Quantum Random, Self-Modifiable Computation

Michael Stephen Fiske

Aemea Institute

Logic Colloquium
13 August 2019

Table of contents

Standard Model of Computation

The Turing Machine is the standard model.

Standard Model of Computation

The Turing Machine is the standard model.

The number of states stays fixed during the computation.

Standard Model of Computation

The Turing Machine is the standard model.

The number of states stays fixed during the computation.

The instructions stay fixed during the computation.

Standard Model of Computation

The Turing Machine is the standard model.

The number of states stays fixed during the computation.

The instructions stay fixed during the computation.

No randomness.

Ex-Machine Intuition

Ex-Machine is derived from the latin *extra machinam*.

Ex-Machine Intuition

Ex-Machine is derived from the latin *extra machinam*.

Add self-modification and randomness to the Turing Machine.

Ex-Machine Intuition

Ex-Machine is derived from the latin *extra machinam*.

Add self-modification and randomness to the Turing Machine.

Ex-machine instructions can evolve with self-modification.

Ex-Machine Intuition

Ex-Machine is derived from the latin *extra machinam*.

Add self-modification and randomness to the Turing Machine.

Ex-machine instructions can evolve with self-modification.

With randomness and self-modification, two instances of the same initial ex-machine can execute and evolve to two distinct ex-machines. (Non-autonomous dynamical system.)

Ex-Machine Intuition

Ex-Machine is derived from the latin *extra machinam*.

Add self-modification and randomness to the Turing Machine.

Ex-machine instructions can evolve with self-modification.

With randomness and self-modification, two instances of the same initial ex-machine can execute and evolve to two distinct ex-machines. (Non-autonomous dynamical system.)

An ex-machine can make a computational mistake on a first instance of a problem and subsequently repair its program before executing on a second instance of the same problem.

Preliminaries

From $\{a\}^*$, define the set of languages $\mathfrak{L} = \bigcup_{L \subset \{a\}^*} \{L\}$.

For $f : \mathbb{N} \to \{0, 1\}$, define language $L_f = \{a^n : f(n) = 1\}$.

$$\mathfrak{L} = \bigcup_{f \in \{0,1\}^{\mathbb{N}}} \{L_f\}$$

Let μ be the Lebesgue measure on $\{0, 1\}^{\mathbb{N}}$. $\mu\big(\{0, 1\}^{\mathbb{N}}\big) = 1$.

μ induces Lebesgue measure ν on $\mathfrak{L}$ via $f \leftrightarrow L_f$. $\nu(\mathfrak{L}) = 1$.

Set Alphabet $A = \{\#, 0, 1, N, Y, a\}$. $\#$ is the blank symbol.
Set States $Q = \{0, h, n, y, t, v, w, x, 8\}$ with halting state h

$\mathfrak{Q}(x)$ Specification: 15 Initial Instructions

$(0, \#, 8, \#, 1)$
$(8, \#, x, \#, 0)$
$(y, \#, h, Y, 0)$
$(n, \#, h, N, 0)$

$(x, \#, x, 0)$
$(x, a, t, 0)$

$(x, 0, v, \#, 0, (|Q| - 1, \#, n, \#, 1))$
$(x, 1, w, \#, 0, (|Q| - 1, \#, y, \#, 1))$
$(t, 0, w, a, 0, (|Q| - 1, \#, n, \#, 1))$
$(t, 1, w, a, 0, (|Q| - 1, \#, y, \#, 1))$

$(v, \#, n, \#, 1, (|Q| - 1, a, |Q|, a, 1))$
$(w, \#, y, \#, 1, (|Q| - 1, a, |Q|, a, 1))$
$(w, a, |Q|, a, 1, (|Q| - 1, a, |Q|, a, 1))$

$(|Q| - 1, a, x, a, 0)$
$(|Q| - 1, \#, x, \#, 0)$

Main Results

$\mathfrak{Q}(x)$ computes Turing incomputable languages in $\mathfrak{L}$ with probability (Lebesgue measure) 1.

After a finite number of computational steps, $\mathfrak{Q}(x)$ uses a finite amount of computing resources.

Consider an enumeration $\mathcal{E}_a(i) = (\mathfrak{M}_i, T_i)$ of all Turing machines $\mathfrak{M}_i$ and initial tapes T_i, each containing a finite number of non-blank symbols.

