Winter School 2016

Physics and Computation
— Session 1: Computing Paradigm —

José Félix Costa[1,2]

[1] Departamento de Matemática, Instituto Superior Técnico
[2] CFCUL – Centro de Filosofia das Ciências da Universidade de Lisboa
fgc@math.tecnico.ulisboa.pt

2 – 4 February, 2016

Stonehenge's Computer

Stonehenge I, Heelstone

Stonehenge III

Stonehenge III

O carvalho – árvore sagrada

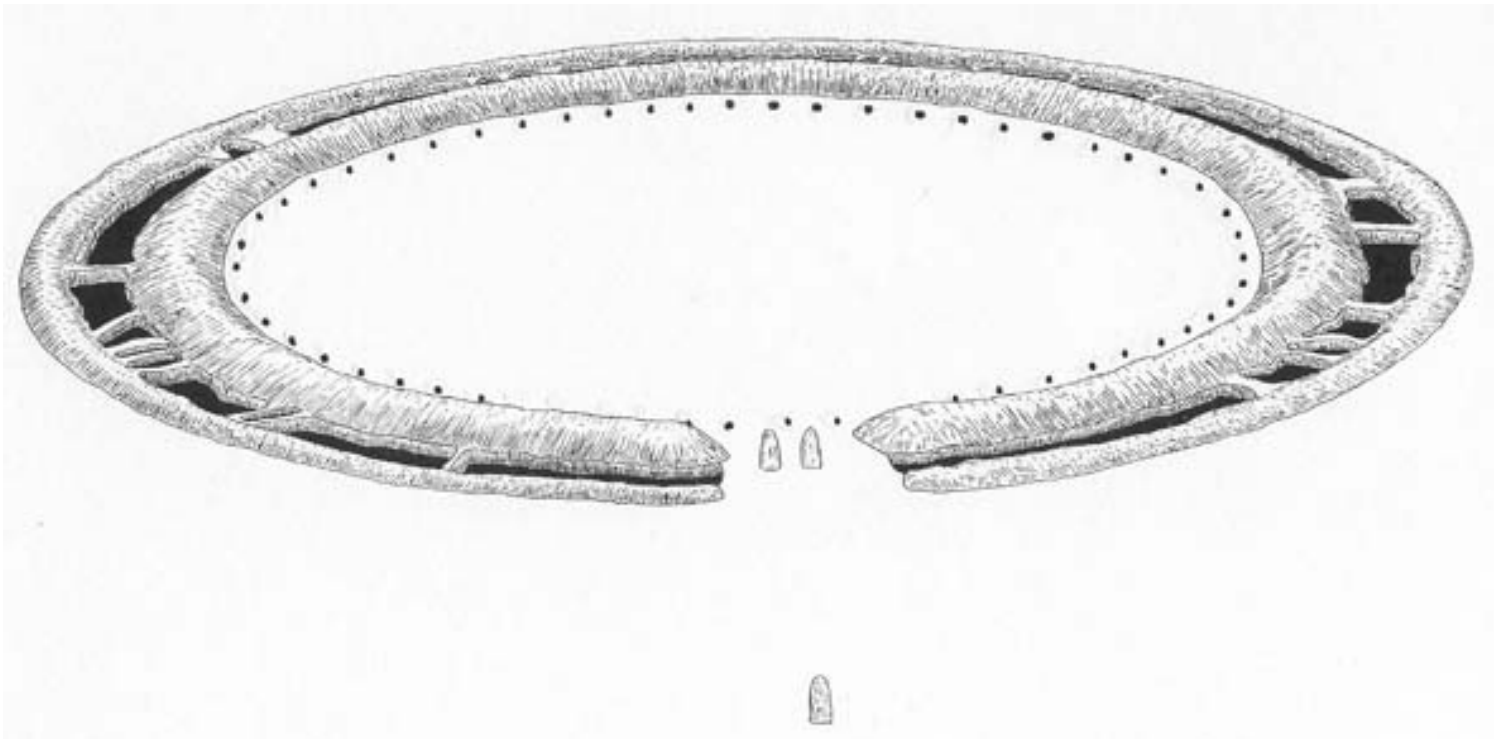
Ray Bradbury

Oh, ventos outonais que queimam
E mergulham todo o mundo numa noite sem luar
Desejo que soprem e me queiram
Num monte de folhas da árvore do Outono me tornar!

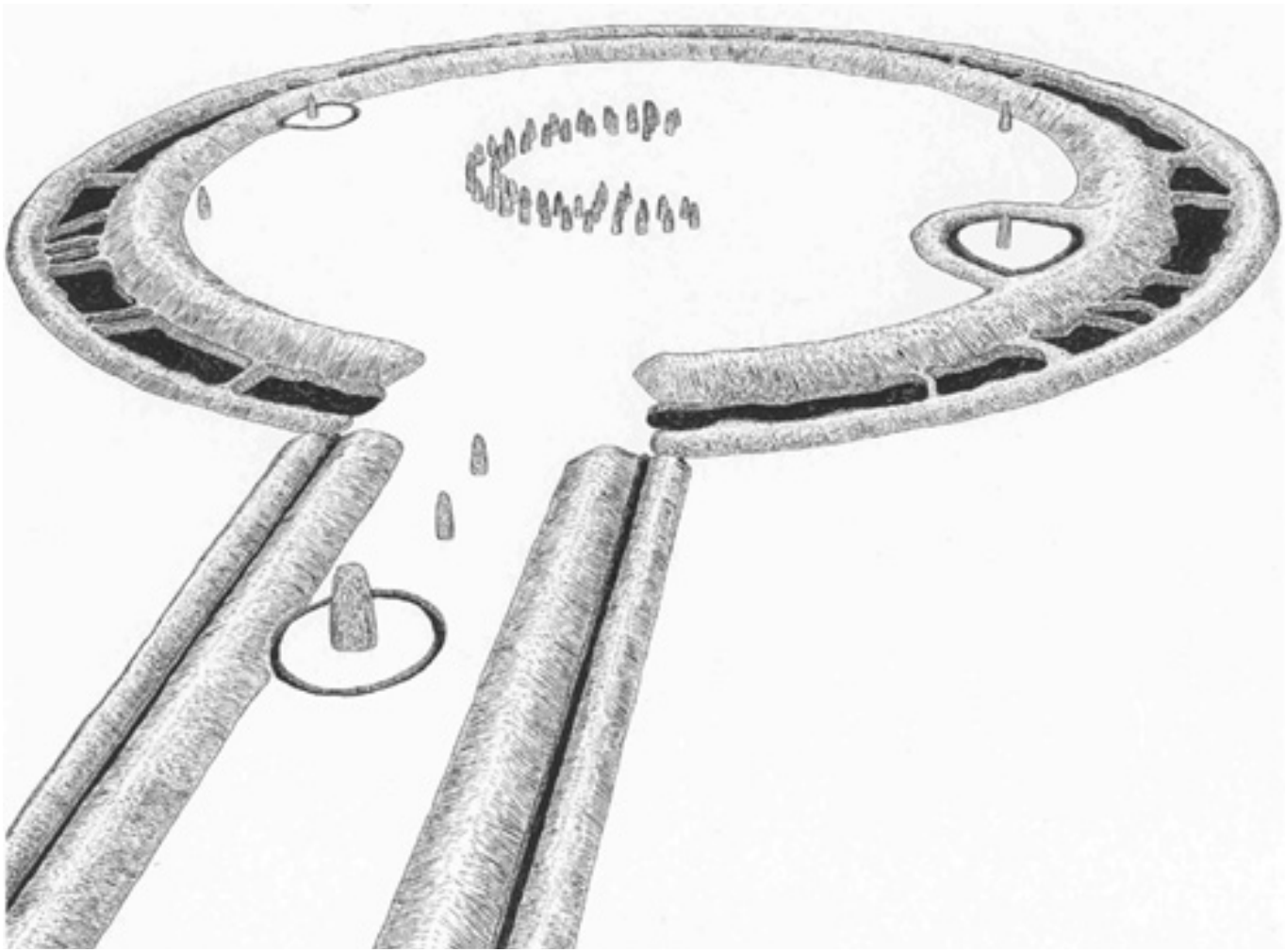
Oh, Samhain, Deus dos Mortos
Escuta-nos!
Nós os sagrados Padres Druídas
Reunidos junto aos carvalhos
Pelas almas dos mortos oramos.
Com fervor oramos pelas almas
Dos que em animais foram transformados
Oh, Deus dos Mortos estes animais sacrificamos
Para que sejam libertadas
As almas dos nossos muito amados
Este ano pela morte arrebatados.

O Algoritmo dos Eclipses

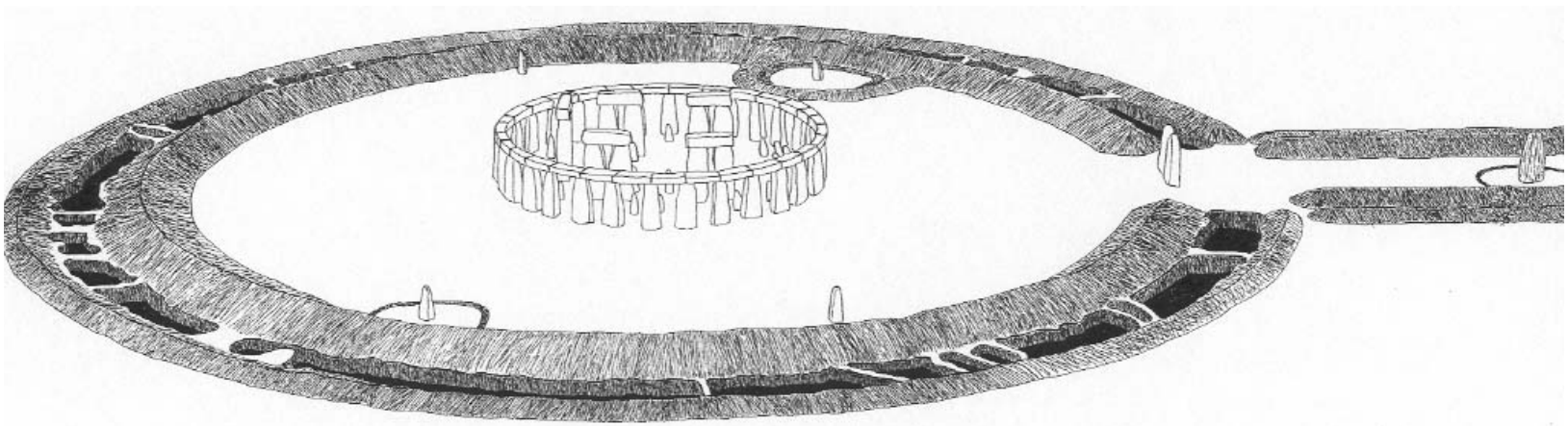
Stonehenge I: 2700 – 2600 a.C.



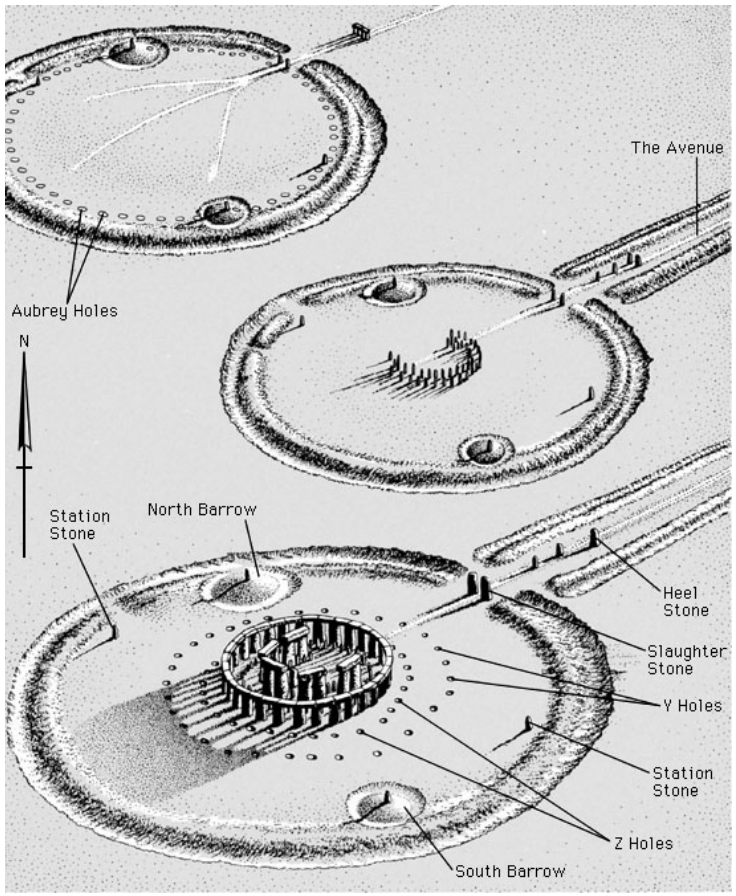
Stonehenge II: 2600 – 2000 a.C.



