

Complexity Theory for Operators in Analysis

Akitoshi Kawamura Stephen Cook

University of Toronto

42nd ACM Symposium on Theory of Computing
June 7, 2010

What this talk is about

Goal: to define complexity for various objects in analysis

	“computable”	“poly-time”
Real functions $f \in C[0, 1]$	✓	✓
Operators $F: C[0, 1] \rightarrow C[0, 1]$	✓	?

$C[0, 1] =$ all continuous functions $f: [0, 1] \rightarrow \mathbb{R}$

What this talk is about

Goal: to define complexity for various objects in analysis

	“computable”	“poly-time”
Real functions $f \in C[0, 1]$	✓	✓
Operators $F: C[0, 1] \rightarrow C[0, 1]$	✓	?

$C[0, 1] =$ all continuous functions $f: [0, 1] \rightarrow \mathbb{R}$

What this talk is about

Goal: to define complexity for various objects in analysis

	“computable”	“poly-time”
Real functions $f \in C[0, 1]$	✓	✓
Operators $F: C[0, 1] \rightarrow C[0, 1]$	✓	?

$C[0, 1] =$ all continuous functions $f: [0, 1] \rightarrow \mathbb{R}$

What this talk is about

Goal: to define complexity for various objects in analysis

	“computable”	“poly-time”
Real functions $f \in C[0, 1]$	✓	✓
Operators $F: C[0, 1] \rightarrow C[0, 1]$	✓	?

$C[0, 1] =$ all continuous functions $f: [0, 1] \rightarrow \mathbb{R}$

What this talk is about

Goal: to define complexity for various objects in analysis

	“computable”	“poly-time”
Real functions $f \in C[0, 1]$	✓	✓
Operators $F: C[0, 1] \rightarrow C[0, 1]$	✓	?

$C[0, 1]$ = all continuous functions $f: [0, 1] \rightarrow \mathbb{R}$

What this talk is about

Goal: to define complexity for various objects in analysis

	“computable”	“poly-time”
Real functions $f \in C[0, 1]$	✓	✓
Operators $F: C[0, 1] \rightarrow C[0, 1]$	✓	?

$C[0, 1] =$ all continuous functions $f: [0, 1] \rightarrow \mathbb{R}$

What this talk is about

Goal: to define complexity for various objects in analysis

	“computable”	“poly-time”
Real functions $f \in C[0, 1]$	✓	✓
Operators $F: C[0, 1] \rightarrow C[0, 1]$	✓	?

$C[0, 1] =$ all continuous functions $f: [0, 1] \rightarrow \mathbb{R}$

Computable Analysis

= Mathematical Analysis from the computability viewpoint

Computable Analysis

= Mathematical Analysis from the computability viewpoint

Example: Cauchy–Lipschitz Theorem

For any Lipschitz function $g: [0, 1] \times \mathbb{R} \rightarrow \mathbb{R}$,
there is a (unique) solution $h: [0, 1] \rightarrow \mathbb{R}$ of the equation

$$h(0) = 0, \quad h'(t) = g(t, h(t)).$$

(We write $h = \text{Sol}(g)$.)



Can we compute h ?

Computable Analysis

= Mathematical Analysis from the computability viewpoint

Example: Cauchy–Lipschitz Theorem

For any Lipschitz function $g: [0, 1] \times \mathbb{R} \rightarrow \mathbb{R}$,
there is a (unique) solution $h: [0, 1] \rightarrow \mathbb{R}$ of the equation

$$h(0) = 0, \quad h'(t) = g(t, h(t)).$$

(We write $h = \text{Sol}(g)$.)



Can we compute h ?

Two ways to formulate computability results

Computable CL Theorem

If g is computable, then $Sol(g)$ is computable.

Computable CL Theorem ☺

Sol is computable.

