Computational Complexity of Ordinary Differential Equations

Akitoshi KAWAMURA

University of Tokyo

Ninth International Conference on
Computability and Complexity in Analysis
Cambridge, UK
June 25, 2012

Problem

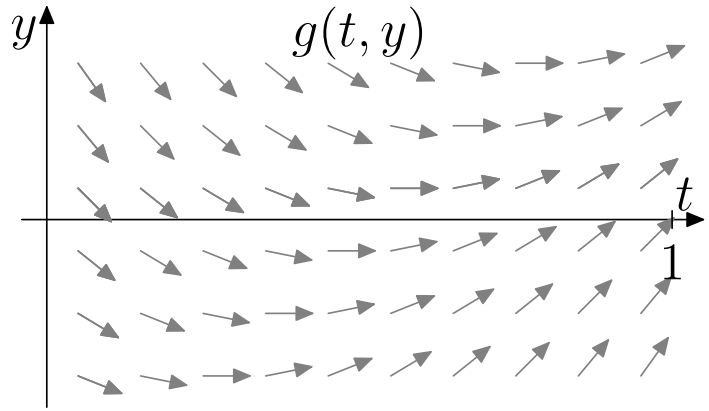
Given $g\colon [0,1] \times \mathbf{R} \to \mathbf{R}$, consider the ODE

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

Problem

Given $g\colon [0,1] \times \mathbf{R} \to \mathbf{R}$, consider the ODE

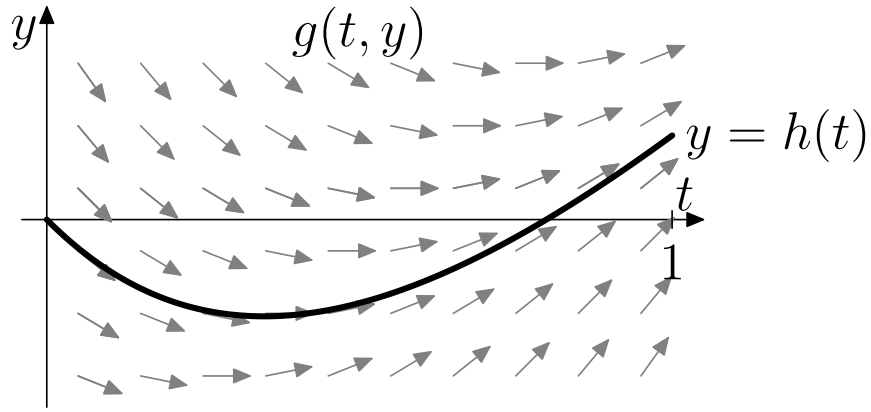
$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big).$$



Problem

Given $g\colon [0,1] \times \mathbf{R} \to \mathbf{R}$, consider the ODE

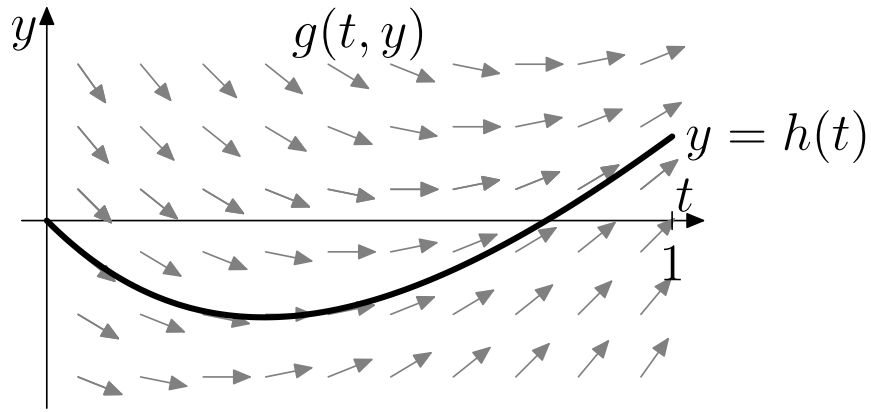
$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big).$$



Problem

Given $g\colon [0,1] \times \mathbf{R} \to \mathbf{R}$, consider the ODE

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



Question

g is simple $\Longrightarrow$ h is simple? (in terms of Computational Complexity)

Summary: Complexity of ODE solutions

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

Assuming g is polytime, how complex can h be?

Assumptions	Upper bound	Lower bound
None	——	can be (non-unique and) all non-computable [Pour-El 1979]
h is the unique solution	computable [implicit in Osgood 1898]	can take arbitrarily long time (or space) [Miller 1970]
g is 'locally' Lipschitz [Ko 1992]	exponential space [Ko 1992]	can be **EXPSPACE**-hard [K 2010]
g is Lipschitz	polyspace [Ko 1983]	can be **PSPACE**-hard [K 2010] (CCC 2009)
g is of class C^1		can be **PSPACE**-hard [K, Ota, Rösnick, Ziegler 2012] (MFCS 2012)
g is of class C^k		can be **CH**-hard [ibid.]
g is analytic	polytime [Müller 1987]	——

Summary: Complexity of ODE solutions

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

Assuming g is polytime, how complex can h be?

Assumptions	Upper bound	Lower bound
None	——	can be (non-unique and) all non-computable [Pour-El 1979]
h is the unique solution	computable [implicit in Osgood 1898]	can take arbitrarily long time (or space) [Miller 1970]
g is 'locally' Lipschitz [Ko 1992]	exponential space [Ko 1992]	can be **EXPSPACE**-hard [K 2010]
g is Lipschitz	polyspace [Ko 1983]	can be **PSPACE**-hard [K 2010] (CCC 2009)
g is of class C^1		can be **PSPACE**-hard [K, Ota, Rösnick, Ziegler 2012] (MFCS 2012)
g is of class C^k		can be **CH**-hard [ibid.]
g is analytic	polytime [Müller 1987]	——

Summary: Complexity of ODE solutions

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

Assuming g is polytime, how complex can h be?

Assumptions	Upper bound	Lower bound
None	——	can be (non-unique and) all non-computable [Pour-El 1979]
➤ h is the unique solution	computable [implicit in Osgood 1898]	can take arbitrarily long time (or space) [Miller 1970]
g is 'locally' Lipschitz [Ko 1992]	exponential space [Ko 1992]	can be **EXPSPACE**-hard [K 2010]
g is Lipschitz	polyspace [Ko 1983]	can be **PSPACE**-hard [K 2010] (CCC 2009)
g is of class C^1		can be **PSPACE**-hard [K, Ota, Rösnick, Ziegler 2012] (MFCS 2012)
g is of class C^k		can be **CH**-hard [ibid.]
g is analytic	polytime [Müller 1987]	——

Summary: Complexity of ODE solutions

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

Assuming g is polytime, how complex can h be?

	Assumptions	Upper bound	Lower bound
	None	——	can be (non-unique and) all non-computable [Pour-El 1979]
	h is the unique solution	computable [implicit in Osgood 1898]	can take arbitrarily long time (or space) [Miller 1970]
	g is 'locally' Lipschitz [Ko 1992]	exponential space [Ko 1992]	can be **EXPSPACE**-hard [K 2010]
➤	g is Lipschitz	polyspace [Ko 1983]	can be **PSPACE**-hard [K 2010] (CCC 2009)
	g is of class C^1		can be **PSPACE**-hard [K, Ota, Rösnick, Ziegler 2012] (MFCS 2012)
	g is of class C^k		can be **CH**-hard [ibid.]
	g is analytic	polytime [Müller 1987]	——

Summary: Complexity of ODE solutions

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

Assuming g is polytime, how complex can h be?

Assumptions	Upper bound	Lower bound
None	——	can be (non-unique and) all non-computable [Pour-El 1979]
h is the unique solution	computable [implicit in Osgood 1898]	can take arbitrarily long time (or space) [Miller 1970]
g is 'locally' Lipschitz [Ko 1992]	exponential space [Ko 1992]	can be **EXPSPACE**-hard [K 2010]
g is Lipschitz	polyspace [Ko 1983]	can be **PSPACE**-hard [K 2010] (CCC 2009)
g is of class C^1		can be **PSPACE**-hard [K, Ota, Rösnick, Ziegler 2012] (MFCS 2012)
➤ g is of class C^k		can be **CH**-hard [ibid.]
g is analytic	polytime [Müller 1987]	——

Summary: Complexity of ODE solutions

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

Assuming g is polytime, how complex can h be?

Assumptions	Upper bound	Lower bound
None	——	can be (non-unique and) all non-computable [Pour-El 1979]
h is the unique solution	computable [implicit in Osgood 1898]	can take arbitrarily long time (or space) [Miller 1970]
g is 'locally' Lipschitz [Ko 1992]	exponential space [Ko 1992]	can be **EXPSPACE**-hard [K 2010]
g is Lipschitz	polyspace [Ko 1983]	can be **PSPACE**-hard [K 2010] (CCC 2009)
g is of class C^1		can be **PSPACE**-hard [K, Ota, Rösnick, Ziegler 2012] (MFCS 2012)
g is of class C^k		can be **CH**-hard [ibid.]
➤ g is analytic	polytime [Müller 1987]	——

Outline

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

1. Some complexity classes
   P, **NP**, **#P**, **PSPACE**, ...
2. Complexity of real functions
   How we define polytime real functions
3. Warm-up: Integration
   Assuming g is polytime and ignores the second argument, how complex can h be?
4. Lipschitz continuous ODE
   Assuming g is polytime and Lipschitz continuous, how complex can h be?
5. Final remarks
   Uniform versions / Summary

Outline

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

P the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that there is a polynomial-time TM M satisfying

$$A(x) = M(x).$$

P the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that there is a $\underbrace{\text{polynomial}}_{\text{in } |x|}$-time TM M satisfying $A(x) = M(x)$.

P the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that there is a $\underbrace{\text{polynomial}}_{\text{in } |x|}$-time TM M satisfying $A(x) = M(x)$.

PSPACE analogously, with polynomial-space.

NP and #P

NP

the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that there are a polynomial p and a polytime TM M satisfying

$$A(x) = \begin{cases} 1 & \text{if there is } y \in \Sigma^{p(|x|)} \text{ with } M(x,y) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

NP and #**P**

NP

the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that there are a polynomial p and a polytime TM M satisfying

$$A(x) = \begin{cases} 1 & \text{if there is } y \in \Sigma^{p(|x|)} \text{ with } M(x,y) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

An **NP**-complete problem: SAT

Tell whether a given (propositional) formula $\varphi(\vec{x})$ is satisfiable.

NP and **#P**

NP

the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that
there are a polynomial p and a polytime TM M satisfying

$$A(x) = \begin{cases} 1 & \text{if there is } y \in \Sigma^{p(|x|)} \text{ with } M(x,y) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

An **NP**-complete problem: SAT

Tell whether a given (propositional) formula $\varphi(\vec{x})$ is satisfiable.

#P

the class of functions $A\colon \Sigma^* \to \Sigma^*$ such that
there are a polynomial p and a polytime TM M satisfying

$$A(x) = \big(\text{the number of } y \in \Sigma^{p(|x|)} \text{ with } M(x,y) = 1\big).$$

NP and **#P**

NP

the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that
there are a polynomial p and a polytime TM M satisfying

$$A(x) = \begin{cases} 1 & \text{if there is } y \in \Sigma^{p(|x|)} \text{ with } M(x,y) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

An **NP**-complete problem: SAT

Tell whether a given (propositional) formula $\varphi(\vec{x})$ is satisfiable.

#P

the class of functions $A\colon \Sigma^* \to \Sigma^*$ such that
there are a polynomial p and a polytime TM M satisfying

$$A(x) = \big(\underbrace{\text{the number}}_{\text{(in binary notation)}} \text{ of } y \in \Sigma^{p(|x|)} \text{ with } M(x,y) = 1\big).$$

NP and #P

NP

the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that
there are a polynomial p and a polytime TM M satisfying

$$A(x) = \begin{cases} 1 & \text{if there is } y \in \Sigma^{p(|x|)} \text{ with } M(x,y) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

An **NP**-complete problem: SAT

Tell whether a given (propositional) formula $\varphi(\vec{x})$ is satisfiable.

#P

the class of functions $A\colon \Sigma^* \to \Sigma^*$ such that
there are a polynomial p and a polytime TM M satisfying

$$A(x) = \big(\underbrace{\text{the number}}_{\text{(in binary notation)}} \text{ of } y \in \Sigma^{p(|x|)} \text{ with } M(x,y) = 1\big).$$

A #**P**-complete problem: #SAT

Tell how many assignments satisfy the given formula $\varphi(\vec{x})$.

NP and #**P** relative to oracle B

$\mathbf{NP}^B$ the class of functions $A\colon \Sigma^* \to \{0,1\}$ such that there are a polynomial p and a polytime TM M satisfying

$$A(x) = \begin{cases} 1 & \text{if there is } y \in \Sigma^{p(|x|)} \text{ with } M^B(x,y) = 1, \\ 0 & \text{otherwise.} \end{cases}$$

An **NP**-complete problem: SAT

Tell whether a given (propositional) formula $\varphi(\vec{x})$ is satisfiable.

$\#\mathbf{P}^B$ the class of functions $A\colon \Sigma^* \to \Sigma^*$ such that there are a polynomial p and a polytime TM M satisfying

$$A(x) = \big(\underbrace{\text{the number}}_{\text{(in binary notation)}} \text{ of } y \in \Sigma^{p(|x|)} \text{ with } M^B(x,y) = 1\big).$$

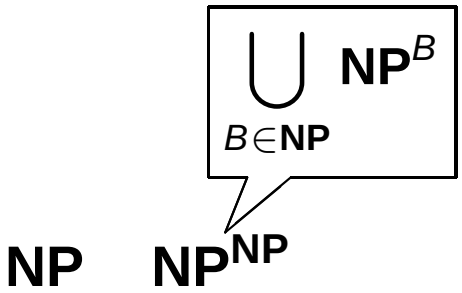
A #**P**-complete problem: #SAT

Tell how many assignments satisfy the given formula $\varphi(\vec{x})$.

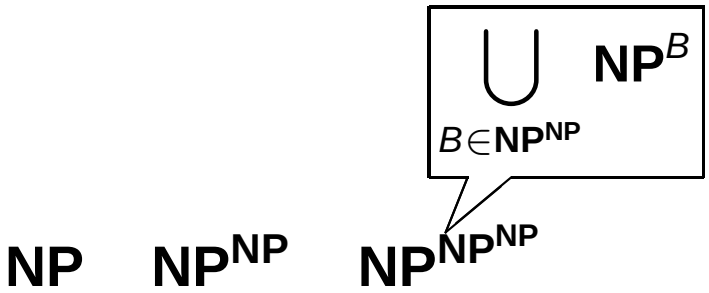
The hierarchies

NP

The hierarchies



The hierarchies



The hierarchies

$$\mathbf{PH} := \mathbf{NP} \cup \mathbf{NP}^{\mathbf{NP}} \cup \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \cup \cdots$$

The hierarchies

$$\mathbf{PH} := \mathbf{NP} \cup \mathbf{NP}^{\mathbf{NP}} \cup \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \cup \cdots$$

An $\mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$-complete problem

Tell whether a given formula of the form
$\exists \vec{x}_1 \forall \vec{x}_2 \exists \vec{x}_3 \varphi(\vec{x}_1 \vec{x}_2 \vec{x}_3)$ is true.

The hierarchies

$$\mathbf{PH} := \mathbf{NP} \cup \mathbf{NP}^{\mathbf{NP}} \cup \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \cup \cdots$$
$$\mathbf{CH} := \#\mathbf{P} \cup \#\mathbf{P}^{\#\mathbf{P}} \cup \#\mathbf{P}^{\#\mathbf{P}^{\#\mathbf{P}}} \cup \cdots$$

An $\mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$-complete problem

Tell whether a given formula of the form
$\exists \vec{x}_1 \forall \vec{x}_2 \exists \vec{x}_3 \varphi(\vec{x}_1 \vec{x}_2 \vec{x}_3)$ is true.

The hierarchies

$$\mathbf{PH} := \mathbf{NP} \cup \mathbf{NP}^{\mathbf{NP}} \cup \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \cup \cdots \subseteq \mathbf{PSPACE}$$

$$\mathbf{CH} := \#\mathbf{P} \cup \#\mathbf{P}^{\#\mathbf{P}} \cup \#\mathbf{P}^{\#\mathbf{P}^{\#\mathbf{P}}} \cup \cdots \subseteq \text{(function version of) } \mathbf{PSPACE}$$

The hierarchies

$$\mathbf{PH} := \mathbf{NP} \cup \mathbf{NP}^{\mathbf{NP}} \cup \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \cup \cdots \subseteq \mathbf{PSPACE}$$

$$\mathbf{CH} := \#\mathbf{P} \cup \#\mathbf{P}^{\#\mathbf{P}} \cup \#\mathbf{P}^{\#\mathbf{P}^{\#\mathbf{P}}} \cup \cdots \subseteq \text{(function version of) } \mathbf{PSPACE}$$

A **PSPACE**-complete problem: QBF

Tell whether a given formula of the form
$\exists \vec{x}_1 \forall \vec{x}_2 \exists \vec{x}_3 \forall \vec{x}_4 \ldots \varphi(\vec{x}_1 \vec{x}_2 \vec{x}_3 \ldots \vec{x}_n)$ is true.

The hierarchies

$$\mathbf{PH} := \mathbf{NP} \cup \mathbf{NP}^{\mathbf{NP}} \cup \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \cup \cdots \subseteq \mathbf{PSPACE}$$
$$\mathbf{CH} := \#\mathbf{P} \cup \#\mathbf{P}^{\#\mathbf{P}} \cup \#\mathbf{P}^{\#\mathbf{P}^{\#\mathbf{P}}} \cup \cdots \subseteq \text{(function version of) } \mathbf{PSPACE}$$

Toda 1991: $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$.

A **PSPACE**-complete problem: QBF

Tell whether a given formula of the form
$\exists \vec{x}_1 \forall \vec{x}_2 \exists \vec{x}_3 \forall \vec{x}_4 \ldots \varphi(\vec{x}_1 \vec{x}_2 \vec{x}_3 \ldots \vec{x}_n)$ is true.

The hierarchies

$\mathbf{PH} := \mathbf{NP} \cup \mathbf{NP}^{\mathbf{NP}} \cup \mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}} \cup \cdots \subseteq \mathbf{PSPACE}$
$\mathbf{CH} := \#\mathbf{P} \cup \#\mathbf{P}^{\#\mathbf{P}} \cup \#\mathbf{P}^{\#\mathbf{P}^{\#\mathbf{P}}} \cup \cdots \subseteq$ (function version of) $\mathbf{PSPACE}$

Toda 1991: $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$.

General belief: These hierarchies do not collapse.
But we don't even have a proof of $\mathbf{P} \neq \mathbf{PSPACE}$.

A **PSPACE**-complete problem: QBF

Tell whether a given formula of the form
$\exists \vec{x}_1 \forall \vec{x}_2 \exists \vec{x}_3 \forall \vec{x}_4 \ldots \varphi(\vec{x}_1 \vec{x}_2 \vec{x}_3 \ldots \vec{x}_n)$ is true.

Outline

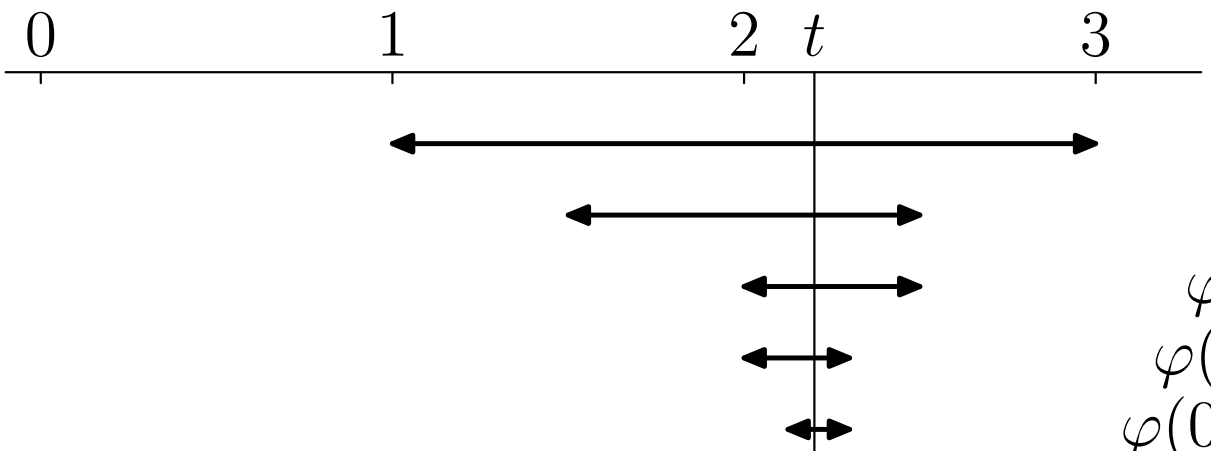
$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

1. Some complexity classes
   P, **NP**, **#P**, **PSPACE**, ...
2. Complexity of real functions
   How we define polytime real functions
3. Warm-up: Integration
   Assuming g is polytime and ignores the second argument, how complex can h be?
4. Lipschitz continuous ODE
   Assuming g is polytime and Lipschitz continuous, how complex can h be?
5. Final remarks
   Uniform versions / Summary

Representation of real numbers

Definition

We say that $\varphi\colon \Sigma^* \to \Sigma^*$ is a name of $t \in \mathbf{R}$ if for each $n \in \mathbf{N}$, $\varphi(0^n)$ is (the binary expansion of) t rounded up or down at the nth bit below the point.