Stonehenge III: 2000 – 1700 a.C.



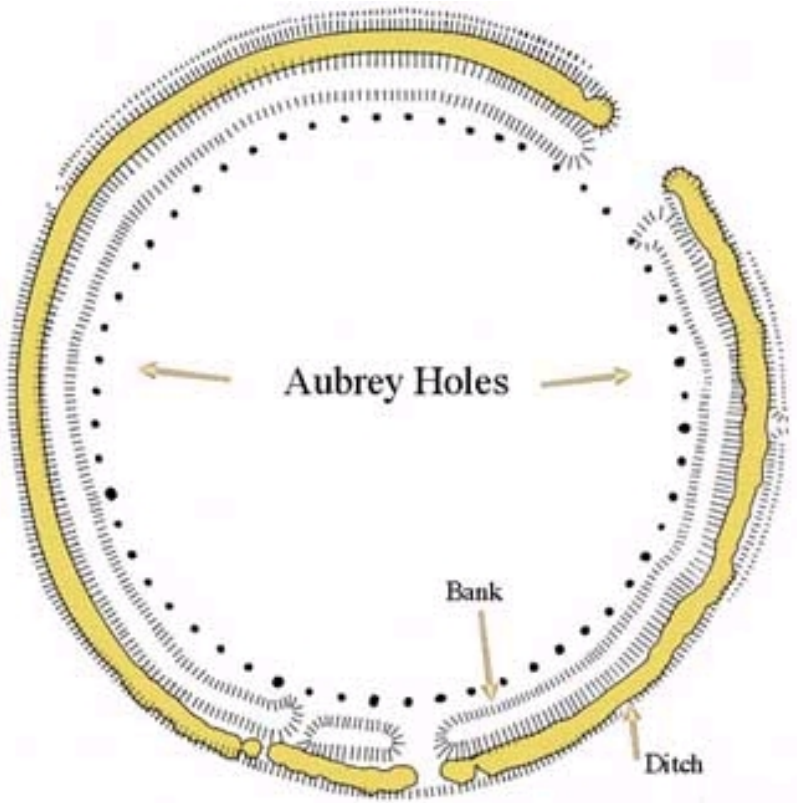
Stonehenge I, II e III



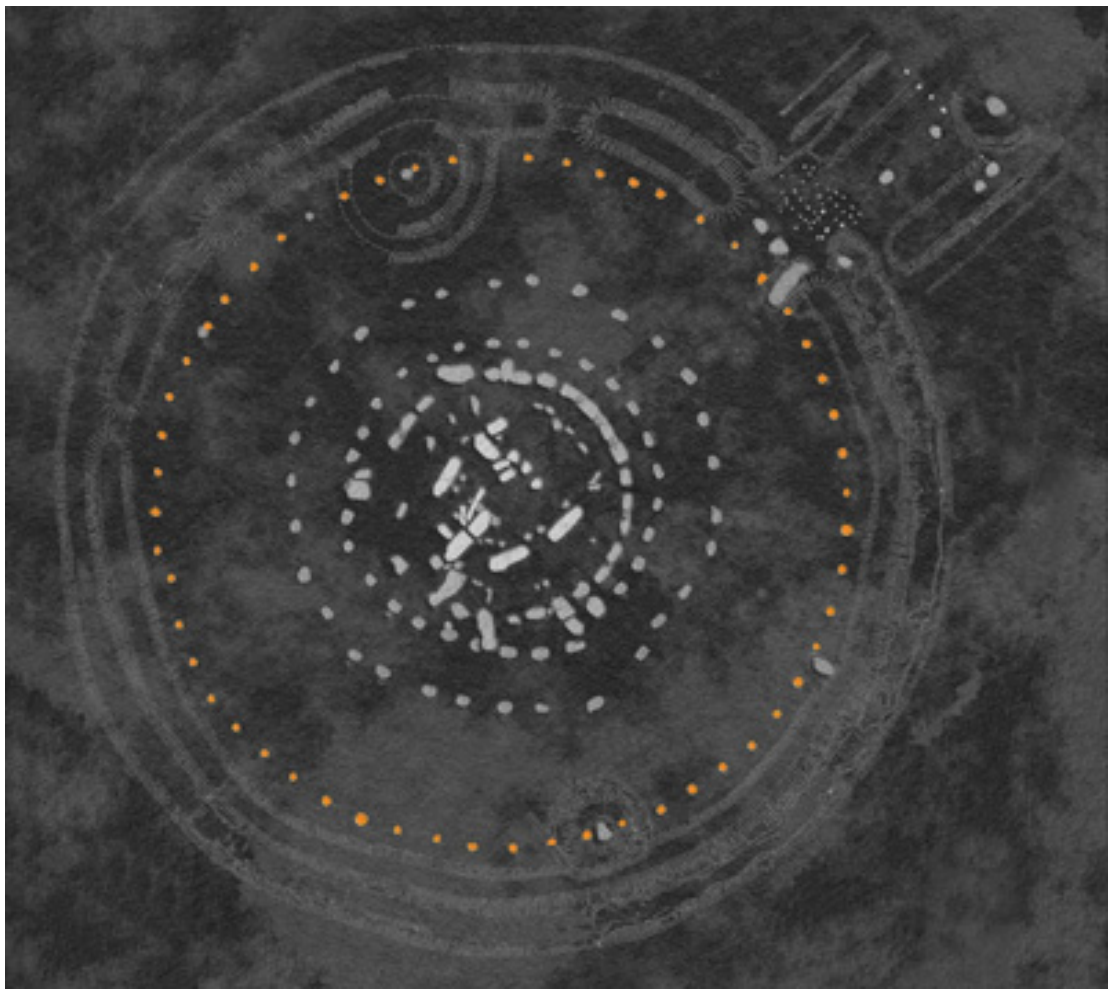
Stonehenge I: "Aubrey holes"

Stonehenge I: "Aubrey holes"

Stonehenge I: "Aubrey holes"



Stonehenge I: "Aubrey holes"



Precessão da linha dos nodos lunares ($18,61$ anos) ([Hoy72])

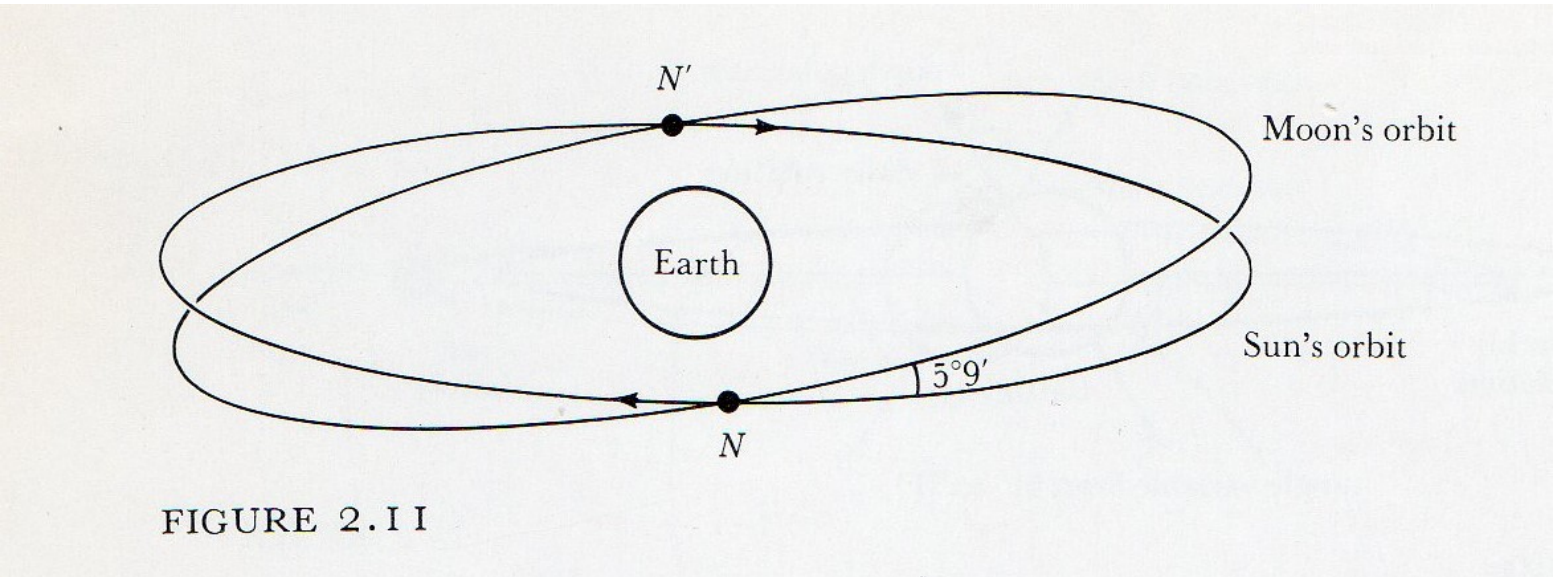
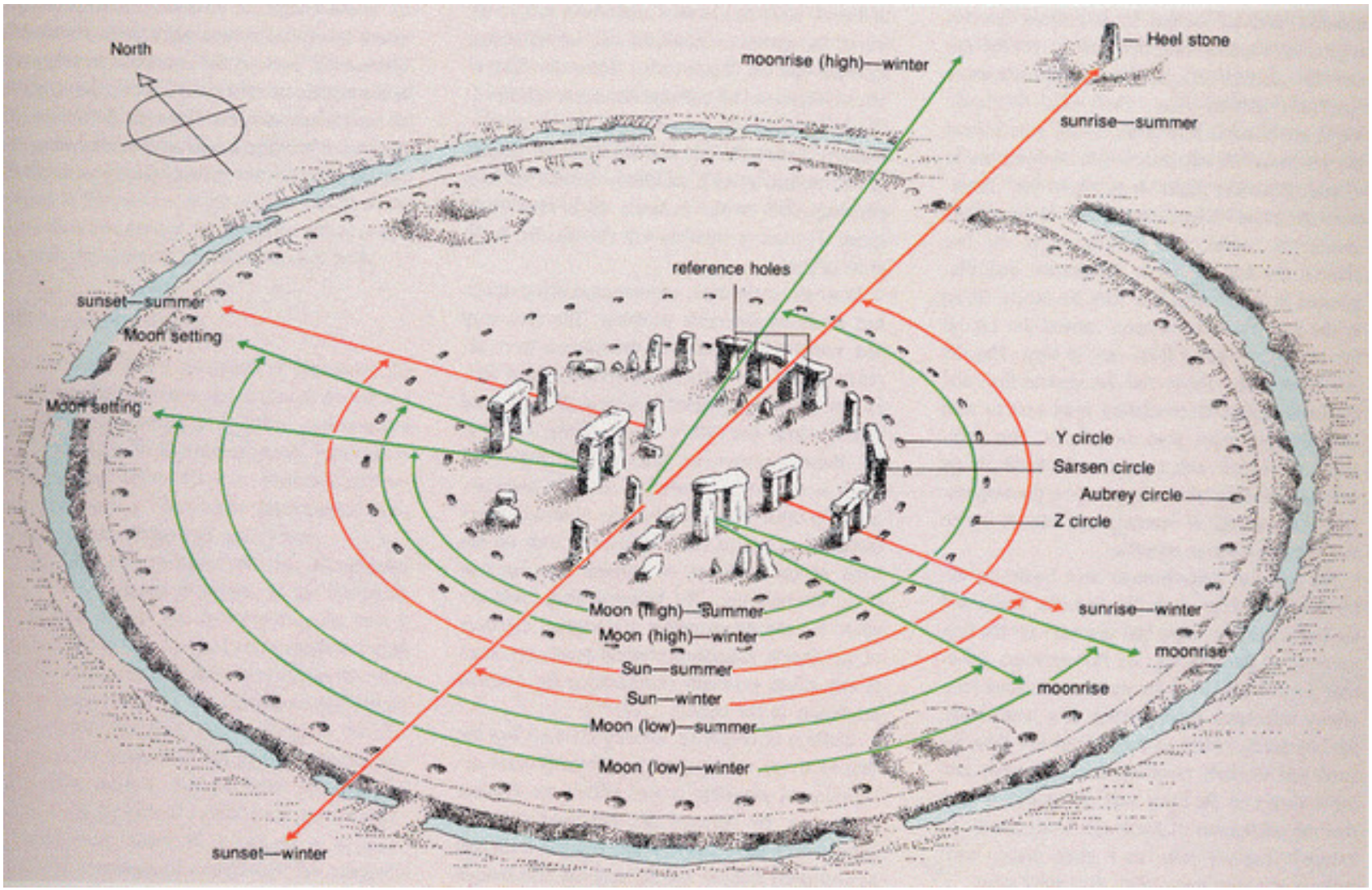
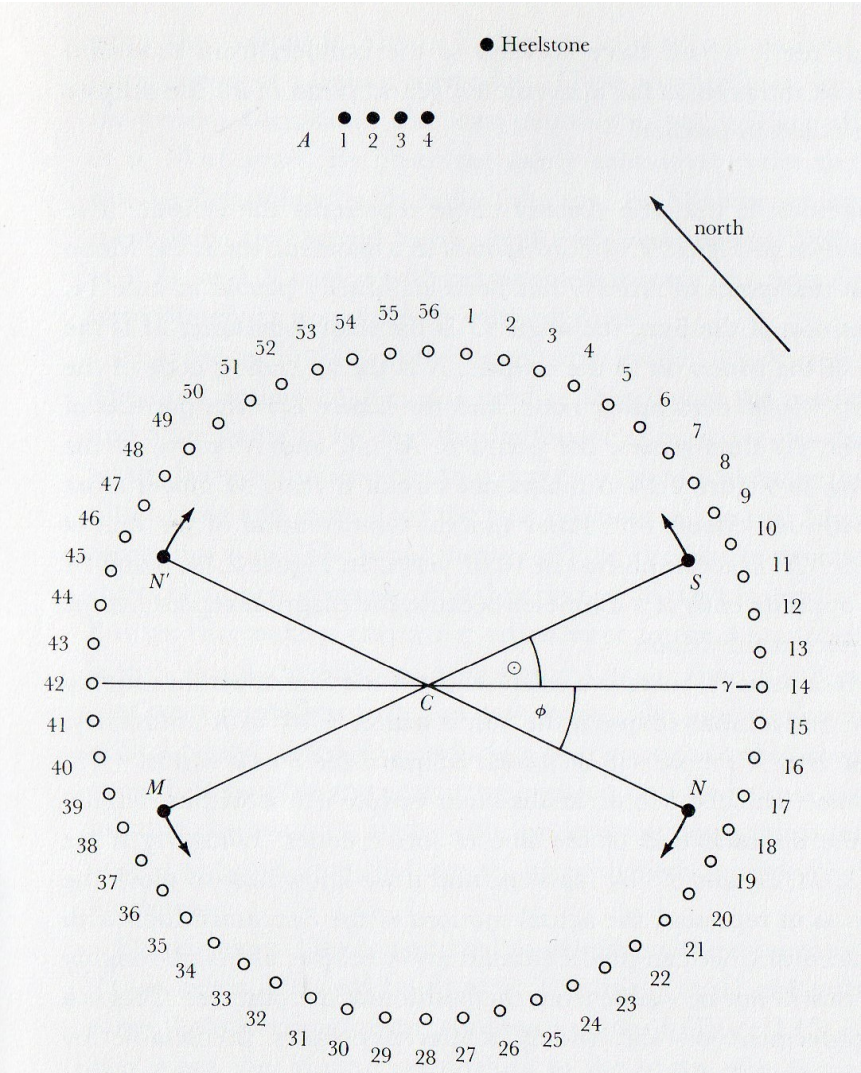


