Kolmogorov Complexity and Chaitin's Incompleteness Theorem

Shixiao Liu

Department of Philosophy, Peking University

Oct, 2018

Suppose I toss two coins for 15 times respectively, and the results are:

Coin I : 111111111111111

Coin II : 100110110001010

Suppose I toss two coins for 15 times respectively, and the results are:

Coin I : 111111111111111

Coin II : 100110110001010

Now consider a third coin:

Coin III : 111111100000000

Introduction

Suppose I toss two coins for 15 times respectively, and the results are:

Coin I : 111111111111111

Coin II : 100110110001010

Now consider a third coin:

Coin III : 111111100000000

Why do we feel that the sequence 100110110001010 is more "random" than 111111111111111 and 111111100000000? How do I express this notion of randomness in a more formal way?

Introduction

Consider the following sequences:

Sequence I : 111111111111111

Sequence II : 111111100000000

Sequence III : 100110110001010

Introduction

Consider the following sequences:

Sequence I : 111111111111111

Sequence II : 111111100000000

Sequence III : 100110110001010

To describe sequence I, I may say "1 times 15".
To describe sequence II, I may say "1 times 7 and 0 times 8".
To describe sequence III, I have no other way but to read it word by word, "100110110001010".

Introduction

Consider the following sequences:

Sequence I : 111111111111111

Sequence II : 111111100000000

Sequence III : 100110110001010

To describe sequence I, I may say "1 times 15".
To describe sequence II, I may say "1 times 7 and 0 times 8".
To describe sequence III, I have no other way but to read it word by word, "100110110001010".

This leads to the definition of Kolmogorov Complexity.

Coding Function

The first thing we need to clarify is the notion of “description”.

In the following, we suppose every string to be binary, and do not distinguish between a string and its enumeration as a natural number.

Coding Function

The first thing we need to clarify is the notion of "description".

In the following, we suppose every string to be binary, and do not distinguish between a string and its enumeration as a natural number.

Definition (Coding Function)

Any function $f : \mathrm{N} \to \mathrm{N}$ can be considered as a *coding function*.
The domain of f is the set of *code words*.
Moreover, we often require f to be computable.

Note that f may not be injective/surjective.

Complexity w.r.t. f

Given a coding function f, we define the complexity of a string (*i.e.* a natural number) with respect to f:

Definition (Complexity w.r.t. f)

$$C_f(n) = \begin{cases} \min\{l(s) : f(s) = n\} & f^{-1}(n) \text{ is non-empty} \\ \infty & \text{otherwise} \end{cases}$$

Complexity w.r.t. f

Given a coding function f, we define the complexity of a string (*i.e.* a natural number) with respect to f:

Definition (Complexity w.r.t. f)

$$C_f(n) = \begin{cases} \min\{l(s) : f(s) = n\} & f^{-1}(n) \text{ is non-empty} \\ \infty & \text{otherwise} \end{cases}$$

And since the complexity is f-related, we can compare the "power" of two coding functions.

Definition (Universal Function)

We say f *minorizes* g, if there is a constant c such that, for all n,

$$C_f(n) \leq C_g(n) + c.$$

And we say f is *universal*, if it minorizes every partial recursive function.

Invariance Theorem

And in fact, such a universal function does exist.

Theorem (Invariance Theorem)

There exists a universal function.

Note that, this theorem can be expressed by a Σ_6-formula in PA.

Invariance Theorem

And in fact, such a universal function does exist.

Theorem (Invariance Theorem)

There exists a universal function.

Note that, this theorem can be expressed by a Σ_6-formula in PA.

Proof.

Let f_0 be the function which the universal Turing machine computes, namely,

$$f_0(\langle n, x \rangle) = \Phi_n(x).$$

Then, given a partial recursive function f, let Φ_n be a Turing machine that computes f. The following holds for all m:

$$C_{f_0}(m) \leq C_f(m) + c_n,$$

where c_n depends only on n. □

With the help of Invariance Theorem, we can now define the complexity of a string which is coding-invariant (up to an additive constant).

