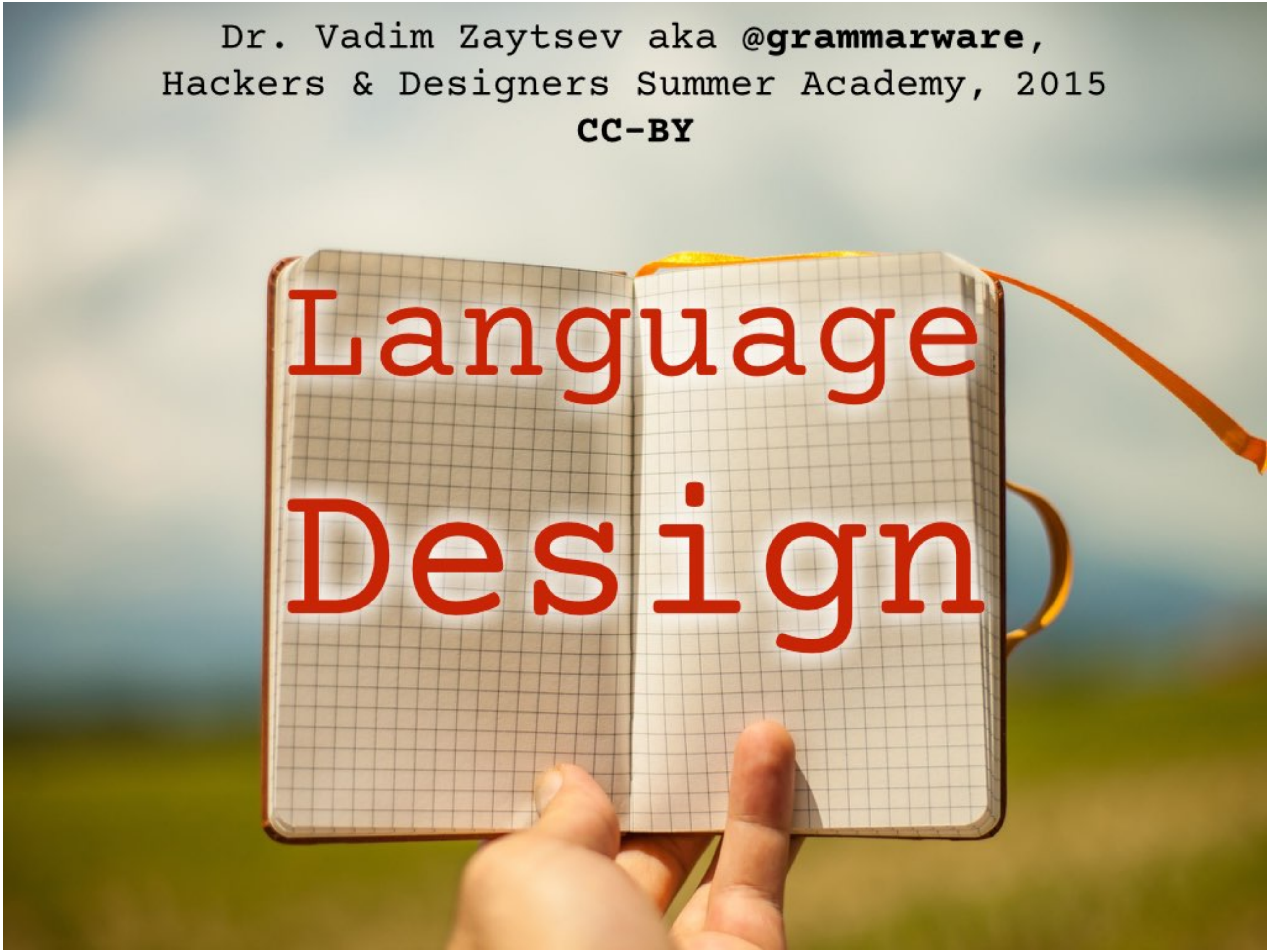
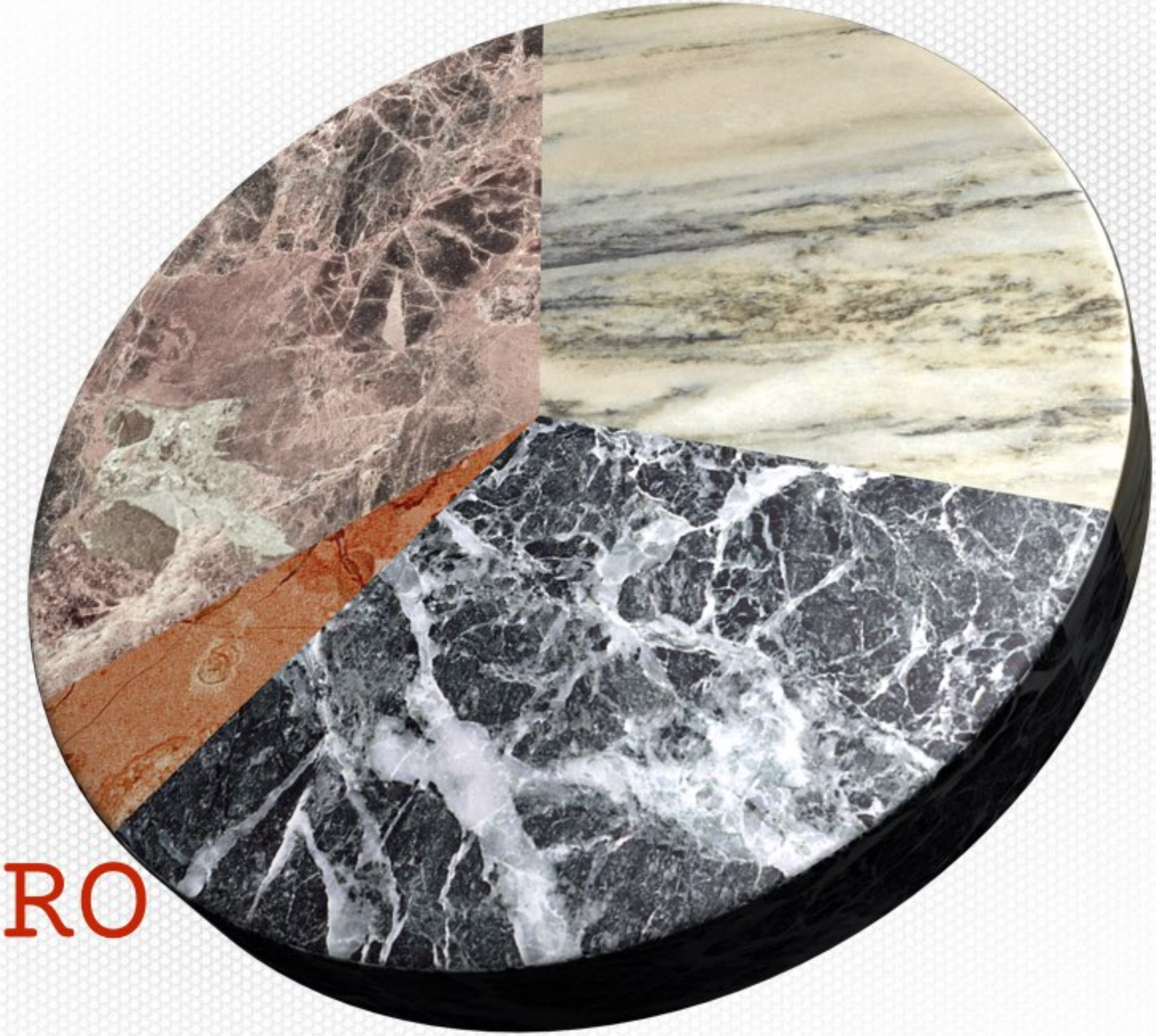
Dr. Vadim Zaytsev aka @**grammarware**,
Hackers & Designers Summer Academy, 2015
CC-BY



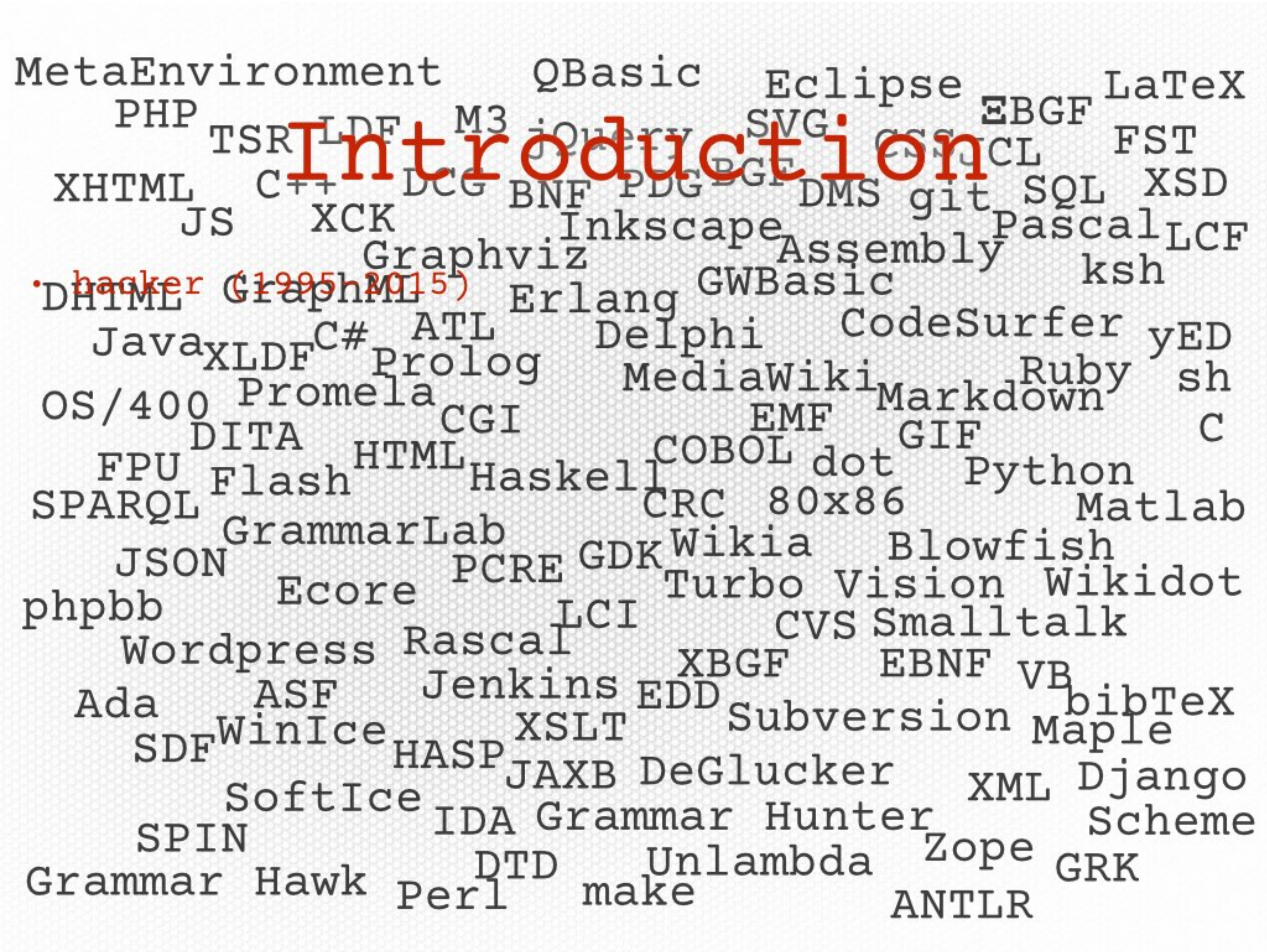
Language Design



INTRO

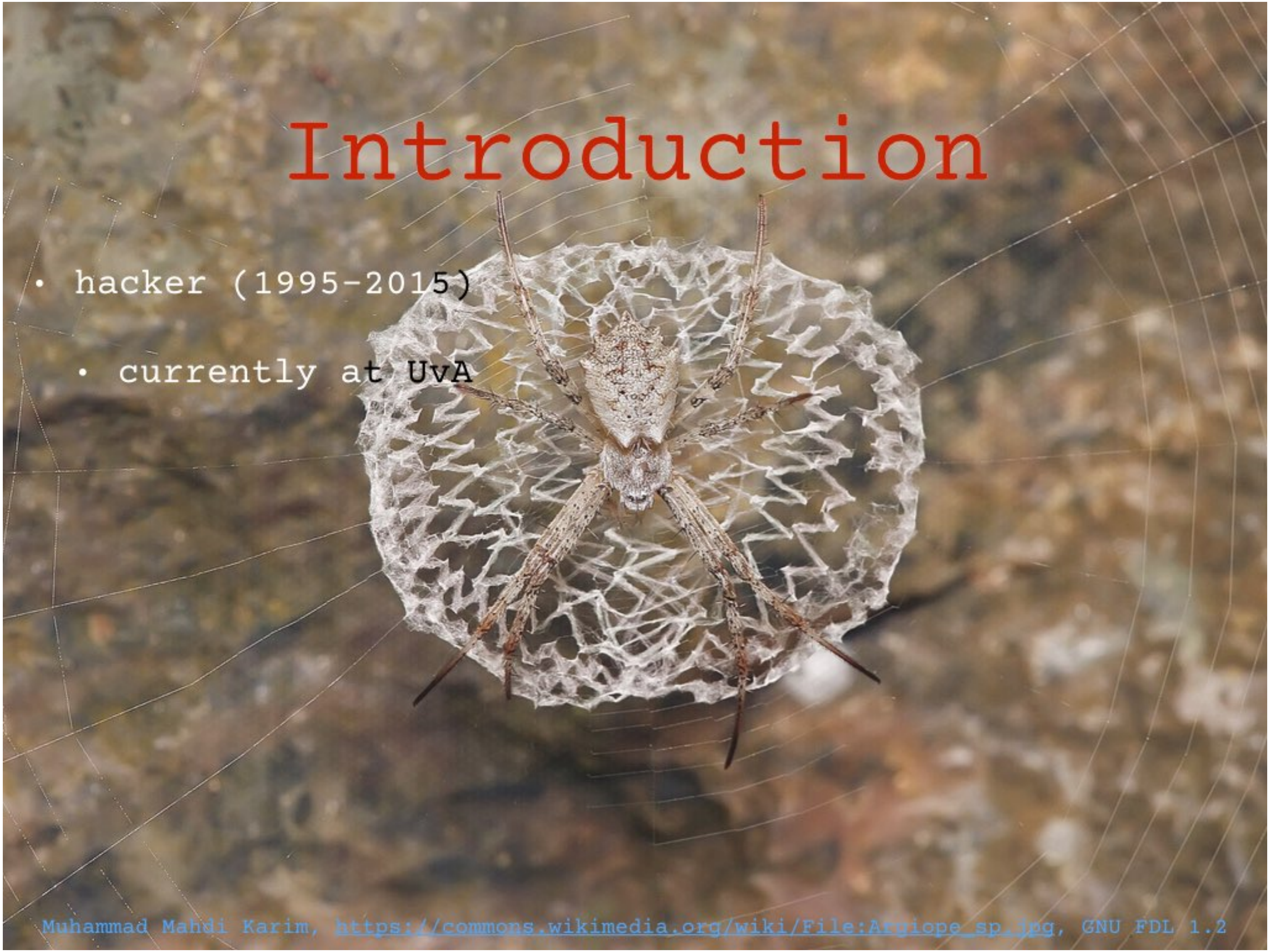
Introduction

- hacker (1995-2015)



Introduction

- hacker (1995-2015)
  - currently at UvA



Muhammad Mahdi Karim, https://commons.wikimedia.org/wiki/File:Argiope_sp.jpg, GNU FDL 1.2

- hacker (1995-2015)
- currently at UvA

photo credit: http://scii.nl/spaces.html

Amsterdam Subversive Center for Information Interchange

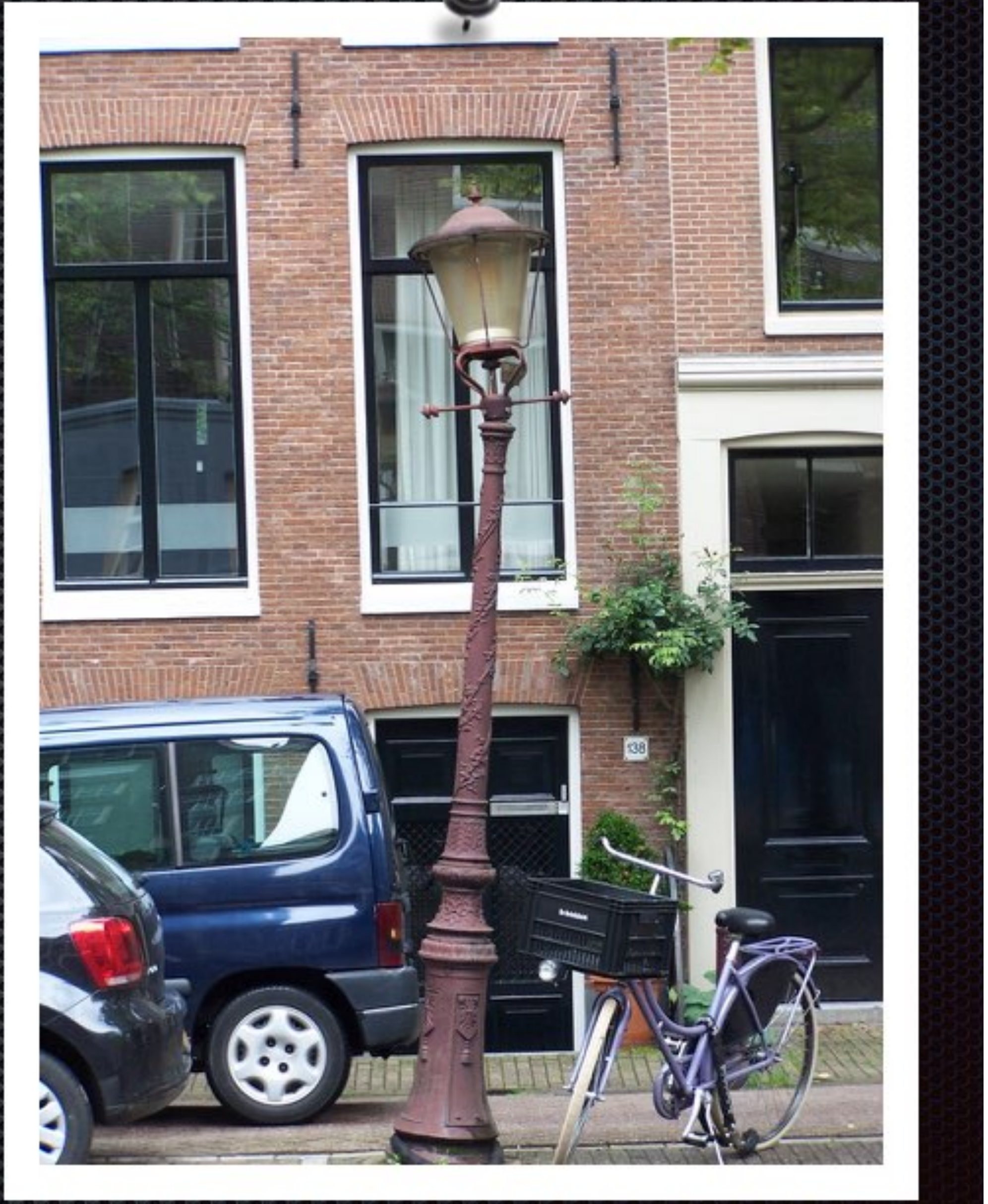
24
24
Het Fort
van Sjakoo
BASE
MENT

- hacker (1995-2015)
  - currently at UvA

- hacker (1995-2015)
  - currently at UvA
- wikipedian (2004-2015)

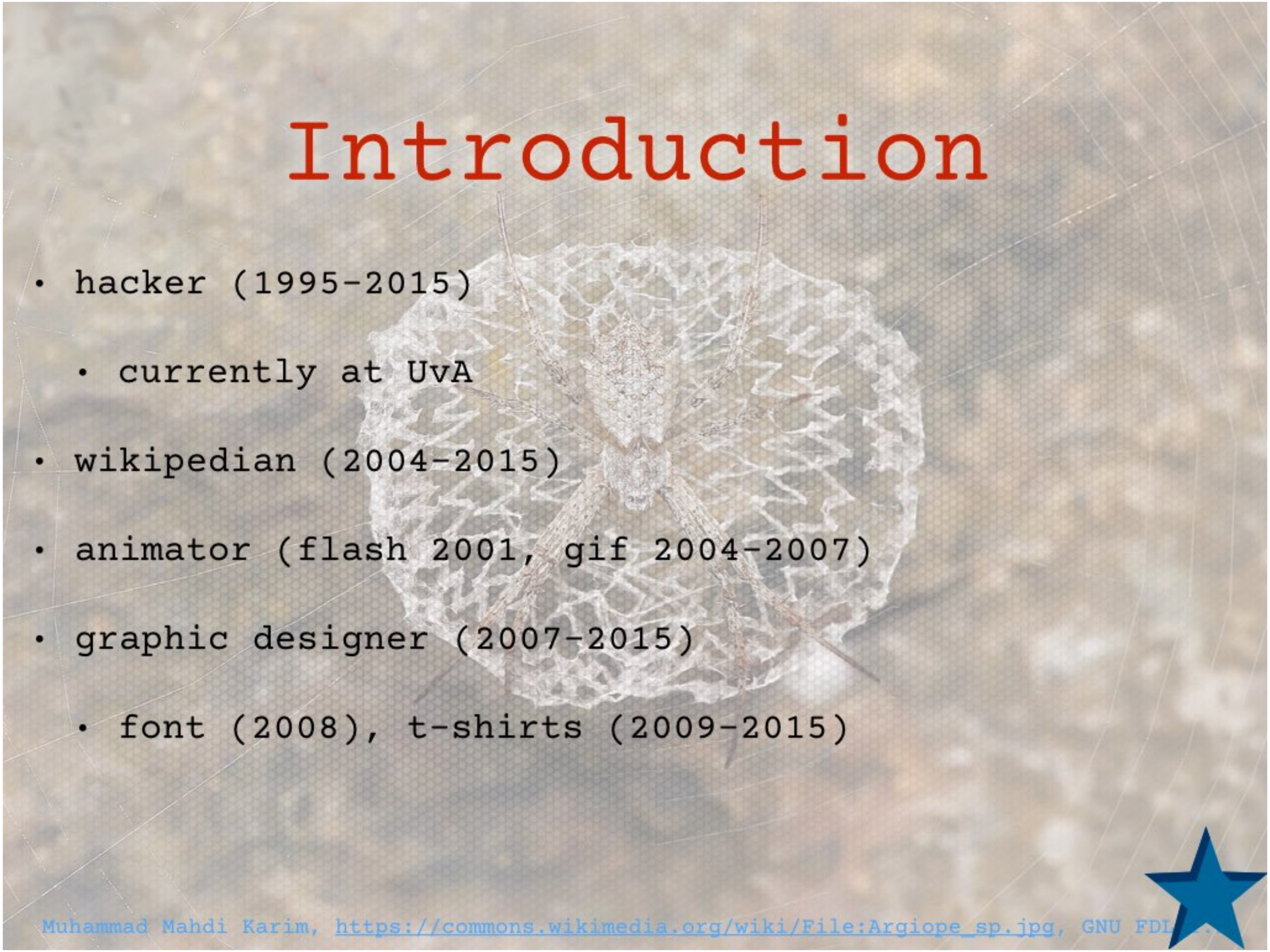
- hacker (1995-2015)
  - currently at UvA

wikipedian (2004-2015)



Introduction

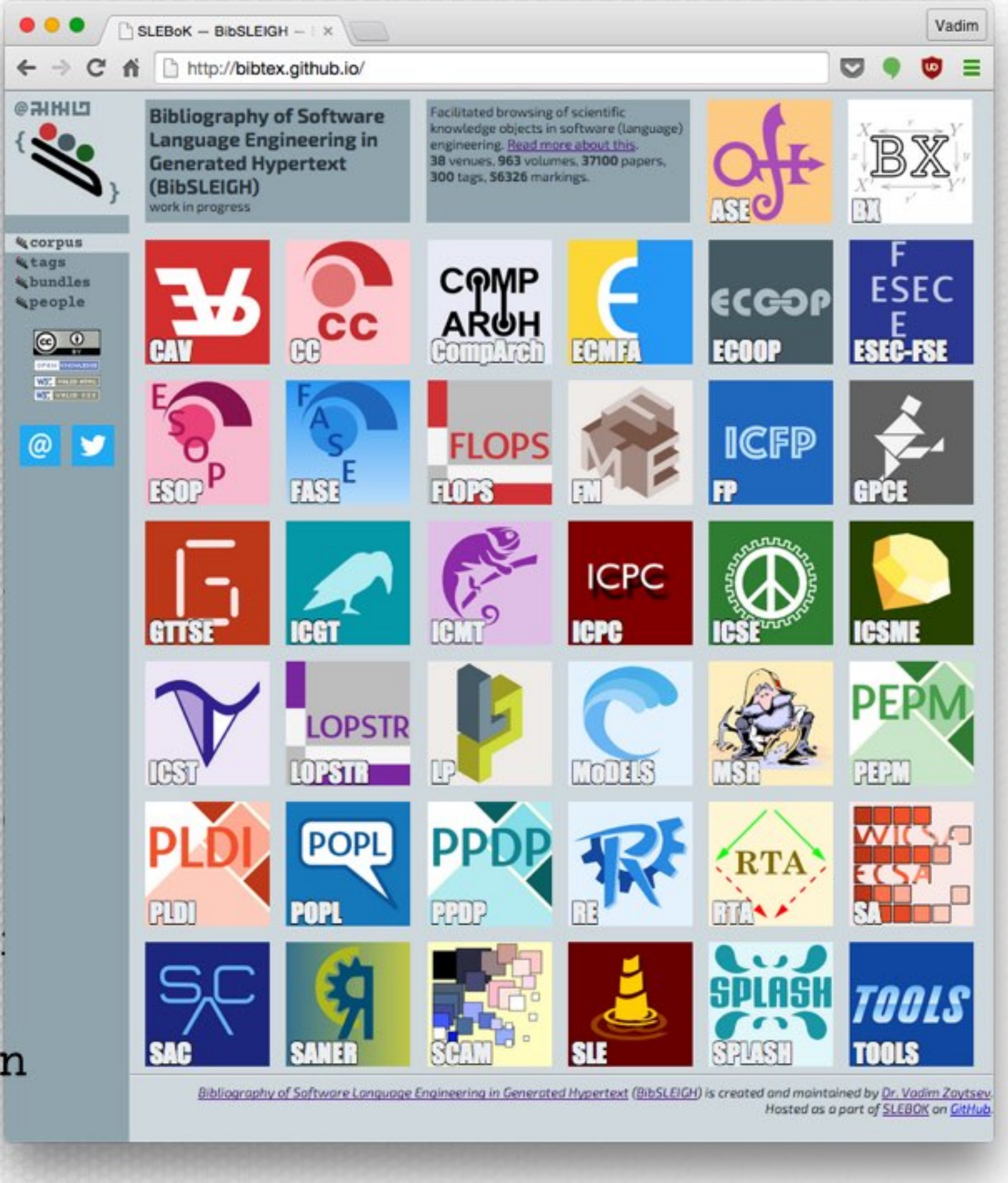
- hacker (1995-2015)
  - currently at UvA
- wikipedian (2004-2015)
- animator (flash 2001, gif 2004-2007)
- graphic designer (2007-2015)
  - font (2008), t-shirts (2009-2015)