FIGURE 2.11

Snonehenge: Direções com significado astronómico ([Hoy72])



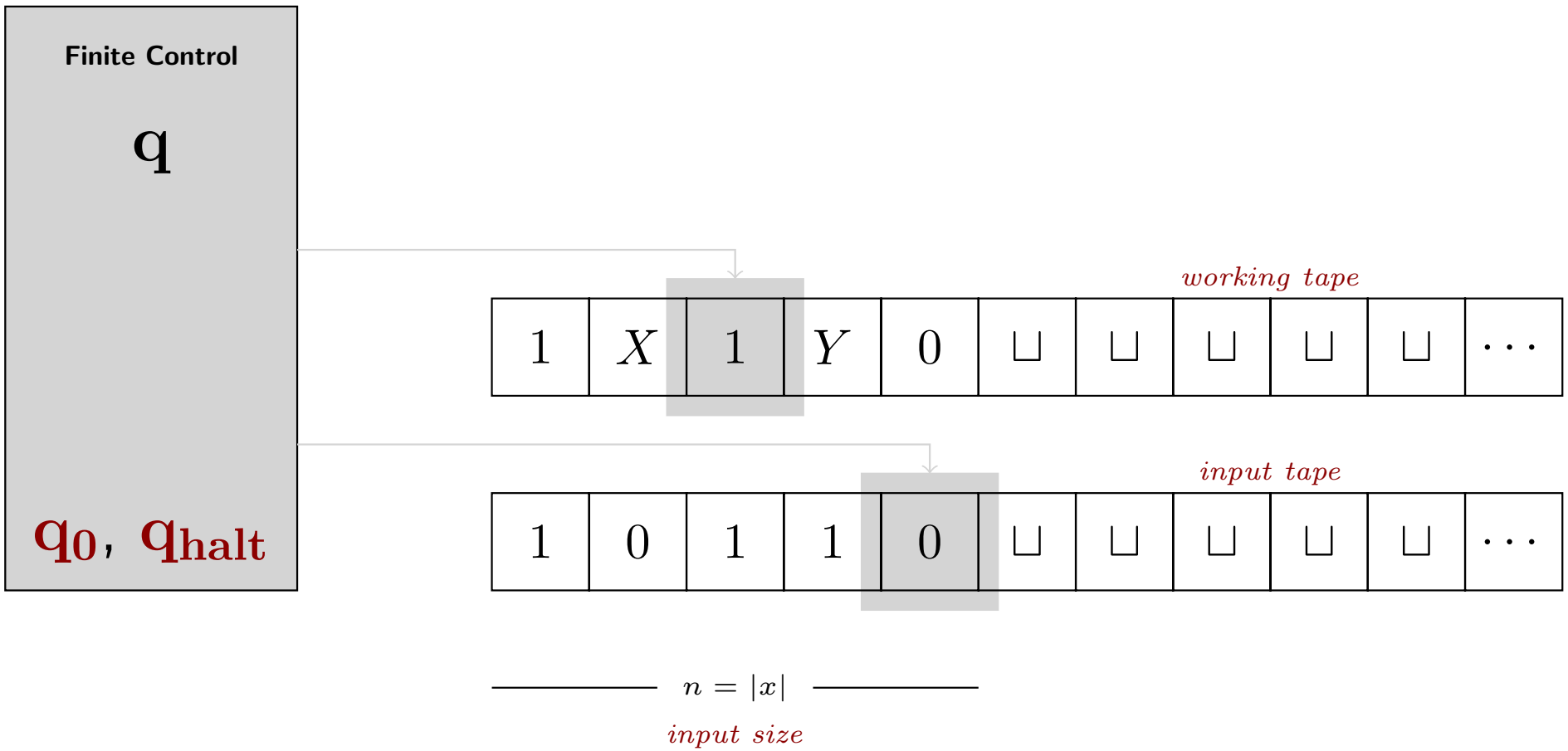
Snonehenge: Algoritmo dos eclipses ([Hoy72])



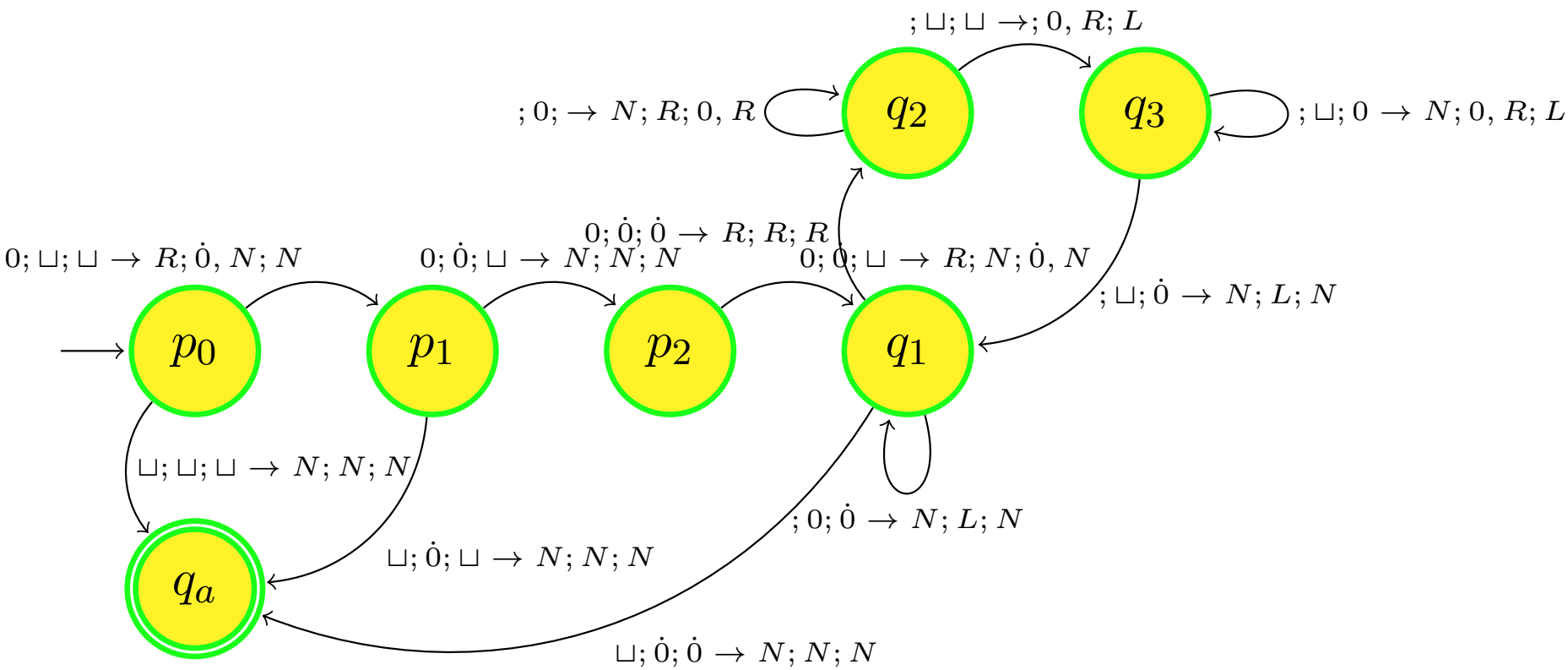
Stonehenge: Raio de Sol no dia mais longo

The Turing Machine

The Turing machine: How it works



The exponential



Sending pi away!

```
import random

import math

countinside = 0

for count in range(0, 10000):

    d =math.hypot

    (
    random.random(), random.random()
    )

    if d < 1: countinside += 1

count += 1

print 4.0* countinside / count
```

Poe, E.

