Quantum Random Active Element Machine Computation

Michael S. Fiske
Unconventional Computation & Natural Computation
Universita degli Studi di Milano, Italy
July 1-5, 2013

mf@aemea.org

Biological Motivation: Active Element Machine

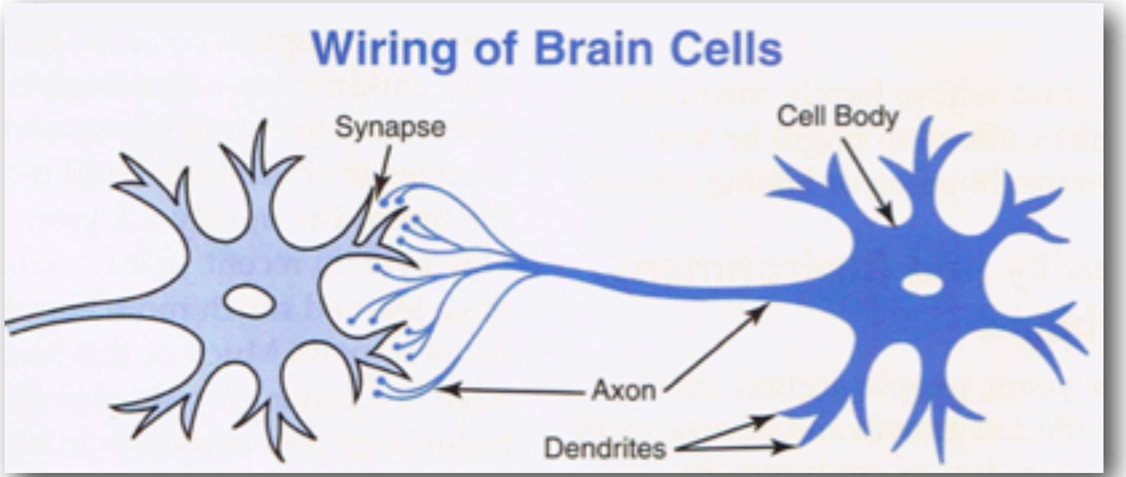
Dendritic Integration

Capture useful computational properties of dendrites. (Wilfrid Rall)

Step functions can approximate any measurable function.

Parallel. The machine uses time.

Synapses. What is being computed can change over time.



Design Criteria for the Active Element Machine

Simple Math. Easy to build in silicon and other hardware

No transfer function as in traditional neural networks.

$\boldsymbol{Z} = \{m + k\,\mathrm{dT}$: m, k are integers and dT is a fixed infinitesimal$\}$.

Math operators $+$, $>$ and time on $\boldsymbol{Z}$ (extended integers).

Machine and Programming Language

Implicitly programmable: evolution and machine learning.

Explicitly programmable: a person can write a program.

Five commands in the programming language.

Time is explicitly specified in the commands.

Machine architecture should be able to change as it is executing.

Useful Properties of the Active Element Machine (AEM)

Parallel Computation & Algorithms.
All active elements compute simultaneously.
No “single instruction” at a time bottleneck.

Avoiding Race Conditions.
Time in the commands helps with coordination. **dT**

The machine can change its rules while executing.
Meta command and time.

The meta command can increase program complexity over time.
Adaptivity and repair can be designed as part of the machine.

AEM computation

Active Elements and Connections.

Active elements compute simultaneously.

Connections connect Elements.

Connections determine the messages (pulses) that are sent between elements.

Elements fire and send pulses along Connections.

An element **E** fires at time $\boldsymbol{s}$ if the sum of **E**'s input pulses is greater than **E**'s threshold $\boldsymbol{\theta}$ and **E**'s refractory period $\boldsymbol{r}$ has expired i.e. $\boldsymbol{s \geq r + e}$ where $\boldsymbol{e}$ is **E**'s most recent firing time.

