Computation on the Real Numbers And Other Uncountable Domains

Russell Miller

Queens College & CUNY Graduate Center
New York, NY

University of Connecticut Logic Seminar, 25 Feb. 2011

Slides available at
`qc.edu/~rmiller/slides.html`

Turing Computation

A Turing machine takes as input a finite binary string, e.g. 1100010111, and (if it ever halts) outputs another such string. But there are only countably many such strings! So we naturally think of a Turing machine as computing a function from $\mathbb{N}$ to $\mathbb{N}$, or from $\mathbb{Q}$ to $\mathbb{Q}$, but not from $\mathbb{R}$ to $\mathbb{R}$.

One might ask what happens if we give the machine an *infinite* binary string as input:

$$1100010111100100001101010111\ldots \in 2^{\omega}.$$

But if the machine halts within finitely many steps, it will only have "seen" finitely many of these bits before halting, and all further bits will be exactly the same in the output as in the input.

So how can we conceive of computation on $\mathbb{R}$ or on $\mathbb{C}$?

Section I: Computable Analysis

A real number can be viewed as a *Cauchy sequence* of rationals:

Defn.

An infinite sequence $q_0, q_1, q_2, \ldots$ of rational numbers *represents* a real number $r \in \mathbb{R}$ if

$$(\forall n)\ |q_n - r| < \frac{1}{2^n}.$$

Section I: Computable Analysis

A real number can be viewed as a *Cauchy sequence* of rationals:

Defn.

An infinite sequence $q_0, q_1, q_2, \ldots$ of rational numbers *represents* a real number $r \in \mathbb{R}$ if

$$(\forall n)\ |q_n - r| < \frac{1}{2^n}.$$

The 2^{-n} bound on the approximation makes it more effective: we know (a bound on) how fast the approximation converges to r. However, this is not always enough to give r in decimal form.

Problem

Given that the following computable sequence does represent a real number r, find the second decimal digit of r:

5, 5.3, 5.32, 5.32, 5.32, 5.32, 5.32, 5.32, 5.32,

Section I: Computable Analysis

A real number can be viewed as a *Cauchy sequence* of rationals:

Defn.

An infinite sequence $q_0, q_1, q_2, \ldots$ of rational numbers *represents* a real number $r \in \mathbb{R}$ if

$$(\forall n)\ |q_n - r| < \frac{1}{2^n}.$$

The 2^{-n} bound on the approximation makes it more effective: we know (a bound on) how fast the approximation converges to r. However, this is not always enough to give r in decimal form.

Problem

Given that the following computable sequence does represent a real number r, find the second decimal digit of r:

5, 5.3, 5.32, 5.32, 5.32, 5.32, 5.32, 5.32, 5.32, 5.31999994,

Section I: Computable Analysis

A real number can be viewed as a *Cauchy sequence* of rationals:

Defn.

An infinite sequence $q_0, q_1, q_2, \ldots$ of rational numbers *represents* a real number $r \in \mathbb{R}$ if

$$(\forall n)\ |q_n - r| < \frac{1}{2^n}.$$

The 2^{-n} bound on the approximation makes it more effective: we know (a bound on) how fast the approximation converges to r. However, this is not always enough to give r in decimal form.

Problem

Given that the following computable sequence does represent a real number r, find the second decimal digit of r:

$$5,\ 5.3,\ 5.32,\ 5.32,\ 5.32,\ 5.32,\ 5.32,\ 5.32,\ 5.32,\ 5.31999994,\ \ldots$$

Computing a Function $f : \mathbb{R} \to \mathbb{R}$

Defn.

In *computable analysis*, we say that a function f is *computable* if there exists an oracle Turing functional Φ such that, for every sequence $Q = (q_0, q_1, q_2, \ldots)$ representing any real number x, the sequence

$$\Phi^Q(0),\ \Phi^Q(1),\ \Phi^Q(2),\ \Phi^Q(3), \ldots$$

represents $f(x)$.

The following functions from $\mathbb{R}$ to $\mathbb{R}$ (or from $\mathbb{R}^2$ to $\mathbb{R}$) are computable, in computable analysis:

- $f(x, y) = x + y$ and $g(x, y) = x \cdot y$;
- $\exp(x) = e^x$ and $\ln(x)$; likewise for any computable base;
- all trigonometric functions (in either radians or degrees);
- $h(x) = \frac{1}{x}$, assuming that we do not care what the machine does on input 0.

Noncomputable Functions in Computable Analysis

The following functions from $\mathbb{R}$ to $\mathbb{R}$ (or from $\mathbb{R}^2$ to $\mathbb{R}$) are *not* computable, in computable analysis:

- the characteristic function of equality:

$$f(x,y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{if } x \neq y. \end{cases}$$

- the characteristic function of the $<$ relation;
- every discontinuous function.

However, if we know that $x \neq y$, then it *is* computable whether $x < y$ or $y < x$.