Near a Raven

Midnights so dreary, tired and weary.
    Silently pondering volumes extolling all by-now obsolete lore.
During my rather long nap – the weirdest tap!
    An ominous vibrating sound disturbing my chamber's antedoor.
        "This", I whispered quietly, "I ignore".

Perfectly, the intellect remembers: the ghostly fires, a glittering ember.
    Inflamed by lightning's outbursts, windows cast penumbras upon this floor.
Sorrowful, as one mistreated, unhappy thoughts I heeded:
    That inimitable lesson in elegance – Lenore –
        Is delighting, exciting... nevermore.

(Mike Keith, 1995)

Specifying a Turing machine

Turing machine with $k > 2$ tapes, with an output tape; dynamic map

$$\delta : Q \times \Gamma^{k-1} \to Q \times \Gamma^{k-1} \times \{L, N, R\}^k$$

Example (Iterating the Collatz function: TM without output tape)

input n;

while $n \neq 1$ do if $even(n)$ then $n := n/2$ else $n := 3n + 1$

Specifying a Turing machine

Turing machine with $k > 2$ tapes, with an output tape; dynamic map

$$\delta : Q \times \Gamma^{k-1} \to Q \times \Gamma^{k-1} \times \{L, N, R\}^k$$

Example (Iterating the Collatz function: TM without output tape)

input n;

while $n \neq 1$ do if $even(n)$ then $n := n/2$ else $n := 3n + 1$

Specifying a Turing machine

Turing machine with $k > 2$ tapes, with an output tape; dynamic map

$$\delta : Q \times \Gamma^{k-1} \to Q \times \Gamma^{k-1} \times \{L, N, R\}^k$$

Example (Iterating the Collatz function: TM without output tape)

input n;

while $n \neq 1$ do if $even(n)$ then $n := n/2$ else $n := 3n + 1$

Semantics of algorithm

Computable function

Each Turing machine $\mathcal{M}$ (with an output tape) computes a function $f_{\mathcal{M}} : \{0,1\}^\star \to \{0,1\}^\star$ (in other words, a function $f_{\mathcal{M}} : \mathbb{N} \to \mathbb{N}$); if the machine does not halt on an input x, then $f_{\mathcal{M}}$ is not defined at x.

Bijection between $\mathbb{N}$ and $\{0,1\}^\star$

Take a binary word, e.g., 0101, affix a leftmost 1 (10101), and read it in decimal having subtracted 1 (20). Take any number in decimal, e.g., 20, add 1, write it in binary (10101), remove the leftmost 1, and read the result as a binary word (0101).

Semantics of algorithm

Computable function

Each Turing machine $\mathcal{M}$ (with an output tape) computes a function $f_{\mathcal{M}} : \{0,1\}^\star \to \{0,1\}^\star$ (in other words, a function $f_{\mathcal{M}} : \mathbb{N} \to \mathbb{N}$); if the machine does not halt on an input x, then $f_{\mathcal{M}}$ is not defined at x.

Bijection between $\mathbb{N}$ and $\{0,1\}^\star$

Take a binary word, e.g., 0101, affix a leftmost 1 (10101), and read it in decimal having subtracted 1 (20). Take any number in decimal, e.g., 20, add 1, write it in binary (10101), remove the leftmost 1, and read the result as a binary word (0101).

Universal Turing machine

Theorem (Universal Turing machine)

There exists a universal Turing machine $\mathcal{U}$ that receives as input $\langle \mathcal{M}, x \rangle$, the binary code of a Turing machine $\mathcal{M}$ and a binary word x, such that, for every such $\mathcal{M}$ and for every such x, it simulates $\mathcal{M}$ on input x:

$$\mathcal{U}(\langle \mathcal{M}, x \rangle) \equiv \mathcal{M}(x)$$

Universal Turing machine

Theorem (Universal Turing machine)

There exists a universal Turing machine $\mathcal{U}$ that receives as input $\langle \mathcal{M}, x \rangle$, the binary code of a Turing machine $\mathcal{M}$ and a binary word x, such that, for every such $\mathcal{M}$ and for every such x, it simulates $\mathcal{M}$ on input x:

$$\mathcal{U}(\langle \mathcal{M}, x \rangle) \equiv \mathcal{M}(x)$$

Universal Turing machine

Theorem (Universal Turing machine)

There exists a universal Turing machine $\mathcal{U}$ that receives as input $\langle \mathcal{M}, x \rangle$, the binary code of a Turing machine $\mathcal{M}$ and a binary word x, such that, for every such $\mathcal{M}$ and for every such x, it simulates $\mathcal{M}$ on input x:

$$\mathcal{U}(\langle \mathcal{M}, x \rangle) \equiv \mathcal{M}(x)$$

Limits

Theorem

There is a countable infinite number of computable functions, but an uncountable number of non-computable functions. (I.e., most of the functions $f : \{0,1\}^\star \to \{0,1\}^\star$ *are non-algorithmic.)*

Growing rate and the halting problem

Theorem (Busy Beaver)

The Turing machine can not compute functions with arbitrary growing rate. There is a well-known established limit of growing rate for computable functions.

Theorem (The halting problem)

The halting function, namely the function

$$f(\langle y, x \rangle) = \begin{cases} 1 & \text{if } \mathcal{M}_y \text{ halts on } x \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Growing rate and the halting problem

Theorem (Busy Beaver)

The Turing machine can not compute functions with arbitrary growing rate. There is a well-known established limit of growing rate for computable functions.

Theorem (The halting problem)

The halting function, namely the function

$$f(\langle y, x\rangle) = \begin{cases} 1 & \text{if } \mathcal{M}_y \text{ halts on } x \\ 0 & \text{otherwise} \end{cases}$$

is not computable.

Scooping The Loop Snooper by Geoffrey K. Pullum

No general procedure for bug checks will do.
Now, I won't just assert that, I'll prove it to you.
I will prove that although you might work till you drop,
you cannot tell if computation will stop.

For imagine we have a procedure called P
that for specified input permits you to see
whether specified source code, with all of its faults,
defines a routine that eventually halts.

You feed in your program, with suitable data,
and P gets to work, and a little while later
(in finite compute time) correctly infers
whether infinite looping behavior occurs.

.
.
.

Scooping The Loop Snooper by Geoffrey K. Pullum

No general procedure for bug checks will do.
Now, I won't just assert that, I'll prove it to you.
I will prove that although you might work till you drop,
you cannot tell if computation will stop.

For imagine we have a procedure called P
that for specified input permits you to see
whether specified source code, with all of its faults,
defines a routine that eventually halts.

You feed in your program, with suitable data,
and P gets to work, and a little while later
(in finite compute time) correctly infers
whether infinite looping behavior occurs.

.
.
.

Scooping The Loop Snooper by Geoffrey K. Pullum

No general procedure for bug checks will do.
Now, I won't just assert that, I'll prove it to you.
I will prove that although you might work till you drop,
you cannot tell if computation will stop.

For imagine we have a procedure called P
that for specified input permits you to see
whether specified source code, with all of its faults,
defines a routine that eventually halts.

You feed in your program, with suitable data,
and P gets to work, and a little while later
(in finite compute time) correctly infers
whether infinite looping behavior occurs.

.
.
.

Scooping The Loop Snooper by Geoffrey K. Pullum

No general procedure for bug checks will do.
Now, I won't just assert that, I'll prove it to you.
I will prove that although you might work till you drop,
you cannot tell if computation will stop.

For imagine we have a procedure called P
that for specified input permits you to see
whether specified source code, with all of its faults,
defines a routine that eventually halts.

You feed in your program, with suitable data,
and P gets to work, and a little while later
(in finite compute time) correctly infers
whether infinite looping behavior occurs.

Scooping The Loop Snooper by Geoffrey K. Pullum

No general procedure for bug checks will do.
Now, I won't just assert that, I'll prove it to you.
I will prove that although you might work till you drop,
you cannot tell if computation will stop.

For imagine we have a procedure called P
that for specified input permits you to see
whether specified source code, with all of its faults,
defines a routine that eventually halts.

You feed in your program, with suitable data,
and P gets to work, and a little while later
(in finite compute time) correctly infers
whether infinite looping behavior occurs.

.
.
.

Halting problem

.
.
.

I've created a paradox, neat as can be?
and simply by using your putative P.
When you posited P you stepped into a snare;
Your assumption has led you right into my lair.

So where can this argument possibly go?
I don't have to tell you; I'm sure you must know.
A reductio: There cannot possibly be
a procedure that acts like the mythical P.

You can never find general mechanical means
for predicting the acts of computing machines;
it's something that cannot be done. So we users
must find our own bugs. Our computers are losers!

Halting problem

.
.
.

I've created a paradox, neat as can be?
and simply by using your putative P.
When you posited P you stepped into a snare;
Your assumption has led you right into my lair.

So where can this argument possibly go?
I don't have to tell you; I'm sure you must know.
A reductio: There cannot possibly be
a procedure that acts like the mythical P.

You can never find general mechanical means
for predicting the acts of computing machines;
it's something that cannot be done. So we users
must find our own bugs. Our computers are losers!

Halting problem

.
.
.

I've created a paradox, neat as can be?
and simply by using your putative P.
When you posited P you stepped into a snare;
Your assumption has led you right into my lair.

So where can this argument possibly go?
I don't have to tell you; I'm sure you must know.
A reductio: There cannot possibly be
a procedure that acts like the mythical P.

You can never find general mechanical means
for predicting the acts of computing machines;
it's something that cannot be done. So we users
must find our own bugs. Our computers are losers!

Halting problem

.
.
.

I've created a paradox, neat as can be?
and simply by using your putative P.
When you posited P you stepped into a snare;
Your assumption has led you right into my lair.

So where can this argument possibly go?
I don't have to tell you; I'm sure you must know.
A reductio: There cannot possibly be
a procedure that acts like the mythical P.

You can never find general mechanical means
for predicting the acts of computing machines;
it's something that cannot be done. So we users
must find our own bugs. Our computers are losers!

Collatz function yet again

Example (Iterating the Collatz function)

input n;

while $n \neq 1$ do if $even(n)$ then $n := n/2$ else $n = 3n + 1$

Example (Sequences of numbers produced for different inputs)

4, 2, 1 HALT

5, 16, 8, 4, 2, 1 HALT

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1 HALT

Collatz function yet again

Example (Iterating the Collatz function)

input n;

while $n \neq 1$ do if $even(n)$ then $n := n/2$ else $n = 3n + 1$

Example (Sequences of numbers produced for different inputs)

4, 2, 1 HALT

5, 16, 8, 4, 2, 1 HALT

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1 HALT

Collatz function yet again

Example (Iterating the Collatz function)

input n;

while $n \neq 1$ do if $even(n)$ then $n := n/2$ else $n = 3n + 1$

Example (Sequences of numbers produced for different inputs)

4, 2, 1 HALT

5, 16, 8, 4, 2, 1 HALT

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1 HALT

Collatz function yet again

Example (Iterating the Collatz function)

input n;

while $n \neq 1$ do if $even(n)$ then $n := n/2$ else $n = 3n + 1$

Example (Sequences of numbers produced for different inputs)

4, 2, 1 HALT

5, 16, 8, 4, 2, 1 HALT

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1 HALT

Collatz function yet again

Example (Iterating the Collatz function)

input n;

while $n \neq 1$ do if $even(n)$ then $n := n/2$ else $n = 3n + 1$

Example (Sequences of numbers produced for different inputs)

4, 2, 1 HALT

5, 16, 8, 4, 2, 1 HALT

7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1 HALT

Proving Fermat's last theorem

Exercise from USA exam: Fermat's last theorem

Until recently one of the most famous unproved statements in mathematics, asserts that there are no integer solutions (x, y, z, n) to the equation $x^n + y^n = z^n$ satisfying $x, y > 0$ and $n > 2$. Show how a solution to the halting problem would allow you to determine whether the statement is true or false.