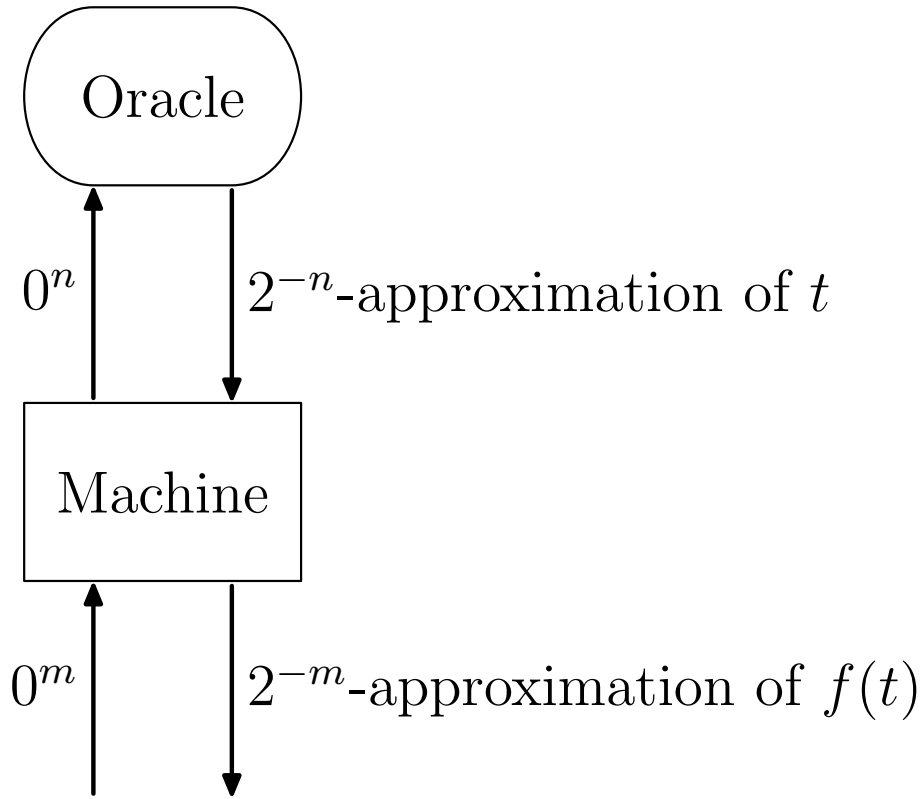
$$\begin{aligned}
\varphi(\varepsilon) &= +10. \\
\varphi(0) &= +10.0 \\
\varphi(00) &= +10.01 \\
\varphi(000) &= +10.001 \\
\varphi(0000) &= +10.0011 \\
&\vdots
\end{aligned}$$

(Each $\varphi(0^n)$ approximates t with precision 2^{-n})

Computing real functions

Definition

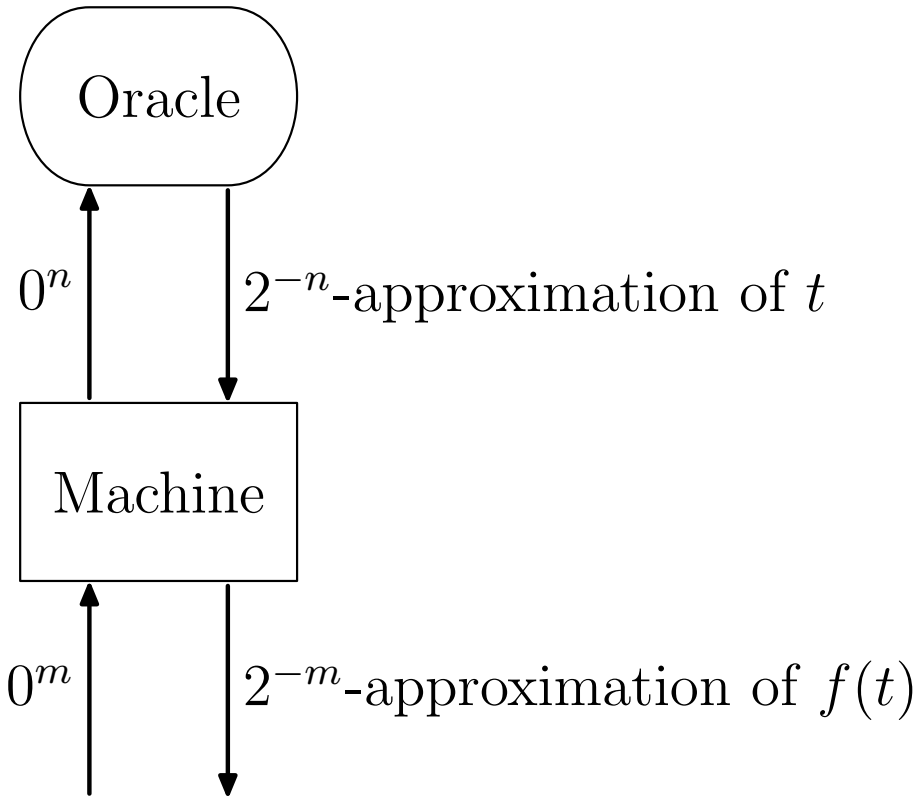
Machine M computes $f : [0,1] \to \mathbf{R}$ if, for any name φ of any $t \in [0,1]$, M^{φ} computes a name of $f(t)$.



Computing real functions

Definition

Machine M computes $f : [0,1] \to \mathbf{R}$ if, for any name φ of any $t \in [0,1]$, M^{φ} computes a name of $f(t)$.



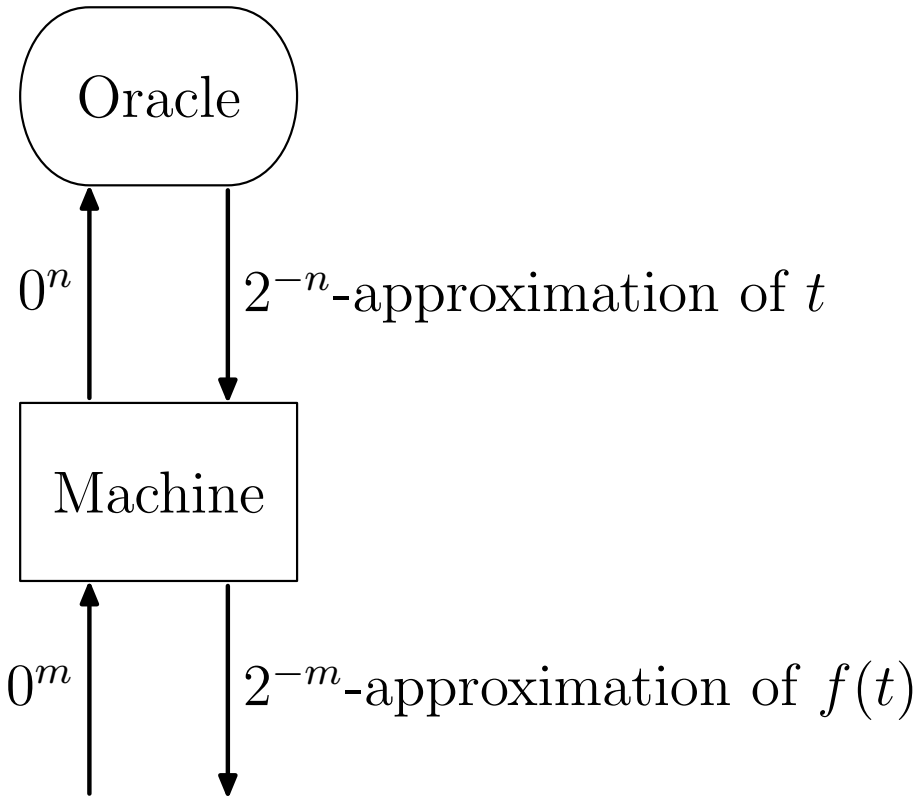
In other words,

- M Turing-reduces $f(t)$ to t.
- Given access to approximations of t to any precision, the machine yields $f(t)$ to any precision.

Computing real functions

Definition

Machine M computes $f : [0,1] \to \mathbf{R}$ if, for any name φ of any $t \in [0,1]$, M^{φ} computes a name of $f(t)$.



In other words,

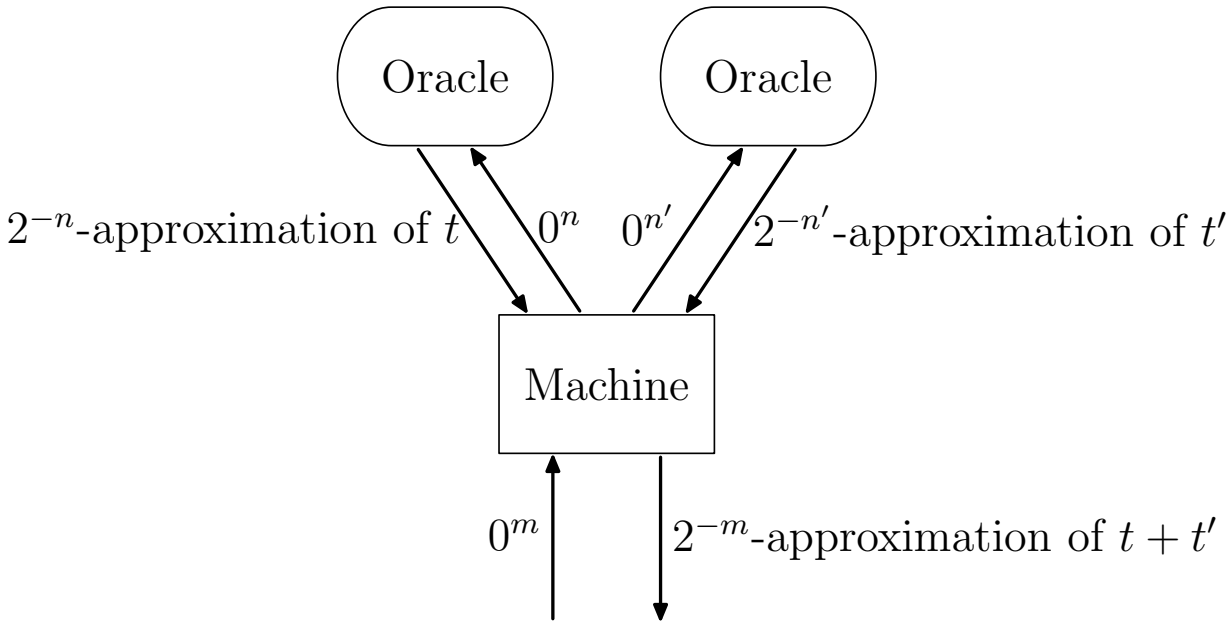
- M Turing-reduces $f(t)$ to t.
- Given access to approximations of t to any precision, the machine yields $f(t)$ to any precision.

Note: Saying M is polytime means that it halts in time poly(m).
In particular, $n \leq$ poly(m).

Example 1

Addition $+\colon [0,1] \times [0,1] \to \mathbf{R}$ is polytime.

To do: Given (names of) t and t' as oracles and 0^m as input, output a 2^{-m}-approximation of $t + t'$.



- Ask the oracles for 2^{-m-2}-approximations of t and t'.
- Add the answers (as rational numbers).
- Output the closest rational number to the sum that has m bits below the point.

Example 2

$\exp\colon [0,1] \to \mathbf{R}$ is polytime.

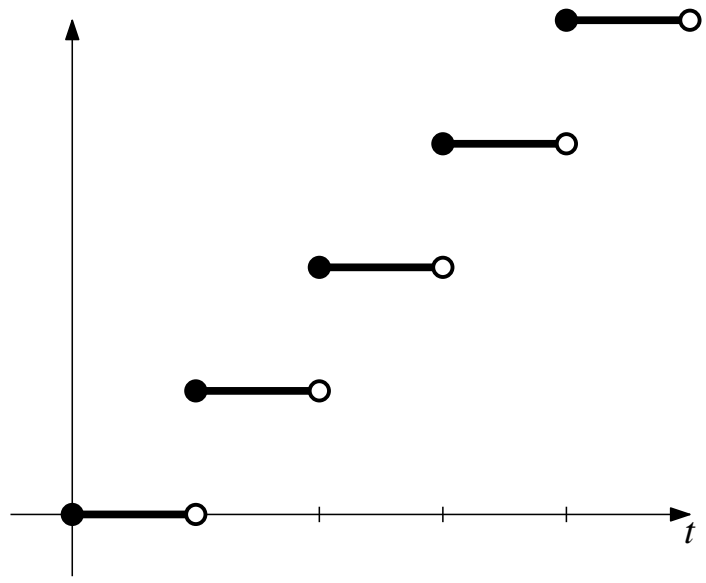
To get a 2^{-m}-approximation of the value

$$\exp t = \frac{1}{0!} + \frac{t}{1!} + \frac{t^2}{2!} + \frac{t^3}{3!} + \cdots,$$

it suffices to compute a $2^{-m}/2$-approximation of the sum of the first m terms (because the remaining terms add up to at most $2^{-m}/2$).

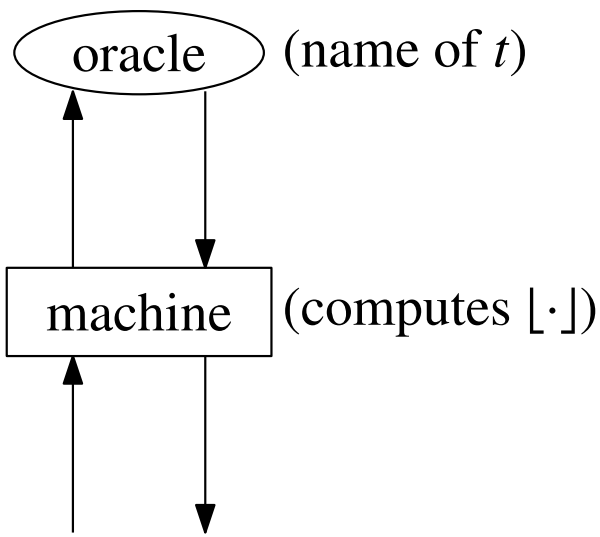
Example 3

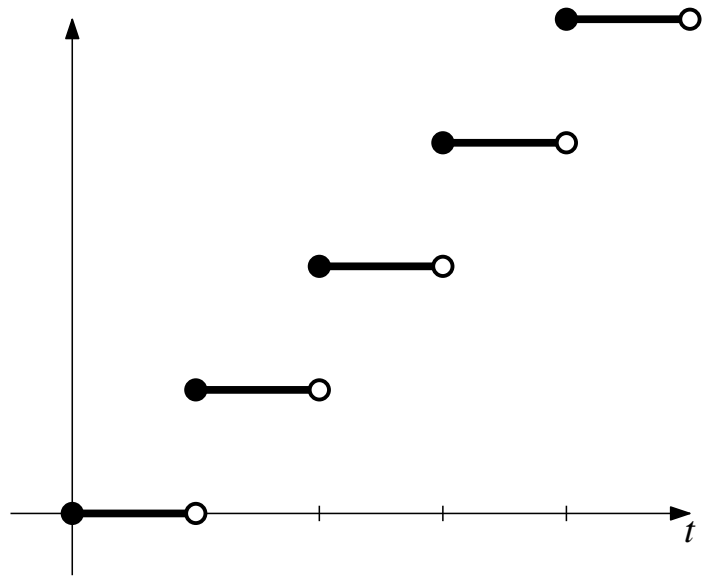
$\lfloor \cdot \rfloor : [0, 5] \to \mathbf{R}$ is not computable.



Example 3

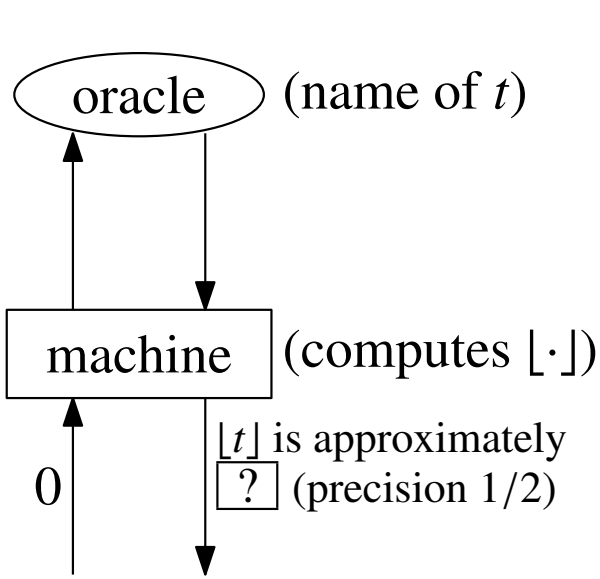
$\lfloor \cdot \rfloor : [0, 5] \to \mathbf{R}$ is not computable.

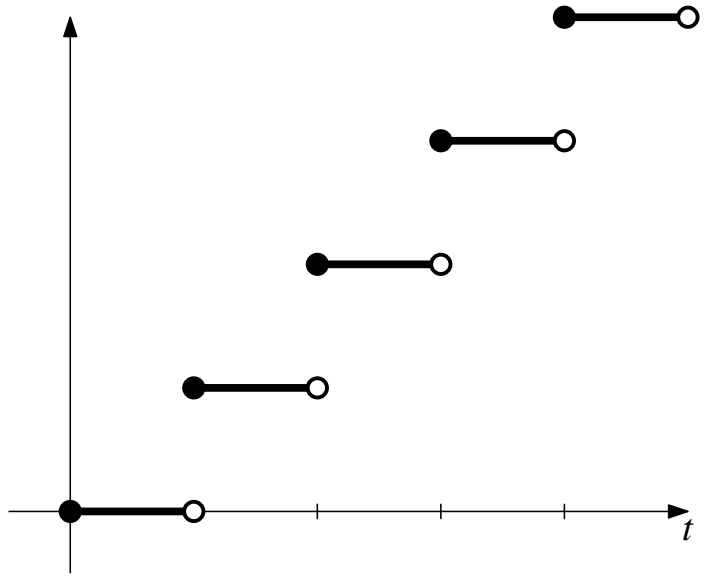




Example 3

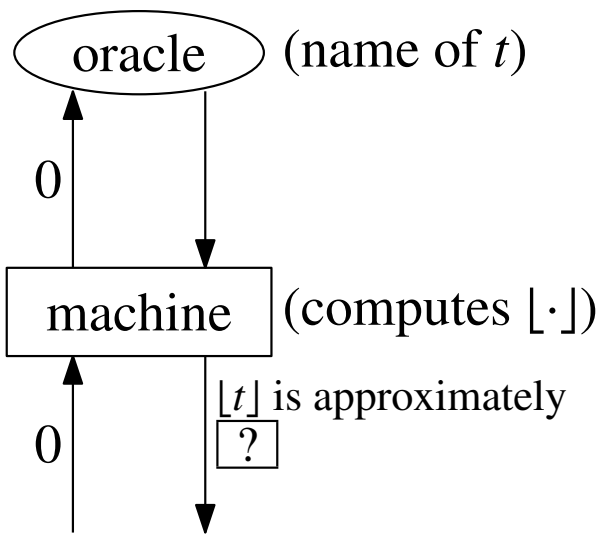
$\lfloor \cdot \rfloor : [0,5] \to \mathbf{R}$ is not computable.

