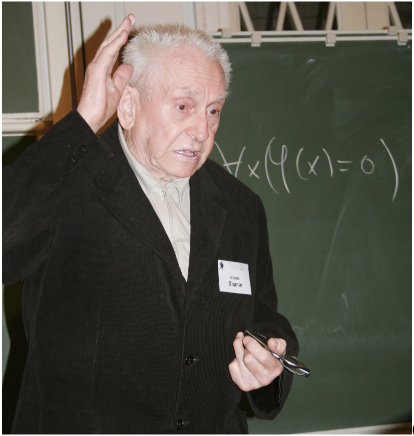
Universal statements and Kolmogorov complexity

Alexander Shen, LIRMM, Montpellier
Based on joint work with Laurent Bienvenu,
Andrei Romashchenko, Antoine Tavenaux, Stijn Vermeeren, Ruslan Ishkuvatov, Daniil Musatov

Николай Александрович Шанин: 100-летие



(CSR2006)

Universal statements in PA

- Π_1: $\forall x\, \Phi(x)$, $\Phi(x)$ is a (primitive) recursive statement
- nontermination of programs [without input]
- Π_1: Fermat's theorem (obvious), Riemann's conjecture (less obvious), no odd perfect numbers,...
- Π_2: infinitely many twin primes, Collatz,...
- Π_1: $\text{Consis}_{\mathrm{T}}$
- Hilbert's program: if PA is consistent, then every Π_1-statement is true

Universal statements in PA

- Π_1: $\forall x\, \Phi(x)$, $\Phi(x)$ is a (primitive) recursive statement
- nontermination of programs [without input]
- Π_1: Fermat's theorem (obvious), Riemann's conjecture (less obvious), no odd perfect numbers,...
- Π_2: infinitely many twin primes, Collatz,...
- Π_1: $\text{Consis}_{\mathrm{T}}$
- Hilbert's program: if PA is consistent, then every Π_1-statement is true

Universal statements in PA

- Π_1: $\forall x\, \Phi(x)$, $\Phi(x)$ is a (primitive) recursive statement
- nontermination of programs [without input]
- Π_1: Fermat's theorem (obvious), Riemann's conjecture (less obvious), no odd perfect numbers,...
- Π_2: infinitely many twin primes, Collatz,...
- Π_1: $\mathrm{Consis}_{\mathrm{T}}$
- Hilbert's program: if PA is consistent, then every Π_1-statement is true

Universal statements in PA

- Π_1: $\forall x\, \Phi(x)$, $\Phi(x)$ is a (primitive) recursive statement
- nontermination of programs [without input]
- Π_1: Fermat's theorem (obvious), Riemann's conjecture (less obvious), no odd perfect numbers,...
- Π_2: infinitely many twin primes, Collatz,...
- Π_1: $\mathrm{Consis}_\mathrm{T}$
- Hilbert's program: if PA is consistent, then every Π_1-statement is true

Universal statements in PA

- Π_1: $\forall x\, \Phi(x)$, $\Phi(x)$ is a (primitive) recursive statement
- nontermination of programs [without input]
- Π_1: Fermat's theorem (obvious), Riemann's conjecture (less obvious), no odd perfect numbers,...
- Π_2: infinitely many twin primes, Collatz,...
- Π_1: Consis_{T}
- Hilbert's program: if PA is consistent, then every Π_1-statement is true

Universal statements in PA

- Π_1: $\forall x\, \Phi(x)$, $\Phi(x)$ is a (primitive) recursive statement
- nontermination of programs [without input]
- Π_1: Fermat's theorem (obvious), Riemann's conjecture (less obvious), no odd perfect numbers,...
- Π_2: infinitely many twin primes, Collatz,...
- Π_1: $\mathrm{Consis}_{\mathrm{T}}$
- Hilbert's program: if PA is consistent, then every Π_1-statement is true

Universal statements in PA

- Π_1: $\forall x\, \Phi(x)$, $\Phi(x)$ is a (primitive) recursive statement
- nontermination of programs [without input]
- Π_1: Fermat's theorem (obvious), Riemann's conjecture (less obvious), no odd perfect numbers,...
- Π_2: infinitely many twin primes, Collatz,...
- Π_1: Consis_{T}
- Hilbert's program: if PA is consistent, then every Π_1-statement is true

Complexity of universal statements

- Cristian Calude, Elena Calude: length of the program whose non-termination is claimed
- optimal programming language that makes the length minimal
- let $A(n)$ be a Π_1-statement:
  $$\mathrm{C}_A(\varphi) = \min\{\log n \colon \mathrm{PA} \vdash \bigl(\varphi \Leftrightarrow A(n)\bigr)\}$$
- "Solomonoff – Kolmogorov optimality": there is A that makes C_A minimal up to a $O(1)$-constant:
  $$\exists A\, \forall B\, \exists c\, \forall \varphi\;\; \mathrm{C}_A(\varphi) \leqslant \mathrm{C}_B(\varphi) + c$$
- fix some optimal A (some optimal programming language):
  $$\mathrm{C}(\varphi) := \mathrm{C}_A(\varphi).$$
- similar definitions for Σ_n/Π_n

Complexity of universal statements

- Cristian Calude, Elena Calude: length of the program whose non-termination is claimed
- optimal programming language that makes the length minimal
- let $A(n)$ be a Π_1-statement:
  $$\mathrm{C}_A(\varphi) = \min\{\log n \colon \mathrm{PA} \vdash (\varphi \Leftrightarrow A(n))\}$$
- "Solomonoff – Kolmogorov optimality": there is A that makes C_A minimal up to a $O(1)$-constant:
  $$\exists A\, \forall B\, \exists c\, \forall \varphi\;\; \mathrm{C}_A(\varphi) \leqslant \mathrm{C}_B(\varphi) + c$$
- fix some optimal A (some optimal programming language):
  $$\mathrm{C}(\varphi) := \mathrm{C}_A(\varphi).$$
- similar definitions for Σ_n/Π_n

Complexity of universal statements

- Cristian Calude, Elena Calude: length of the program whose non-termination is claimed
- optimal programming language that makes the length minimal
- let $A(n)$ be a Π_1-statement:
  $$\mathrm{C}_A(\varphi) = \min\{\log n : \mathrm{PA} \vdash (\varphi \Leftrightarrow A(n))\}$$
- "Solomonoff – Kolmogorov optimality": there is A that makes C_A minimal up to a $O(1)$-constant:
  $$\exists A\, \forall B\, \exists c\, \forall \varphi\ \mathrm{C}_A(\varphi) \leqslant \mathrm{C}_B(\varphi) + c$$
- fix some optimal A (some optimal programming language):
  $$\mathrm{C}(\varphi) := \mathrm{C}_A(\varphi).$$
- similar definitions for Σ_n/Π_n

Complexity of universal statements

- Cristian Calude, Elena Calude: length of the program whose non-termination is claimed
- optimal programming language that makes the length minimal
- let $A(n)$ be a Π_1-statement:
$$\mathrm{C}_A(\varphi) = \min\{\log n \colon \mathrm{PA} \vdash \big(\varphi \Leftrightarrow A(n)\big)\}$$
- "Solomonoff – Kolmogorov optimality": there is A that makes C_A minimal up to a $O(1)$-constant:
$$\exists A\, \forall B\, \exists c\, \forall \varphi\;\; \mathrm{C}_A(\varphi) \leqslant \mathrm{C}_B(\varphi) + c$$
- fix some optimal A (some optimal programming language):
$$\mathrm{C}(\varphi) := \mathrm{C}_A(\varphi).$$
- similar definitions for Σ_n/Π_n