Two ways to formulate computability results

Computable CL Theorem

If g is **computable**, then $Sol(g)$ is **computable**.

real function

real function

Computable CL Theorem ☺

Sol is **computable**.

operator

Two ways to formulate computability results

Computable CL Theorem

If g is **computable**, then $Sol(g)$ is **computable**.

real function

real function



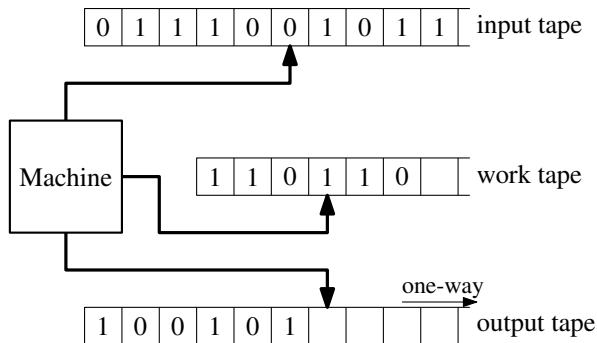
Computable CL Theorem ☺

Sol is **computable**.

operator

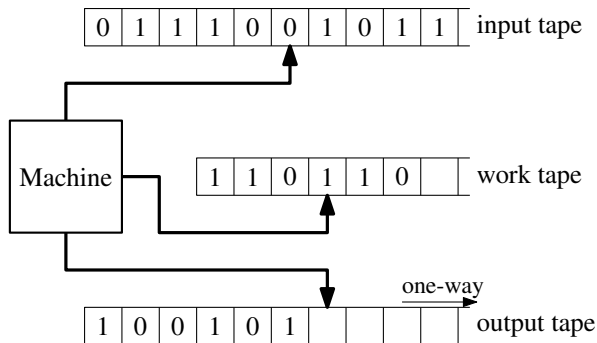
Definition of computability (1/2)

Type-Two Machine: converts infinite strings to infinite strings



Definition of computability (1/2)

Type-Two Machine: converts infinite strings to infinite strings



We encode various objects into infinite strings. For example, ...

Definition of computability (2/2)

- ▶ Name of a real number $t \in \mathbb{R}$

3 # 3.2 # 3.14 # 3.141 # 3.1416 # ...

Definition of computability (2/2)

- ▶ Name of a real number $t \in \mathbb{R}$

3 # 3.2 # 3.14 # 3.141 # 3.1416 # ...

This defines computability of real functions.

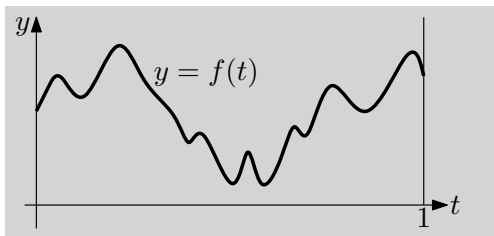
Definition of computability (2/2)

- ▶ Name of a real number $t \in \mathbb{R}$

3 # 3.2 # 3.14 # 3.141 # 3.1416 # ...

This defines computability of real functions.

- ▶ Name of a function $f \in C[0, 1]$



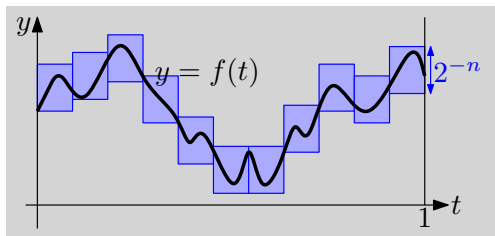
Definition of computability (2/2)

- ▶ Name of a real number $t \in \mathbb{R}$

3 # 3.2 # 3.14 # 3.141 # 3.1416 # ...

This defines computability of real functions.

- ▶ Name of a function $f \in C[0, 1]$
= a sequence s_1, s_2, \dots where each s_n is a finite list of rational boxes specifying f with precision 2^{-n}



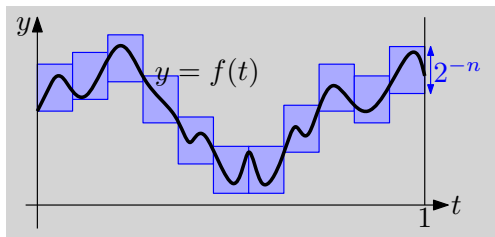
Definition of computability (2/2)

- ▶ Name of a real number $t \in \mathbb{R}$

3 # 3.2 # 3.14 # 3.141 # 3.1416 # ...

