Most Programs Stop Quickly or Never Halt

C. S. Calude (University of Auckland)
Joint work with M. Stay (University of California Riverside)

Outline

Outline

The halting problem

The halting problem for Turing machines is the problem to decide whether an arbitrary Turing machine eventually halts on an arbitrary input.

Does there exist a Turing machine T_{halt} which given the code $code(T)$ of a Turing machine T, and the input x, eventually stops and produces 1 if $T(x)$ stops and 0 if $T(x)$ does not stop?

Turing's (in)famous result states that the halting problem cannot be solved by any Turing machine, i.e. there is no such T_{halt}.

The halting problem

The halting problem for Turing machines is the problem to decide whether an arbitrary Turing machine eventually halts on an arbitrary input.

Does there exist a Turing machine T_{halt} which given the code $code(T)$ of a Turing machine T, and the input x, eventually stops and produces 1 if $T(x)$ stops and 0 if $T(x)$ does not stop?

Turing's (in)famous result states that the halting problem cannot be solved by any Turing machine, i.e. there is no such T_{halt}.

The halting problem

The halting problem for Turing machines is the problem to decide whether an arbitrary Turing machine eventually halts on an arbitrary input.

Does there exist a Turing machine T_{halt} which given the code $code(T)$ of a Turing machine T, and the input x, eventually stops and produces 1 if $T(x)$ stops and 0 if $T(x)$ does not stop?

Turing's (in)famous result states that the halting problem cannot be solved by any Turing machine, i.e. there is no such T_{halt}.

Outline

The halting problem and mathematical knowledge

How large is the (clearly infinite) set of Turing machines for which we can solve the halting problem?

Probabilistically solving the halting problem, even for relatively small-size programs, can be a useful mathematical tool.

AIT classical approach (Omega and variants) does not provide a solution. Is there any other way?

The halting problem and mathematical knowledge

How large is the (clearly infinite) set of Turing machines for which we can solve the halting problem?

Probabilistically solving the halting problem, even for relatively small-size programs, can be a useful mathematical tool.

AIT classical approach (Omega and variants) does not provide a solution. Is there any other way?

The halting problem and mathematical knowledge

How large is the (clearly infinite) set of Turing machines for which we can solve the halting problem?

Probabilistically solving the halting problem, even for relatively small-size programs, can be a useful mathematical tool.

AIT classical approach (Omega and variants) does not provide a solution. Is there any other way?

A universal prefix-free machine 1

A register machine has a finite number of registers, each of which may contain an arbitrarily large non-negative integer. The register machine *U* (labelled) instructions are:

L: EQ R1 R2 R3
L: SET R1 R2
L: ADD R1 R2
L: READ R1
L: HALT

Goldbach's Conjecture, Riemann Hypothesis, Collatz's Conjecture

There exists a program T_{Goldbach} (135 instructions totalling 3,484 bits) which never stops iff Goldbach's Conjecture is true.

There exists a program T_{Riemann} (290 instructions totalling 7,780 bits) which never stops iff the Riemann Hypothesis is true.

There is a *non-constructive* way to prove the existence of a program T_{Collatz} which never stops iff the Collatz's Conjecture is true.

Goldbach's Conjecture, Riemann Hypothesis, Collatz's Conjecture

There exists a program T_{Goldbach} (135 instructions totalling 3,484 bits) which never stops iff Goldbach's Conjecture is true.

There exists a program T_{Riemann} (290 instructions totalling 7,780 bits) which never stops iff the Riemann Hypothesis is true.

There is a *non-constructive* way to prove the existence of a program T_{Collatz} which never stops iff the Collatz's Conjecture is true.

Goldbach's Conjecture, Riemann Hypothesis, Collatz's Conjecture

There exists a program T_{Goldbach} (135 instructions totalling 3,484 bits) which never stops iff Goldbach's Conjecture is true.

There exists a program T_{Riemann} (290 instructions totalling 7,780 bits) which never stops iff the Riemann Hypothesis is true.

There is a *non-constructive* way to prove the existence of a program T_{Collatz} which never stops iff the Collatz's Conjecture is true.

Main question

We are given a universal prefix-free machine U and a randomly chosen program p of length N that we know not to stop by time t.

Can we effectively evaluate the probability that $U(p)$ eventually stops?

Main question

We are given a universal prefix-free machine U and a randomly chosen program p of length N that we know not to stop by time t.

Can we effectively evaluate the probability that $U(p)$ eventually stops?

