Robustness in computer science

Dave Ackley
University of New Mexico

Santa Fe Institute
Complex Systems Summer School
June 20, 2011

Robustness in computer science

- **Part 1: Looking back**
  - History: Our computers are fragile by design
  - Programmability, robustness & efficiency
  - Computer architecture in space & time, examples
- **Part 2: Looking forward**
  - Serial vs parallel scaling
  - Thought experiment: Indefinite scalability
  - Computer architecture as actual architecture
  - Example: The Movable Feast Machine

Robustness principles and traditional computing

- Modularity
- Multiple pathways
- Redundancy
- Feedback control
- Sloppiness
- Spatial compart-mentalization
- Canalization & niche construction
- Purging
- Conflict mechanism
- Error-correction & repair
- Distributed processing

Robustness principles and traditional computing

- **Modularity**
- Multiple pathways
- **Redundancy**
- Feedback control
- Sloppiness
- Spatial compart-mentalization
- Canalization & niche construction
- Purging
- Conflict mechanism
- **Error-correction & repair**
- Distributed processing

Are you a machine?

Yes
No
Mu

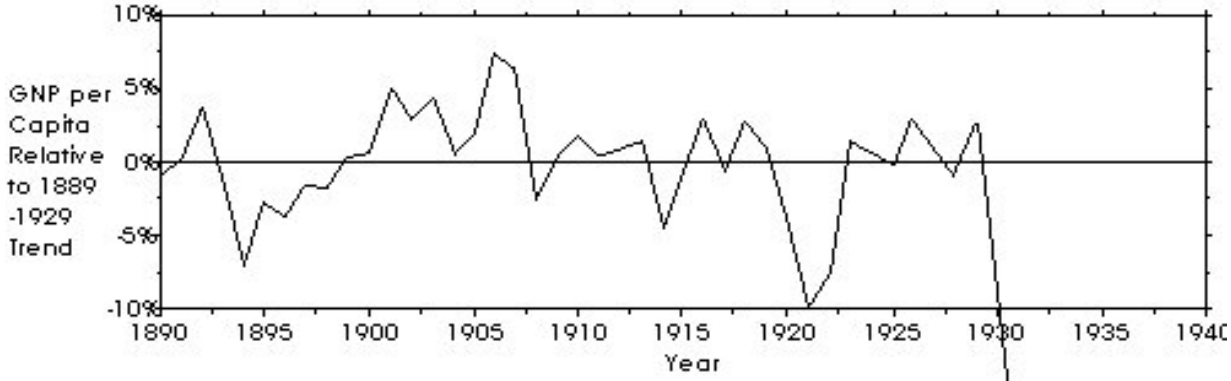
“Are you a machine? And what of it?”

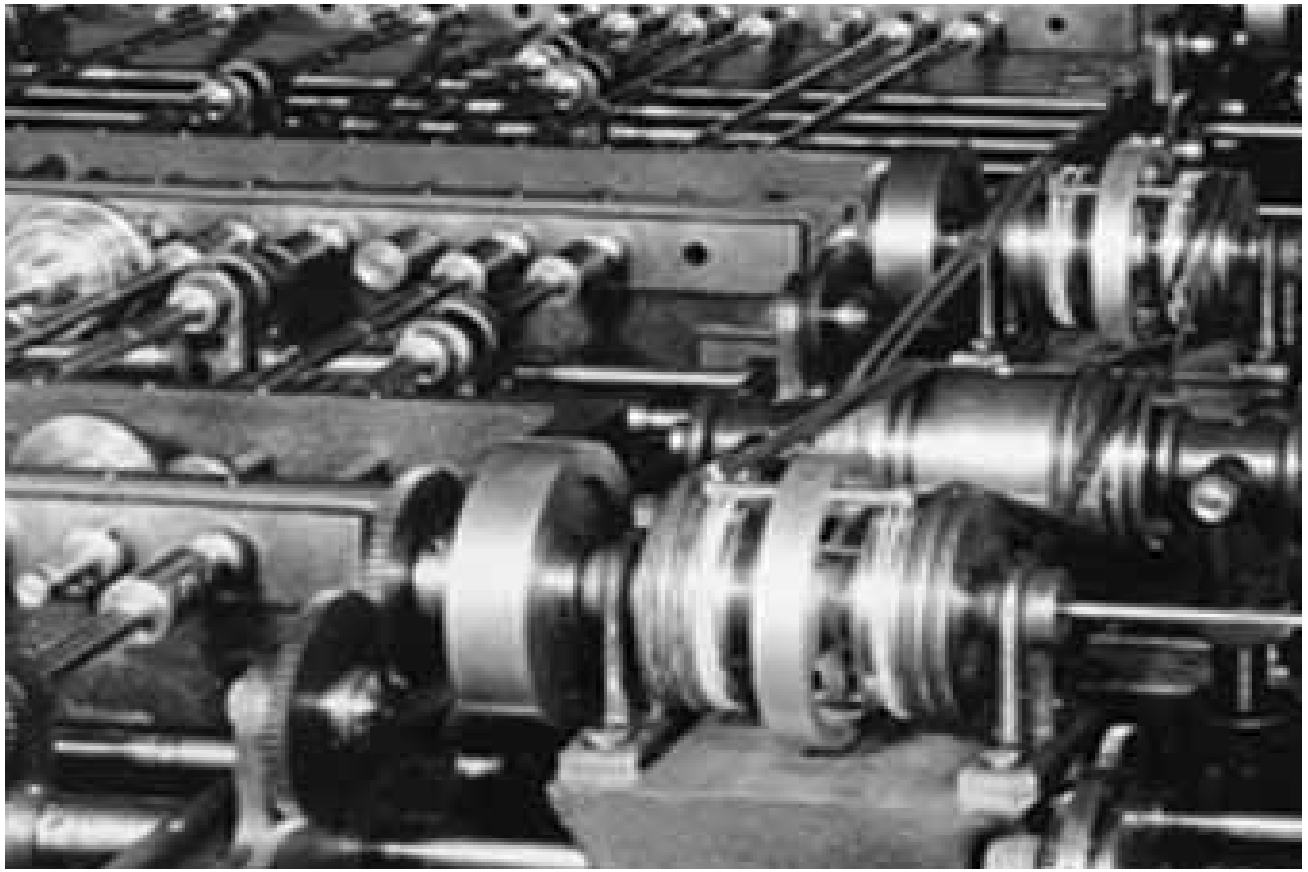
— W.H.Roberts, in the *Journal of Philosophy,* ***1931***

