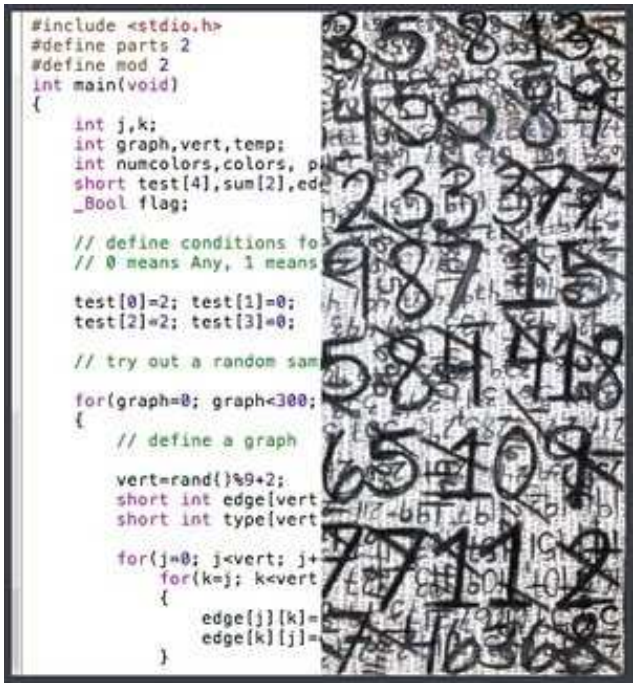
Taking the machine seriously. *A* study of 'mechanized mathematics'



Liesbeth De Mol
Centre for Logic and Philosophy of Science, Belgium
elizabeth.demol@ugent.be

Intro.

Introduction

⇒ **Motivation:** "computers [are] changing the way we do mathematics" (Borwein, 2008)

⇒ **Extent impact??**

- Mathematics proper
- Philosophy of Mathematics

⇒ ... and their interactions

⇒ ***Current research on Mechanized Math (MM) quite unsatisfying (including my own)***

⇒ *Thinking* in progress

Introduction (2)

- PART I: General approach(es)
- PART II: Two case studies
  - Tag systems
  - The chaos game (quick)
- Discussion

General Approach(es)

General Approach(es): "Traditional" approach

⇒ "Traditional" problems from philosophy of mathematics in the light of computer

- Are aspects of **mathematical knowledge** "quasi-heuristic" (Tymoczko, 1979)
- What is **mathematical understanding** in the context of computer-assisted research? (Avigad, 2008)
- ...

⇒ Computer is not *so* special: "*[N]one of the core issues are specific to the use of the computer per se*"

⇒ "**Ask not what the use of computers in mathematics can do for philosophy***; ask what philosophy can do for computers in mathematics [...]* **What we need now is not a philosophy of computers in mathematics***; what we need is simply a better philosophy of mathematics*" (Avigad, 2008)

⇒ **(Problem 1)** Neglect of technical details and history of CS

⇒ **(Problem 2)** Risk of not detecting problems that *are* inherent to the use of computer *per se* and could affect math and phil of math

General Approach(es): Another approach?

⇒ Bottom-up – and see where one gets

- Take computer *seriously* – as a **medium** (Kittler, 1985): "*Media are no tools. Far more than things at our disposal they constitute the interaction of thinking and perception – mainly unconsciously.* (Carlé, 2010). The core issues become visible *through* computer *per se* and are hence shaped by it

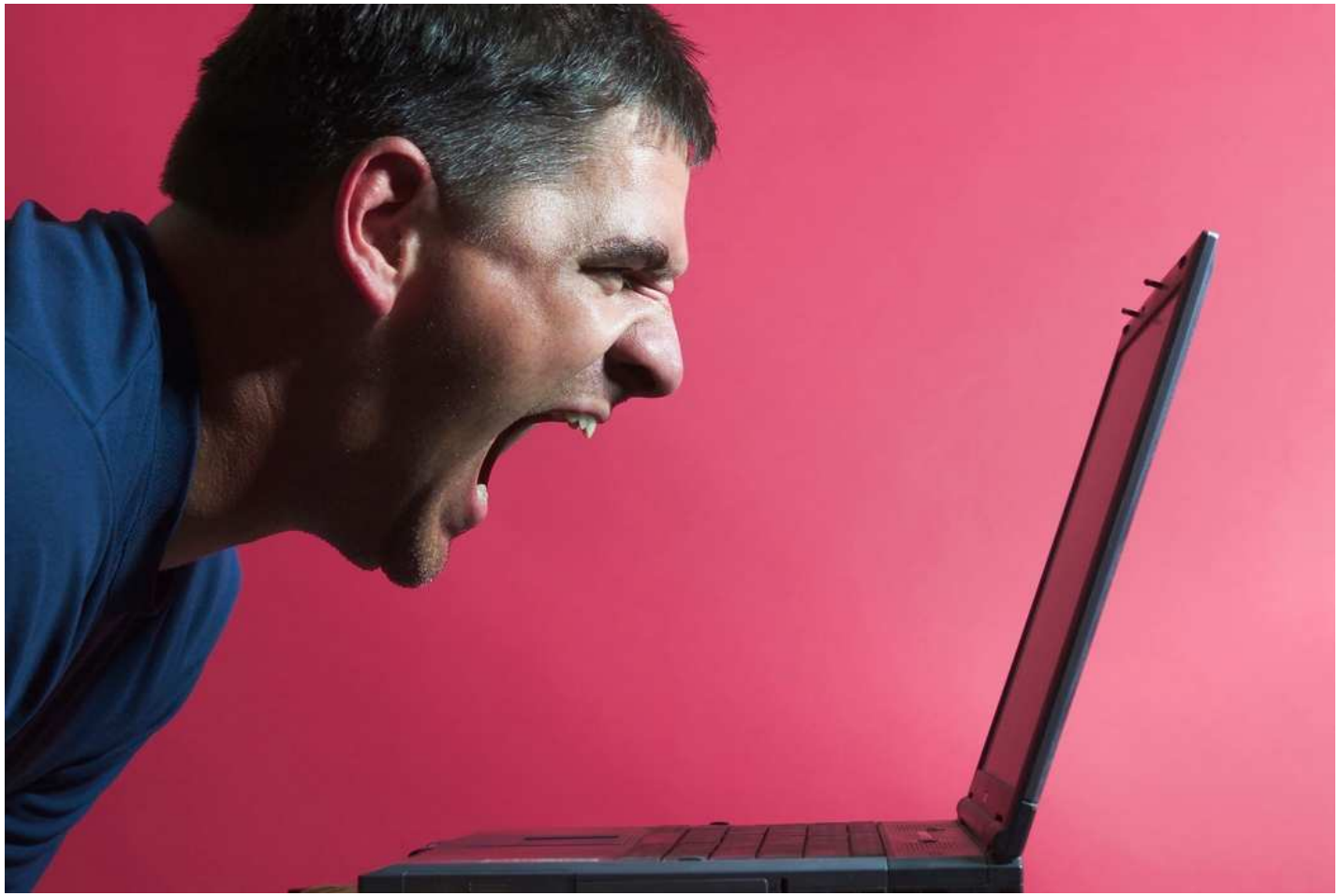
⇒ Philosophy of mathematical **practice(s)** that is *really* guided by that practice → Study "gory" details of (history of) computer-assisted math

⇒ (Phil of Mat)

⇒ We **do** need a philosophy of the computer (in mathematics)

General Approach(es)

Taking the computer seriously....