Definition (Kolmogorov Complexity)

Fix f_0 as the function computed by the universal Turing machine. The *Kolmogorov Complexity* of a string is defined by

$$K(n) = C_{f_0}(n).$$

Kolmogorov Complexity

With the help of Invariance Theorem, we can now define the complexity of a string which is coding-invariant (up to an additive constant).

Definition (Kolmogorov Complexity)

Fix f_0 as the function computed by the universal Turing machine. The *Kolmogorov Complexity* of a string is defined by

$$K(n) = C_{f_0}(n).$$

The code we use for defining the Kolmogorov complexity of a string is essentially the following: first, describe a Turing machine; then, give that machine an input.
For instance, to code the string "10101010101010101010101010", we first select a Turing machine that duplicates the string "10" n times for n given, and then assign "$n = 12$".

Kolmogorov Complexity

With the help of Invariance Theorem, we can now define the complexity of a string which is coding-invariant (up to an additive constant).

Definition (Kolmogorov Complexity)

Fix f_0 as the function computed by the universal Turing machine. The *Kolmogorov Complexity* of a string is defined by

$$K(n) = C_{f_0}(n).$$

The code we use for defining the Kolmogorov complexity of a string is essentially the following: first, describe a Turing machine; then, give that machine an input.
For instance, to code the string "10101010101010101010101010", we first select a Turing machine that duplicates the string "10" n times for n given, and then assign "$n = 12$".

Note that, the definition of K depends on the particular enumeration of Turing machines. But one can prove that, if two enumerations are recursively isomorphic, their corresponding Kolmogorov complexity differ only by an additive constant.

Conditional Kolmogorov Complexity

We can also define the conditional complexity of a string. Given information m for free, what is the shortest string that expresses n?

Definition (Conditional complexity w.r.t. f)

$C_f(n|m) = \min\{l(s) : f(\langle m, s\rangle) = n\}$

Conditional Kolmogorov Complexity

We can also define the conditional complexity of a string. Given information m for free, what is the shortest string that expresses n?

Definition (Conditional complexity w.r.t. f)

$C_f(n|m) = \min\{l(s) : f(\langle m, s\rangle) = n\}$

And we can prove the parallel result:

Theorem (Invariance Theorem)

Let f_0 be the function computed by the universal Turing machine, such that $f_0(\langle m, \langle n, x\rangle\rangle) = \Phi_n(\langle m, x\rangle)$. Then for every partial recursive function f, there exists a constant c such that $C_{f_0}(n|m) \leq C_f(n|m) + c$ for all m, n.

Conditional Kolmogorov Complexity

We can also define the conditional complexity of a string. Given information m for free, what is the shortest string that expresses n?

Definition (Conditional complexity w.r.t. f)

$C_f(n|m) = \min\{l(s) : f(\langle m, s\rangle) = n\}$

And we can prove the parallel result:

Theorem (Invariance Theorem)

Let f_0 be the function computed by the universal Turing machine, such that $f_0(\langle m, \langle n, x\rangle\rangle) = \Phi_n(\langle m, x\rangle)$. Then for every partial recursive function f, there exists a constant c such that $C_{f_0}(n|m) \leq C_f(n|m) + c$ for all m, n.

Therefore we also can define the conditional Kolmogorov complexity.

Definition (Conditional Kolmogorov Complexity)

Let f_0 be as above. The *conditional Kolmogorov complexity* is defined by

$$K(n|m) = C_{f_0}(n|m)$$

We have the following observations that estimate the upper and lower bounds of Kolmogoroc complexity.

Observation

- *There exists a constant c such that $K(n) \leq l(n) + c$ for all n.*
- *There exists a constant c such that $K(n|m) \leq K(n) + c$ for all m, n.*
- *For each k, there exists n such that $K(n) \geq l(n) = k$.*

We have the following observations that estimate the upper and lower bounds of Kolmogoroc complexity.

Observation

- *There exists a constant c such that $K(n) \leq l(n) + c$ for all n.*
- *There exists a constant c such that $K(n|m) \leq K(n) + c$ for all m, n.*
- *For each k, there exists n such that $K(n) \geq l(n) = k$.*

Proof.

- Use the Turing machine that copies its input as output.
- Use the Turing machine that erases the first input, followed by the universal Turing machine.
- There are 2^k binary strings with length k, but only $2^k - 1$ shorter codes.