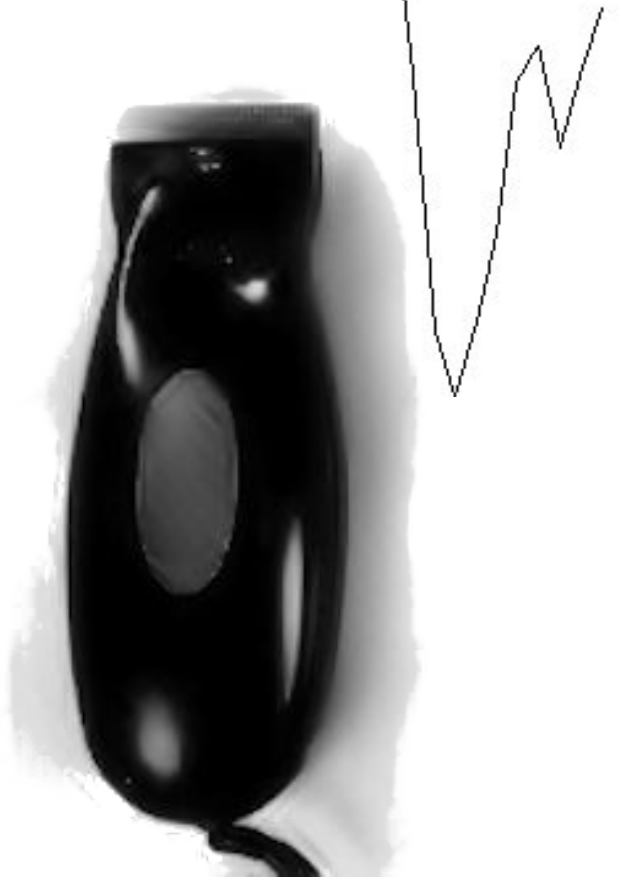
- Positive qualities:
  - Accurate
  - Regular
  - Dependable
  - “Transparency of relationships”
- Negative qualities:
  - Lacks knowledge of self, uses, effects
  - Has no purpose
  - Has no feelings
  - Actions are rigidly determined
  - Unable to adjust to changes

But, also in 1931

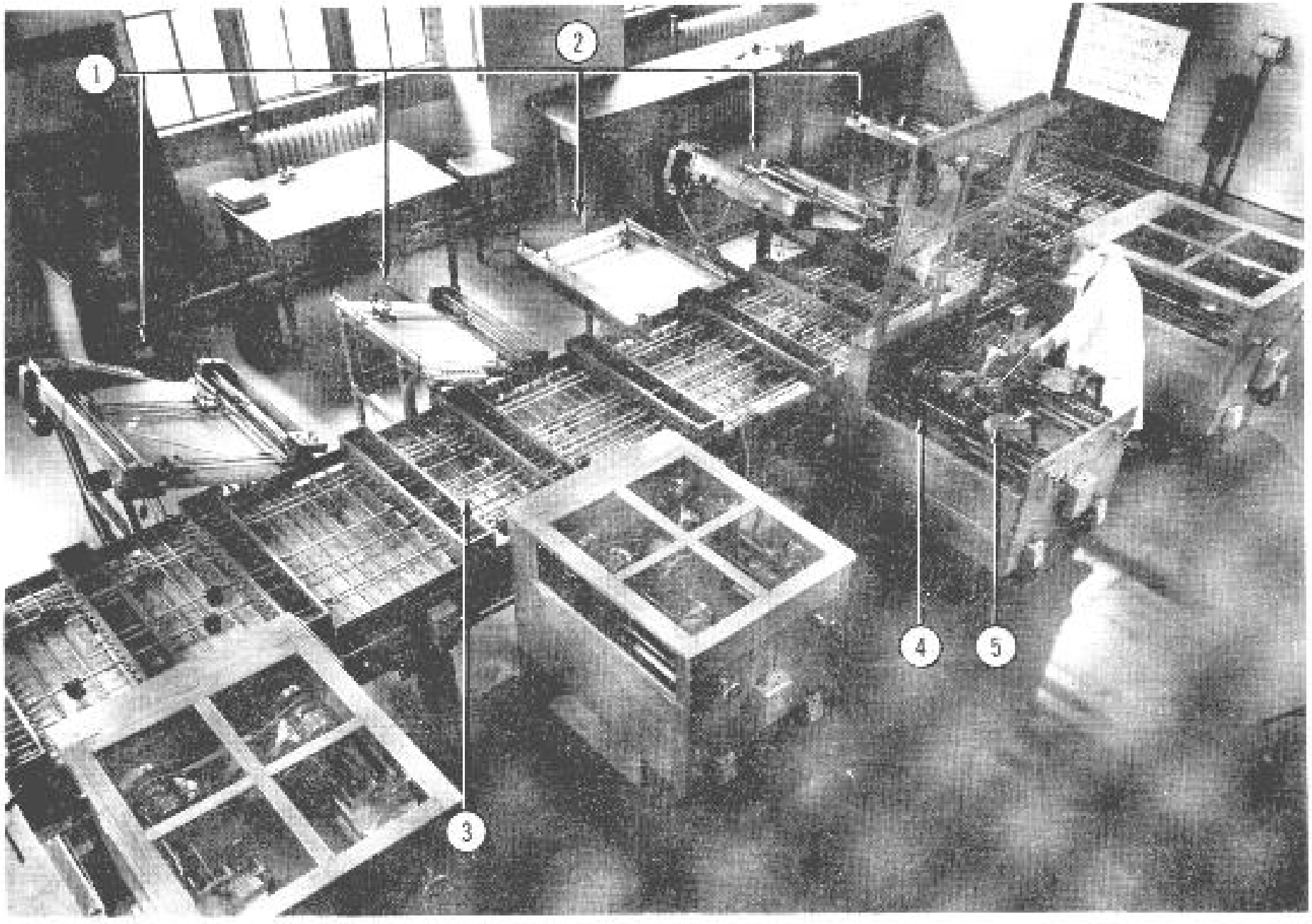
Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I