General Approach(es)

Taking the computer seriously – two classical "myths"

- "*Another argument that continually arises is that* **machines can do nothing we cannot do ourselves**, *though it is admitted that they can do many things faster and more accurately. The statement is true, but also false. It is like the statement that, regarded solely as a form of transportation, modern automobiles and aeroplanes are no different than walking.* **[T]hus the change by six orders of magnitude in computing have produced many fundamentally new effects that are being simply ignored when the statement is made that computers can only do what we could do ourselves if we wished to take the time**" (Hamming, 1965)

- " '**computers can only do what they are told to do**'. *True, but that is like saying that, insofar as mathematics is deductive, once the postulates are given all the rest is trivial. [...]The truth is that in moderately complex situations, such as the postulates of geometry or a complicated program for a computer,* **it is not possible on a practical level to foresee all of the consequences**" (Hamming, 1965)

General Approach(es)

Taking the computer seriously – two background assumptions/inspirations

- **Heideggerian assumption: "Everywhere everything [also man] is ordered to stand by, to be immediately on hand" The danger of technique is that it is hidden away.**
  - (Dijkstra, 1985): "The point is that the computer user [...] is not a real person. [L]arge sections of computer science are paralyzed by accepting this moron as their typical customer **[U]ser friendliness is, among other things the cause of a frantic effort to hide the fact that eo ipso computers are mathematical machines**"

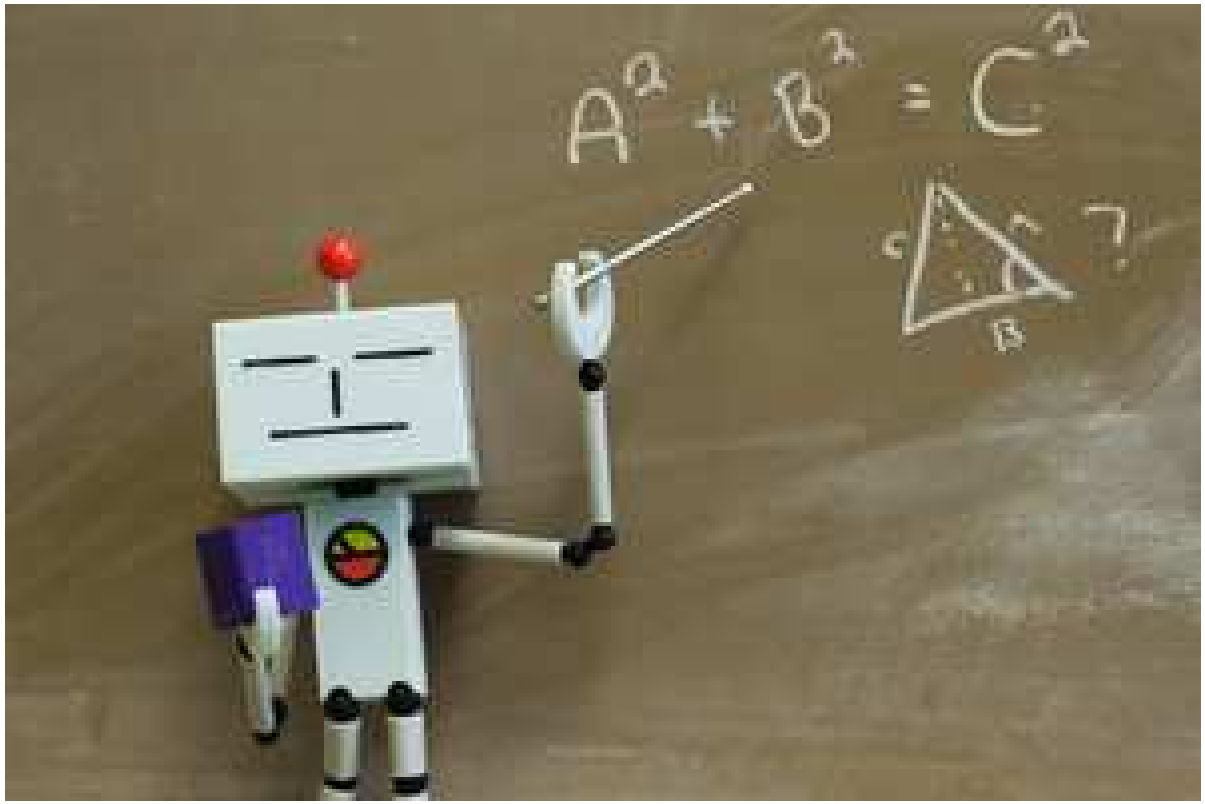
  $\Rightarrow$ Necessity to dig into the technical details of the machine and its programming!

- **The-fundamentally-different-assumption/inspiration: interaction with something that is fundamentally different from us and allow it as such**
  - Dijkstra, 1985: " Instead of trying to imitate what we are good at, I think it is much more fascinating to investigate what we are poor at. **It is foolish to use machines to imitate human beings, while machines are very good at being machines, and *that* is precisely something that human beings are very poor at.** Any successful AI project by its very nature would castrate the machine."

$\Rightarrow$ Not looking into MM to have a machine capable of human math, but one which is more suited for collaboration and interaction

General Approach(es)

What kind of MM?



General Approach(es)
What kind of MM (here)?

- Not: Computer not as a communication tool between humans (e.g. Gowers' blog – polymath)

⇒ Lehmer's explorative math – Human and machine collaborations over a reasonable amount of time – *humanly and machine impractical*:
Licklider, 1960: "Computing machines can do readily, well, and rapidly many things that are difficult or impossible for man, and men can do readily and well, though not rapidly, many things that are difficult or impossible for computers. **That suggests that a symbiotic cooperation, if successful in integrating the positive characteristics of men and computers, would be of great value.**" (Licklider, 1960)

General Approach(es)

Some previous 'results': a plurality of micro-approaches?