AEM computation – Outgoing Pulses

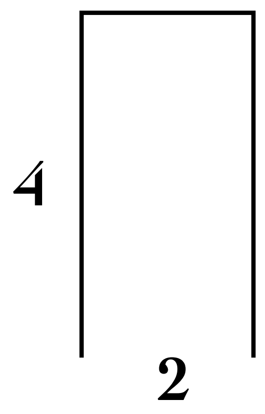
(Connection (Time 4) (From E) (To Y) (Amp -2) (Width 3) (Delay 5)

(Connection (Time 4) (From E) (To Z) (Amp 4) (Width 2) (Delay 3)

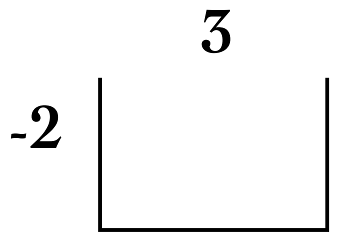
If element **E** fires at time 4, then

A pulse of time width 3 and amplitude -2 (height -2) arrives at element **Y** at time 9.

A pulse of time width 2 and amplitude (height 4) arrives at element **Z** at time 7.



is sent to **Z**



is sent to **Y**

Element, Connection & Fire Commands

(Element (Time 0) (Name E) (Threshold 9) (Refractory 4) (Last 0))

(Connection (Time -1)(From A) (To E) (Amp 5) (Width 4+2dT) (Delay 3−dT))

Input Active elements: (Fire (Time 0) (Name A))

Keyword dT

dT is an infinitesimal amount of time. **dT** > 0 and **dT** is less than every positive rational. If $m < n$, then $m\mathbf{dT} < n\mathbf{dT}$.

dT helps with concurrency. $3-2\mathbf{dT} < 3-\mathbf{dT} < 3 < 3+\mathbf{dT} < 3+2\mathbf{dT}$
For every integer $m > 0$, $2 < 3-m\mathbf{dT}$ and $3+m\mathbf{dT} < 4$.

$st(k + m\mathbf{dT}) = k$ is the standard part of extended integer $k + m\mathbf{dT}$.

AEM Programming Language

Keyword clock

clock evaluates to an integer which is the standard part of the current time of the active element machine clock.

Meta command — enables the machine to self-modify

(Meta (Name E) (Window b e) **(C (Args clock)))**

If active element **E** fires at time ***s*** in window [***b***, ***e***] where ***b*** ≤ ***s*** ≤ ***e*** then command **(C clock)** executes at time ***s.***

If there is no window specified, then if **E** fires at any time ***s***, then **(C s)** executes at time ***s***. (No restrictions on time ***s*** when **E** fires.)

Cybersecurity Motivation

I. **Preserving the purpose of the machine is more fundamental than the confidentiality (encryption) and integrity (authentication) of the data.**

Why? The purpose of the machine can be hijacked, which can compromise the confidentiality and integrity of the data.

II. **Malware is the nemesis of cybersecurity.**

No digital computer program can detect all malware.[1] Malware authors use NP problems to encrypt and hide the malware.[2]

1. Fred Cohen. Computer Viruses and Experiments. *Computers & Security*. 22-35 (1987)
2. Eric Filiol. Malicious Cryptology and Mathematics. *Intech*. 23-50 (2012)

Turing Machine Model

The conventional computing model.

Finite set of states $\boldsymbol{Q} = \{\boldsymbol{q}_1, . . . , \boldsymbol{q}_\mathrm{n}\}$. Finite alphabet $\boldsymbol{A} = \{\boldsymbol{a}_1, . . . , \boldsymbol{a}_\mathrm{m}\}$.

Tape $\boldsymbol{T} : \boldsymbol{Z} \rightarrow \boldsymbol{A}$ with an alphabet symbol on each tape square.

The Turing program $\eta : \boldsymbol{Q} \times \boldsymbol{A} \rightarrow (\boldsymbol{Q} \cup \{\boldsymbol{h}\}) \times \boldsymbol{A} \times \{-1, +1\}$ is a finite set of rules that stays fixed for the whole computation.