Noncomputable Functions in Computable Analysis

The following functions from $\mathbb{R}$ to $\mathbb{R}$ (or from $\mathbb{R}^2$ to $\mathbb{R}$) are *not* computable, in computable analysis:

- the characteristic function of equality:

$$f(x,y) = \begin{cases} 1, & \text{if } x = y \\ 0, & \text{if } x \neq y. \end{cases}$$

- the characteristic function of the $<$ relation;
- every discontinuous function.

However, if we know that $x \neq y$, then it *is* computable whether $x < y$ or $y < x$.

Also, for every computable function $f : \mathbb{R} \to \mathbb{R}$, the integral $\int_0^x f(y)dy$ is also computable on $\mathbb{R}$. The derivative $f'(x)$, even if it is continuous, can be a noncomputable function. But if f is computable and $\mathcal{C}^2$ on $[0,1]$, then f' is computable there, by a theorem of Pour-El/Richards.

Section II: Infinite-Time Computation

An *infinite-time Turing machine* has three one-way tapes and a reading head:

input	1	0	0	1	1	$\cdots$
output	0	0	0	0	0	$\cdots$
scratch	0	0	0	0	0	$\cdots$

At stage 0, the machine is in the *start state*, with an input from 2^ω.

At stage $\alpha + 1$, the machine acts according to a finite program, using the configuration at stage α, just like an ordinary Turing machine.

At a limit stage λ, the machine begins in the *limit state*, with each box displaying the $\limsup$ of its values at stages $< \lambda$.

One state is designated as the *halt state*, and the machine may or may not ever enter this state.

Any element of 2^ω can be the input to such a machine.

$\mathbb{R}$ in Infinite Time Computation

Fact

The ordered field $\mathbb{R}$ is infinite-time computably presentable.

First, code $\mathbb{R}$ into 2^ω. Example: $12\frac{1}{3}$ in binary is $1100.010101\ldots$, which we code as $\{2,3\} \oplus \{1,3,5,\ldots\}$.

Addition of any two reals can be done in $(\omega+\omega)$ steps, and multiplication, subtraction, and division the same.

Equality and $<$ are both decidable in $(\omega+1)$ steps.

All functions computable in computable analysis are computable by these machines in time $< \omega^\omega$.

More Infinite Time Computation

Obviously, computing the ordered field $\mathbb{R}$ does not use the full power of infinite time computation! In fact, the Π^1_1-complete set

$$\mathbf{WO} = \{W \subseteq \omega \times \omega : W \text{ is a well-ordering of } \omega\}$$

is infinite-time decidable.

- The arithmetic subsets of ω are those decidable in time bounded by some $\alpha < \omega^\omega$.
- The hyperarithmetic subsets of ω are those decidable in time bounded by a Turing-computable ordinal.
- All infinite-time decidable sets are Δ^1_2.

Infinite Time Computation: A Curiosity

Lost Melody Theorem (Hamkins-Lewis)

There exists $c \in 2^\omega$ such that no infinite-time Turing machine on input $\emptyset$ halts with output c, yet the set $\{c\}$ is infinite-time decidable.

Corollary

The constant function $f(x) = c$ has an infinite-time decidable graph, but is not infinite-time computable.

In the proof, c codes an ordinal α so large that every program which halts on input $\emptyset$ must halt by stage α.

The main point, however, is that one cannot just use the decision procedure for $\{c\}$ to identify c, because, although one can check whether any given x is equal to c, one cannot search through all x in 2^ω.

Other Infinite Time Machines

Computation can be generalized to ordinals in other ways.

- Allow the (one-way) *tape* to have ordinal-many cells, not just ω-many. (Problem: if the reading head is on cell number ω, and the program says to move one cell to the left, what happens?)
- Generalize *register machines*: run for ordinal-many stages, with each register containing a single ordinal at each stage. Registers can be incremented, copied to each other, or set to zero.
- Each of the above can be done with ordinal bounds α on time and β on space (with $\beta \leq \alpha$). Usually α and β would be *admissible* ordinals.

Other Infinite Time Machines

Computation can be generalized to ordinals in other ways.

- Allow the (one-way) *tape* to have ordinal-many cells, not just ω-many. (Problem: if the reading head is on cell number ω, and the program says to move one cell to the left, what happens?)
- Generalize *register machines*: run for ordinal-many stages, with each register containing a single ordinal at each stage. Registers can be incremented, copied to each other, or set to zero.
- Each of the above can be done with ordinal bounds α on time and β on space (with $\beta \leq \alpha$). Usually α and β would be *admissible* ordinals.