Incompressibility and Randomness

Recall that our initial intention was to study the randomness of strings.

Definition

We say a string is *incompressible* if $K(n) \geq l(n)$.
We say a string is *c-incompressible* if $K(n) \geq l(n) - c$.

Intuitively, incompressible strings are random. Now we're going to formalize what it means by saying a finite string is "random", and prove incompressible strings are indeed random in this sense.

Martin-Löf Test

The initial idea of Martin-Löf test starts by setting hierarchies for randomness. A string with no zero is absolutely nonrandom; but it is unnatural to fix a frequency p and say all strings with less than p of its total digits being zero are nonrandom. We have to agree that there are different levels of randomness.

Martin-Löf Test

The initial idea of Martin-Löf test starts by setting hierarchies for randomness. A string with no zero is absolutely nonrandom; but it is unnatural to fix a frequency p and say all strings with less than p of its total digits being zero are nonrandom. We have to agree that there are different levels of randomness.

For instance, a string with too many initial zeros is not random. Take

$$V_m = \{s : s \text{ starts with } m \text{ zeros}\}$$

and we have a hierarchy of randomness with $V_0 \supset V_1 \supset V_2 \supset ...$
The larger m is, the more nonrandom strings in V_m are.

Martin-Löf Test

The initial idea of Martin-Löf test starts by setting hierarchies for randomness. A string with no zero is absolutely nonrandom; but it is unnatural to fix a frequency p and say all strings with less than p of its total digits being zero are nonrandom. We have to agree that there are different levels of randomness.

For instance, a string with too many initial zeros is not random. Take

$$V_m = \{s : s \text{ starts with } m \text{ zeros}\}$$

and we have a hierarchy of randomness with $V_0 \supset V_1 \supset V_2 \supset ...$
The larger m is, the more nonrandom strings in V_m are.

We generalize the above case to obtain the notion of Martin-Löf test.

Martin-Löf Test

Definition (Martin-Löf Test)

Given a total function $\delta : \mathrm{N} \to \mathrm{N}$, we call the sets $V_m = \{x : \delta(x) \geq m\}$ *critical regions*. We say that δ is a *Martin-Löf test* if

1. The set $V = \{\langle m, x\rangle : x \in V_m\}$ is recursively enumerable;
2. $|\{x \in V_m \mid l(x) = n\}| \leq 2^{n-m}$ for all m, n.

By definition we have $V_0 \supset V_1 \supset V_2 \supset \ldots$ and each V_m is an *r.e.* set

Definition (Martin-Löf Test)

Given a total function $\delta : \mathrm{N} \to \mathrm{N}$, we call the sets $V_m = \{x : \delta(x) \geq m\}$ *critical regions*. We say that δ is a *Martin-Löf test* if

1. The set $V = \{\langle m, x\rangle : x \in V_m\}$ is recursively enumerable;
2. $|\{x \in V_m \mid l(x) = n\}| \leq 2^{n-m}$ for all m, n.

By definition we have $V_0 \supset V_1 \supset V_2 \supset \ldots$ and each V_m is an *r.e.* set

Again, we want a universal test instead of a specific one.

Definition (Universal Martin-Löf Test)

We say δ_0 is a *universal Martin-Löf test*, if for every Martin-Löf test δ, there is a constant c such that $\delta_0(x) \geq \delta(x) - c$ for all x.

Intuitively, this means that the universal test considers every string to be more nonrandom than any other test, if an additive deviation is ignored. Therefore, it is the tightest test we can ever find.

And there is indeed a universal test. It is generated by applying the Kolmogorov complexity.

Theorem

There exists a universal Martin-Löf test, namely $\delta_0(x) = l(x) - K(x|l(x)) - 1$.

And there is indeed a universal test. It is generated by applying the Kolmogorov complexity.

Theorem

There exists a universal Martin-Löf test, namely $\delta_0(x) = l(x) - K(x|l(x)) - 1$.

Proof.

We omit the proof of $\{\langle m, x\rangle : \delta_0(x) \geq m\}$ being recursively enumerable, which is done by building a non-increasing sequence $\phi_n(x)$ of recursive functions whose limit is $K(x|l(x))$.