η **doesn't change as the program executes. No reference to time.**

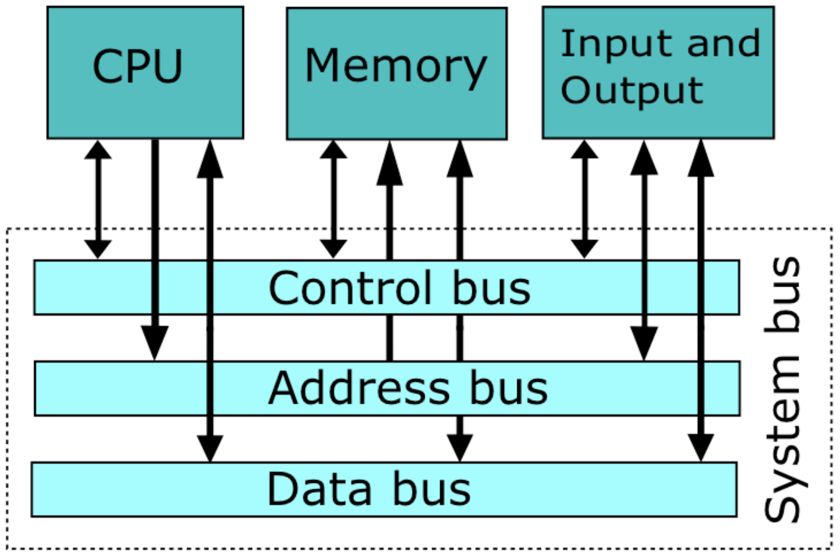
$\eta(\boldsymbol{q}, \boldsymbol{a}) = (\boldsymbol{r}, \boldsymbol{b}, -1)$ or $\eta(\boldsymbol{q}, \boldsymbol{a}) = (\boldsymbol{r}, \boldsymbol{b}, +1)$ is a computational step. One rule is selected, based on state $\boldsymbol{q}$ and symbol $\boldsymbol{a}$. A step replaces $\boldsymbol{a}$ with symbol $\boldsymbol{b}$ on the tape, moves to a new state $\boldsymbol{r}$ and the tape head moves left (-1) or right (+1).

Computational steps are executed sequentially.

"One instruction at a time" deep vulnerability

Current processors — based on a von Neumann architecture — execute one machine instruction at a time, which causes a deep vulnerability in processor architectures.

To initiate execution of malignant code, malware need only corrupt or transform a single machine instruction.



Hiding computation versus cryptography

What is cryptography?

Transformation of a message under the control of a secret key.
See talk aemea.org/Diffie (Whitfield Diffie).

Distinct from cryptography: Hiding the AEM computation is a transformation of computation, not a transformation of a message.

Diffie's definition doesn't address or explain how the cryptography transformation is securely executed.

Does "hiding computation" help address Adi Shamir's observation?
"Cryptography is typically bypassed, not penetrated."

Using the ***R*** rules of quantum mechanics

Von Neumann's axioms distinguished the ***U*** (unitary evolution) and ***R*** **(**reduction) rules of quantum mechanics.

"Now, quantum computing so far (in the work of Feynman, Shor, Deutsch, etc.) is based on the ***U*** process and is computable. It has not made serious use of the ***R*** process: the unpredictable element that comes in with reduction, measurement, or collapse of the wave function."

– Andrew Hodges in *What would have Turing done after 1954?*

Godel numbering is a special type of interpretation

“The Postulates of Mathematics Were Not on the Stone Tablets that Moses Brought Down from Mt. Sinai …

It is not that the postulation approach is wrong, only that its arbitrariness should be clearly recognized, and we should be prepared to change postulates when the need becomes apparent.”

– Richard Hamming. The Unreasonable Effectiveness of Mathematics. *American Mathematical Monthly*. Volume 87, Number 2 (1980).

Changing the meaning of the symbols & computation

Today:	'H' has ASCII binary code 0100 1000.
Tomorrow:	'H' has ASCII binary code 0100 1000.

"What if '6' were '9' and '9' were '6' ?"

– Haley joking about **6** and **9** being upside down.

Based on quantum randomness, the meaning of the symbols and the computation changes. Half the time 1 means 0 and 0 means 1.

