

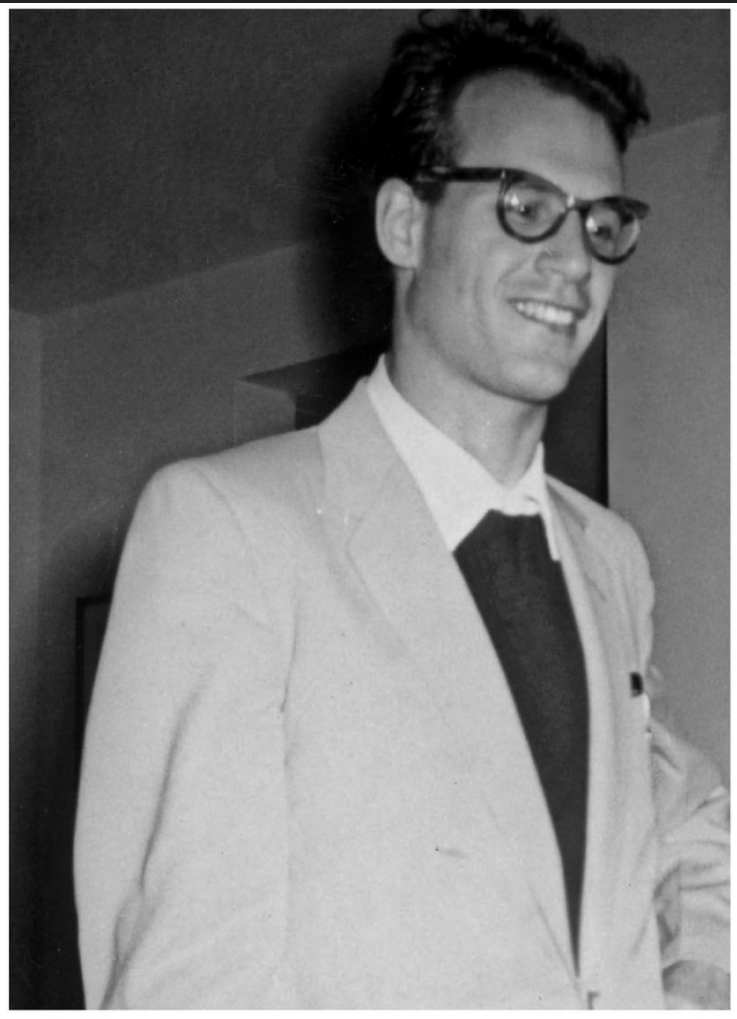
The Rising Sea

Matt Drury, March 2023

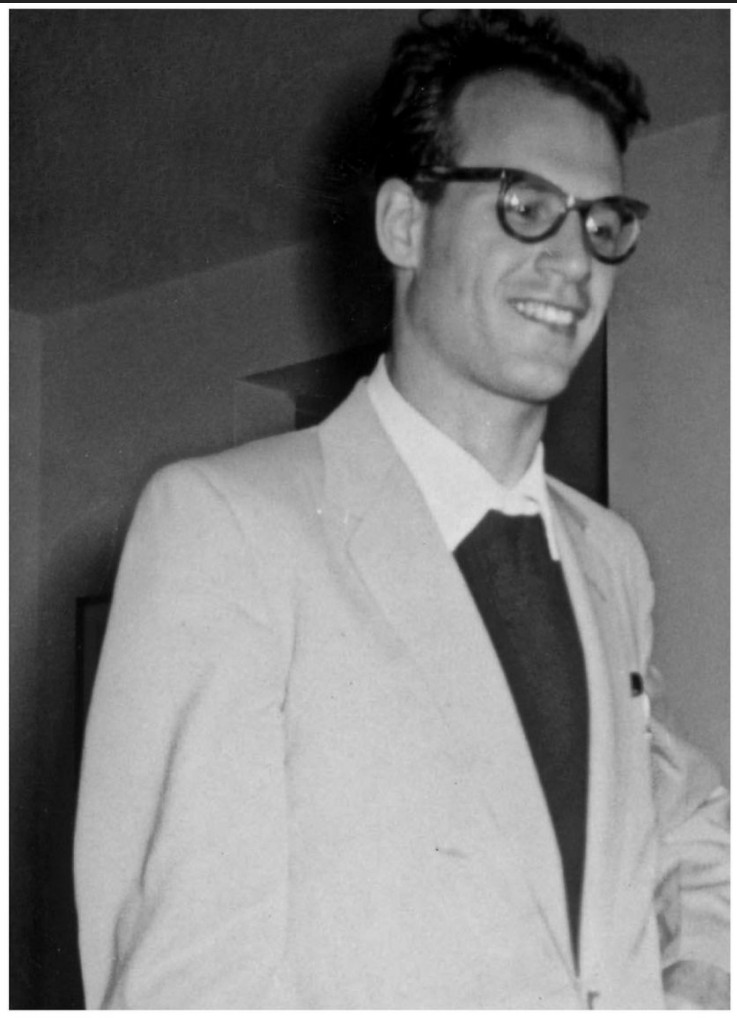
<https://github.com/madrury/the-rising-sea>