Research in progress

Some previous 'results': a plurality of micro-approaches (I)? "Reasoning with computer experiments in math"

⇒ Taking into account "material" and "social" changes of computer (changes in architecture, programming techniques, etc) in a study of computer-assisted math to detect global changes

⇒ Four (intrinsically related) core features of MM *per se*:

- ⇒ **Time-squeezing**
- ⇒ **Space-squeezing**
- ⇒ **Internalization**
- ⇒ **Mathematician-computer interactions**

⇒ Changing computer technology results in very different type of interaction from a process of clearly separated blocks of computer work (number-crunching; heuristics; less inspection) vs. human work (programming, inspecting and processing the results, publishing results etc) to an interaction which is more continuous with a mixing and distributing of computation, exploration and interpretation *into* the experimental process *between* C and H

⇒ Significance of 'time' in computer-assisted math

⇒ Husserl's paradox of progress (problem of 'hidden" knowledge in Maple and Mathematica – you need to forget in order to get at something new)

Some previous 'results': a plurality of micro-approaches (II)?
The Busy Beaver case (De Mol, 2011)

- Detailed study of a computer-assisted proof à la Lehmer from the late 70s, beginning 80s

⇒ Significant use of heuristic and explorative methods located *in-between* the mathematicians and the machine used (both contribute)

⇒ Idea of the proof is in the process (affects how proof is communicated) – as long as not all machines are tested, the result is entirely heuristic – the published proof is account of how it can be found.

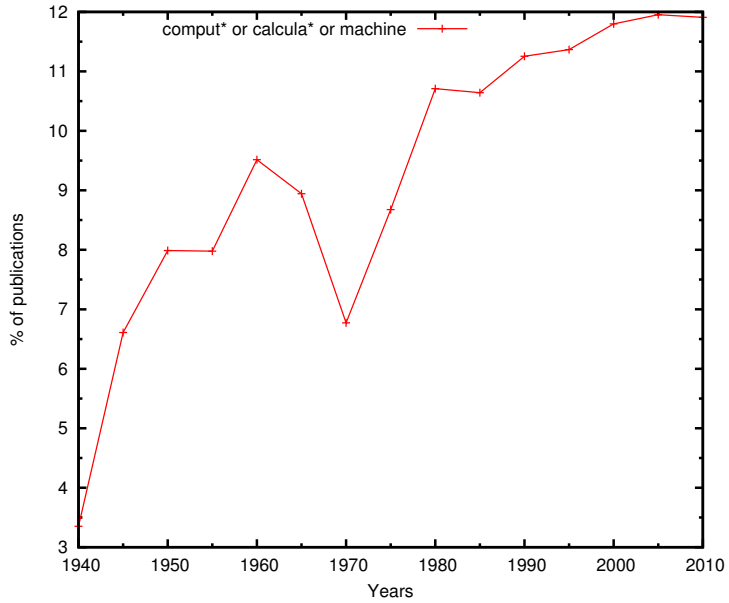
⇒ "Classical" problems (understanding, unsurveyability and problem error) *are* dealt with on a local level

  – E.g. Significance of corroboration

Some previous 'results': a plurality of micro-approaches (III)?

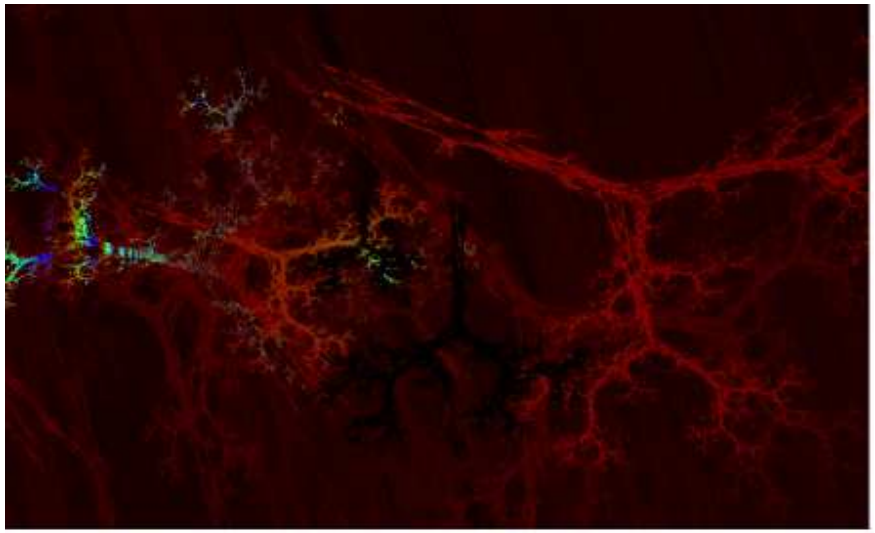
What is the impact of the computer on math? A quantitative approach (APMP, 2010)

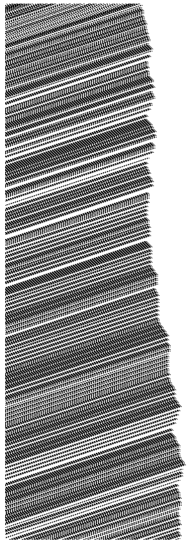
⇒ How to select relevant case studies and how to relate these "micro" studies with macro developments?



- Study quantitative evolution of usage computer-related terminology on on-line databases (MathSciNet; Zentralblatt; JSTOR)
- Yes, the computer does have a significant impact (quantitatively speaking)!
- Heuristics for finding where to look (e.g. finding that algorit* and program* are at least as important as compute*)

Part II: Two cases





Two cases

- Case I: Tag systems (De Mol, 2010; 2011)
- Case II (quick): A property of the chaos game? (De Mol, 2005)

⇒ Extremely tricky (!!)

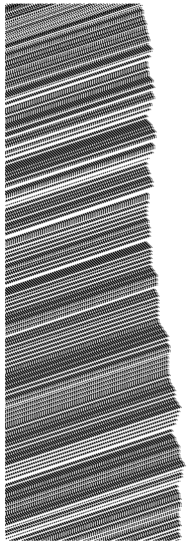
⇒ Where it all began....

⇒ Convey part of the *experience* and the kind of changes MM can result in. Two levels of 'change':

- The 'proof' is in the process – changing experience affects one's way of thinking and reasoning which is echoed in: "style", concepts, "proofs", communication (micro) – the 'practice'
- New types of results, methods and problems arising from the communication between human and machine (macro)

⇒ How to relate them?