There exists an evolutionary path $\mathfrak{Q}(x) \to \mathfrak{Q}_1 \to \mathfrak{Q}_2 \to \dots \to \mathfrak{Q}_m$, so at the mth stage $\mathfrak{Q}_m$ correctly determines for $0 \le i \le m$ whether $\mathfrak{M}_i$'s execution on tape T_i halts.

Key Observations

$p = \frac{1}{2}$. Our random instructions use no hidden tricks, e.g. p is Turing incomputable. [1]

In practice, $\mathfrak{Q}(x)$ will not find this evolutionary halting path, even though it is possible. (Impossible for a Turing machine.)

$\mathfrak{Q}(x)$'s dynamical behavior circumvents the contradiction in an information-theoretic proof of Turing's halting problem.[2]

The circumvention occurs because $\mathfrak{Q}(x)$'s meta instructions increase the number of states and instructions in $\mathfrak{Q}(x)$.

[1] K. de Leeuw, E.F. Moore, C. Shannon, N. Shapiro. Computability by Probabilistic Machines. Princeton University Press. 1956.

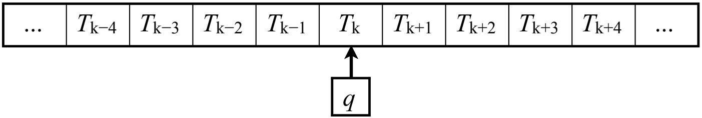
[2] C. Calude. Information and Randomness. Springer. 1994. pp. 184-185

Standard Instructions $\mathcal{S}$: (q, T_k, r, b, y)

Standard Instructions are Turing machine instructions.

$Q = \{0, 1, \ldots, n-1\} \subset \mathbb{N}$ is the set of states.

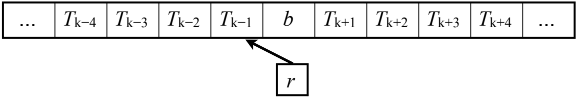
Alphabet $A = \{0, 1, \#\} \cup \{a_1, \ldots a_m\}$. $\#$ is the blank symbol.



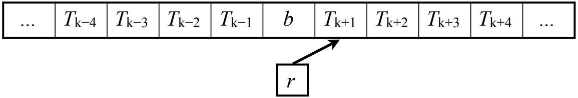
Standard Instructions $\mathcal{S}$

Reading T_k on the tape in state q, write b on tape and move to state r. Move tape head $y \in \{-1, 0, 1\}$.

$(q, T_k, r, b, -1)$



$(q, T_k, r, b, +1)$



Random Instructions $\mathcal{R}$: (q, a, r, y)

When scanning alphabet symbol a and lying in state q, random instruction (q, a, r, y) executes as follows.

Measure a quantum event that returns a random $b \in \{0, 1\}$.

On the tape, replace alphabet symbol a with random bit b. (Alphabet A always contains symbols $\{0, 1\}$.)

The *ex*-machine state moves to state r.

The *ex*-machine moves its tape head left if $y = -1$, right if $y = +1$, or does not move if $y = 0$.

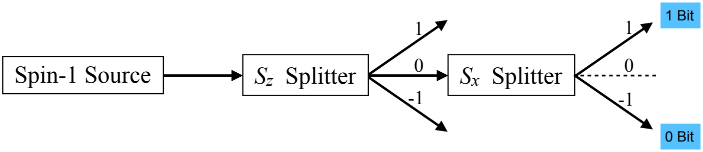
Quantum Random Axioms

Unbiased Trials: Consider bit sequence $(x_1\ x_2\ x_3\ \ldots)$ in the infinite product space $\{0,1\}^{\mathbb{N}}$. A single outcome x_i generated by quantum randomness is unbiased. The probabilities satisfy $P(x_i = 1) = P(x_i = 0) = \frac{1}{2}$.

Stochastic Independence: History has no effect on the next quantum random measurement. Each outcome x_i is independent of the history. Expressed as conditional probabilities, $P(x_i = a \mid x_1 = b_1, \ldots, x_{i-1} = b_{i-1}) = \frac{1}{2}$ for $a = 0$, $a = 1$ and for each $(b_1, b_2, \ldots, b_{i-1}) \in \{0,1\}^{i-1}$.