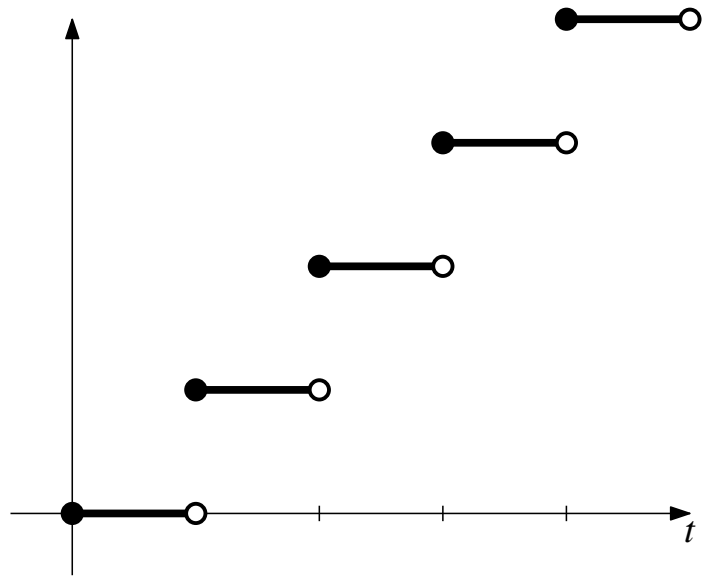




Example 3

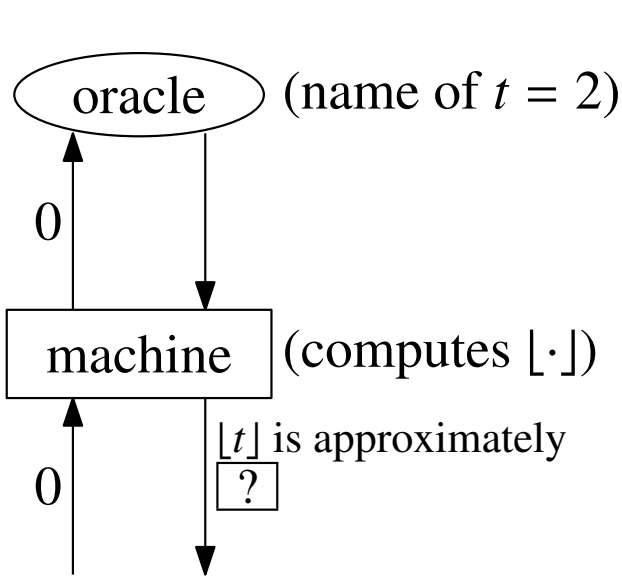
$\lfloor \cdot \rfloor : [0,5] \to \mathbf{R}$ is not computable.

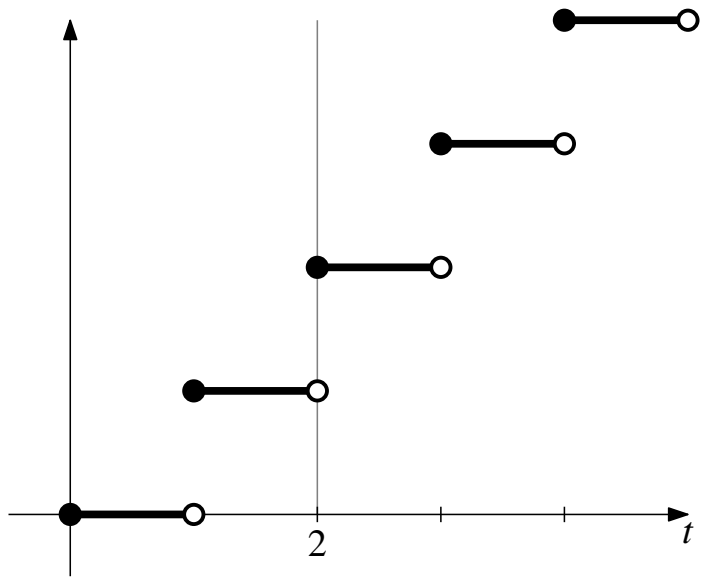




Example 3

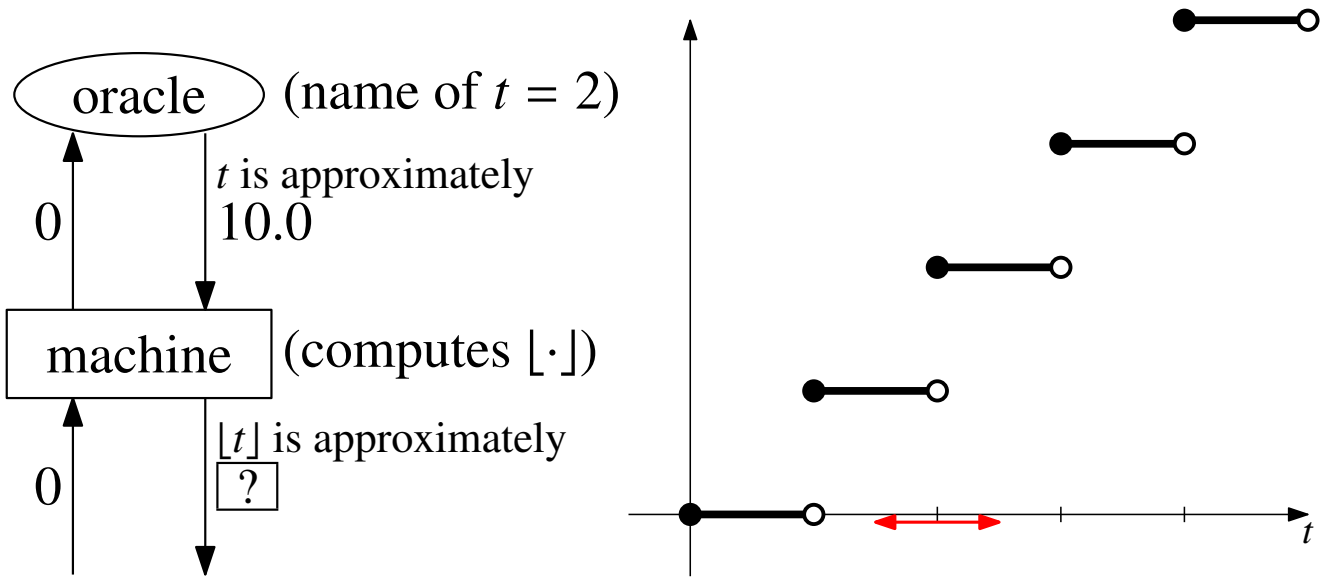
$\lfloor \cdot \rfloor : [0, 5] \to \mathbf{R}$ is not computable.





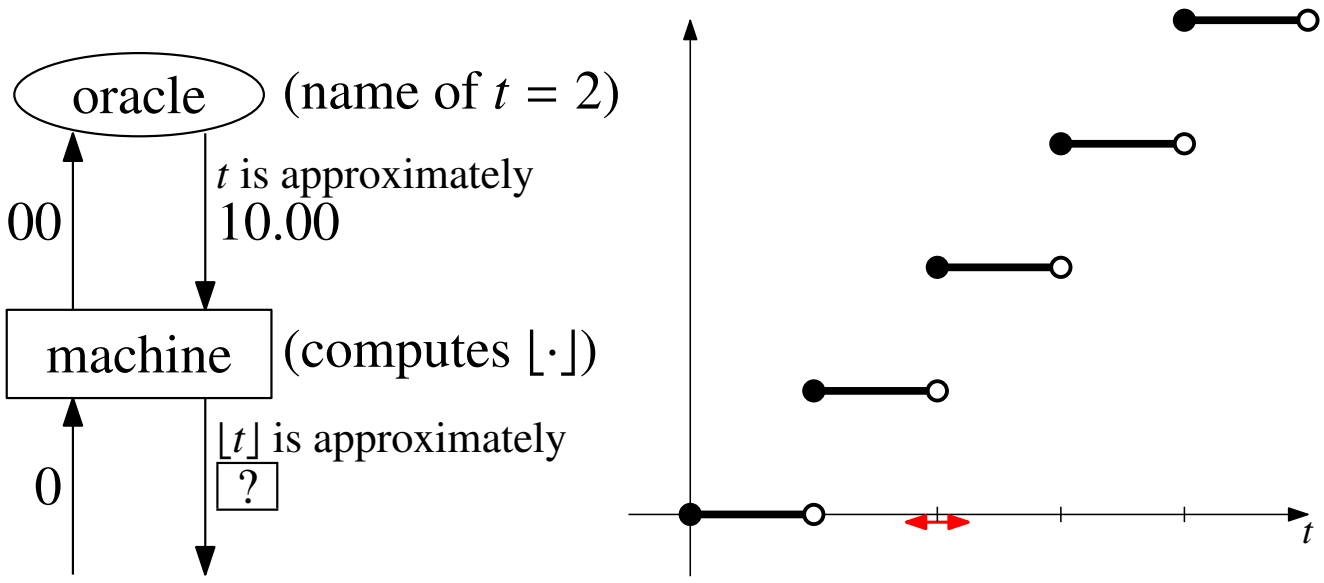
Example 3

$\lfloor \cdot \rfloor : [0, 5] \to \mathbf{R}$ is not computable.



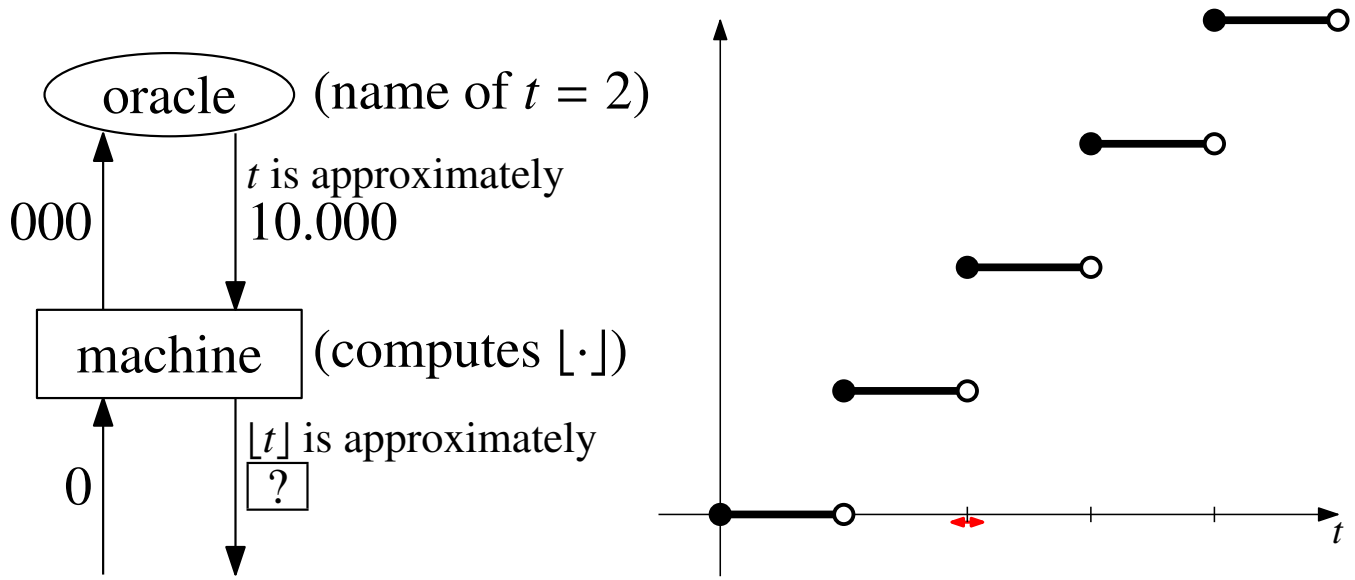
Example 3

$\lfloor \cdot \rfloor : [0,5] \to \mathbf{R}$ is not computable.



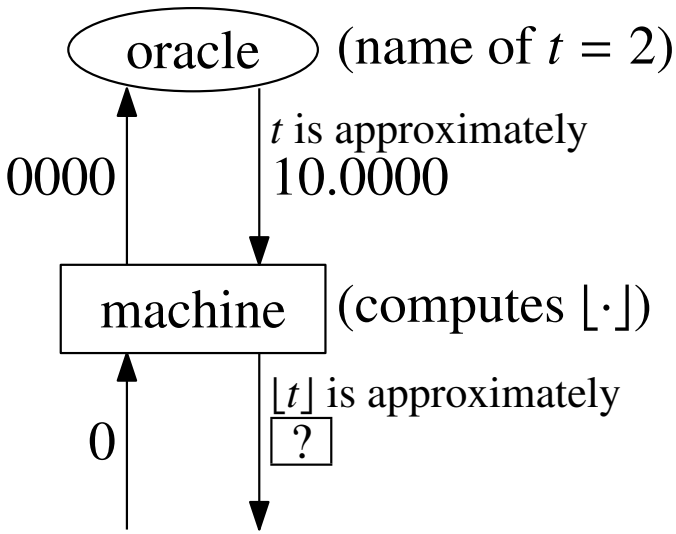
Example 3

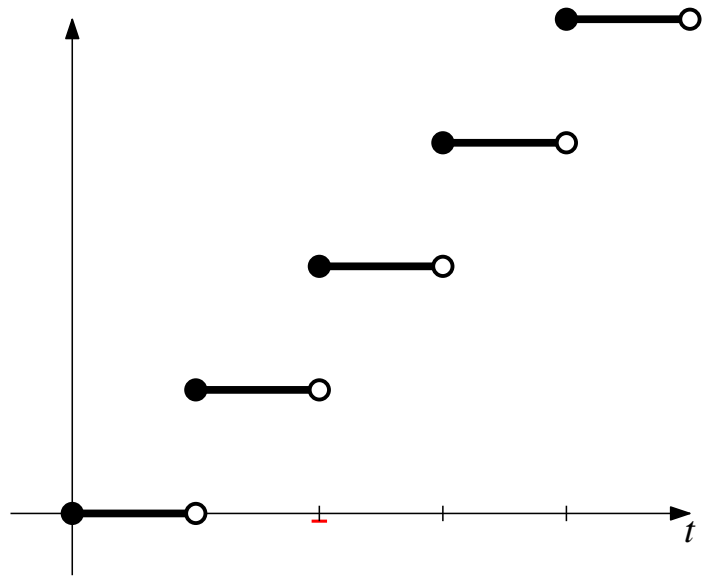
$\lfloor \cdot \rfloor : [0, 5] \to \mathbf{R}$ is not computable.



Example 3

$\lfloor \cdot \rfloor : [0, 5] \to \mathbf{R}$ is not computable.





Computable functions are continuous

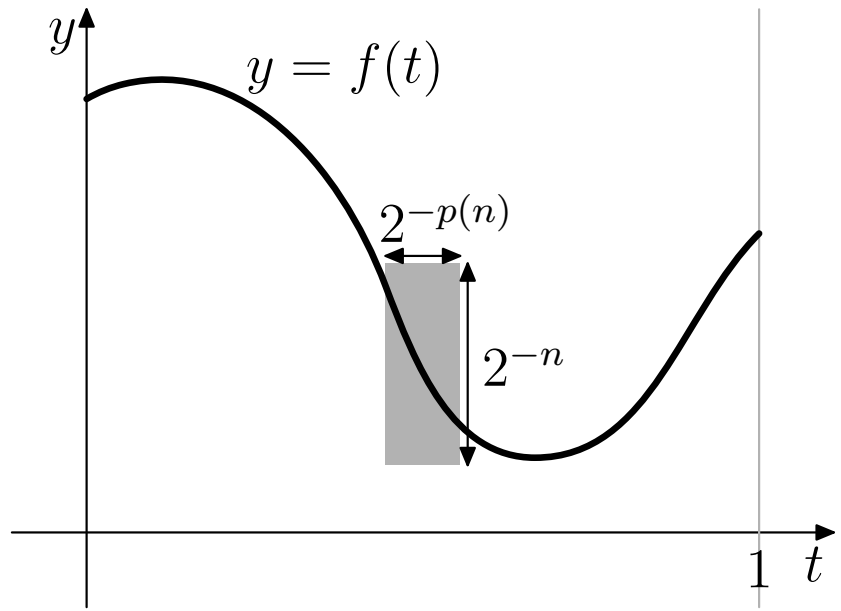
Theorem

- A computable real function is continuous.
- A polytime real function has a polynomial modulus of continuity.

Modulus of continuity p:

$|t - t'| < 2^{-p(n)}$

$\Longrightarrow |f(t) - f(t')| < 2^{-n}$.



Modulus of continuity

Theorem

$f : [0,1] \to \mathbf{R}$ is polytime iff

- f has a polynomial modulus of continuity, and
- there is a function $\varphi : (\mathbf{Q} \cap [0,1]) \times \{0\}^* \to \mathbf{Q}$ in **FP** such that $|\varphi(u, 0^n) - f(u)| \leq 2^{-n}$ for each u and n.

Modulus of continuity

Theorem

$f : [0,1] \to \mathbf{R}$ is polytime iff

- f has a polynomial modulus of continuity, and
- there is a function $\varphi : (\mathbf{Q} \cap [0,1]) \times \{0\}^* \to \mathbf{Q}$ in **FP** such that $|\varphi(u, 0^n) - f(u)| \leq 2^{-n}$ for each u and n.

Example: For $A \subseteq \{0,1\}^*$, define f_A:

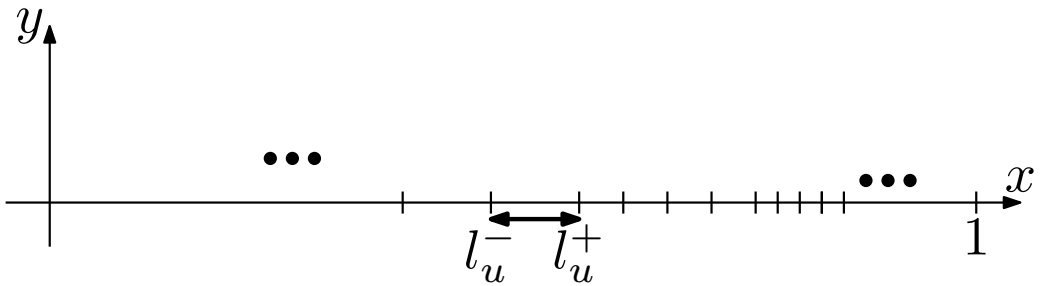
Modulus of continuity

Theorem

$f : [0,1] \to \mathbf{R}$ is polytime iff

- f has a polynomial modulus of continuity, and
- there is a function $\varphi : (\mathbf{Q} \cap [0,1]) \times \{0\}^* \to \mathbf{Q}$ in **FP** such that $|\varphi(u, 0^n) - f(u)| \leq 2^{-n}$ for each u and n.

Example: For $A \subseteq \{0,1\}^*$, define f_A:



Assign interval $[l_u^-, l_u^+]$ (of length $\approx 2^{-\Theta(|u|)}$) for each $u \in \Sigma^*$.

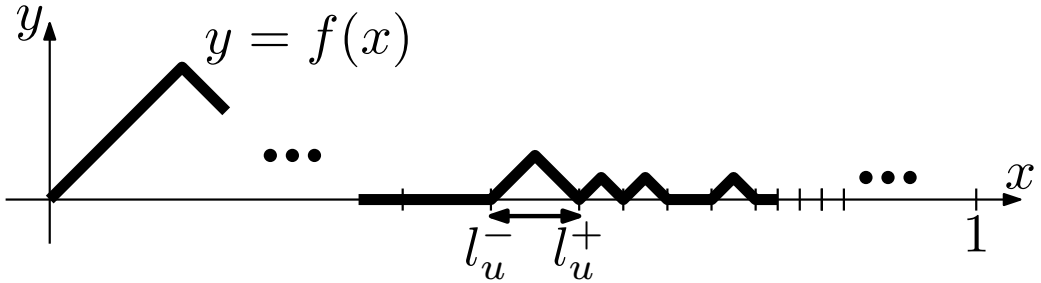
Modulus of continuity

Theorem

$f : [0,1] \to \mathbf{R}$ is polytime iff

- f has a polynomial modulus of continuity, and
- there is a function $\varphi : (\mathbf{Q} \cap [0,1]) \times \{0\}^* \to \mathbf{Q}$ in **FP** such that $|\varphi(u, 0^n) - f(u)| \leq 2^{-n}$ for each u and n.

Example: For $A \subseteq \{0,1\}^*$, define f_A:



Assign interval $[l_u^-, l_u^+]$ (of length $\approx 2^{-\Theta(|u|)}$) for each $u \in \Sigma^*$. Make a bump iff $u \in A$.

Then f_A is polytime iff $A \in \mathbf{P}$.

Outline

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

1. Some complexity classes
   P, **NP**, **#P**, **PSPACE**, ...
2. Complexity of real functions
   How we define polytime real functions
3. Warm-up: Integration
   Assuming g is polytime and ignores the second argument, how complex can h be?
4. Lipschitz continuous ODE
   Assuming g is polytime and Lipschitz continuous, how complex can h be?
5. Final remarks
   Uniform versions / Summary

Complexity of integration

Theorem (essentially [Friedman 1984])

There are $g\colon [0,1] \to \mathbf{R}$ and $h\colon [0,1] \to \mathbf{R}$ such that

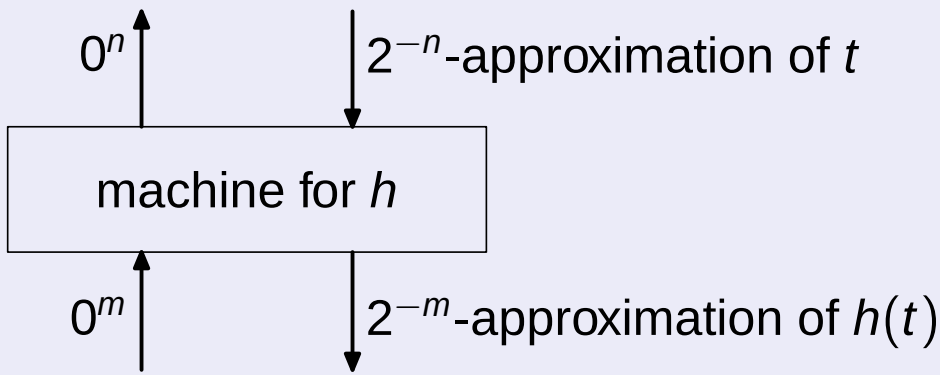
- g is polytime;
- $h(0) = 0,\ h'(t) = g(t)$;
- h is #**P**-hard.