Case I: Tag systems



Definition of tag systems

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \to 1101, 0 \to 00$
$A_0 = 101110111010000000$

Definition of tag systems

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \to 1101, 0 \to 00$

$A_0 = 101110111010000000$

~~101~~1101110101000000~~1~~1101

Definition of tag systems

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \to 1101, 0 \to 00$

$A_0 = 101110111010000000$

~~101~~1101110100000001101

~~110~~1110100000011011101

Definition of tag systems

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \to 1101, 0 \to 00$

$A_0 = 101110111010000000$

~~101~~1101110100000001101

~~110~~1110100000011011101

~~111~~0100000011011101101

Definition of tag systems

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \to 1101, 0 \to 00$

$A_0 = 101110111010000000$

~~101~~1101110100000001101

~~110~~1110100000011011101

~~111~~0100000011011101101

~~010~~0000011011101110100

Definition of tag systems

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \to 1101, 0 \to 00$

$A_0 = 101110111010000000$

~~101~~1101110100000001101

~~110~~1110100000011011101

~~111~~0100000011011101101

~~010~~0000011011101110100

~~000~~0011011101110100000

Definition of tag systems

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \to 1101, 0 \to 00$

$A_0 = 101110111010000000$

~~101~~1101110100000001101

~~110~~1110100000011011101

~~111~~0100000011011101101

~~010~~0000011011101110100

~~000~~0011011101110100000

~~001~~$\underbrace{01110111010000000}_{A_0}$

Definition of tag systems

Let T_{Post} be defined by $\Sigma = \{0, 1\}, v = 3, 1 \to 1101, 0 \to 00$

$A_0 = 101110111010000000$

~~101~~1101110100000001101

~~110~~1110100000011011101

~~111~~0100000011011101101

~~010~~0000011011101110100

~~000~~0011011101110100000

~~001~~$\underbrace{10111011101000000}_{A_0}$0

$\Rightarrow$ Definition of a *class* of symbolic logics according to a form

$\Rightarrow$ Two decision problems (finiteness problems) for tag systems: the halting and reachability problem (later, modified reachability problem, see (De Mol, 2010))

$\Rightarrow$ Logically equivalent to Turing machines

Studying tag systems with a computer: my own (frustrating) experiences (De Mol 2010; De Mol, 2011)

"Of course, unless one has a theory, one cannot expect much help from a computer [...] except for clerical aid in studying examples; but if the reader tries to study the behavior of [tag systems] without such aid, he will be sorry" (Minsky, 1967)

Goal?

⇒ To study Post's example in 'wider' context

What did I do?

- Theoretical results (reduction $3n+1$ problem to small TS (De Mol, 2007); proof of solvability class $\mu = v = 2$, involving "*considerable labor*" (De Mol, 2010))'
- Experimental results – six large computer experiments on the class of tag systems with $\mu = 2, v > 2$ (De Mol, 2011)
  Results?
  - No 'major' breakthrough – small results after about 1,5 year of research
  - Conjectures
  - 'Discovery' of the main types of periodic behavior
  - Loads of suggestions for further research – experimental and theoretical
  - Different classes of tag systems (according to periodic behavior and random behavior)

Where to start?

⇒ 'Pure' observation to build up intuition??? This means (very basically):

- **Writing** out a lot of tag systems on paper to get a feeling for them – the experience *in* the computational process (lost when programming!)
- Choosing a **programming language** (due to circumstances, BASIC)
- **Learning** a programming language (in my case)
- Necessity of developing some technical knowledge related to the language used
- Writing a program that 'visualizes' tag systems

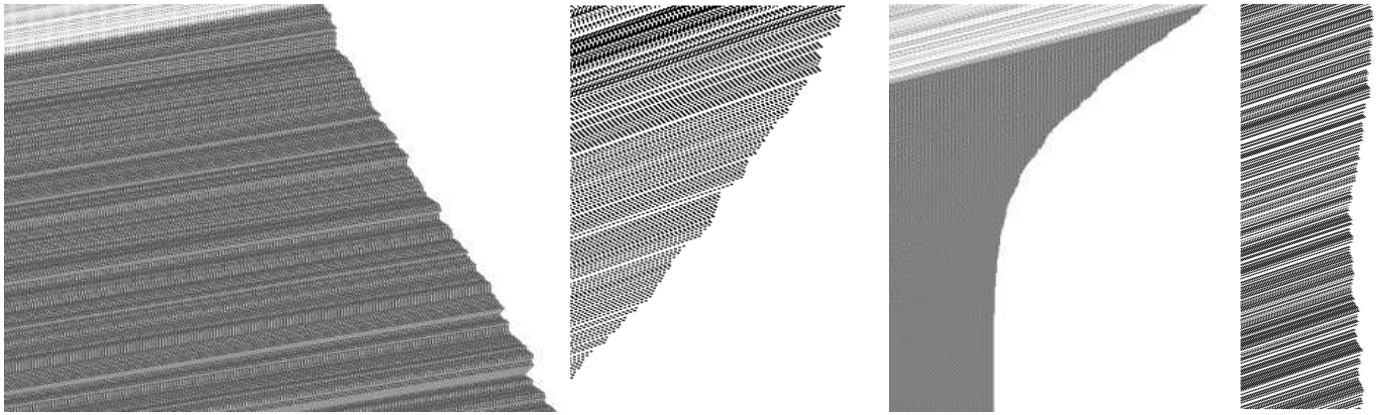
⇒ Already on this basic level, **lots of local and program-related questions** e.g. What is a **good** visualization? **Stepwise process of building code** which not only contains code for tag productions but also visualization, debugging, detection of classes of behavior etc

⇒ Observation of a process rather than being *in* the process

⇒ Questions determined by the "behavior" of the **processes** under investigation

- ↔ pencil-and-paper of Post: the need for developing intermediary language is extremely explicit.

What next? Focus on 'intractable' behavior



'Experiment' 0: How to study Post's TS *in context* experimentally?

- Need for other tag systems ⇒ constraints for intractable behavior – heuristic and theoretical constraints
- Generation of 50 different tag systems (originally 56 – 6 eliminated later with additional program)

⇒ **Process of writing code:** development of techniques depending on time and space efficiency; observations made etc, e.g. periodicity as **heuristic and 'ad hoc' programming**: "*The trouble with the brute-force method is that it requires too much brute force. For a pattern that runs through a million iterations, before entering a cycle, with an average string lengths of 1000 digits, the storage requirement would be at least a gigabyte. Furthermore, roughly 500 million string comparisons would be needed.*" (Hayes, 1986)

52 tag systems with $\mu = 2$.

Tag System	**Word$_0$**	**Word$_1$**	**Deletion Number v**
T1	00	1101	3
T2	00101	1011010	6
T3	111	01000	4
T4	11101	1100000	6
T5	010110	11100100	7
T6	0	01011	3
T7	101011	00011010	7
T8	011	111100	5
T9	101	0000111	5
T10	001	10110	4

Experiment 1: Distribution of the behavior

- For each TS, test 2000 initial words, running for at most 10,000,000 computation steps – count number of periodic words, unbounded growth, halts and 'immortals?'

$\Rightarrow$ Took weeks – day and night of work (observations, interruptions, etc) – dealing with the physicality of computer: the toilet experience!

$\Rightarrow$ On the problem of error: how do I know that I the program 'works'? Different ways of testing by interrupting – necessity to make intermediary observations!!

$\Rightarrow$ Observation exponential decade of 'hold-outs' resulting in extension experiment (and learn to work with MetaPost)!

Experiment 1: Results $\Rightarrow$ Plot nr. of words that have not halted, become periodic or led to unbounded growth vs. Number of computation steps.