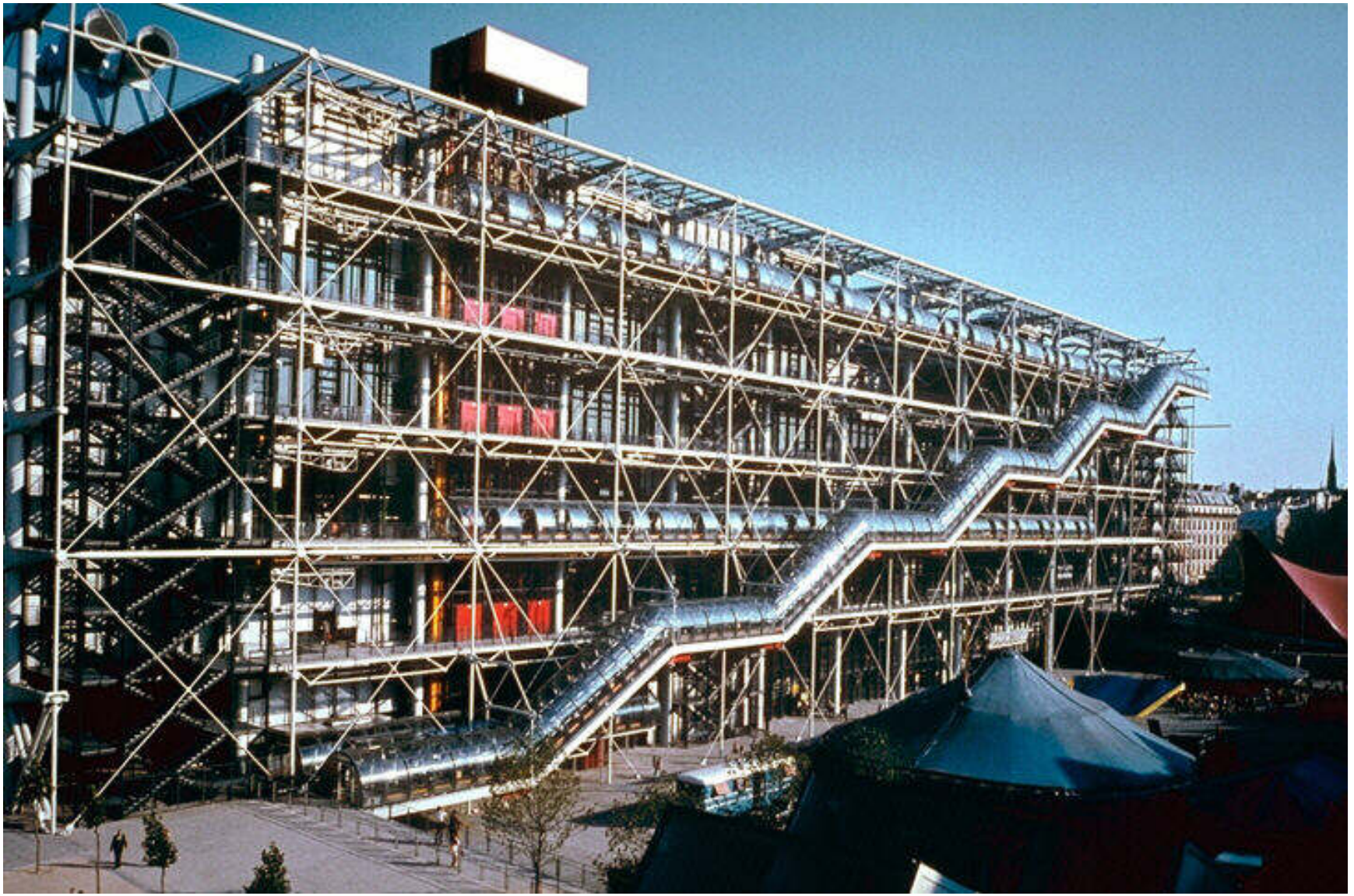


The Bush differential analyzer

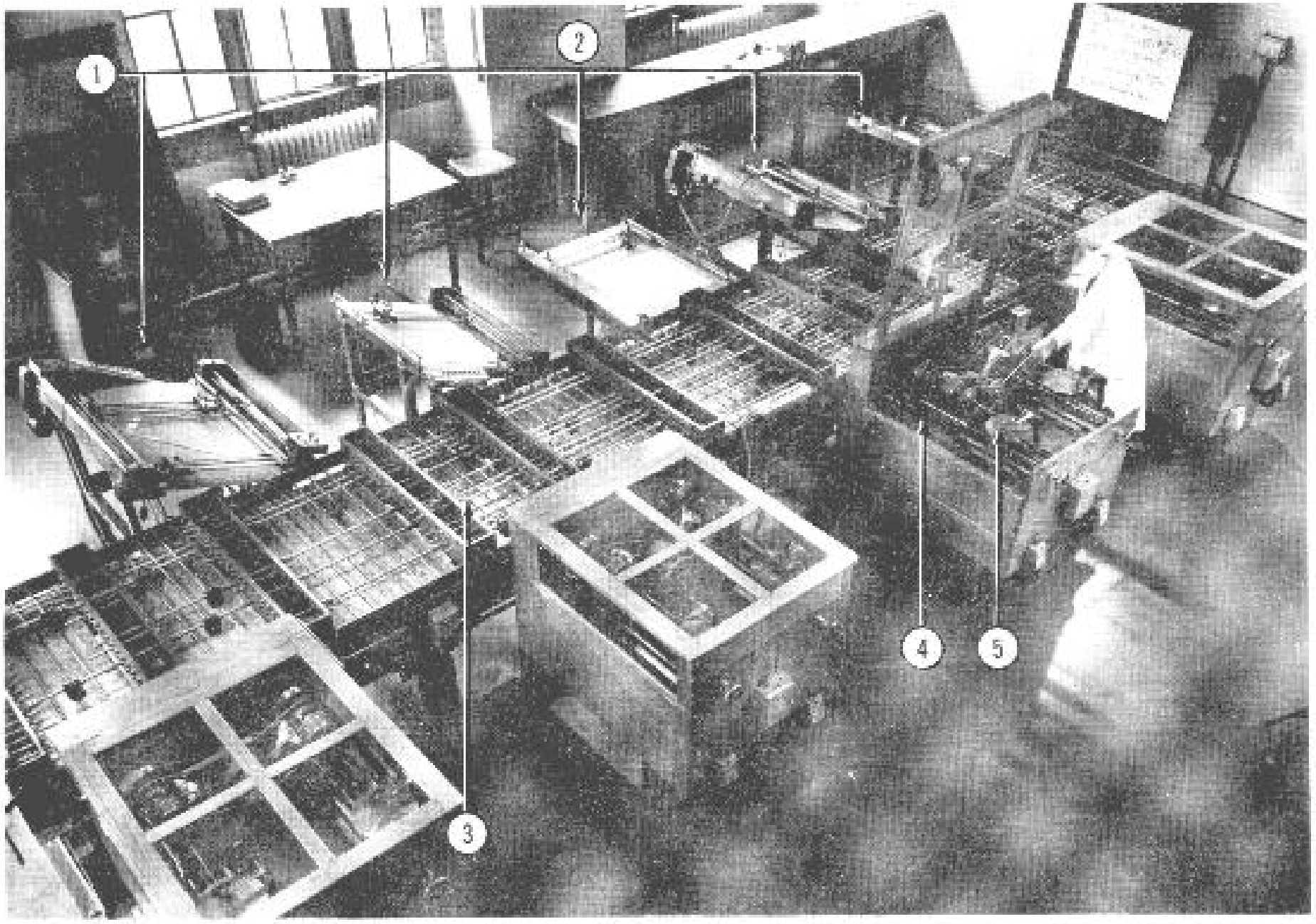


1 Input table
2 Output table
3 Shafts and gears used for interconnection
4 Torque amplifier
5 Integrator disk

Sidebar: What is this?