Complexity of universal statements

- Cristian Calude, Elena Calude: length of the program whose non-termination is claimed
- optimal programming language that makes the length minimal
- let $A(n)$ be a Π_1-statement:
  $$\mathrm{C}_A(\varphi) = \min\{\log n : \mathrm{PA} \vdash \big(\varphi \Leftrightarrow A(n)\big)\}$$
- "Solomonoff – Kolmogorov optimality": there is A that makes C_A minimal up to a $O(1)$-constant:
  $$\exists A\, \forall B\, \exists c\, \forall \varphi\ \mathrm{C}_A(\varphi) \leqslant \mathrm{C}_B(\varphi) + c$$
- fix some optimal A (some optimal programming language):
  $$\mathrm{C}(\varphi) := \mathrm{C}_A(\varphi).$$
- similar definitions for Σ_n/Π_n

Complexity of universal statements

- Cristian Calude, Elena Calude: length of the program whose non-termination is claimed
- optimal programming language that makes the length minimal
- let $A(n)$ be a Π_1-statement:
  $$\mathrm{C}_A(\varphi) = \min\{\log n \colon \mathrm{PA} \vdash (\varphi \Leftrightarrow A(n))\}$$
- "Solomonoff – Kolmogorov optimality": there is A that makes C_A minimal up to a $O(1)$-constant:
  $$\exists A \, \forall B \, \exists c \, \forall \varphi \ \mathrm{C}_A(\varphi) \leqslant \mathrm{C}_B(\varphi) + c$$
- fix some optimal A (some optimal programming language):
  $$\mathrm{C}(\varphi) := \mathrm{C}_A(\varphi).$$
- similar definitions for Σ_n/Π_n

Complexity of universal statements

- Cristian Calude, Elena Calude: length of the program whose non-termination is claimed
- optimal programming language that makes the length minimal
- let $A(n)$ be a Π_1-statement:
  $$\mathrm{C}_A(\varphi) = \min\{\log n \colon \mathrm{PA} \vdash \big(\varphi \Leftrightarrow A(n)\big)\}$$
- "Solomonoff – Kolmogorov optimality": there is A that makes C_A minimal up to a $O(1)$-constant:
  $$\exists A\,\forall B\,\exists c\,\forall\varphi\ \mathrm{C}_A(\varphi) \leqslant \mathrm{C}_B(\varphi) + c$$
- fix some optimal A (some optimal programming language):
  $$\mathrm{C}(\varphi) := \mathrm{C}_A(\varphi).$$
- similar definitions for Σ_n/Π_n

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Some remarks about the complexity

- all provable / refutable statements have $O(1)$-complexity
- there exists $\Theta(2^n)$ statements of complexity at most n:
- (upper bound) the number of programs
- (lower bound) construct A such that $A(n)$ are all independent from PA and each other
- let $a(\cdot)$ be a program of a unary function that is not provably different from any program (fixed-point argument)
- $A(n) :=$ «$a(n)$ never terminates»
- all $A(n)$ are independent (otherwise one can construct a program provably non-equivalent to a)
- there is a true universal statement of complexity at most n that implies (in PA) all true universal statements of complexity at most $n - O(1)$

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- "amount of information in x measured in bits"
- defined up to $O(1)$ additive term
- "algorithmic transformation does not create new information": $\forall A\, \exists c\, \forall x\, [\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- "amount of information in x measured in bits"
- defined up to $O(1)$ additive term
- "algorithmic transformation does not create new information": $\forall A\, \exists c\, \forall x\, [\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- "amount of information in x measured in bits"
- defined up to $O(1)$ additive term
- "algorithmic transformation does not create new information": $\forall A\, \exists c\, \forall x\, [\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- “amount of information in x measured in bits”
- defined up to $O(1)$ additive term
- “algorithmic transformation does not create new information”: $\forall A\,\exists c\,\forall x\,[\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- "amount of information in x measured in bits"
- defined up to $O(1)$ additive term
- "algorithmic transformation does not create new information": $\forall A\, \exists c\, \forall x\, [\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- "amount of information in x measured in bits"
- defined up to $O(1)$ additive term
- "algorithmic transformation does not create new information": $\forall A\, \exists c\, \forall x\, [\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- “amount of information in x measured in bits”
- defined up to $O(1)$ additive term
- “algorithmic transformation does not create new information”: $\forall A\, \exists c\, \forall x\, [\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- "amount of information in x measured in bits"
- defined up to $O(1)$ additive term
- "algorithmic transformation does not create new information": $\forall A\, \exists c\, \forall x\, [\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- "amount of information in x measured in bits"
- defined up to $O(1)$ additive term
- "algorithmic transformation does not create new information": $\forall A\,\exists c\,\forall x\,[\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- ...so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Kolmogorov complexity: a quick reminder

- $\mathrm{C}(x)$ is the minimal length of a program (without input) that produces x; depends on the programming language
- there is an optimal one that makes the complexity minimal up to $+O(1)$
- fix some optimal language and the corresponding $\mathrm{C}(x)$
- "amount of information in x measured in bits"
- defined up to $O(1)$ additive term
- "algorithmic transformation does not create new information": $\forall A\, \exists c\, \forall x\, [\mathrm{C}(A(x)) \leqslant \mathrm{C}(x) + c]$
- there are at most $1 + 2 + \ldots + 2^{n-1} < 2^n$ strings of complexity less than n
- …so for each n there is an incompressible x of length n: $\mathrm{C}(x) \geqslant n$ (in fact $\Theta(2^n)$ of them)

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Universal *complexity* statements

- Chatin's proof of Gödel's incompleteness theorem:
- «$\mathrm{C}(x) \geqslant n$» where x and n are constants (string/number);
- Chaitin: All provable universal complexity statements $\mathrm{C}(x) \geqslant n$ have $n \leqslant O(1)$
- otherwise trying all proofs we may generate strings of arbitrarily high complexity ($\mathrm{C}(x_m) \geqslant m$) effectively, but
$$m \leqslant \mathrm{C}(x_m) \leqslant O(\log m)$$
- $C(x) \geqslant m$ is a universal statement: program looking for a short description of x never terminates
- ...of complexity at most $|x| + O(\log m)$ and at least $m - O(1)$
- «x is incompressible»:= $\mathrm{C}(x) \geqslant |x|$
- ...complexity $|x| + O(1)$

Complexity and incompressibility statements

Theorem: all true universal statements of complexity at most m do not imply any statement $\mathrm{C}(x) \geqslant m'$ for $m' > m + O(1)$

Proof:

- the list of all true universal statements of complexity of most m has complexity $m + O(1)$. Why?
- indeed, the program of length at most m with maximal computation time determines this list and m (we add trailing zeros after separator, to get length m)
- knowing this list and m', we can enumerate all PA-consequences of the list waiting for the first provable statement of the form $\mathrm{C}(x) \geqslant m'$.
- this x has complexity at most $m + O(\log(m' - m))$, and it should be at least m', so $m' - m \leqslant O(1)$.