Complexity of integration

Theorem (essentially [Friedman 1984])

There are $g\colon [0,1] \to \mathbf{R}$ and $h\colon [0,1] \to \mathbf{R}$ such that

- g is polytime;
- $h(0) = 0,\ h'(t) = g(t)$;
- h is #**P**-hard, in the sense that

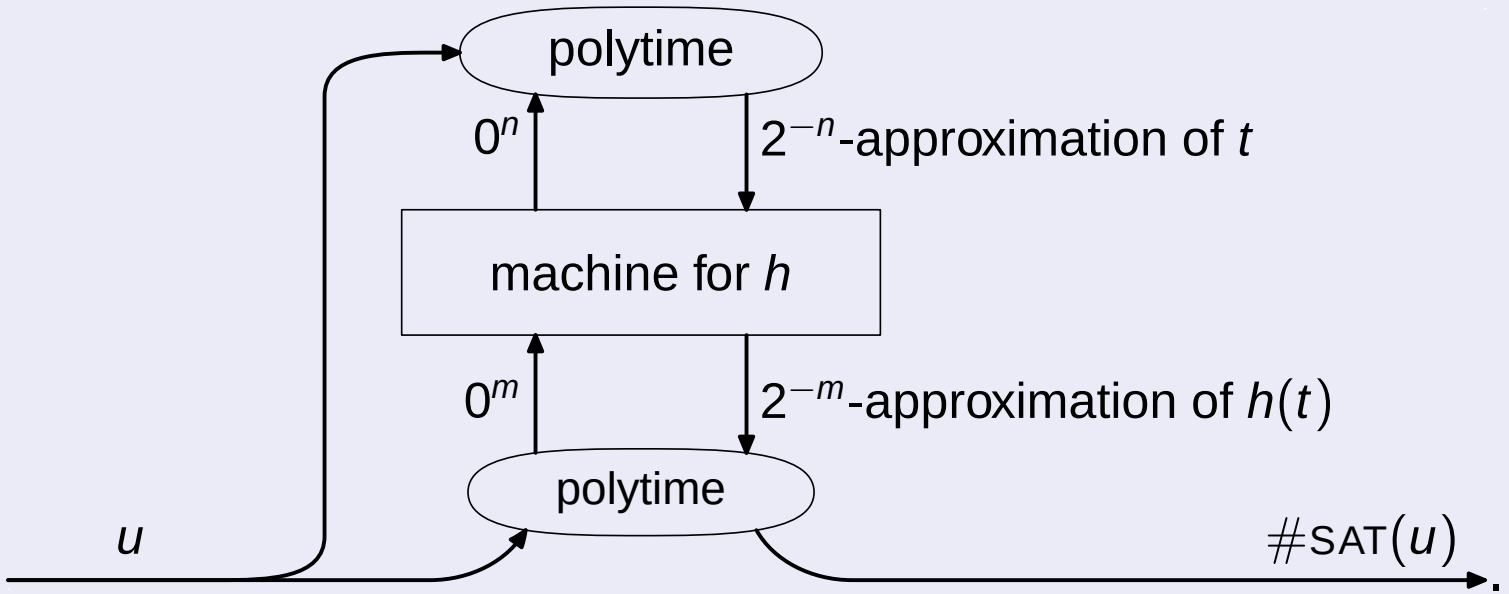


Complexity of integration

Theorem (essentially [Friedman 1984])

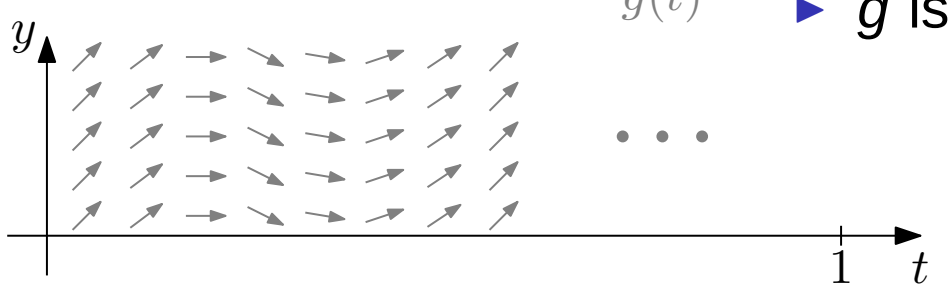
There are $g\colon [0,1] \to \mathbf{R}$ and $h\colon [0,1] \to \mathbf{R}$ such that

- g is polytime;
- $h(0) = 0,\ h'(t) = g(t)$;
- h is #**P**-hard, in the sense that



$\#\mathrm{SAT}(u)$ = number of satisfying assignments of formula u

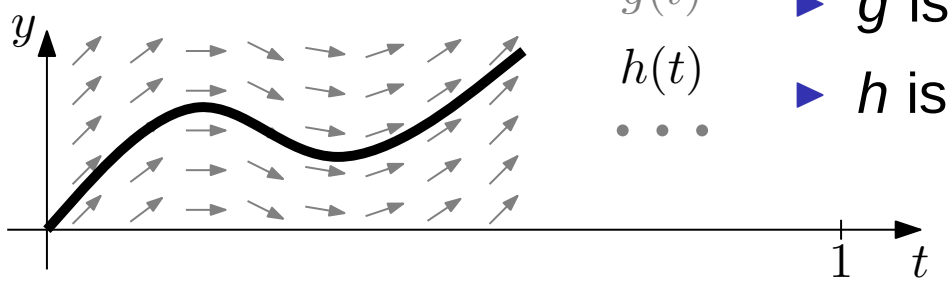
Reducing #SAT to integration



What we want:

- g is easy (polytime);

Reducing #SAT to integration



$g(t)$

$h(t)$

• • •

What we want:

- g is easy (polytime);
- h is hard (#SAT reduces to it).

Reducing #SAT to integration

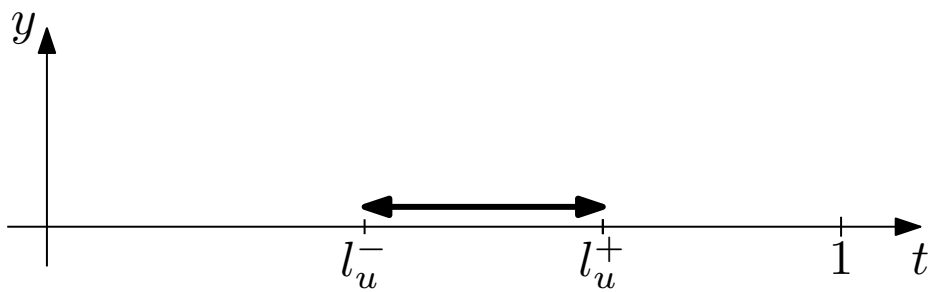
Reducing $\#\mathsf{SAT}$ to integration

$\#\mathsf{SAT}(u) =$ number of v
with $(u, v) \in R$
(for some $R \in \mathbf{P}$).

Reducing $\#\mathsf{SAT}$ to integration

$\#\mathsf{SAT}(u) =$ number of v
with $(u, v) \in R$
(for some $R \in \mathbf{P}$).

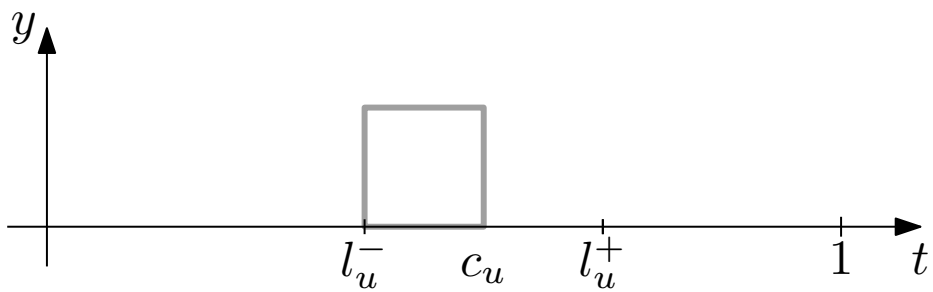
For each $u \in \Sigma^*$,
assign interval $[l_u^-, l_u^+]$.



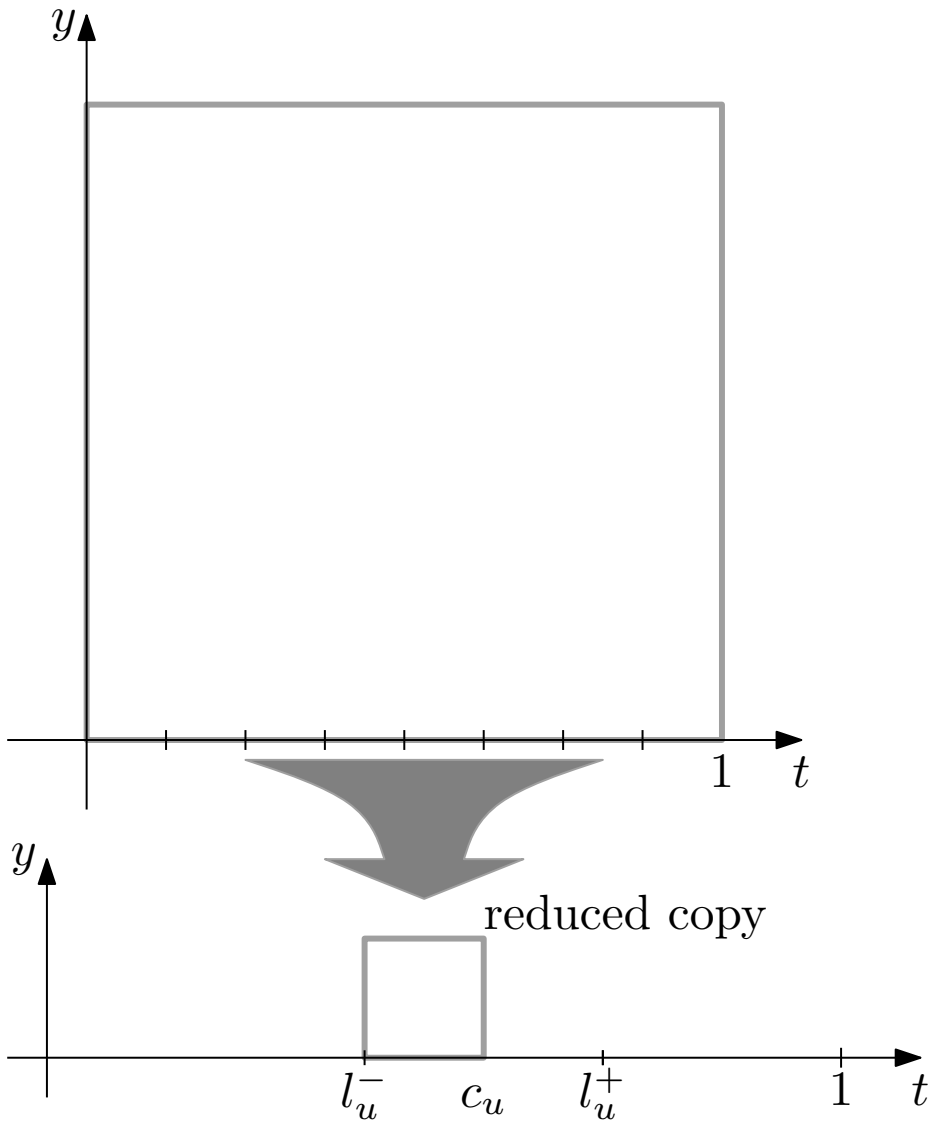
Reducing #SAT to integration

$\#\mathrm{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$.