The Rising Sea

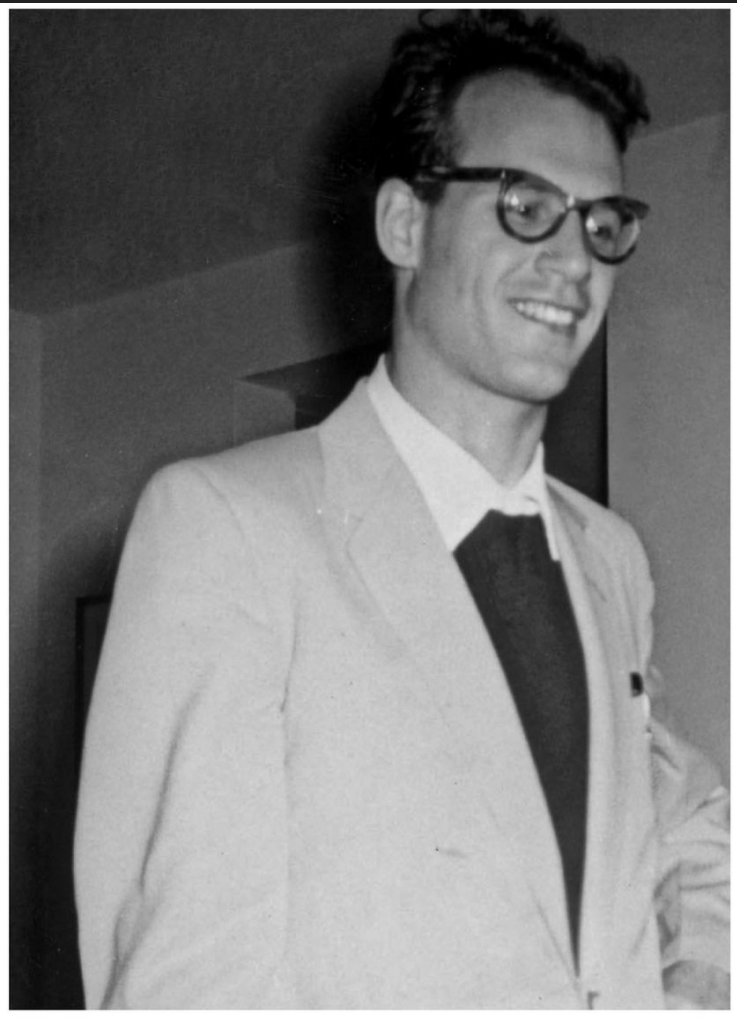
A talk about a great problem solver and his approach.



???