Complexity and incompressibility statements

Theorem: all true universal statements of complexity at most m do not imply any statement $\mathrm{C}(x) \geqslant m'$ for $m' > m + O(1)$

Proof:

- the list of all true universal statements of complexity of most m has complexity $m + O(1)$. Why?
- indeed, the program of length at most m with maximal computation time determines this list and m (we add trailing zeros after separator, to get length m)
- knowing this list and m', we can enumerate all PA-consequences of the list waiting for the first provable statement of the form $\mathrm{C}(x) \geqslant m'$.
- this x has complexity at most $m + O(\log(m' - m))$, and it should be at least m', so $m' - m \leqslant O(1)$.

Complexity and incompressibility statements

Theorem: all true universal statements of complexity at most m do not imply any statement $\mathrm{C}(x) \geqslant m'$ for $m' > m + O(1)$

Proof:

- the list of all true universal statements of complexity of most m has complexity $m + O(1)$. Why?
- indeed, the program of length at most m with maximal computation time determines this list and m (we add trailing zeros after separator, to get length m)
- knowing this list and m', we can enumerate all PA-consequences of the list waiting for the first provable statement of the form $\mathrm{C}(x) \geqslant m'$.
- this x has complexity at most $m + O(\log(m' - m))$, and it should be at least m', so $m' - m \leqslant O(1)$.

Complexity and incompressibility statements

Theorem: all true universal statements of complexity at most m do not imply any statement $\mathrm{C}(x) \geqslant m'$ for $m' > m + O(1)$

Proof:

- the list of all true universal statements of complexity of most m has complexity $m + O(1)$. Why?
- indeed, the program of length at most m with maximal computation time determines this list and m (we add trailing zeros after separator, to get length m)
- knowing this list and m', we can enumerate all PA-consequences of the list waiting for the first provable statement of the form $\mathrm{C}(x) \geqslant m'$.
- this x has complexity at most $m + O(\log(m' - m))$, and it should be at least m', so $m' - m \leqslant O(1)$.

Complexity and incompressibility statements

Theorem: all true universal statements of complexity at most m do not imply any statement $\mathrm{C}(x) \geqslant m'$ for $m' > m + O(1)$

Proof:

- the list of all true universal statements of complexity of most m has complexity $m + O(1)$. Why?
- indeed, the program of length at most m with maximal computation time determines this list and m (we add trailing zeros after separator, to get length m)
- knowing this list and m', we can enumerate all PA-consequences of the list waiting for the first provable statement of the form $\mathrm{C}(x) \geqslant m'$.
- this x has complexity at most $m + O(\log(m' - m))$, and it should be at least m', so $m' - m \leqslant O(1)$.

Complexity and incompressibility statements

Theorem: all true universal statements of complexity at most m do not imply any statement $\mathrm{C}(x) \geqslant m'$ for $m' > m + O(1)$

Proof:

- the list of all true universal statements of complexity of most m has complexity $m + O(1)$. Why?
- indeed, the program of length at most m with maximal computation time determines this list and m (we add trailing zeros after separator, to get length m)
- knowing this list and m', we can enumerate all PA-consequences of the list waiting for the first provable statement of the form $\mathrm{C}(x) \geqslant m'$.
- this x has complexity at most $m + O(\log(m' - m))$, and it should be at least m', so $m' - m \leqslant O(1)$.

Strong incompressibility statement

Let r_n be the first incompressible string of length n

Theorem: $\mathrm{C}(r_n) \geqslant n$ implies (in PA) all true universal statements of complexity at most $n - O(1)$.

- Complexity theory: let T be the time needed to find that all strings before r_n are compressible. Then all programs of length $n - O(1)$ stop in time T (or do not stop at all)
- ...can be formalized in PA
- with additional axiom $\mathrm{C}(r_n) \geqslant n$ we can confirm in PA that r_n is the first incompressible string
- ...and the value of T
- so we can prove in PA the non-termination of non-terminating programs of size $n - O(1)$ by checking that they do not terminate in T steps

Strong incompressibility statement

Let r_n be the first incompressible string of length n

Theorem: $\mathrm{C}(r_n) \geqslant n$ implies (in PA) all true universal statements of complexity at most $n - O(1)$.

- Complexity theory: let T be the time needed to find that all strings before r_n are compressible. Then all programs of length $n - O(1)$ stop in time T (or do not stop at all)
- ...can be formalized in PA
- with additional axiom $\mathrm{C}(r_n) \geqslant n$ we can confirm in PA that r_n is the first incompressible string
- ...and the value of T
- so we can prove in PA the non-termination of non-terminating programs of size $n - O(1)$ by checking that they do not terminate in T steps

Strong incompressibility statement

Let r_n be the first incompressible string of length n

Theorem: $\mathrm{C}(r_n) \geqslant n$ implies (in PA) all true universal statements of complexity at most $n - O(1)$.

- Complexity theory: let T be the time needed to find that all strings before r_n are compressible. Then all programs of length $n - O(1)$ stop in time T (or do not stop at all)
- ...can be formalized in PA
- with additional axiom $\mathrm{C}(r_n) \geqslant n$ we can confirm in PA that r_n is the first incompressible string
- ...and the value of T
- so we can prove in PA the non-termination of non-terminating programs of size $n - O(1)$ by checking that they do not terminate in T steps

Strong incompressibility statement

Let r_n be the first incompressible string of length n
Theorem: $\mathrm{C}(r_n) \geqslant n$ implies (in PA) all true universal statements of complexity at most $n - O(1)$.

- Complexity theory: let T be the time needed to find that all strings before r_n are compressible. Then all programs of length $n - O(1)$ stop in time T (or do not stop at all)
- ...can be formalized in PA
- with additional axiom $\mathrm{C}(r_n) \geqslant n$ we can confirm in PA that r_n is the first incompressible string
- ...and the value of T
- so we can prove in PA the non-termination of non-terminating programs of size $n - O(1)$ by checking that they do not terminate in T steps

Strong incompressibility statement

Let r_n be the first incompressible string of length n
Theorem: $\mathrm{C}(r_n) \geqslant n$ implies (in PA) all true universal statements of complexity at most $n - O(1)$.

- Complexity theory: let T be the time needed to find that all strings before r_n are compressible. Then all programs of length $n - O(1)$ stop in time T (or do not stop at all)
- ...can be formalized in PA
- with additional axiom $\mathrm{C}(r_n) \geqslant n$ we can confirm in PA that r_n is the first incompressible string
- ...and the value of T
- so we can prove in PA the non-termination of non-terminating programs of size $n - O(1)$ by checking that they do not terminate in T steps

Strong incompressibility statement

Let r_n be the first incompressible string of length n

Theorem: $\mathrm{C}(r_n) \geqslant n$ implies (in PA) all true universal statements of complexity at most $n - O(1)$.