Outline

Intuition

It is easy to write a short program that does not stop, but

it seems difficult to write a short program which

- halts, and
- takes a very long time to stop.

Intuition

It is easy to write a short program that does not stop, but

it seems difficult to write a short program which

- halts, and
- takes a very long time to stop.

Intuition

It is easy to write a short program that does not stop, but

it seems difficult to write a short program which

- halts, and
- takes a very long time to stop.

Intuition

It is easy to write a short program that does not stop, but

it seems difficult to write a short program which

- halts, and
- takes a very long time to stop.

A bit of notation

Let $\mathrm{bin}(i)$ be the ith binary string: $\mathrm{bin}(1) =$ the empty string, $\mathrm{bin}(2) = 0$, $\mathrm{bin}(3) = 1$, $\mathrm{bin}(4) = 00$, $\mathrm{bin}(5) = 01$,...
and let

$$H(x) = H_U(x) = \min\{|p| \mid U(p) = x\}$$

be the program-size complexity (induced by U).

Is there a preferred time when a program can stop? 1

Theorem [Chaitin, 1987]

There is a constant c such that if $U(\text{bin}(i))$ halts exactly in time t, then

$$H(\text{bin}(t)) \leq |\text{bin}(i)| + c.$$

Difficulty: H is not computable.

Problem: Can we "effectively extract" any information from the above inequality?

Is there a preferred time when a program can stop? 1

Theorem [Chaitin, 1987]

There is a constant c such that if $U(\text{bin}(i))$ halts exactly in time t, then

$$H(\text{bin}(t)) \leq |\text{bin}(i)| + c.$$

Difficulty: H is not computable.

Problem: Can we "effectively extract" any information from the above inequality?

Is there a preferred time when a program can stop? 1

Theorem [Chaitin, 1987]

There is a constant c such that if $U(\text{bin}(i))$ halts exactly in time t, then

$$H(\text{bin}(t)) \leq |\text{bin}(i)| + c.$$

Difficulty: H is not computable.

Problem: Can we "effectively extract" any information from the above inequality?

Is there a preferred time when a program can stop? 2

A time t is algorithmically random if

$$H(\text{bin}(t)) \geq |\text{bin}(t)| - \log |\text{bin}(t)|.$$

Fact

Algorithmically random strings of any length exit. They have effective density one.

Is there a preferred time when a program can stop? 2

A time t is algorithmically random if

$$H(\text{bin}(t)) \geq |\text{bin}(t)| - \log |\text{bin}(t)|.$$

Fact

Algorithmically random strings of any length exit. They have effective density one.

Is there a preferred time when a program can stop? 3

Theorem

Assume that an N-bit program $U(p)$ has not stopped by time $2^{2N+2c+1}$, where $N \geq 2$ and c comes from Chaitin's theorem. Then,

$$U(p) \text{ cannot stop exactly at any algorithmically random time } t \geq 2^{2N+2c+1}.$$

Density

A time t is called "exponential stopping time" if there is a program p which stops on U exactly at time $t > 2^{2|p|+2c+1}$.

Theorem

The set of exponential stopping times has effectively zero density.

Density

A time t is called "exponential stopping time" if there is a program p which stops on U exactly at time $t > 2^{2|p|+2c+1}$.

Theorem

The set of exponential stopping times has effectively zero density.

Outline

Postulate

We assume an a priori computable probability distribution on all possible runtimes: we choose a time i randomly from a probability distribution $\rho(i)$ which effectively converges to 1, that is, there exists a computable function B such that for every $n \geq B(k)$,

$$|1 - \sum_{i=1}^{n} \rho(i)| < 2^{-k}.$$

Consequently, the probability space is the product of the space of programs and the time space:

$$Space_{\{\rho_i\}} = \{0,1\}^* \times \{1,2,\ldots\},$$

where N-bit programs are assumed to be uniformly distributed, and the observer time flows according to the computable probability distribution $\{\rho_i\}$.

Postulate

We assume an a priori computable probability distribution on all possible runtimes: we choose a time i randomly from a probability distribution $\rho(i)$ which effectively converges to 1, that is, there exists a computable function B such that for every $n \geq B(k)$,

$$|1 - \sum_{i=1}^{n} \rho(i)| < 2^{-k}.$$

Consequently, the probability space is the product of the space of programs and the time space:

$$Space_{\{\rho_i\}} = \{0,1\}^* \times \{1,2,\ldots\},$$

where N-bit programs are assumed to be uniformly distributed, and the observer time flows according to the computable probability distribution $\{\rho_i\}$.