A Theory for a Quantum Random Number Generator

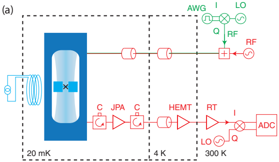
Protocol for a QRNG Based on Value Indefiniteness[3]

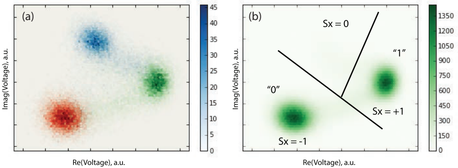


[3] Abbott, Calude, Conder, Svozil. Strong Kochen-Specker theorem and incomputability of quantum randomness. Physical Review A 86, 062109. 2012.

A Physical Realization of a QRNG

Measurement Setup and S_x Data [4]





[4] Kulikov et al. Realization of a QRNG certified with the Kochen-Specker Theorem. 2017. arXiv:1709.03687v1

Meta Instructions $\mathcal{M}$

Meta instructions can add new states.

Meta instructions can add new instructions

Meta instructions can replace instructions.

Meta instructions $\mathcal{M}$ are a subset of $\{(q, a, r, \alpha, y, J) : q \in Q$ and $r \in Q \cup \{|Q|\}$ and $a, \alpha \in A$ and instruction $J \in \mathcal{S} \cup \mathcal{R}\}$.

J is a standard or random instruction

Execution of Meta Instruction (q, a, r, α, y, J)

$\mathcal{I} = \mathcal{S} \cup \mathcal{R} \cup \mathcal{M}$.

Quintuple (q, a, r, α, y) executes as a standard instruction with one caveat:

State q may be expressed as $|Q| - c_1$ and state r may be expressed as $|Q|$ or $|Q| - c_2$, where $0 < c_1, c_2 \leq |Q|$. When (q, a, r, α, y) is executed, if q is expressed as $|Q| - c_1$, the value of q is instantiated to the current value of $|Q|$ minus c_1.

If state r is expressed as $|Q|$ or $|Q| - c_2$, the value of r instantiates to the current value of $|Q|$ or $|Q| - c_2$, respectively.

Execution of Meta Instruction (q, a, r, α, y, J)

Unique state, scanning condition: for any two distinct instructions chosen from $\mathcal{I}$ at least one of the first two coordinates must differ.

Next, instruction J modifies $\mathcal{I}$, where instruction J has one of the two forms: $J = (q, a, r, \alpha, y)$ or $J = (q, a, r, y)$.

For both forms, if $\mathcal{I} \cup \{J\}$ still satisfies the unique state, scanning symbol condition, then $\mathcal{I}$ is updated to $\mathcal{I} \cup \{J\}$.

Otherwise, there is an instruction I in $\mathcal{I}$ whose first two coordinates q, a are equal to instruction J's first two coordinates. In this case, instruction J replaces instruction I in $\mathcal{I}$. That is, $\mathcal{I}$ is updated to $\mathcal{I} \cup \{J\} - \{I\}$

Example that Executes a Meta Instruction

Consider meta instruction $(q, a_1, |Q| - 1, \alpha_1, y_1, J)$, where $J = (|Q| - 1, a_2, |Q|, \alpha_2, y_2)$.

After standard instruction $(q, a_1, |Q| - 1, \alpha_1, y_1)$ executes, this meta instruction adds a new state $|Q|$ to the states Q and adds instruction J, instantiated with the current value of $|Q|$.

Set states $Q = \{0, 1, 2, 3, 4, 5, 6, 7\}$. Alphabet $A = \{\#, 0, 1\}$.

An initial configuration is shown below.

State	Tape
5	##11 01##

Example that Executes a Meta Instruction

Meta instruction $(5, 0, |Q| - 1, 1, 0, J)$ executes with values $q = 5$, $a_1 = 0$, $\alpha_1 = 1$, $y_1 = 0$, $a_2 = 1$, $\alpha_2 = \#$, and $y_2 = -1$.

Instruction $J = (|Q| - 1,\ 1, |Q|, \#, -1)$

Since $|Q| = 8$, instruction $(5, 0, 7, 1, 0)$ executes.
$J = (7, 1, 8, \#, -1)$ is added as a new standard instruction.