- Complexity theory: let T be the time needed to find that all strings before r_n are compressible. Then all programs of length $n - O(1)$ stop in time T (or do not stop at all)
- ...can be formalized in PA
- with additional axiom $\mathrm{C}(r_n) \geqslant n$ we can confirm in PA that r_n is the first incompressible string
- ...and the value of T
- so we can prove in PA the non-termination of non-terminating programs of size $n - O(1)$ by checking that they do not terminate in T steps

Busy beaver numbers: a digression

- $BB(n)$: the longest computation time of program of size at most n
- $B(n)$: the maximal integer of complexity at most n
- $B(n - O(1)) \leqslant BB(n) \leqslant B(n + O(1))$
- $BB(n) \leqslant B(n + O(1))$ since $BB(n)$ has complexity at most $n + O(1)$, being determined by the program of size at most n (with maximal computational time)
- $B(n - O(1)) \leqslant BB(n)$: all numbers $t > BB(n)$ have complexity greater than $n - O(1)$, since they determine a string of complexity greater than n (try all programs for time t on all inputs and take a string different from all outputs)

Busy beaver numbers: a digression

- $BB(n)$: the longest computation time of program of size at most n
- $B(n)$: the maximal integer of complexity at most n
- $B(n - O(1)) \leqslant BB(n) \leqslant B(n + O(1))$
- $BB(n) \leqslant B(n + O(1))$ since $BB(n)$ has complexity at most $n + O(1)$, being determined by the program of size at most n (with maximal computational time)
- $B(n - O(1)) \leqslant BB(n)$: all numbers $t > BB(n)$ have complexity greater than $n - O(1)$, since they determine a string of complexity greater than n (try all programs for time t on all inputs and take a string different from all outputs)

Busy beaver numbers: a digression

- $BB(n)$: the longest computation time of program of size at most n
- $B(n)$: the maximal integer of complexity at most n
- $B(n - O(1)) \leqslant BB(n) \leqslant B(n + O(1))$
- $BB(n) \leqslant B(n + O(1))$ since $BB(n)$ has complexity at most $n + O(1)$, being determined by the program of size at most n (with maximal computational time)
- $B(n - O(1)) \leqslant BB(n)$: all numbers $t > BB(n)$ have complexity greater than $n - O(1)$, since they determine a string of complexity greater than n (try all programs for time t on all inputs and take a string different from all outputs)

Busy beaver numbers: a digression

- $BB(n)$: the longest computation time of program of size at most n
- $B(n)$: the maximal integer of complexity at most n
- $B(n - O(1)) \leqslant BB(n) \leqslant B(n + O(1))$
- $BB(n) \leqslant B(n + O(1))$ since $BB(n)$ has complexity at most $n + O(1)$, being determined by the program of size at most n (with maximal computational time)
- $B(n - O(1)) \leqslant BB(n)$: all numbers $t > BB(n)$ have complexity greater than $n - O(1)$, since they determine a string of complexity greater than n (try all programs for time t on all inputs and take a string different from all outputs)

Busy beaver numbers: a digression

- $BB(n)$: the longest computation time of program of size at most n
- $B(n)$: the maximal integer of complexity at most n
- $B(n - O(1)) \leqslant BB(n) \leqslant B(n + O(1))$
- $BB(n) \leqslant B(n + O(1))$ since $BB(n)$ has complexity at most $n + O(1)$, being determined by the program of size at most n (with maximal computational time)
- $B(n - O(1)) \leqslant BB(n)$: all numbers $t > BB(n)$ have complexity greater than $n - O(1)$, since they determine a string of complexity greater than n (try all programs for time t on all inputs and take a string different from all outputs)

Busy beaver numbers: a digression

- $BB(n)$: the longest computation time of program of size at most n
- $B(n)$: the maximal integer of complexity at most n
- $B(n - O(1)) \leqslant BB(n) \leqslant B(n + O(1))$
- $BB(n) \leqslant B(n + O(1))$ since $BB(n)$ has complexity at most $n + O(1)$, being determined by the program of size at most n (with maximal computational time)
- $B(n - O(1)) \leqslant BB(n)$: all numbers $t > BB(n)$ have complexity greater than $n - O(1)$, since they determine a string of complexity greater than n (try all programs for time t on all inputs and take a string different from all outputs)

Finishing the proof of the complexity statement

- r_n is the first incompressible bit strings in some order
- T is the time needed to find out that all previous strings are compressible
- why $T \geqslant B(n - O(1))$?
- ...or $T \geqslant BB(n - O(1))$?
- i.e., every $t > T$ has complexity greater than $n - O(1)$
- and this is because it can be used to find r_n
- technicality: we need also n, but it can be reconstructed: if t has complexity $n - d$, the program of length $n - d$ and $O(\log d)$ bits to encode d are enough to reconstruct n and r_n, so $n - d + O(\log d) + O(1) \geqslant \mathrm{C}(r_n) \geqslant n$ and $d = O(1)$.

Finishing the proof of the complexity statement

- r_n is the first incompressible bit strings in some order
- T is the time needed to find out that all previous strings are compressible
- why $T \geqslant B(n - O(1))$?
- ...or $T \geqslant BB(n - O(1))$?
- i.e., every $t > T$ has complexity greater than $n - O(1)$
- and this is because it can be used to find r_n
- technicality: we need also n, but it can be reconstructed: if t has complexity $n - d$, the program of length $n - d$ and $O(\log d)$ bits to encode d are enough to reconstruct n and r_n, so $n - d + O(\log d) + O(1) \geqslant \mathrm{C}(r_n) \geqslant n$ and $d = O(1)$.

Finishing the proof of the complexity statement

- r_n is the first incompressible bit strings in some order
- T is the time needed to find out that all previous strings are compressible
- why $T \geqslant B(n - O(1))$?
- ...or $T \geqslant BB(n - O(1))$?
- i.e., every $t > T$ has complexity greater than $n - O(1)$
- and this is because it can be used to find r_n
- technicality: we need also n, but it can be reconstructed: if t has complexity $n - d$, the program of length $n - d$ and $O(\log d)$ bits to encode d are enough to reconstruct n and r_n, so $n - d + O(\log d) + O(1) \geqslant \mathrm{C}(r_n) \geqslant n$ and $d = O(1)$.

Finishing the proof of the complexity statement

- r_n is the first incompressible bit strings in some order
- T is the time needed to find out that all previous strings are compressible
- why $T \geqslant B(n - O(1))$?
- ...or $T \geqslant BB(n - O(1))$?
- i.e., every $t > T$ has complexity greater than $n - O(1)$
- and this is because it can be used to find r_n
- technicality: we need also n, but it can be reconstructed: if t has complexity $n - d$, the program of length $n - d$ and $O(\log d)$ bits to encode d are enough to reconstruct n and r_n, so $n - d + O(\log d) + O(1) \geqslant \mathrm{C}(r_n) \geqslant n$ and $d = O(1)$.

Finishing the proof of the complexity statement

- r_n is the first incompressible bit strings in some order
- T is the time needed to find out that all previous strings are compressible
- why $T \geqslant B(n - O(1))$?
- ...or $T \geqslant BB(n - O(1))$?
- i.e., every $t > T$ has complexity greater than $n - O(1)$
- and this is because it can be used to find r_n
- technicality: we need also n, but it can be reconstructed: if t has complexity $n - d$, the program of length $n - d$ and $O(\log d)$ bits to encode d are enough to reconstruct n and r_n, so $n - d + O(\log d) + O(1) \geqslant \mathrm{C}(r_n) \geqslant n$ and $d = O(1)$.

Finishing the proof of the complexity statement

- r_n is the first incompressible bit strings in some order
- T is the time needed to find out that all previous strings are compressible
- why $T \geqslant B(n - O(1))$?
- ...or $T \geqslant BB(n - O(1))$?
- i.e., every $t > T$ has complexity greater than $n - O(1)$
- and this is because it can be used to find r_n
- technicality: we need also n, but it can be reconstructed: if t has complexity $n - d$, the program of length $n - d$ and $O(\log d)$ bits to encode d are enough to reconstruct n and r_n, so $n - d + O(\log d) + O(1) \geqslant \mathrm{C}(r_n) \geqslant n$ and $d = O(1)$.