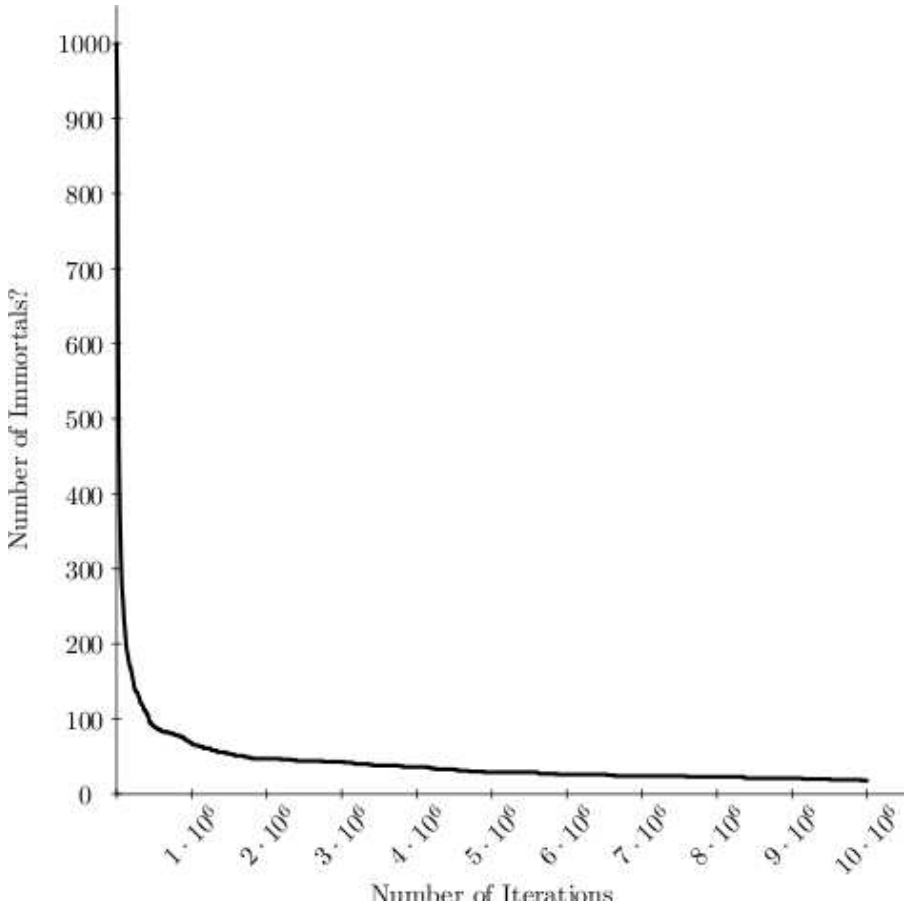


Figure 1: Plot of $\mathbf{T_{Post}}$

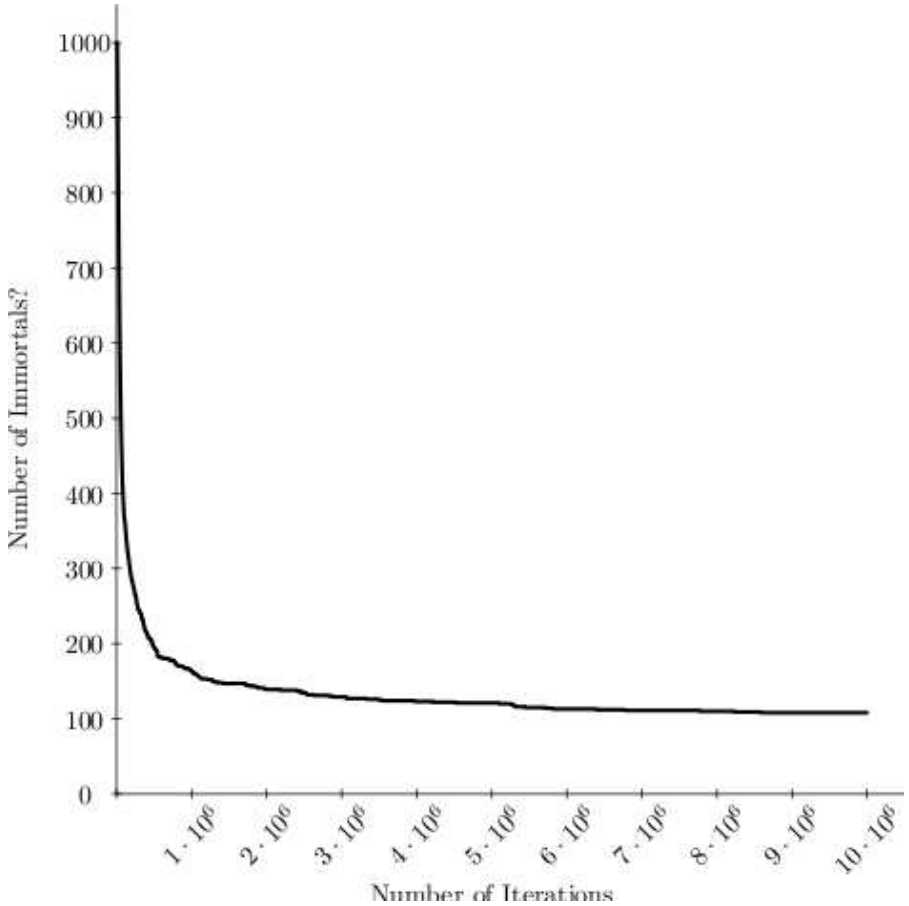


Figure 2: Plot of **T2**

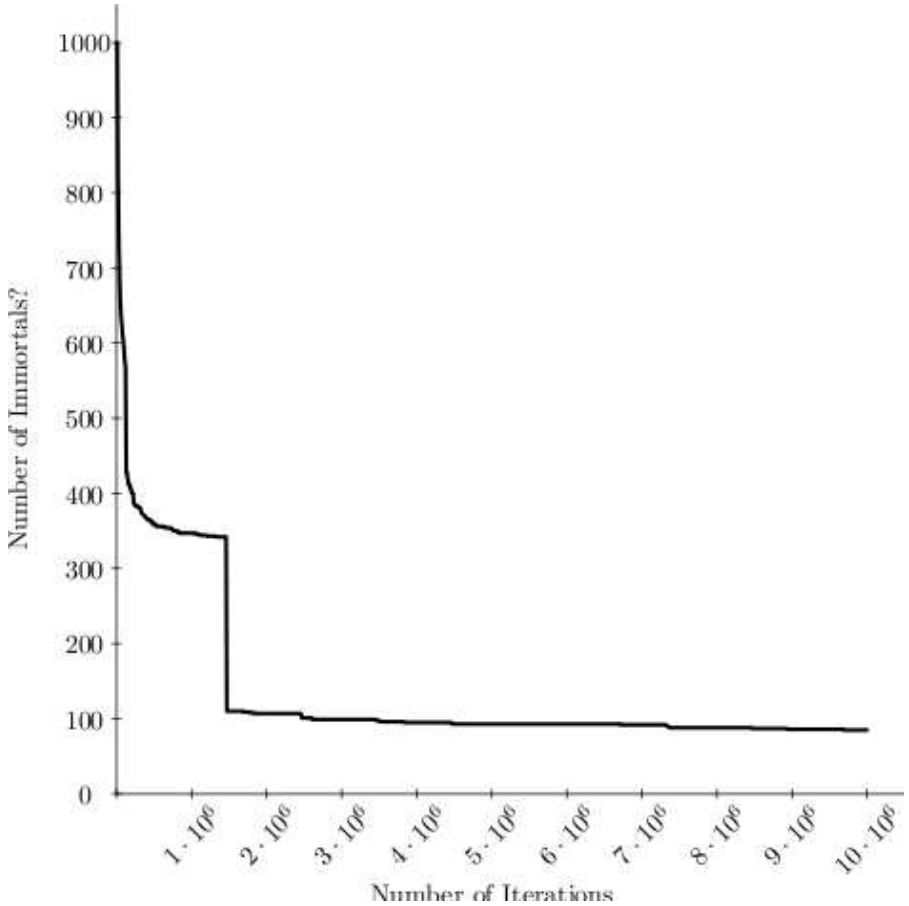


Figure 3: Plot of **T38**

Experiment 2: Periodic behavior

- Discovery of two new periods in Post's tag systems (Watanabe!): determine typography of periods
- Based on data experiment 1: new code for 'classifying' all the periods

⇒ 32 MB file

```
------------------
"There were ",3," different structures for period ",7
"Structure ",1
"000011100001110000111000011100"
"000111000011100001110000111000"
"001110000111000011100001110000"
"011100001110000111000011100001"
"111000011100001110000111000011"
"110000111000011100001110000111"
"100001110000111000011100001110"
"Structure ",2
"011100001110000111000011100001110000111000011100001"
"111000011100001110000111000011100001110000111000011"
"110000111000011100001110000111000011100001110000111"
"100001110000111000011100001110000111000011100001110"
"000011100001110000111000011100001110000111000011100"
"000111000011100001110000111000011100001110000111000"
"001110000111000011100001110000111000011100001110000"
"Structure ",3
"000011100001110000111000"
"000111000011100001110000"
"001110000111000011100000"
```

⇒ 'Feeling' of human-machine collaboration was extremely real here – constant interaction between writing code, using previous knowledge, combine it with new observations, formulate new code etc:

⇒ Interaction theory and observation: Resulted in a typography of periodic behavior in tag systems (originally four, but reduced to two main types and two subtypes): theoretical results on computing with periodic behavior

Experiment 3–6: Complex behavior?

- Experiment 3: Distributions 0s and 1s – motivated by difference between 'definition' and the process it defines!!

  > Note, in the (00, 1101) problem, that the read head advances three units at each step, while the write head advances by two or four units. Statistically, one can see, the latter has the same average speed as the former. Therefore, one would expect the string to vanish, or become periodic [...] Is there an initial string that grows forever, in spite of this statistical obstacle? No one knows.

Experiment 3–6: Complex behavior?

- Experiment 4: The DIEHARD test for randomness:

  > I hope you will inform me of results, good or bad, of new kinds of generators you have tested, particularly deterministic generators, but also the output of physical devices.(I have found none of the latter that get past DIEHARD, and would like to learn of any that do.) [I]n my opinion, there is no true randomness

- (Experiment 5: Lyapounov exponents)
- Experiment 6: Measuring the information entropy: the case of the binary RB tree....On being side-tracked)

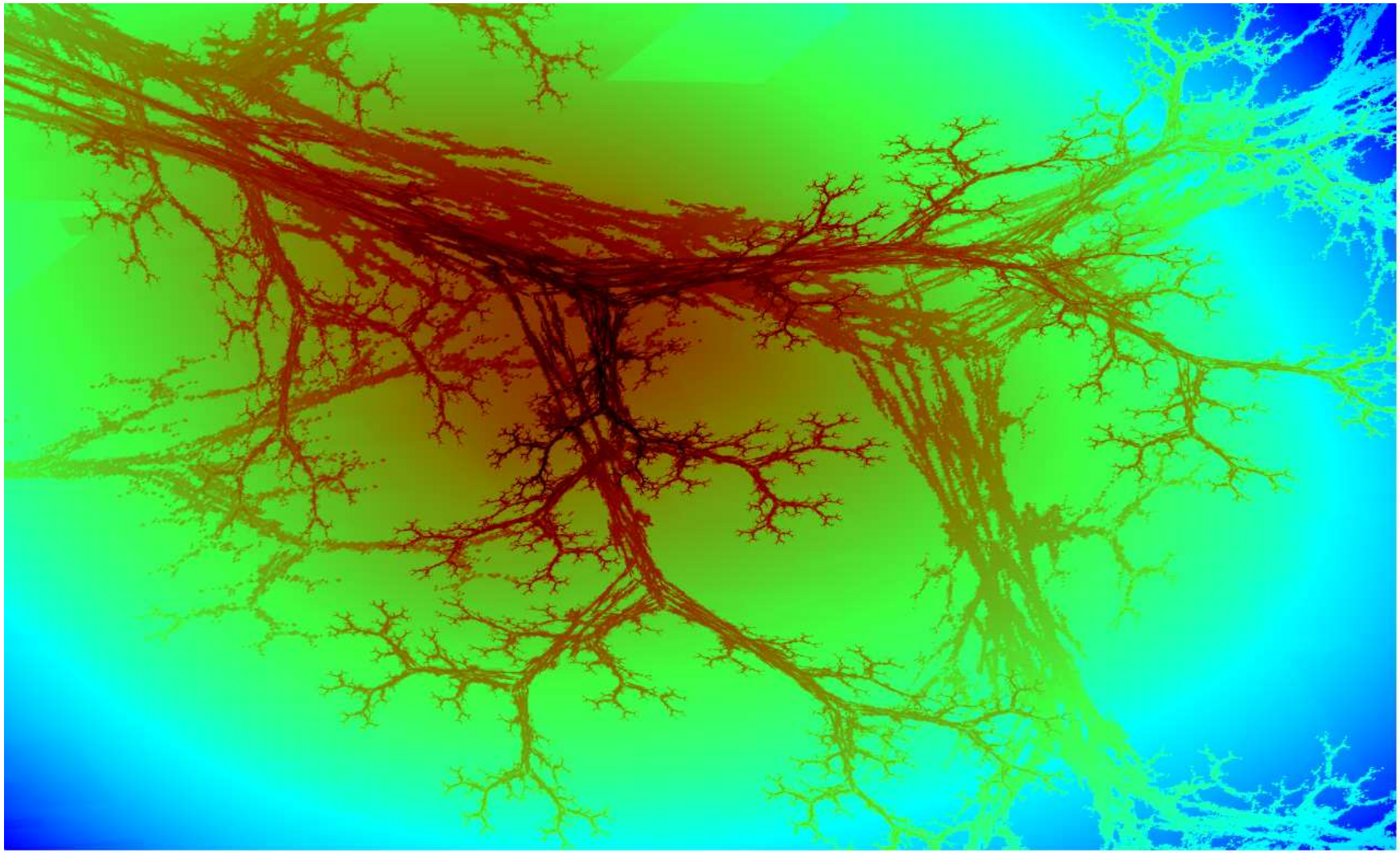
⇒ Coding as a process of integrating background and observational knowledge + t+s limitations