Proof:

We can construct a Turing Machine T that enumerates 4-tuples (x, y, z, n) of positive integers. After each 4-tuple is enumerated, the machine T checks if $x^n + y^n = z^n$ and checks that $n > 2$. If both checks succeed then the machine halts, otherwise it moves on to the next 4-tuple. Furthermore, since there infinitely many 4-tuples of integers, machine T halts if and only if there exists a 4-tuple (with $n > 2$) such that $x^n + y^n = z^n$. In other words, T halts if and only if Fermat's Last Theorem is false. If we had a solution T_H to the halting problem, then we could feed T to T_H and get the output. If the output was "1" then T halts and Fermat's Last Theorem is false. If the output was "0" then T doesnt halt, and Fermat's Last Theorem is true.

Proving Fermat's last theorem

Exercise from USA exam: Fermat's last theorem

Until recently one of the most famous unproved statements in mathematics, asserts that there are no integer solutions (x, y, z, n) to the equation $x^n + y^n = z^n$ satisfying $x, y > 0$ and $n > 2$. Show how a solution to the halting problem would allow you to determine whether the statement is true or false.

Proof:

We can construct a Turing Machine T that enumerates 4-tuples (x, y, z, n) of positive integers. After each 4-tuple is enumerated, the machine T checks if $x^n + y^n = z^n$ and checks that $n > 2$. If both checks succeed then the machine halts, otherwise it moves on to the next 4-tuple. Furthermore, since there infinitely many 4-tuples of integers, machine T halts if and only if there exists a 4-tuple (with $n > 2$) such that $x^n + y^n = z^n$. In other words, T halts if and only if Fermat's Last Theorem is false. If we had a solution T_H to the halting problem, then we could feed T to T_H and get the output. If the output was "1" then T halts and Fermat's Last Theorem is false. If the output was "0" then T doesnt halt, and Fermat's Last Theorem is true.

Decidable and semidecidable sets

Definition (Decidable set)

A set A (e.g., a subset of $\{0,1\}^\star$) is said to be decidable if the (characteristic) function (of A)

$$\xi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

is Turing computable.

Definition (Semidecidable set)

A set A (e.g., a subset of $\{0,1\}^\star$) is said to be semidecidable if the (partial characteristic) function (of A)

$$\xi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ \bot & \text{if } x \notin A \end{cases}$$

is Turing computable.

Decidable and semidecidable sets

Definition (Decidable set)

A set A (e.g., a subset of $\{0,1\}^\star$) is said to be decidable if the (characteristic) function (of A)

$$\xi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases}$$

is Turing computable.

Definition (Semidecidable set)

A set A (e.g., a subset of $\{0,1\}^\star$) is said to be semidecidable if the (partial characteristic) function (of A)

$$\not\xi_A(x) = \begin{cases} 1 & \text{if } x \in A \\ \bot & \text{if } x \notin A \end{cases}$$

is Turing computable.

Church-Turing Thesis I

Church-Turing Thesis

Postulate

No reasonable definition of algorithm produces more computable functions than the Turing machine.

Evidences

- The Turing machine mimics the activity of a computor.
- No one ever specified an intuitively algorithmic function that escapes to the computability criterion of the Turing machine.
- Different concepts of abstract device produce exactly the same class of functions.

Church-Turing Thesis

Postulate

No reasonable definition of algorithm produces more computable functions than the Turing machine.

Evidences

- The Turing machine mimics the activity of a computor.
- No one ever specified an intuitively algorithmic function that escapes to the computability criterion of the Turing machine.
- Different concepts of abstract device produce exactly the same class of functions.

Church-Turing Thesis

Postulate

No reasonable definition of algorithm produces more computable functions than the Turing machine.

Evidences

- The Turing machine mimics the activity of a computor.
- No one ever specified an intuitively algorithmic function that escapes to the computability criterion of the Turing machine.
- Different concepts of abstract device produce exactly the same class of functions.

Church-Turing Thesis

Postulate

No reasonable definition of algorithm produces more computable functions than the Turing machine.

Evidences

- The Turing machine mimics the activity of a computor.
- No one ever specified an intuitively algorithmic function that escapes to the computability criterion of the Turing machine.
- Different concepts of abstract device produce exactly the same class of functions.

Church-Turing Thesis

Postulate

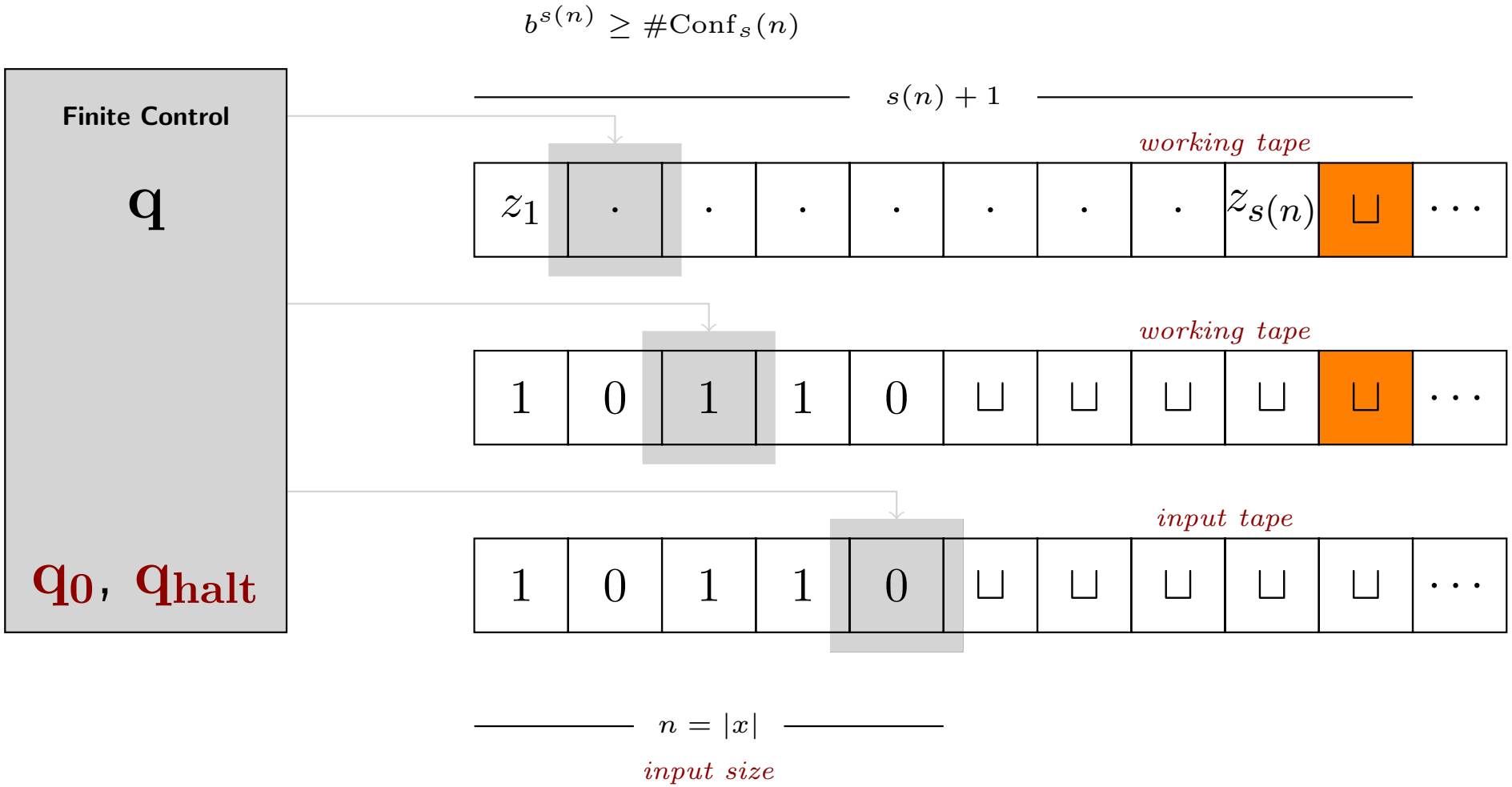
No reasonable definition of algorithm produces more computable functions than the Turing machine.

Evidences

- The Turing machine mimics the activity of a computor.
- No one ever specified an intuitively algorithmic function that escapes to the computability criterion of the Turing machine.
- Different concepts of abstract device produce exactly the same class of functions.

Bounded Space

Bounded Space



Bounded Space

Theorem

The halting problem of Turing machines bounded in space is decidable.

Proof:

$$\begin{aligned} \#\mathrm{Conf}_s(n) &= |Q| \times 3^{s(n)} \times s(n) \times n \\ &\in O(2^{O(s(n)} \times n) \\ &= 2^{O(s(n))} \\ &= b^{s(n)} \end{aligned}$$

Bounded Space

Theorem

The halting problem of Turing machines bounded in space is decidable.

Proof:

$$\begin{aligned} \#\mathrm{Conf}_s(n) &= |Q| \times 3^{s(n)} \times s(n) \times n \\ &\in O(2^{O(s(n)} \times n) \\ &= 2^{O(s(n))} \\ &= b^{s(n)} \end{aligned}$$

Acceleration

Accelerated Turing machine

Máquina de Zenão (*vide* Copeland in [1, 2])

Engenho físico capaz de executar a computação de uma máquina de Turing de tal modo que dispende 1 segundo na primeira transição, $1/2$ do segundo na segunda transição, $1/4$ do segundo na terceira transição e assim sucessivamente, de modo a que, na n-ésima transição, gasta a fração $1/2^{n-1}$ do segundo.

Equação dos tempos

A equação dos tempos para a máquina de Turing acelerada é, para $n \geq 1$, dada por:

$$n \text{ transições} \quad = \quad 2 - \frac{1}{2^{n-1}} \text{ segundos.}$$

Accelerated Turing machine

Máquina de Zenão (*vide* Copeland in [1, 2])

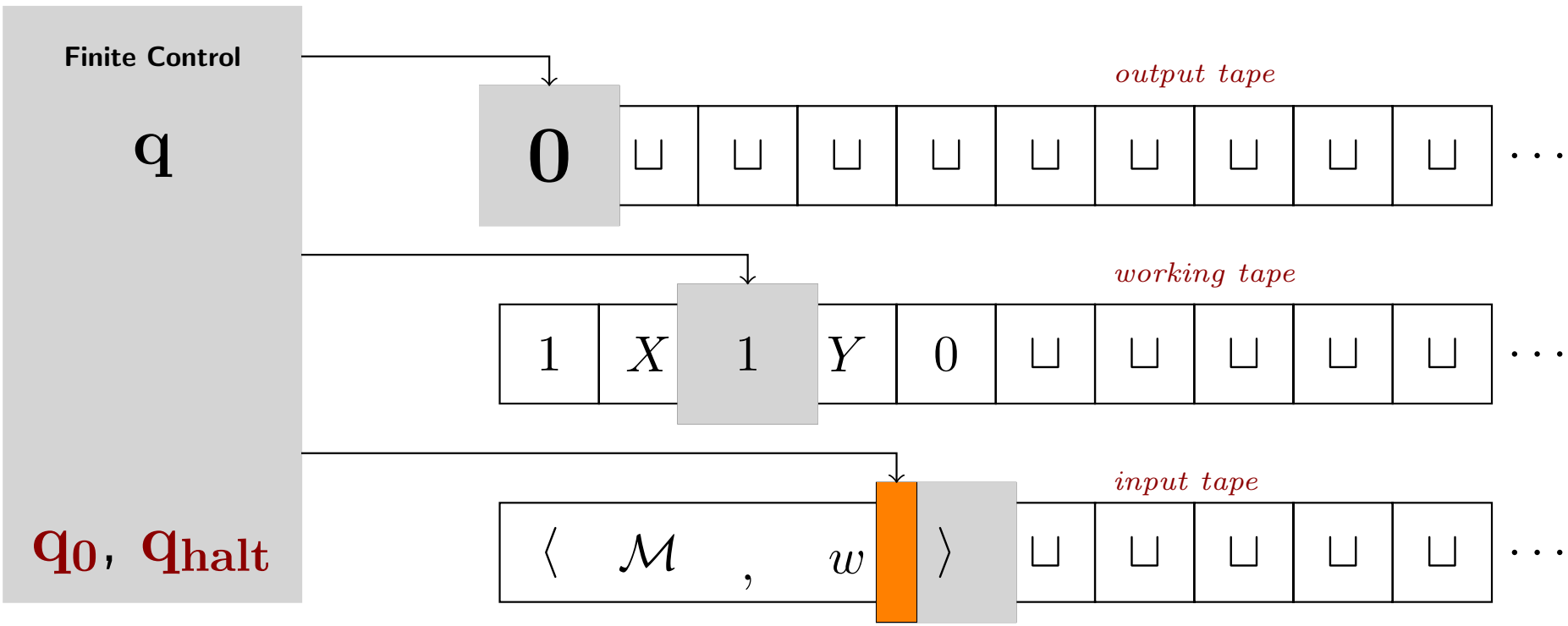
Engenho físico capaz de executar a computação de uma máquina de Turing de tal modo que dispende 1 segundo na primeira transição, $1/2$ do segundo na segunda transição, $1/4$ do segundo na terceira transição e assim sucessivamente, de modo a que, na n-ésima transição, gasta a fração $1/2^{n-1}$ do segundo.