The instantiation of $|Q| = 8$ in J adds state 8. The states are updated to $Q = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$.

The new ex-machine configuration is shown below.

State	Tape
7	##11 11##

Example that Executes a Meta Instruction

State	Tape
7	$\#\#11\ \ 11\#\#$

Now, the ex-machine is scanning a 1 and lying in state 7, so the standard instruction $J = (7, 1, 8, \#, -1)$ executes.

Note J was just added to the instructions.

After J executes, the new configuration is shown below.

State	Tape
8	$\#\#1\ \ 1\#1\#\#$

Simple Meta Instructions

A simple meta instruction has syntax, where $0 < c_1, c_2 \leq |Q|$:

$$(q, a, |Q| - c_2, \alpha, y) \quad \text{or} \quad (q, a, |Q|, \alpha, y)$$

$$(|Q| - c_1, a, r, \alpha, y)$$

$$(|Q| - c_1, a, |Q| - c_2, \alpha, y) \quad \text{or} \quad (|Q| - c_1, a, |Q|, \alpha, y).$$

Expressions $|Q| - c_1$, $|Q| - c_2$ and $|Q|$ are instantiated to a state based on the current value of $|Q|$ when the meta instruction executes.

$\mathfrak{Q}(x)$ self-reflects with the symbols $|Q| - 1$ and $|Q|$.

Finite Initial Conditions

A *finitely bounded tape* means the tape has a finite number of non-blank symbols.

An ex-machine has *finite initial conditions* if the following 4 conditions are satisfied before the ex-machine starts executing.

1. The number of states $|Q|$ is finite.
2. The number of alphabet symbols $|A|$ is finite.
3. The number of machine instructions $|\mathcal{I}|$ is finite.
4. The tape is finitely bounded.

Evolving an Ex-machine

T_0, T_1, ... T_{i-1} are finitely bounded tapes. Ex-machine $\mathfrak{X}_0$ has finite initial conditions.

$\mathfrak{X}_0$ starts executing with tape T_0 and evolves to ex-machine $\mathfrak{X}_1$ with tape S_1.

Next, $\mathfrak{X}_1$ starts executing with tape T_1 and evolves to $\mathfrak{X}_2$ with tape S_2. This means that when ex-machine $\mathfrak{X}_1$ starts executing on tape T_1, its instructions are preserved after the halt with tape S_1.

The ex-machine evolution continues until $\mathfrak{X}_{i-1}$ starts executing with tape T_{i-1} and evolves to ex-machine $\mathfrak{X}_i$ with tape S_i.

Evolutionary Path, Ancestors and Descendants

One says that ex-machine $\mathfrak{X}_0$ with finitely bounded tapes T_0, T_1, T_2 ... T_{i-1} *evolves* to ex-machine $\mathfrak{X}_i$ after i halts.

When ex-machine $\mathfrak{X}_0$ evolves to $\mathfrak{X}_1$ and subsequently $\mathfrak{X}_1$ evolves to $\mathfrak{X}_2$ and so on up to ex-machine $\mathfrak{X}_n$, then ex-machine $\mathfrak{X}_i$ is called an *ancestor* of ex-machine $\mathfrak{X}_j$ whenever $0 \leq i < j \leq n$.

Similarly, ex-machine $\mathfrak{X}_j$ is called a *descendant* of ex-machine $\mathfrak{X}_i$ whenever $0 \leq i < j \leq n$.

The sequence of ex-machines $\mathfrak{X}_0 \to \mathfrak{X}_1 \to \ldots \to \mathfrak{X}_n \ldots$ is called an *evolutionary path*.

Languages that $\mathfrak{Q}(x)$ Evolves to Compute

Recall that $\mathfrak{L} = \bigcup_{L \subset \{a\}^*} \{L\}$. Alphabet $A = \{\#, 0, 1, N, Y, a\}$. The initial states are $Q = \{0, h, n, y, t, v, w, x, 8\}$ with halting state h

Let $\mathfrak{X}$ be an ex-machine that is a descendant of $\mathfrak{Q}(x)$. The language L in $\mathfrak{L}$ that $\mathfrak{X}$ computes is defined as follows.