Example: Change how XOR is computed with the AEM

$0 = 0 \oplus 0 = 1 \oplus 1$ and $1 = 1 \oplus 0 = 0 \oplus 1$

Level Sets. $\oplus^{-1}\{1\} = \{(1, 0), (0, 1)\}$ and $\oplus^{-1}\{0\} = \{(0, 0), (1, 1)\}$.

$(2, -2, 1, \mathbf{X_1})$

$(-2, 2, 1, \mathbf{X_2})$

$$\begin{tikzcd}
\mathbf{A} \arrow[d, "2"'] \arrow[rr, dotted, "2"] & & \mathbf{B} \arrow[d, "-2"] \arrow[ll, dotted, "-2"'] \\
\mathbf{X_1} \arrow[dr] & & \mathbf{X_2} \arrow[dl] \\
& \mathbf{P} &
\end{tikzcd}$$

$\mathbf{A \oplus B = P}$

Quantum random bit $\mathbf{R_1}$ and the meta command can dynamically flip the level set of active element $\mathbf{X_1}$.

$\mathbf{R_1} = 1$: $\{(1, 0)\} / \{(0, 0), (0, 1), (1, 1)\}$. $\mathbf{R_1} = 0$: $\{(0, 0), (0, 1), (1, 1)\} / \{(1, 0)\}$.

Similarly, quantum bit $\mathbf{R_2}$ helps select the level set of element $\mathbf{X_2}$.

$\mathbf{R_2} = 1$: $\{(0, 1)\} / \{(0, 0), (1, 0), (1, 1)\}$. $\mathbf{R_2} = 0$: $\{(0, 0), (1, 0), (1, 1)\} / \{(0, 1)\}$.

Some Results

I. For an external observer, it is algorithmically impossible to reverse engineer the computation: aemea.org/Turing100

II. If one accepts the assumptions in [3], then the construction theoretically crosses the Turing barrier; the active element firing pattern performing this hidden computation is a bi-immune sequence.

Bi-immune means this random sequence of 0 and 1's contains no infinite subsequence that is Turing computable.

III. Let $A \oplus B = (A - B) \cup (B - A)$. If set R is c.e. and A is bi-immune, created by a quantum random number generator, then a quantum random AEM can compute bi-immune $A \oplus R$.

3. Abbott, Calude, Conder & Svozil. Strong Kochen-Specker theorem and Incomputability of Quantum Randomness. *Physical Review A* (2012).

Some Observations

IV. These methods pertain to what a machine is capable of computing rather than how fast. Publicly disclosed methods in cryptography typically depend upon the assumption $P \neq NP$.

V. Introduces the notion of non-Turing interpretations. Computability theory assumes interpretations (Godel coding) are Turing computable. – Hartley Rogers, Jr. Theory of Recursive Functions and Effective Computability (1987).

VI. In proofs of the undecidability of the halting problem using Cantor's diagonalization method, a contradiction is no longer reached with a machine that performs non-Turing interpretations.

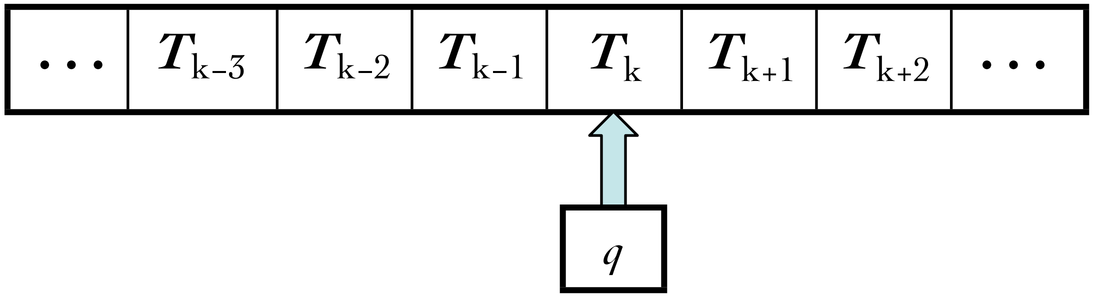
Perspective: Turing Machine is an autonomous dynamical system

Define mapping φ that creates a one-to-one correspondence from the Turing program η to a finite set of two dimensional affine functions in the x-y plane.