This defines computability of real functions.

- ▶ Name of a function $f \in C[0, 1]$
= a sequence s_1, s_2, \dots where each s_n is a finite list of rational boxes specifying f with precision 2^{-n}



This defines computability of operators on $C[0, 1]$.

Formulating complexity results

Poly-Space CL Theorem [Ko 1983]

If g is **poly-space**, then $Sol(g)$ is **poly-space**.

Formulating complexity results

Poly-Space CL Theorem [Ko 1983]

If g is **poly-space**, then $Sol(g)$ is **poly-space**.

real function

real function

Formulating complexity results

Poly-Space CL Theorem [Ko 1983]

If g is **poly-space**, then $Sol(g)$ is **poly-space**.

real function

real function



Poly-Space CL Theorem ☺

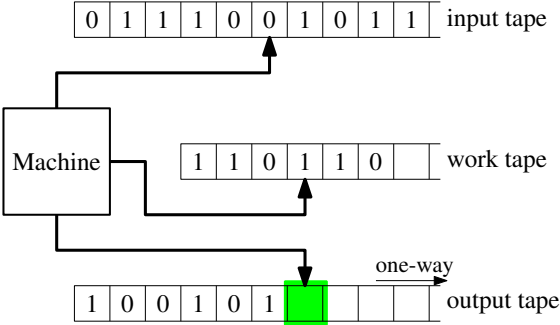
Sol is **poly-space**.

operator

?

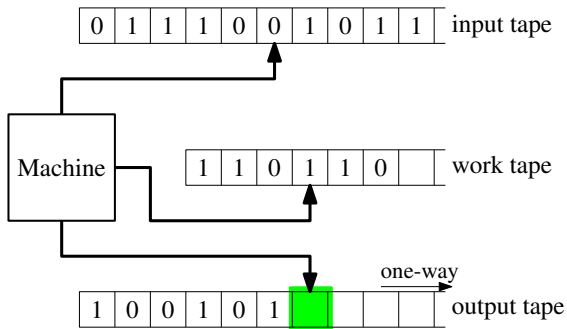
Poly-time/space Type-Two Machine

Takes $\text{poly}(n)$ time/space to write the **n th bit** of the output (regardless of the input)



Poly-time/space Type-Two Machine

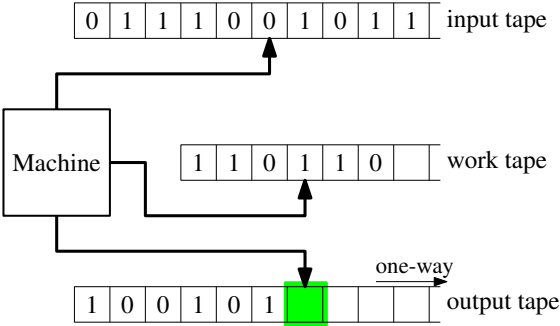
Takes $\text{poly}(n)$ time/space to write the n th bit of the output (regardless of the input)



This definition works for computation over $[0, 1]$,

Poly-time/space Type-Two Machine

Takes $\text{poly}(n)$ time/space to write the **n th bit** of the output (regardless of the input)



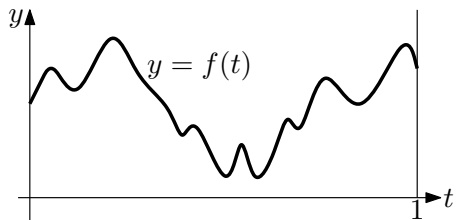
This definition works for computation over $[0, 1]$, because

0.2 # 0.23 # 0.237 # 0.2368 # 0.23684 # . . .

$\leftarrow O(n^2) \text{ bits} = \text{precision } 2^{-n} \rightarrow$

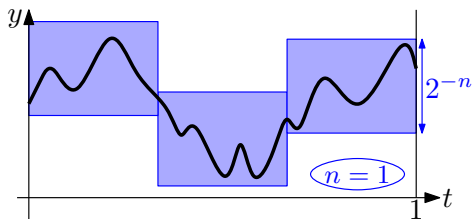
Limitations of the infinite string model

However, for other objects (\mathbb{R} , $C[0, 1]$, ...) we do not know how many bits must be read in order to get a certain precision.