⇒ You are not just working on e.g. TS! Many side- and background knowledge – 'wide instrumentation' (Sörensen, 2010): mixing of tools from different disciplines → Computer science as an anarchistic field (Tedre, 2006)

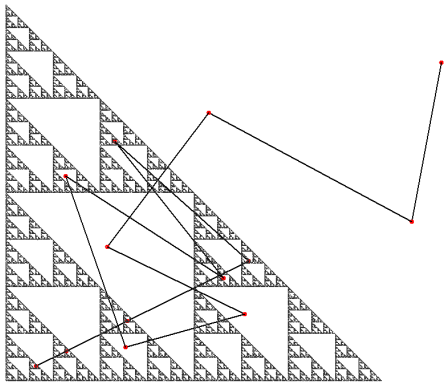
⇒ NKS (Wolfram) and EM (Borwein, et al) developments as attempts to 'unify' this 'wide instrumentation'

⇒ (Husserl's paradox? The more side knowledge you need to search, implement, develop the less time you have to spend on the object at hand)

Case II: A property of the chaos game?



Case II: A property of the chaos game? (De Mol, 2005)



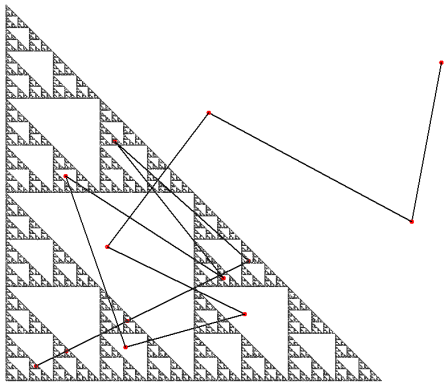
Sierpinski triangle

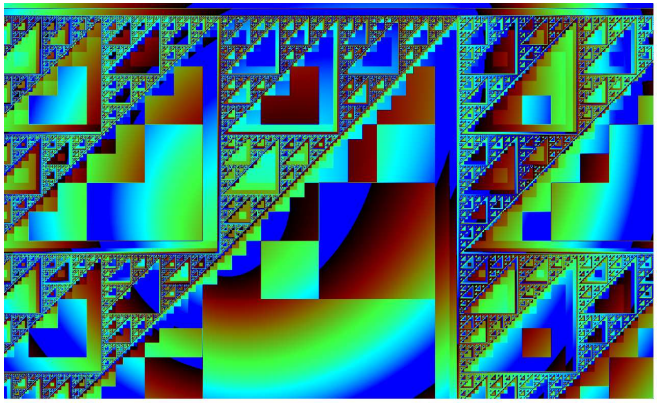
$f_1(x_i, y_i) = (0.5x_{i-1}, 0.5y_{i-1})$

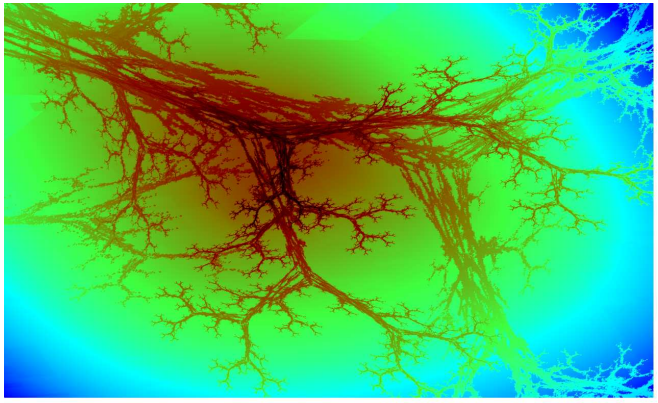
$f_2(x_i, y_i) = (0.5x_{i-1} + 0.5, 0.5y_{i-1})$

$f_3(x_i, y_i) = (0.5x_{i-1}, y_{i-1} + 0.5)$

Case II: A property of the chaos game? (De Mol, 2005)







Sierpinski triangle

$f_1(x_i, y_i) = (0.5x_{i-1}, 0.5y_{i-1})$

$f_2(x_i, y_i) = (0.5x_{i-1} + 0.5, 0.5y_{i-1})$

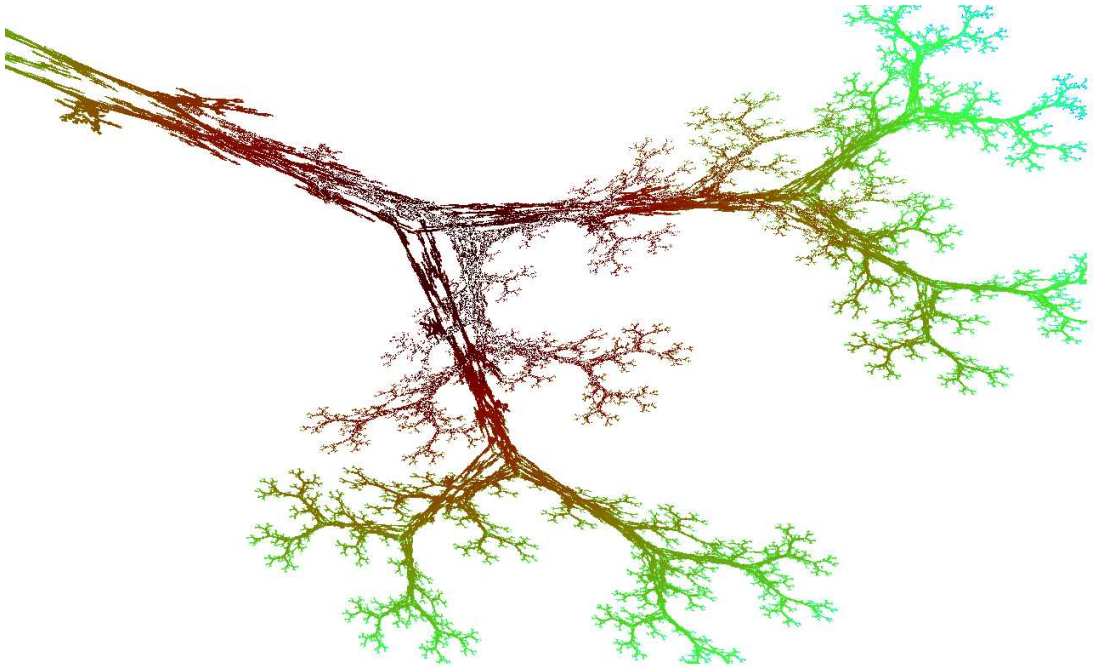
$f_3(x_i, y_i) = (0.5x_{i-1}, y_{i-1} + 0.5)$