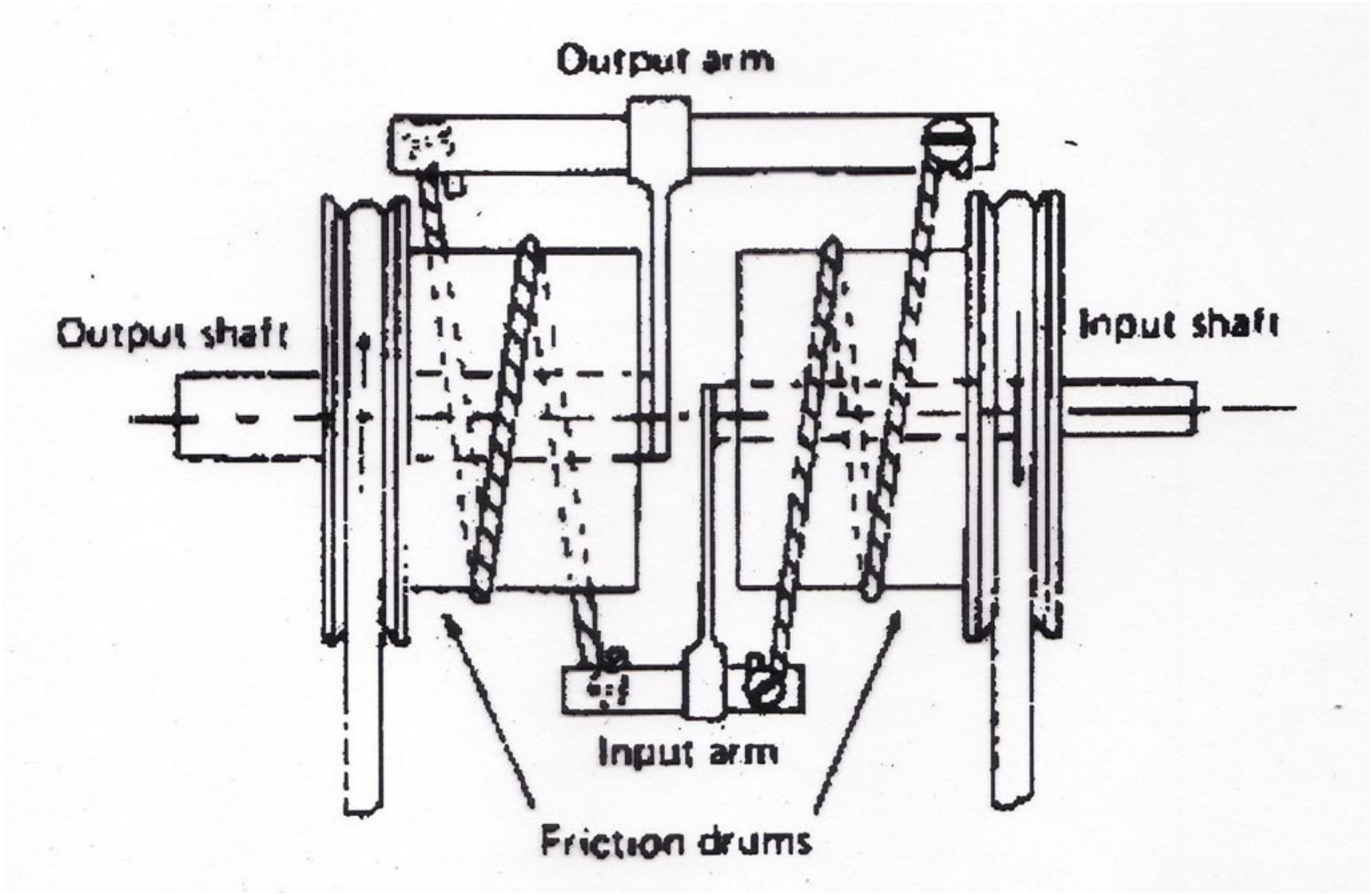


The Bush differential analyzer

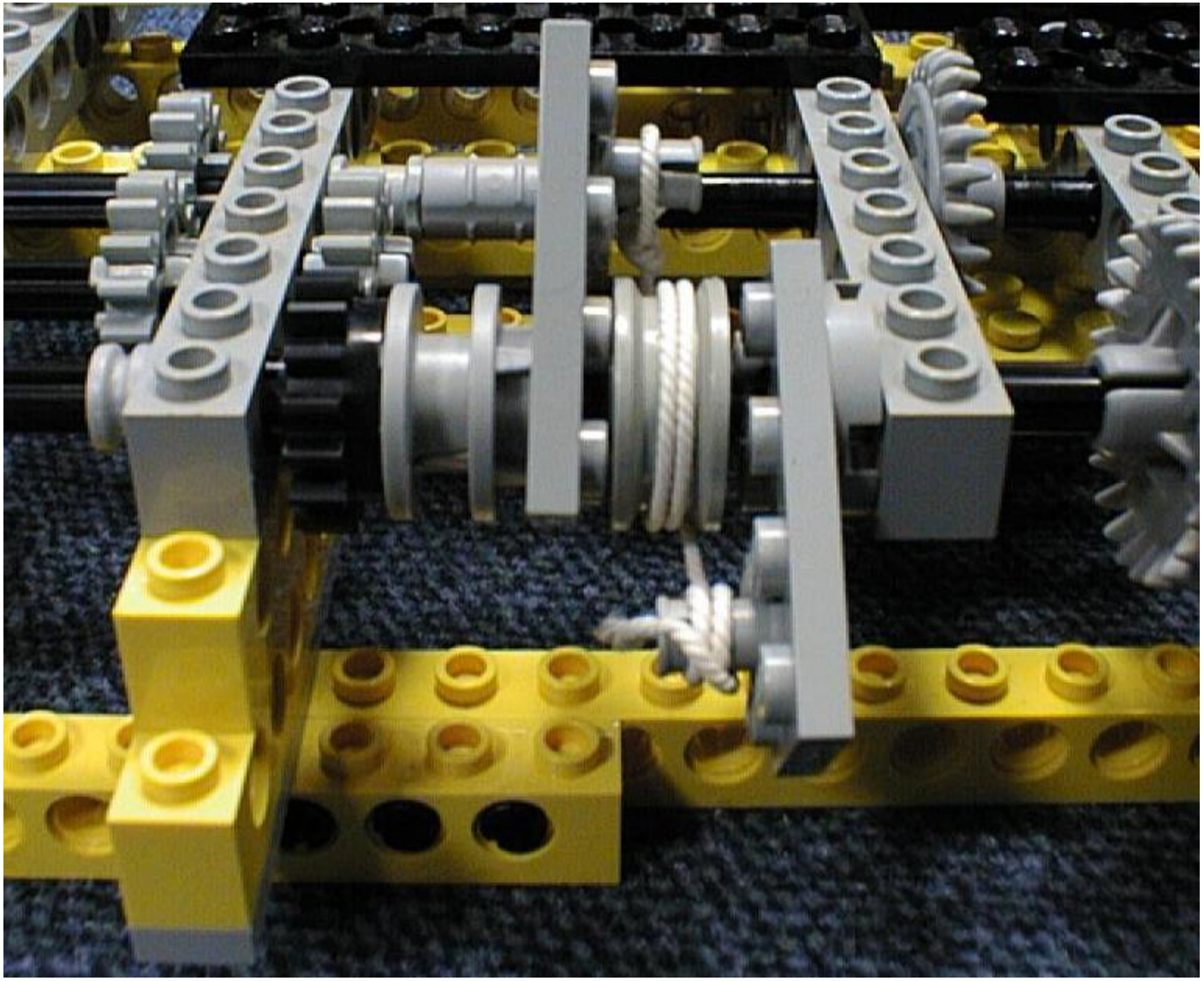


1 Input table
2 Output table
3 Shafts and gears used for interconnection
4 Torque amplifier
5 Integrator disk

The Torque Amplifier



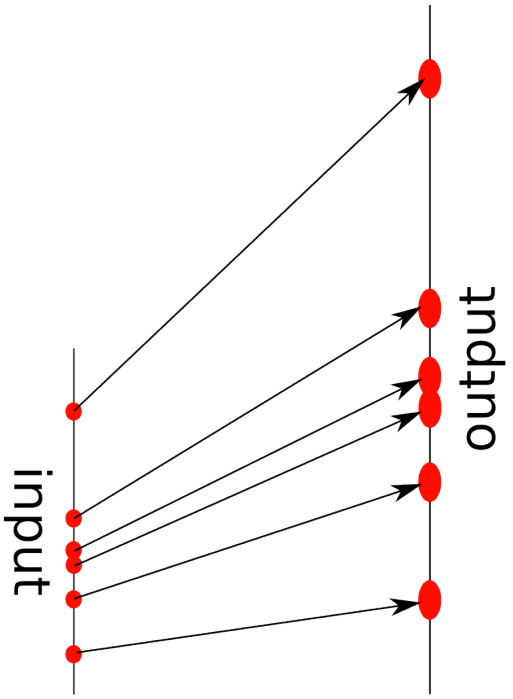
Torque Amplifier Today



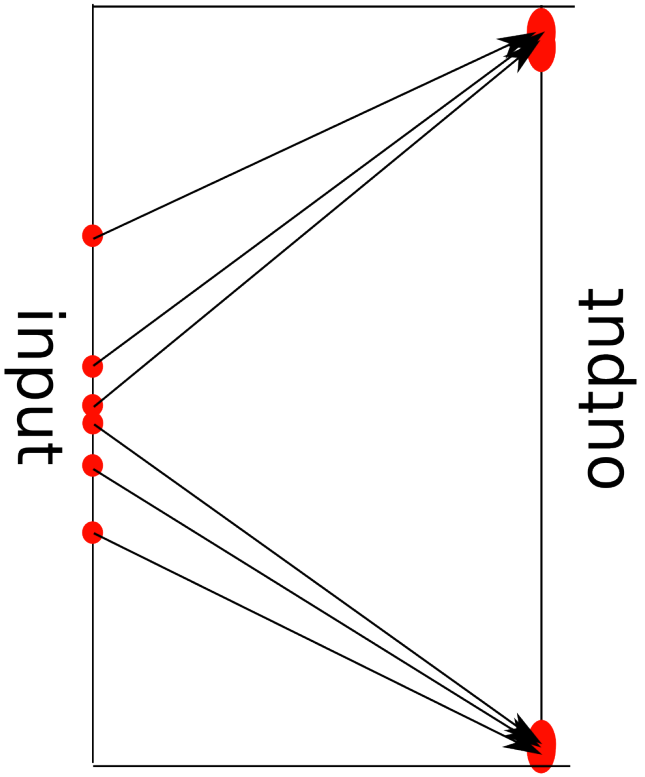
http://homepage.mac.com/a.eppendahl/work/torque-amp.html

The problem is error

Analog computation



Ideal linear amplifier ->
No information loss



Ideal non-linear amplifier ->
Massive information loss

Digital computation

The problem is error

- John von Neumann, 1952, *Probabilistic logics and the synthesis of reliable organisms from unreliable components*

Fig. 1

Figure 14.

$$m(a,b,c) \equiv ab + ac + bc \equiv (a+b)(a+c)(b+c) :$$

Reliability from Unreliability

- Suppose: $\varepsilon < ½$ is the probability of error per "organ" (assume i.i.d.).
- **Question**: Given $\delta > 0$, can a machine made from such organs that will give the correct answer with probability $\leq \delta$?
- **Observation**: There must be a "final output organ" of the whole machine. It will fail with probability ε. So, $\delta \geq \varepsilon$.
- **Thought:** But if we redefined the output to be a *majority* of *several* lines..

Reliability from Unreliability

- *(von Neumann 1952)*: Triplicate each input line and connect them to a three input majority organ. Assume each replicate fails with probability η, and the majority organ fails with probability ε. Then if:

  $$\varepsilon + 3\eta^2 < \eta,$$

  the output of the majority organ is *more reliable* than its input.

- For this construction, $\eta < 1/6 = \sim 16\%$, and $\varepsilon < \sim 8\%$, to keep error from growing.

Robustness for scale

Analog: An amplifier per *integration:*

$$\frac{d^2x}{dt^2} + k\frac{dx}{dt} + g = 0$$

Complex math via analog computation: Two amplifiers