And there is indeed a universal test. It is generated by applying the Kolmogorov complexity.

Theorem

There exists a universal Martin-Löf test, namely $\delta_0(x) = l(x) - K(x|l(x)) - 1$.

Proof.

We omit the proof of $\{\langle m, x\rangle : \delta_0(x) \geq m\}$ being recursively enumerable, which is done by building a non-increasing sequence $\phi_n(x)$ of recursive functions whose limit is $K(x|l(x))$.

For size of critical regions, $V_m = \{x : K(x|l(x)) \leq l(x) - m - 1\}$, and there are only $2^{l(x)-m} - 1$ many strings with length less than or equal to $l(x) - m - 1$.

And there is indeed a universal test. It is generated by applying the Kolmogorov complexity.

Theorem

There exists a universal Martin-Löf test, namely $\delta_0(x) = l(x) - K(x|l(x)) - 1$.

Proof.

We omit the proof of $\{\langle m, x\rangle : \delta_0(x) \geq m\}$ being recursively enumerable, which is done by building a non-increasing sequence $\phi_n(x)$ of recursive functions whose limit is $K(x|l(x))$.

For size of critical regions, $V_m = \{x : K(x|l(x)) \leq l(x) - m - 1\}$, and there are only $2^{l(x)-m} - 1$ many strings with length less than or equal to $l(x) - m - 1$.

For universality, let $\{\delta_i\}_{i\geq 1}$ be an effective enumeration of all Martin-Löf tests. Fix a test δ_i and an x. We sort all strings s of length $l(x)$ in descending order of $\langle \delta_i(s), s\rangle$, and suppose x is the j-th element. Then, there is a Turing machine that computes x with input $\langle l(x), i, j\rangle$. By Invariance Theorem, $K(x|l(x)) \leq l(i) + l(j) + c$. On the other hand, $j \leq |V_{\delta_i(x)}|$ and therefore $l(j) \leq l(x) - \delta_i(x)$. In conclusion, $\delta_i(x) \leq l(x) - K(x|l(x)) + c'$. □

And there is indeed a universal test. It is generated by applying the Kolmogorov complexity.

Theorem

There exists a universal Martin-Löf test, namely $\delta_0(x) = l(x) - K(x|l(x)) - 1$.

Proof.

We omit the proof of $\{\langle m, x\rangle : \delta_0(x) \geq m\}$ being recursively enumerable, which is done by building a non-increasing sequence $\phi_n(x)$ of recursive functions whose limit is $K(x|l(x))$.

For size of critical regions, $V_m = \{x : K(x|l(x)) \leq l(x) - m - 1\}$, and there are only $2^{l(x)-m} - 1$ many strings with length less than or equal to $l(x) - m - 1$.

For universality, let $\{\delta_i\}_{i\geq 1}$ be an effective enumeration of all Martin-Löf tests. Fix a test δ_i and an x. We sort all strings s of length $l(x)$ in descending order of $\langle \delta_i(s), s\rangle$, and suppose x is the j-th element. Then, there is a Turing machine that computes x with input $\langle l(x), i, j\rangle$. By Invariance Theorem, $K(x|l(x)) \leq l(i) + l(j) + c$. On the other hand, $j \leq |V_{\delta_i(x)}|$ and therefore $l(j) \leq l(x) - \delta_i(x)$. In conclusion, $\delta_i(x) \leq l(x) - K(x|l(x)) + c'$. □

We conclude that randomness defined by Martin-Löf test and incompressibility defined by Kolmogorov complexity coincide.

Undecidability of K

The function K does not look like a recursive function. $K(n)$ is the length of the shortest input with n as the output; however, whether a calculation of a Turing machines halts with the given output is not decidable. We're going to prove that K is indeed not a recursive function.

Undecidability of K

The function K does not look like a recursive function. $K(n)$ is the length of the shortest input with n as the output; however, whether a calculation of a Turing machines halts with the given output is not decidable. We're going to prove that K is indeed not a recursive function.

Theorem (Noncomputability Theorem)

K is not partial recursive.

Proof.

We prove an even stronger statement: there is no partial recursive function with infinite domain that coincides with K on its domain.