Theorem (Koepke)

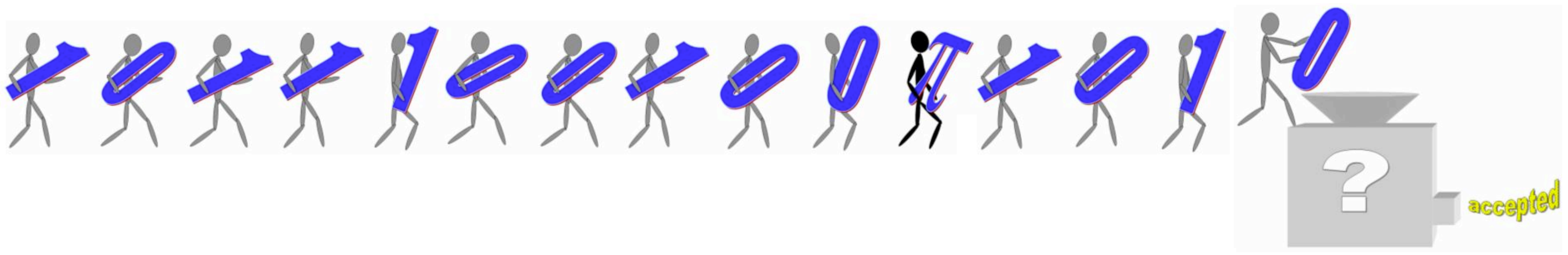
The sets computable by ordinal-time Turing machines (equivalently, by ordinal-time register machines) with no time or space bounds, and with finitely many ordinal parameters allowed, are exactly the sets in Gödel's constructible universe **L**.

Section III: BSS Computation on $\mathbb{R}$

Roughly, a BSS machine *M* on $\mathbb{R}$ operates like a Turing machine, but with a real number in each cell, rather than a bit.

- *M* can compute full-precision $+.\ -.\ \cdot$, and $\div$ on numbers in its cells.
- *M* can compare 0 to the number in any cell, using $=$ or $<$, and fork according to the answer.
- *M* is allowed finitely many real numbers $z_0, \ldots, z_m$ as *parameters* in its program. The input and output (if *M* halts) are tuples $\vec{y} \in \mathbb{R}^\infty = \{$ finite tuples from $\mathbb{R}$ $\}$.

A subset $S \subseteq \mathbb{R}^\infty$ is BSS-*decidable* iff its characteristic function χ_S is computable by a BSS machine, and BSS-*semidecidable* iff *S* is the domain of some BSS-computable function.


?
accepted

Basic Facts about BSS Computation

For a machine M with parameters $\vec{z}$, running on input $\vec{y}$, only elements of the field $\mathbb{Q}(\vec{z}, \vec{y})$ can ever appear in the cells of M.

Cell: 0	$\cdots$	m	$m+1$	$\cdots$	$m+n$	$m+n+1$	$\cdots$
z_0	$\cdots$	z_m	y_1	$\cdots$	y_n		
z_0	$\cdots$	z_m	y_1	$\cdots$	y_n	$z_m + y_n$	
$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$	
$f_{0,s}(\vec{y})$	$\cdots$	$f_{m,s}(\vec{y})$	$f_{m+1,s}(\vec{y})$	$\cdots$	$f_{m+n,s}(\vec{y})$	$f_{m+n+1,s}(\vec{y})$	$\cdots$
$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$	

Basic Facts about BSS Computation

For a machine M with parameters $\vec{z}$, running on input $\vec{y}$, only elements of the field $\mathbb{Q}(\vec{z}, \vec{y})$ can ever appear in the cells of M.

Cell: 0	$\cdots$	m	$m+1$	$\cdots$	$m+n$	$m+n+1$	$\cdots$
z_0	$\cdots$	z_m	y_1	$\cdots$	y_n		
z_0	$\cdots$	z_m	y_1	$\cdots$	y_n	$z_m + y_n$	
$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$	
$f_{0,s}(\vec{y})$	$\cdots$	$f_{m,s}(\vec{y})$	$f_{m+1,s}(\vec{y})$	$\cdots$	$f_{m+n,s}(\vec{y})$	$f_{m+n+1,s}(\vec{y})$	$\cdots$
$\vdots$		$\vdots$	$\vdots$		$\vdots$	$\vdots$	

For each input $\vec{y}$, every $f_{i,s}(Y_1, \ldots, Y_n)$ is a rational function with coefficients from the field $\mathbb{Q}(\vec{z})$. If the input $\{y_1, \ldots, y_n\}$ is algebraically independent over $\mathbb{Q}(\vec{z})$, then each $f_{i,s}(\vec{Y})$ is uniquely defined.

BSS-Noncomputable Functions

The function $f(x) = x^3$ is certainly BSS-computable. However, its inverse $g(x) = \sqrt[3]{x}$ is not!

Proof: Suppose M computed g. Let $z_1, \ldots, z_n \in \mathbb{R}$ be the parameters used by the program for M. Then for every $x \in \mathbb{R}$, $g(x) = \sqrt[3]{x}$ would lie in the field $\mathbb{Q}(z_1, \ldots, z_n, x)$. In particular, all cube roots of rational numbers would lie in $\mathbb{Q}(z_1, \ldots, z_n)$. But the cube roots of the primes generate an infinite-degree field extension of $\mathbb{Q}$, so this is impossible.