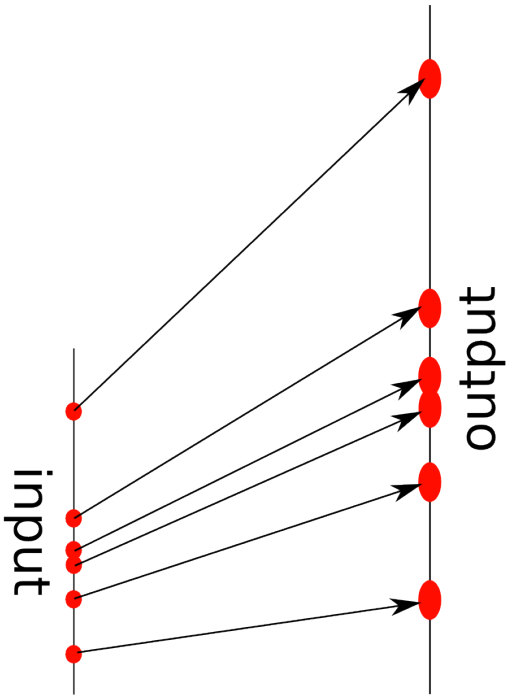
Digital: An amplifier per *bit*

```
x = y+1;
```

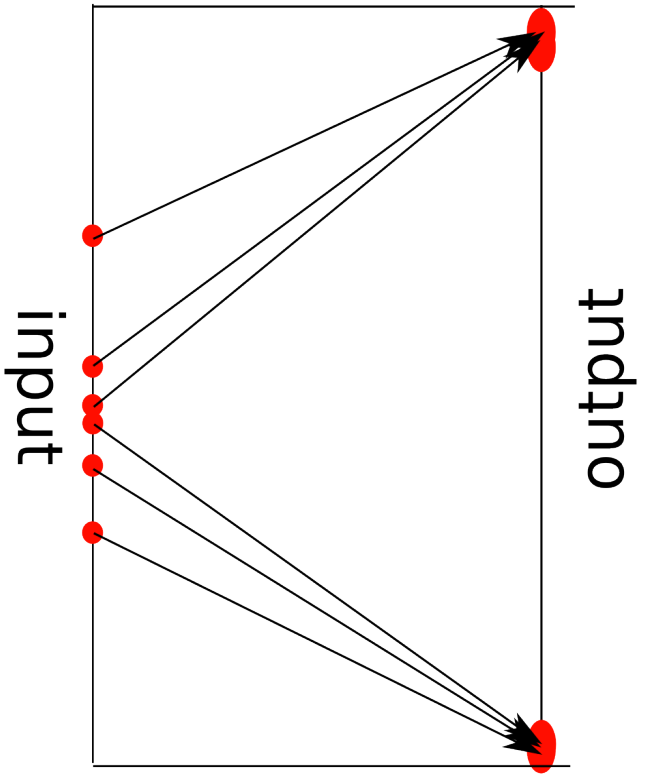
Simple arithmetic via digital computation: *Dozens* of amplifiers

Non-linear amplifiers: "Restoring organs"

Analog computation



Ideal linear amplifier ->
No information loss



Ideal non-linear amplifier ->
Massive information loss

Digital computation

Robustness for scale

Digital processing is *massively* redundant

The Faustian Bargain:

Hardware shall be *reliable*

Software shall be *efficient*

Efficiency and Robustness Are Mortal Enemies

Efficiency and Computer Science

- The coin of the realm
- *"Big O"* asymptotic analysis
- Bubble sort vs quick sort
- The framing of the problem
- The problem of the framing

Sixty years of efficiency later: State of the art

- Software:

- Hardware:

And hardware's getting less stable..

Computer Architecture

- A physical computation system is a map between concrete reality and abstract mathematics
  - Reality: Seconds, centimeters, watts...
  - Mathematics: Sums, songs, spam...
- Architecture#1: The serial computer:
  - Single active *Central Processing Unit* "CPU"
  - Large passive *Random Access Memory* "RAM"
  - Control flow: Input → Process → Output

Robustness & serial computation

- What is the dimensionality of the CPU+RAM model of computation?
  - All opportunities for spatial compartmentalization have already been abstracted away.
- What is modularity in CPU/RAM model?
  - Temporal and/or illusory
- What can happen after an error or a bug?
  - Anything

Robustness principles

- **Modularity**
- **Multiple pathways**
- **Redundancy**
- **Feedback control**
- **Sloppiness**
- **Spatial compart-mentalization**
- Canalization & niche construction
- Purging
- Conflict mechanism
- **Error-correction & repair**
- **Distributed processing**

How would you do it better?