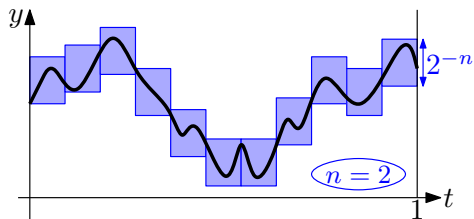
Limitations of the infinite string model

However, for other objects (\mathbb{R} , $C[0, 1]$, ...) we do not know how many bits must be read in order to get a certain precision.



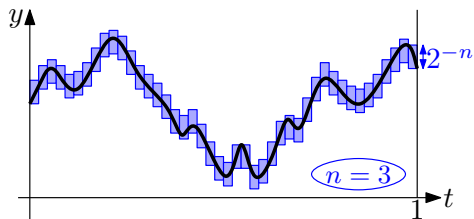
Limitations of the infinite string model

However, for other objects (\mathbb{R} , $C[0, 1]$, ...) we do not know how many bits must be read in order to get a certain precision.



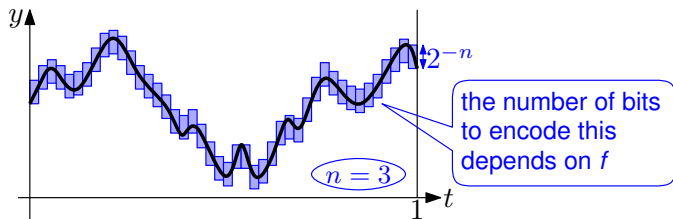
Limitations of the infinite string model

However, for other objects (\mathbb{R} , $C[0, 1]$, ...) we do not know how many bits must be read in order to get a certain precision.



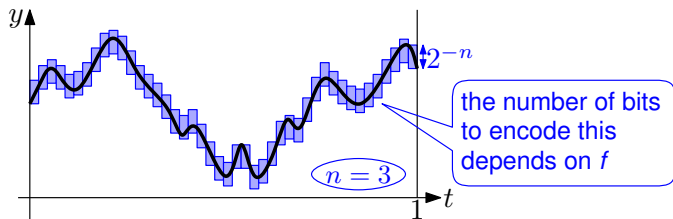
Limitations of the infinite string model

However, for other objects (\mathbb{R} , $C[0, 1]$, ...) we do not know how many bits must be read in order to get a certain precision.



Limitations of the infinite string model

However, for other objects (\mathbb{R} , $C[0, 1]$, ...) we do not know how many bits must be read in order to get a certain precision.

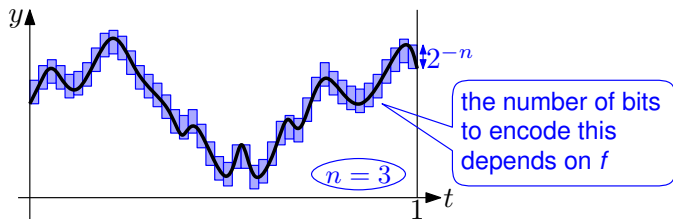


For usual (discrete) problems,

- ▶ there are instances of different sizes,
- ▶ solving big instances (long strings) take long time/space,
- ▶ we measure the time/space relative to the input size.

Limitations of the infinite string model

However, for other objects (\mathbb{R} , $C[0, 1]$, ...) we do not know how many bits must be read in order to get a certain precision.