Example of a computable time distribution

For every program bin(i) let $t_{\text{bin}(i)}$ be the exact time $U(\text{bin}(i))$ stops, if $U(\text{bin}(i)$ halts, or $t_{\text{bin}(i)} = \infty$, otherwise.

Define the computable number:

$$0 < \Upsilon_U = \sum_{i \geq 1} 2^{-i} / t_{\text{bin}(i)} < 1.$$

Then, define the computable probability distribution

$$\rho_i = \frac{2^{-i}}{t_{\text{bin}(i)} \cdot \Upsilon_U}.$$

Example of a computable time distribution

For every program bin(i) let $t_{\text{bin}(i)}$ be the exact time $U(\text{bin}(i))$ stops, if $U(\text{bin}(i)$ halts, or $t_{\text{bin}(i)} = \infty$, otherwise.

Define the computable number:

$$0 < \Upsilon_U = \sum_{i \geq 1} 2^{-i} / t_{\text{bin}(i)} < 1.$$

Then, define the computable probability distribution

$$\rho_i = \frac{2^{-i}}{t_{\text{bin}(i)} \cdot \Upsilon_U}.$$

Example of a computable time distribution

For every program bin(i) let $t_{\text{bin}(i)}$ be the exact time $U(\text{bin}(i))$ stops, if $U(\text{bin}(i)$ halts, or $t_{\text{bin}(i)} = \infty$, otherwise.

Define the computable number:

$$0 < \Upsilon_U = \sum_{i \geq 1} 2^{-i} / t_{\text{bin}(i)} < 1.$$

Then, define the computable probability distribution

$$\rho_i = \frac{2^{-i}}{t_{\text{bin}(i)} \cdot \Upsilon_U}.$$

Halting probability

Theorem

On time we consider a probability distribution ρ_i which effectively converges to 1. Then, for each $k > 0$ we can effectively compute a time t_k (which depends upon U, ρ) such that the probability that an N-bit program which hasn't stopped on U by time t_k will eventually halt, is smaller than 2^{-k}.

Corollary

Assume that $U(p)$ has not stopped by time $T > k - \lfloor \log \Upsilon_U \rfloor$. Then, the probability (according to the above probability space $Space_{\{\rho_i\}}$) that $U(p)$ eventually halts is smaller than 2^{-k}.

Halting probability

Theorem

On time we consider a probability distribution ρ_i which effectively converges to 1. Then, for each $k > 0$ we can effectively compute a time t_k (which depends upon U, ρ) such that the probability that an N-bit program which hasn't stopped on U by time t_k will eventually halt, is smaller than 2^{-k}.

Corollary

Assume that $U(p)$ has not stopped by time $T > k - \lfloor \log \Upsilon_U \rfloor$. Then, the probability (according to the above probability space $Space_{\{\rho_i\}}$) that $U(p)$ eventually halts is smaller than 2^{-k}.

Decomposition 1

Theorem

Assume that U and $Space_{\{\rho(i)\}}$ have been fixed. For every integer $k > 0$, the set of halting programs for U can be written as a disjoint union of a computable set and a set of probability effectively smaller than 2^{-k}.

Decomposition 2

The asymptotic density of a set B of one-way Turing machine programs is the limit (if it exists) of the proportion of all programs with n non-final states in B in the set of all programs with n non-final states.

Theorem [Hamkins, Miasnikov, 2006]

There is a set B of one-way infinite Turing machine programs such that

1. B has asymptotic probability one,
2. B is polynomial time decidable,
3. The halting problem $H \cap B$ is polynomial time decidable.

Decomposition 3

Hamkins–Miasnikov Theorem is machine-dependent.

Theorem [Ryabalov, 2007]

There is no strongly generic set (i.e. its asymptotic density exponentially converges to zero) on which the halting problem is decidable.

Outline

- C. S. Calude, Elena Calude, M. J. Dinneen. A new measure of the difficulty of problems, *Journal for Multiple-Valued Logic and Soft Computing* 12 (2006), 285–307.
- C. S. Calude, M. A. Stay. Most programs stop quickly or never halt, *Adv. Appl. Math.* (2007), doi:10.1016/j.amm.2007.01.001.
- J. D. Hamkins, A. Miasnikov. The halting problem is decidable on a set of asymptotic probability one, *Notre Dame J. Formal Logic* 47, 4 (2006), 515–524.
- A. Rybalov. On the strongly generic undecidability of the Halting Problem, *Theoret. Comput. Sci.* 377 (2007), 268–270.