Equação dos tempos

A equação dos tempos para a máquina de Turing acelerada é, para $n \geq 1$, dada por:

$$n \text{ transições} \quad = \quad 2 - \frac{1}{2^{n-1}} \text{ segundos.}$$

Accelerated Turing machine



Drawbacks of the accelerating Turing machine

Problema da configuração

Pode calcular-se a configuração da máquina no instante $1 - (1/2)^n$, para todo o valor de n $(0, 1, 2, 3, \dots)$, mas não pode dizer-se qual é a configuração da máquina exatamente no instante $t = 2$ segundos.

Problema da equivalência

A máquina acelerada resolve exatamente o mesmo problema que a máquina de Turing $\mathcal{M}$, ao simulá-la: se $\mathcal{M}$ para para o *input* w, a máquina acelerada dá resultado 1, mas se a máquina $\mathcal{M}$ não para para o *input* w, então a máquina acelerada não dá, de facto, solução, pois a solução 0 foi nela instalada no início.

Problema do critério de aceitação

O problema, considerado pelos filósofos da ciência, tais com Oron Shagrir (*vide* [Sha12]), prendem-se com *a incapacidade de a máquina acelerada transitar para o comum estado de aceitação*, isto é, não se consegue definir a máquina acelerada de modo a que, aos 2 segundos de funcionamento, a máquina se encontre no estado de aceitação ou de rejeição, nomeadamente no caso das computações infinitas.

Drawbacks of the accelerating Turing machine

Problema da configuração

Pode calcular-se a configuração da máquina no instante $1 - (1/2)^n$, para todo o valor de n $(0, 1, 2, 3, \dots)$, mas não pode dizer-se qual é a configuração da máquina exatamente no instante $t = 2$ segundos.

Problema da equivalência

A máquina acelerada resolve exatamente o mesmo problema que a máquina de Turing $\mathcal{M}$, ao simulá-la: se $\mathcal{M}$ para para o *input* w, a máquina acelerada dá resultado 1, mas se a máquina $\mathcal{M}$ não para para o *input* w, então a máquina acelerada não dá, de facto, solução, pois a solução 0 foi nela instalada no início.

Problema do critério de aceitação

O problema, considerado pelos filósofos da ciência, tais com Oron Shagrir (*vide* [Sha12]), prendem-se com *a incapacidade de a máquina acelerada transitar para o comum estado de aceitação*, isto é, não se consegue definir a máquina acelerada de modo a que, aos 2 segundos de funcionamento, a máquina se encontre no estado de aceitação ou de rejeição, nomeadamente no caso das computações infinitas.

Drawbacks of the accelerating Turing machine

Problema da configuração

Pode calcular-se a configuração da máquina no instante $1 - (1/2)^n$, para todo o valor de n $(0, 1, 2, 3, \dots)$, mas não pode dizer-se qual é a configuração da máquina exatamente no instante $t = 2$ segundos.

Problema da equivalência

A máquina acelerada resolve exatamente o mesmo problema que a máquina de Turing $\mathcal{M}$, ao simulá-la: se $\mathcal{M}$ para para o *input* w, a máquina acelerada dá resultado 1, mas se a máquina $\mathcal{M}$ não para para o *input* w, então a máquina acelerada não dá, de facto, solução, pois a solução 0 foi nela instalada no início.

Problema do critério de aceitação

O problema, considerado pelos filósofos da ciência, tais com Oron Shagrir (*vide* [Sha12]), prendem-se com *a incapacidade de a máquina acelerada transitar para o comum estado de aceitação*, isto é, não se consegue definir a máquina acelerada de modo a que, aos 2 segundos de funcionamento, a máquina se encontre no estado de aceitação ou de rejeição, nomeadamente no caso das computações infinitas.

Drawbacks of the accelerating Turing machine

Problema da configuração

Pode calcular-se a configuração da máquina no instante $1 - (1/2)^n$, para todo o valor de n $(0, 1, 2, 3, \dots)$, mas não pode dizer-se qual é a configuração da máquina exatamente no instante $t = 2$ segundos.

Problema da equivalência

A máquina acelerada resolve exatamente o mesmo problema que a máquina de Turing $\mathcal{M}$, ao simulá-la: se $\mathcal{M}$ para para o *input* w, a máquina acelerada dá resultado 1, mas se a máquina $\mathcal{M}$ não para para o *input* w, então a máquina acelerada não dá, de facto, solução, pois a solução 0 foi nela instalada no início.

Problema do critério de aceitação

O problema, considerado pelos filósofos da ciência, tais com Oron Shagrir (*vide* [Sha12]), prendem-se com *a incapacidade de a máquina acelerada transitar para o comum estado de aceitação*, isto é, não se consegue definir a máquina acelerada de modo a que, aos 2 segundos de funcionamento, a máquina se encontre no estado de aceitação ou de rejeição, nomeadamente no caso das computações infinitas.

Diophantine Sets

Diophantine sets

Hilbert's tenth problem is the tenth on the list of Hilbert's problems of 1900. Its statement is as follows:

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

Diophantine sets

Definition (Diophantine sets)

We say that a relation D is Diophantine if there exists a polynomial p with integer coefficients, such that,

$$\langle m_1, \ldots, m_k \rangle \in D \text{ iff } \exists x_1, \ldots, x_n \in \mathbb{N}_1 \ [\ p(m_1, \ldots, m_k, x_1, \ldots, x_n) = 0\ ]$$

Example (Composite numbers, divisible numbers, prime numbers, etc.)

Diophantine sets

Definition (Diophantine sets)

We say that a relation D is Diophantine if there exists a polynomial p with integer coefficients, such that,

$$\langle m_1, \ldots, m_k \rangle \in D \text{ iff } \exists x_1, \ldots, x_n \in \mathbb{N}_1 \ [\ p(m_1, \ldots, m_k, x_1, \ldots, x_n) = 0\ ]$$

Example (Composite numbers, divisible numbers, prime numbers, etc.)

Diophantine sets

Definition (Diophantine sets)

We say that a relation D is Diophantine if there exists a polynomial p with integer coefficients, such that,

$$\langle m_1, \ldots, m_k \rangle \in D \text{ iff } \exists x_1, \ldots, x_n \in \mathbb{N}_1 \ [\ p(m_1, \ldots, m_k, x_1, \ldots, x_n) = 0\ ]$$

Example (Composite numbers, divisible numbers, prime numbers, etc.)

$$x \in \mathit{Composite} \quad \text{iff} \quad \exists y, z \in \mathbb{N}_1 \ [\ (y+1)(z+1) - x = 0\ ]$$

Diophantine sets

Definition (Diophantine sets)

We say that a relation D is Diophantine if there exists a polynomial p with integer coefficients, such that,

$$\langle m_1, \ldots, m_k \rangle \in D \text{ iff } \exists x_1, \ldots, x_n \in \mathbb{N}_1 \ [\ p(m_1, \ldots, m_k, x_1, \ldots, x_n) = 0\ ]$$

Example (Composite numbers, divisible numbers, prime numbers, etc.)

$$x|y \quad \text{iff} \quad \exists z \in \mathbb{N}_1 \ [\ xz - y = 0\ ]$$

Diophantine sets

Definition (Diophantine sets)

We say that a relation D is Diophantine if there exists a polynomial p with integer coefficients, such that,

$$\langle m_1, \ldots, m_k \rangle \in D \text{ iff } \exists x_1, \ldots, x_n \in \mathbb{N}_1 \ [\ p(m_1, \ldots, m_k, x_1, \ldots, x_n) = 0\ ]$$

Example (Composite numbers, divisible numbers, prime numbers, etc.)

$$x|y \ and \ x < y \quad \text{iff} \quad \exists u, v \in \mathbb{N}_1 \ [\ (xu - y)^2 + (y - x - v)^2 = 0\ ]$$

Diophantine sets

Definition (Diophantine sets)

We say that a relation D is Diophantine if there exists a polynomial p with integer coefficients, such that,

$$\langle m_1, \ldots, m_k \rangle \in D \text{ iff } \exists x_1, \ldots, x_n \in \mathbb{N}_1 \ [\ p(m_1, \ldots, m_k, x_1, \ldots, x_n) = 0\ ]$$

Example (Composite numbers, divisible numbers, prime numbers, etc.)

$$x \ is \ not \ a \ power \ of \ 2 \quad \text{iff} \quad \exists y, z \in \mathbb{N}_1 \ [\ x - y(2z+1) = 0\ ]$$

Diophantine sets

Definition (Diophantine sets)

We say that a relation D is Diophantine if there exists a polynomial p with integer coefficients, such that,

$$\langle m_1, \ldots, m_k \rangle \in D \text{ iff } \exists x_1, \ldots, x_n \in \mathbb{N}_1 \ [\ p(m_1, \ldots, m_k, x_1, \ldots, x_n) = 0\ ]$$

Example (Composite numbers, divisible numbers, prime numbers, etc.)

$$k + 2 \ \textit{is prime} \quad \text{iff} \quad \ldots$$

Primes

$$\exists a\ \exists b\ \exists c\ \exists d\ \exists e\ \exists f\ \exists g\ \exists h\ \exists i\ \exists j\ \exists \ell\ \exists m\ \exists n\ \exists o\ \exists p\ \exists q\ \exists r\ \exists s\ \exists t\ \exists u\ \exists v\ \exists x\ \exists w\ \exists y\ \exists z$$

$$\begin{aligned}
& [wz + h + j - q]^2 \\
+ \quad & [(g\mathbf{k} + 2g + \mathbf{k} + 1)(h + j) + h - z]^2 \\
+ \quad & [16(\mathbf{k} + 1)^3(\mathbf{k} + 2)(n + 1)^2 + 1 - f^2]^2 \\
+ \quad & [2n + p + q + z - e]^2 \\
+ \quad & [e^3(e + 2)(a + 1)^2 + 1 - o^2]^2 \\
+ \quad & [(a^2 - 1)y^2 + 1 - x^2]^2 \\
+ \quad & [16r^2y^4(a^2 - 1) + 1 - u^2]^2 \\
+ \quad & [n + \ell + v - y]^2 \\
+ \quad & [(a^2 - 1)\ell^2 + 1 - m^2]^2 \\
+ \quad & [ai + \mathbf{k} + 1 - \ell - i]^2 \\
+ \quad & [((a + u^2(u^2 - a))^2 - 1)(n + 4dy)^2 + 1 - (x + cu)^2]^2 \\
+ \quad & [p + \ell(a - n - 1) + b(2an + 2a - n^2 - 2n - 2) - m]^2 \\
+ \quad & [q + y(a - p - 1) + s(2ap + 2a - p^2 - 2p - 2) - x]^2 \\
+ \quad & [z + p\ell(a - p) + t(2ap - p^2 - 1) - pm]^2) \\
= \quad & 0
\end{aligned}$$