Set base B = $|\boldsymbol{A}| + |\boldsymbol{Q}| + 1$. h is the halting state.

Define value function $\nu: \{h\} \cup \boldsymbol{Q} \cup \boldsymbol{A} \rightarrow \boldsymbol{N}$ where $\boldsymbol{N}$ is the natural numbers

and $\nu(h) = 0$, $\nu(a_i) = i$, $\nu(q_i) = i + |\boldsymbol{A}|$ and $\nu(q_{|\boldsymbol{Q}|}) = |\boldsymbol{Q}| + |\boldsymbol{A}|$.



φ maps computational step $\eta(q, \boldsymbol{T}_k) = (r, \alpha, +1)$ to the affine function

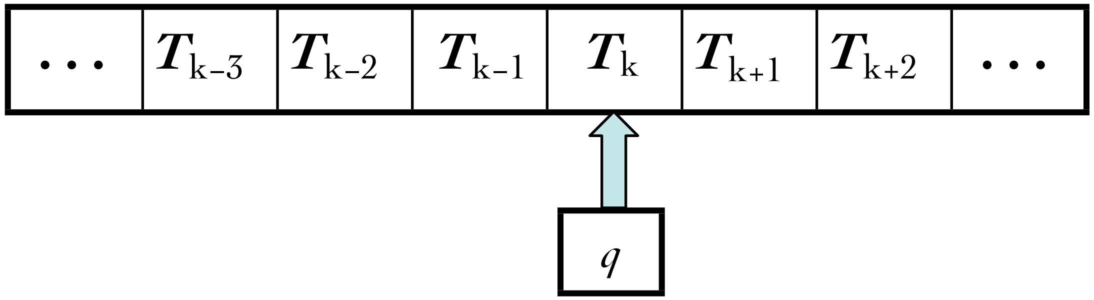
$f(x, y) = (\mathrm{B}x + m,\ \mathrm{B}^{-1}y + n)$ where $m = -\mathrm{B}^2\nu(\boldsymbol{T}_k)$ and $n = \mathrm{B}\nu(r) + \nu(\alpha) - \nu(q)$

Turing Machine is mapped to a finite set of affine functions

φ maps computational step $\eta(q, \boldsymbol{T}_k) = (r, \alpha, -1)$ to the affine function

$f(x, y) = (B^{-1}x + m,\ By + n)$ where

$m = B\nu(\boldsymbol{T}_{k-1}) + \nu(\alpha) - \nu(\boldsymbol{T}_k)$ and $n = B\nu(r) - B^2\nu(q) - B\nu(\boldsymbol{T}_{k-1})$.



A machine configuration $(q, k, \boldsymbol{T}) \in \boldsymbol{Q} \times \boldsymbol{Z} \times \boldsymbol{A}^{\boldsymbol{Z}}$ is mapped to a point in the x-y plane as $\varphi(q, k, \boldsymbol{T}) = (x(q, k, \boldsymbol{T}),\ y(q, k, \boldsymbol{T}))$ where the

x-coordinate function is $x(q, k, \boldsymbol{T}) = \boldsymbol{T}_k\, \boldsymbol{T}_{k+1} \,.\, \boldsymbol{T}_{k+2}\, \boldsymbol{T}_{k+3}\, \boldsymbol{T}_{k+4} \ldots$ in base B and

the y-coordinate function is $y(q, k, \boldsymbol{T}) = q\, \boldsymbol{T}_{k-1} \,.\, \boldsymbol{T}_{k-2}\, \boldsymbol{T}_{k-3}\, \boldsymbol{T}_{k-4} \ldots$ in base B.