Alexander Grothendieck
1928 - 2014



Alexander Grothendieck
1928 - 2014
Mathematician

Analogies in Other Fields:

- Mathematics : Grothendieck
- Physics : Paul Dirac
- Computer Science : Alonzo Church
- Programming Languages : John McCarthy
- Literature: Cormac McCarthy
- Music: Brian Eno

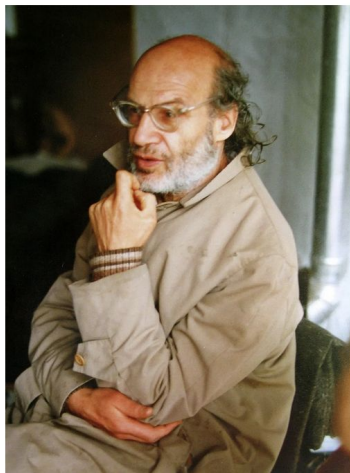
Alexander Grothendieck, Math Enigma, Dies at 86

By [Bruce Weber](#) and Julie Rehmeyer

Nov. 14, 2014



Share full article



Alexander Grothendieck in 1988.
Erika Ifang

nature

[Explore content](#) ▾

[About the journal](#) ▾

[Publish with us](#) ▾

[nature](#) > [obituaries](#) > article

Obituary | [Published: 14 January 2015](#)

Alexander Grothendieck (1928–2014)

[David Mumford](#) ✉ & [John Tate](#) ✉

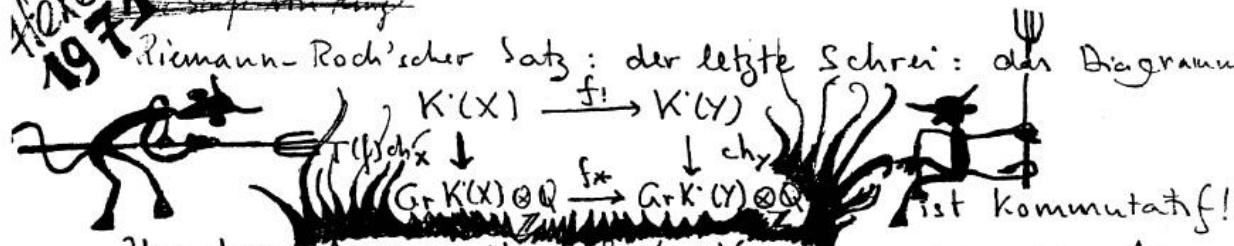
“The sad thing is that this was rejected as much too technical for their readership. Their editor wrote me that 'higher degree polynomials', 'infinitesimal vectors' and 'complex space' (even complex numbers) were things at least half their readership had never come across.”

David Mumford

Alexander
1974

~~Stefan Lange~~

Riemann-Roch'scher Satz: der letzte Schrei: das Diagramm



ist kommutativ!

Um dieser Aussage über $f: X \rightarrow Y$ einen approximativen Sinn zu geben, musste ich nahezu zwei Stunden lang die Geduld der Zuhörer missbrauchen. Schwartz auf Weiss (in Springer's Lecture Notes) nimmt's wohl an die 400,500 Seiten. Ein packendes Beispiel dafür, wie unser Wissens- und Entdeckungsdrang sich immer mehr in einen lebensentrickten logischen Delirium verlebt, während das Leben selbst auf tausendfache Art zum Teufel geht - und ~~mit~~ ^{mit} endgültiger Vernichtung bedroht ist. Höchste Zeit, unsern Kurs zu ändern!

(6.12.1974)

Alexander Grothendieck



The Rising Sea



“A different image came to me a few weeks ago. The unknown thing to be known appeared to me as some stretch of earth or hard marl, resisting penetration... the sea advances insensibly in silence, nothing seems to happen, nothing moves, the water is so far off you hardly hear it... yet it finally surrounds the resistant substance...”

Grothendieck

*“I have also learned not to take glory in the difficulty of a proof: difficulty means we have not understood. **The ideal is to be able to paint a landscape in which the proof is obvious.**”*

Pierre Deligne



Nut Cracking





Jean-Paul
Serre

Graduate Texts in Mathematics

Jean-Pierre Serre

**A Course in
Arithmetic**

“All young number theorists should read Serre’s A Course in Arithmetic; it somehow captures an immense amount of the spirit of the subject in a tiny amount of space.”