Finishing the proof of the complexity statement

- r_n is the first incompressible bit strings in some order
- T is the time needed to find out that all previous strings are compressible
- why $T \geqslant B(n - O(1))$?
- ...or $T \geqslant BB(n - O(1))$?
- i.e., every $t > T$ has complexity greater than $n - O(1)$
- and this is because it can be used to find r_n
- technicality: we need also n, but it can be reconstructed: if t has complexity $n - d$, the program of length $n - d$ and $O(\log d)$ bits to encode d are enough to reconstruct n and r_n, so $n - d + O(\log d) + O(1) \geqslant \mathrm{C}(r_n) \geqslant n$ and $d = O(1)$.

Finishing the proof of the complexity statement

- r_n is the first incompressible bit strings in some order
- T is the time needed to find out that all previous strings are compressible
- why $T \geqslant B(n - O(1))$?
- ...or $T \geqslant BB(n - O(1))$?
- i.e., every $t > T$ has complexity greater than $n - O(1)$
- and this is because it can be used to find r_n
- technicality: we need also n, but it can be reconstructed: if t has complexity $n - d$, the program of length $n - d$ and $O(\log d)$ bits to encode d are enough to reconstruct n and r_n, so $n - d + O(\log d) + O(1) \geqslant \mathrm{C}(r_n) \geqslant n$ and $d = O(1)$.

Corollary and question

Corollary: Adding all true universal *complexity* statements as axioms, we can prove all true universal statements.

But is it true that every universal statement is provably equivalent to some universal *complexity* statement?

Theorem: Not every universal statement is provably equivalent to some universal complexity statement.

(If it were true, it would imply the corollary above)

Proof: easily follows from some results of An. Muchnik and S. Positselsky about non-m-completeness of the overgraph of the complexity function. (Again we have a Kolmogorov complexity result that translates into proof theory)

Corollary and question

Corollary: Adding all true universal *complexity* statements as axioms, we can prove all true universal statements.

But is it true that every universal statement is provably equivalent to some universal *complexity* statement?

Theorem: Not every universal statement is provably equivalent to some universal complexity statement.

(If it were true, it would imply the corollary above)

Proof: easily follows from some results of An. Muchnik and S. Positselsky about non-m-completeness of the overgraph of the complexity function. (Again we have a Kolmogorov complexity result that translates into proof theory)

Corollary and question

Corollary: Adding all true universal *complexity* statements as axioms, we can prove all true universal statements.

But is it true that every universal statement is provably equivalent to some universal *complexity* statement?

Theorem: Not every universal statement is provably equivalent to some universal complexity statement.

(If it were true, it would imply the corollary above)

Proof: easily follows from some results of An. Muchnik and S. Positselsky about non-m-completeness of the overgraph of the complexity function. (Again we have a Kolmogorov complexity result that translates into proof theory)

Corollary and question

Corollary: Adding all true universal *complexity* statements as axioms, we can prove all true universal statements.

But is it true that every universal statement is provably equivalent to some universal *complexity* statement?

Theorem: Not every universal statement is provably equivalent to some universal complexity statement.

(If it were true, it would imply the corollary above)

Proof: easily follows from some results of An. Muchnik and S. Positselsky about non-m-completeness of the overgraph of the complexity function. (Again we have a Kolmogorov complexity result that translates into proof theory)

Corollary and question

Corollary: Adding all true universal *complexity* statements as axioms, we can prove all true universal statements.

But is it true that every universal statement is provably equivalent to some universal *complexity* statement?

Theorem: Not every universal statement is provably equivalent to some universal complexity statement.

(If it were true, it would imply the corollary above)

Proof: easily follows from some results of An. Muchnik and S. Positselsky about non-*m*-completeness of the overgraph of the complexity function. (Again we have a Kolmogorov complexity result that translates into proof theory)

Why the non-m-completeness of the overgraph is enough

- $U = \{\langle x, n\rangle : \mathrm{C}(x) < n\}$
- enumerable but not decidable set
- is it complete?
- (is Turing complete, but) not *m*-complete
- follows from the results of An. Muchnik and S. Positselski
- if every universal statement were provably equivalent to some complexity statement, then U would be *m*-complete: to find out whether p terminates or not, find the statement $\mathrm{C}(x_p) \geqslant n_p$ provably equivalent to non-termination;
- $p \mapsto \langle x_p, n_p\rangle$ reduces the halting problem to U

Why the non-m-completeness of the overgraph is enough

- $U = \{\langle x, n\rangle : \mathrm{C}(x) < n\}$
- enumerable but not decidable set
- is it complete?
- (is Turing complete, but) not *m*-complete
- follows from the results of An. Muchnik and S. Positselski
- if every universal statement were provably equivalent to some complexity statement, then U would be m-complete: to find out whether p terminates or not, find the statement $\mathrm{C}(x_p) \geqslant n_p$ provably equivalent to non-termination;
- $p \mapsto \langle x_p, n_p\rangle$ reduces the halting problem to U

Why the non-m-completeness of the overgraph is enough

- $U = \{\langle x, n\rangle : \mathrm{C}(x) < n\}$
- enumerable but not decidable set
- is it complete?
- (is Turing complete, but) not *m*-complete
- follows from the results of An. Muchnik and S. Positselski
- if every universal statement were provably equivalent to some complexity statement, then *U* would be *m*-complete: to find out whether *p* terminates or not, find the statement $\mathrm{C}(x_p) \geqslant n_p$ provably equivalent to non-termination;
- $p \mapsto \langle x_p, n_p\rangle$ reduces the halting problem to *U*

Why the non-m-completeness of the overgraph is enough

- $U = \{\langle x, n\rangle : \mathrm{C}(x) < n\}$
- enumerable but not decidable set
- is it complete?
- (is Turing complete, but) not *m*-complete
- follows from the results of An. Muchnik and S. Positselski
- if every universal statement were provably equivalent to some complexity statement, then U would be *m*-complete: to find out whether p terminates or not, find the statement $\mathrm{C}(x_p) \geqslant n_p$ provably equivalent to non-termination;
- $p \mapsto \langle x_p, n_p\rangle$ reduces the halting problem to U

Why the non-m-completeness of the overgraph is enough

- $U = \{\langle x, n\rangle : \mathrm{C}(x) < n\}$
- enumerable but not decidable set
- is it complete?
- (is Turing complete, but) not *m*-complete
- follows from the results of An. Muchnik and S. Positselski
- if every universal statement were provably equivalent to some complexity statement, then U would be *m*-complete: to find out whether p terminates or not, find the statement $\mathrm{C}(x_p) \geqslant n_p$ provably equivalent to non-termination;
- $p \mapsto \langle x_p, n_p\rangle$ reduces the halting problem to U

Why the non-m-completeness of the overgraph is enough

- $U = \{\langle x, n\rangle : \mathrm{C}(x) < n\}$
- enumerable but not decidable set
- is it complete?
- (is Turing complete, but) not *m*-complete
- follows from the results of An. Muchnik and S. Positselski
- if every universal statement were provably equivalent to some complexity statement, then U would be m-complete: to find out whether p terminates or not, find the statement $\mathrm{C}(x_p) \geqslant n_p$ provably equivalent to non-termination;
- $p \mapsto \langle x_p, n_p\rangle$ reduces the halting problem to U