Muhammad Mahdi Karim, https://commons.wikimedia.org/wiki/File:Argiope_sp.jpg, GNU FDL

Intr

- hacker (1995-2015)
    - currently at UvA
- wikipedian (2004-201
- animator (flash 2001
- graphic designer (20
    - font (2008), t-shi
    - mostly illustration



SLEBoK — BibSLEIGH —

http://bibtex.github.io/

Vadim

corpus
tags
bundles
people

Bibliography of Software Language Engineering in Generated Hypertext (BibSLEIGH)
work in progress

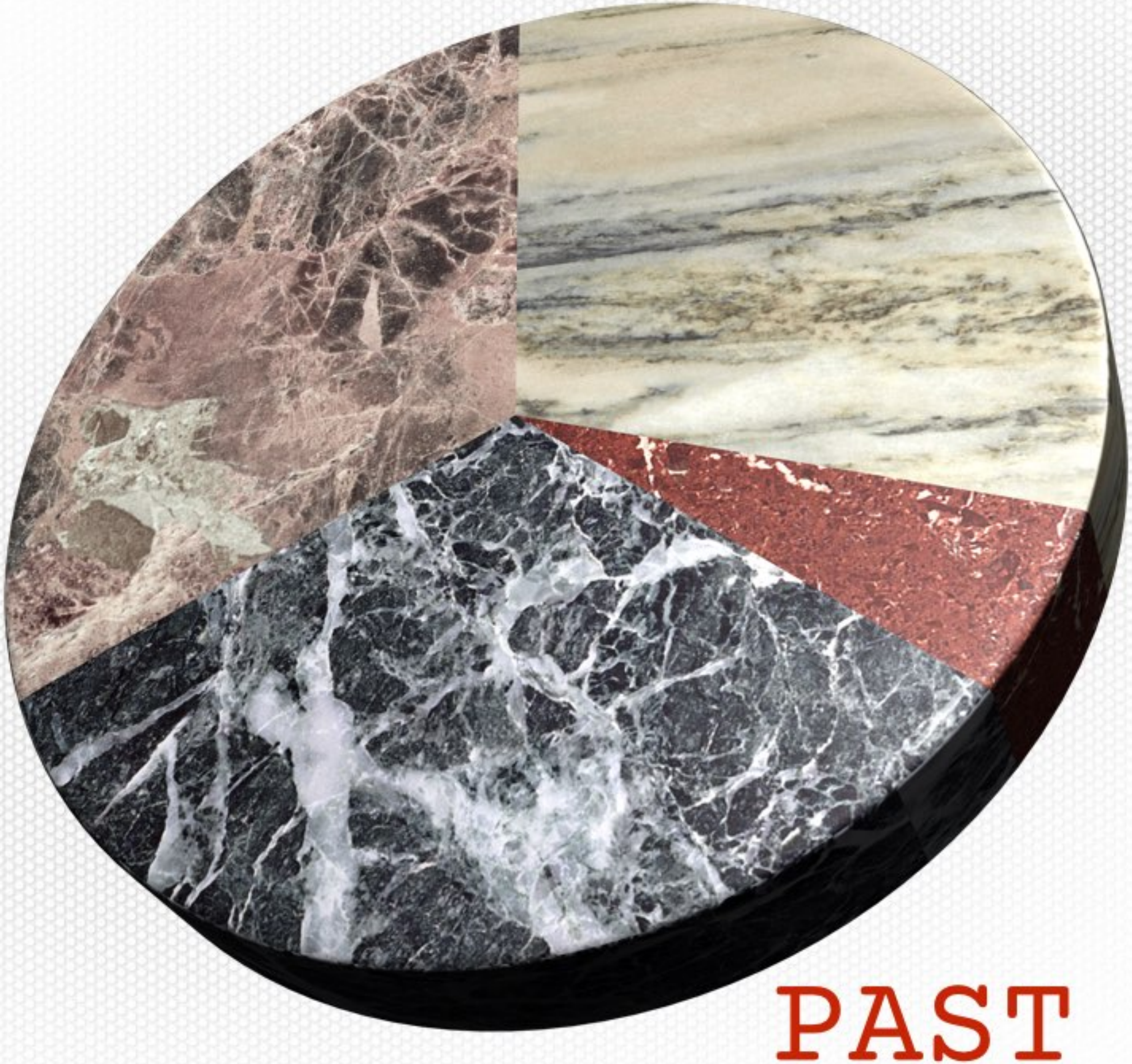
Facilitated browsing of scientific knowledge objects in software (language) engineering. Read more about this.
38 venues, 963 volumes, 37100 papers, 300 tags, 56326 markings.

Bibliography of Software Language Engineering in Generated Hypertext (BibSLEIGH) is created and maintained by Dr. Vadim Zaytsev.
Hosted as a part of SLEBOK on GitHub.

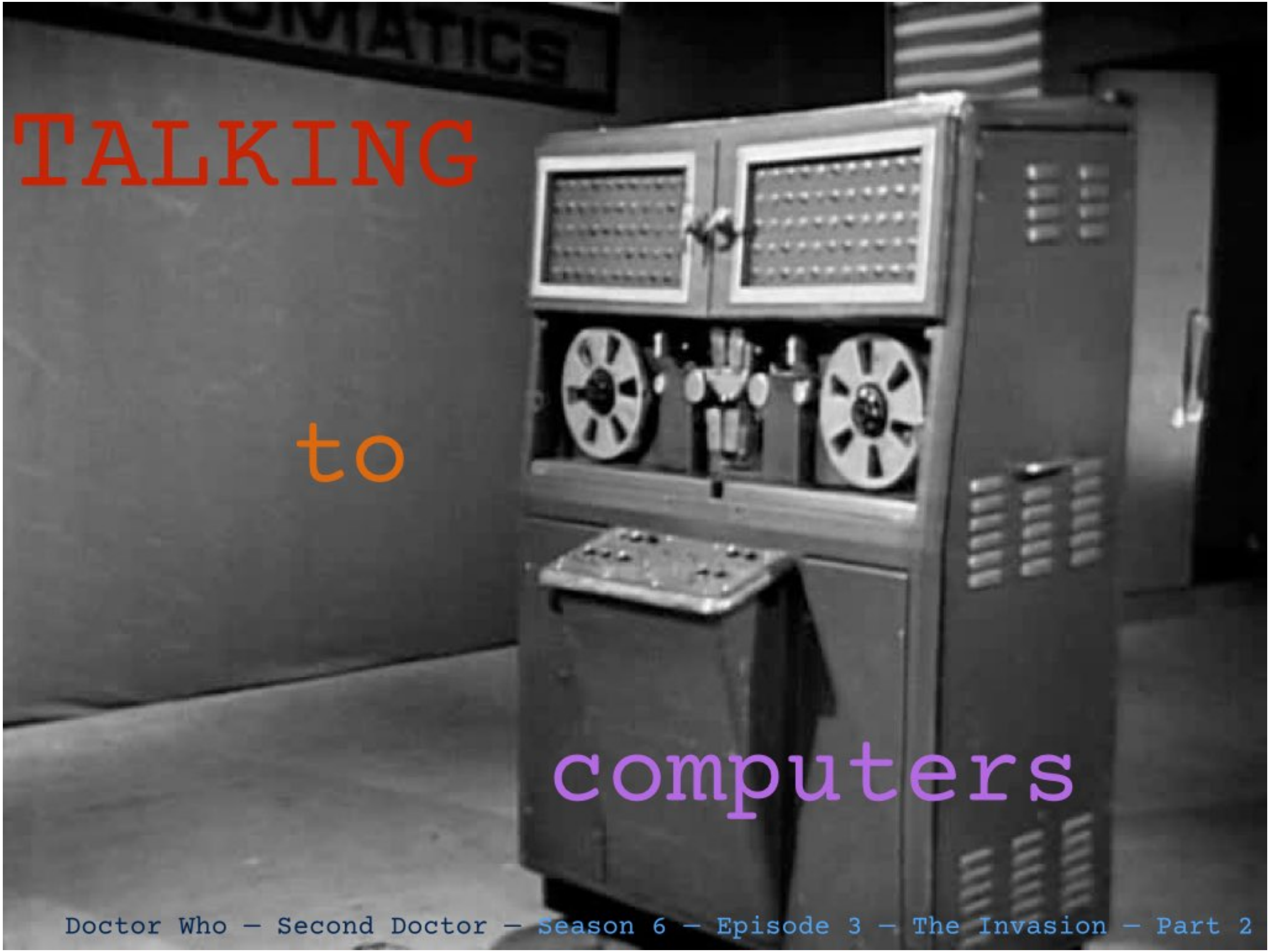
BibSLEIGH: http://bibtex.github.io

Introduction

- hacker (1995-2015)
  - currently at UvA
- wikipedian (2004-2015)
- animator (flash 2001, gif 2004-2007)
- graphic designer (2007-2015)
  - font (2008), t-shirts (2009-2015)
  - mostly illustrations
- indie game designer (1999-2015)



PAST



TALKING to computers

Doctor Who — Second Doctor — Season 6 — Episode 3 — The Invasion — Part 2

Computer?

Perhaps the professor
could use your computer.

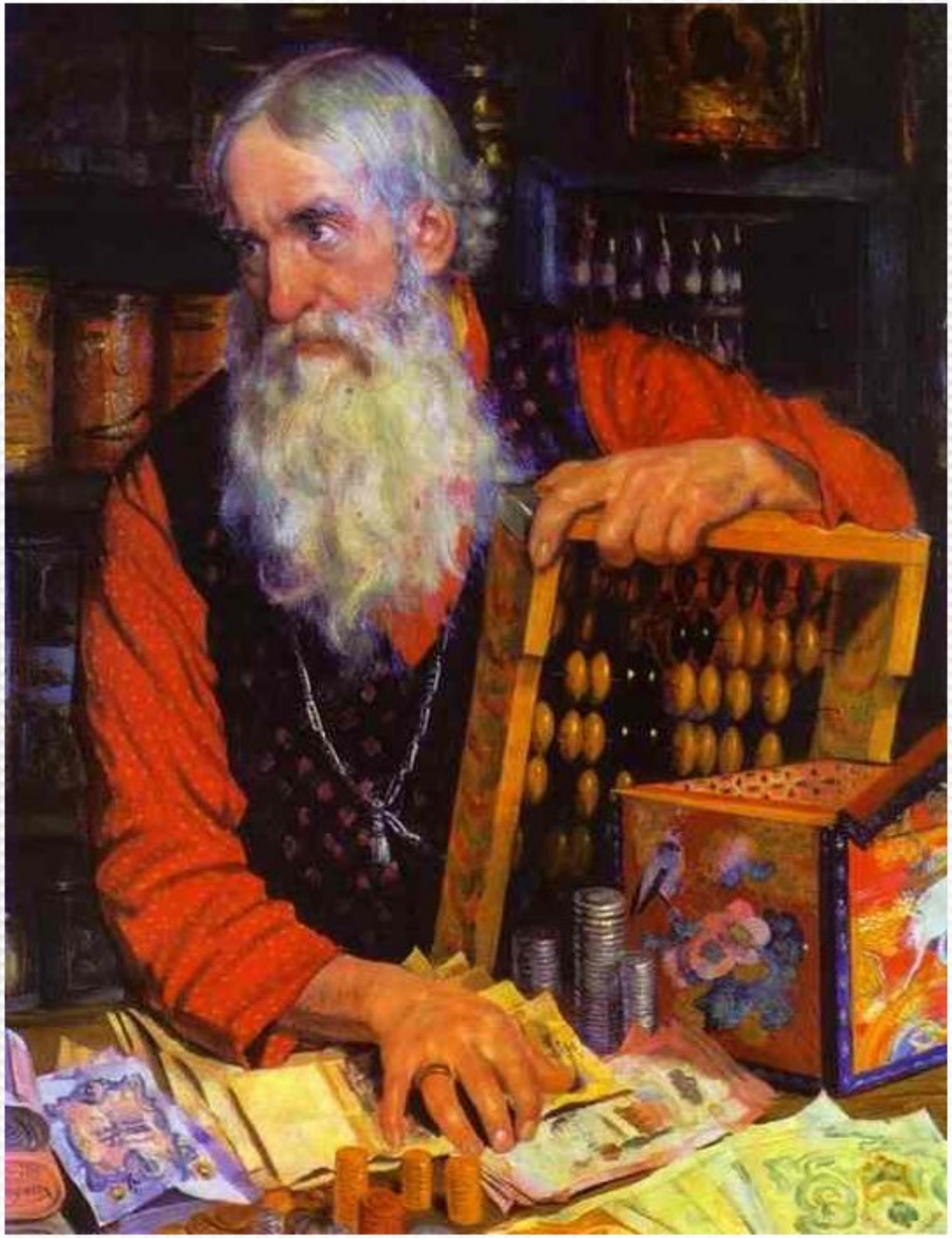
https://youtu.be/LkqiDu1BQXY?t=1m

NA

1930

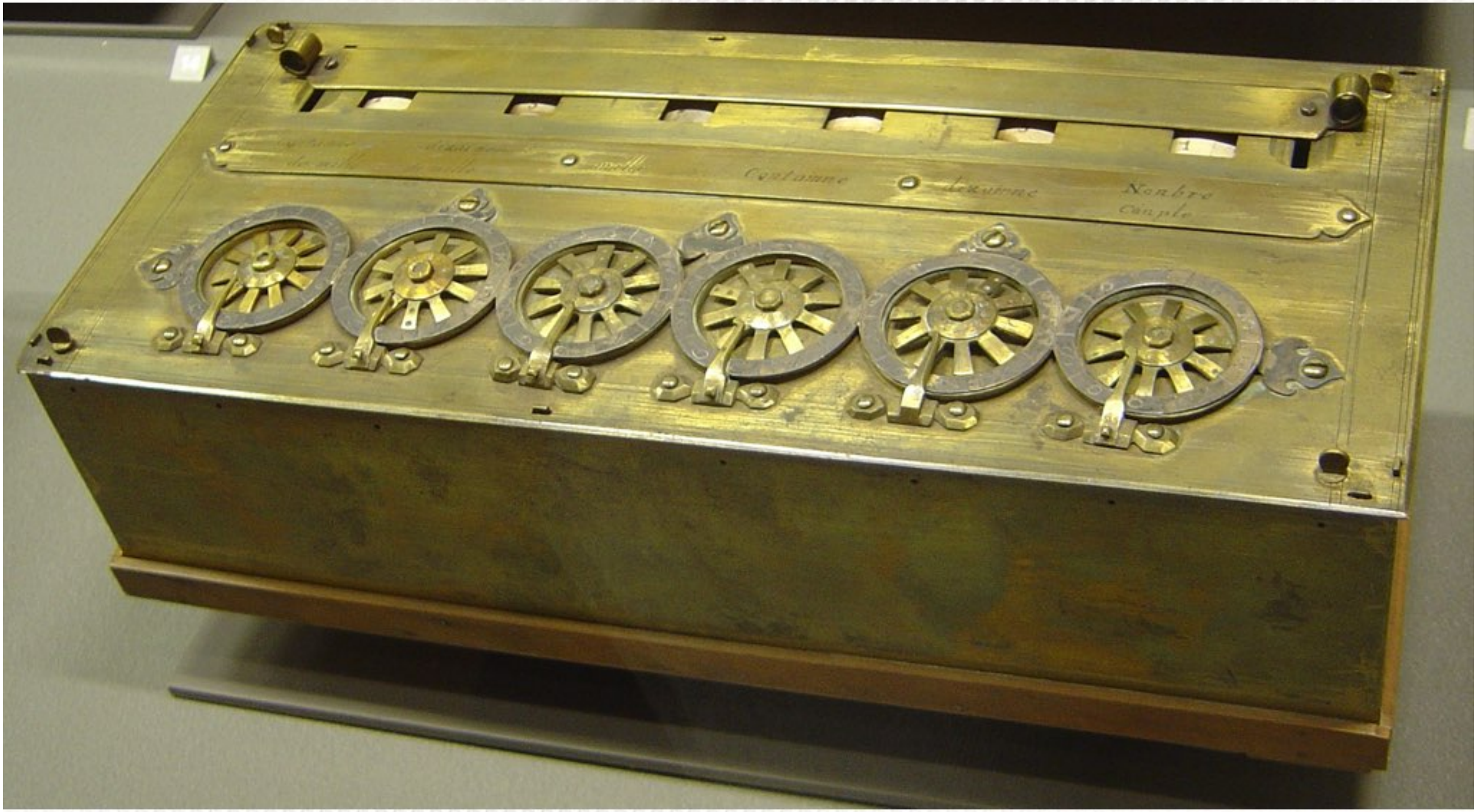
2077

1918



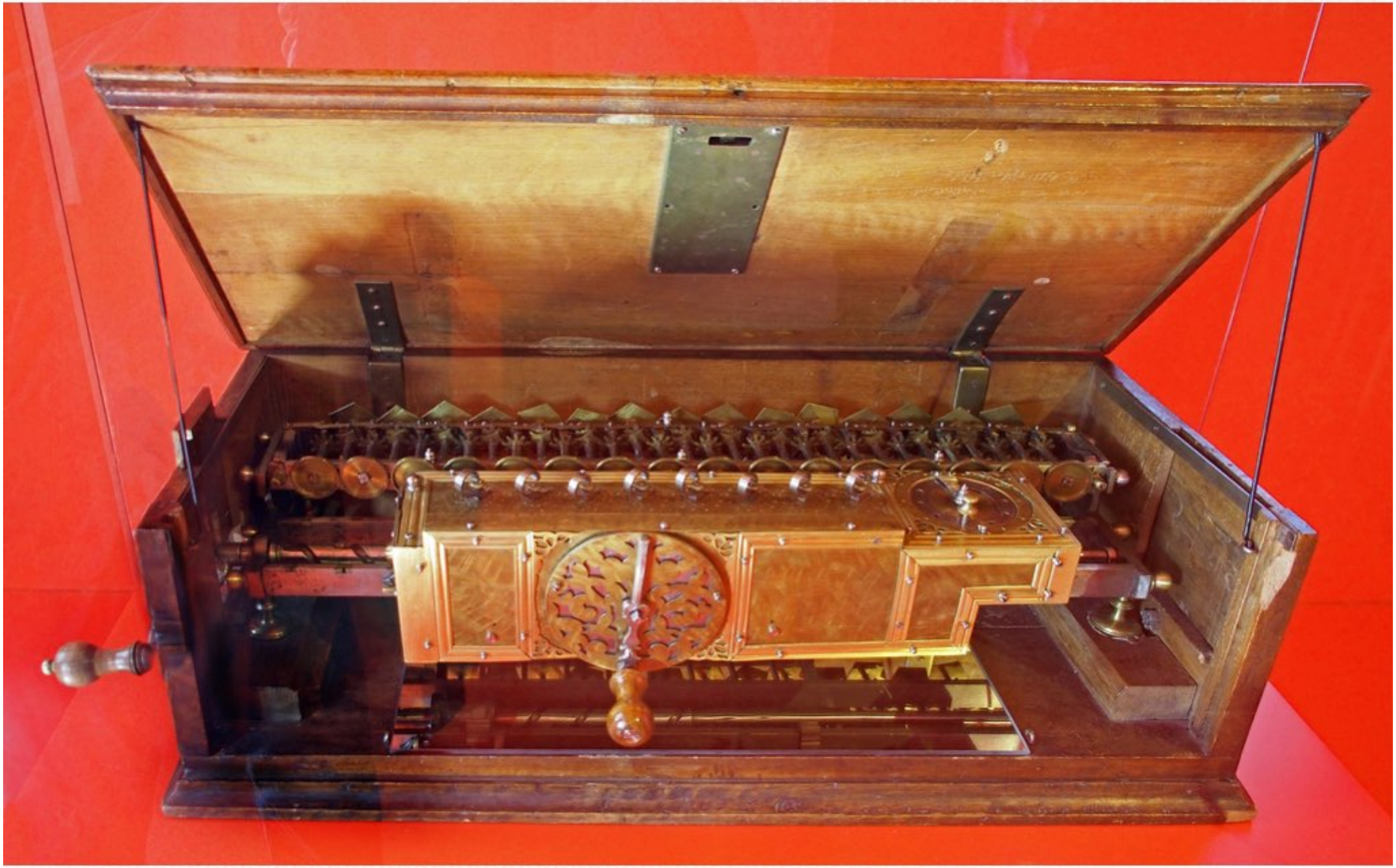
Boris Kustodiev, Merchant, 1918

1642

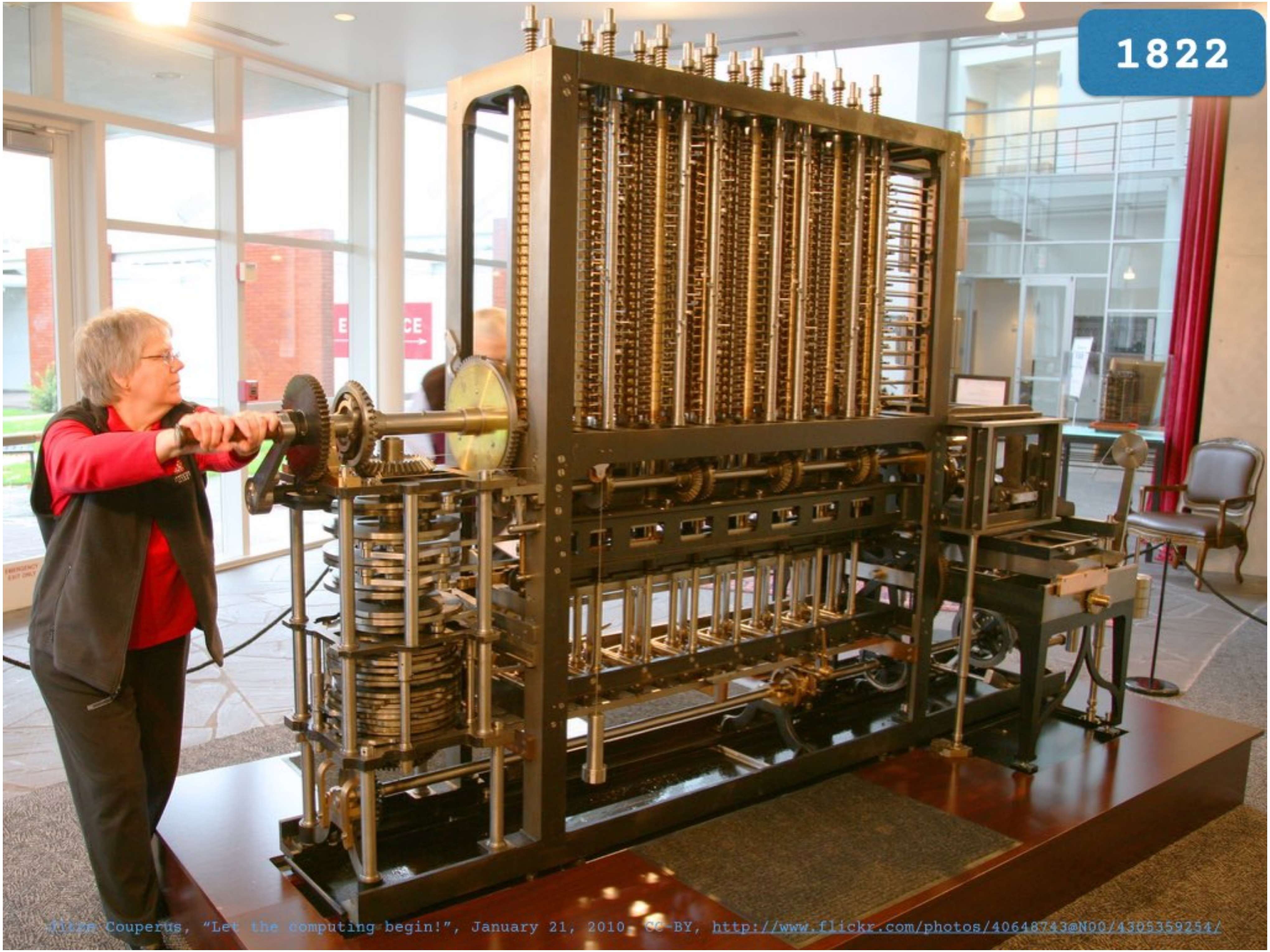


David Monniaux, “Machine à calculer de Blaise Pascal sans sous ni deniers”, https://commons.wikimedia.org/wiki/File:Arts_et_Metiers_Pascaline_dsc03869.jpg, CC-BY-SA