Lagrange

Anotações I

Para especificar um conjunto diofantino podemos usar um sistema de equações simultâneas, pois

$$P_1(\vec{x}) = 0, P_2(\vec{x}) = 0, \ldots, P_k(\vec{x}) = 0 \quad \text{se e só se} \quad P_1(\vec{x})^2 + P_2(\vec{x})^2 + \cdots + P_k(\vec{x})^2 = 0 \ .$$

Anotações II

A equação $P(x_1, \ldots, x_n) = 0$ tem soluções positivas $x_1, \ldots, x_n$ se e só se a equação seguinte tem soluções inteiras $p_1, q_1, r_1, s_1, \ldots, p_n, q_n, r_n, s_n$:

$$P(1 + p_1^2 + q_1^2 + r_1^2 + s_1^2, \ldots, 1 + p_n^2 + q_n^2 + r_n^2 + s_n^2) = 0$$

Anotações III

Uma função n-ária diz-se diofantina se for diofantino o conjunto assim definido:

$$\{\langle x_1, \cdots, x_n, y\rangle : y = f(x_1, \ldots, x_n)\}$$

Lagrange

Anotações I

Para especificar um conjunto diofantino podemos usar um sistema de equações simultâneas, pois

$$P_1(\vec{x}) = 0, P_2(\vec{x}) = 0, \ldots, P_k(\vec{x}) = 0 \quad \text{se e só se} \quad P_1(\vec{x})^2 + P_2(\vec{x})^2 + \cdots + P_k(\vec{x})^2 = 0 \ .$$

Anotações II

A equação $P(x_1, \ldots, x_n) = 0$ tem soluções positivas $x_1, \ldots, x_n$ se e só se a equação seguinte tem soluções inteiras $p_1, q_1, r_1, s_1, \ldots, p_n, q_n, r_n, s_n$:

$$P(1 + p_1^2 + q_1^2 + r_1^2 + s_1^2, \ldots, 1 + p_n^2 + q_n^2 + r_n^2 + s_n^2) = 0$$

Anotações III

Uma função n-ária diz-se diofantina se for diofantino o conjunto assim definido:

$$\{\langle x_1, \cdots, x_n, y\rangle : y = f(x_1, \ldots, x_n)\}$$

Lagrange

Anotações I

Para especificar um conjunto diofantino podemos usar um sistema de equações simultâneas, pois

$$P_1(\vec{x}) = 0, P_2(\vec{x}) = 0, \ldots, P_k(\vec{x}) = 0 \quad \text{se e só se} \quad P_1(\vec{x})^2 + P_2(\vec{x})^2 + \cdots + P_k(\vec{x})^2 = 0 \ .$$

Anotações II

A equação $P(x_1, \ldots, x_n) = 0$ tem soluções positivas $x_1, \ldots, x_n$ se e só se a equação seguinte tem soluções inteiras $p_1, q_1, r_1, s_1, \ldots, p_n, q_n, r_n, s_n$:

$$P(1 + p_1^2 + q_1^2 + r_1^2 + s_1^2, \ldots, 1 + p_n^2 + q_n^2 + r_n^2 + s_n^2) = 0$$

Anotações III

Uma função n-ária diz-se diofantina se for diofantino o conjunto assim definido:

$$\{\langle x_1, \cdots, x_n, y\rangle : y = f(x_1, \ldots, x_n)\}$$

Lagrange

Anotações I

Para especificar um conjunto diofantino podemos usar um sistema de equações simultâneas, pois

$$P_1(\vec{x}) = 0, P_2(\vec{x}) = 0, \ldots, P_k(\vec{x}) = 0 \quad \text{se e só se} \quad P_1(\vec{x})^2 + P_2(\vec{x})^2 + \cdots + P_k(\vec{x})^2 = 0 \ .$$

Anotações II

A equação $P(x_1, \ldots, x_n) = 0$ tem soluções positivas $x_1, \ldots, x_n$ se e só se a equação seguinte tem soluções inteiras $p_1, q_1, r_1, s_1, \ldots, p_n, q_n, r_n, s_n$:

$$P(1 + p_1^2 + q_1^2 + r_1^2 + s_1^2, \ldots, 1 + p_n^2 + q_n^2 + r_n^2 + s_n^2) = 0$$

Anotações III

Uma função n-ária diz-se diofantina se for diofantino o conjunto assim definido:

$$\{\langle x_1, \cdots, x_n, y\rangle : y = f(x_1, \ldots, x_n)\}$$

Positive range of polynomials

Teorema

Um conjunto S de inteiros positivos é diofantino se e só se existe um polinómio P tal que S é precisamente o conjunto dos inteiros positivos no contradomínio de P.

Proof:

$$Q(x, x_1, \ldots, x_n) = P(x_1, \ldots, x_n) - x$$

$x \in S$ se e só se $\exists x_1 \ldots \exists x_n \, [x = P(x_1, \ldots, x_n)]$ se e só se $\exists x_1 \ldots \exists x_n \, Q(x, x_1, \ldots, x_n) = 0$

$$P(x, x_1, \ldots, \ldots, x_n) = x[1 - Q(x, x_1, \ldots, x_n)^2]$$

Positive range of polynomials

Teorema

Um conjunto S de inteiros positivos é diofantino se e só se existe um polinómio P tal que S é precisamente o conjunto dos inteiros positivos no contradomínio de P.

Proof:

$$Q(x, x_1, \ldots, x_n) = P(x_1, \ldots, x_n) - x$$

$x \in S$ se e só se $\exists x_1 \ldots \exists x_n \, [x = P(x_1, \ldots, x_n)]$ se e só se $\exists x_1 \ldots \exists x_n \, Q(x, x_1, \ldots, x_n) = 0$

$$P(x, x_1, \ldots, \ldots, x_n) = x[1 - Q(x, x_1, \ldots, x_n)^2]$$

4 degrees are enough ([V.M93])

$$4x^3y + 5z = 2x^2z^3 + 3y^2x$$

$$p_1 = 4x \qquad p_2 = p_1x \qquad p_3 = p_2x \qquad p_4 = p_3y$$

$$q_1 = 5z$$

$$r_1 = 2x \qquad r_2 = r_1x \qquad r_3 = r_2z \qquad r_4 = r_3z \qquad r_5 = r_4z$$

$$s_1 = 3y \qquad s_2 = s_1y \qquad s_3 = s_2x$$

$$t_1 = p_4 + q_1 \qquad u_1 = r_5 + s_3 \qquad t_1 = u_1$$

Theorem

To solve Hilbert's Tenth Problem positively, it is sufficient to find a method for deciding whether a Diophantine equation of degree 4 has a solution.

4 degrees are enough ([V.M93])

$$4x^3y + 5z = 2x^2z^3 + 3y^2x$$

$$p_1 = 4x \qquad p_2 = p_1x \qquad p_3 = p_2x \qquad p_4 = p_3y$$

$$q_1 = 5z$$

$$r_1 = 2x \qquad r_2 = r_1x \qquad r_3 = r_2z \qquad r_4 = r_3z \qquad r_5 = r_4z$$

$$s_1 = 3y \qquad s_2 = s_1y \qquad s_3 = s_2x$$

$$t_1 = p_4 + q_1 \qquad u_1 = r_5 + s_3 \qquad t_1 = u_1$$

Theorem

To solve Hilbert's Tenth Problem positively, it is sufficient to find a method for deciding whether a Diophantine equation of degree 4 has a solution.

4 degrees are enough ([V.M93])

$$4x^3y + 5z = 2x^2z^3 + 3y^2x$$

$$p_1 = 4x \qquad p_2 = p_1x \qquad p_3 = p_2x \qquad p_4 = p_3y$$

$$q_1 = 5z$$

$$r_1 = 2x \qquad r_2 = r_1x \qquad r_3 = r_2z \qquad r_4 = r_3z \qquad r_5 = r_4z$$

$$s_1 = 3y \qquad s_2 = s_1y \qquad s_3 = s_2x$$

$$t_1 = p_4 + q_1 \qquad u_1 = r_5 + s_3 \qquad t_1 = u_1$$

Theorem

To solve Hilbert's Tenth Problem positively, it is sufficient to find a method for deciding whether a Diophantine equation of degree 4 has a solution.

4 degrees are enough ([V.M93])

$$4x^3y + 5z = 2x^2z^3 + 3y^2x$$

$$p_1 = 4x \qquad p_2 = p_1x \qquad p_3 = p_2x \qquad p_4 = p_3y$$

$$q_1 = 5z$$

$$r_1 = 2x \qquad r_2 = r_1x \qquad r_3 = r_2z \qquad r_4 = r_3z \qquad r_5 = r_4z$$

$$s_1 = 3y \qquad s_2 = s_1y \qquad s_3 = s_2x$$

$$t_1 = p_4 + q_1 \qquad u_1 = r_5 + s_3 \qquad t_1 = u_1$$

Theorem

To solve Hilbert's Tenth Problem positively, it is sufficient to find a method for deciding whether a Diophantine equation of degree 4 has a solution.

4 degrees are enough ([V.M93])

$$4x^3y + 5z = 2x^2z^3 + 3y^2x$$

$$p_1 = 4x \qquad p_2 = p_1x \qquad p_3 = p_2x \qquad p_4 = p_3y$$

$$q_1 = 5z$$

$$r_1 = 2x \qquad r_2 = r_1x \qquad r_3 = r_2z \qquad r_4 = r_3z \qquad r_5 = r_4z$$

$$s_1 = 3y \qquad s_2 = s_1y \qquad s_3 = s_2x$$

$$t_1 = p_4 + q_1 \qquad u_1 = r_5 + s_3 \qquad t_1 = u_1$$

Theorem

To solve Hilbert's Tenth Problem positively, it is sufficient to find a method for deciding whether a Diophantine equation of degree 4 has a solution.

4 degrees are enough ([V.M93])

$$4x^3y + 5z = 2x^2z^3 + 3y^2x$$

$$p_1 = 4x \qquad p_2 = p_1x \qquad p_3 = p_2x \qquad p_4 = p_3y$$

$$q_1 = 5z$$

$$r_1 = 2x \qquad r_2 = r_1x \qquad r_3 = r_2z \qquad r_4 = r_3z \qquad r_5 = r_4z$$

$$s_1 = 3y \qquad s_2 = s_1y \qquad s_3 = s_2x$$

$$t_1 = p_4 + q_1 \qquad u_1 = r_5 + s_3 \qquad t_1 = u_1$$

Theorem

To solve Hilbert's Tenth Problem positively, it is sufficient to find a method for deciding whether a Diophantine equation of degree 4 has a solution.

4 degrees are enough ([V.M93])

$$4x^3y + 5z = 2x^2z^3 + 3y^2x$$

$$p_1 = 4x \qquad p_2 = p_1x \qquad p_3 = p_2x \qquad p_4 = p_3y$$

$$q_1 = 5z$$

$$r_1 = 2x \qquad r_2 = r_1x \qquad r_3 = r_2z \qquad r_4 = r_3z \qquad r_5 = r_4z$$

$$s_1 = 3y \qquad s_2 = s_1y \qquad s_3 = s_2x$$

$$t_1 = p_4 + q_1 \qquad u_1 = r_5 + s_3 \qquad t_1 = u_1$$

Theorem

To solve Hilbert's Tenth Problem positively, it is sufficient to find a method for deciding whether a Diophantine equation of degree 4 has a solution.

RDP conjecture

Conjetura RDP

A função exponencial (de expressão x^y) é diofantina.