Undecidability of K

The function K does not look like a recursive function. $K(n)$ is the length of the shortest input with n as the output; however, whether a calculation of a Turing machines halts with the given output is not decidable. We're going to prove that K is indeed not a recursive function.

Theorem (Noncomputability Theorem)

K is not partial recursive.

Proof.

We prove an even stronger statement: there is no partial recursive function with infinite domain that coincides with K on its domain.

Suppose not. Let ϕ be such a function. Its domain is an infinite *r.e.* set and thus contains an infinite recursive set, denoted by A. Take

$$\psi(x) = \min\{y \in A : K(y) \geq x\}.$$

It is total recursive because K and ϕ agree on A. By definition of ψ, we have $K(\psi(x)) \geq x$.

Undecidability of K

The function K does not look like a recursive function. $K(n)$ is the length of the shortest input with n as the output; however, whether a calculation of a Turing machines halts with the given output is not decidable. We're going to prove that K is indeed not a recursive function.

Theorem (Noncomputability Theorem)

K is not partial recursive.

Proof.

We prove an even stronger statement: there is no partial recursive function with infinite domain that coincides with K on its domain.

Suppose not. Let ϕ be such a function. Its domain is an infinite *r.e.* set and thus contains an infinite recursive set, denoted by A. Take

$$\psi(x) = \min\{y \in A : K(y) \geq x\}.$$

It is total recursive because K and ϕ agree on A. By definition of ψ, we have $K(\psi(x)) \geq x$. However, by Invariance Theorem,

$$K(\psi(x)) \leq C_\psi(\psi(x)) + c \leq l(x) + c$$

and thus $x \leq l(x) + c$, a contradiction. □

Undecidability of K

With a similar proof, we'll see another interesting property of the function K.

Theorem

Let $m(x) = \min\{K(y) : y \geq x\}$. *Then*

1. $m(x)$ *goes monotonically to infinity;*
2. *but it does so more slowly than any unbounded partial recursive function; more precisely, for any monotonic unbounded partial recursive function* $\phi(x)$, *we have* $m(x) < \phi(x)$ *except for finitely many* x.

Undecidability of K

With a similar proof, we'll see another interesting property of the function K.

Theorem

Let $m(x) = \min\{K(y) : y \geq x\}$. *Then*

1. $m(x)$ *goes monotonically to infinity;*
2. *but it does so more slowly than any unbounded partial recursive function; more precisely, for any monotonic unbounded partial recursive function* $\phi(x)$, *we have* $m(x) < \phi(x)$ *except for finitely many* x.

Note that, a function that goes to infinity faster than any recursive function is easy to build. Let $\{f_i\}_{i \in \mathbb{N}}$ be an enumeration of all partial recursive functions, and $F(n) = \max_{i \leq n} f_i(n)$ meets the requirement.

Undecidability of K

With a similar proof, we'll see another interesting property of the function K.

Theorem

Let $m(x) = \min\{K(y) : y \geq x\}$. Then

1. *$m(x)$ goes monotonically to infinity;*
2. *but it does so more slowly than any unbounded partial recursive function; more precisely, for any monotonic unbounded partial recursive function $\phi(x)$, we have $m(x) < \phi(x)$ except for finitely many x.*

Note that, a function that goes to infinity faster than any recursive function is easy to build. Let $\{f_i\}_{i \in \mathbb{N}}$ be an enumeration of all partial recursive functions, and $F(n) = \max_{i \leq n} f_i(n)$ meets the requirement.

Proof.

Monotonicity is obvious.
For unboundedness, it suffices to see that there are only finitely many codes with length $\leq n$ for any given n.

Proof. (Cont'd).

Now assume for contradiction that there is a monotonic unbounded partial recursive function ϕ such that $\phi(x) \leq m(x)$ for infinitely many x. The domain of ϕ is an infinite *r.e.* set. It contains an infinite recursive subset A.

Proof. (Cont'd).

Now assume for contradiction that there is a monotonic unbounded partial recursive function ϕ such that $\phi(x) \leq m(x)$ for infinitely many x. The domain of ϕ is an infinite *r.e.* set. It contains an infinite recursive subset A.