1673

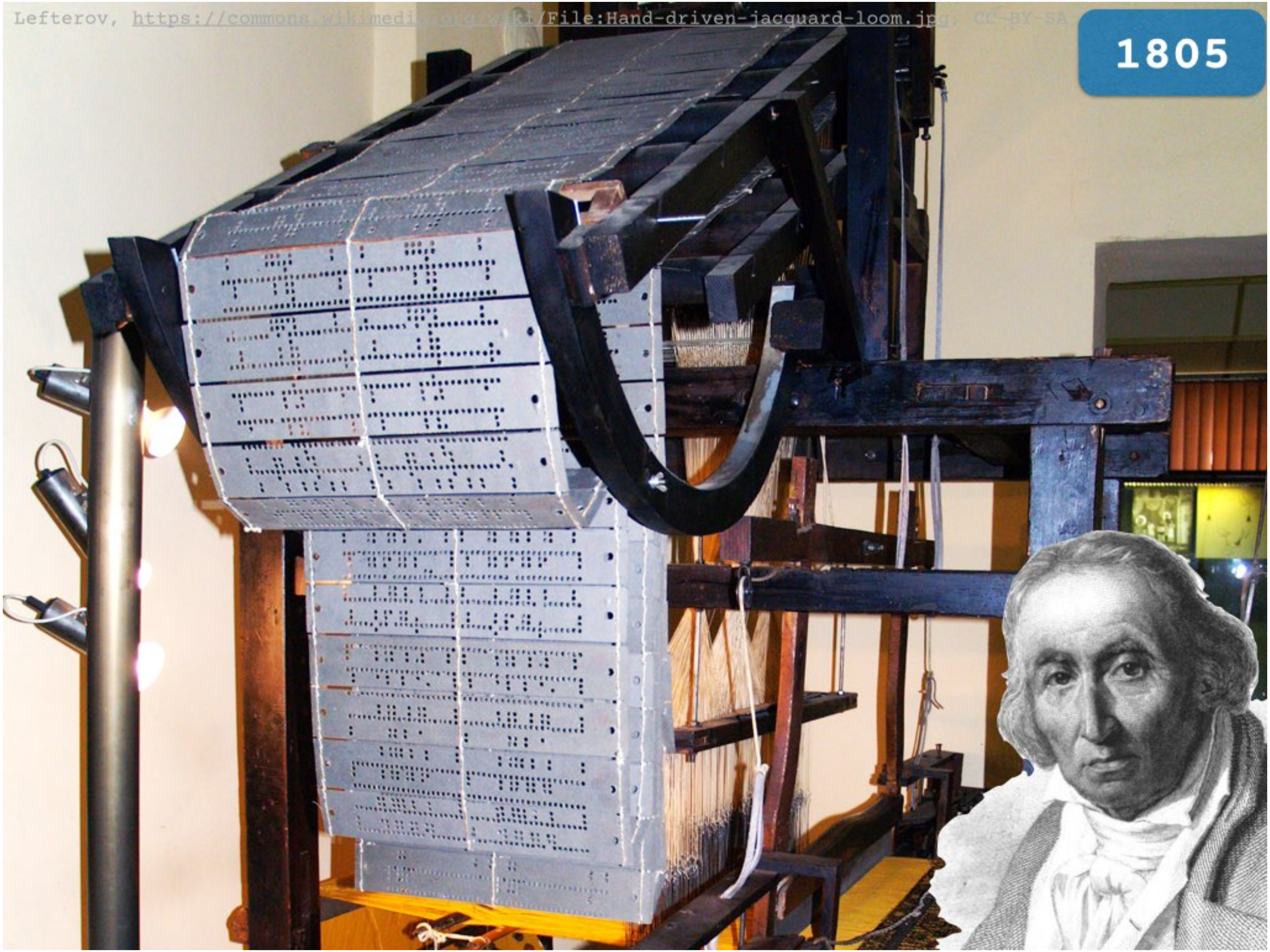


Hajotthu, “Leibniz' Vier-Spezies-Rechenmaschine”,
https://commons.wikimedia.org/wiki/File:Leibniz_Rechenmaschine_(1690).jpg, CC-BY



1822

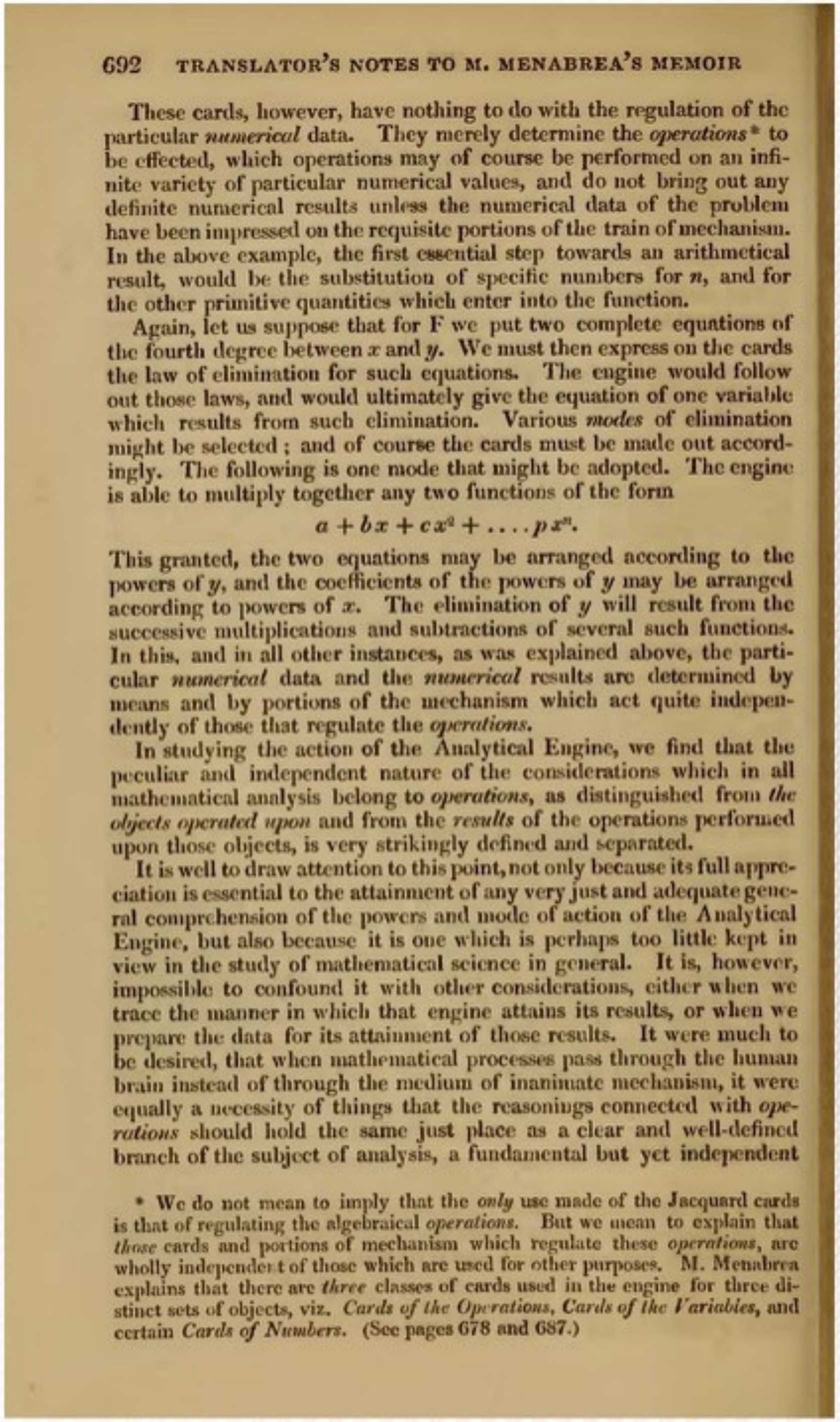
Jitze Couperus, “Let the computing begin!”, January 21, 2010, CC-BY, http://www.flickr.com/photos/40648743@N00/4305359254/



Lefterov, https://commons.wikimedia.org/wiki/File:Hand-driven-jacquard-loom.jpg, CC-BY-SA

1805

1843



692 TRANSLATOR'S NOTES TO M. MENABREA'S MEMOIR

These cards, however, have nothing to do with the regulation of the particular *numerical* data. They merely determine the *operations*[*] to be effected, which operations may of course be performed on an infinite variety of particular numerical values, and do not bring out any definite numerical results unless the numerical data of the problem have been impressed on the requisite portions of the train of mechanism. In the above example, the first essential step towards an arithmetical result, would be the substitution of specific numbers for *n*, and for the other primitive quantities which enter into the function.

Again, let us suppose that for F we put two complete equations of the fourth degree between *x* and *y*. We must then express on the cards the law of elimination for such equations. The engine would follow out those laws, and would ultimately give the equation of one variable which results from such elimination. Various *modes* of elimination might be selected ; and of course the cards must be made out accordingly. The following is one mode that might be adopted. The engine is able to multiply together any two functions of the form

$$a + bx + cx^2 + \dots . px^n.$$

This granted, the two equations may be arranged according to the powers of *y*, and the coefficients of the powers of *y* may be arranged according to powers of *x*. The elimination of *y* will result from the successive multiplications and subtractions of several such functions. In this, and in all other instances, as was explained above, the particular *numerical* data and the *numerical* results are determined by means and by portions of the mechanism which act quite independently of those that regulate the *operations*.

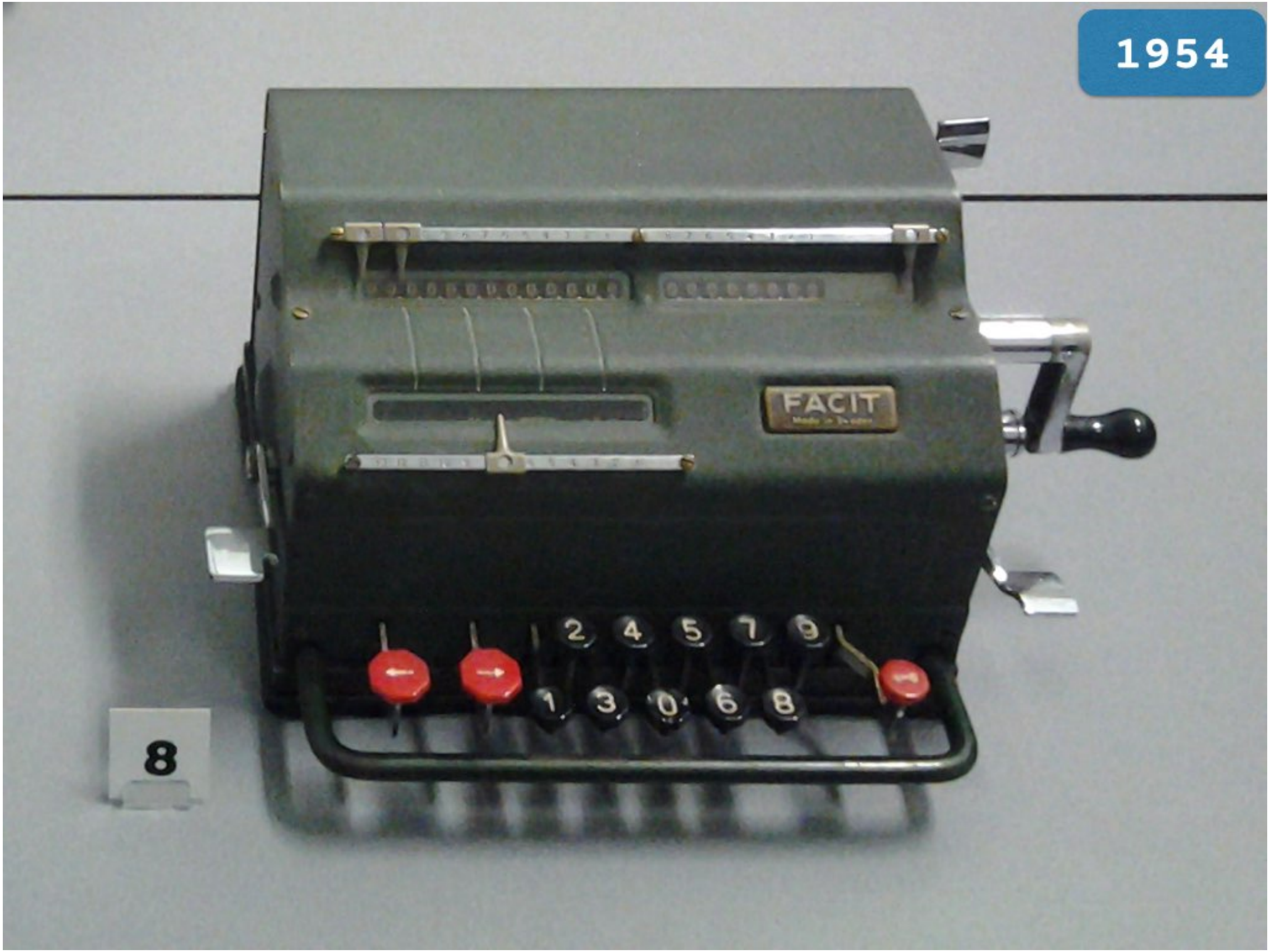
In studying the action of the Analytical Engine, we find that the peculiar and independent nature of the considerations which in all mathematical analysis belong to *operations*, as distinguished from *the objects operated upon* and from the *results* of the operations performed upon those objects, is very strikingly defined and separated.

It is well to draw attention to this point, not only because its full appreciation is essential to the attainment of any very just and adequate general comprehension of the powers and mode of action of the Analytical Engine, but also because it is one which is perhaps too little kept in view in the study of mathematical science in general. It is, however, impossible to confound it with other considerations, either when we trace the manner in which that engine attains its results, or when we prepare the data for its attainment of those results. It were much to be desired, that when mathematical processes pass through the human brain instead of through the medium of inanimate mechanism, it were equally a necessity of things that the reasonings connected with *operations* should hold the same just place as a clear and well-defined branch of the subject of analysis, a fundamental but yet independent

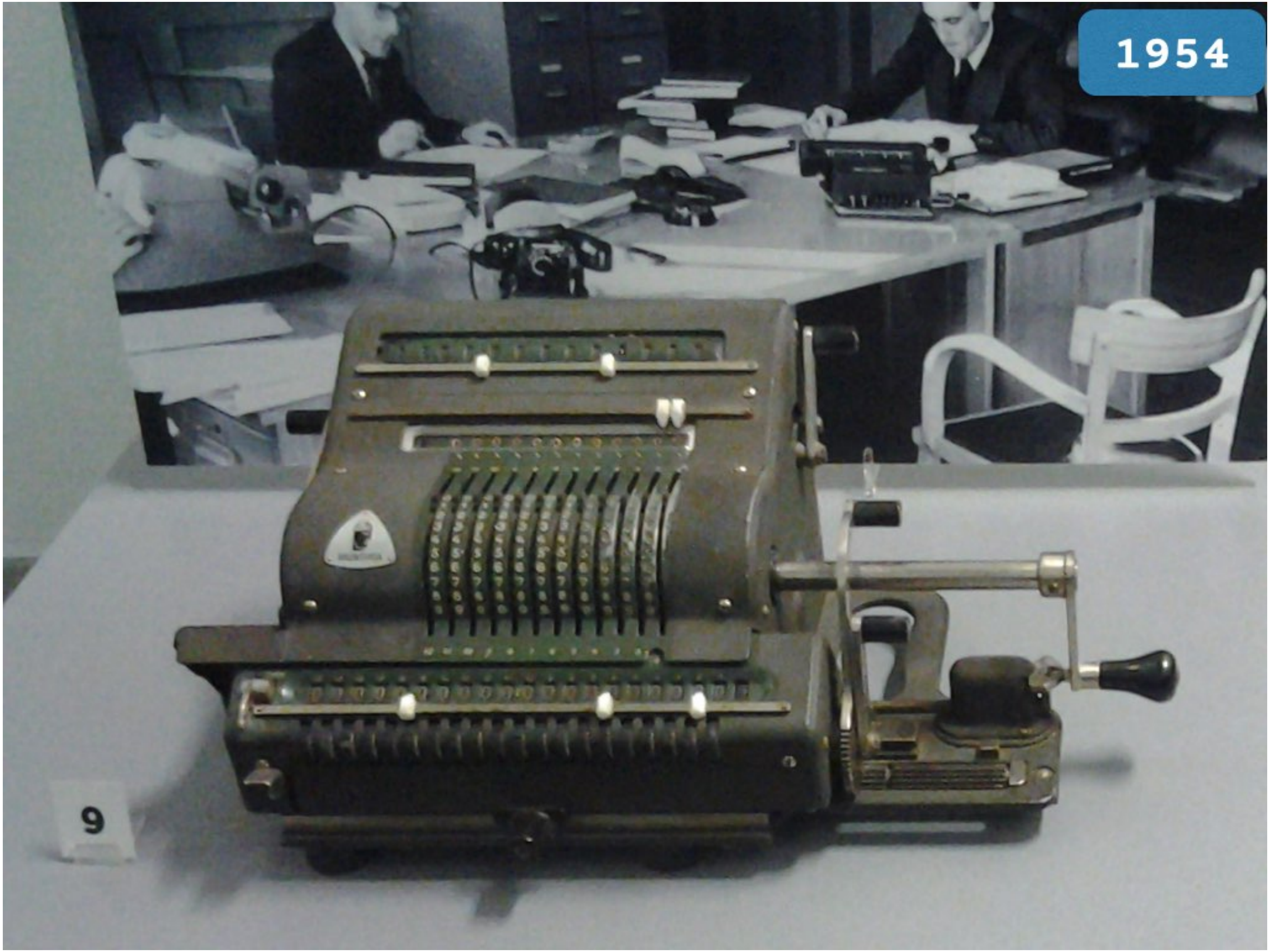
[*] We do not mean to imply that the *only* use made of the Jacquard cards is that of regulating the algebraical *operations*. But we mean to explain that *those* cards and portions of mechanism which regulate these *operations*, are wholly independent of those which are used for other purposes. M. Menabrea explains that there are *three* classes of cards used in the engine for three distinct sets of objects, viz. *Cards of the Operations*, *Cards of the Variables*, and certain *Cards of Numbers*. (See pages 678 and 687.)

Ada Lovelace (1815-1852)

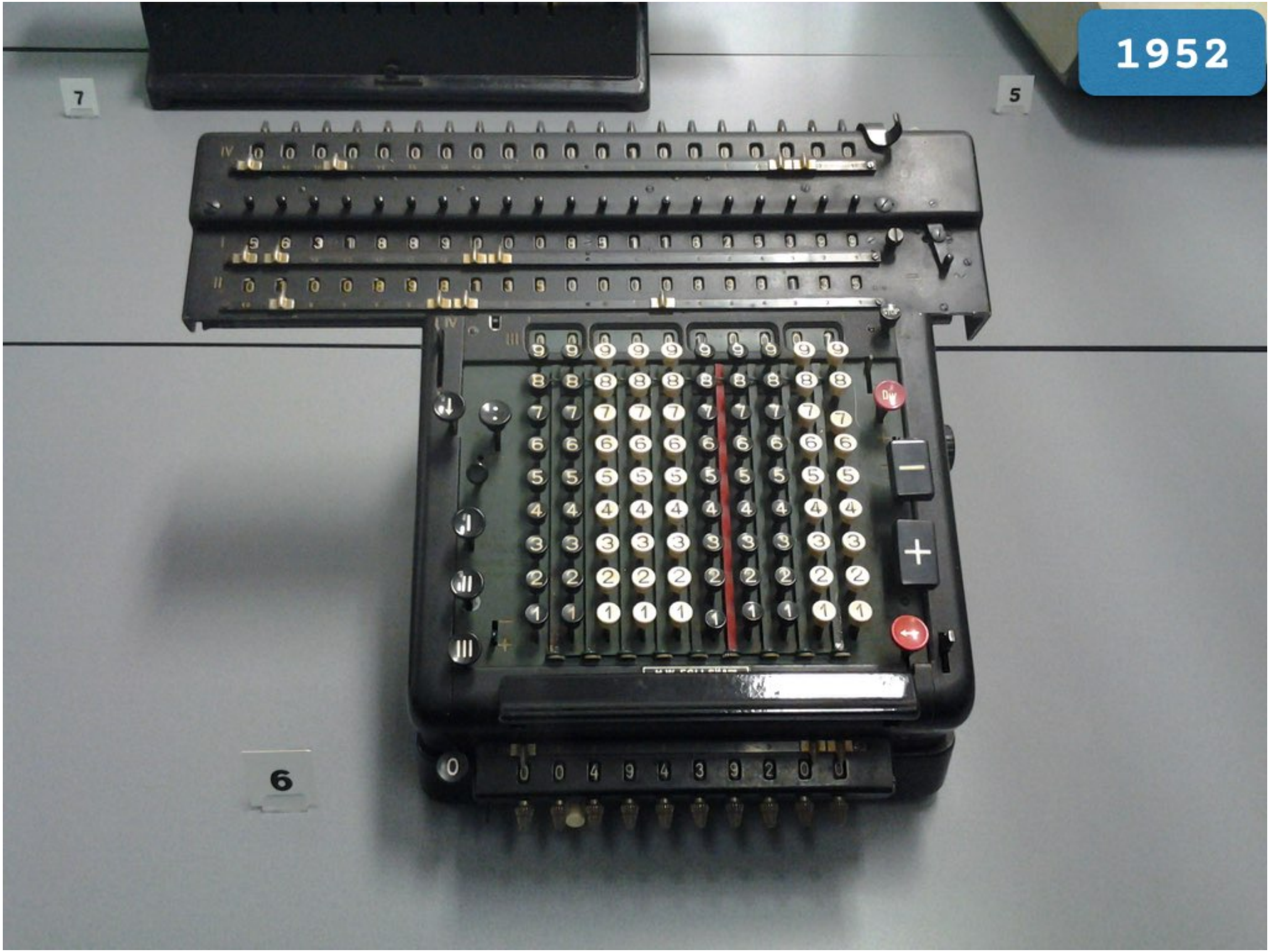
"Sketch of the Analytical Engine Invented by Charles Babbage, Esq.," Scientific Memoirs:3, "Notes by the Translator", https://en.wikisource.org/wiki/Page%3AScientific_Memoirs%2C_Vol._3_(1843).djvu/702



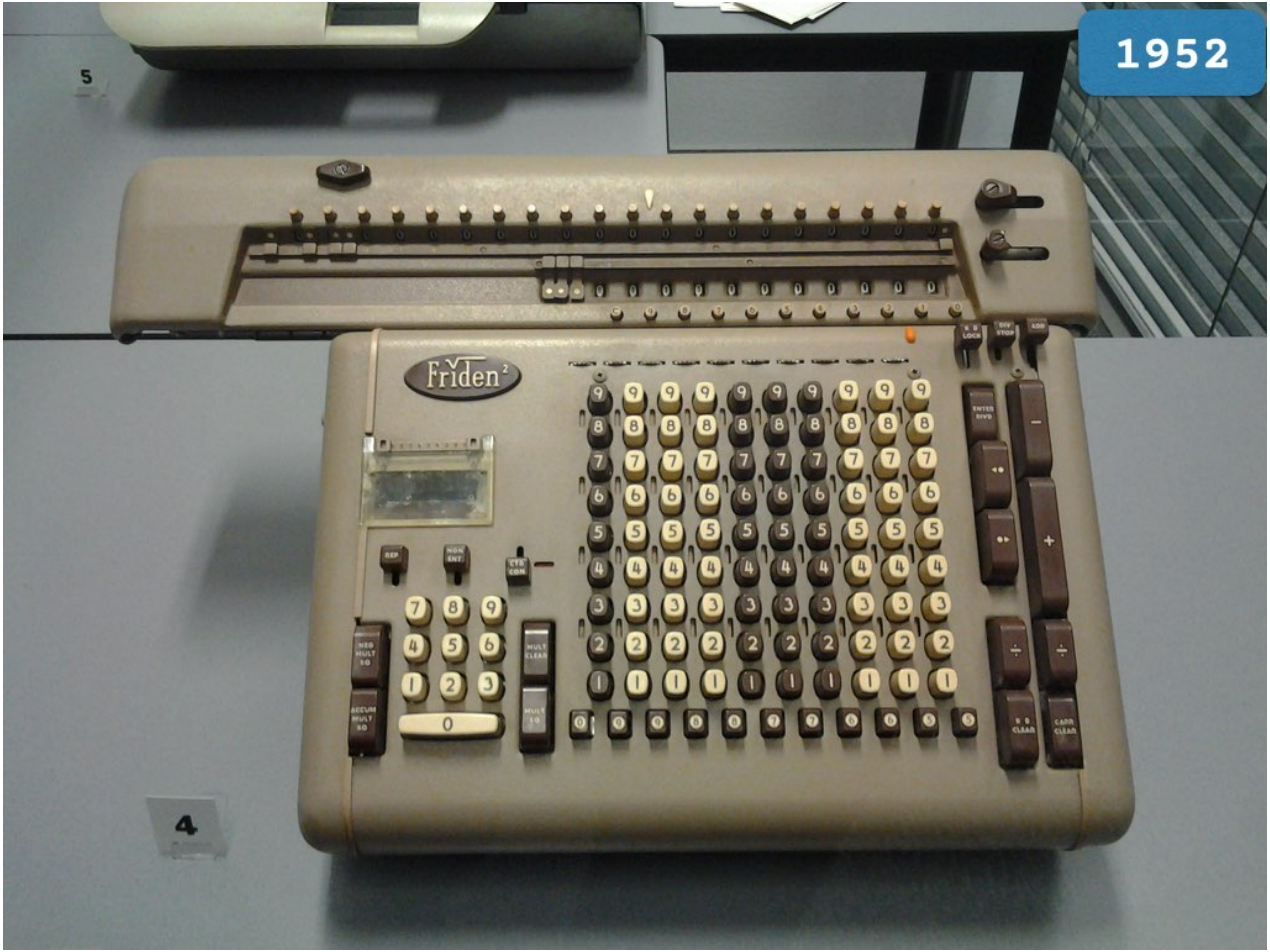
1954



1954



1952



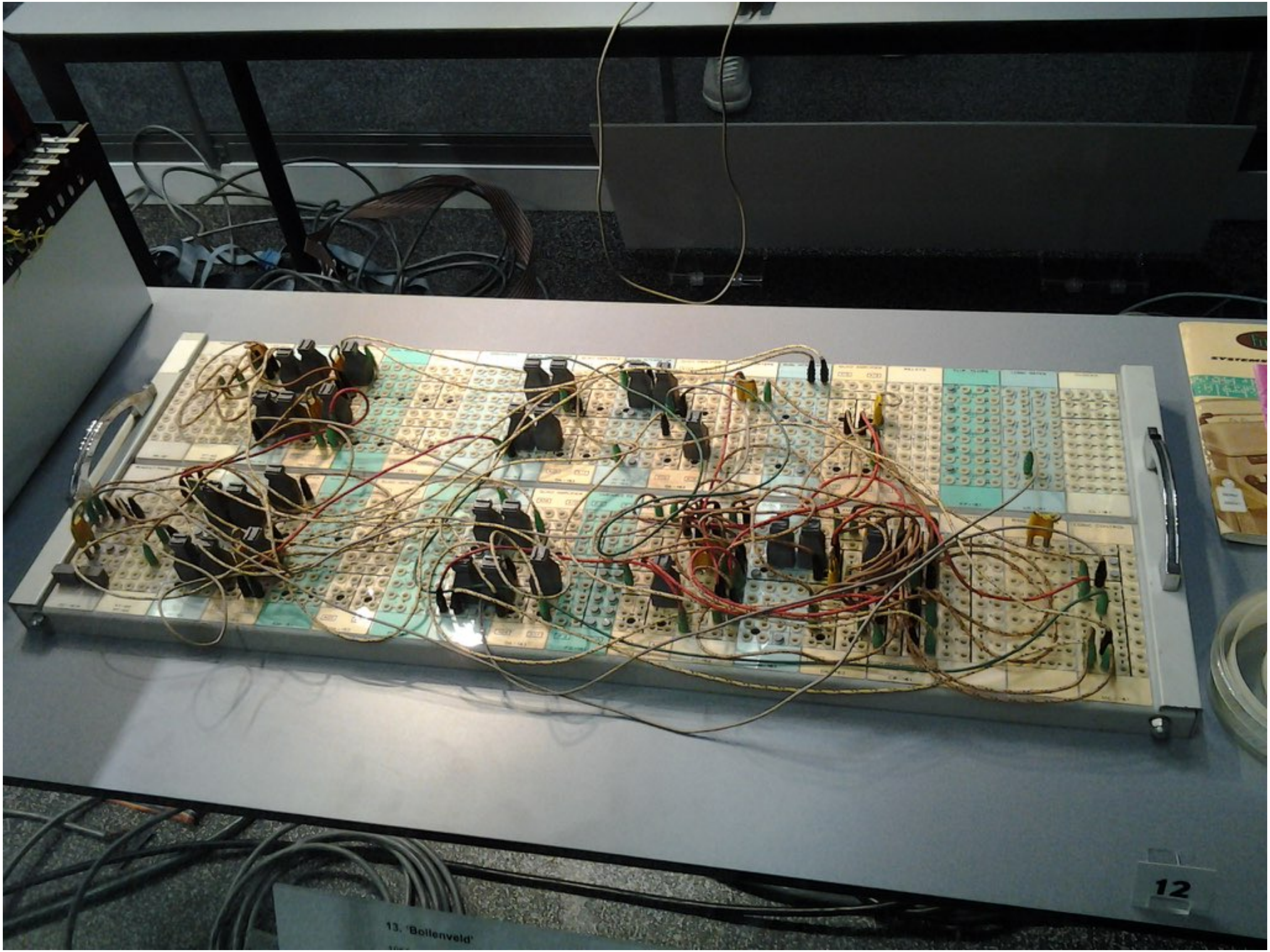
1952

photo credit: http://academictree.org/physics/peopleinfo.php?pid=25212

195x

Adriaan J. van Wijngaarden (1916-1987)

photo credit: http://www.kennislink.nl/publicaties/rekenmeisjes-en-rekentuig


12
13. 'Bollenveld'