For usual (discrete) problems,

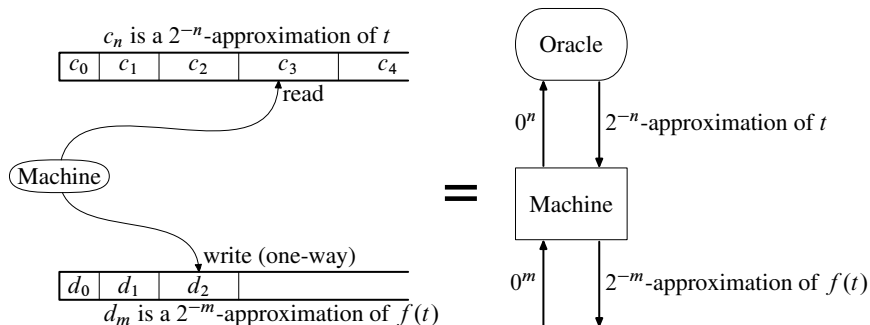
- ▶ there are instances of different sizes,
- ▶ solving big instances (long strings) take long time/space,
- ▶ we measure the time/space relative to the input size.

But infinite strings do not have a “size”.

Infinite strings are a special case of oracles

We use functions $\varphi: \Sigma^* \rightarrow \Sigma^*$ as names.

For real numbers, this is a generalization of infinite strings:



Size of a function (Idea from [Kapron & Cook 1996])

Definition (size of a function)

For a function $\varphi: \Sigma^* \rightarrow \Sigma^*$ with $|x| = |y| \implies |\varphi(x)| = |\varphi(y)|$, its **size** $|\varphi|: \mathbb{N} \rightarrow \mathbb{N}$ is defined by

$$|\varphi|(|x|) = |\varphi(x)|.$$

Size of a function (Idea from [Kapron & Cook 1996])

Definition (size of a function)

For a function $\varphi: \Sigma^* \rightarrow \Sigma^*$ with $|x| = |y| \implies |\varphi(x)| = |\varphi(y)|$, its **size** $|\varphi|: \mathbb{N} \rightarrow \mathbb{N}$ is defined by

$$|\varphi|(|x|) = |\varphi(x)|.$$

Definition (polynomial time/space *in a function*)

A machine is **poly-time/space** if, on input string x and oracle φ , it runs in time/space $P(|\varphi|)(|x|)$, where P is some *second-order polynomial*.

Size of a function (Idea from [Kapron & Cook 1996])

Definition (size of a function)

For a function $\varphi: \Sigma^* \rightarrow \Sigma^*$ with $|x| = |y| \implies |\varphi(x)| = |\varphi(y)|$, its **size** $|\varphi|: \mathbb{N} \rightarrow \mathbb{N}$ is defined by

$$|\varphi|(|x|) = |\varphi(x)|.$$

Definition (polynomial time/space *in a function*)

A machine is **poly-time/space** if, on input string x and oracle φ , it runs in time/space $P(|\varphi|)(|x|)$, where P is some *second-order polynomial*.

E.g. $|\varphi|(|\varphi|(|x| \cdot |x|)) + |\varphi|(|\varphi|(|x|) \cdot |\varphi|(|x|)) + |\varphi|(|x|) + 4$.

Size of a function (Idea from [Kapron & Cook 1996])

Definition (size of a function)

For a function $\varphi: \Sigma^* \rightarrow \Sigma^*$ with $|x| = |y| \implies |\varphi(x)| = |\varphi(y)|$, its **size** $|\varphi|: \mathbb{N} \rightarrow \mathbb{N}$ is defined by

$$|\varphi|(|x|) = |\varphi(x)|.$$

Definition (polynomial time/space *in a function*)

A machine is **poly-time/space** if, on input string x and oracle φ , it runs in time/space $P(|\varphi|)(|x|)$, where P is some *second-order polynomial*.

E.g. $|\varphi|(|\varphi|(|x| \cdot |x|)) + |\varphi|(|\varphi|(|x|) \cdot |\varphi|(|x|)) + |\varphi|(|x|) + 4$.

Under this definition (and the right choice of representations), we can prove. . .

Complexity of operators

Poly-Space CL Theorem ☺

Sol is poly-space.

Complexity of operators

Poly-Space CL Theorem ☺

Sol is **poly-space**.

Also we can formulate and prove the lower bound:

Theorem ☺

Sol is **poly-space complete** (wrt a suitable reduction).

Corollary [K 2009]

There is a poly-time function g such that $Sol(g)$ is poly-space complete.