BSS-Noncomputable Functions

The function $f(x) = x^3$ is certainly BSS-computable. However, its inverse $g(x) = \sqrt[3]{x}$ is not!

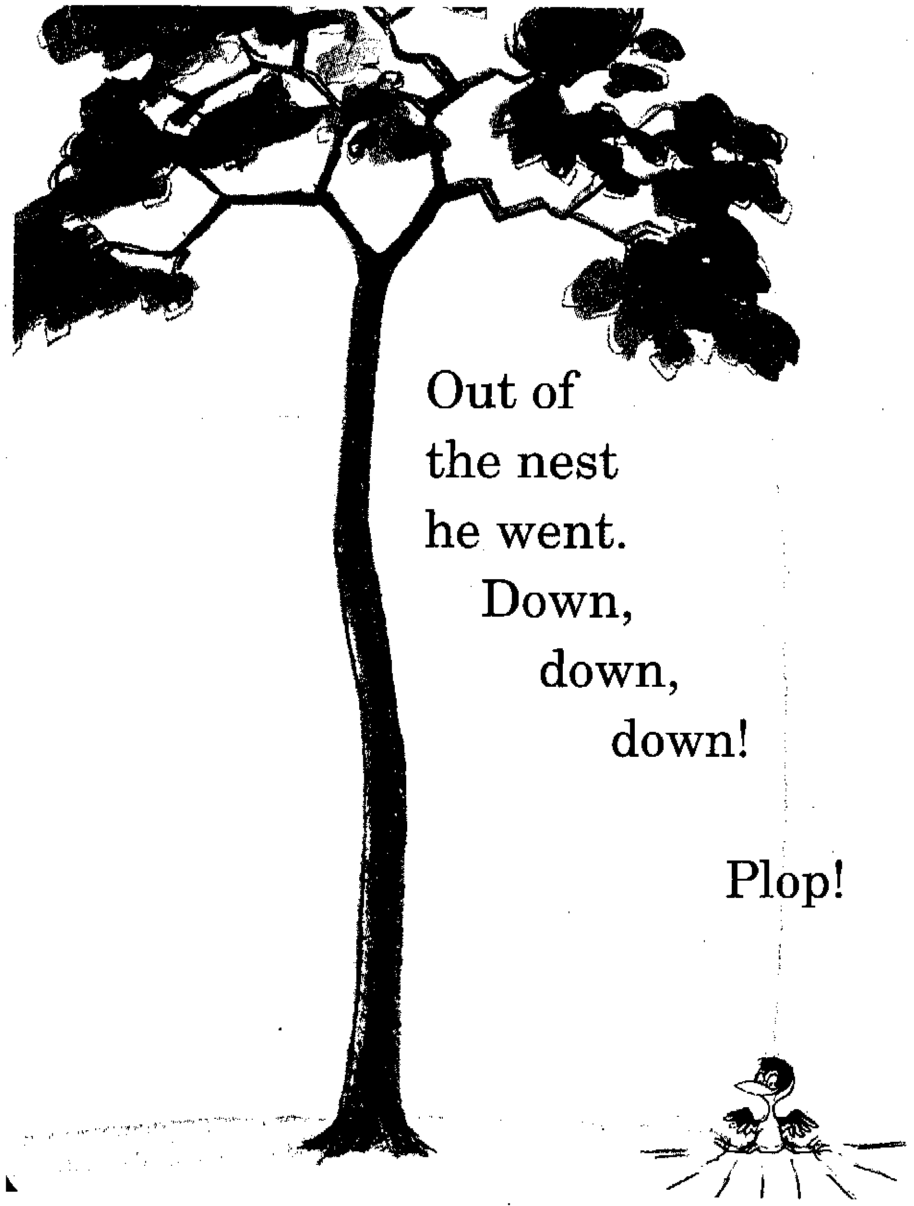
Proof: Suppose M computed g. Let $z_1, \ldots, z_n \in \mathbb{R}$ be the parameters used by the program for M. Then for every $x \in \mathbb{R}$, $g(x) = \sqrt[3]{x}$ would lie in the field $\mathbb{Q}(z_1, \ldots, z_n, x)$. In particular, all cube roots of rational numbers would lie in $\mathbb{Q}(z_1, \ldots, z_n)$. But the cube roots of the primes generate an infinite-degree field extension of $\mathbb{Q}$, so this is impossible.

This contrasts with the Turing-computable presentation of the (countable) field F of all algebraic real numbers. In F, we can find the cube root of any x: just search through F for some y with $y^3 = x$.

Are You My Mother?