$\Rightarrow$ Side-tracked again: the means became the result

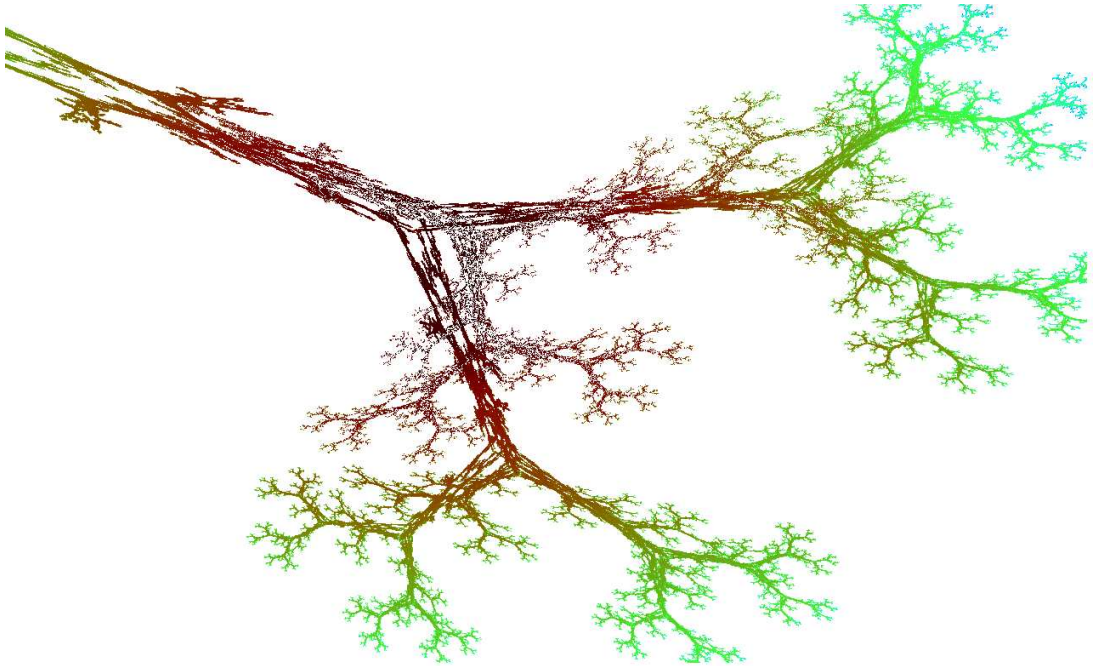
Case II: A property of the chaos game?

Time and space squeezing: A "toy" example II....

Case II: A property of the chaos game?



Case II: A property of the chaos game?



⇒ Squeezing objects and their properties in humanly observable space (zooming-out)

⇒ "*When seeking new insights, I look, look, look and play with many pictures (One picture is* never enough)" (Mandelbrot, 2004)

⇒ The 'surprise' factor adds to the 'feeling' of exploration

⇒ How do I know that the result is 'true'?

Discussion

Discussion (1) The proof is in the process?

- Work on finite approximations/processes of possibly infinite objects – with all the problems that brings to 'experimental' math:

  > It is certainly tempting to use computers to try to tame these wild beasts. [...] Im afraid the results can be quite frustrating. There is the problem of the infinite tail. Any computation can only show the beginning and there is no way to know how well the patterns one can discern in the results of such a computation apply to the far out regions computation cannot penetrate.

  (Martin Davis, private communication)

- Confrontation with unpredictability and processes of construction; not "stable", "eternal" objects but finite and dynamical objects ("live " math) – you study the process not the object; **focus on other types of problems**

- The process is also in the **writing and developing of code**, sensitive to my own (limited) knowledge, the language I use and the answers I get from the machine → results are at the interface of complex set of changing and developing interactions.

Discussion (1) The proof is in the process?

- Time is also in the code itself:

  > Mathematical theories make use of reversible time [...] Let us note that in our discrete time of computations, time is irreversible: it is very often extremely difficult to run an algorithm backward. At the highest level of generality it is impossible. Let us note that the formal definition of sequential algorithms induces an irreversible time which is present in the very notations [e.g. an assignment].

  (Maurice Margenstern, 2012)

- The machine as a physical object restricted by time and space – in the programming you *need* to deal with time and space issues (e.g. heuristic programming).

- Algorithmic thinking is by its very nature oriented towards processes

$\Rightarrow$ Does the computer re-inject time into math?

Discussion (2) Taking the machine seriously. Towards Human-machine conversations?

- Experiments as Human-machine conversations that develop over time?

⇒ Communication through intermediary languages (programing language; programs; visualizations; data etc) that are being 'adapted'/used *during* interaction – the communication process as it develops in the programming and computer feedback is highly unpredictable; time sensitive and has some of the properties of an interesting conversation

⇒ 'Content' determined by the feedback; surprises and unpredictability: you do not control the machine! "*[W]e only know what we have said, when we have seen our listener reacted to it; we only know what the things we are going to say will mean in as far as we can predict his reaction. [...] As we do not master the behavior of the other, we badly need in speaking the feed back, known as "conversation".* (Dijkstra, 1961)"

Discussion (2) Taking the machine seriously. Towards Human-machine conversations?

⇒ One of the main aims of man-computer symbiosis is **to bring the computing machine effectively into the formulative parts of technical problems**. The other main aim is closely related. It is **to bring computing machines effectively into processes of thinking that must go on in "real time"**, time that moves too fast to permit using computers in conventional ways. [...]To think in interaction with a computer in the same way that you think with a colleague whose competence supplements your own will require much tighter coupling between man and machine" (Licklider, 1960)

⇒ Both human and machine drawn into the communicative process

⇒ It is not enough to investigate the cognitive science of human mathematicians! What about the 'cognitive' science of the machine as a non-human?

⇒ How to pin down the 'power' of this interaction?

⇒ Applications? We need an appropriate model for 'conversing' with the machine that takes into account its unpredictability; its several means to communicate; its particularities etc drawing from formal properties of HH-conversation (e.g. turn-taking; error-correction etc)?

Discussion: some further points

"*unless he loves his tools it is highly improbable that he will ever create something of superior quality*" (Dijkstra, 1962)
⇒ Time, unpredictability and conversational mode as essential features of MM? Affects not only the 'experience', but, because of that, also the math that results from it! BUT:

- My results are very much determined by a particular view (this is unavoidable!)
- Necessity of plurality of approaches

⇒ We *do* need a good theory of computer-assisted mathematics and, ultimately, the machine itself (just as we need a better phil. of math.)

By way of a challenge.....

"*If computers are the first machines to reduce the contingency or incomputability of some, though not all futures to a finite degree, its own contingency should remain as open as possible. [...] If somebody went and wrote all the programmes hitherto running under the name of philosophy into hardware, that would be the goal itself.*" (Kittler 1987)