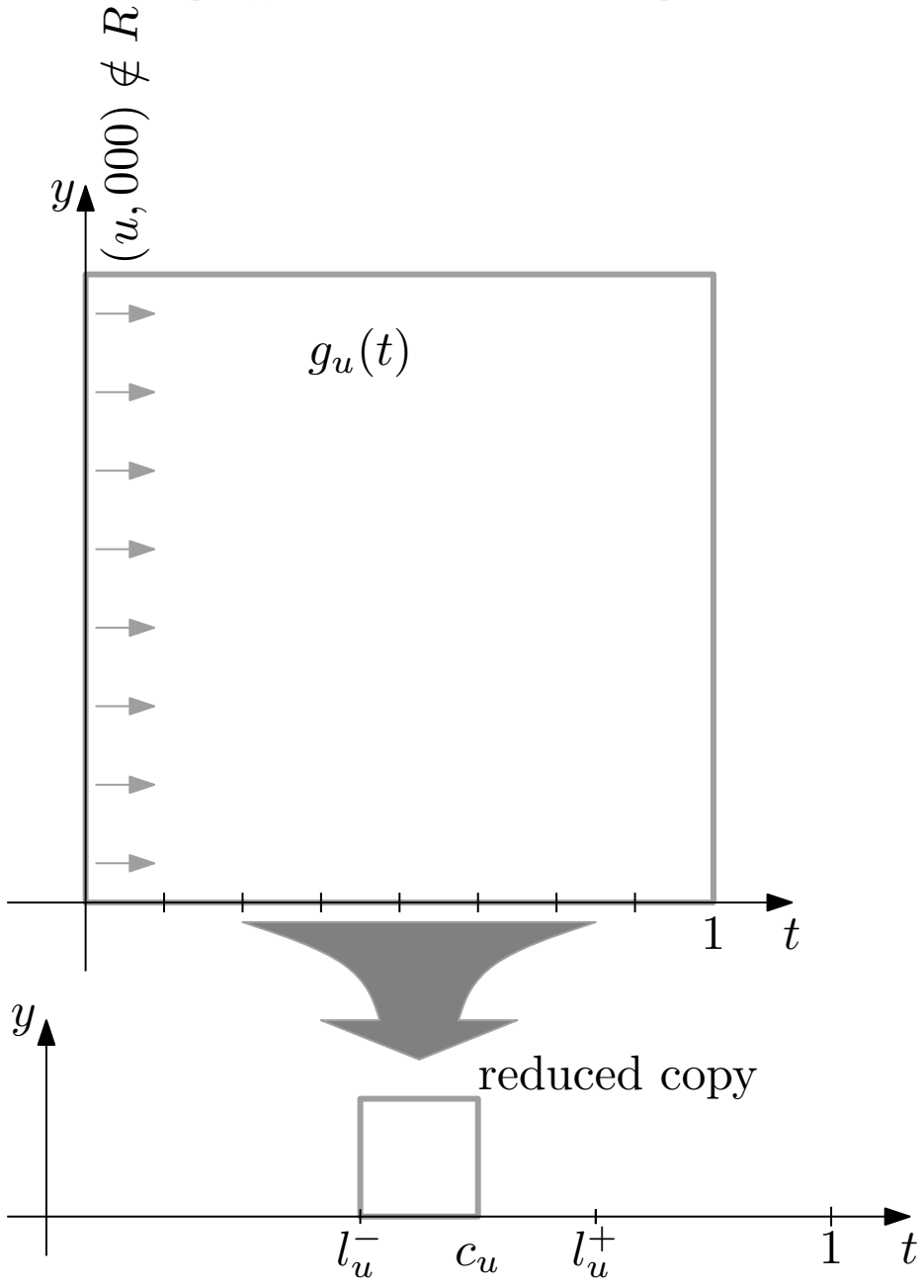
Reducing #SAT to integration



$\#\mathrm{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

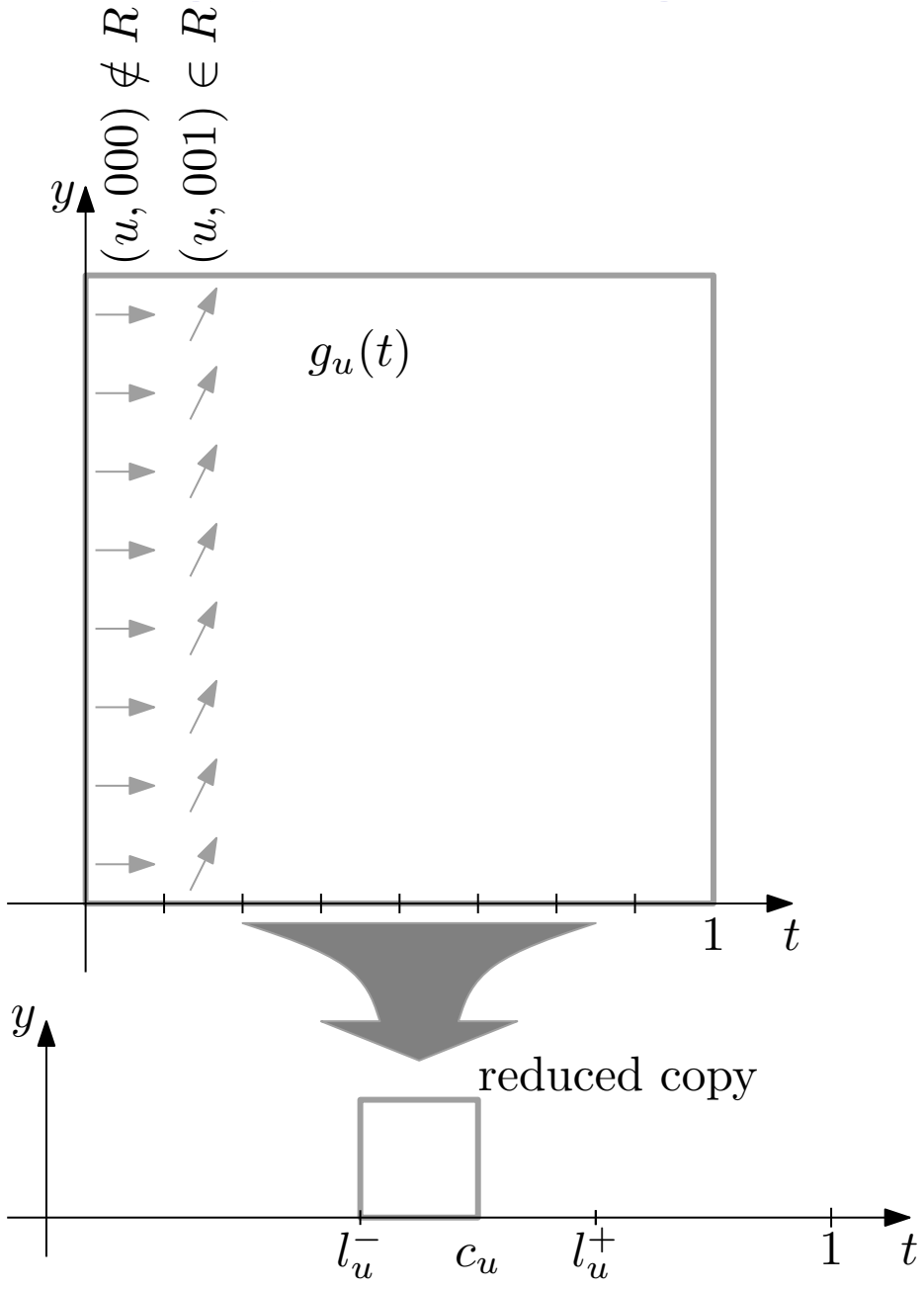
Reducing #SAT to integration



$\#\mathrm{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

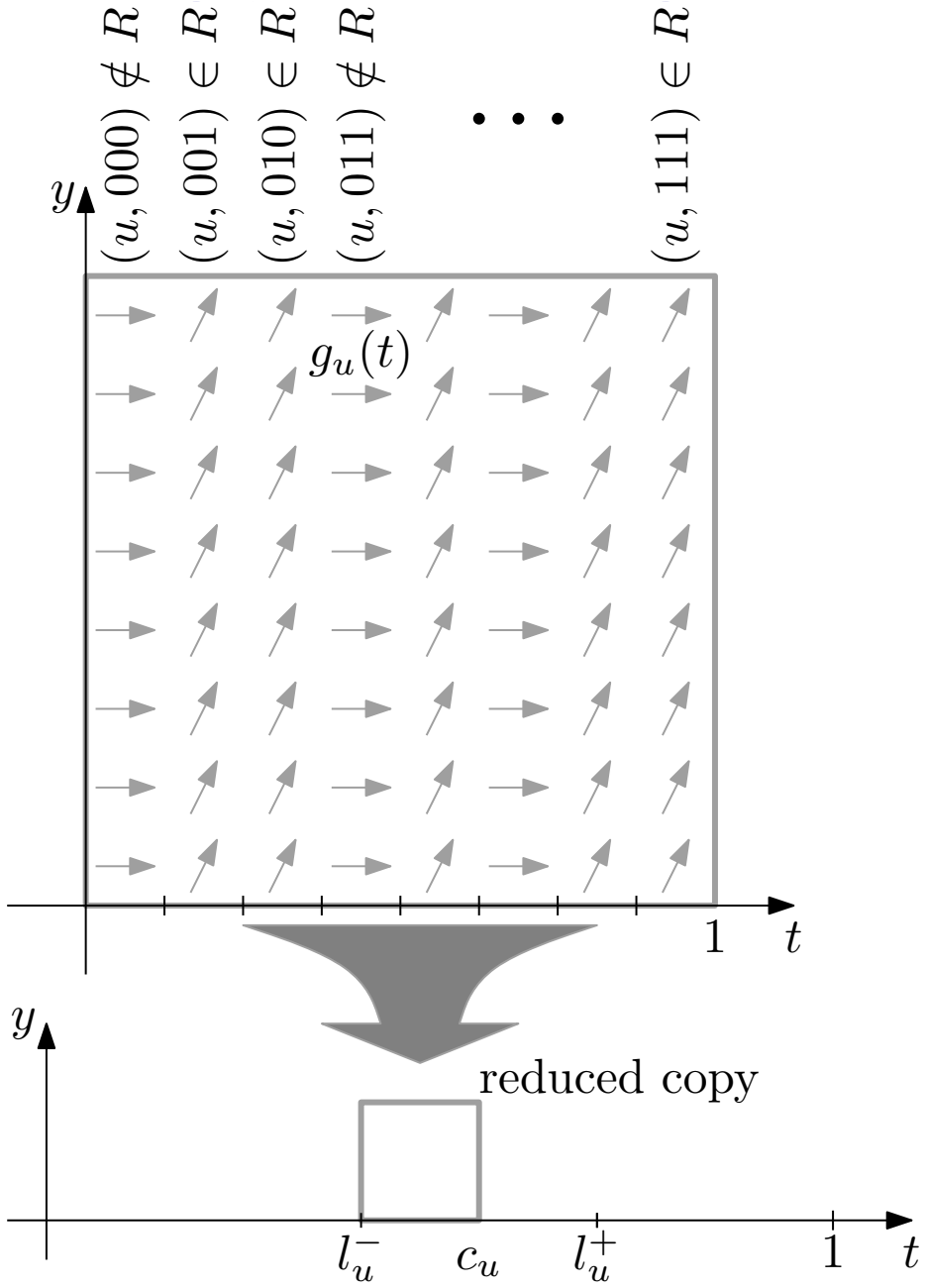
Reducing #SAT to integration



$\#\mathrm{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

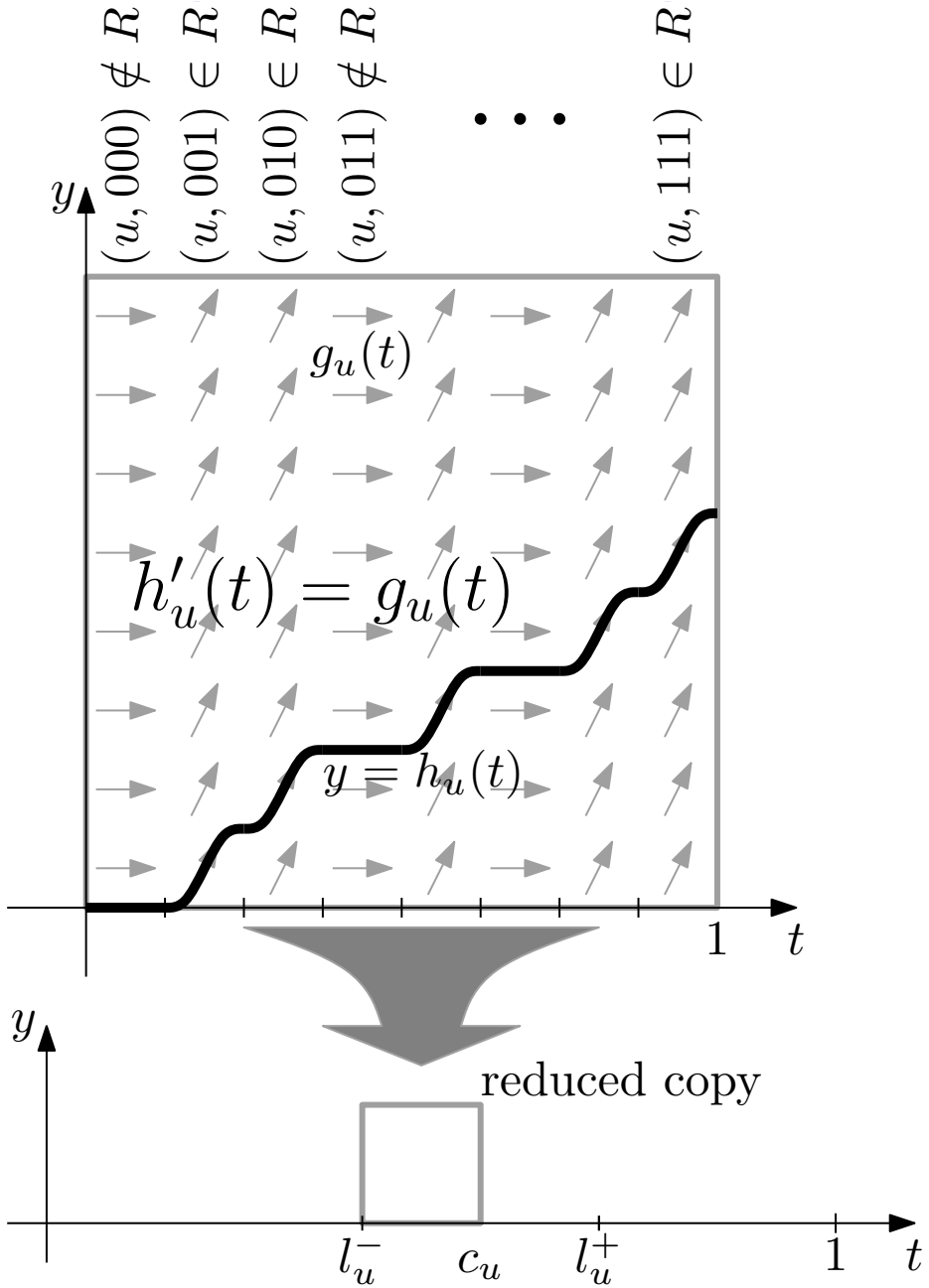
Reducing #SAT to integration



$\#\mathrm{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

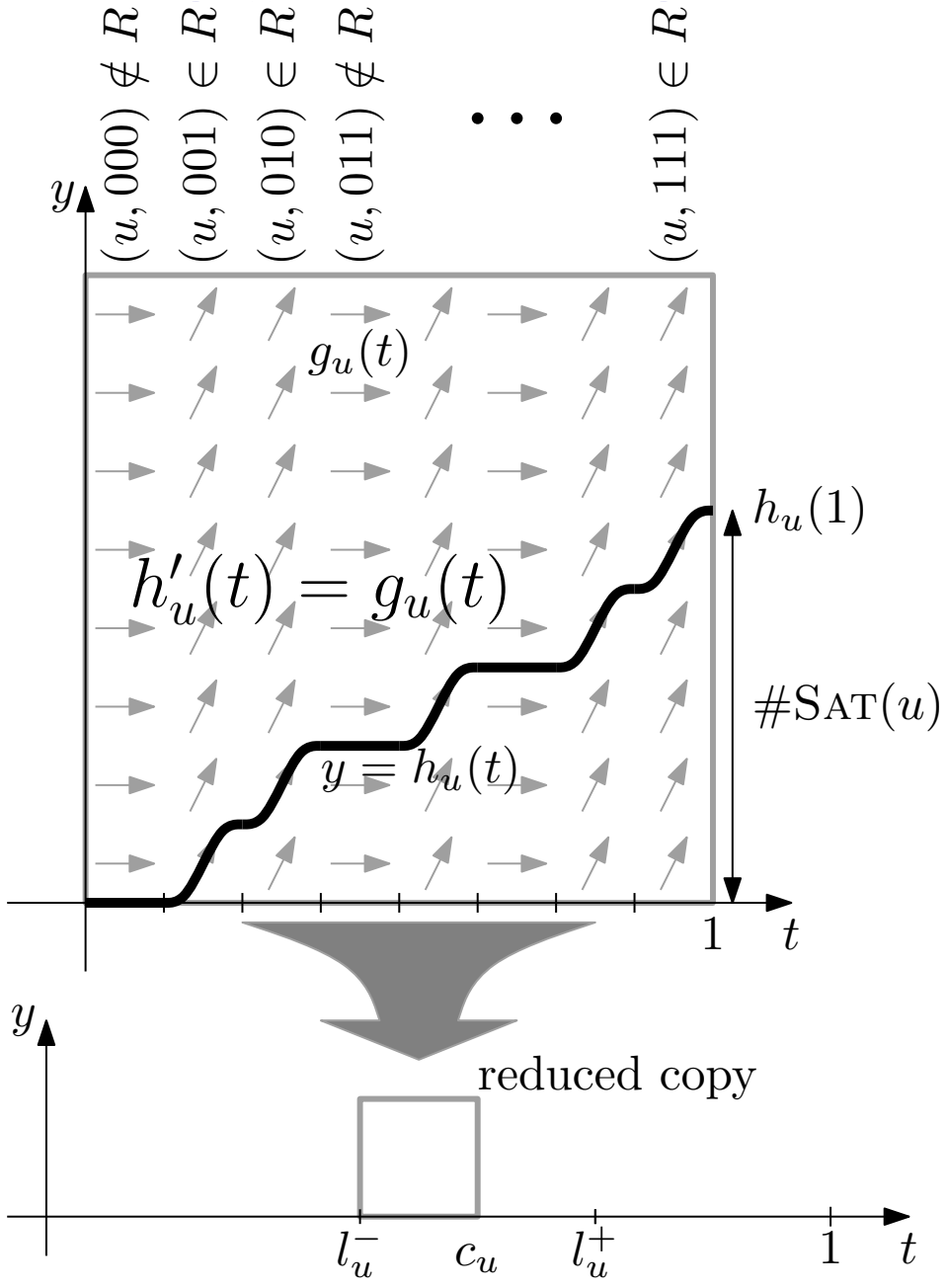
Reducing #SAT to integration



$\#\mathrm{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

Reducing #SAT to integration

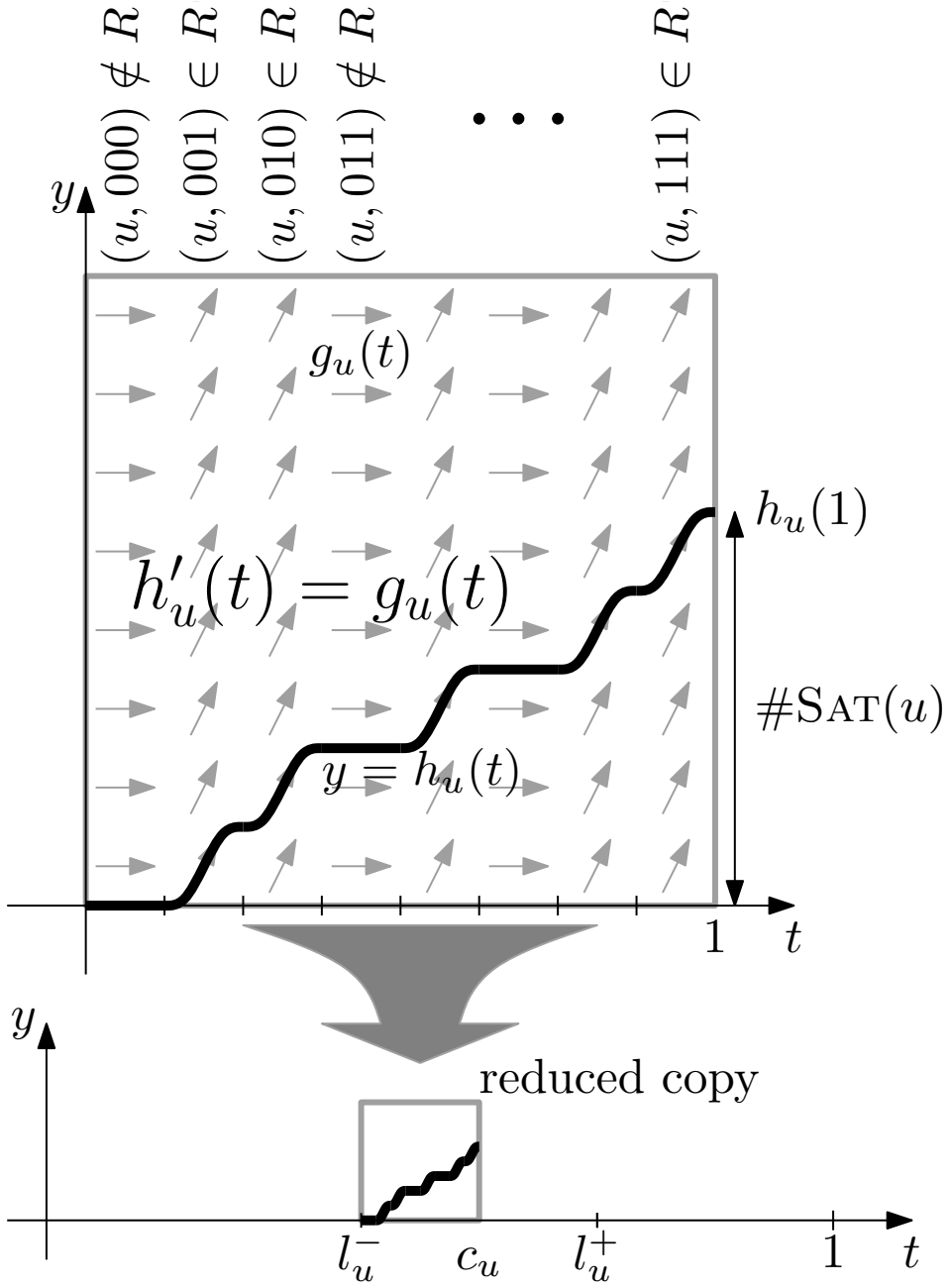


$\#\text{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

$\#\text{SAT}(u)$ is encoded in $h_u(1)$.

Reducing #SAT to integration

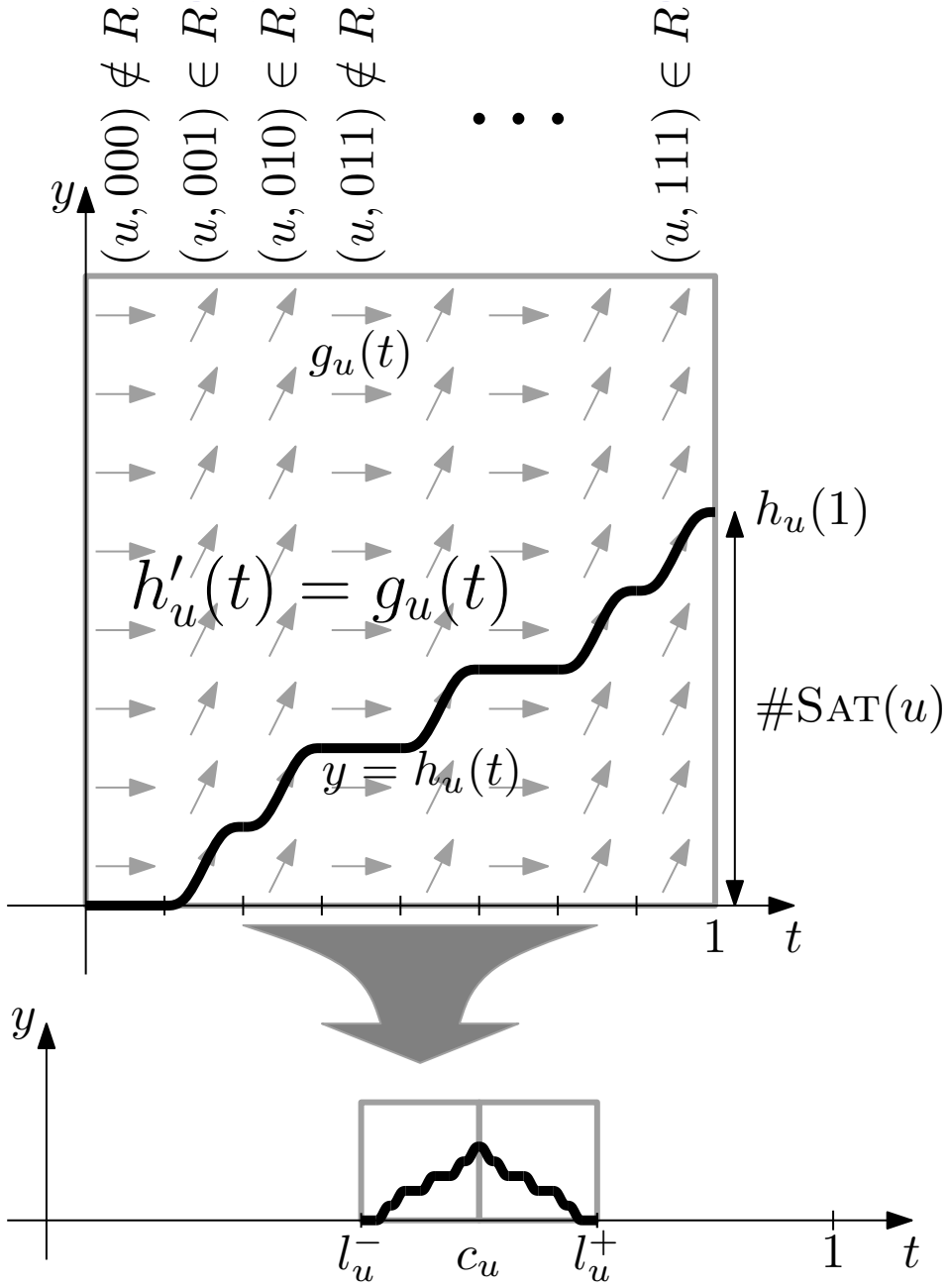


$\#\mathrm{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

$\#\mathrm{SAT}(u)$ is encoded in $h_u(1)$, or in $h(c_u)$.

Reducing #SAT to integration

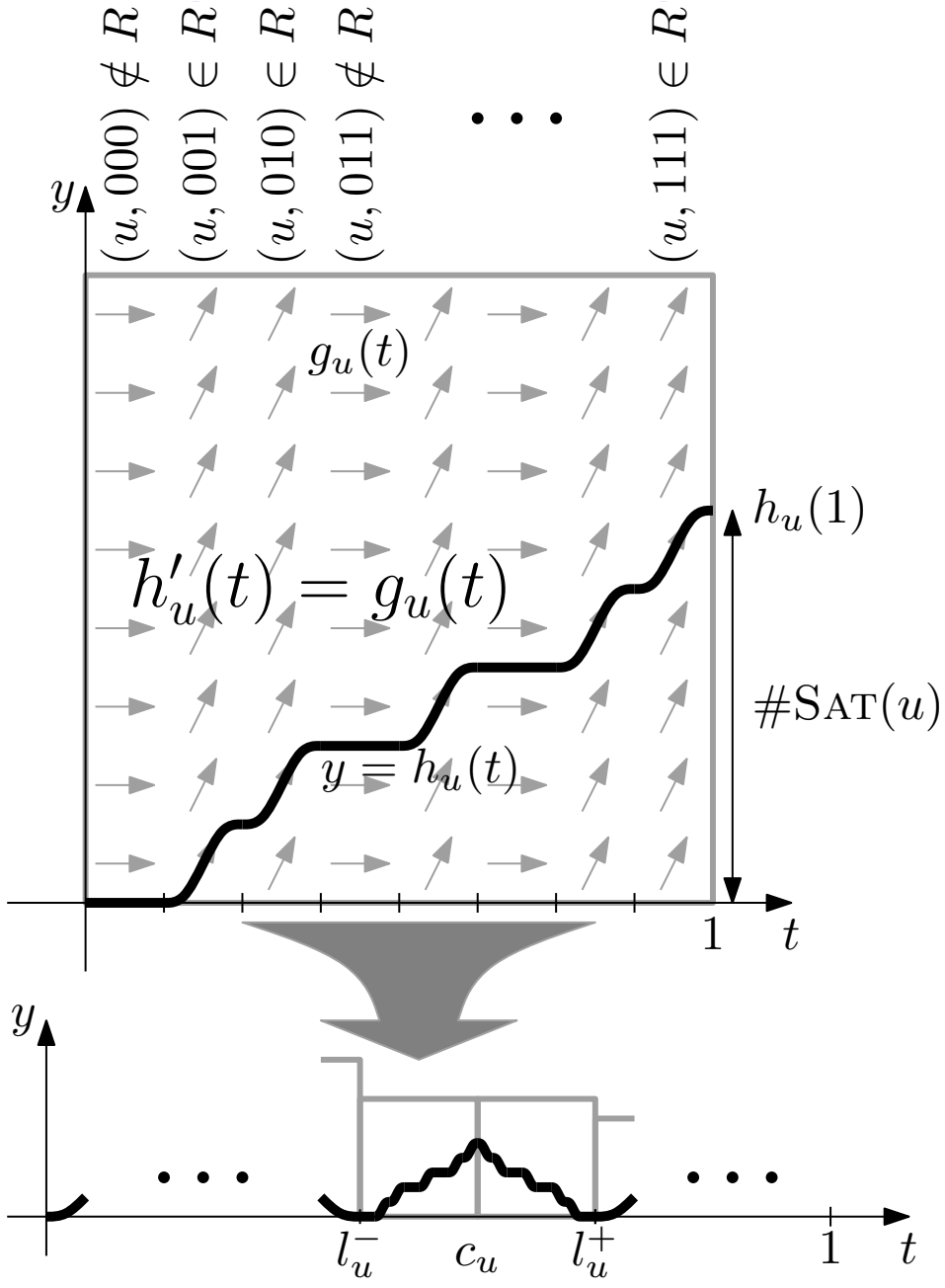


$\#\text{SAT}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

$\#\text{SAT}(u)$ is encoded in $h_u(1)$, or in $h(c_u)$.