A valid initial tape has the form $\# \quad \#a^n\#$. The valid initial tape $\# \quad \#\#$ represents the empty string.

After machine $\mathfrak{X}$ starts executing with initial tape $\# \quad \#a^n\#$, string a^n is in $\mathfrak{X}$'s language if $\mathfrak{X}$ halts with tape $\#a^n\# \quad Y\#$.

a^n is not in $\mathfrak{X}$'s language if $\mathfrak{X}$ halts with tape $\#a^n\# \quad N\#$.

$\mathfrak{Q}(x)$ Specification

$(0, \#, 8, \#, 1)$
$(8, \#, x, \#, 0)$
$(y, \#, h, Y, 0)$
$(n, \#, h, N, 0)$

$(x, \#, x, 0)$
$(x, a, t, 0)$

$(x, 0, v, \#, 0, (|Q| - 1, \#, n, \#, 1))$
$(x, 1, w, \#, 0, (|Q| - 1, \#, y, \#, 1))$
$(t, 0, w, a, 0, (|Q| - 1, \#, n, \#, 1))$
$(t, 1, w, a, 0, (|Q| - 1, \#, y, \#, 1))$

$(v, \#, n, \#, 1, (|Q| - 1, a, |Q|, a, 1))$
$(w, \#, y, \#, 1, (|Q| - 1, a, |Q|, a, 1))$
$(w, a, |Q|, a, 1, (|Q| - 1, a, |Q|, a, 1))$

$(|Q| - 1, a, x, a, 0)$
$(|Q| - 1, \#, x, \#, 0)$

$\mathfrak{Q}(x)$ Starts with Tape $\#\quad \#aaaa\#\#$ and State 0

STATE	TAPE	INSTRUCTION EXECUTED	NEW INSTRUCTION
8	$\#\#\quad aaaa\#\#\#$	$(0, \#, 8, \#, 1)$	
x	$\#\#\quad aaaa\#\#\#$	$(\lvert Q\rvert - 1, a, x, a, 0)$	$(8, a, x, a, 0)$
t	$\#\#\quad 1aaa\#\#\#$	$(x, a, t, 1_{qr}, 0)$	
w	$\#\#\quad aaaa\#\#\#$	$(t, 1, w, a, 0, (\lvert Q\rvert - 1, \#, y, \#, 1))$	$(8, \#, y, \#, 1)$
9	$\#\#a\quad aaa\#\#\#$	$(w, a, \lvert Q\rvert, a, 1, (\lvert Q\rvert - 1, a, \lvert Q\rvert, a, 1))$	$(8, a, 9, a, 1)$
x	$\#\#a\quad aaa\#\#\#$	$(\lvert Q\rvert - 1, a, x, a, 0)$	$(9, a, x, a, 0)$
t	$\#\#a\quad 1aa\#\#\#$	$(x, a, t, 1_{qr}, 0)$	
w	$\#\#a\quad aaa\#\#\#$	$(t, 1, w, a, 0, (\lvert Q\rvert - 1, \#, y, \#, 1))$	$(9, \#, y, \#, 1)$
10	$\#\#aa\quad aa\#\#\#$	$(w, a, \lvert Q\rvert, a, 1, (\lvert Q\rvert - 1, a, \lvert Q\rvert, a, 1))$	$(9, a, 10, a, 1)$
x	$\#\#aa\quad aa\#\#\#$	$(\lvert Q\rvert - 1, a, x, a, 0)$	$(10, a, x, a, 0)$
t	$\#\#aa\quad 0a\#\#\#$	$(x, a, t, 0_{qr}, 0)$	
w	$\#\#aa\quad aa\#\#\#$	$(t, 0, w, a, 0, (\lvert Q\rvert - 1, \#, n, \#, 1))$	$(10, \#, n, \#, 1)$
11	$\#\#aaa\quad a\#\#\#$	$(w, a, \lvert Q\rvert, a, 1, (\lvert Q\rvert - 1, a, \lvert Q\rvert, a, 1))$	$(10, a, 11, a, 1)$
x	$\#\#aaa\quad a\#\#\#$	$(\lvert Q\rvert - 1, a, x, a, 0)$	$(11, a, x, a, 0)$
t	$\#\#aaa\quad 1\#\#\#$	$(x, a, t, 1_{qr}, 0)$	
w	$\#\#aaa\quad a\#\#\#$	$(t, 1, w, a, 0, (\lvert Q\rvert - 1, \#, y, \#, 1))$	$(11, \#, y, \#, 1)$
12	$\#\#aaaa\quad \#\#\#$	$(w, a, \lvert Q\rvert, a, 1, (\lvert Q\rvert - 1, a, \lvert Q\rvert, a, 1))$	$(11, a, 12, a, 1)$
x	$\#\#aaaa\quad \#\#\#$	$(\lvert Q\rvert - 1, \#, x, \#, 0)$	$(12, \#, x, \#, 0)$
x	$\#\#aaaa\quad 0\#\#$	$(x, \#, x, 0_{qr}, 0)$	
v	$\#\#aaaa\quad \#\#\#$	$(x, 0, v, \#, 0, (\lvert Q\rvert - 1, \#, n, \#, 1))$	$(12, \#, n, \#, 1)$
n	$\#\#aaaa\#\quad \#\#$	$(v, \#, n, \#, 1, (\lvert Q\rvert - 1, a, \lvert Q\rvert, a, 1))$	$(12, a, 13, a, 1)$
h	$\#\#aaaa\#\quad N\#$	$(n, \#, h, N, 0)$	