BRIGHT and EARLY BOOKS

by P. D. Eastman

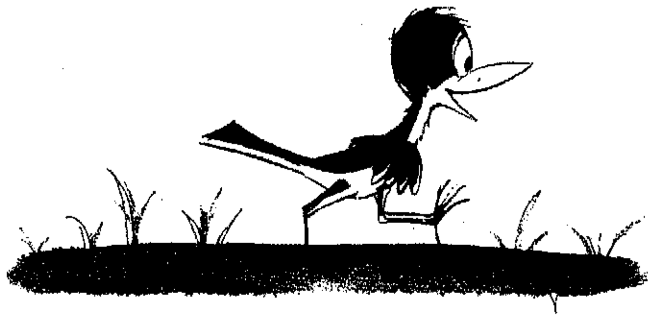


Out of
the nest
he went.
Down,
down,
down!

Plop!

The baby bird could not fly.

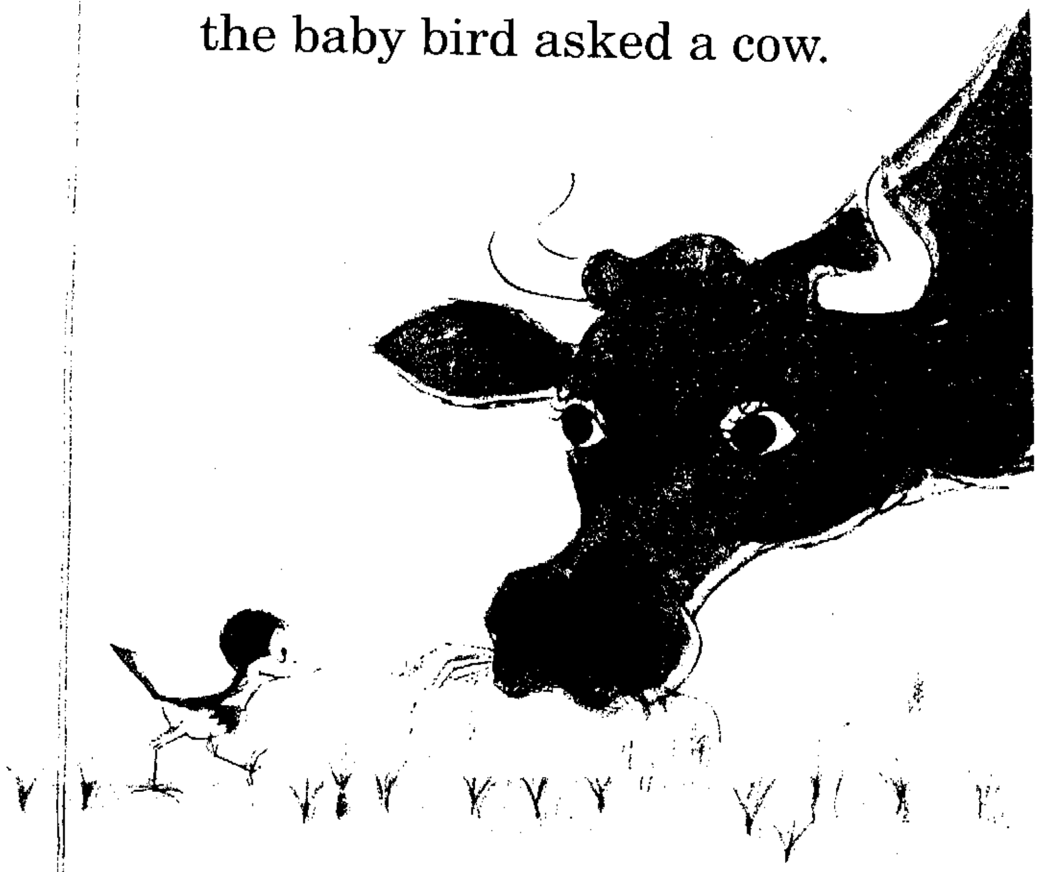
But he could walk.
"Now I will go and find
my mother," he said.



"Are you my mother?"
the baby bird asked a dog.

"I am not your mother.
I am a dog," said the dog.