Ex: Turing Machine as a discrete, autonomous dynamical system

Alphabet $\boldsymbol{A} = \{\#, \boldsymbol{a}, \boldsymbol{b}\}$

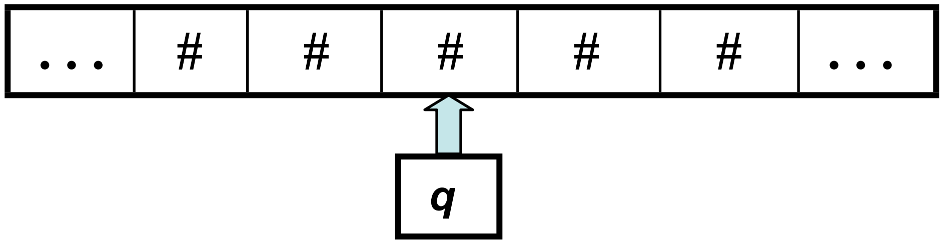
States $\boldsymbol{Q} = \{\boldsymbol{q}, \boldsymbol{r}, \boldsymbol{s}\}$

Program

η	#	*a*	*b*
q	(***r***, ***a***, +1)	(***q***, ***a***, -1)	(***q***, ***b***, -1)
r	(***q***, ***b***, -1)	(***r***, ***a***, +1)	(***r***, ***b***, +1)
s	(***h***, #,+1)	(***h***, ***a***,+1)	(***h***, ***b***,+1)

Base B = $|\boldsymbol{A}| + |\boldsymbol{Q}| + 1 = 7$.

$\nu(\boldsymbol{h}) = 0$, $\nu(\#) = 1$, $\nu(\boldsymbol{a}) = 2$, $\nu(\boldsymbol{b}) = 3$, $\nu(\boldsymbol{q}) = 4$, $\nu(\boldsymbol{r}) = 5$, $\nu(\boldsymbol{s}) = 6$.



Initial machine configuration

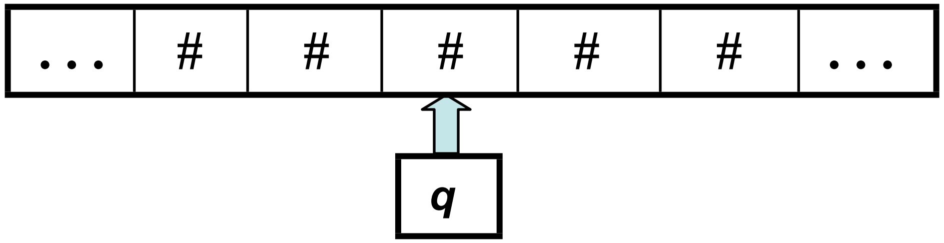
The initial point $\mathbf{p} = (p_x, p_y)$ where

$p_x = \mathrm{B}\,\nu(\#) + \nu(\#) / (1 - 1/7) = 7 + 7/6 = 8\ 1/6$

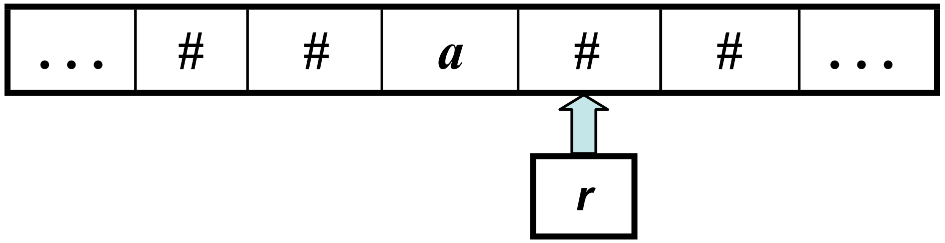
$p_y = \mathrm{B}\,\nu(\boldsymbol{q}) + \nu(\#) / (1 - 1/7) = 28 + 7/6 = 29\ 1/6$