- Computer security is today a nightmare / bad joke / oxymoron / impossibility / …
- OK, you're a computer designer: How could you use the robustness principles to ***compute better***?

End of part 1

Thus the logic of automata will differ from the present system of formal logic in two relevant respects.

*1. The **actual length of "chains of reasoning,"** that is, of the chains of operations, will have to be considered.*

*2. The operations of logic (syllogisms, conjunctions, disjunctions, negations, etc., that is, in the terminology that is customary for automata, various forms of gating, coincidence, anti-coincidence, blocking, etc., actions) will all have to be treated by procedures which allow exceptions (**malfunctions**) with low but **non-zero probabilities**.*

—John von Neumann 1948

Robustness in computer science

- **Part 1: Looking back**
  - History: Our computers are fragile by design
  - Programmability, robustness & efficiency
  - Computer architecture in space & time, examples
- **Part 2: Looking forward**
  - Serial vs parallel scaling
  - Thought experiment: Indefinite scalability
  - Computer architecture as actual architecture
  - Example: The Movable Feast Machine

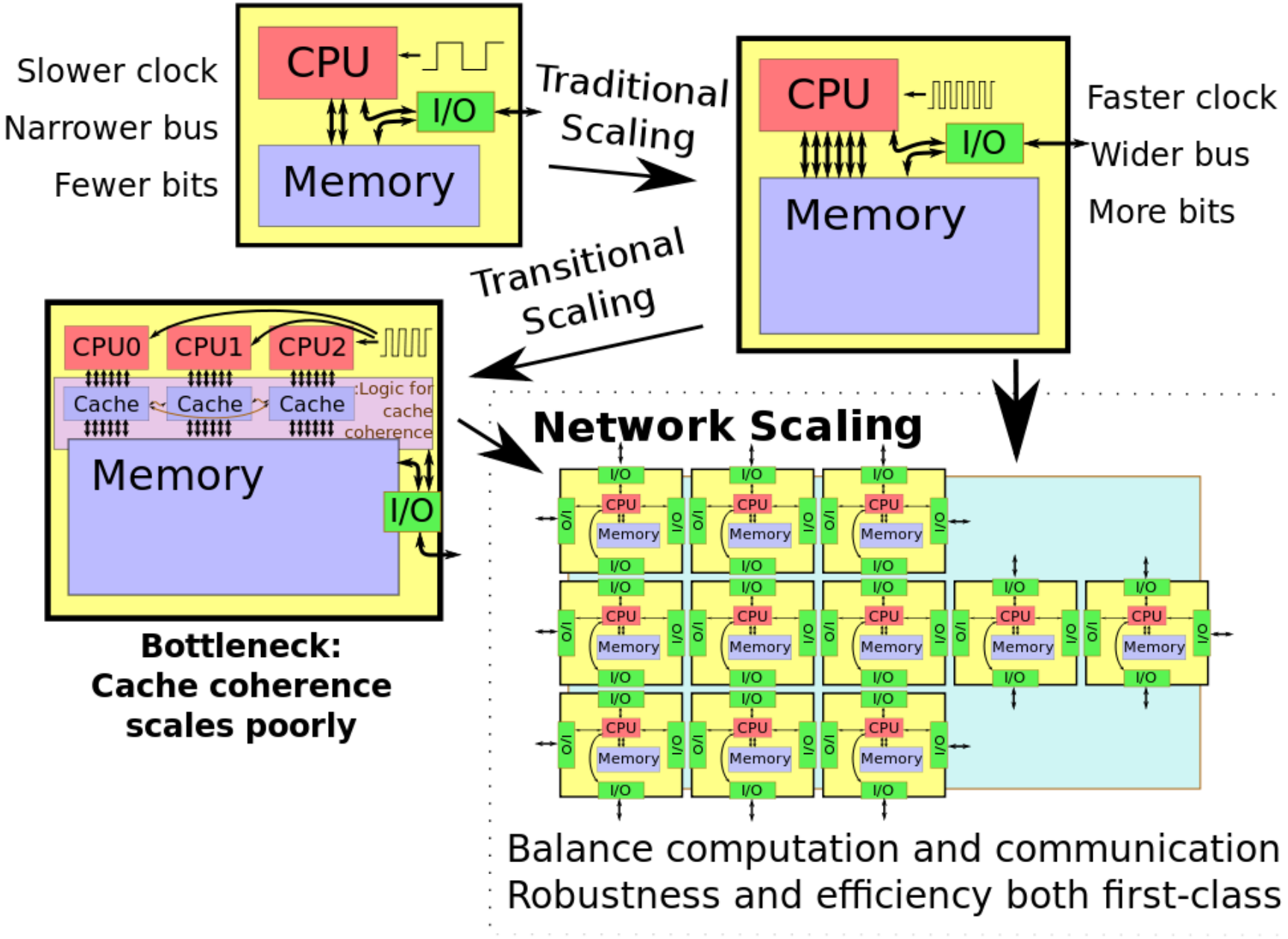
Recap

- Why did we want robustness again?
- Are we willing to pay for it?
- Forces converging on: Scalability

Plan

- Individual vs network scaling
- Computer architectures beyond the CPU
- Spatial computing
- Real computer architecture
- Example: The Movable Feast Machine

Transitions in computer architecture



Beyond the CPU

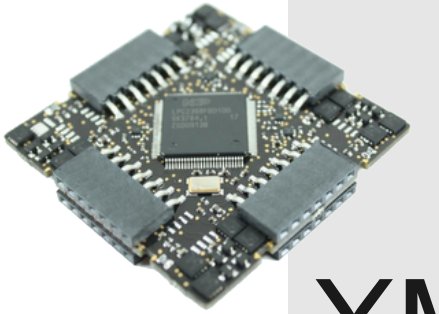
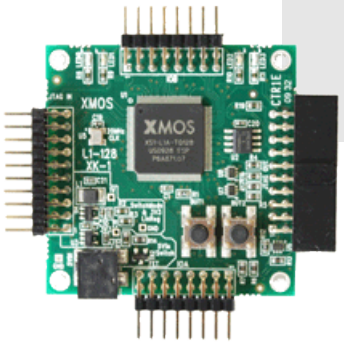
- MIMD, multicore
- SIMD, vectorizing
- Dataflow, GPU
- High performance computing
- FPGAs
- Spatial computing, cellular automata

Beyond the CPU: Limits

- MIMD, multicore: Cache coherence
- SIMD, vectorizing: Idle cells
- Dataflow, GPU: Programmability?
- High performance computing: Reliability at the exaflop barrier
- FPGAs: Pinout/bandwidth off-package
- Spatial computing, cellular automata

Renegotiation: Dare to think big

- A *finite machine* is a circuit
  - All conventional computers are finite machines
  - The internet is a finite machine (modulo *anycasting*)
- An *infinite machine* is a **spatial tiling** of finite machines.
- Infinite machine metrics (h/w peak #'s, s/w delivered #'s):
  - *Peak Computational Density:*
    - PCD_{IM} *= peak MIPS per tile / milligrams mass per tile*
    - *Informally: 'one stross'* $\stackrel{def}{=}$ *1 MIPS/milligram (* PCD_{IM} *)*
      - *After Charles Stross, author of Accelerando*
  - *Peak Communications Velocity:*
    - PCV_{IM} *= tile spacing / intertile hosted bit latency*
  - *Average Event Rate:*
    - AER_{IM} *= events per second per tile / sites per tile*