Reducing #SAT to integration



$\#\textsc{sat}(u) =$ number of v with $(u, v) \in R$ (for some $R \in \mathbf{P}$).

For each $u \in \Sigma^*$, assign interval $[l_u^-, l_u^+]$. Put there a pair of reduced copies of g_u.

$\#\textsc{sat}(u)$ is encoded in $h_u(1)$, or in $h(c_u)$.

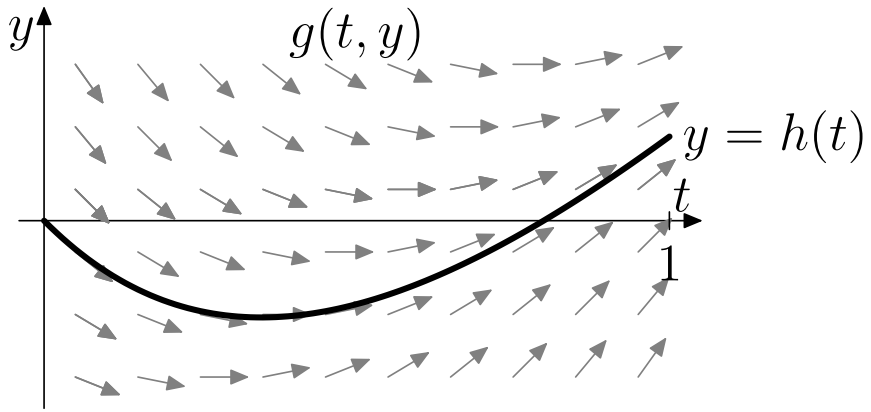
Outline

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

1. Some complexity classes
   P, **NP**, **#P**, **PSPACE**, ...
2. Complexity of real functions
   How we define polytime real functions
3. Warm-up: Integration
   Assuming g is polytime and ignores the second argument, how complex can h be?
4. ➤ Lipschitz continuous ODE
   Assuming g is polytime and Lipschitz continuous, how complex can h be?
5. Final remarks
   Uniform versions / Summary

The Lipschitz condition

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$



A sufficient condition to have a unique solution h is that g is Lipschitz continuous: there is some constant L such that

$$|g(t, y_0) - g(t, y_1)| \leq L|y_0 - y_1|.$$

Complexity of Lipschitz ODEs

Theorem [K 2010]

There are $g\colon [0,1]\times[-1,1]\to \mathbf{R}$ and $h\colon [0,1]\to[-1,1]$ such that

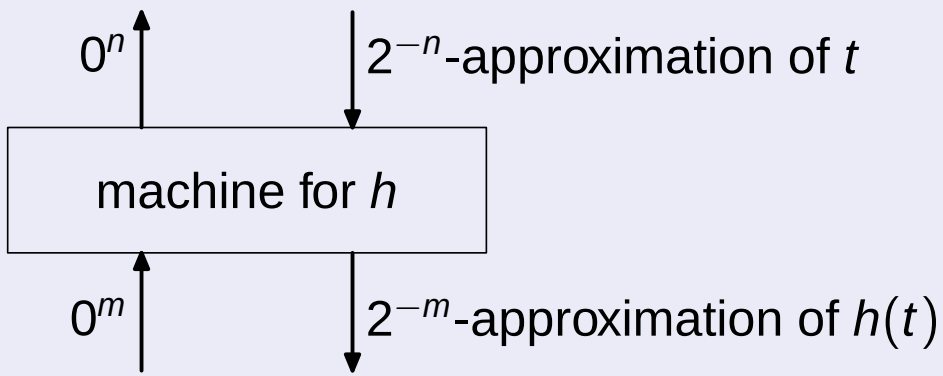
- g is polytime and Lipschitz continuous;
- $h(0)=0, \quad h'(t)=g\big(t,h(t)\big)$;
- h is **PSPACE**-hard in the sense that

Complexity of Lipschitz ODEs

Theorem [K 2010]

There are $g\colon [0,1]\times[-1,1]\to \mathbf{R}$ and $h\colon [0,1]\to[-1,1]$ such that

- g is polytime and Lipschitz continuous;
- $h(0)=0, \quad h'(t)=g\big(t,h(t)\big)$;
- h is **PSPACE**-hard in the sense that

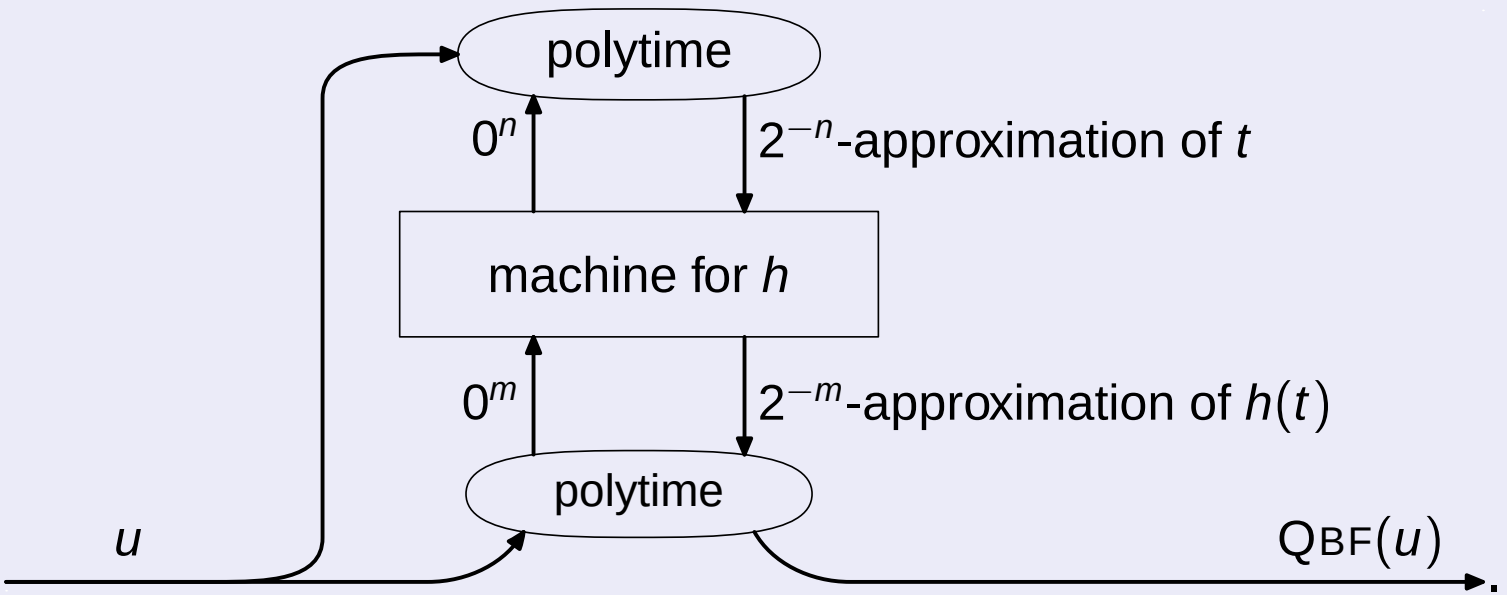


Complexity of Lipschitz ODEs

Theorem [K 2010]

There are $g\colon [0,1]\times[-1,1]\to \mathbf{R}$ and $h\colon [0,1]\to[-1,1]$ such that

- g is polytime and Lipschitz continuous;
- $h(0)=0, \quad h'(t)=g\big(t,h(t)\big)$;
- h is **PSPACE**-hard in the sense that

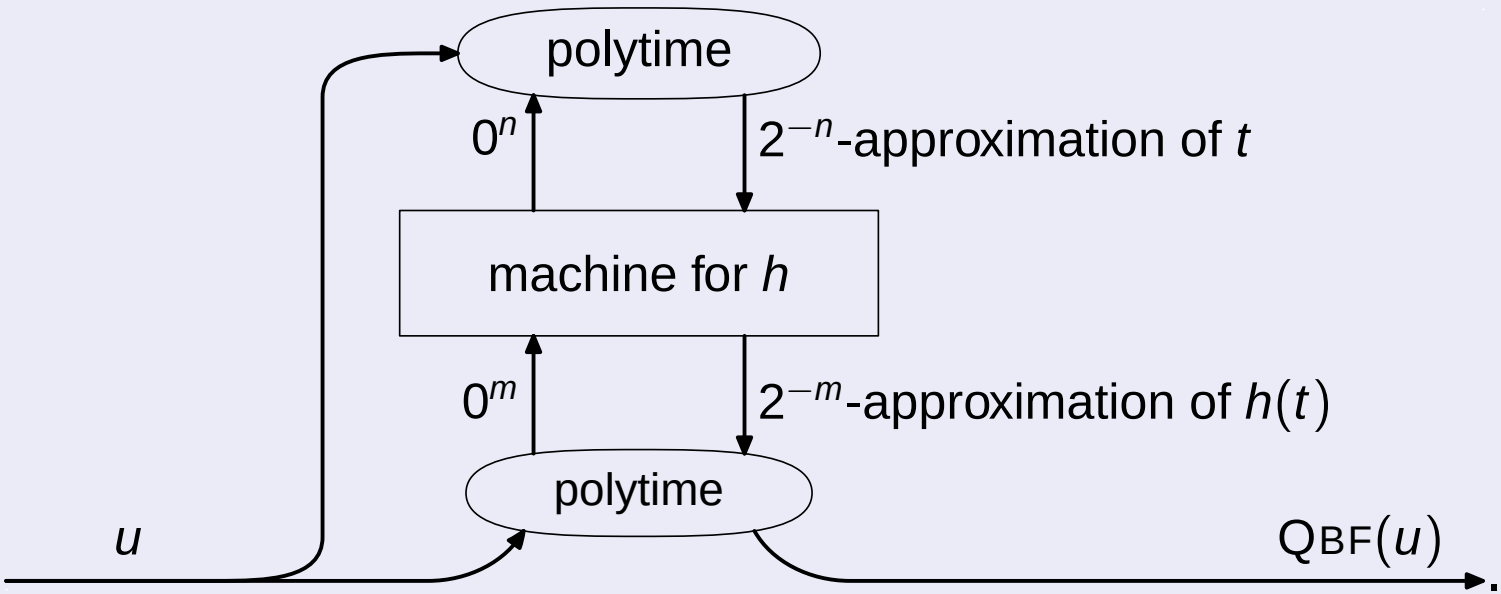


Complexity of Lipschitz ODEs

Theorem [K 2010]

There are $g\colon [0,1]\times[-1,1]\to \mathbf{R}$ and $h\colon [0,1]\to[-1,1]$ such that

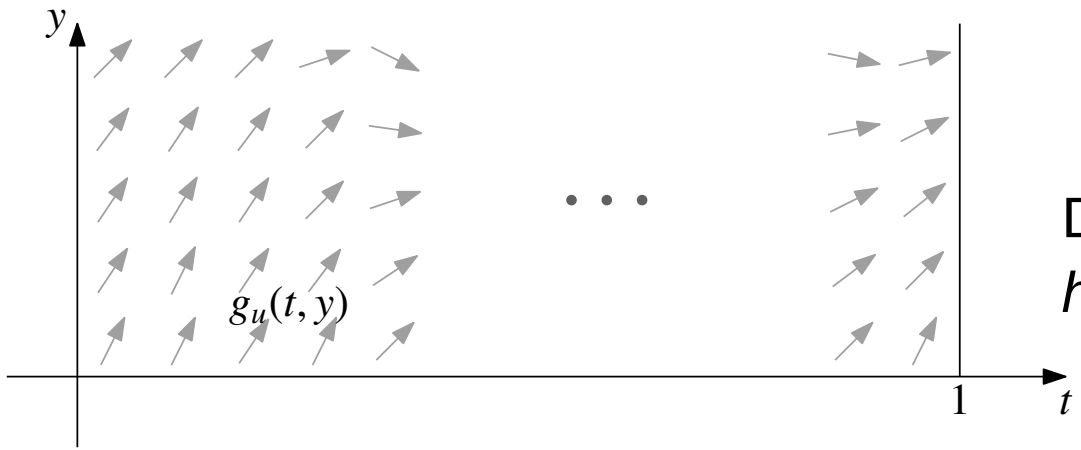
- g is polytime and Lipschitz continuous;
- $h(0)=0, \quad h'(t)=g\big(t,h(t)\big)$;
- h is **PSPACE**-hard in the sense that



Cf. Upper bound: h is polyspace by the Euler method [Ko 1983].

Proof (1/4): An attempt to reduce **PSPACE** to ODE

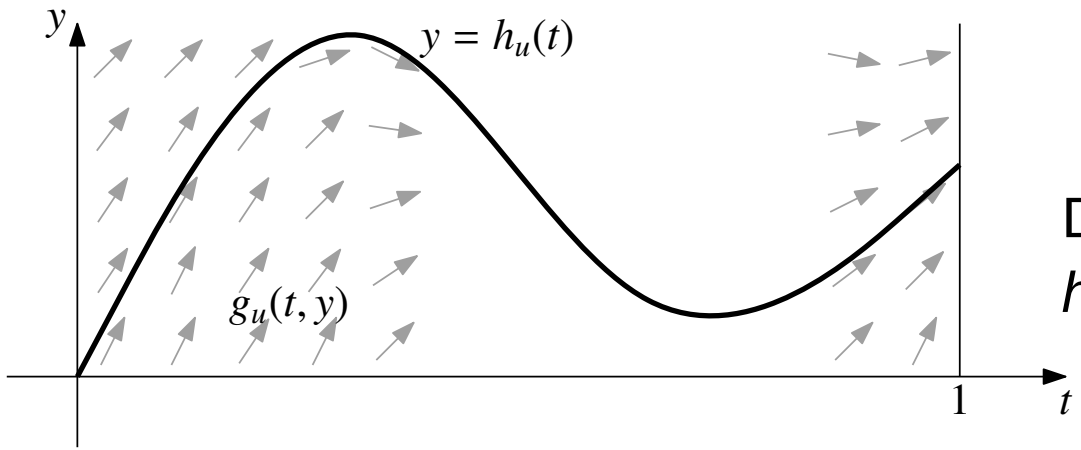
As before, we need blocks g_u such that $h_u(1)$ indicates if $u \in \text{QBF}$.



Differential equation:
$h'_u(t) = g_u\big(t, h(t)\big)$

Proof (1/4): An attempt to reduce **PSPACE** to ODE

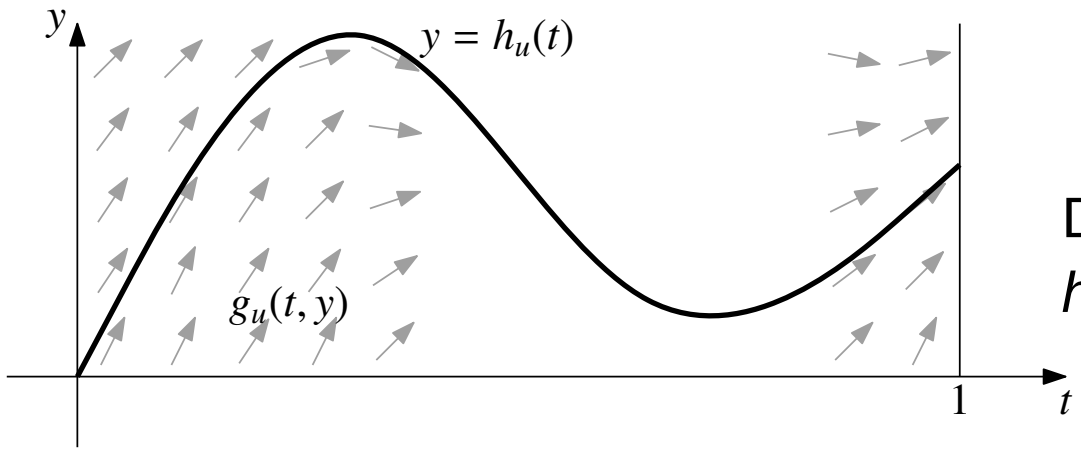
As before, we need blocks g_u such that $h_u(1)$ indicates if $u \in \text{QBF}$.



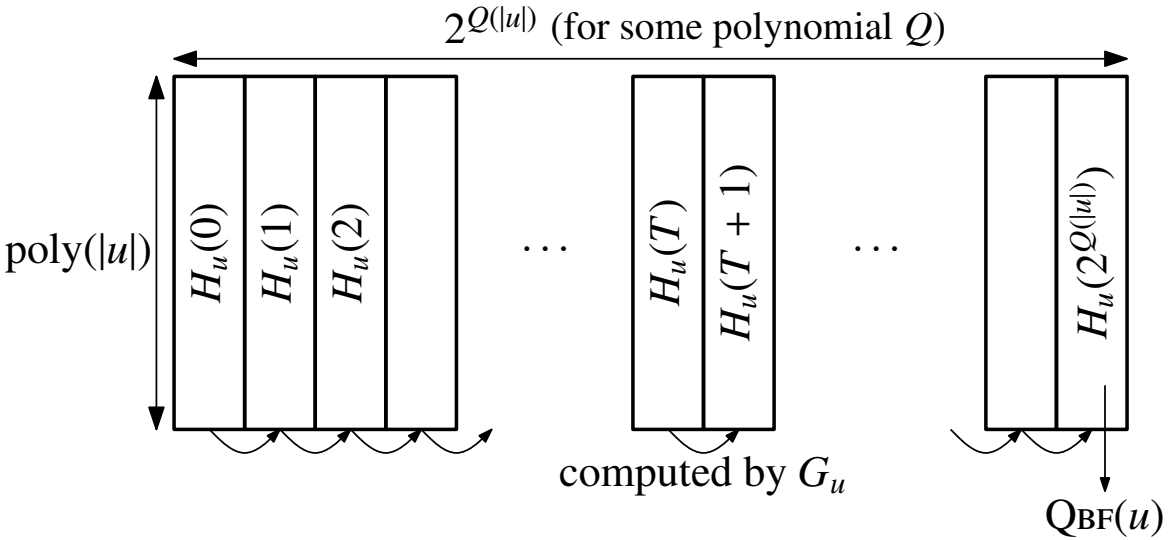
Differential equation:
$h'_u(t) = g_u\big(t, h(t)\big)$

Proof (1/4): An attempt to reduce **PSPACE** to ODE

As before, we need blocks g_u such that $h_u(1)$ indicates if $u \in \text{QBF}$.