Ex: Turing Machine as a discrete, autonomous dynamical system II

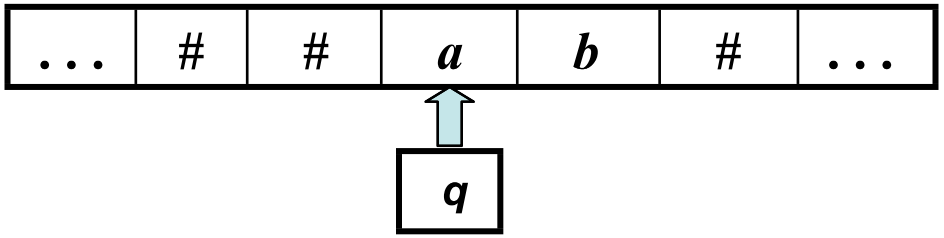
η	#	*a*	*b*
q	(*r*, *a*, +1)	(*q*, *a*, -1)	(*q*, *b*, -1)
r	(*q*, *b*, -1)	(*r*, *a*, +1)	(*r*, *b*, +1)
s	(*h*, #,+1)	(*h*, *a*,+1)	(*h*, *b*,+1)



Apply affine function $\mathbf{f_1(x,y) = (7x - 49, \frac{1}{7}y + 33)}$ to $\mathbf{p = (8\frac{1}{6}, 29\frac{1}{6})}$



Apply affine function $\mathbf{f_{15}(x,y) = (\frac{1}{7}x + 16, 7y - 231)}$ to $\mathbf{p = (8\frac{1}{6}, 37\frac{1}{6})}$



Halting Problem: Does the orbit of point $\mathbf{p}$ —w.r.t. this discrete autonomous, dynamical system — remain in the attractor?

If machine configuration $(q, k, \boldsymbol{T})$ halts after n computational steps, then the orbit of $\mathbf{p} = \varphi(q, k, \boldsymbol{T})$ exits one of the unit squares on the nth iteration.

If machine configuration $(r, j, \boldsymbol{S})$ is immortal, then the orbit of $\varphi(r, j, \boldsymbol{S})$ remains in these unit squares (the attractor) forever.

The proof of the undecidability of the halting problem depends on the universality (existence of universal Turing machines) of this class of discrete, autonomous dynamical systems.

Theory of Quantum Randomness

Strong Kochen-Specker theorem and Incomputability of Quantum Randomness.

Measurement assumption.
Measurement yields a physically meaningful and unique result.

Non-contextuality assumption. The set of observables O is non-contextual.

Elements of physical reality (e.p.r.) assumption. If there exists a Turing computable function $f : N \times O \times C \rightarrow \{0, 1\}$ such that $\forall\, k,\; f(k, o_k, C_k) = x_k$, then there is a definite value associated with o_k at each step.

Eigenstate assumption. Let $|\psi>$ be a normalized quantum state and v a faithful assignment function. Then $v(P_\psi, C) = 1$ and $v(P_\varphi, C) = 0$ for any context C with P_φ and P_ψ lying in C.

Conclusion. Assume the measurement, non-contextuality, eigenstate and e.p.r. assumptions. Then there exists a quantum random number generator that generates a bi-immune binary sequence. http://arxiv.org/pdf/1207.2029v3.pdf

Two Hardware Assumptions & Practical Applications

Assumptions that guide a hardware implementation.

1. That a physical device can be built that protects the secrecy of the random bits and the underlying dynamic connections.

2. That the firing of the active elements is the physical part of the system that leaks electromagnetic radiation.

Practical Applications.

A. A superior method of stopping differential power analysis attacks. No branching instructions and timing delays to exploit.

B. Secures intellectual property and algorithms.

C. Hinders reverse engineering and malware exploits.

Quantum Randomness can be used today

Quantum Random Number Generators (QRNG) exist and work.

Since 2006, the following tests were performed on a noteworthy QRNG: NIST STS, pLab, Geneva test, fast version of the Kolmogorov complexity test, minimum entropy and the Borel test.

0 0 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 1
1 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0
0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 1 1 0 0 1
1 1 1 0 1 0 0 1 0 0 1 1 0 1 1 1 0 0 1 0
0 1 0 0 1 0 1 0 1 1 0 0 1 1 1 0 0 1 0 1 . . .

After 6 years of testing, no flaws have been found in this QRNG.