Tile	Estimated PCD_I
 HP65	10 nanostross
 IXM-1	4.4 millistross
 XMOS XK1	26 millistross

Indefinite scalability

- Thought experiment: Design a computing system that could be built arbitrarily large. As big as buildings. As big as cities. As big as solar systems.
  - What kind of architecture could work?
  - Asynchronous
  - Locally-connected
  - Robust

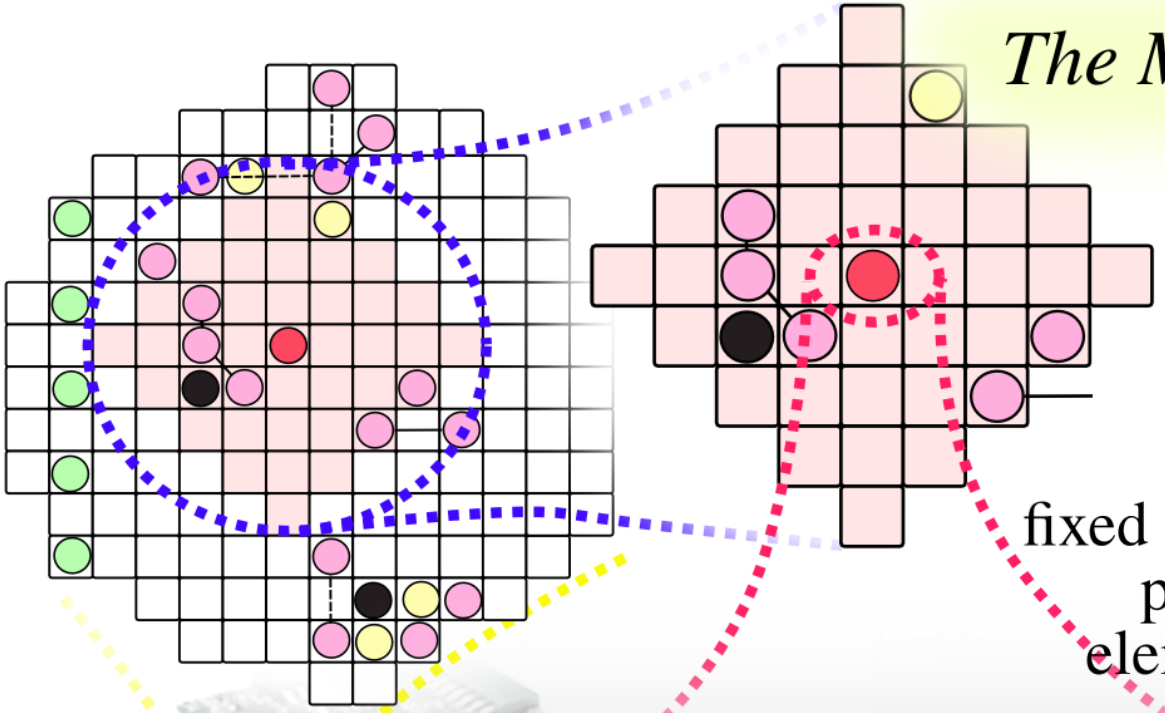
Indefinitely scalable architectures

- How about cellular automata?

Principles of indefinite scalability

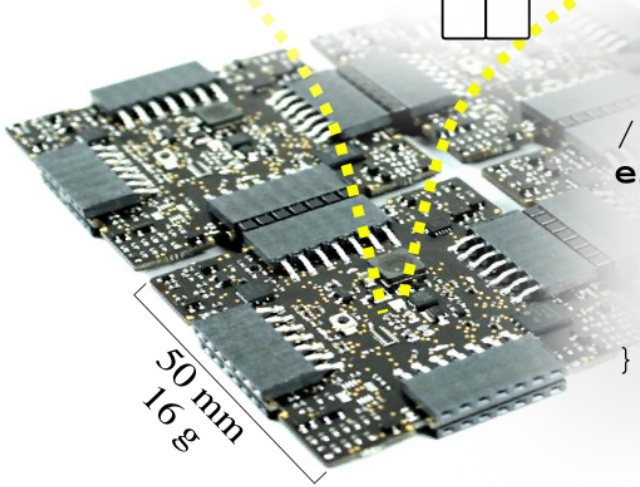
- No infinities: Specify a process for building.
  - Machine must operate during construction
- Stay inside the light cone
  - Local communications only. E.g., no "small world networks" (or any log diameter).
- Relative spatial addressing
  - No unique ids. No global coordinates.
- Robustness first
  - No "privileged points" in space or time.

Software engineering meets artificial chemistry



The Movable Feast Machine

spatial computing
relative addressing
local connectivity
indefinitely scalable
fixed size atoms/sites for mobility
parallel asynchronous update
element-oriented programming

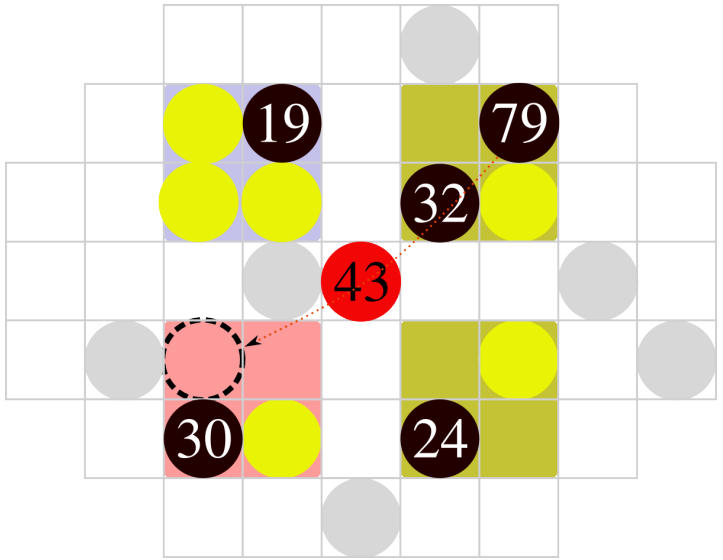



```
/* DReg: Dynamic Regulator.  */
element DReg() = 0xdba {
  if n:anyAt(1), n is Empty, odds(1,1000) then n = DReg;
  if n is Empty, odds(1,200) then n = Res;
  if n is DReg, odds(1,10) then n = Empty; // limit DRegs
  if odds(1,100) then n = Empty;
}
```