Take

$$\psi(x) = \begin{cases} \phi(\max\{y \in A : y \leq x\}) & x \geq \min A \\ 0 & \text{otherwise} \end{cases}$$

Then ψ is total recursive, monotonic, unbounded, and $\psi(x) \leq m(x)$ for infinitely many x.

Proof. (Cont'd).

Now assume for contradiction that there is a monotonic unbounded partial recursive function ϕ such that $\phi(x) \leq m(x)$ for infinitely many x. The domain of ϕ is an infinite *r.e.* set. It contains an infinite recursive subset A.

Take

$$\psi(x) = \begin{cases} \phi(\max\{y \in A : y \leq x\}) & x \geq \min A \\ 0 & \text{otherwise} \end{cases}$$

Then ψ is total recursive, monotonic, unbounded, and $\psi(x) \leq m(x)$ for infinitely many x.

Define $M(x) = \max\{y : K(y) \leq x\}$ and $F(x) = \max\{y : \psi(y) \leq x + 1\}$. Then,

$$M(x) + 1 = \min\{y : m(y) > x\} \leq \min\{y : \psi(y) > x\} \leq F(x)$$

for infinitely many x. In other words, $F(x) > M(x)$ for infinitely many x. Now that F is total recursive, this implies that $K(F(x)) > x$ for infinitely many x.

Proof. (Cont'd).

Now assume for contradiction that there is a monotonic unbounded partial recursive function ϕ such that $\phi(x) \leq m(x)$ for infinitely many x. The domain of ϕ is an infinite *r.e.* set. It contains an infinite recursive subset A.

Take

$$\psi(x) = \begin{cases} \phi(\max\{y \in A : y \leq x\}) & x \geq \min A \\ 0 & \text{otherwise} \end{cases}$$

Then ψ is total recursive, monotonic, unbounded, and $\psi(x) \leq m(x)$ for infinitely many x.

Define $M(x) = \max\{y : K(y) \leq x\}$ and $F(x) = \max\{y : \psi(y) \leq x + 1\}$. Then,

$$M(x) + 1 = \min\{y : m(y) > x\} \leq \min\{y : \psi(y) > x\} \leq F(x)$$

for infinitely many x. In other words, $F(x) > M(x)$ for infinitely many x. Now that F is total recursive, this implies that $K(F(x)) > x$ for infinitely many x. However, by Invariance Theorem,

$$K(F(x)) \leq C_F(F(x)) + c \leq l(x) + c$$

and thus $x < l(x) + c$ for infinitely many x, a contradiction. □

Berry Paradox

Consider the following description:

> The smallest positive integer not definable in under sixty letters.

The Berry paradox says, since there are only finitely many descriptions in under sixty letters, such integer exists. But then, this integer is defined by this sentence, which is under sixty letters.

Berry Paradox

Consider the following description:

> The smallest positive integer not definable in under sixty letters.

The Berry paradox says, since there are only finitely many descriptions in under sixty letters, such integer exists. But then, this integer is defined by this sentence, which is under sixty letters.

In Kolmogorov's theory of coding, this paradox does not appear. Recall that we demand every coding function to be recursive. And if you try to find "the smallest positive integer not definable in under sixty letters" in an effective way, the program ends up in a loop.

Berry Paradox

Consider the following description:

> The smallest positive integer not definable in under sixty letters.

The Berry paradox says, since there are only finitely many descriptions in under sixty letters, such integer exists. But then, this integer is defined by this sentence, which is under sixty letters.

In Kolmogorov's theory of coding, this paradox does not appear. Recall that we demand every coding function to be recursive. And if you try to find "the smallest positive integer not definable in under sixty letters" in an effective way, the program ends up in a loop.

However, the idea of Berry paradox does give us an important insight, which leads to the proof of Chaitin's incompleteness theorem.

Chaitin's Incompleteness Theorem

Theorem (Chaitin's Incompleteness Theorem)

If T is a consistent recursively enumerable theory that extends PA, then there this a constant c such that $K(\ulcorner n \urcorner) \geq \ulcorner c \urcorner$ is not provable in T for all n.

Chaitin's Incompleteness Theorem

Theorem (Chaitin's Incompleteness Theorem)

If T is a consistent recursively enumerable theory that extends PA, then there this a constant c such that $K(\ulcorner n \urcorner) \geq \ulcorner c \urcorner$ is not provable in T for all n.