$\mathfrak{Q}(x)$ Evolved to $\mathfrak{Q}(11010\ x)$

$(0, \#, 8, \#, 1)$ $(y, \#, h, Y, 0)$ $(n, \#, h, N, 0)$

$(x, \#, x, 0)$
$(x, a, t, 0)$

$(x, 0, v, \#, 0, (|Q| - 1, \#, n, \#, 1))$
$(x, 1, w, \#, 0, (|Q| - 1, \#, y, \#, 1))$
$(t, 0, w, a, 0, (|Q| - 1, \#, n, \#, 1))$
$(t, 1, w, a, 0, (|Q| - 1, \#, y, \#, 1))$

$(v, \#, n, \#, 1, (|Q| - 1, a, |Q|, a, 1))$
$(w, \#, y, \#, 1, (|Q| - 1, a, |Q|, a, 1))$
$(w, a, |Q|, a, 1, (|Q| - 1, a, |Q|, a, 1))$

$(|Q| - 1, a, x, a, 0)$
$(|Q| - 1, \#, x, \#, 0)$

$(8, \#, y, \#, 1)$ $(8, a, 9, a, 1)$
$(9, \#, y, \#, 1)$ $(9, a, 10, a, 1)$
$(10, \#, n, \#, 1)$ $(10, a, 11, a, 1)$
$(11, \#, y, \#, 1)$ $(11, a, 12, a, 1)$

$(12, \#, n, \#, 1)$ $(12, a, 13, a, 1)$

$\mathfrak{Q}(x)$ Evolving to Compute Some L_f

Each infinite downard path in the infinite binary tree corresponds to a unique language L_f, where string a^n lies in L_f if and only if the $n + 1$th branch of the downward path is a 1.

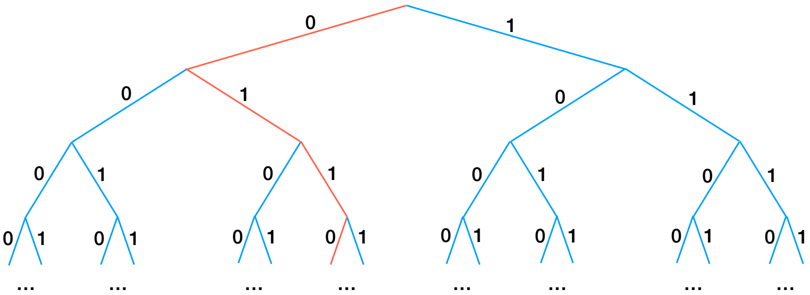


Figure: Infinite Binary Tree

$\mathfrak{Q}(x)$ Evolving to Compute Some L_f

An execution of $\mathfrak{Q}(x)$ on initial tape $\#\ \ \#a^n\#$ executes a random instruction $n+1$ times, creating a finite downward path of length $n+1$. After this execution, the descendant is $\mathfrak{Q}(f(0)f(1)\dots f(n)\ x)$, where $f(i)$ is the random bit measured in the $i+1$th execution of a random instruction.