Adele Katz Goldstine (1920-1964)

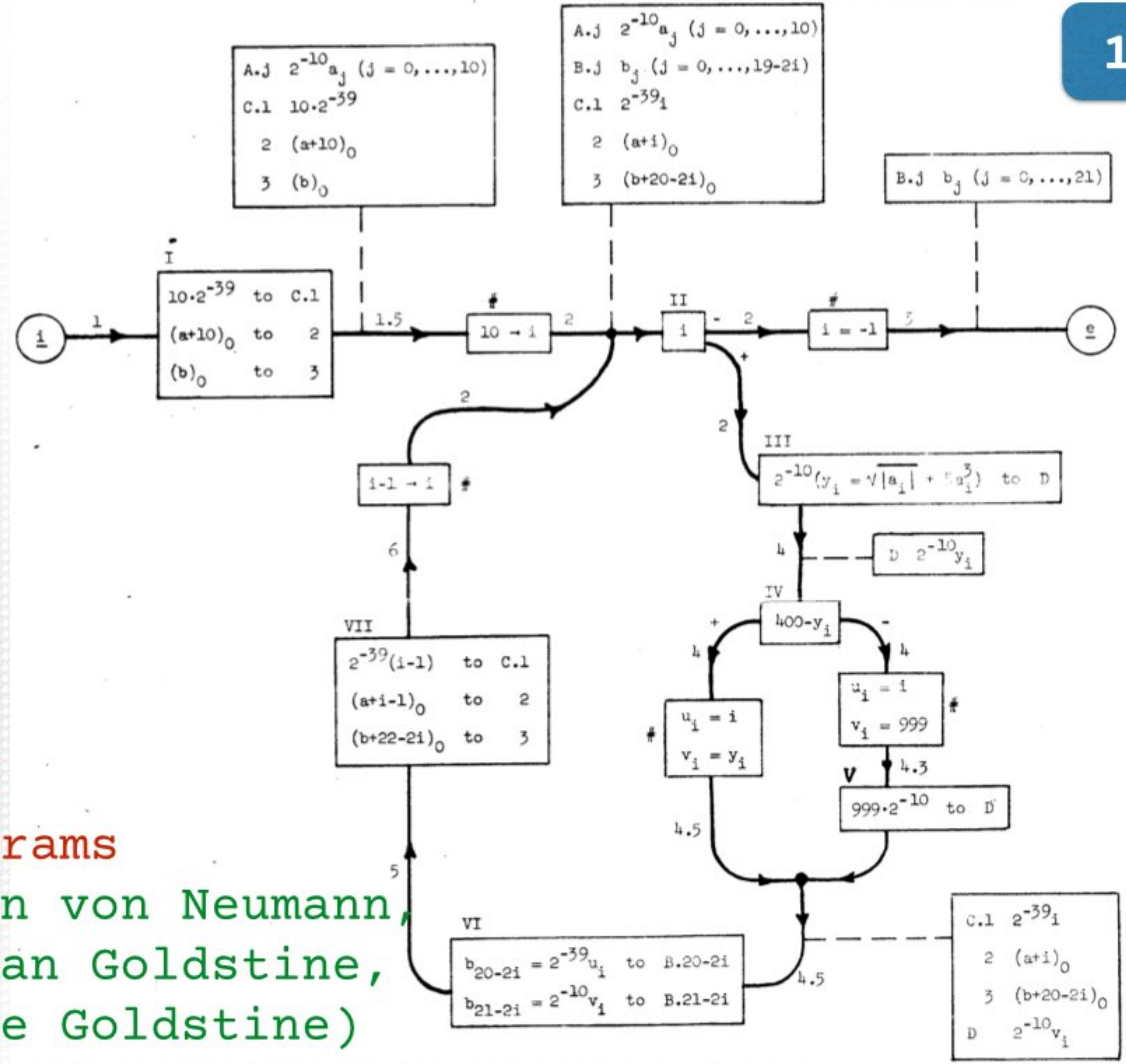
1946

photo credit: http://nurdangolbek.blogspot.com/2012/08/teknoloji-tarihindeki-15-onemli-kadn.html

1945

John von Neumann (1903–1957)

1947



Flow Diagrams (John von Neumann, Herman Goldstine, Adele Goldstine)

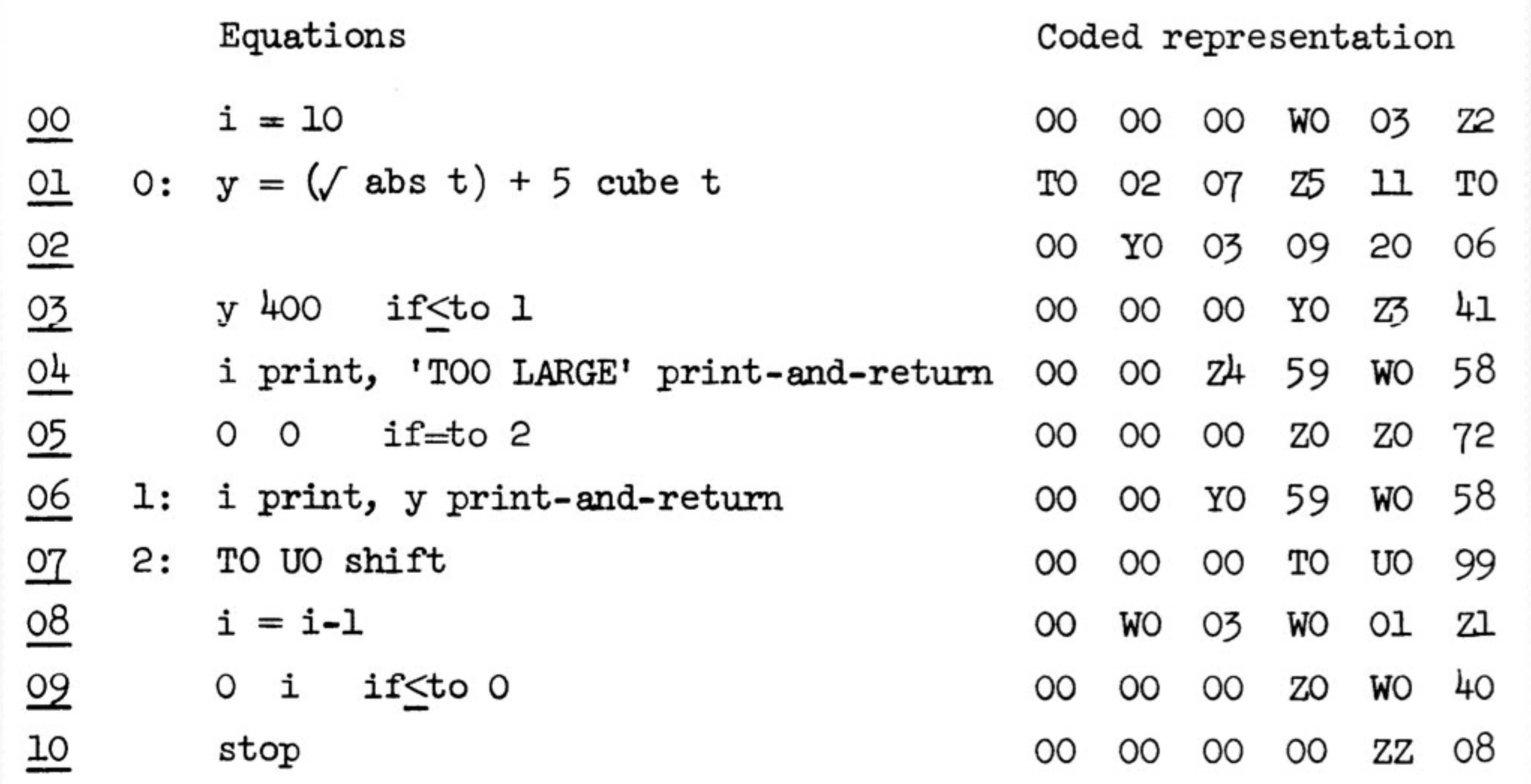
1950

"the technique of program composition can be mechanised"

Haskell Brooks Curry (1900-1982)

John Mauchly, William Schmitt

1950



```
         Equations                                     Coded representation

00       i = 10                                        00  00  00  WO  03  Z2
01   0:  y = (√ abs t) + 5 cube t                      TO  02  07  Z5  11  TO
02                                                     00  YO  03  09  20  06
03       y 400   if≤to 1                               00  00  00  YO  Z3  41
04       i print, 'TOO LARGE' print-and-return         00  00  Z4  59  WO  58
05       0  0    if=to 2                               00  00  00  ZO  ZO  72
06   1:  i print, y print-and-return                   00  00  YO  59  WO  58
07   2:  TO UO shift                                   00  00  00  TO  UO  99
08       i = i-1                                       00  WO  03  WO  01  Z1
09       0  i   if≤to 0                                00  00  00  ZO  WO  40
10       stop                                          00  00  00  00  ZZ  08
```

Short Code

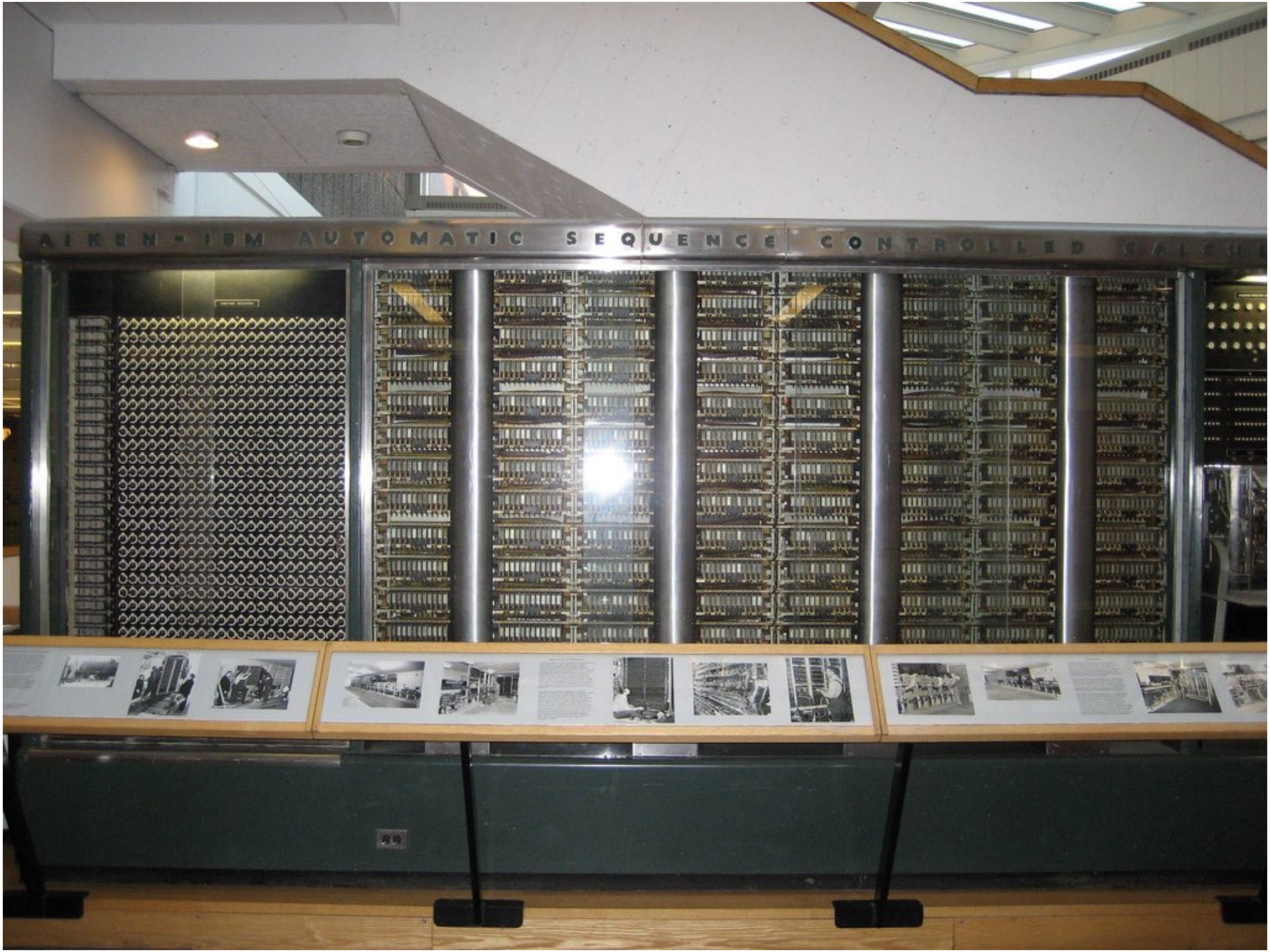
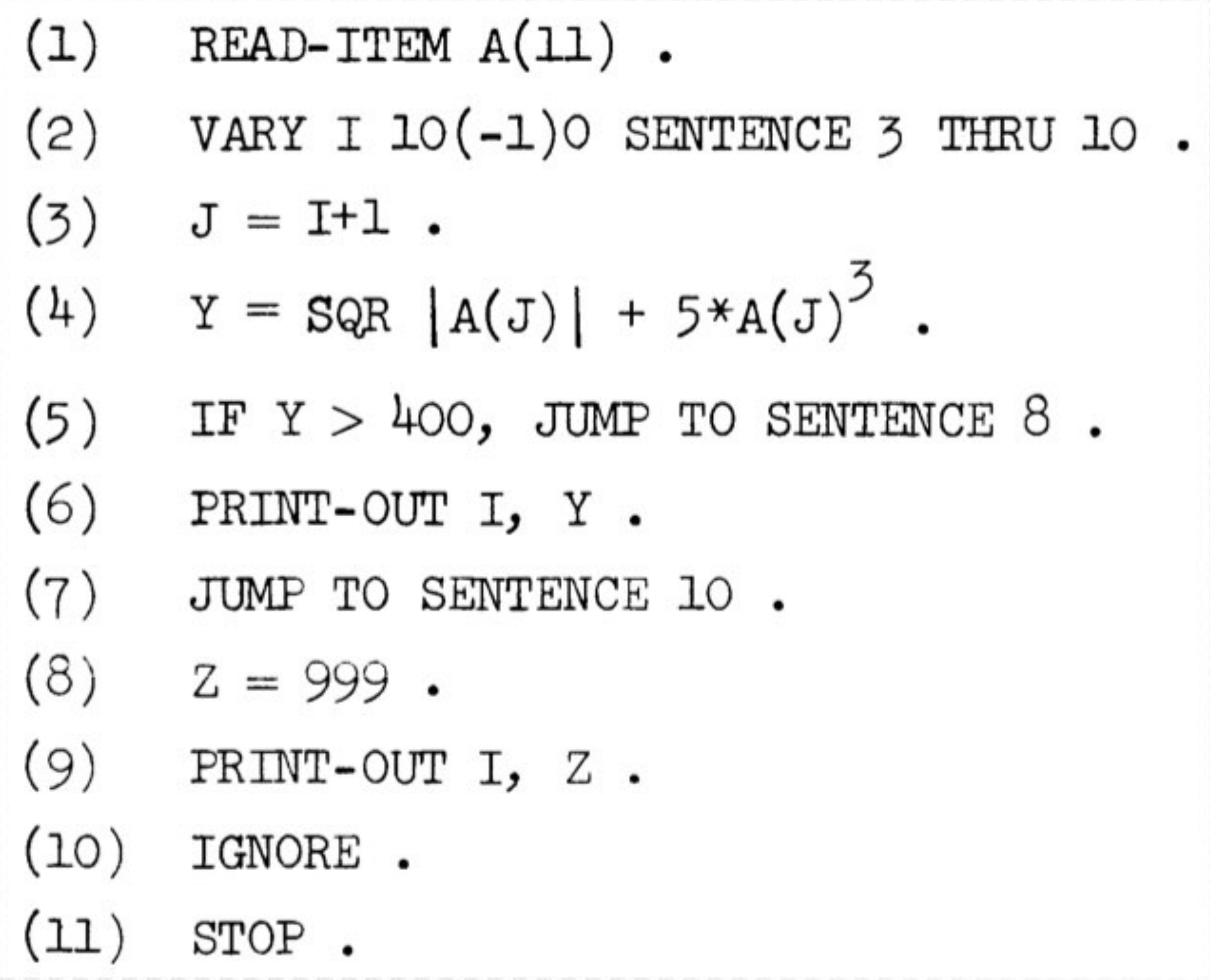

AIKEN - IBM AUTOMATIC SEQUENCE CONTROLLED CALCU

photo credit: http://techie.com/amazing-grace/ / personal.psu.edu

Grace Murray Hopper (1906-1992)

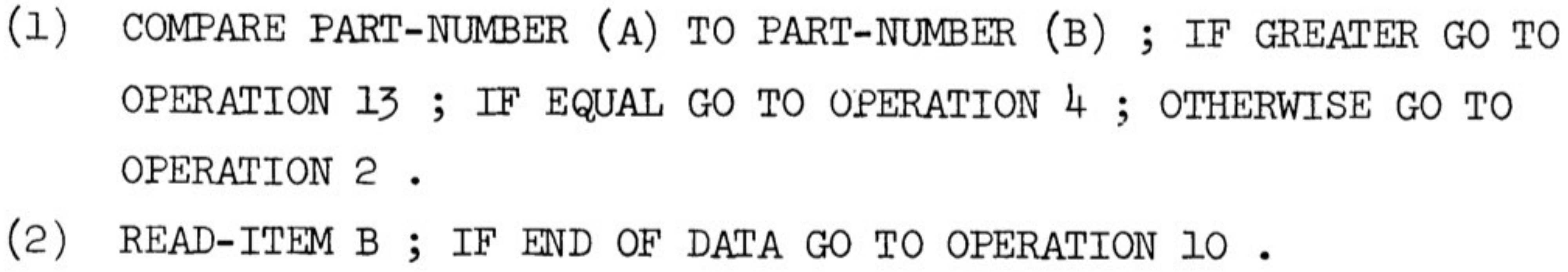
1957



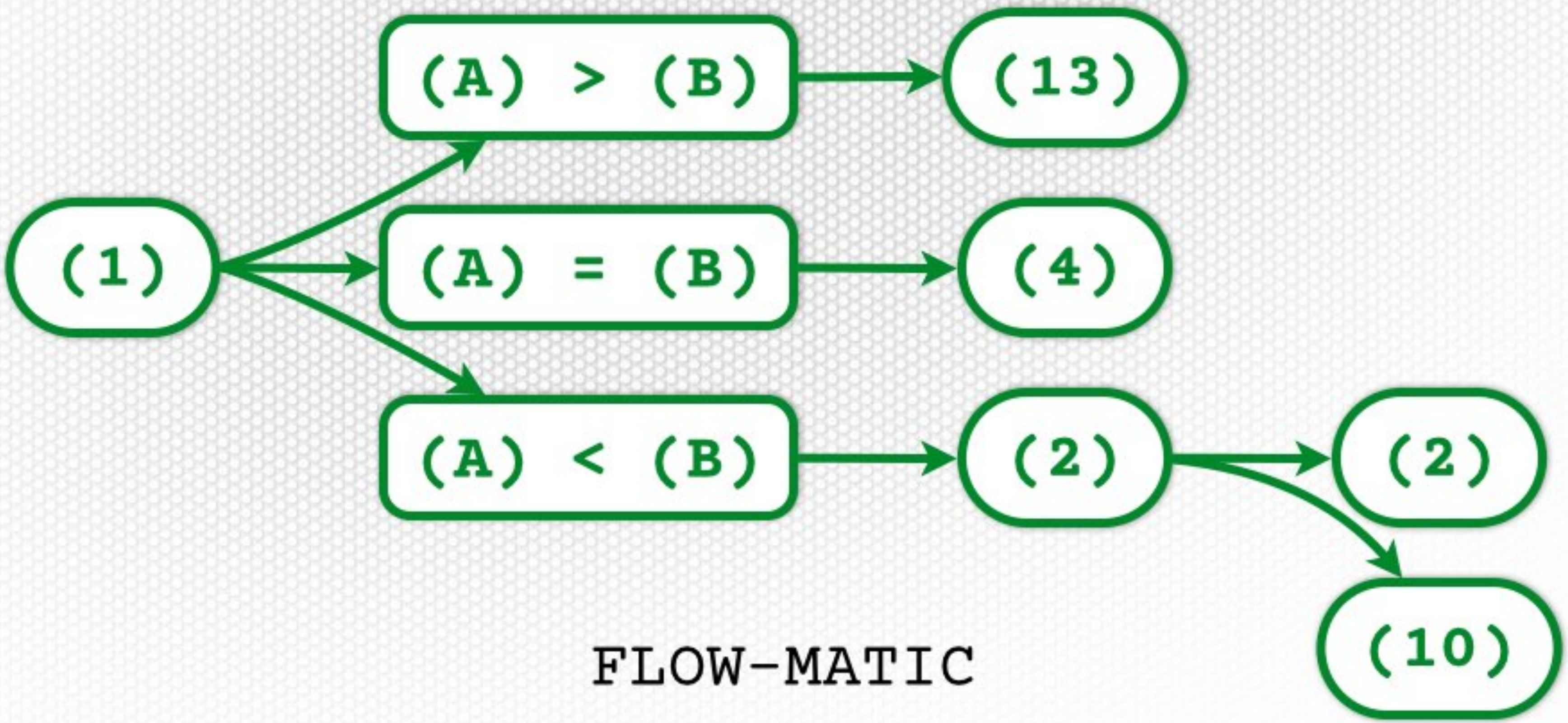
```
(1)   READ-ITEM A(11) .
(2)   VARY I 10(-1)0 SENTENCE 3 THRU 10 .
(3)   J = I+1 .
                                    3
(4)   Y = SQR |A(J)| + 5*A(J)  .
(5)   IF Y > 400, JUMP TO SENTENCE 8 .
(6)   PRINT-OUT I, Y .
(7)   JUMP TO SENTENCE 10 .
(8)   Z = 999 .
(9)   PRINT-OUT I, Z .
(10)  IGNORE .
(11)  STOP .
```