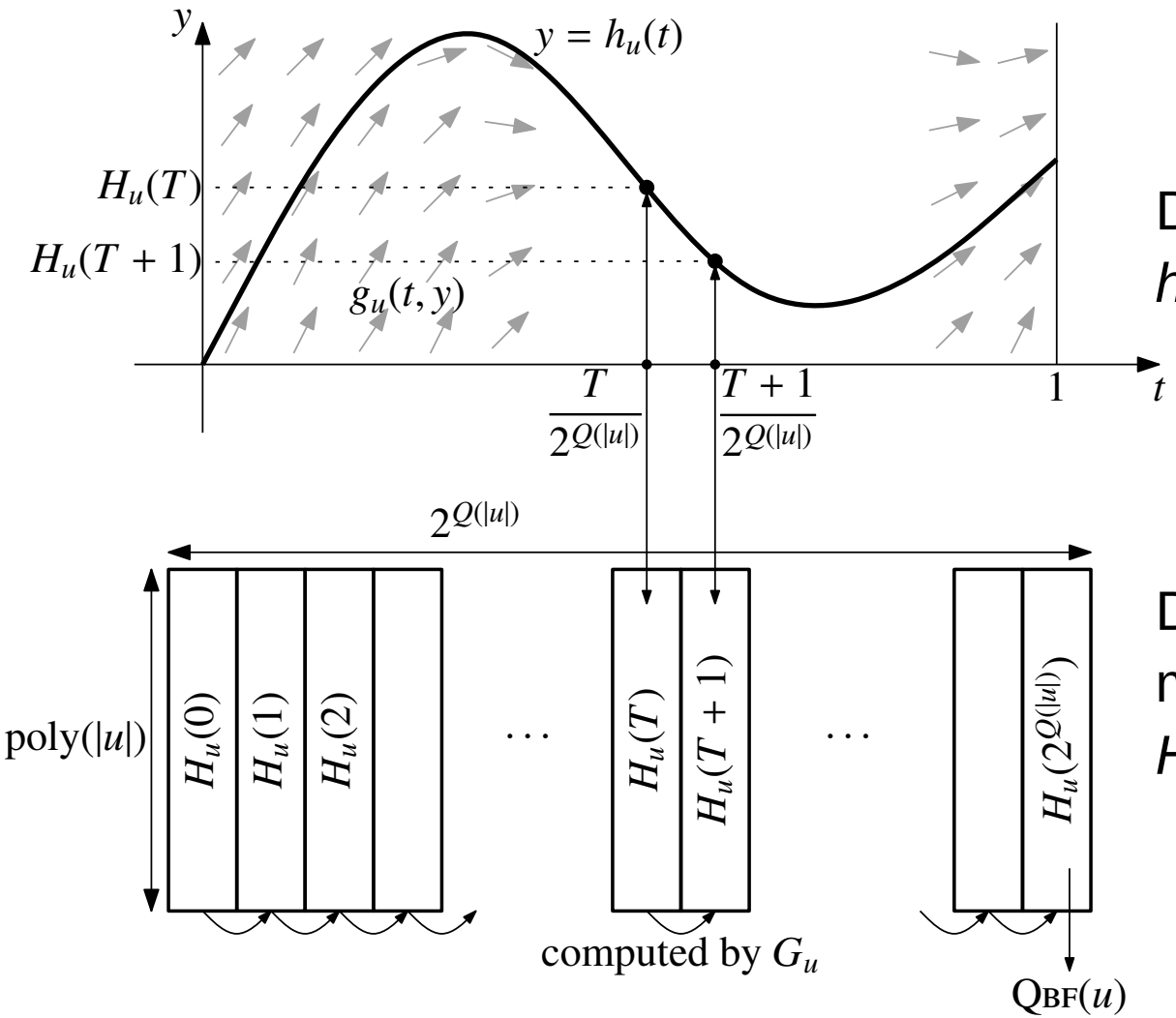
Differential equation:
$h'_u(t) = g_u\big(t, h(t)\big)$



Description of a **PSPACE** machine on input u:
$H_u(T+1) = G_u(T, H_u(T))$

Proof (1/4): An attempt to reduce **PSPACE** to ODE

As before, we need blocks g_u such that $h_u(1)$ indicates if $u \in \text{QBF}$.



Differential equation:

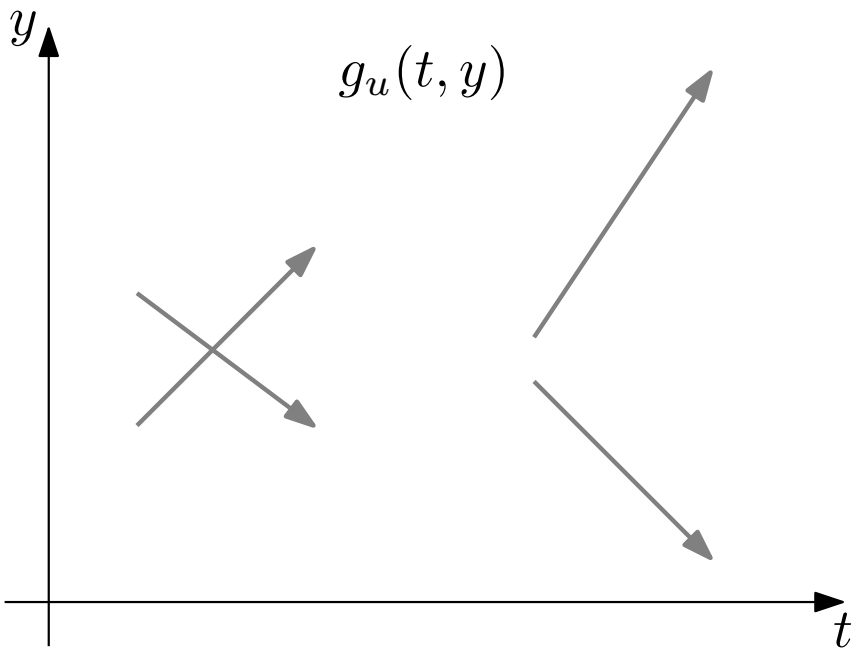
$$h'_u(t) = g_u\big(t, h(t)\big)$$

Description of a **PSPACE** machine on input u:

$$H_u(T+1) = G_u\big(T, H_u(T)\big)$$

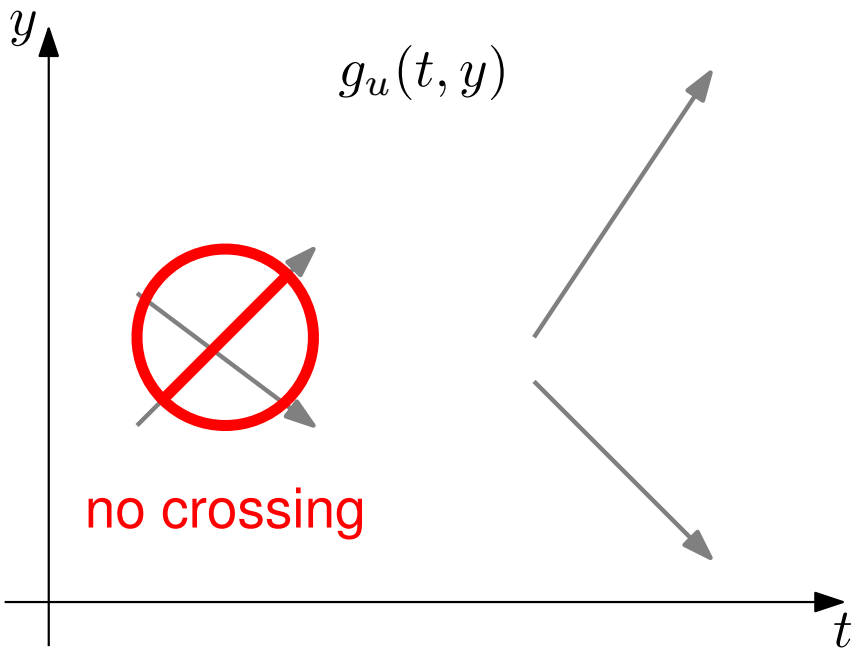
Proof (2/4): Why this attempt does not work

Not all G_u can be translated to g_u.



Proof (2/4): Why this attempt does not work

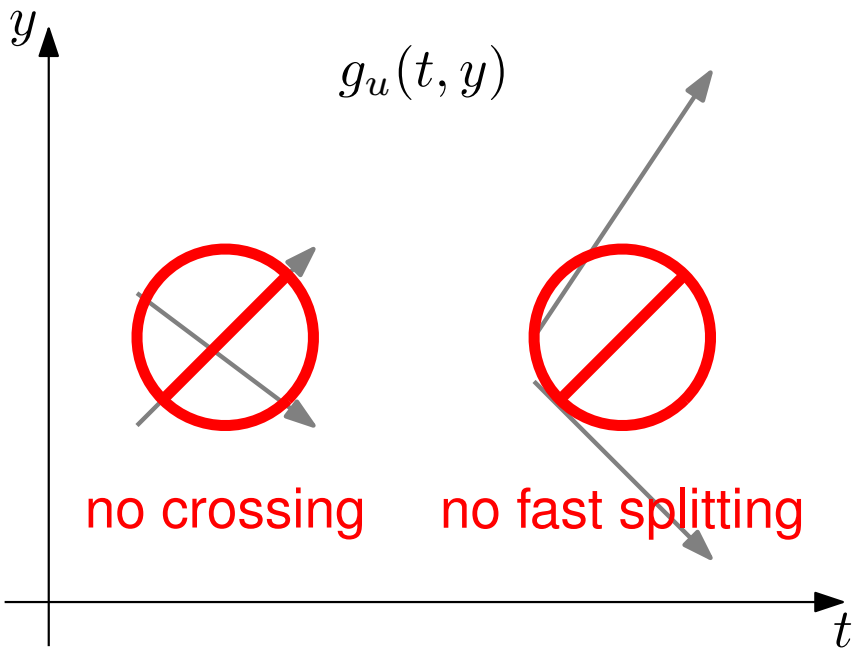
Not all G_u can be translated to g_u.



Flows can never cross.

Proof (2/4): Why this attempt does not work

Not all G_u can be translated to g_u.

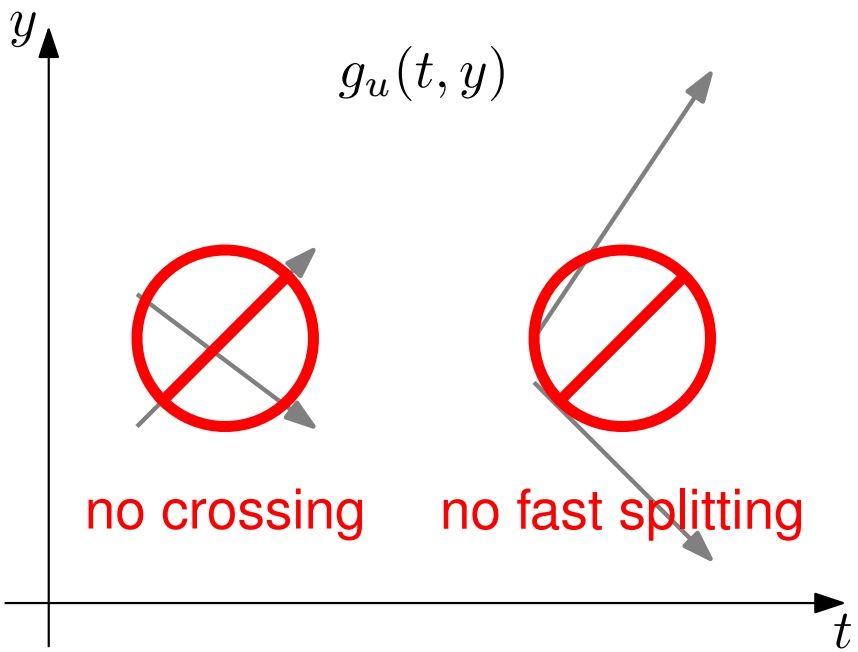


Flows can never cross.

In fact, the Lipschitz condition forbids the flows from widening or narrowing fast.

Proof (2/4): Why this attempt does not work

Not all G_u can be translated to g_u.

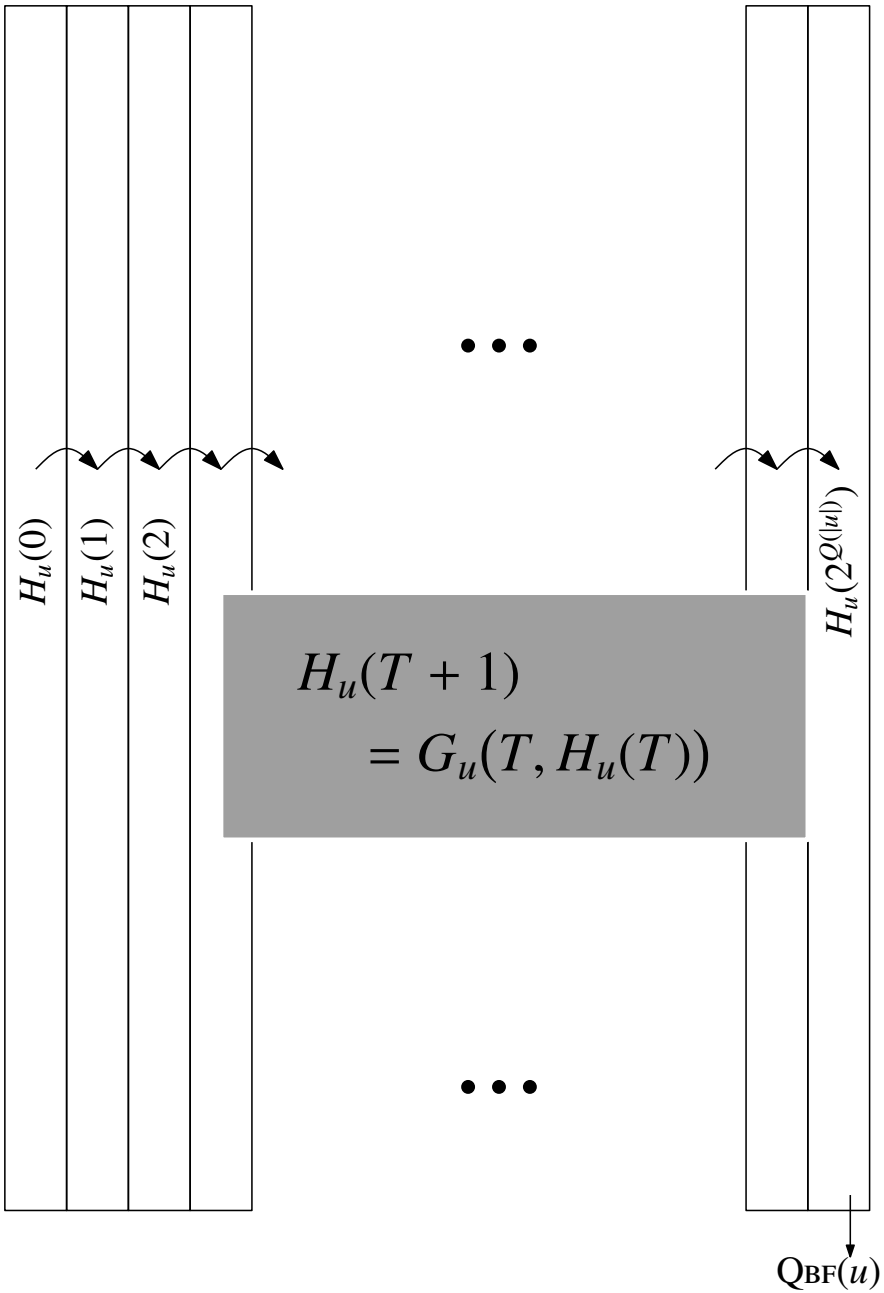


Flows can never cross.

In fact, the Lipschitz condition forbids the flows from widening or narrowing fast.

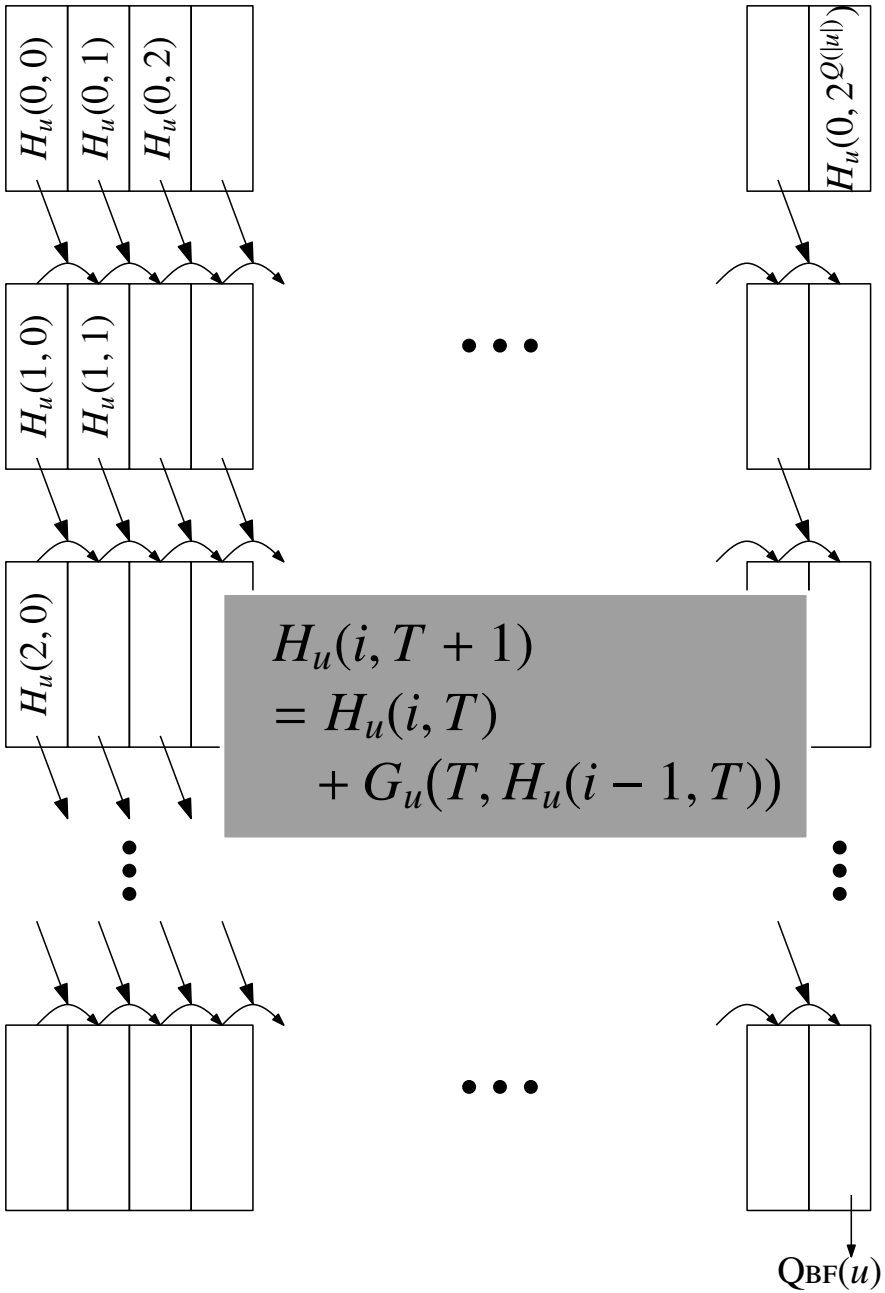
Thus, the 'feedback' (on h_u) described by the equation $h'_u(t) = g_u\big(t, h_u(t)\big)$ is very weak.

Proof (3/4): Layered **PSPACE** tableaux



Differential equations cannot simulate general **PSPACE** computation tableaux.

Proof (3/4): Layered **PSPACE** tableaux



Differential equations cannot simulate general **PSPACE** computation tableaux.

But it can simulate layered tableaux where each cell is split into parts and each part only affects parts below itself.

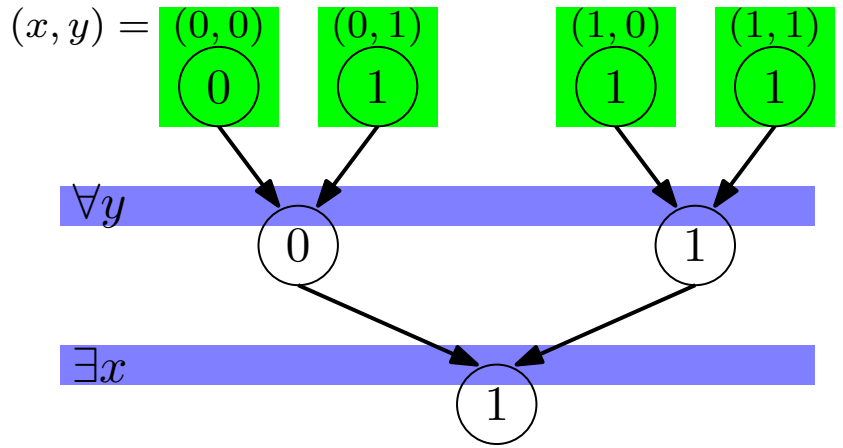
Proof (4/4): Layered tableaux is still hard

Despite the restriction, the layered tableau is **PSPACE**-complete, by a reduction from QBF.

Proof (4/4): Layered tableaux is still hard

Despite the restriction, the layered tableau is **PSPACE**-complete, by a reduction from QBF.