Why the non-m-completeness of the overgraph is enough

- $U = \{\langle x, n\rangle : \mathrm{C}(x) < n\}$
- enumerable but not decidable set
- is it complete?
- (is Turing complete, but) not *m*-complete
- follows from the results of An. Muchnik and S. Positselski
- if every universal statement were provably equivalent to some complexity statement, then U would be *m*-complete: to find out whether p terminates or not, find the statement $\mathrm{C}(x_p) \geqslant n_p$ provably equivalent to non-termination;
- $p \mapsto \langle x_p, n_p\rangle$ reduces the halting problem to U

Why the non-m-completeness of the overgraph is enough

- $U = \{\langle x, n\rangle : \mathrm{C}(x) < n\}$
- enumerable but not decidable set
- is it complete?
- (is Turing complete, but) not *m*-complete
- follows from the results of An. Muchnik and S. Positselski
- if every universal statement were provably equivalent to some complexity statement, then U would be *m*-complete: to find out whether p terminates or not, find the statement $\mathrm{C}(x_p) \geqslant n_p$ provably equivalent to non-termination;
- $p \mapsto \langle x_p, n_p\rangle$ reduces the halting problem to U

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is m-reducible to U
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p\colon\; \mathrm{C}(x_p) < n_p$ or $n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is m-reducible to U
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p\colon\ \mathrm{C}(x_p) < n_p \text{ or } n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is m-reducible to U
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p\colon\ \mathrm{C}(x_p) < n_p \text{ or } n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is *m*-reducible to *U*
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p\colon\ \mathrm{C}(x_p) < n_p \text{ or } n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is m-reducible to U
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p\colon\ \mathrm{C}(x_p) < n_p \text{ or } n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is m-reducible to U
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p\colon\ \mathrm{C}(x_p) < n_p \text{ or } n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is m-reducible to U
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p : \ \mathrm{C}(x_p) < n_p \text{ or } n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is *m*-reducible to *U*
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p : \mathrm{C}(x_p) < n_p$ or $n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is *m*-reducible to U
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p : \ \mathrm{C}(x_p) < n_p \text{ or } n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Why the overgraph is not complete

- K_0, K_1: two inseparable enumerable sets
- assume K_0 is m-reducible to U
- i.e, $p \in K_0 \Leftrightarrow \mathrm{C}(x_p) < n_p$
- then $\mathrm{C}(x_p) \geqslant n_p$ for all $p \in K_1$
- so all n_p for $p \in K_1$ are bounded by some c
- separator: $S = \{p\colon\ \mathrm{C}(x_p) < n_p$ or $n_p > c\}$
- S is decidable since only values of C not exceeding c matter, and they are determined by a finite table
- for $p \in K_0$ the first part is true
- for $p \in K_1$ both parts are false

Approximating complexity

- Kolmogorov complexity function $\mathrm{C}(\cdot)$ is non-computable (otherwise the first string of complexity at least n would have complexity $O(\log n)$)
- $\mathrm{C}(\cdot)$ cannot be approximated up to factor (say) 2
- moreover, one cannot separate (uniformly in n) incompressible n-bit strings from strings that have complexity at most $n/4$. Indeed, under this assumption we could effectively find a string of complexity greater than $n/4$ (given n) by taking a string from the separator.

Approximating complexity

- Kolmogorov complexity function $\mathrm{C}(\cdot)$ is non-computable (otherwise the first string of complexity at least n would have complexity $O(\log n)$)
- $\mathrm{C}(\cdot)$ cannot be approximated up to factor (say) 2
- moreover, one cannot separate (uniformly in n) incompressible n-bit strings from strings that have complexity at most $n/4$. Indeed, under this assumption we could effectively find a string of complexity greater than $n/4$ (given n) by taking a string from the separator.

Approximating complexity

- Kolmogorov complexity function $\mathrm{C}(\cdot)$ is non-computable (otherwise the first string of complexity at least n would have complexity $O(\log n)$)
- $\mathrm{C}(\cdot)$ cannot be approximated up to factor (say) 2
- moreover, one cannot separate (uniformly in n) incompressible n-bit strings from strings that have complexity at most $n/4$. Indeed, under this assumption we could effectively find a string of complexity greater than $n/4$ (given n) by taking a string from the separator.

Approximating complexity

- Kolmogorov complexity function $\mathrm{C}(\cdot)$ is non-computable (otherwise the first string of complexity at least n would have complexity $O(\log n)$)
- $\mathrm{C}(\cdot)$ cannot be approximated up to factor (say) 2
- moreover, one cannot separate (uniformly in n) incompressible n-bit strings from strings that have complexity at most $n/4$. Indeed, under this assumption we could effectively find a string of complexity greater than $n/4$ (given n) by taking a string from the separator.

Approximating complexity as a mass problem

- mass problem: a set of total functions called (its) *solutions* (Medvedev)
- $\mathcal{A}$ is reducible to $\mathcal{B}$ if there is an oracle machine that, being supplied by arbitrary solution of $\mathcal{B}$, computes some solution of $\mathcal{A}$.
- the separation problem (for the sets of incompressible strings and highly compressible strings) is reducible to the problem "approximate $\mathrm{C}(\cdot)$ up to factor 2"
- moreover, the halting problem is reducible to this separation problem

Approximating complexity as a mass problem

- mass problem: a set of total functions called (its) *solutions* (Medvedev)
- $\mathcal{A}$ is reducible to $\mathcal{B}$ if there is an oracle machine that, being supplied by arbitrary solution of $\mathcal{B}$, computes some solution of $\mathcal{A}$.
- the separation problem (for the sets of incompressible strings and highly compressible strings) is reducible to the problem "approximate $\mathrm{C}(\cdot)$ up to factor 2"
- moreover, the halting problem is reducible to this separation problem

Approximating complexity as a mass problem

- mass problem: a set of total functions called (its) *solutions* (Medvedev)
- $\mathcal{A}$ is reducible to $\mathcal{B}$ if there is an oracle machine that, being supplied by arbitrary solution of $\mathcal{B}$, computes some solution of $\mathcal{A}$.
- the separation problem (for the sets of incompressible strings and highly compressible strings) is reducible to the problem "approximate $\mathrm{C}(\cdot)$ up to factor 2"
- moreover, the halting problem is reducible to this separation problem

Approximating complexity as a mass problem

- mass problem: a set of total functions called (its) *solutions* (Medvedev)
- $\mathcal{A}$ is reducible to $\mathcal{B}$ if there is an oracle machine that, being supplied by arbitrary solution of $\mathcal{B}$, computes some solution of $\mathcal{A}$.
- the separation problem (for the sets of incompressible strings and highly compressible strings) is reducible to the problem "approximate $\mathrm{C}(\cdot)$ up to factor 2"
- moreover, the halting problem is reducible to this separation problem

Approximating complexity as a mass problem

- mass problem: a set of total functions called (its) *solutions* (Medvedev)
- $\mathcal{A}$ is reducible to $\mathcal{B}$ if there is an oracle machine that, being supplied by arbitrary solution of $\mathcal{B}$, computes some solution of $\mathcal{A}$.
- the separation problem (for the sets of incompressible strings and highly compressible strings) is reducible to the problem "approximate $\mathrm{C}(\cdot)$ up to factor 2"
- moreover, the halting problem is reducible to this separation problem