MATH-MATIC

1958



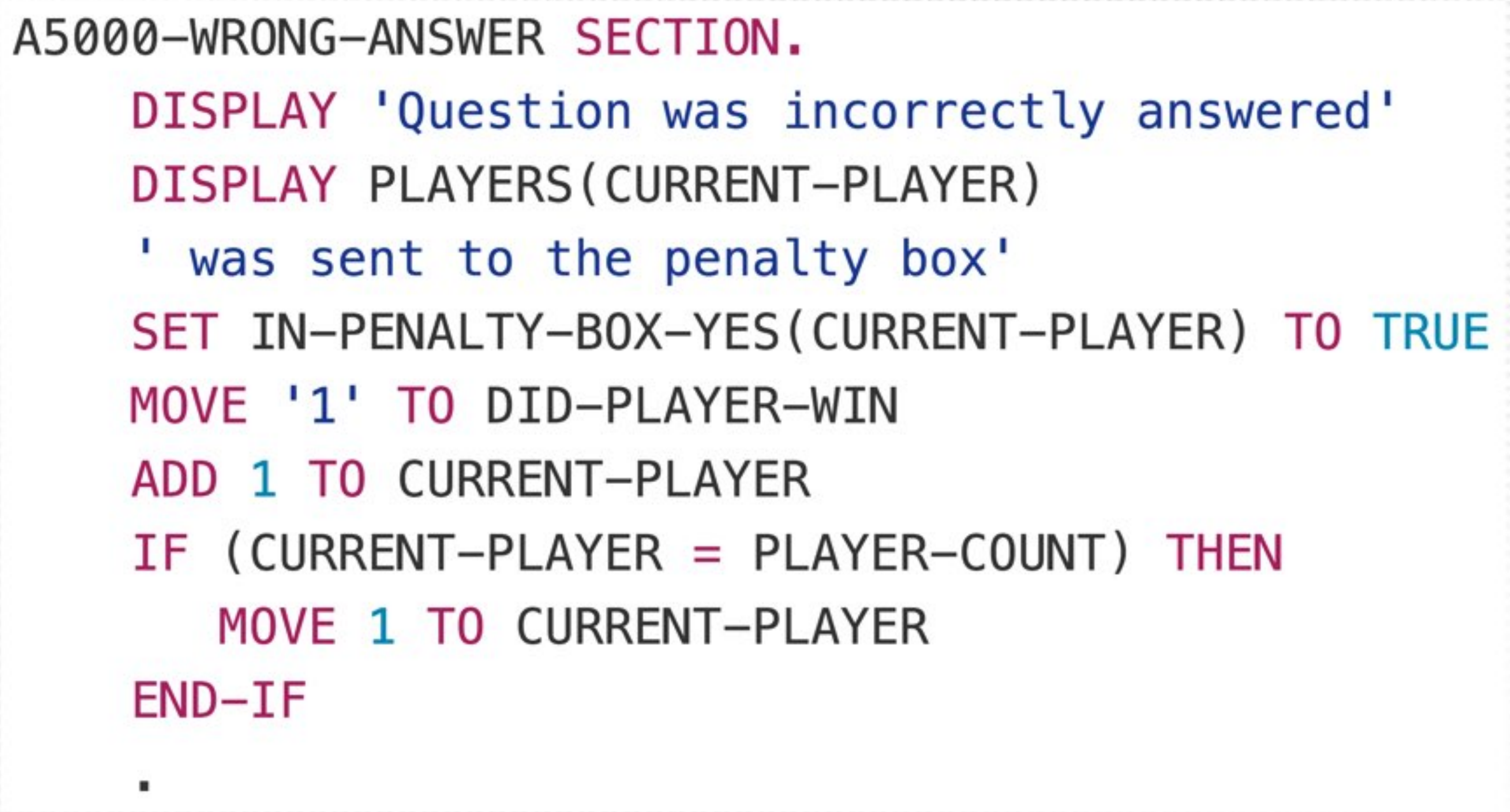
```
(1)  COMPARE PART-NUMBER (A) TO PART-NUMBER (B) ; IF GREATER GO TO
     OPERATION 13 ; IF EQUAL GO TO OPERATION 4 ; OTHERWISE GO TO
     OPERATION 2 .
(2)  READ-ITEM B ; IF END OF DATA GO TO OPERATION 10 .
```



FLOW-MATIC

code by: SLFitch,
https://github.com/jbrains/trivia/blob/master/COBOL/game.cob

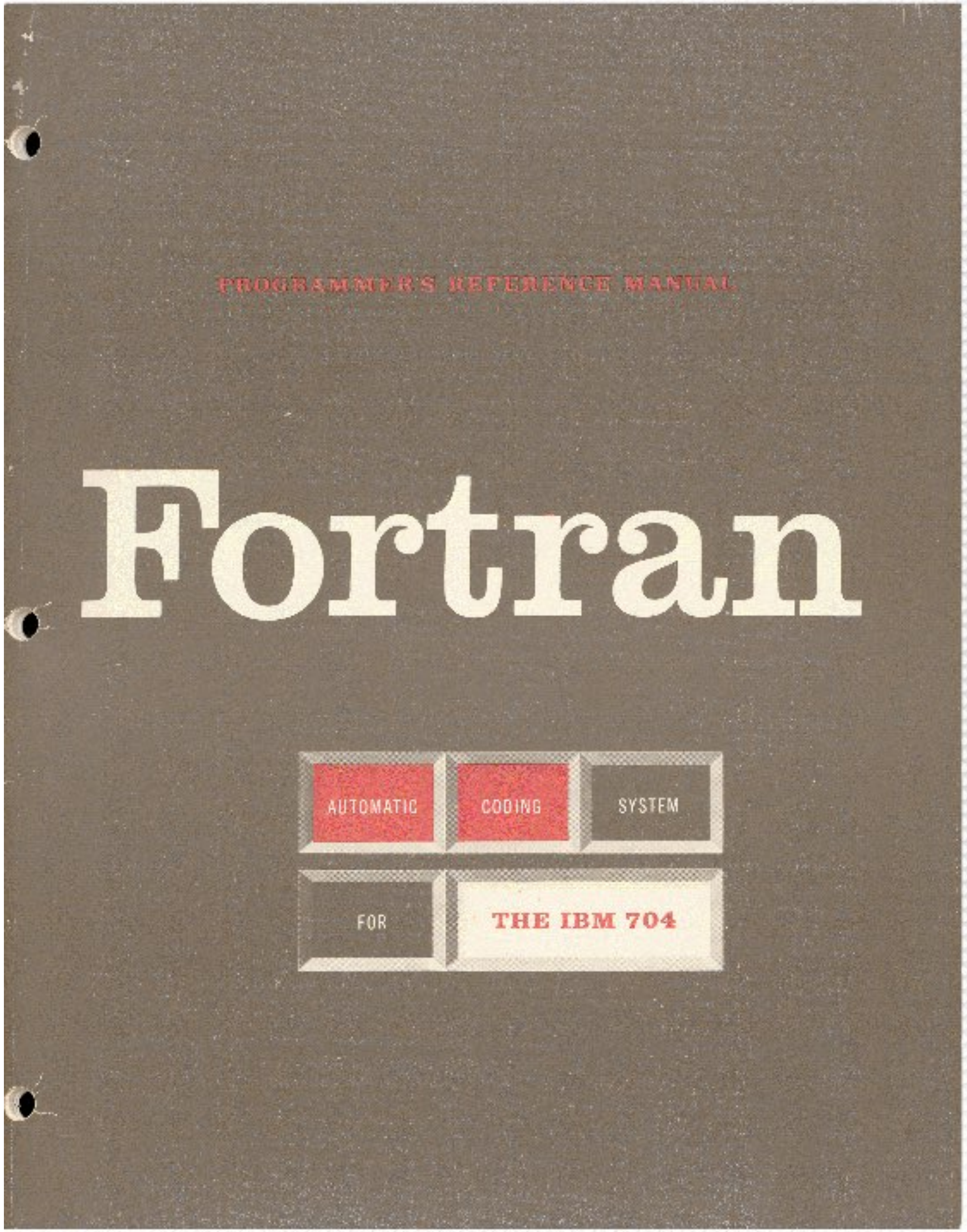
1959



```
A5000-WRONG-ANSWER SECTION.
    DISPLAY 'Question was incorrectly answered'
    DISPLAY PLAYERS(CURRENT-PLAYER)
    ' was sent to the penalty box'
    SET IN-PENALTY-BOX-YES(CURRENT-PLAYER) TO TRUE
    MOVE '1' TO DID-PLAYER-WIN
    ADD 1 TO CURRENT-PLAYER
    IF (CURRENT-PLAYER = PLAYER-COUNT) THEN
       MOVE 1 TO CURRENT-PLAYER
    END-IF
    .
```

COBOL

1957



PROGRAMMER'S REFERENCE MANUAL

Fortran

AUTOMATIC CODING SYSTEM

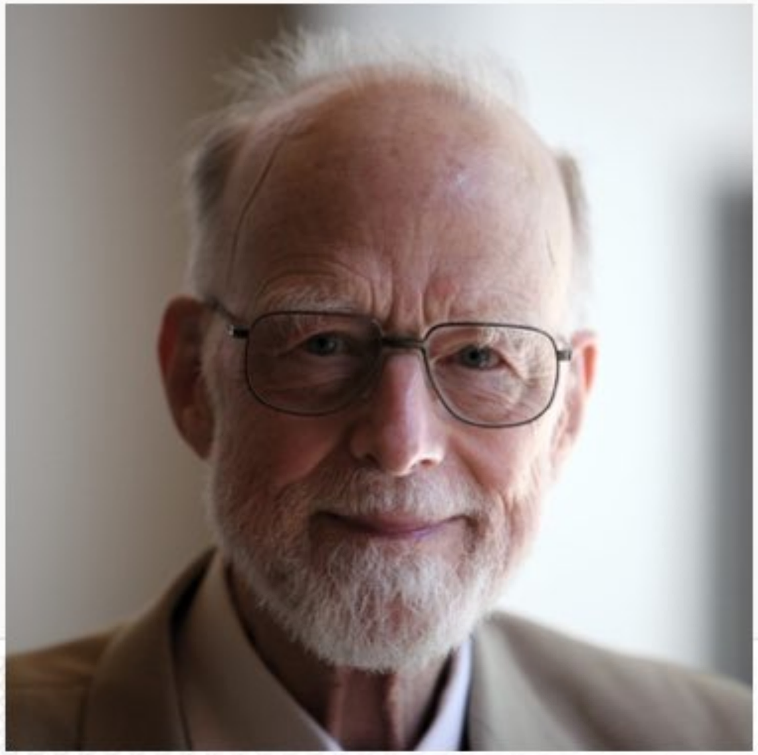
FOR THE IBM 704

John Backus
(1924–2007)

"a language so far ahead of its time that it was not only an improvement on its predecessors but also on nearly all its successors"

1958

```
proc abs max = ([,]real a, ref real y, ref int i, k)real:
comment The absolute greatest element of the matrix a, of size ⌈a by 2⌈a
is transferred to y, and the subscripts of this element to i and k; comment
begin
   real y := 0; i := ⌊a; k := 2⌊a;
   for p from ⌊a to ⌈a do
     for q from 2⌊a to 2⌈a do
       if abs a[p, q] > y then
           y := abs a[p, q];
           i := p; k := q
       fi
     od
   od;
   y
end # abs max #
```



Tony Hoare (b. 1934)

CC-BY-SA; code credit: https://en.wikipedia.org/wiki/ALGOL#ALGOL_68;
photo credit: Rama, Sir Charles Antony Richard Hoare, https://commons.wikimedia.org/wiki/File:Sir_Tony_Hoare_IMG_5125.jpg

1964

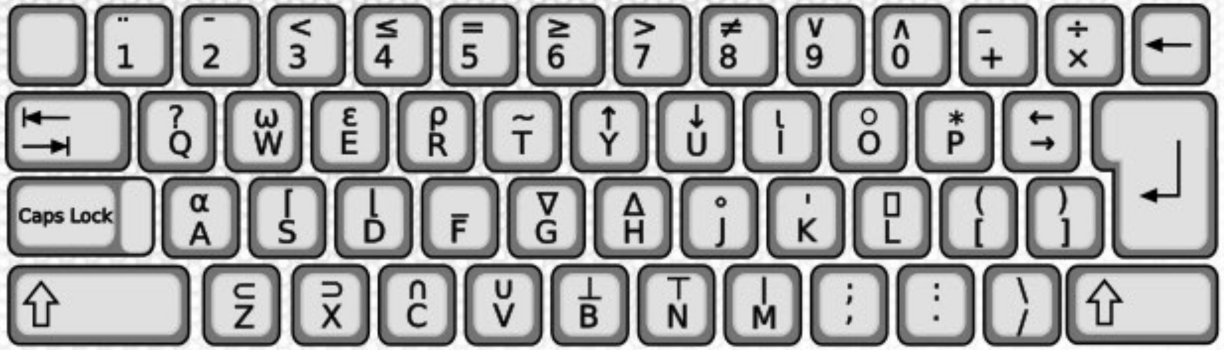
APL

Kenneth E. Iverson
(1920-2004)

```
(~R∊R∘.×R)/R←1↓⍳R
```

```
⎕←{ω/⍨~{ω∨≠\ω}ω∊'<>'}txt
```

```
life←{↑1 ω∨.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂ω}
```


¨ 1 ¯ 2 < 3 ≤ 4 = 5 ≥ 6 > 7 ≠ 8 ∨ 9 ∧ 0 - + ÷ × ←
? Q ⍵ W ∊ E ⍴ R ~ T ↑ Y ↓ U ⍳ I ○ O * P ← →
Caps Lock ⍺ A ⌈ S ⌊ D _ F ∇ G ∆ H ∘ J ' K ⎕ L ([ )]
⊂ Z ⊃ X ∩ C ∪ V ⊥ B ⊤ N | M ; ; : : \ /

APL keyboard by Rursus https://commons.wikimedia.org/wiki/File:APL-keybd2.svg CC-BY-SA
photo from http://archive.vector.org.uk/art10002990 (Michel Dumontier, Hommage à Ken Iverson)
Quote from http://www.azquotes.com/quote/1119653

1964

Kenneth E. Iverson
(1920-2004)

APL

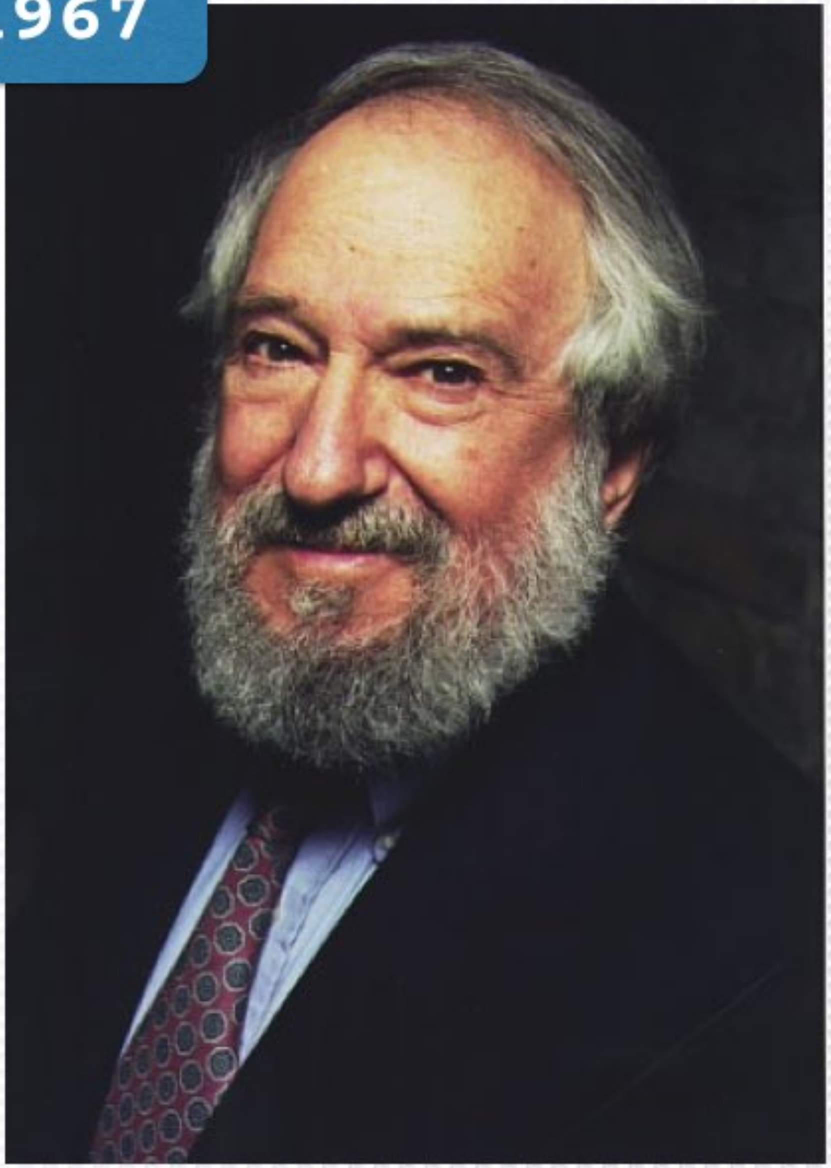
```
(~R∊R∘.×R)/R←1↓⍳R
```

```
⎕←{⍵/⍨~{⍵∨≠\⍵}⍵∊'<>'}txt
```

> "It is important to distinguish the difficulty of describing and learning a piece of notation from the difficulty of mastering its implications. [...] The very suggestiveness of a notation may make it seem harder to learn because of the many properties it suggests for exploration"

photo from http://archive.vector.org.uk/art10002990 (Michel Dumontier, Hommage à Ken Iverson)
Quote from http://www.azquotes.com/quote/1119653

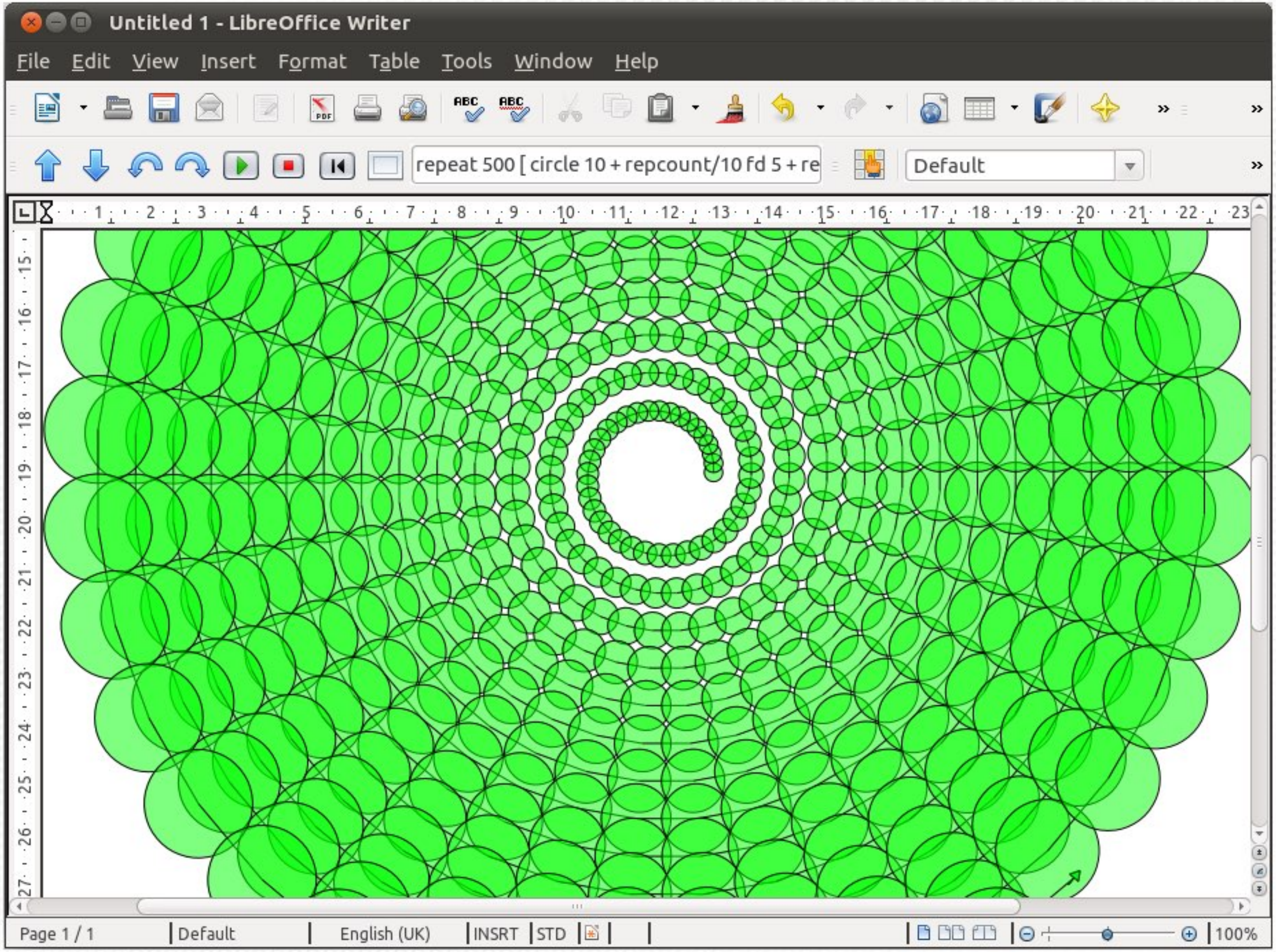
1967



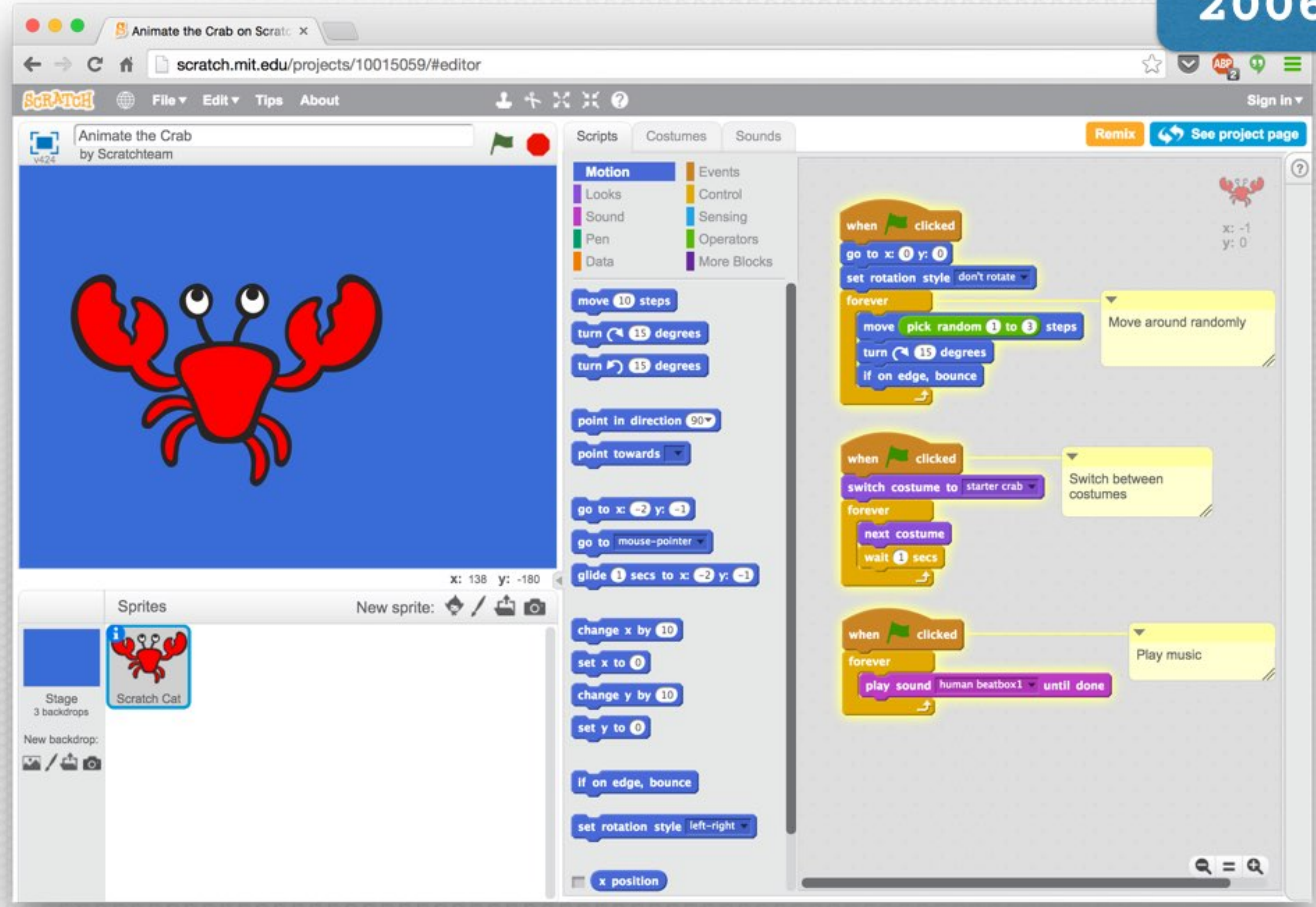
Seymour Papert
(b. 1928)

1974

Radia Perlman
(b. 1951)



2006



```
when [flag] clicked
go to x: 0 y: 0
set rotation style [don't rotate]
forever
    move (pick random 1 to 3) steps
    turn ↻ 15 degrees
    if on edge, bounce
```

Move around randomly

```
when [flag] clicked
switch costume to [starter crab]
forever
    next costume
    wait 1 secs
```

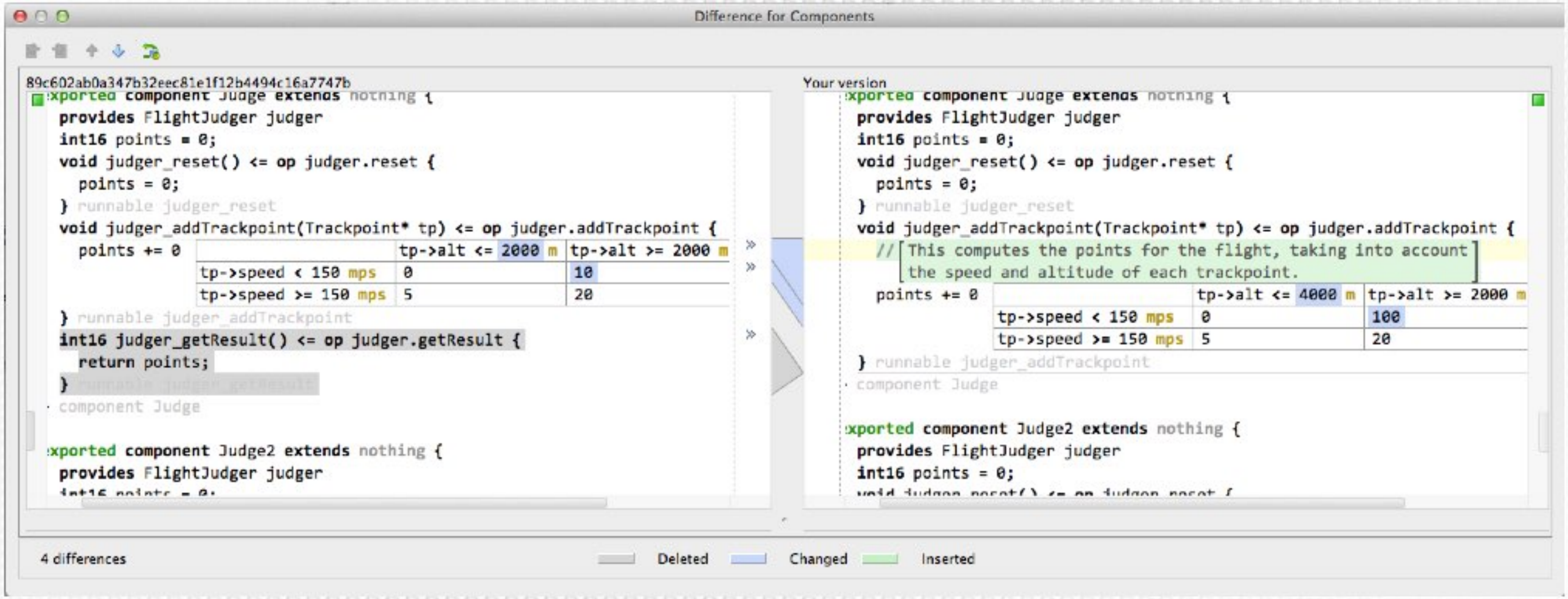
Switch between costumes

```
when [flag] clicked
forever
    play sound [human beatbox1] until done
```