Teorema (Yuri Matiyasevich, January 1970 ([V.M93]))

O conjunto dos números de Fibonacci é diofantino,
ou (alternativamente)...
$m = n^k$ *se e só se as equações numeradas de I a XIII têm solução nas demais variáveis.*

RDP conjecture

Conjetura RDP

A função exponencial (de expressão x^y) é diofantina.

Teorema (Yuri Matiyasevich, January 1970 ([V.M93]))

O conjunto dos números de Fibonacci é diofantino,
ou (alternativamente)...
$m = n^k$ *se e só se as equações numeradas de I a XIII têm solução nas demais variáveis.*

RDP conjecture

I $x^2 - (a^2 - 1)y^2 = 1$

II $u^2 - (a^2 - 1)v^2 = 1$

III $s^2 - (b^2 - 1)t^2 = 1$

IV $v = ry^2$

V $b = 1 + 4py = a + qu$

VI $s = x + cu$

VII $t = \mathbf{k} + 4(d - 1)y$

VIII $y = \mathbf{k} + \ell - 1$

IX $a = z + 1$

X $(x - y(a - \mathbf{n}) - \mathbf{m})^2 = (f - 1)^2(2a\mathbf{n} - \mathbf{n}^2 - 1)^2$

XI $\mathbf{m} + g = 2a\mathbf{n} - \mathbf{n}^2 - 1$

XII $w = \mathbf{n} + h = \mathbf{k} + \ell$

XIII $a^2 - (w^2 - 1)(w - 1)^2 z^2 = 1$

MRDP theorem

Theorem (Matiyasevich, Robinson, Davis, Putnam ([Dav73, V.M93]))

A set is Diophantine if and only if it is semidecidable.

Sturm's Theorem

Sturm's Algorithm

Theorem

There is an algorithm to compute the number and an interval containing the zeros of a polynomial of integer coefficients.

Example

For the physicist, the undecidability of Hilbert's tenth problem may appear intriguing.

Sturm's Algorithm

Theorem

There is an algorithm to compute the number and an interval containing the zeros of a polynomial of integer coefficients.

Example

For the physicist, the undecidability of Hilbert's tenth problem may appear intriguing.

Sturm's Algorithm

Theorem

There is an algorithm to compute the number and an interval containing the zeros of a polynomial of integer coefficients.

Example

For the physicist, the undecidability of Hilbert's tenth problem may appear intriguing.

Sturm's Algorithm

Definition (Sturm's sequence of $p(n)$:)

1. $p_0 = p$;
2. $p_1 = p'$;
3. p_{i+1} é o simétrico do resto da divisão de p_{i-1} por p_i;
4. $SEQ[p(n)] = p_0(n), p_1(n), p_2(n), \ldots, p_m(n)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

Sturm's Algorithm

Definition (Sturm's sequence of $p(n)$:)

1. $p_0 = p$;
2. $p_1 = p'$;
3. p_{i+1} é o simétrico do resto da divisão de p_{i-1} por p_i;
4. $SEQ[p(n)] = p_0(n), p_1(n), p_2(n), \ldots, p_m(n)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

Sturm's Algorithm

Definition (Sturm's sequence of $p(n)$:)

1. $p_0 = p$;
2. $p_1 = p'$;
3. p_{i+1} é o simétrico do resto da divisão de p_{i-1} por p_i;
4. $SEQ[p(n)] = p_0(n), p_1(n), p_2(n), \ldots, p_m(n)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

$$p_0(n) = n^4 + n^3 - n - 1$$

Sturm's Algorithm

Definition (Sturm's sequence of $p(n)$:)

1. $p_0 = p$;
2. $p_1 = p'$;
3. p_{i+1} é o simétrico do resto da divisão de p_{i-1} por p_i;
4. $SEQ[p(n)] = p_0(n), p_1(n), p_2(n), \ldots, p_m(n)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

$$p_1(n) = 4n^3 + 3n^2 - 1$$

Sturm's Algorithm

Definition (Sturm's sequence of $p(n)$:)

1. $p_0 = p$;
2. $p_1 = p'$;
3. p_{i+1} é o simétrico do resto da divisão de p_{i-1} por p_i;
4. $SEQ[p(n)] = p_0(n), p_1(n), p_2(n), \ldots, p_m(n)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

$$
\begin{array}{r}
(\ n^4 + n^3 - n - 1) \div (4n^3 + 3n^2 - 1) = \frac{1}{4}n + \frac{1}{16} + \dfrac{-\frac{3}{16}n^2 - \frac{3}{4}n - \frac{15}{16}}{4n^3 + 3n^2 - 1} \\
\underline{-n^4 - \frac{3}{4}n^3 + \frac{1}{4}n} \\
\frac{1}{4}n^3 - \frac{3}{4}n - 1 \\
\underline{-\frac{1}{4}n^3 - \frac{3}{16}n^2 + \frac{1}{16}} \\
-\frac{3}{16}n^2 - \frac{3}{4}n - \frac{15}{16}
\end{array}
$$

Sturm's Algorithm

Definition (Sturm's sequence of $p(n)$:)

1. $p_0 = p$;
2. $p_1 = p'$;
3. p_{i+1} é o simétrico do resto da divisão de p_{i-1} por p_i;
4. $SEQ[p(n)] = p_0(n), p_1(n), p_2(n), \ldots, p_m(n)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

$$
\begin{array}{rl}
(\ 4n^3 + 3n^2 & -1) \div (\frac{3}{16}n^2 + \frac{3}{4}n + \frac{15}{16}) = \frac{64}{3}n - \frac{208}{3} + \frac{32n+64}{\frac{3}{16}n^2 + \frac{3}{4}n + \frac{15}{16}} \\
-4n^3 - 16n^2 - 20n & \\
\hline
-13n^2 - 20n & -1 \\
13n^2 + 52n + 65 & \\
\hline
32n + 64 &
\end{array}
$$

Sturm's Algorithm

Definition (Sturm's sequence of $p(n)$:)

1. $p_0 = p$;
2. $p_1 = p'$;
3. p_{i+1} é o simétrico do resto da divisão de p_{i-1} por p_i;
4. $SEQ[p(n)] = p_0(n), p_1(n), p_2(n), \ldots, p_m(n)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

$$
\begin{array}{l}
\left(\frac{3}{16}n^2 + \frac{3}{4}n + \frac{15}{16}\right) \div (-32n - 64) = -\frac{3}{512}n - \frac{3}{256} + \frac{\frac{3}{16}}{-32n - 64} \\
\underline{-\frac{3}{16}n^2 - \frac{3}{8}n} \\
\quad \frac{3}{8}n + \frac{15}{16} \\
\quad \underline{-\frac{3}{8}n - \frac{3}{4}} \\
\quad \frac{3}{16}
\end{array}
$$

Sturm's Algorithm

Definition (Sturm's sequence of $p(n)$:)

1. $p_0 = p$;
2. $p_1 = p'$;
3. p_{i+1} é o simétrico do resto da divisão de p_{i-1} por p_i;
4. $SEQ[p(n)] = p_0(n), p_1(n), p_2(n), \ldots, p_m(n)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

$$n^4 + n^3 - n - 1 \qquad 4n^3 + 3n^2 - 1 \qquad \frac{3}{16}n^2 + \frac{3}{4}n + \frac{15}{16} \qquad -32n - 64 \qquad -\frac{3}{16}$$

Sturm's Algorithm

Theorem (Cauchy)

As raízes do polinómio $p(n) = a_p n^p + \cdots + a_1 n + a_0$, *caso existam, pertencem ao intervalo* $[-M, M]$, *onde*

$$M = \frac{|a_{p-1}| + |a_{p-2}| + \cdots + |a_1| + |a_0|}{|a_p|} \ .$$

Example $(p(n) = n^4 + n^3 - n - 1)$

$$M = \frac{|+1| + |0| + |-1| + |-1|}{|+1|} = 3$$

Sturm's Algorithm

Theorem (Cauchy)

As raízes do polinómio $p(n) = a_p n^p + \cdots + a_1 n + a_0$, *caso existam, pertencem ao intervalo* $[-M, M]$, *onde*

$$M = \frac{|a_{p-1}| + |a_{p-2}| + \cdots + |a_1| + |a_0|}{|a_p|} \ .$$

Example $(p(n) = n^4 + n^3 - n - 1)$

$$M = \frac{|+1| + |0| + |-1| + |-1|}{|+1|} = 3$$

Sturm's Algorithm

Definition (Alternância de sinal num ponto $c \in \mathbb{Q}$)

$\delta(c)$ é o número de alternâncias de sinal ao longo da sequência $SEQ[p(c)] = p_0(c), p_1(c), p_2(c), \ldots, p_m(c)$, ignorando possíveis zeros.

Example ($p(n) = n^4 + n^3 - n - 1$ nos pontos -3 e 3)

$n^4 + n^3 - n - 1$	$4n^3 + 3n^2 - 1$	$\frac{3}{16}n^2 + \frac{3}{4}n + \frac{15}{16}$	$-32n - 64$	$-\frac{3}{16}$
$+$	$-$	$+$	$+$	$-$
$+$	$+$	$+$	$-$	$-$

Sturm's Algorithm

Definition (Alternância de sinal num ponto $c \in \mathbb{Q}$)

$\delta(c)$ é o número de alternâncias de sinal ao longo da sequência $SEQ[p(c)] = p_0(c), p_1(c), p_2(c), \ldots, p_m(c)$, ignorando possíveis zeros.

Example ($p(n) = n^4 + n^3 - n - 1$ nos pontos -3 e 3)

$n^4 + n^3 - n - 1$	$4n^3 + 3n^2 - 1$	$\frac{3}{16}n^2 + \frac{3}{4}n + \frac{15}{16}$	$-32n - 64$	$-\frac{3}{16}$
$+$	$-$	$+$	$+$	$-$
$+$	$+$	$+$	$-$	$-$

Sturm's Algorithm

Theorem (Sturm)

Seja $p(n)$ um polinómio de raízes de multiplicidade 1 . Se $a, b \in \mathbb{R}$ são tais que $a < b$ e $p(a), p(b) \neq 0$, então o número de zeros de $p(n)$ no intervalo $[a, b]$ é $\delta(a) - \delta(b)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

- $p(-3) = 56 \neq 0$ e $p(3) = 104 \neq 0$
- $\delta(-3) - \delta(3) = 3 - 1 = 2$
- As únicas raízes reais de $n^4 + n^3 - n - 1 = 0$ são -1 e $+1$.

Sturm's Algorithm

Theorem (Sturm)

Seja $p(n)$ um polinómio de raízes de multiplicidade 1 . Se $a, b \in \mathbb{R}$ são tais que $a < b$ e $p(a), p(b) \neq 0$, então o número de zeros de $p(n)$ no intervalo $[a, b]$ é $\delta(a) - \delta(b)$.

Example ($p(n) = n^4 + n^3 - n - 1$)

- $p(-3) = 56 \neq 0$ e $p(3) = 104 \neq 0$
- $\delta(-3) - \delta(3) = 3 - 1 = 2$
- As únicas raízes reais de $n^4 + n^3 - n - 1 = 0$ são -1 e $+1$.

END OF PART I

Bibliography I


Jack B. Copeland.
Even Turing machines can compute uncomputable functions.
Unconventional Models of Computation.
Christian Calude, John Cast e M. J. Dinneen (editores),
Lecture Notes in Computer Science, Springer, 150–164.
Springer, 1998.


Jack B. Copeland.
Super Turing-machines.
Complexity, 4: 30–32, 1998.


Martin Davis.
Hilbert's tenth problem is unsolvable.
The American Mathematical Monthly, 80(3):233–269, 1973.

Bibliography II


Fred Hoyle.
From Stonehenge to Modern Cosmology.
W. H. Freeman, 1972.


Oron Shagrir.
Supertasks do not increase computational power.
Natural Computing, 11(1):51–58, 2012.


Yuri V.Matiyasevich.
Hilbert's Tenth Problem.
Massachusetts Institute of Technology, 1993.