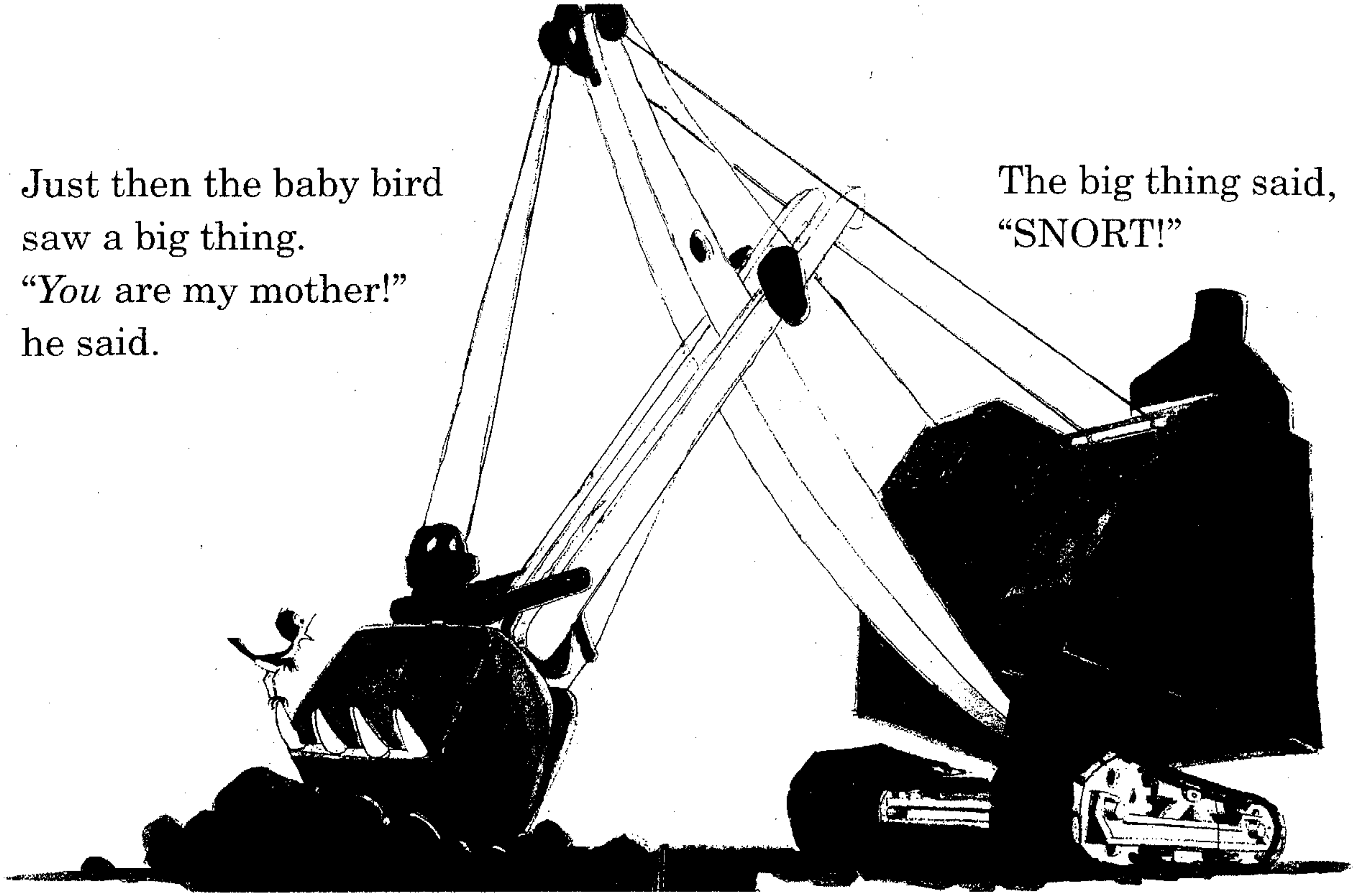
"Are you my mother?"
the baby bird asked a cow.