Play music

Scratch, http://scratch.mit.edu

2014



89c602ab0a347b32eec81e1f12b4494c16a7747b

```
:xported component Judge extends nothing {
  provides FlightJudger judger
  int16 points = 0;
  void judger_reset() <= op judger.reset {
    points = 0;
  } runnable judger_reset
  void judger_addTrackpoint(Trackpoint* tp) <= op judger.addTrackpoint {
    points += 0                          tp->alt <= 2000 m  tp->alt >= 2000 m
                  tp->speed < 150 mps    0                  10
                  tp->speed >= 150 mps   5                  20
  } runnable judger_addTrackpoint
  int16 judger_getResult() <= op judger.getResult {
    return points;
  } runnable judger_getResult
· component Judge

:xported component Judge2 extends nothing {
  provides FlightJudger judger
  int16 points = 0;
```

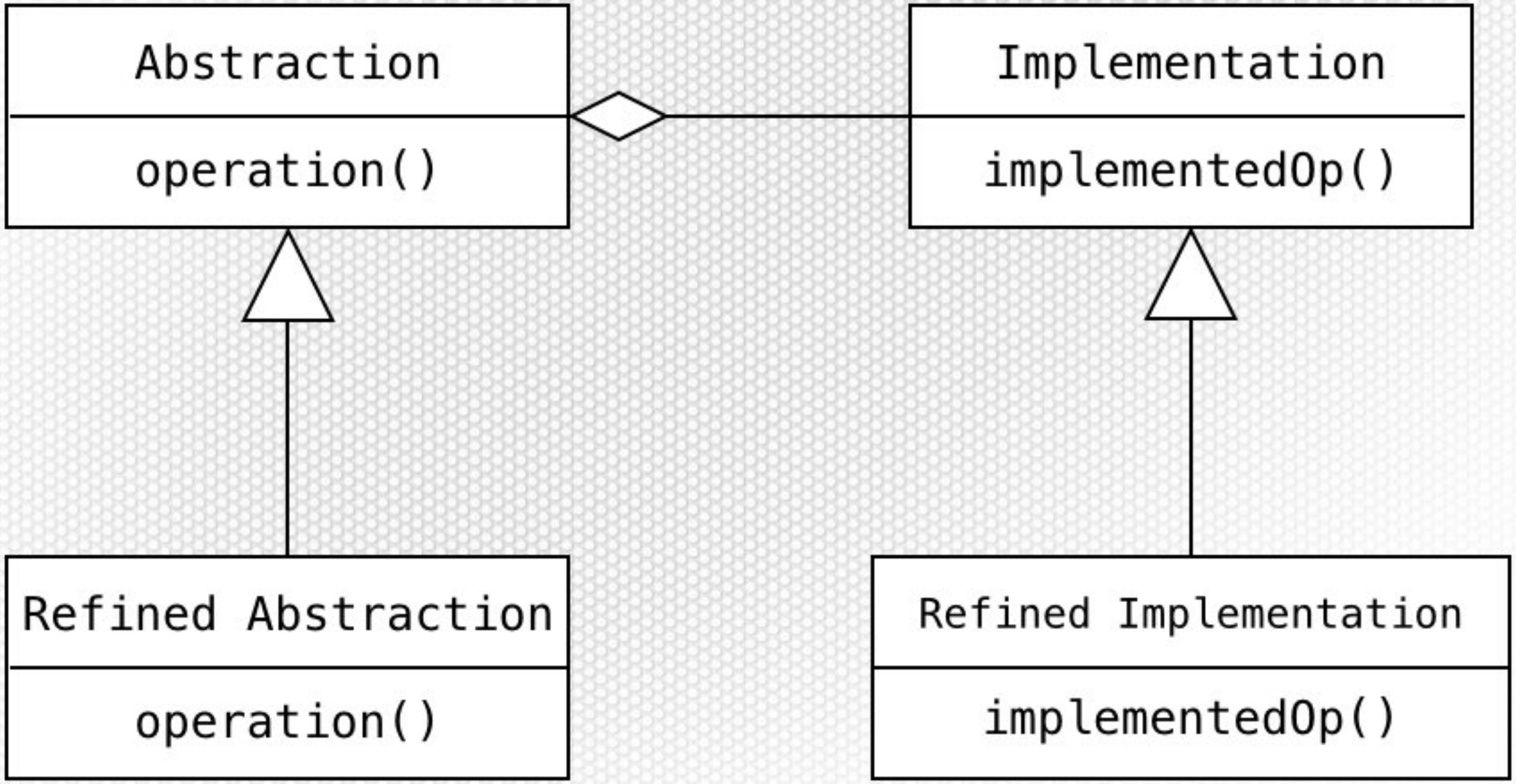
Your version

```
:xported component Judge extends nothing {
  provides FlightJudger judger
  int16 points = 0;
  void judger_reset() <= op judger.reset {
    points = 0;
  } runnable judger_reset
  void judger_addTrackpoint(Trackpoint* tp) <= op judger.addTrackpoint {
    // This computes the points for the flight, taking into account
       the speed and altitude of each trackpoint.
    points += 0                          tp->alt <= 4000 m  tp->alt >= 2000 m
                  tp->speed < 150 mps    0                  100
                  tp->speed >= 150 mps   5                  20
  } runnable judger_addTrackpoint
· component Judge

:xported component Judge2 extends nothing {
  provides FlightJudger judger
  int16 points = 0;
  void judger_reset() <= op judger.reset {
```

Markus Völter, Generic Tools, Specific Languages, PhD Thesis, 2014. p. 110.

1994

UML



Abstraction

operation()

Implementation

implementedOp()

Refined Abstraction

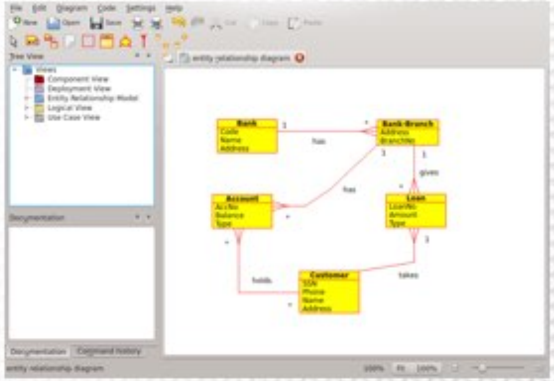
operation()

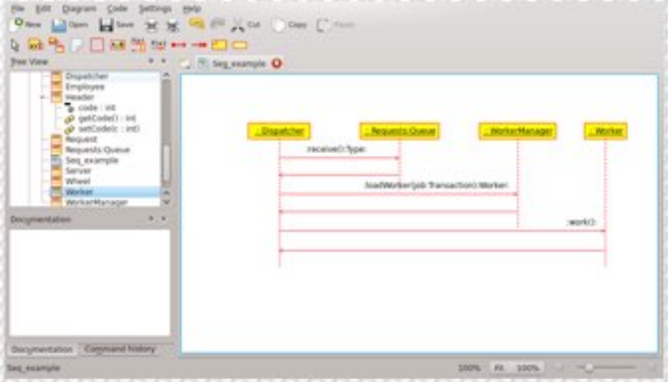
Refined Implementation

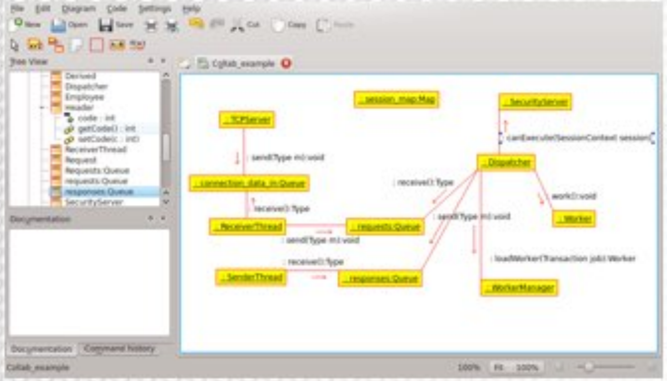
implementedOp()

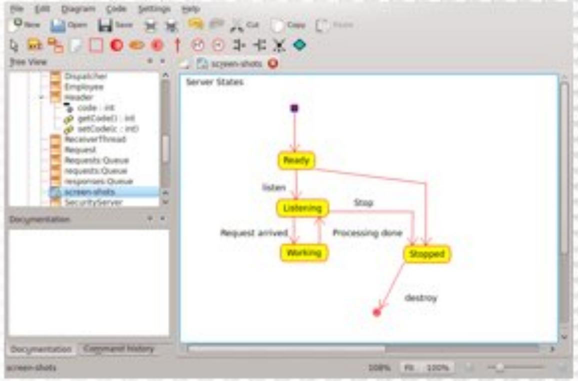
1994

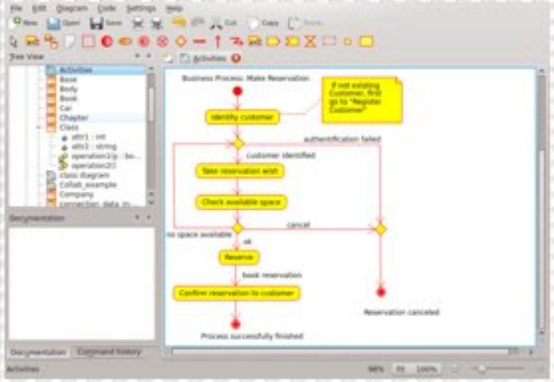
UML

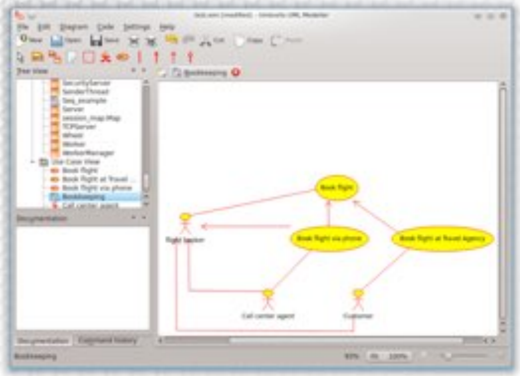






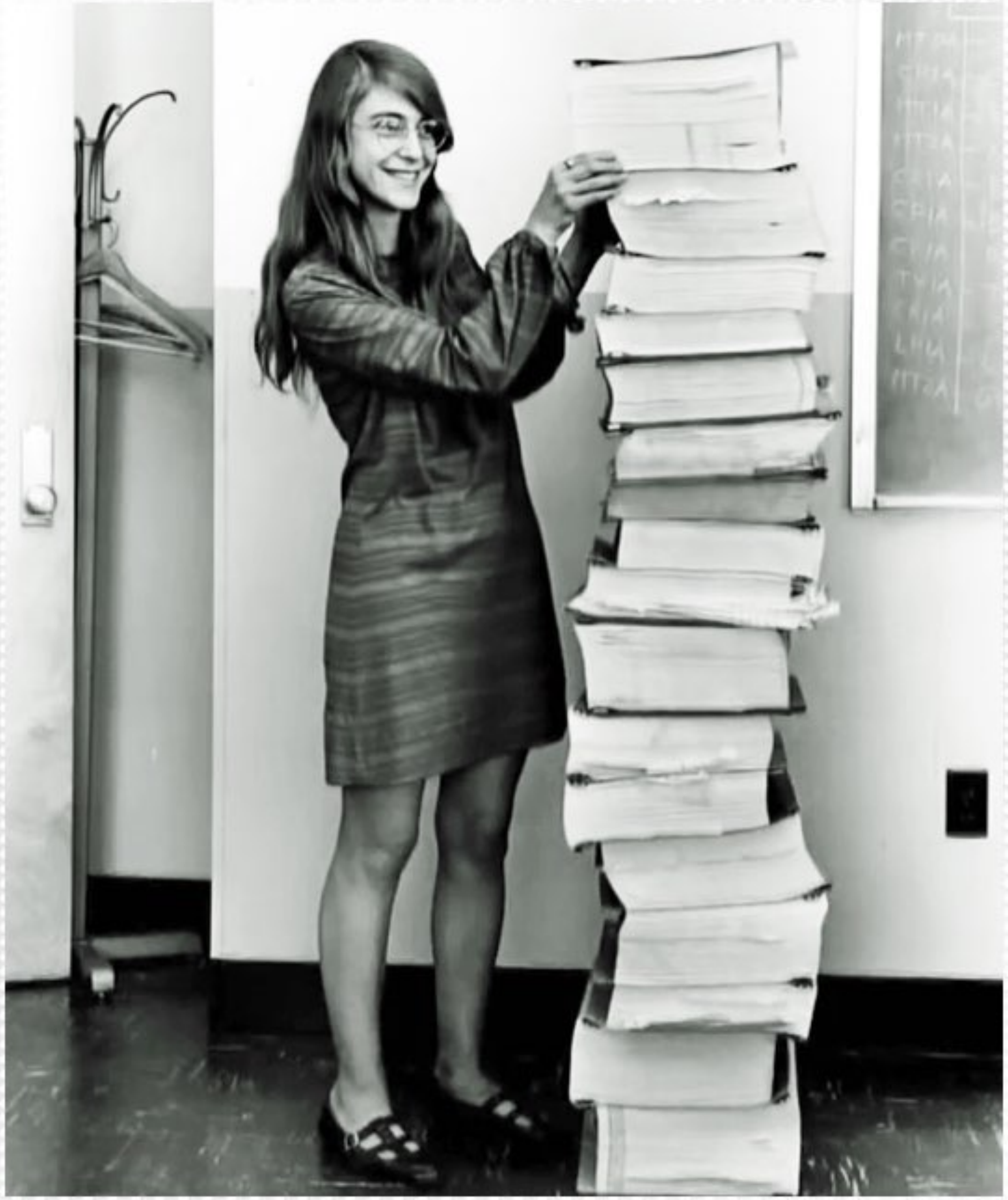






https://docs.kde.org/trunk4/en/kdesdk/umbrello/uml-elements.html

1971

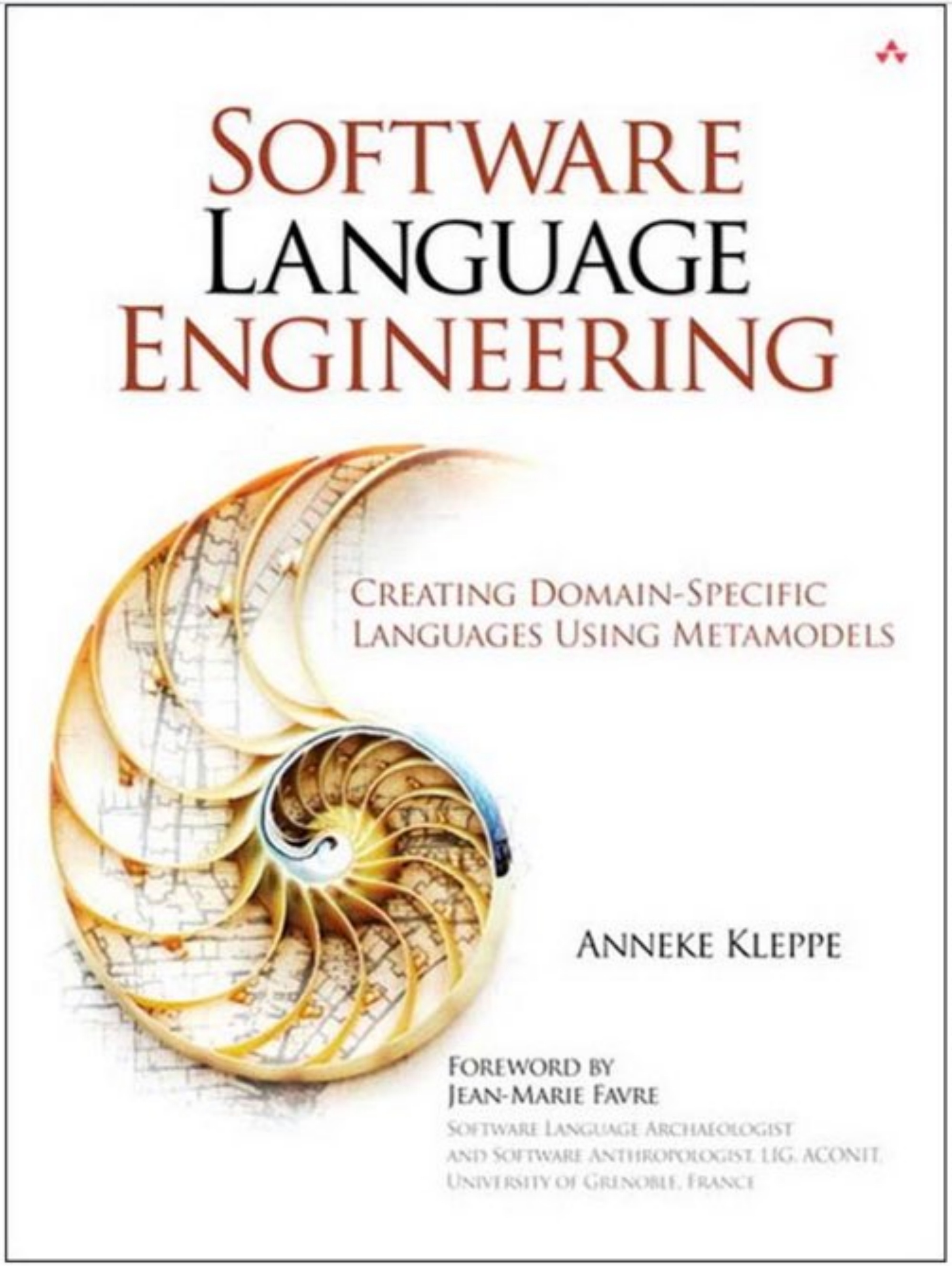


Margaret H. Hamilton (b. 1936)

http://threefingeredfox.net/?p=143, PD

photo credit: Eelco Visser, https://www.flickr.com/photos/eelcovisser/2006308962/

2008

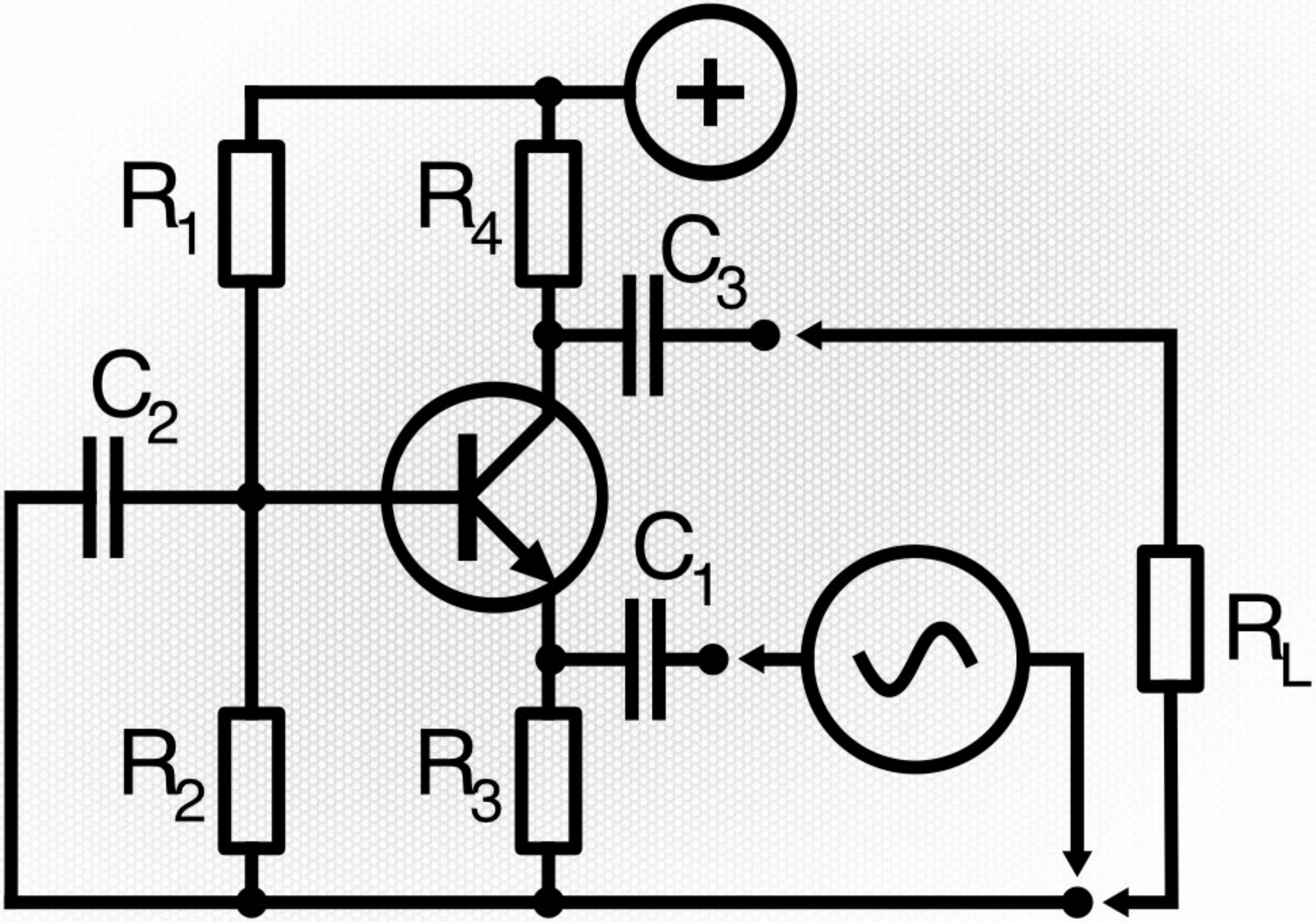


SOFTWARE LANGUAGE ENGINEERING

CREATING DOMAIN-SPECIFIC LANGUAGES USING METAMODELS

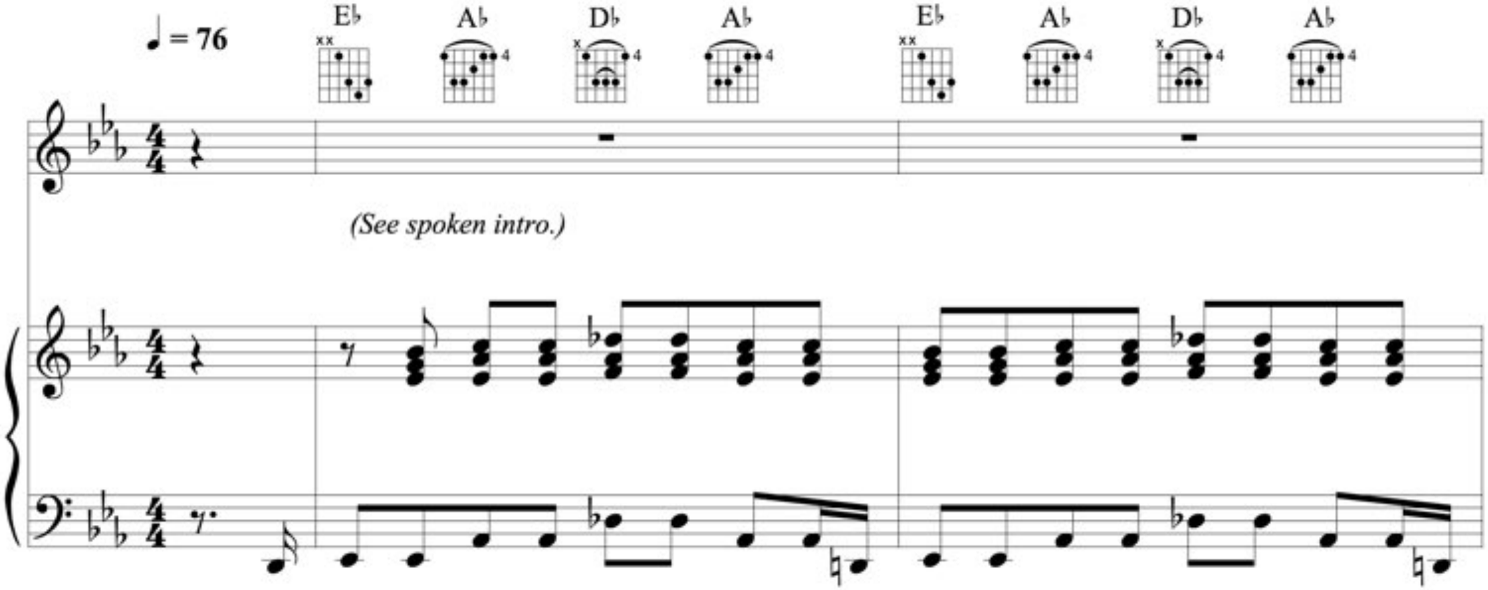
ANNEKE KLEPPE

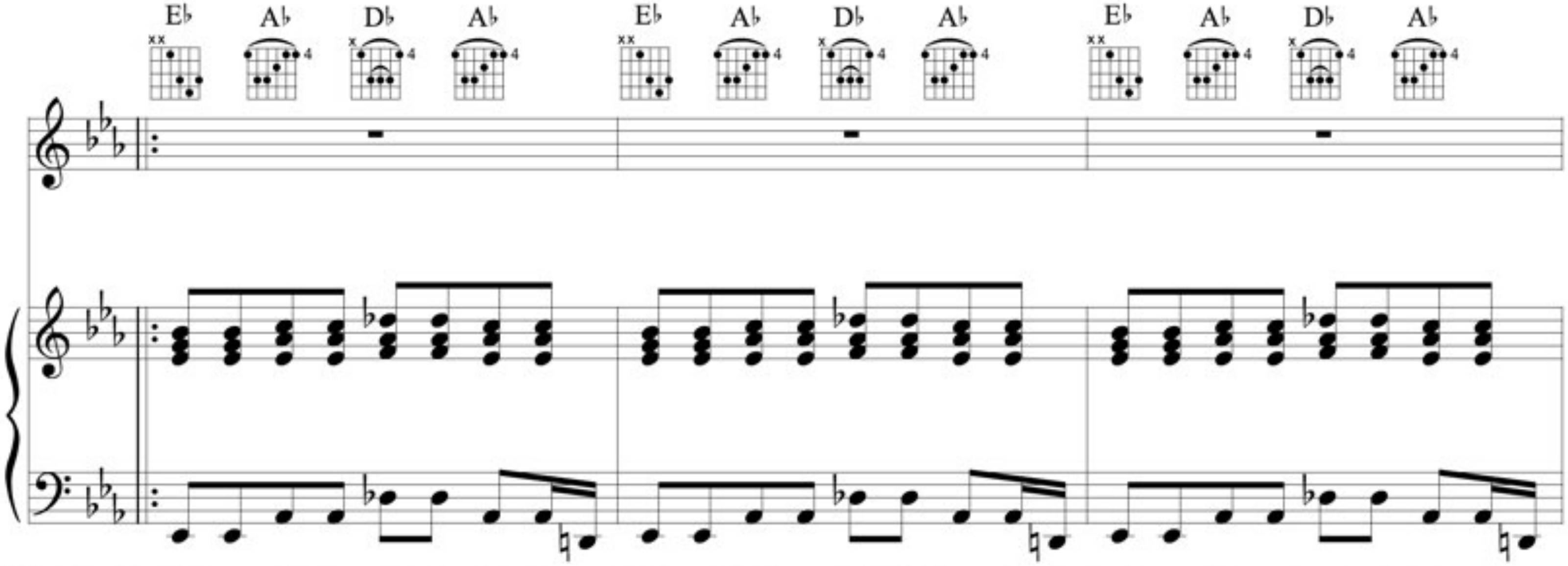
FOREWORD BY
JEAN-MARIE FAVRE
SOFTWARE LANGUAGE ARCHAEOLOGIST
AND SOFTWARE ANTHROPOLOGIST, LIG, ACONIT,
UNIVERSITY OF GRENOBLE, FRANCE