LEMMA: Assume $i \leq n$. If $f(i) = 1$, then $\mathfrak{Q}(f(0)f(1)\dots f(n)\ x)$ on initial tape $\#\ \ \#a^i\#$ halts with tape $\#a^i\#\ \ Y\#$. If $f(i) = 0$, then $\mathfrak{Q}(f(0)f(1)\dots f(n)\ x)$ halts with $\#a^i\#\ \ N\#$.

THEOREM: For functions $f : \mathbb{N} \to \{0, 1\}$, the probability that language L_f is Turing incomputable has measure 1 in $(\nu, \mathfrak{L})$.

COROLLARY: $\mathfrak{Q}(x)$ evolves to compute a Turing Incomputable language L_f with probability measure 1 in $(\nu, \mathfrak{L})$.

$\mathfrak{Q}(x)$ Evolving to Compute $L_{h_{\epsilon_a}}$

Universal Turing machine / enumeration theorem, there is a Turing computable enumeration $\mathcal{E} : \mathbb{N} \rightarrow \{$ all Turing machines $\mathcal{M}\} \times \{$ Each of $\mathcal{M}$'s states as an initial state $\}$

This enumeration uses the blank-tape halting problem.

Set alphabet $\mathcal{A} = \{\#, 0, 1, a, A, B, M, N, S, X, Y\}$.

Let $\mathcal{M}_{\mathcal{E}}$ be the Turing machine that computes $\mathcal{E}_a : \mathbb{N} \rightarrow \mathcal{A}^* \times \mathbb{N}$, where tape $\#\ \ \#a^n\#$ represents the natural number n in the domain of $\mathcal{E}_a$.

$\mathbb{N}$ in the range of $\mathcal{E}_a$ holds the initial state of machine $\mathcal{E}_a(n)$.

$\mathfrak{Q}(x)$ Evolving to Compute Halting Language $L_{h_{\epsilon_a}}$

REMARK: For each $n \in \mathbb{N}$, with blank initial tape and initial state $\mathcal{E}_a(n)$ (2nd coordinate), then Turing machine $\mathcal{E}_a(n)$ (first coordinate) either halts or does not halt.

Define *halting function* $h_{\mathcal{E}_a} : \mathbb{N} \to \{0, 1\}$, where $h_{\mathcal{E}_a}(n) = 1$ if Turing machine $\mathcal{E}_a(n)$ halts with blank initial tape and initial state $\mathcal{E}_a(n)$. $h_{\mathcal{E}_a}(n) = 0$ if Turing machine $\mathcal{E}_a(n)$ does not halt.

Define *halting language* $L_{h_{\mathcal{E}_a}} = \{a^n : h_{\mathcal{E}_a}(n) = 1\}$.

THEOREM: The evolutionary path $\mathfrak{Q}(h_{\mathcal{E}_a}(0)\ \ x) \to \mathfrak{Q}(h_{\mathcal{E}_a}(0)\ h_{\mathcal{E}_a}(1)\ \ x) \to \dots\ \mathfrak{Q}(h_{\mathcal{E}_a}(0)\ h_{\mathcal{E}_a}(1) \dots h_{\mathcal{E}_a}(m)\ \ x)\ \dots$ computes halting language $L_{h_{\mathcal{E}_a}}$

Halting Complexity Questions

Can we find or define a measure of halting complexity to appropriately ask the next question?

(The Shannon complexity $|Q||A|$ is not adequate.)

Can this finite halting complexity of a Turing machine H be characterized as follows? There exists a threshold halting complexity $\theta(M)$ so that if M's halting complexity is greater than $\theta(M)$, then H cannot determine M's halting behavior.

(One approach is to assume an initially blank tape to assure that there is not complexity hidden in the different initial tapes.)

Self-Modification Questions

For a fixed Turing machine M, do there exist ex-machine self-modification procedures in some ex-machine X that start with a finite number of standard, random and meta instructions that can evolve to determine M's halting behavior with probability measure 1?

Does there exist a sufficiently high halting complexity for M, where these self-modification procedures fail?