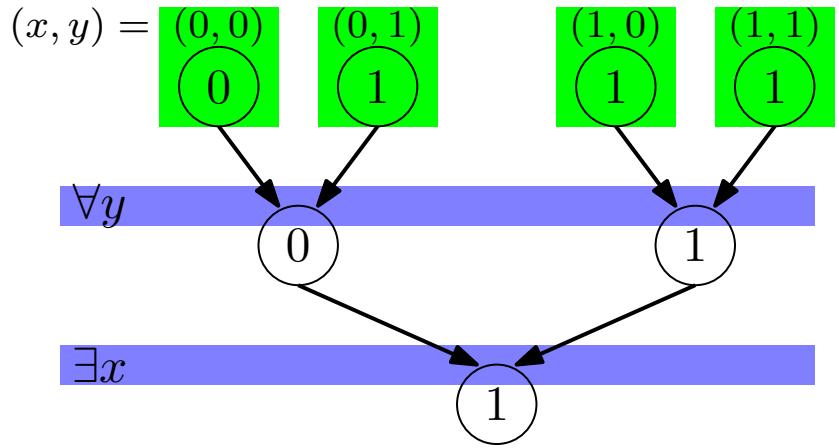
$\exists x\ \forall y\ (x \vee y)$:

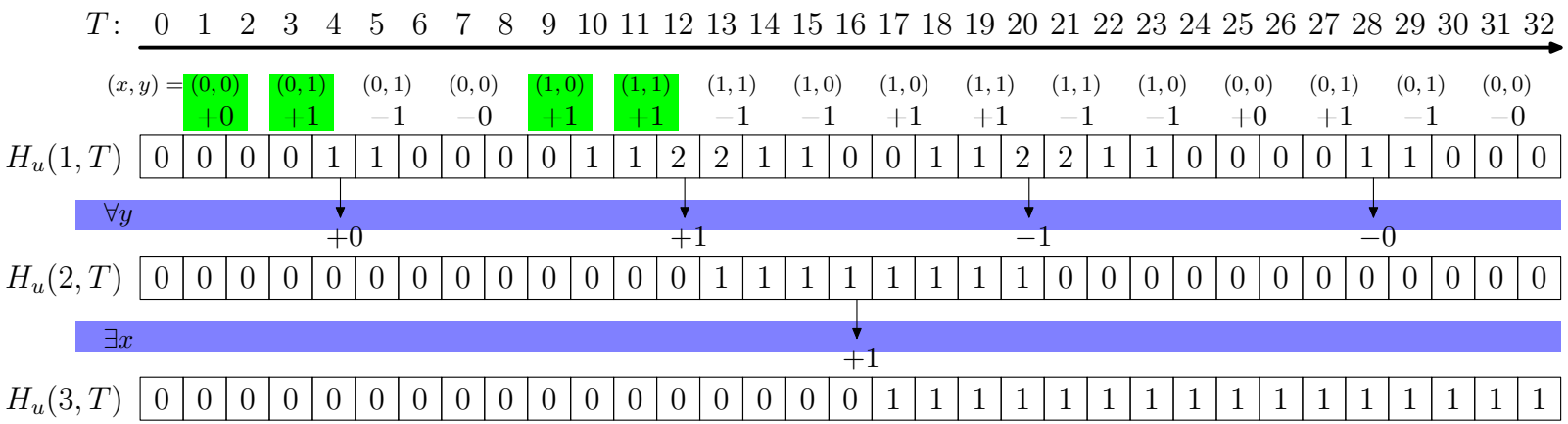


Proof (4/4): Layered tableaux is still hard

Despite the restriction, the layered tableau is **PSPACE**-complete, by a reduction from QBF.

$\exists x\ \forall y\ (x \vee y)$:





Complexity of smooth ODEs

Theorem [K, Ota, Rösnick, Ziegler 2012]

For any fixed $k \in \mathbf{N}$, there are g and h such that

- g is polytime and of class C^k;
- $h(0) = 0, \quad h'(t) = g\big(t, h(t)\big)$;
- h is **CH**-hard.

Similar argument, but with only constant number of layers.

Complexity of smooth ODEs

Theorem [K, Ota, Rösnick, Ziegler 2012]

For any fixed $k \in \mathbf{N}$, there are g and h such that

- g is polytime and of class C^k;
- $h(0) = 0, \quad h'(t) = g\big(t, h(t)\big)$;
- h is **CH**-hard.

Similar argument, but with only constant number of layers.

Open:

- Close the gap between this and the polyspace upper bound.
- C^∞ functions?
- Does smoothness generally help, for other operators?
  - Not for integration (still #**P**-hard for C^∞ functions)
  - nor for maximization (still **NP**-hard for C^∞ functions)

Outline

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

1. Some complexity classes
   P, **NP**, **#P**, **PSPACE**, ...
2. Complexity of real functions
   How we define polytime real functions
3. Warm-up: Integration
   Assuming g is polytime and ignores the second argument, how complex can h be?
4. Lipschitz continuous ODE
   Assuming g is polytime and Lipschitz continuous, how complex can h be?
5. ➤ Final remarks
   Uniform versions / Summary

Non-uniform and uniform statements

Our theorem stated:

Theorem [K 2010]

There is $g : [0,1] \times [-1,1] \to \mathbf{R}$ such that

- g is polytime and Lipschitz continuous;
- $Solve_{\mathrm{Lip}}(g)$ is **PSPACE**-hard.

where $Solve_{\mathrm{Lip}}$ is the operator mapping g to the solution h.

Non-uniform and uniform statements

Our theorem stated:

Theorem [K 2010]

There is $g : [0,1] \times [-1,1] \to \mathbf{R}$ such that

- g is polytime and Lipschitz continuous;
- $Solve_{\mathrm{Lip}}(g)$ is **PSPACE**-hard.

where $Solve_{\mathrm{Lip}}$ is the operator mapping g to the solution h.

But can we say something like:

Theorem (?)

$Solve_{\mathrm{Lip}}$ is **PSPACE**-hard.

Uniform version (1/4): Encoding

"What is the complexity of the operator that maps g to h?"

Uniform version (1/4): Encoding

"What is the complexity of the operator that maps g to h?"

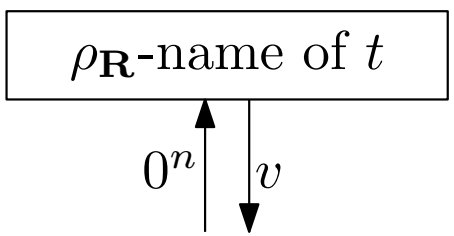
$$Solve_{\mathrm{Lip}}(g) := \text{the unique solution } h \text{ of the equation.}$$

Uniform version (1/4): Encoding

"What is the complexity of the operator that maps g to h?"

$$\mathit{Solve}_{\mathrm{Lip}}(g) := \text{the unique solution } h \text{ of the equation.}$$

Polytime real functions were defined by a polytime oracle machine and an encoding $\rho_{\mathbf{R}}$ of real numbers by string functions.
(Henceforth we will say $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime, etc.)



$$|v - t| \leq 2^{-n}$$

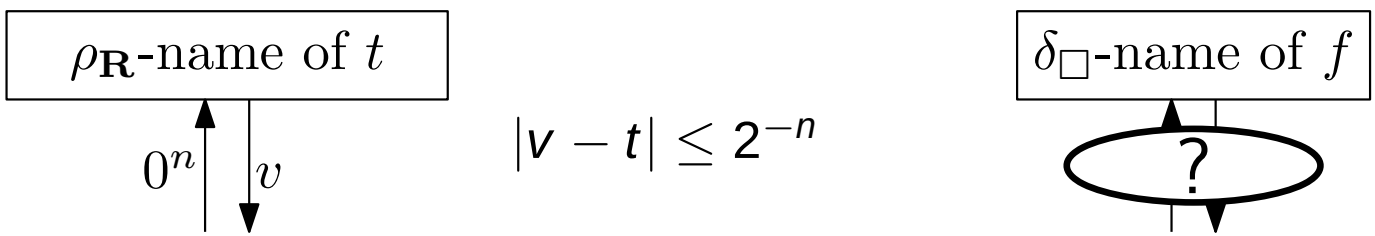
Uniform version (1/4): Encoding

"What is the complexity of the operator that maps g to h?"

$$Solve_{\mathrm{Lip}}(g) := \text{the unique solution } h \text{ of the equation.}$$

Polytime real functions were defined by a polytime oracle machine and an encoding $\rho_{\mathbf{R}}$ of real numbers by string functions.
(Henceforth we will say $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime, etc.)

To define polytime operators taking real functions to real functions, we need an encoding $\delta_{\square}$ of real functions by string functions.



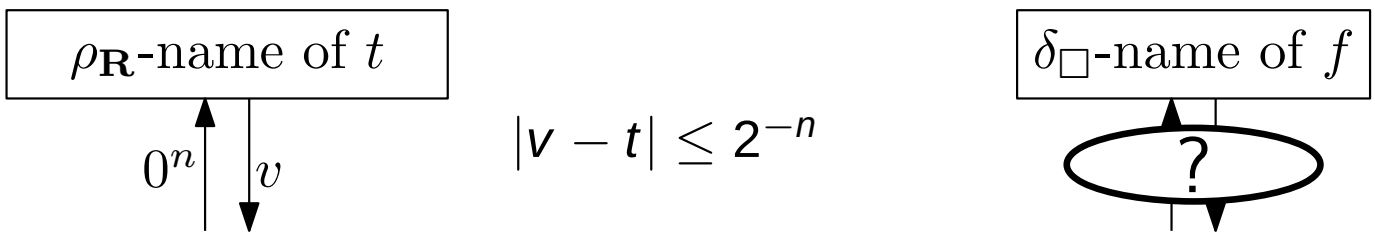
Uniform version (1/4): Encoding

"What is the complexity of the operator that maps g to h?"

$$Solve_{\mathrm{Lip}}(g) := \text{the unique solution } h \text{ of the equation.}$$

Polytime real functions were defined by a polytime oracle machine and an encoding $\rho_{\mathbf{R}}$ of real numbers by string functions.
(Henceforth we will say $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime, etc.)

To define polytime operators taking real functions to real functions, we need an encoding $\delta_{\square}$ of real functions by string functions.



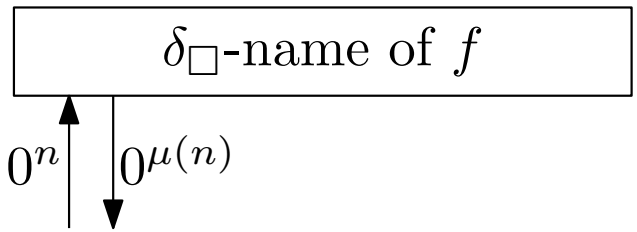
To make this work,
we need to use oracles whose answers are unbounded in length and extend the notion of polytime accordingly [K and Cook 2012].

Uniform version (2/4): Encoding of real functions

Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:

Uniform version (2/4): Encoding of real functions

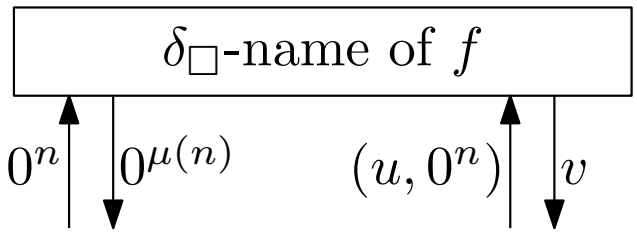
Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:



μ: modulus of continuity of f

Uniform version (2/4): Encoding of real functions

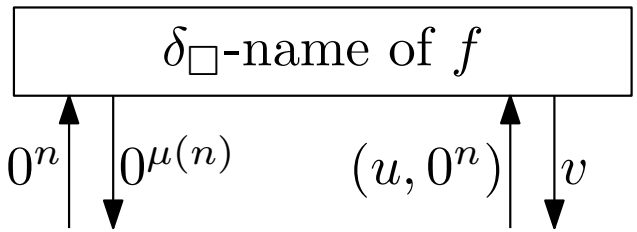
Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:



μ: modulus of continuity of f
$|v - f(u)| \leq 2^{-n}$

Uniform version (2/4): Encoding of real functions

Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:

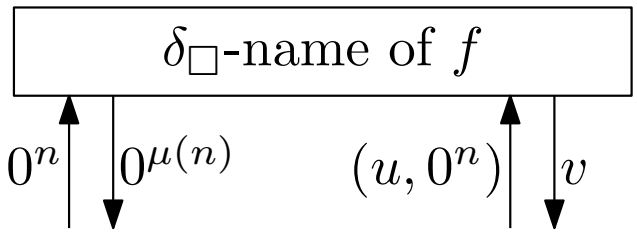


μ: modulus of continuity of f
$|v - f(u)| \leq 2^{-n}$

- Every continuous function has a name

Uniform version (2/4): Encoding of real functions

Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:

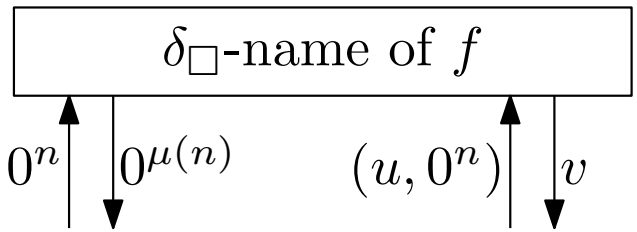


μ: modulus of continuity of f
$|v - f(u)| \leq 2^{-n}$

- Every continuous function has a name
- A function is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime iff it has a name in **FP**

Uniform version (2/4): Encoding of real functions

Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:

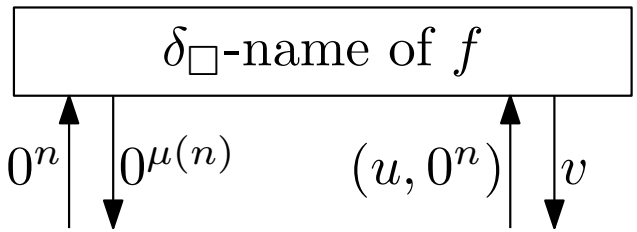


μ: modulus of continuity of f
$|v - f(u)| \leq 2^{-n}$

- Every continuous function has a name
- A function is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime iff it has a name in **FP**
- Function evaluation is $(\delta_\square, \rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime

Uniform version (2/4): Encoding of real functions

Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:

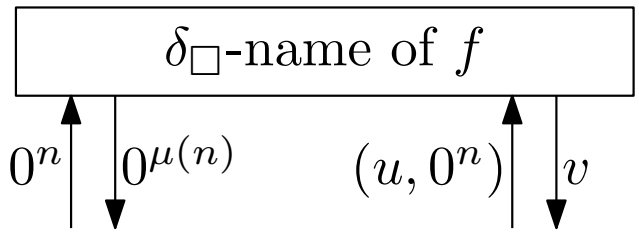


μ: modulus of continuity of f
$|v - f(u)| \leq 2^{-n}$

- Every continuous function has a name
- A function is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime iff it has a name in **FP**
- Function evaluation is $(\delta_\square, \rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime
- $\delta_\square$ is the poorest representation with this property

Uniform version (2/4): Encoding of real functions

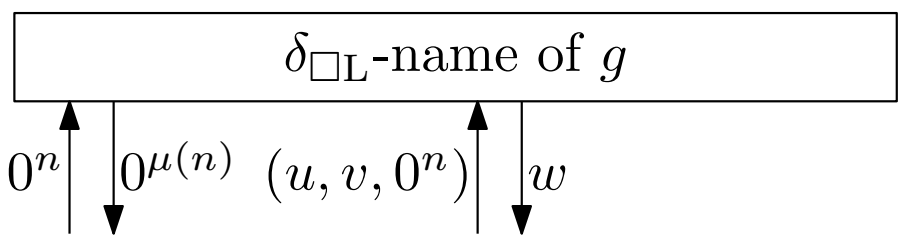
Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:



μ: modulus of continuity of f
$|v - f(u)| \leq 2^{-n}$

- Every continuous function has a name
- A function is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime iff it has a name in **FP**
- Function evaluation is $(\delta_\square, \rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime
- $\delta_\square$ is the poorest representation with this property

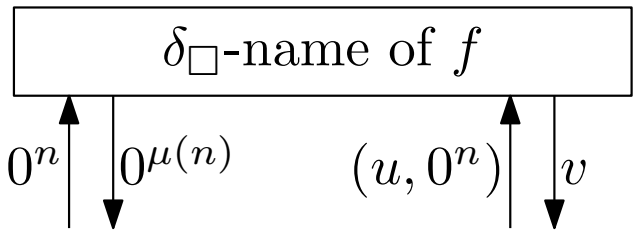
Encoding $\delta_{\square\mathrm{L}}$ of Lipschitz continuous functions $g : [0,1] \times \mathbf{R} \to \mathbf{R}$:



μ: modulus of continuity of g
$|w - g(u,v)| \leq 2^{-n}$

Uniform version (2/4): Encoding of real functions

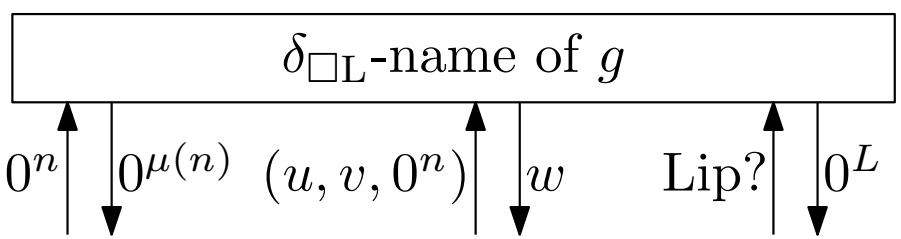
Encoding $\delta_\square$ of continuous functions $f : [0,1] \to \mathbf{R}$:



μ: modulus of continuity of f
$|v - f(u)| \leq 2^{-n}$

- Every continuous function has a name
- A function is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime iff it has a name in **FP**
- Function evaluation is $(\delta_\square, \rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime
- $\delta_\square$ is the poorest representation with this property

Encoding $\delta_{\square\mathrm{L}}$ of Lipschitz continuous functions $g : [0,1] \times \mathbf{R} \to \mathbf{R}$:



μ: modulus of continuity of g
$|w - g(u,v)| \leq 2^{-n}$
$L \in \mathbf{N}$: Lipschitz constant of g

Uniform version (3/4): Lipschitz ODE

By relativizing the proof, we can show:

Theorem (uniform)

- $\mathit{Solve}_{\mathrm{Lip}}$ is $(\delta_{\square \mathrm{L}}, \delta_{\square})$-polyspace.
- $\mathit{Solve}_{\mathrm{Lip}}$ is $(\delta_{\square \mathrm{L}}, \delta_{\square})$-polyspace-hard (wrt a suitable reduction).

Uniform version (3/4): Lipschitz ODE

By relativizing the proof, we can show:

Theorem (uniform)

- $Solve_{\mathrm{Lip}}$ is $(\delta_{\square \mathrm{L}}, \delta_{\square})$-polyspace.
- $Solve_{\mathrm{Lip}}$ is $(\delta_{\square \mathrm{L}}, \delta_{\square})$-polyspace-hard (wrt a suitable reduction).

Corollary (non-uniform)

- If g is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polyspace, so is $Solve_{\mathrm{Lip}}(g)$.
- There is a $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime function g such that $Solve_{\mathrm{Lip}}(g)$ is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polyspace-hard.

Uniform version (4/4): Analytic ODE

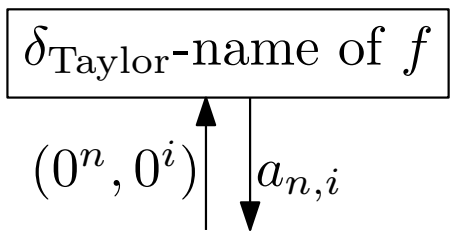
Corollary (non-uniform)

If g is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime, so is $Solve_{\text{Analytic}}(g)$.

Uniform version (4/4): Analytic ODE

Theorem (uniform)

$Solve_{\text{Analytic}}$ is $(\delta_{\text{Taylor}}, \delta_{\text{Taylor}})$-polytime.



$f(t) = \sum_{i \in \mathbf{N}} a_i x^i$ for t near 0
$|a_{n,i} - a_i| \leq 2^{-n}$

Corollary (non-uniform)

If g is $(\rho_{[0,1]}, \rho_{\mathbf{R}})$-polytime, so is $Solve_{\text{Analytic}}(g)$.

Summary: Complexity of ODE solutions

$$h(0) = 0, \qquad h'(t) = g\big(t, h(t)\big)$$

Assuming g is polytime, how complex can h be?

Assumptions	Upper bound	Lower bound
None	——	can be (non-unique and) all non-computable [Pour-El 1979]
h is the unique solution	computable [implicit in Osgood 1898]	can take arbitrarily long time (or space) [Miller 1970]
g is 'locally' Lipschitz [Ko 1992]	exponential space [Ko 1992]	can be **EXPSPACE**-hard [K 2010]
g is Lipschitz	polyspace [Ko 1983]	can be **PSPACE**-hard [K 2010] (CCC 2009)
g is of class C^1		can be **PSPACE**-hard [K, Ota, Rösnick, Ziegler 2012] (MFCS 2012)
g is of class C^k		can be **CH**-hard [ibid.]
g is analytic	polytime [Müller 1987]	——

These results can be made uniform, wrt different representations.