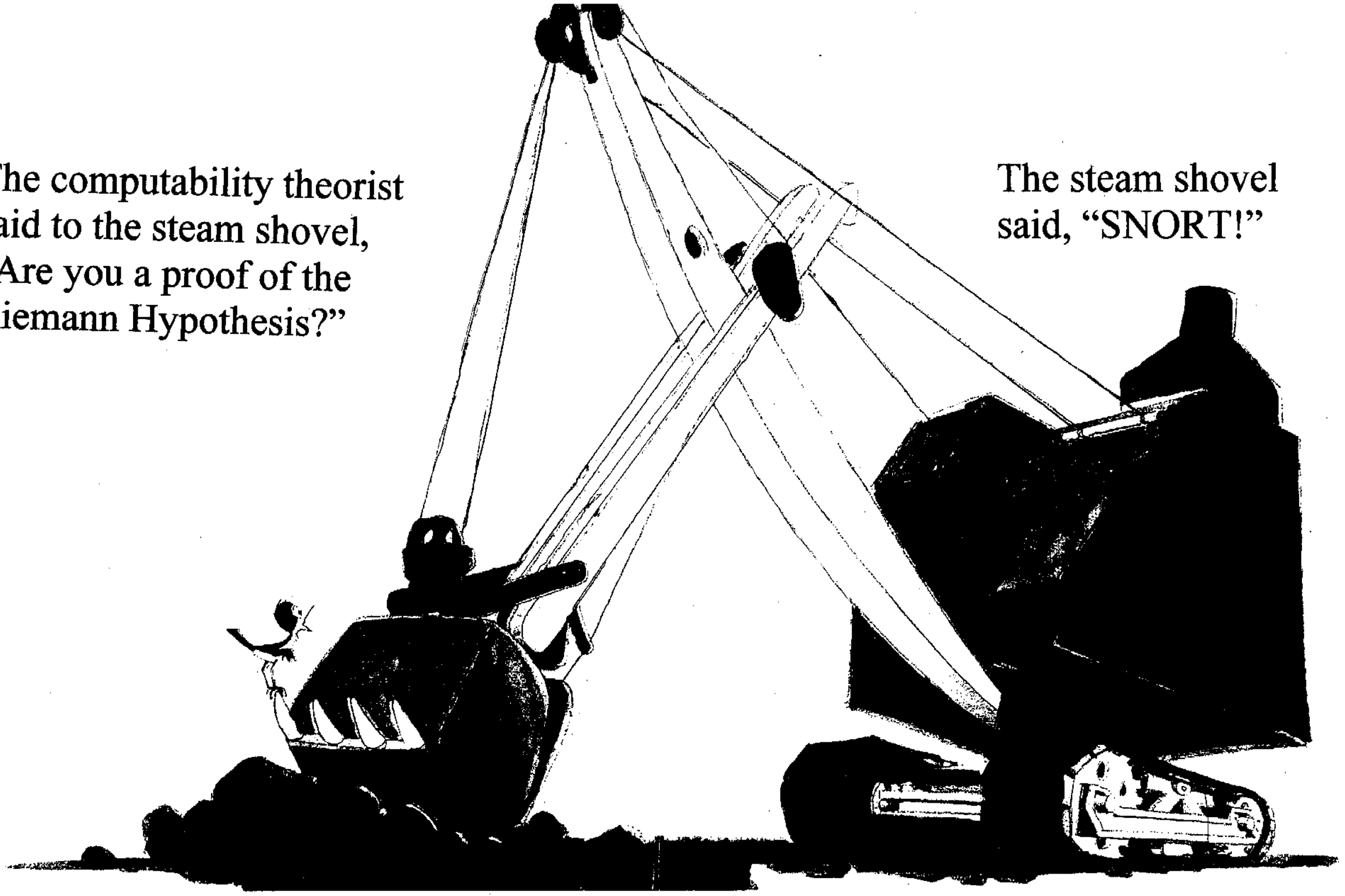
"How could I be your mother?"
said the cow. "I am a cow."

Just then the baby bird
saw a big thing.
"*You* are my mother!"
he said.

The big thing said,
"SNORT!"



The computability theorist said to the steam shovel, “Are you a proof of the Riemann Hypothesis?”



The steam shovel said, “SNORT!”

Search Procedures in Other Models

The same phenomenon occurs in infinite-time computation, for functions from 2^ω to 2^ω: there is no procedure for searching through the entire domain 2^ω.

However, in the more general infinite-time models, search procedures allow us to compute inverses of all computable functions, just as in finite-time Turing computation.

And in computable analysis, every 1-1 function from $\mathbb{R}$ to $\mathbb{R}$ is strictly monotonic, and therefore has computable inverse. (To approximate $f^{-1}(y)$, find rational numbers q and r such that $f(q) < y < f(r)$, but $f(q)$ and $f(r)$ are very close to each other.)

Restrictions on BSS Computation

Given a machine M with parameters $\vec{z}$, choose any input $\vec{y}$ algebraically independent over $\mathbb{Q}(\vec{z})$. If $M(\vec{y})$ halts after t steps, then only finitely many functions $f_{i,s}$ appear. So there is an $\epsilon > 0$ such that for all inputs $\vec{x}$ within ϵ of $\vec{y}$, M at stage s contains:

$f_{0,s}(\vec{x})$	$\cdots$	$f_{m,s}(\vec{x})$	$f_{m+1,s}(\vec{x})$	$\cdots$	$f_{m+n,s}(\vec{x})$	$f_{m+n+1,s}(\vec{x})$	$\cdots$

with the same functions $f_{i,s}$ as for $\vec{y}$.
Therefore, on any $\vec{x} \in \mathbb{R}^n$ in an ϵ-ball around $\vec{y}$, M always halts after t steps, and computes the function $\langle f_{0,t}(\vec{x}), \ldots, f_{m+n+t,t}(\vec{x})\rangle$.

Restrictions on BSS Computation

Given a machine M with parameters $\vec{z}$, choose any input $\vec{y}$ algebraically independent over $\mathbb{Q}(\vec{z})$. If $M(\vec{y})$ halts after t steps, then only finitely many functions $f_{i,s}$ appear. So there is an $\epsilon > 0$ such that for all inputs $\vec{x}$ within ϵ of $\vec{y}$, M at stage s contains:

$f_{0,s}(\vec{x})$	$\cdots$	$f_{m,s}(\vec{x})$	$f_{m+1,s}(\vec{x})$	$\cdots$	$f_{m+n,s}(\vec{x})$	$f_{m+n+1,s}(\vec{x})$	$\cdots$

with the same functions $f_{i,s}$ as for $\vec{y}$.
Therefore, on any $\vec{x} \in \mathbb{R}^n$ in an ϵ-ball around $\vec{y}$, M always halts after t steps, and computes the function $\langle f_{0,t}(\vec{x}), \ldots, f_{m+n+t,t}(\vec{x})\rangle$.