Reducing halting problem to separation

- for some n, consider the separator S_n that contains all strings of complexity at most $n/4$ and does not contain incompressible strings
- trying all programs in parallel, wait until all strings from S_n get a program shorter than n: time $T(n)$
- $T(n) \geqslant B(n/4 - O(\log n))$: indeed, if $t > T(n)$, then, knowing n and t, we may wait t steps and then consider a string that still is incompressible after t steps; it will have true complexity at least $n/4$
- so $T(n)$ can be used to decide the halting problem up to length $n/4 - O(\log n)$, and n is arbitrary

Reducing halting problem to separation

- for some n, consider the separator S_n that contains all strings of complexity at most $n/4$ and does not contain incompressible strings
- trying all programs in parallel, wait until all strings from S_n get a program shorter than n: time $T(n)$
- $T(n) \geqslant B(n/4 - O(\log n))$: indeed, if $t > T(n)$, then, knowing n and t, we may wait t steps and then consider a string that still is incompressible after t steps; it will have true complexity at least $n/4$
- so $T(n)$ can be used to decide the halting problem up to length $n/4 - O(\log n)$, and n is arbitrary

Reducing halting problem to separation

- for some n, consider the separator S_n that contains all strings of complexity at most $n/4$ and does not contain incompressible strings
- trying all programs in parallel, wait until all strings from S_n get a program shorter than n: time $T(n)$
- $T(n) \geqslant B(n/4 - O(\log n))$: indeed, if $t > T(n)$, then, knowing n and t, we may wait t steps and then consider a string that still is incompressible after t steps; it will have true complexity at least $n/4$
- so $T(n)$ can be used to decide the halting problem up to length $n/4 - O(\log n)$, and n is arbitrary

Reducing halting problem to separation

- for some n, consider the separator S_n that contains all strings of complexity at most $n/4$ and does not contain incompressible strings
- trying all programs in parallel, wait until all strings from S_n get a program shorter than n: time $T(n)$
- $T(n) \geqslant B(n/4 - O(\log n))$: indeed, if $t > T(n)$, then, knowing n and t, we may wait t steps and then consider a string that still is incompressible after t steps; it will have true complexity at least $n/4$
- so $T(n)$ can be used to decide the halting problem up to length $n/4 - O(\log n)$, and n is arbitrary

Reducing halting problem to separation

- for some n, consider the separator S_n that contains all strings of complexity at most $n/4$ and does not contain incompressible strings
- trying all programs in parallel, wait until all strings from S_n get a program shorter than n: time $T(n)$
- $T(n) \geqslant B(n/4 - O(\log n))$: indeed, if $t > T(n)$, then, knowing n and t, we may wait t steps and then consider a string that still is incompressible after t steps; it will have true complexity at least $n/4$
- so $T(n)$ can be used to decide the halting problem up to length $n/4 - O(\log n)$, and n is arbitrary

Proof-theoretic consequences

Theorem: Add axioms $\mathrm{C}(x) > n/4$ for all incompressible strings of length n. Then in this theory one can prove all universal statements of complexity $n/4 - O(\log n)$.

- wait until we find a short program (of length less than n) for every n-bit string that is not incompressible.
- let T be the corresponding time (numeral)
- using the axioms, we can prove that "every string that has complexity at most $n/4$, can be compressed at least by one bit in time T" (case analysis)
- formalizing the argument for $T > BB(n/4) - O(\log n)$, we prove that every program of length at most $n/4 - O(\log n)$ that does not terminate in T steps, does not terminate at all
- wait for T steps and prove nontermination for every nonterminating program of length at most $n/4 - O(\log n)$

Proof-theoretic consequences

Theorem: Add axioms $\mathrm{C}(x) > n/4$ for all incompressible strings of length n. Then in this theory one can prove all universal statements of complexity $n/4 - O(\log n)$.

- wait until we find a short program (of length less than n) for every n-bit string that is not incompressible.
- let T be the corresponding time (numeral)
- using the axioms, we can prove that "every string that has complexity at most $n/4$, can be compressed at least by one bit in time T" (case analysis)
- formalizing the argument for $T > BB(n/4) - O(\log n)$, we prove that every program of length at most $n/4 - O(\log n)$ that does not terminate in T steps, does not terminate at all
- wait for T steps and prove nontermination for every nonterminating program of length at most $n/4 - O(\log n)$

Proof-theoretic consequences

Theorem: Add axioms $\mathrm{C}(x) > n/4$ for all incompressible strings of length n. Then in this theory one can prove all universal statements of complexity $n/4 - O(\log n)$.

- wait until we find a short program (of length less than n) for every n-bit string that is not incompressible.
- let T be the corresponding time (numeral)
- using the axioms, we can prove that "every string that has complexity at most $n/4$, can be compressed at least by one bit in time T" (case analysis)
- formalizing the argument for $T > BB(n/4) - O(\log n)$, we prove that every program of length at most $n/4 - O(\log n)$ that does not terminate in T steps, does not terminate at all
- wait for T steps and prove nontermination for every nonterminating program of length at most $n/4 - O(\log n)$

Proof-theoretic consequences

Theorem: Add axioms $\mathrm{C}(x) > n/4$ for all incompressible strings of length n. Then in this theory one can prove all universal statements of complexity $n/4 - O(\log n)$.

- wait until we find a short program (of length less than n) for every n-bit string that is not incompressible.
- let T be the corresponding time (numeral)
- using the axioms, we can prove that "every string that has complexity at most $n/4$, can be compressed at least by one bit in time T" (case analysis)
- formalizing the argument for $T > BB(n/4) - O(\log n)$, we prove that every program of length at most $n/4 - O(\log n)$ that does not terminate in T steps, does not terminate at all
- wait for T steps and prove nontermination for every nonterminating program of length at most $n/4 - O(\log n)$

Proof-theoretic consequences

Theorem: Add axioms $\mathrm{C}(x) > n/4$ for all incompressible strings of length n. Then in this theory one can prove all universal statements of complexity $n/4 - O(\log n)$.

- wait until we find a short program (of length less than n) for every n-bit string that is not incompressible.
- let T be the corresponding time (numeral)
- using the axioms, we can prove that "every string that has complexity at most $n/4$, can be compressed at least by one bit in time T" (case analysis)
- formalizing the argument for $T > BB(n/4) - O(\log n)$, we prove that every program of length at most $n/4 - O(\log n)$ that does not terminate in T steps, does not terminate at all
- wait for T steps and prove nontermination for every nonterminating program of length at most $n/4 - O(\log n)$

Proof-theoretic consequences

Theorem: Add axioms $\mathrm{C}(x) > n/4$ for all incompressible strings of length n. Then in this theory one can prove all universal statements of complexity $n/4 - O(\log n)$.