Jordan Ellenberg

*“Serre created a series of concise elegant
tools which Grothendieck and coworkers
simplified into thousands of pages of category
theory”*

Colin McLarty

Which style makes for better
computer programs?

Advent of Code, 2022

Problem #5



[D]

[N] [C]

[Z] [M] [P]

1

2

3



move 1 from 2 to 1

[D]

[N] [C]

[Z] [M] [P]

1

2

3



move 2 from 1 to 3

		[D]
	[C]	[N]
[Z]	[M]	[P]

1	2	3

[T]		[Q]				[S]		
[R]		[M]				[L]	[V]	[G]
[D]	[V]	[V]				[Q]	[N]	[C]
[H]	[T]	[S]	[C]			[V]	[D]	[Z]
[Q]	[J]	[D]	[M]		[Z]	[C]	[M]	[F]
[N]	[B]	[H]	[N]	[B]	[W]	[N]	[J]	[M]
[P]	[G]	[R]	[Z]	[Z]	[C]	[Z]	[G]	[P]
[B]	[W]	[N]	[P]	[D]	[V]	[G]	[L]	[T]

1	2	3	4	5	6	7	8	9

Serre: “You should write code to directly solve the problem.”

Grothendieck: “You should structure your data so that the problem solves itself.”

Concepts in program space we need to model:

- Instruction
- Program
- Stack of Boxes
- Harbor (multiple stacks) of boxes.

move 2 from 1 to 3

```
@dataclass  
class Instruction:  
    source: int  
    destination: int  
    count: int
```

```
move 1 from 2 to 1  
move 2 from 1 to 3  
move 1 from 3 to 1  
...
```

```
Program = list[Instruction]
```

```
class Harbor:
```

```
    # Some representation of the data.
```

```
    def execute(self, i: Instruction):
```

```
        # Some code that moves around the stacks.
```

```
    def run(self, p: Program):
```

```
        for instruction in p:
```

```
            self.execute(instruction)
```

```
@dataclass
```

```
class Harbor:
```

```
    stacks: List[Stack]
```

```
    def execute(self, i: Instruction):
```

```
        source = self.stacks[i.source]
```

```
        destination = self.stacks[i.destination]
```

```
        payload = source.popsome(i.count)
```

```
        destination.extend(payload)
```

```
@dataclass
```

```
class Stack:
```

```
    boxes: list[Box]
```

```
    def popsome(self, count: int) -> List[Box]:
```

```
        tail = self.boxes[-count:]
```

```
        self.boxes = self.boxes[:-count]
```

```
        return tail
```

```
    def extend(self, some: List[Box]):
```

```
        self.boxes.extend(some)
```

All our choices were made
here...

- Instruction
- Program
- Stack of Boxes
- Harbor (multiple stacks) of boxes.



The Darkness



Harbor?

[T]		[Q]				[S]		
[R]		[M]				[L]	[V]	[G]
[D]	[V]	[V]				[Q]	[N]	[C]
[H]	[T]	[S]	[C]			[V]	[D]	[Z]
[Q]	[J]	[D]	[M]		[Z]	[C]	[M]	[F]
[N]	[B]	[H]	[N]	[B]	[W]	[N]	[J]	[M]
[P]	[G]	[R]	[Z]	[Z]	[C]	[Z]	[G]	[P]
[B]	[W]	[N]	[P]	[D]	[V]	[G]	[L]	[T]

1

2

3

4

5

6

7

8

9

```

def parse_harbor(stackstr: Iterable[str]) -> Harbor:
    bottomup = list(stackitr)[: -1]
    harbor: Harbor = Harbor([[[] for _ in
range(N_STACKS)]])
    for line in bottomup:
        tokens = [
            ''.join(x for _, x in g[1])
            for g in itertools.groupby(
                enumerate(line),
                lambda t: t[0] // N_CHARS_IN_BOX_TOKEN
            )
        ]
    tokens] boxes: List[Box] = [t.strip('[ ] ') for t in
harbor.iter_stack()]
    for box, stack in zip(boxes,
        if box: stack.append(box)
    return stacks

```