Corollary: No BSS-decidable set can be dense and codense within any nonempty open subset of $\mathbb{R}^n$.

Theorems on Oracle BSS Computation

(All this work is joint with Wesley Calvert and Ken Kramer.)

- **Thm.:** Let $\mathbb{A}_{=d} := \{y \in \mathbb{R} : y$ is algebraic of degree d over $\mathbb{Q}\}$. Then for all $d \geq 0$, $\mathbb{A}_{=d+1} \not\leq_{BSS} \mathbb{A}_{=d}$. Indeed $\mathbb{A}_{=d+1} \not\leq_{BSS} \cup_{c \leq d} \mathbb{A}_c$.
- **Prop.:** Let p and r be any positive integers. Then $\mathbb{A}_{=p} \leq_{BSS} \mathbb{A}_{=r}$ if and only if p divides r.

Theorems on Oracle BSS Computation

(All this work is joint with Wesley Calvert and Ken Kramer.)

- **Thm.:** Let $\mathbb{A}_{=d} := \{y \in \mathbb{R} : y$ is algebraic of degree d over $\mathbb{Q}\}$. Then for all $d \geq 0$, $\mathbb{A}_{=d+1} \not\leq_{BSS} \mathbb{A}_{=d}$. Indeed $\mathbb{A}_{=d+1} \not\leq_{BSS} \cup_{c \leq d} \mathbb{A}_c$.
- **Prop.:** Let p and r be any positive integers. Then $\mathbb{A}_{=p} \leq_{BSS} \mathbb{A}_{=r}$ if and only if p divides r.
- **Prop.:** Let P be the set of all prime numbers in ω and let $S \subseteq P$ and $T \subseteq P$, Then $A_S \leq_{BSS} A_T$ if and only if $S \subseteq T$.
  (Here $\mathbb{A}_S = \cup_{d \in S} \mathbb{A}_{=d}$.)

Theorems on Oracle BSS Computation

(All this work is joint with Wesley Calvert and Ken Kramer.)

- **Thm.:** Let $\mathbb{A}_{=d} := \{y \in \mathbb{R} : y \text{ is algebraic of degree } d \text{ over } \mathbb{Q}\}$. Then for all $d \geq 0$, $\mathbb{A}_{=d+1} \not\leq_{BSS} \mathbb{A}_{=d}$. Indeed $\mathbb{A}_{=d+1} \not\leq_{BSS} \cup_{c \leq d} \mathbb{A}_c$.
- **Prop.:** Let p and r be any positive integers. Then $\mathbb{A}_{=p} \leq_{BSS} \mathbb{A}_{=r}$ if and only if p divides r.
- **Prop.:** Let P be the set of all prime numbers in ω and let $S \subseteq P$ and $T \subseteq P$, Then $A_S \leq_{BSS} A_T$ if and only if $S \subseteq T$.
  (Here $\mathbb{A}_S = \cup_{d \in S} \mathbb{A}_{=d}$.)
- **Cor.:** There exists a subset $\mathcal{L}$ of the BSS-semidecidable degrees such that $(\mathcal{L}, \leq_{BSS}) \cong (\mathcal{P}(\omega), \subseteq)$.

Theorems on Oracle BSS Computation

(All this work is joint with Wesley Calvert and Ken Kramer.)

- **Thm.:** Let $\mathbb{A}_{=d} := \{y \in \mathbb{R} : y \text{ is algebraic of degree } d \text{ over } \mathbb{Q}\}$. Then for all $d \geq 0$, $\mathbb{A}_{=d+1} \not\leq_{BSS} \mathbb{A}_{=d}$. Indeed $\mathbb{A}_{=d+1} \not\leq_{BSS} \cup_{c \leq d} \mathbb{A}_c$.
- **Prop.:** Let p and r be any positive integers. Then $\mathbb{A}_{=p} \leq_{BSS} \mathbb{A}_{=r}$ if and only if p divides r.
- **Prop.:** Let P be the set of all prime numbers in ω and let $S \subseteq P$ and $T \subseteq P$, Then $A_S \leq_{BSS} A_T$ if and only if $S \subseteq T$.
  (Here $\mathbb{A}_S = \cup_{d \in S} \mathbb{A}_{=d}$.)
- **Cor.:** There exists a subset $\mathcal{L}$ of the BSS-semidecidable degrees such that $(\mathcal{L}, \leq_{BSS}) \cong (\mathcal{P}(\omega), \subseteq)$.
- **Thm.:** If $C \subseteq \mathbb{R}^\infty$ is a set such that the BSS Halting Problem H satisfies $H \leq_{BSS} C$, then $|C| = 2^\omega$. Indeed $\mathbb{R}$ must have finite transcendence degree over the field generated by the coordinates of the tuples in C.