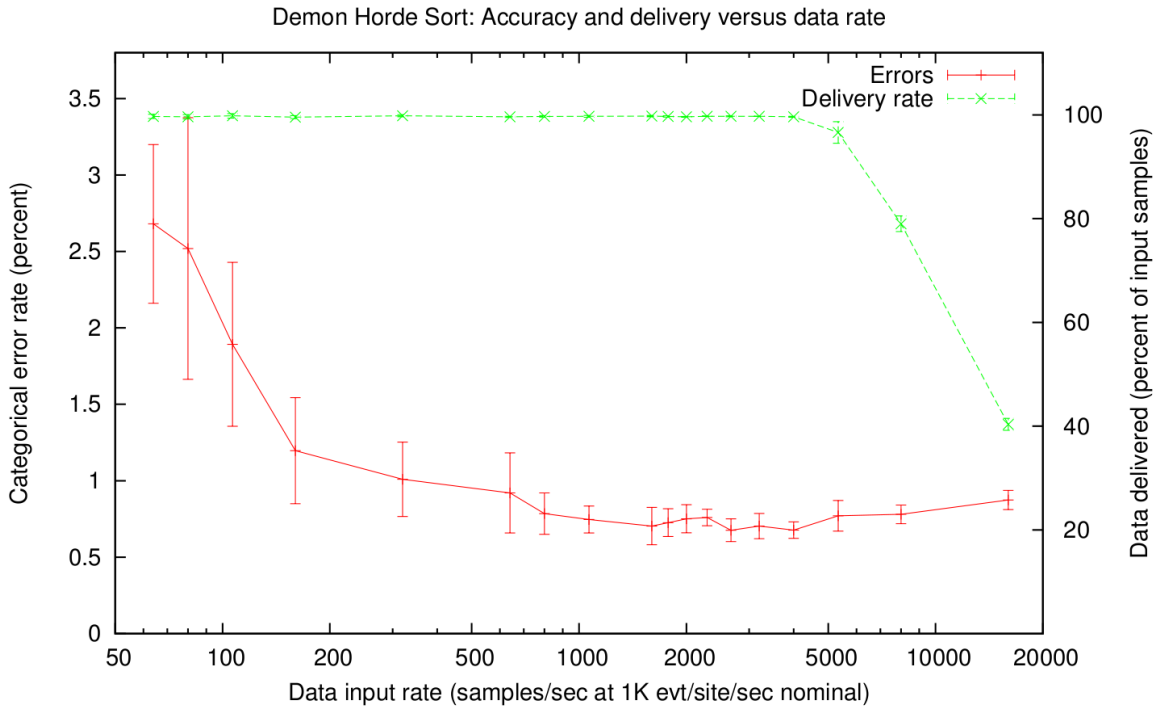
MFM example: DReg Homeostasis

- Two types of particles: **Dreg** and **Stem**
- **Stem** simply diffuses (in this example)
- **Dreg** examines a random neighbor and:
  - If empty, then
    - if odds(1,1000), create a **Dreg** there
    - Else if odds(1,200), create a **Stem** there
  - Else if a **Dreg**, then if odds(1,10), set empty
  - Else if odds(1,100), set that location empty
  - After all that, diffuse
- Start with 1 **Dreg.** What will happen?

Demon Horde Sorting: Robust Computation Example

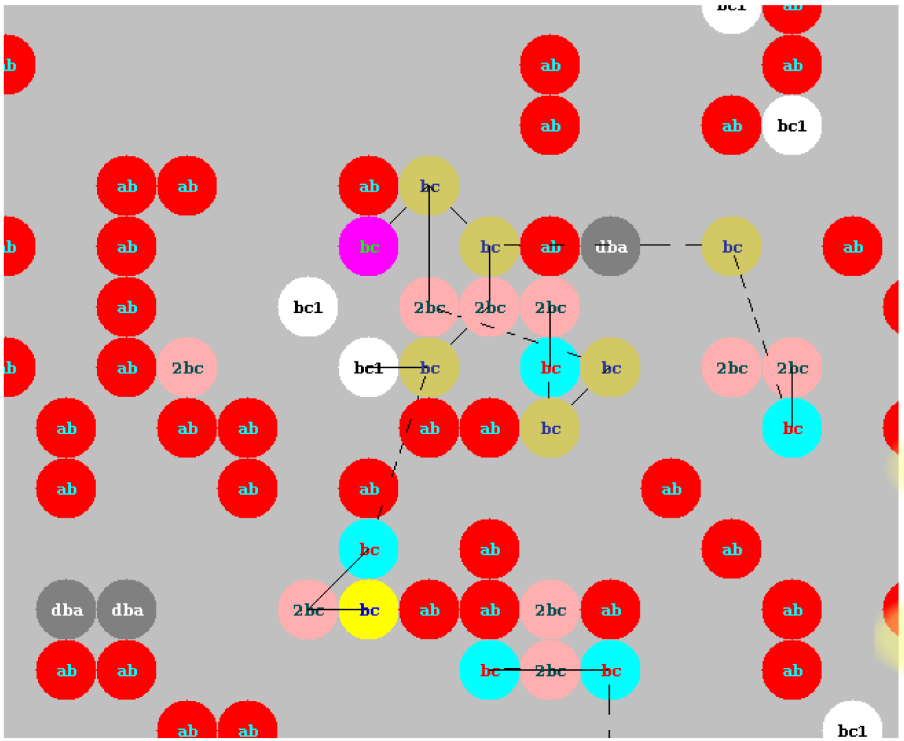


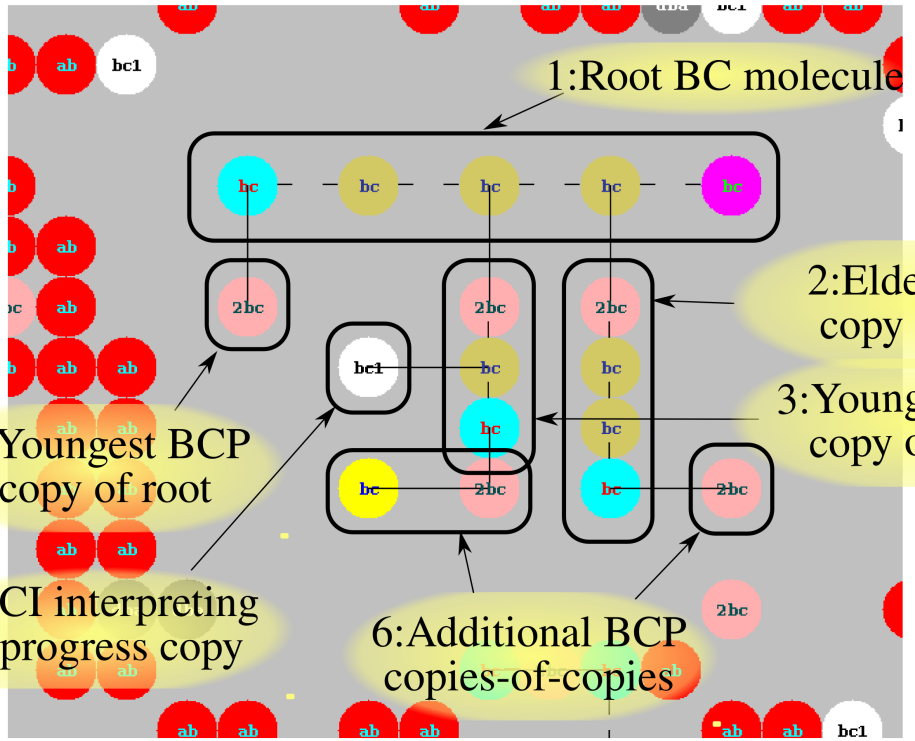
- Task: Flow sort data stream from 'input grid' to 'output grid'
- 'Maxwell's Demon' sorting elements built and stabilized by BC+DReg
- Surprise: Sorting quality vs data rate..



Base chain polymerase

- 'Base chain': [LCR]+prev&next, dock+16 bit payload
- 'Base chain polymerase': Copies a BC sequence
- 'Base chain interpreter': Executes data as byte codes against onboard registers and neighboring atoms.





Robustness principles in the movable feast machine

- Modularity
- Multiple pathways
- Redundancy
- Feedback control
- Sloppiness
- Spatial compart-mentalization
- Canalization & niche construction
- Purging
- Conflict mechanism
- Error-correction & repair
- Distributed processing

Conclusions

- The future of computing requires robustness
- Robustness and efficiency are mortal enemies
- Indefinite scalability escapes the Turing tarpit
- Robustness is everybody's job
- Software engineering meets artificial chemistry