+
R1
R4
C3
C2
C1
RL
R2
R3

Everybody Needs Somebody To Love

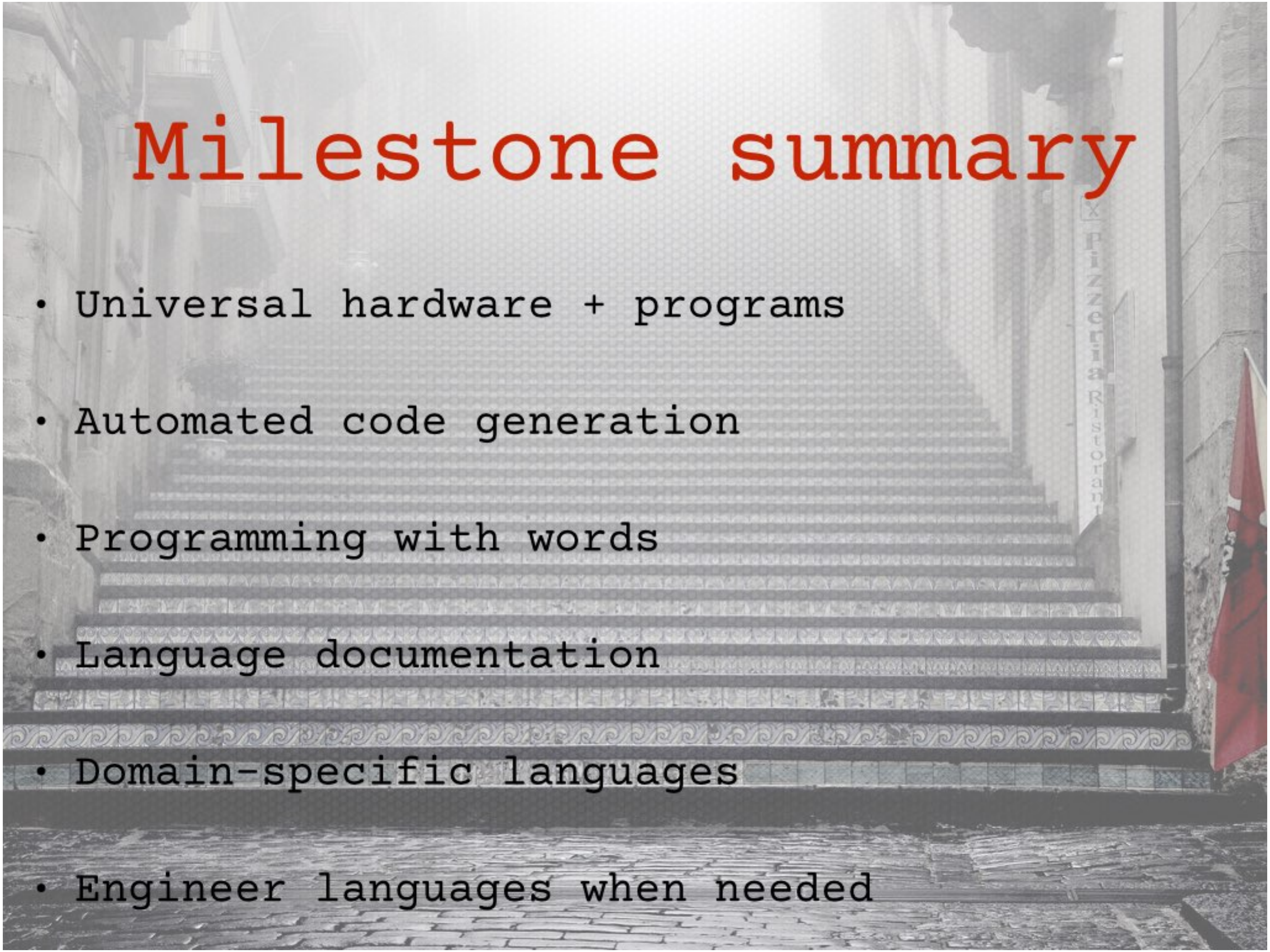
Words & Music by Bert Russell, Jerry Wexler & Solomon Burke





http://www.musicnotes.com/sheetmusic/mtd.asp?ppn=MN0064902

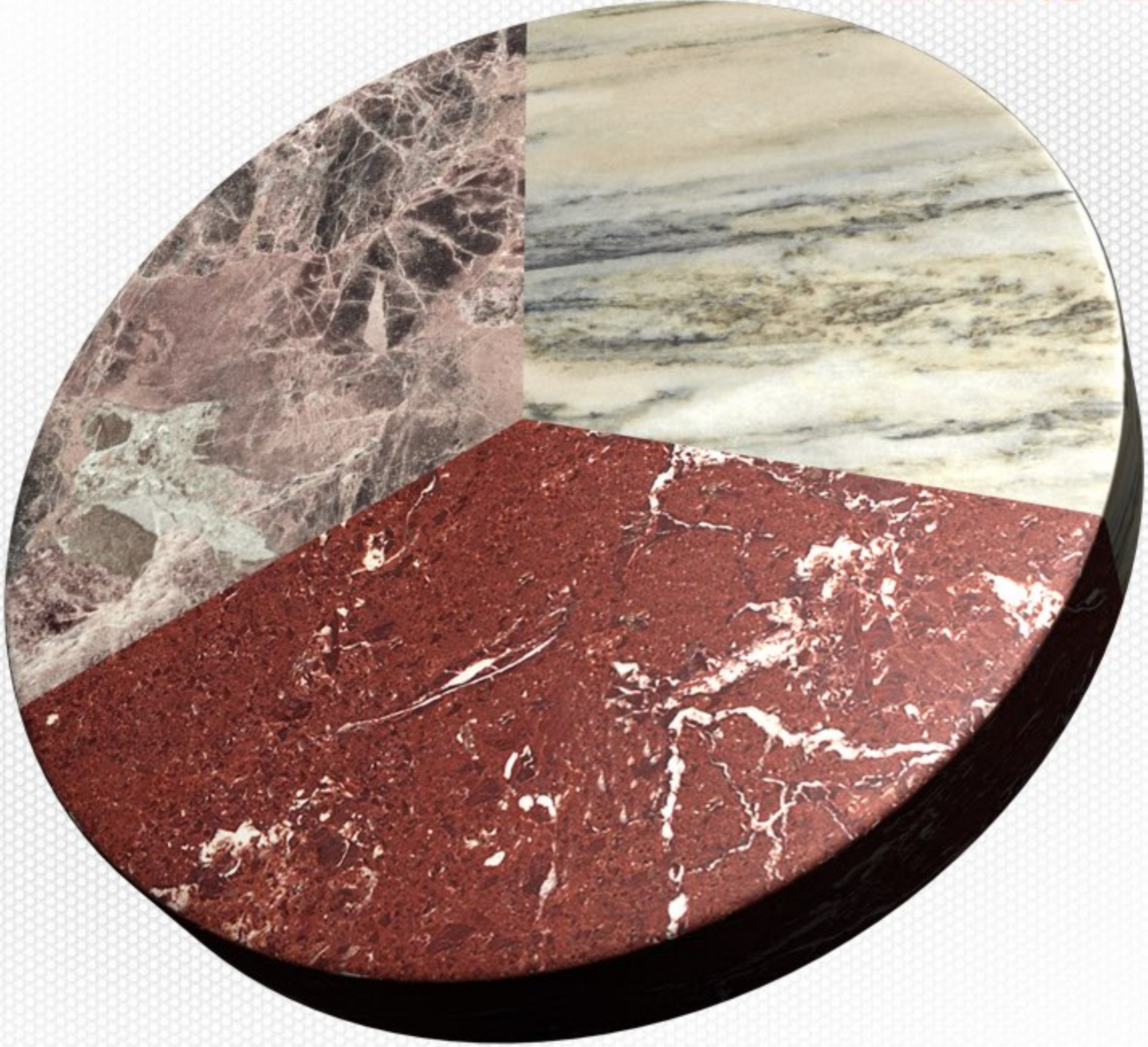
Edison
StorageTek



Milestone summary

- Universal hardware + programs
- Automated code generation
- Programming with words
- Language documentation
- Domain-specific languages
- Engineer languages when needed

DESIGN



Part II

Domain

Ontology

Schema

Grammar

what kind of problems will you tell the machine to solve?

Domain

- What will the language be used for?

- Algorithms?
- Markup?
- Data?
- Constraints
- Finance?
- Visual?
- Drawing?
- Parallel?
- Spreadsheet
- Formulae?
- Queries?
- Music?
- Dance?
- Space?
- Food?

Part II

Domain

topic
theme
problems
concerns

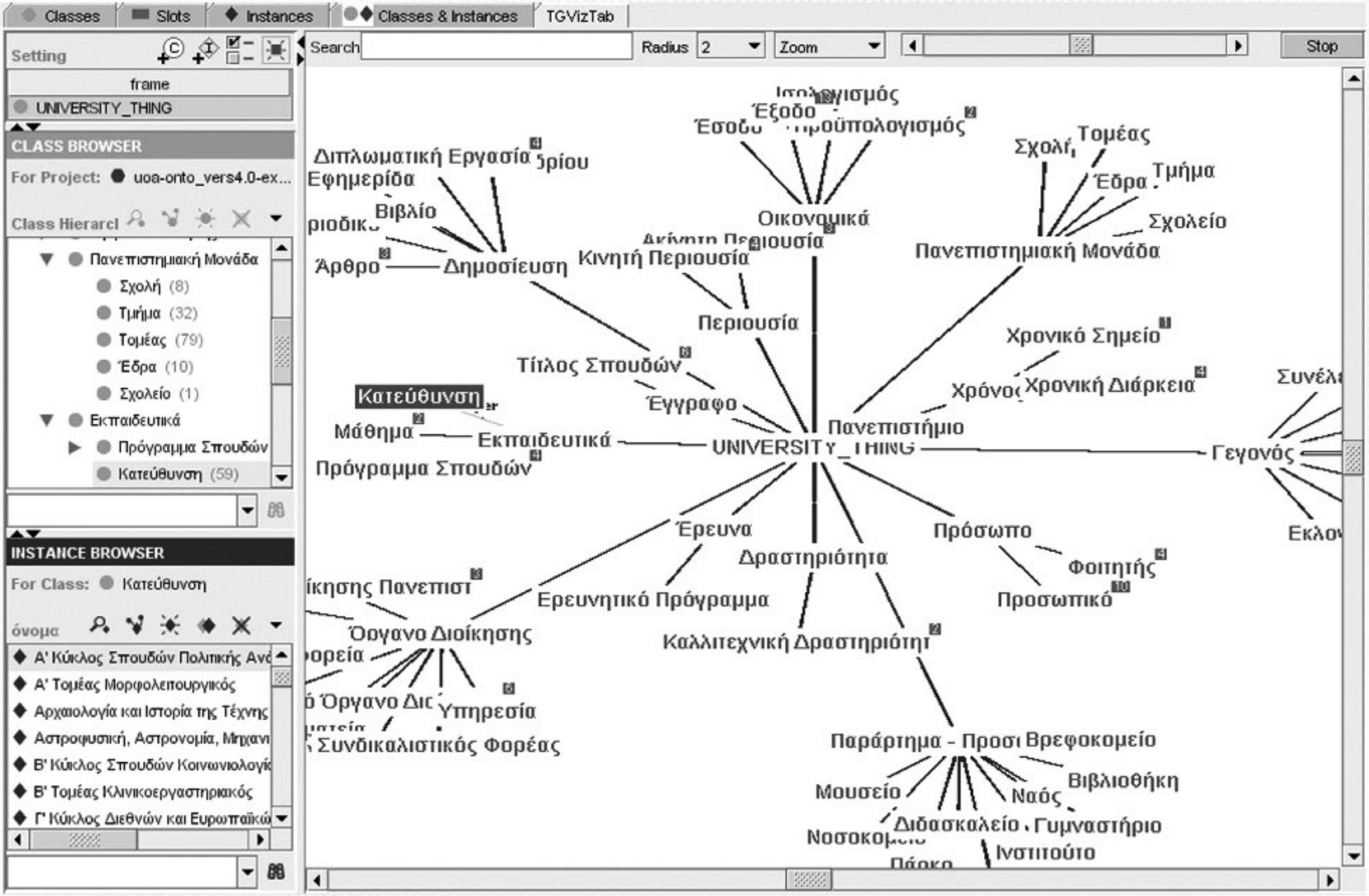
Ontology

Schema

Grammar

Ontology

- Fundamental entities of the domain
- Their properties
- Interrelationships
- (Could be a mindmap or mindmap-like)



UNIVERSITY_THING

For Project: ● uoa-onto_vers4.0-ex...

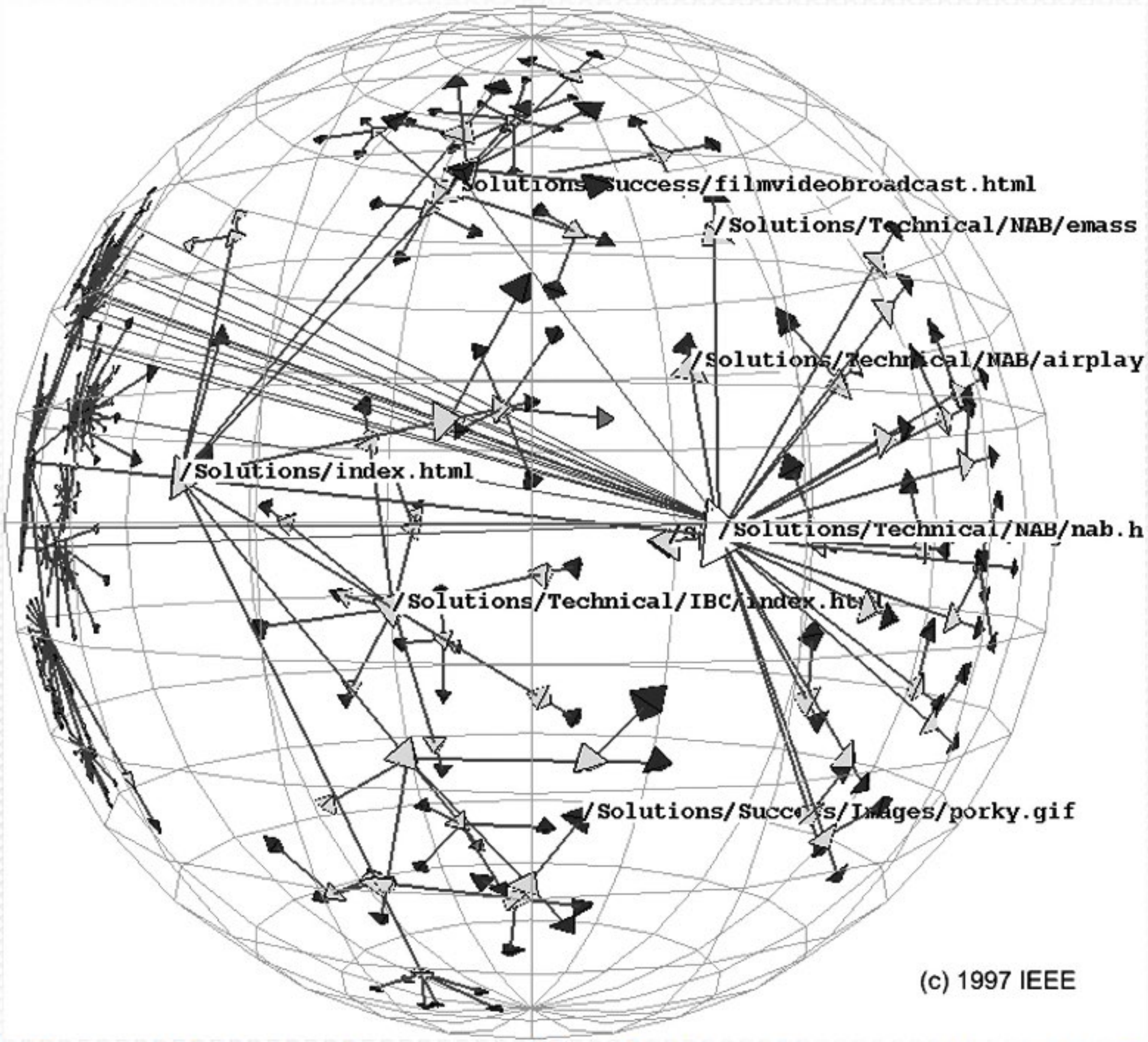
- ▼ ● Πανεπιστημιακή Μονάδα
  - ● Σχολή (8)
  - ● Τμήμα (32)
  - ● Τομέας (79)
  - ● Έδρα (10)
  - ● Σχολείο (1)
- ▼ ● Εκπαιδευτικά
  - ▶ ● Πρόγραμμα Σπουδών
  - ● Κατεύθυνση (59)

For Class: ● Κατεύθυνση

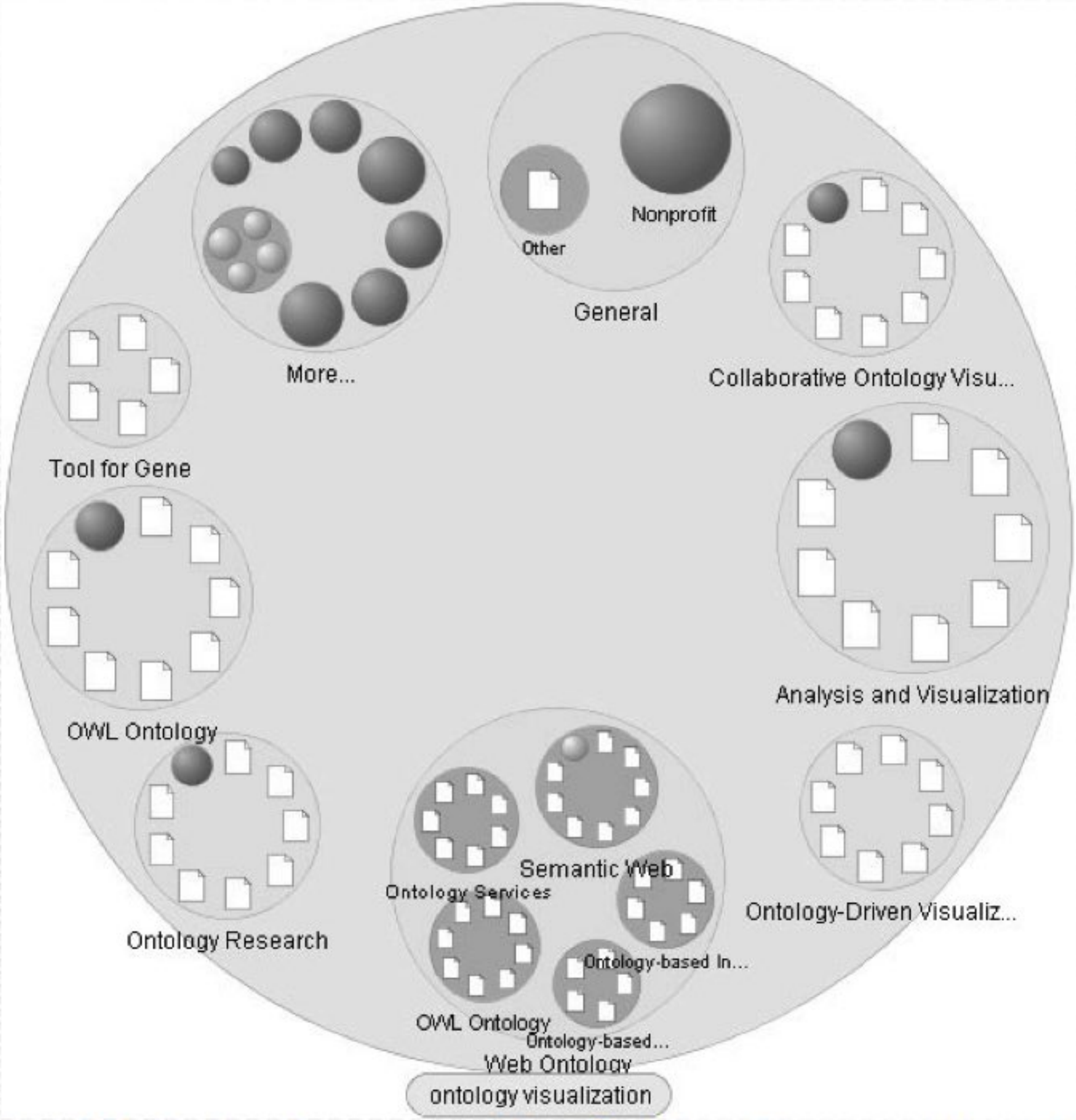
- ◆ Α' Κύκλος Σπουδών Πολιτικής Ανά...
- ◆ Α' Τομέας Μορφολειτουργικός
- ◆ Αρχαιολογία και Ιστορία της Τέχνης
- ◆ Αστροφυσική, Αστρονομία, Μηχανι...
- ◆ Β' Κύκλος Σπουδών Κοινωνιολογί...
- ◆ Β' Τομέας Κλινικοεργαστηριακός
- ◆ Γ' Κύκλος Διεθνών και Ευρωπαϊκώ...