- wait until we find a short program (of length less than n) for every n-bit string that is not incompressible.
- let T be the corresponding time (numeral)
- using the axioms, we can prove that "every string that has complexity at most $n/4$, can be compressed at least by one bit in time T" (case analysis)
- formalizing the argument for $T > BB(n/4) - O(\log n)$, we prove that every program of length at most $n/4 - O(\log n)$ that does not terminate in T steps, does not terminate at all
- wait for T steps and prove nontermination for every nonterminating program of length at most $n/4 - O(\log n)$

One weak complexity axiom is not enough

- conservative extension: no new provable statements of some type
- “quasi-conservative”: no new provable simple statements
- let $\mathrm{C}(x) = m$, and $m' < m$.
  Theorem: axiom $\mathrm{C}(x) > m'$ does not prove any new statements of complexity at most $m - m' + O(\log m')$.
- assume that it proves some φ. Then, knowing φ and m', we can enumerate all strings y such that $(\mathrm{C}(y) > m') \vdash \varphi$
- there are at most $O(2^{m'})$ of them unless φ is provable
- so each of these y's has complexity at most $O(\log m') + m' + \mathrm{C}(\varphi)$ (m', ordinal number in the enumeration and desciption of φ):
  $m \leqslant m' + O(\log m') + \mathrm{C}(\varphi)$

One weak complexity axiom is not enough

- conservative extension: no new provable statements of some type
- "quasi-conservative": no new provable simple statements
- let $\mathrm{C}(x) = m$, and $m' < m$.
  Theorem: axiom $\mathrm{C}(x) > m'$ does not prove any new statements of complexity at most $m - m' + O(\log m')$.
- assume that it proves some φ. Then, knowing φ and m', we can enumerate all strings y such that $(\mathrm{C}(y) > m') \vdash \varphi$
- there are at most $O(2^{m'})$ of them unless φ is provable
- so each of these y's has complexity at most $O(\log m') + m' + \mathrm{C}(\varphi)$ (m', ordinal number in the enumeration and desciption of φ):
  $m \leqslant m' + O(\log m') + \mathrm{C}(\varphi)$

One weak complexity axiom is not enough

- conservative extension: no new provable statements of some type
- “quasi-conservative”: no new provable simple statements
- let $C(x) = m$, and $m' < m$.
  Theorem: axiom $C(x) > m'$ does not prove any new statements of complexity at most $m - m' + O(\log m')$.
- assume that it proves some φ. Then, knowing φ and m', we can enumerate all strings y such that $(C(y) > m') \vdash \varphi$
- there are at most $O(2^{m'})$ of them unless φ is provable
- so each of these y's has complexity at most $O(\log m') + m' + C(\varphi)$ (m', ordinal number in the enumeration and desciption of φ):
  $m \leqslant m' + O(\log m') + C(\varphi)$

One weak complexity axiom is not enough

- conservative extension: no new provable statements of some type
- "quasi-conservative": no new provable simple statements
- let $\mathrm{C}(x) = m$, and $m' < m$.
  Theorem: axiom $\mathrm{C}(x) > m'$ does not prove any new statements of complexity at most $m - m' + O(\log m')$.
- assume that it proves some φ. Then, knowing φ and m', we can enumerate all strings y such that $(\mathrm{C}(y) > m') \vdash \varphi$
- there are at most $O(2^{m'})$ of them unless φ is provable
- so each of these y's has complexity at most $O(\log m') + m' + \mathrm{C}(\varphi)$ (m', ordinal number in the enumeration and desciption of φ):
  $m \leqslant m' + O(\log m') + \mathrm{C}(\varphi)$

One weak complexity axiom is not enough

- conservative extension: no new provable statements of some type
- "quasi-conservative": no new provable simple statements
- let $\mathrm{C}(x) = m$, and $m' < m$.
  Theorem: axiom $\mathrm{C}(x) > m'$ does not prove any new statements of complexity at most $m - m' + O(\log m')$.
- assume that it proves some φ. Then, knowing φ and m', we can enumerate all strings y such that $(\mathrm{C}(y) > m') \vdash \varphi$
- there are at most $O(2^{m'})$ of them unless φ is provable
- so each of these y's has complexity at most $O(\log m') + m' + \mathrm{C}(\varphi)$ (m', ordinal number in the enumeration and desciption of φ): $m \leqslant m' + O(\log m') + \mathrm{C}(\varphi)$

One weak complexity axiom is not enough

- conservative extension: no new provable statements of some type
- "quasi-conservative": no new provable simple statements
- let $\mathrm{C}(x) = m$, and $m' < m$.
  Theorem: axiom $\mathrm{C}(x) > m'$ does not prove any new statements of complexity at most $m - m' + O(\log m')$.
- assume that it proves some φ. Then, knowing φ and m', we can enumerate all strings y such that $(\mathrm{C}(y) > m') \vdash \varphi$
- there are at most $O(2^{m'})$ of them unless φ is provable
- so each of these y's has complexity at most $O(\log m') + m' + \mathrm{C}(\varphi)$ (m', ordinal number in the enumeration and desciption of φ):
  $m \leqslant m' + O(\log m') + \mathrm{C}(\varphi)$

One weak complexity axiom is not enough

- conservative extension: no new provable statements of some type
- “quasi-conservative”: no new provable simple statements
- let $\mathrm{C}(x) = m$, and $m' < m$.
  Theorem: axiom $\mathrm{C}(x) > m'$ does not prove any new statements of complexity at most $m - m' + O(\log m')$.
- assume that it proves some φ. Then, knowing φ and m', we can enumerate all strings y such that $(\mathrm{C}(y) > m') \vdash \varphi$
- there are at most $O(2^{m'})$ of them unless φ is provable
- so each of these y's has complexity at most $O(\log m') + m' + \mathrm{C}(\varphi)$ (m', ordinal number in the enumeration and desciption of φ):
  $m \leqslant m' + O(\log m') + \mathrm{C}(\varphi)$

Questions

- what are the possible values of complexity fot Π_1-statement $C(x) \geqslant m$ in the interval $[m - O(1), |x| + O(\log m)]$?
- is it true that for every (reasonable) decompressor every universal statement is equivalent to some statement of the form $C(x|y) \leqslant n$? (For some decompressors it is true for obvious reasons, even for statements of type $C(x|x) \leqslant 0$.)
- what are the possible consequences of, say, two statements $C(x) \geqslant n/2$ and $C(y) \geqslant n/2$ for two incompressible bit strings x and y of length n?

Questions

- what are the possible values of complexity fot Π_1-statement $\mathrm{C}(x) \geqslant m$ in the interval $[m - O(1), |x| + O(\log m)]$?
- is it true that for every (reasonable) decompressor every universal statement is equivalent to some statement of the form $\mathrm{C}(x|y) \leqslant n$? (For some decompressors it is true for obvious reasons, even for statements of type $\mathrm{C}(x|x) \leqslant 0$.)
- what are the possible consequences of, say, two statements $\mathrm{C}(x) \geqslant n/2$ and $\mathrm{C}(y) \geqslant n/2$ for two incompressible bit strings x and y of length n?

Questions

- what are the possible values of complexity fot Π_1-statement $C(x) \geqslant m$ in the interval $[m - O(1), |x| + O(\log m)]$?
- is it true that for every (reasonable) decompressor every universal statement is equivalent to some statement of the form $C(x|y) \leqslant n$? (For some decompressors it is true for obvious reasons, even for statements of type $C(x|x) \leqslant 0$.)
- what are the possible consequences of, say, two statements $C(x) \geqslant n/2$ and $C(y) \geqslant n/2$ for two incompressible bit strings x and y of length n?

Questions

- what are the possible values of complexity fot Π_1-statement $\mathrm{C}(x) \geqslant m$ in the interval $[m - O(1), |x| + O(\log m)]$?
- is it true that for every (reasonable) decompressor every universal statement is equivalent to some statement of the form $\mathrm{C}(x|y) \leqslant n$? (For some decompressors it is true for obvious reasons, even for statements of type $\mathrm{C}(x|x) \leqslant 0$.)
- what are the possible consequences of, say, two statements $\mathrm{C}(x) \geqslant n/2$ and $\mathrm{C}(y) \geqslant n/2$ for two incompressible bit strings x and y of length n?

Merci! Thanks! Спасибо!