Radius 2

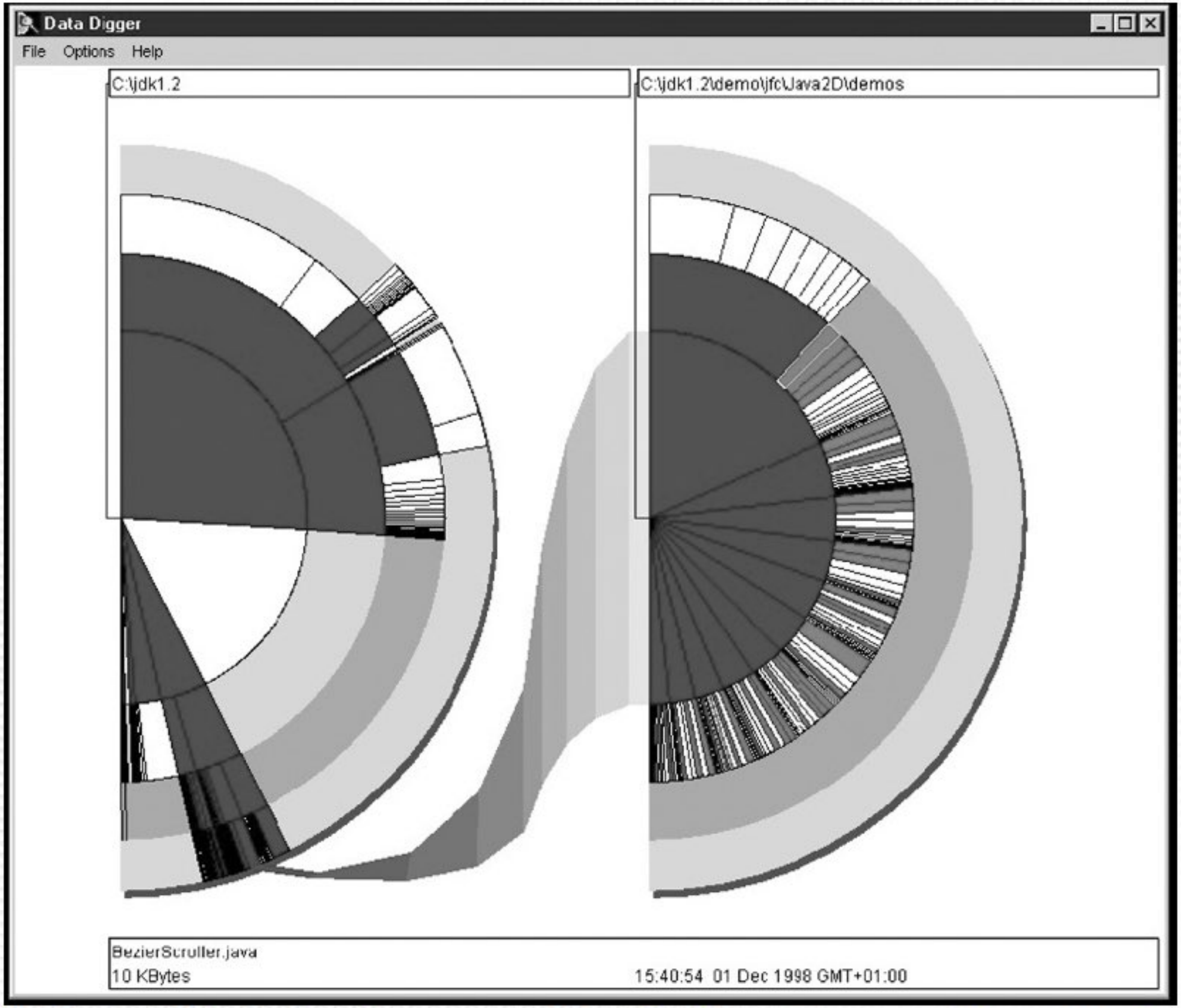
Katifori, Halatsis, Lepouras, Vassilakis, Giannopoulou, Ontology Visualization Methods — A Survey, http://dx.doi.org/10.1145/1287620.1287621


/Solutions/Success/filmvideobroadcast.html
/Solutions/Technical/NAB/emass
/Solutions/Technical/NAB/airplay
/Solutions/index.html
/Solutions/Technical/NAB/nab.h
/Solutions/Technical/IBC/index.html
/Solutions/Success/Images/porky.gif
(c) 1997 IEEE

Katifori, Halatsis, Lepouras, Vassilakis, Giannopoulou, Ontology Visualization Methods – A Survey, http://dx.doi.org/10.1145/1287620.1287621



Katifori, Halatsis, Lepouras, Vassilakis, Giannopoulou, Ontology Visualization Methods – A Survey, http://dx.doi.org/10.1145/1287620.1287621



Katifori, Halatsis, Lepouras, Vassilakis, Giannopoulou, Ontology Visualization Methods – A Survey, http://dx.doi.org/10.1145/1287620.1287621

Part II

Domain

topic
theme
problems
concerns

Ontology

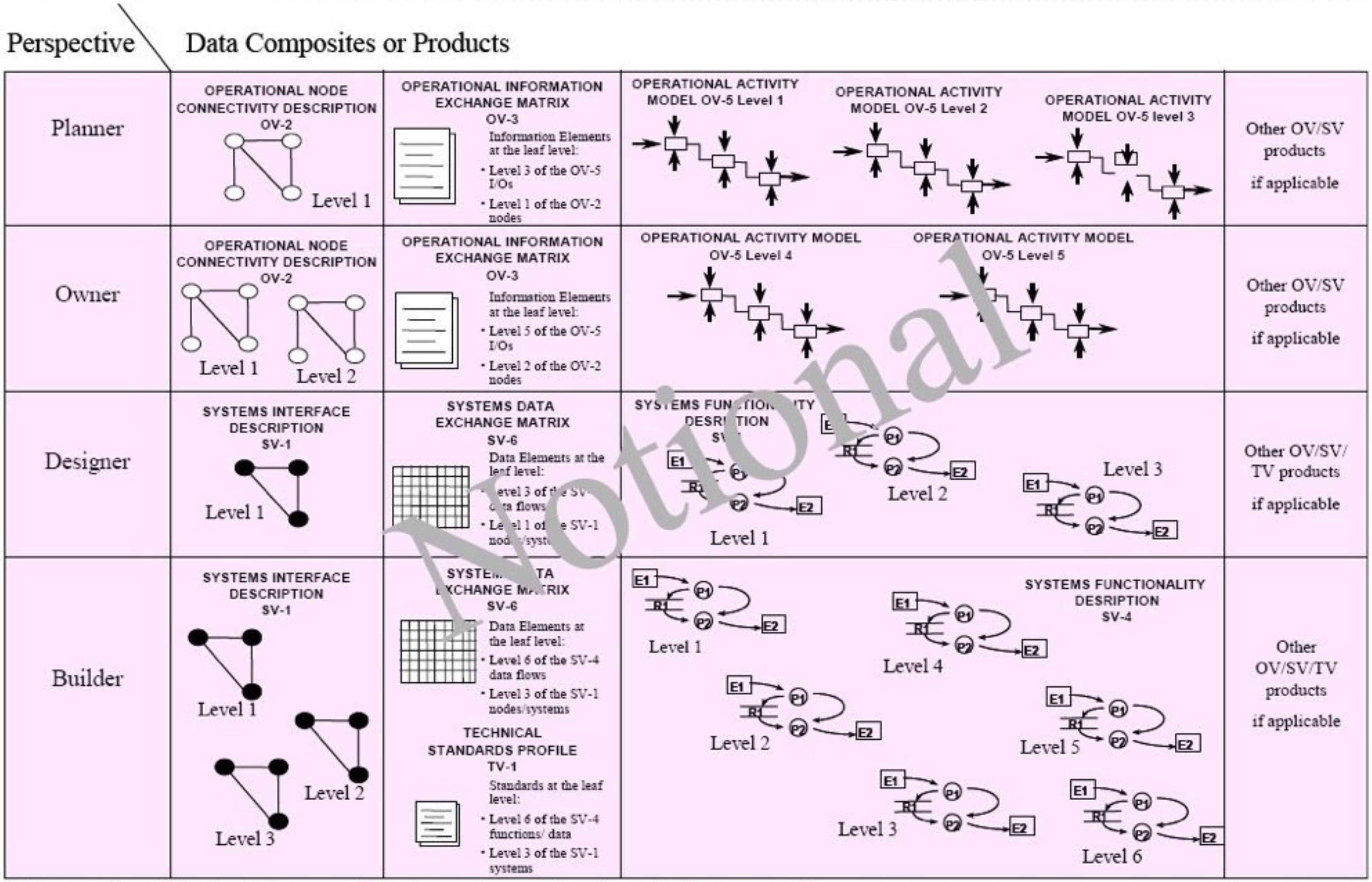
state
things
events
concepts
properties
composition

Schema

Grammar

Schema

- What are “sentences”, conceptually?
- Lists? Sets? Trees? Graphs? Tables?
- Looking inside a sentence, what is there?
- Are there different kinds of sentences?
- (Explicit language modelling)



Perspective	Data Composites or Products			
Planner	OPERATIONAL NODE CONNECTIVITY DESCRIPTION OV-2 Level 1	OPERATIONAL INFORMATION EXCHANGE MATRIX OV-3 Information Elements at the leaf level: • Level 3 of the OV-5 I/Os • Level 1 of the OV-2 nodes	OPERATIONAL ACTIVITY MODEL OV-5 Level 1 OPERATIONAL ACTIVITY MODEL OV-5 Level 2 OPERATIONAL ACTIVITY MODEL OV-5 level 3	Other OV/SV products if applicable
Owner	OPERATIONAL NODE CONNECTIVITY DESCRIPTION OV-2 Level 1 Level 2	OPERATIONAL INFORMATION EXCHANGE MATRIX OV-3 Information Elements at the leaf level: • Level 5 of the OV-5 I/Os • Level 2 of the OV-2 nodes	OPERATIONAL ACTIVITY MODEL OV-5 Level 4 OPERATIONAL ACTIVITY MODEL OV-5 Level 5	Other OV/SV products if applicable
Designer	SYSTEMS INTERFACE DESCRIPTION SV-1 Level 1	SYSTEMS DATA EXCHANGE MATRIX SV-6 Data Elements at the leaf level: • Level 3 of the SV-4 data flows • Level 1 of the SV-1 nodes/systems	SYSTEMS FUNCTIONALITY DESRIPTION SV-4 Level 1 Level 2 Level 3	Other OV/SV/ TV products if applicable
Builder	SYSTEMS INTERFACE DESCRIPTION SV-1 Level 1 Level 2 Level 3	SYSTEMS DATA EXCHANGE MATRIX SV-6 Data Elements at the leaf level: • Level 6 of the SV-4 data flows • Level 3 of the SV-1 nodes/systems TECHNICAL STANDARDS PROFILE TV-1 Standards at the leaf level: • Level 6 of the SV-4 functions/ data • Level 3 of the SV-1 systems	SYSTEMS FUNCTIONALITY DESRIPTION SV-4 Level 1 Level 4 Level 2 Level 5 Level 3 Level 6	Other OV/SV/TV products if applicable

No more than 6 levels of decomposition for each type of product within a perspective

All products within a perspective remain cohesive as to level of detail provided in each

DoDAF Perspectives and Decomposition Levels
DoDAF = Department of Defence Architecture Framework

Schema

- Algebraic data type:

```
data Bool
        = tt()
        | ff()
        | conj(Bool L, Bool R)
        | disj(Bool L, Bool R)
        ;
```

http://tutor.rascal-mpl.org/Rascal/Declarations/AlgebraicDataType/AlgebraicDataType.html

Part II

Domain

topic
theme
problems
concerns

Ontology

state
things
events
concepts
properties
composition

Schema

data types
containment
manipulation
initialisation
abstract structure

Grammar

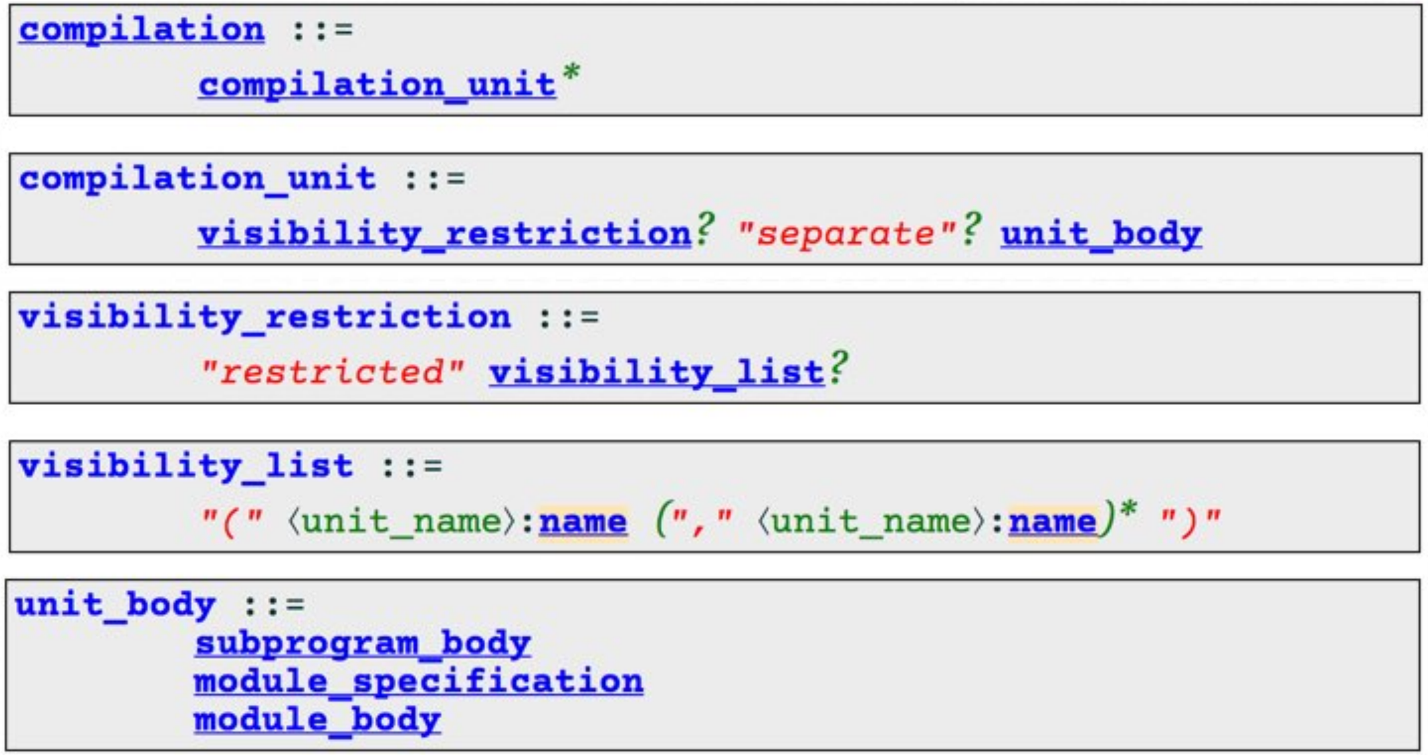
Grammar

- How do you write sentences down?
- What alphabet do you use?
- How symbols are constructed in it?
- Text? Table? Diagrams? Unicode? Colours?

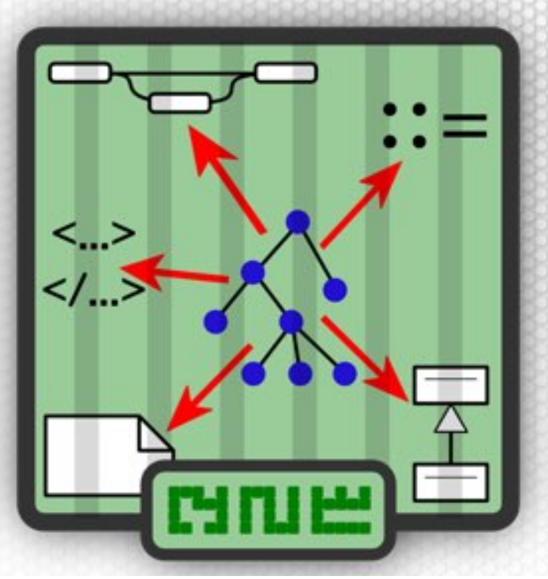
photo credits: http://www.cs.cornell.edu/gries/banquets/iticse2002/iticse2002.html

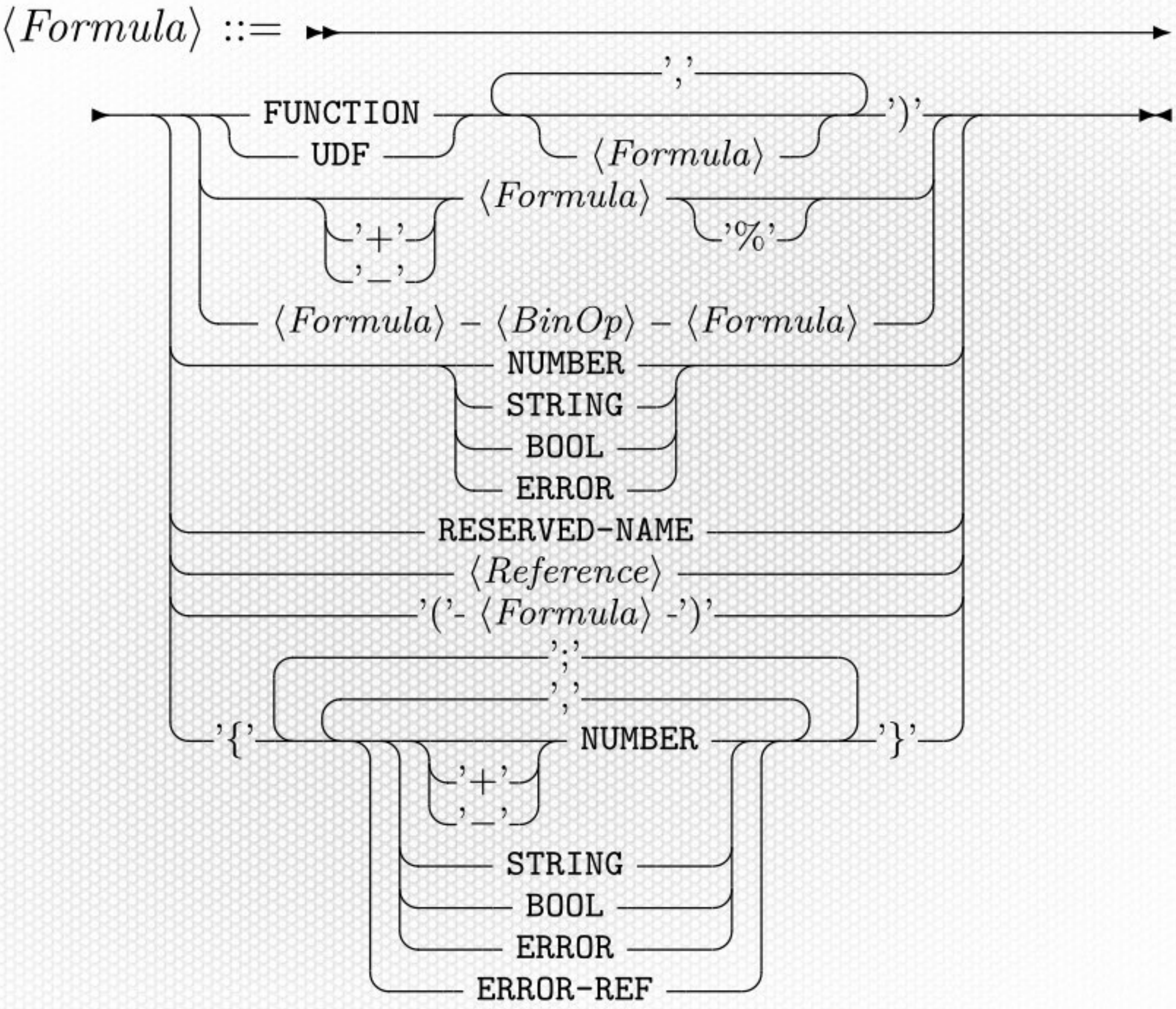
1958

BNF



```
compilation ::=
        compilation_unit*

compilation_unit ::=
        visibility_restriction? "separate"? unit_body

visibility_restriction ::=
        "restricted" visibility_list?

visibility_list ::=
        "(" ⟨unit_name⟩:name ("," ⟨unit_name⟩:name)* ")"

unit_body ::=
        subprogram_body
        module_specification
        module_body
```





Aivaloglou, Hoepelman, Hermans, A Grammar for Spreadsheet Formulas Evaluated on Two Large Datasets, SCAM 2015

wave to activate

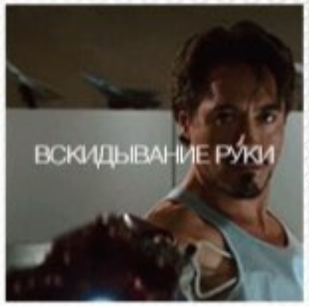
push to move

turn to rotate

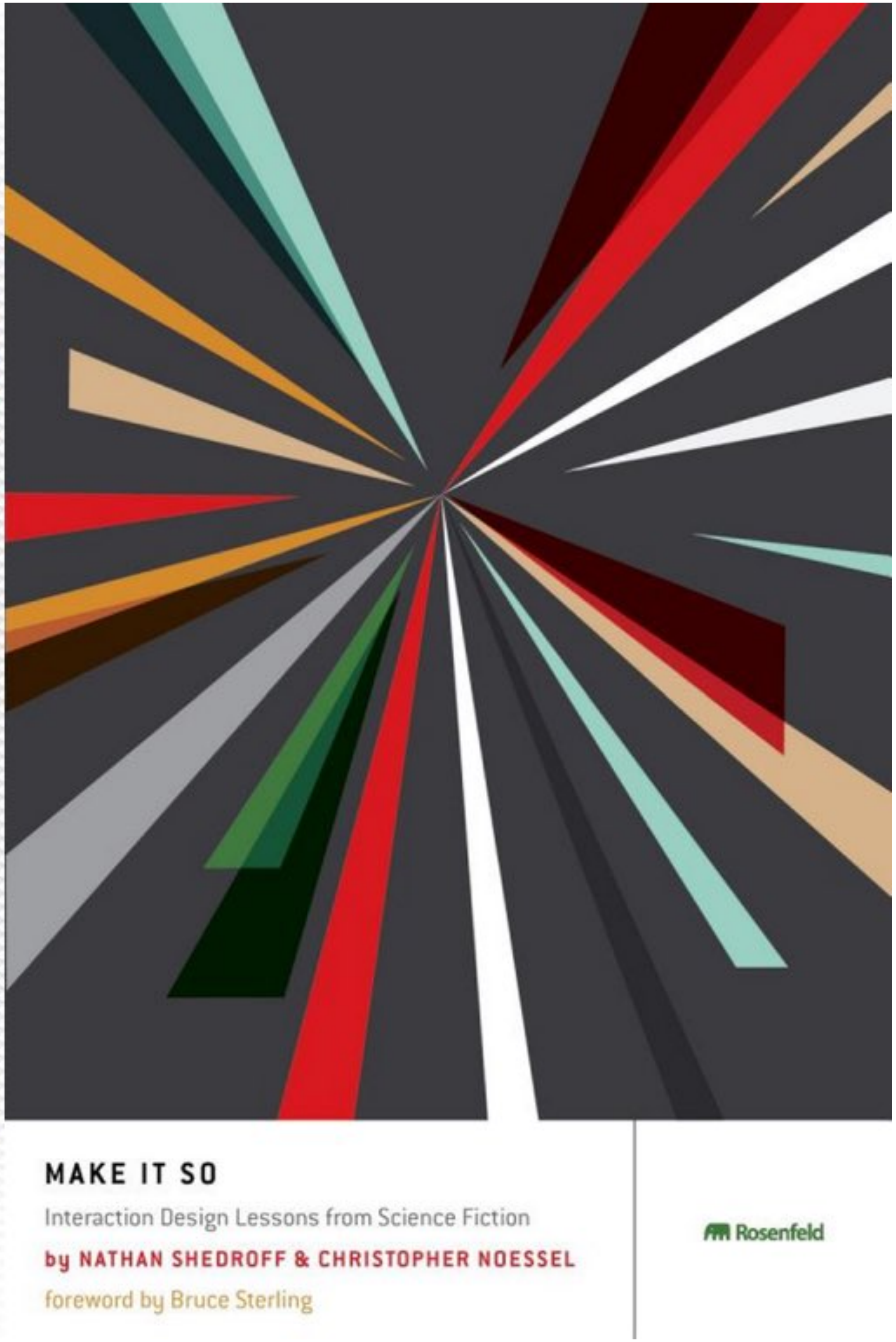
swipe to dismiss

touch to select

spread to scale



raise hand to shoot



MAKE IT SO

Interaction Design Lessons from Science Fiction

by NATHAN SHEDROFF & CHRISTOPHER NOESSEL

foreword by Bruce Sterling

Rosenfeld

Sources & recs

- Chris Noessel, What Sci-Fi Tells Interaction Designers About Gestural Interfaces, http://www.smashingmagazine.com/2013/03/sci-fi-interaction-designers-gestural-interfaces/

- Nathan Shedroff, Christopher Noessel, Make It So, 2012.

- Tema Ra, Aesthetics of Futuristic Interfaces, http://rhzm.ru/posts/114

- Vladimir Zavertailov, Computer Interfaces in Cinema - Evolution of Imagination, http://habrahabr.ru/post/250829/

Don’t Let Your Dreams Be Dreams!

Domain

topic
theme
problems
concerns

Schema

data types
containment
manipulation
initialisation
abstract structure

Ontology

state
things
events
concepts
properties
composition

Grammar

symbols
alphabet
sentences

Domain

topic
theme
problems
concerns

Ontology

state
things
events
concepts
properties
composition

Schema

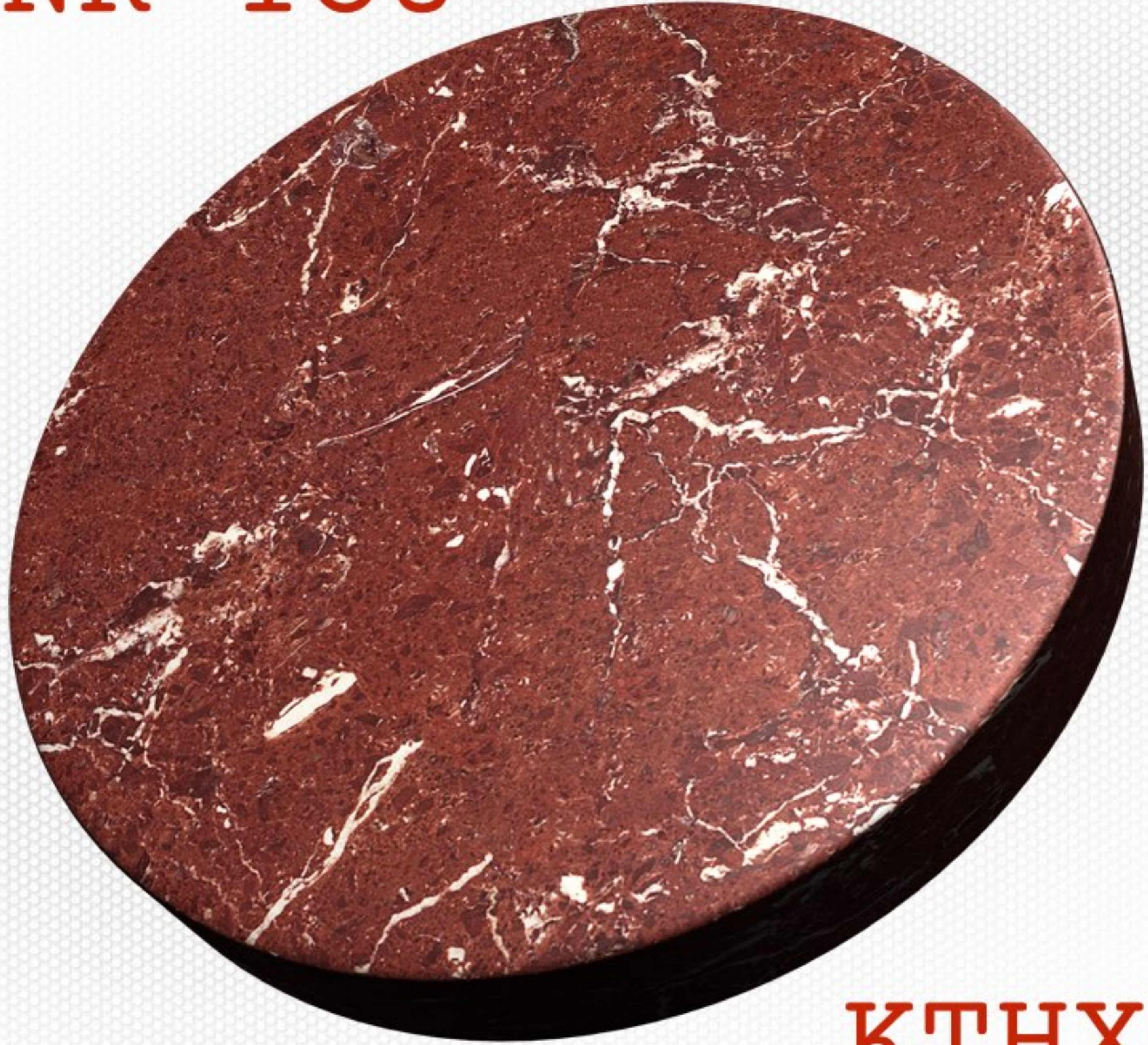
data types
containment
manipulation
initialisation
abstract structure

Grammar

symbols
alphabet
sentences

PRESENT

THANK YOU



KTHXBYE

Follow @grammarware