Proof.

Suppose not. We abuse notation; for each c, there exists n such that $K(n) \geq c$ is provable in T.
Let Φ_m be the Turing machine that enumerates theorems of T. Now we can modify Φ_m into a Turing machine such that, given c as an input, by enumerating all theorems of T with Φ_m, the machine finds the first proof of $K(n) \geq c$ for some n, and send n as the output. Therefore, we have a coding of this n with length $l(c)$. By Invariance Theorem, $K(n) \leq l(c) + c'$, where c' depends only upon m. However, now that we have $c \leq K(n) \leq l(c) + c'$, for c sufficiently large this results in a contradiction. $\square$

Chaitin's Incompleteness Theorem

Theorem (Chaitin's Incompleteness Theorem)

If T is a consistent recursively enumerable theory that extends PA, then there this a constant c such that $K(\ulcorner n \urcorner) \geq \ulcorner c \urcorner$ is not provable in T for all n.

Proof.

Suppose not. We abuse notation; for each c, there exists n such that $K(n) \geq c$ is provable in T.
Let Φ_m be the Turing machine that enumerates theorems of T. Now we can modify Φ_m into a Turing machine such that, given c as an input, by enumerating all theorems of T with Φ_m, the machine finds the first proof of $K(n) \geq c$ for some n, and send n as the output. Therefore, we have a coding of this n with length $l(c)$. By Invariance Theorem, $K(n) \leq l(c) + c'$, where c' depends only upon m. However, now that we have $c \leq K(n) \leq l(c) + c'$, for c sufficiently large this results in a contradiction. □

Note that, although this is a proof by contradiction, we do obtain a concrete constant c witnessing the theorem. In fact, it suffices to choose c such that $c > l(c) + c'$.

Chaitin's Incompleteness Theorem

Theorem (Chaitin's Incompleteness Theorem)

If T is a consistent recursively enumerable theory that extends PA, then there this a constant c such that $K(\ulcorner n \urcorner) \geq \ulcorner c \urcorner$ is not provable in T for all n.

Proof.

Suppose not. We abuse notation; for each c, there exists n such that $K(n) \geq c$ is provable in T.
Let Φ_m be the Turing machine that enumerates theorems of T. Now we can modify Φ_m into a Turing machine such that, given c as an input, by enumerating all theorems of T with Φ_m, the machine finds the first proof of $K(n) \geq c$ for some n, and send n as the output. Therefore, we have a coding of this n with length $l(c)$. By Invariance Theorem, $K(n) \leq l(c) + c'$, where c' depends only upon m. However, now that we have $c \leq K(n) \leq l(c) + c'$, for c sufficiently large this results in a contradiction. □

Note that, although this is a proof by contradiction, we do obtain a concrete constant c witnessing the theorem. In fact, it suffices to choose c such that $c > l(c) + c'$.
Question: Where in the proof do we require the consistency of T?

And as a corollary, we have:

Corollary (Gödel's First Incompleteness Theorem)

If T is a consistent recursively enumerable theory that extends PA, then there is a formula ϕ such that both ϕ and $\neg\phi$ are not provable in T.

Proof.

We've proven that there are incompressible strings of each length, and thus there are infinitely many n such that $\neg K(\ulcorner n \urcorner) \geq \ulcorner c \urcorner$ is not provable. $\square$

And as a corollary, we have:

Corollary (Gödel's First Incompleteness Theorem)

If T is a consistent recursively enumerable theory that extends PA, then there is a formula ϕ such that both ϕ and $\neg\phi$ are not provable in T.

Proof.

We've proven that there are incompressible strings of each length, and thus there are infinitely many n such that $\neg K(\ulcorner n \urcorner) \geq \ulcorner c \urcorner$ is not provable. □

Note that, Gödel's second incompleteness theorem does NOT follow from the above proof.

Reflections on Chaitin's Incompleteness Theorem

Gödel's Incompleteness Theorem – Russell's Paradox
Chaition's Incompleteness Theorem – Berry Paradox
But somehow, they're closely related.

Reflections on Chaitin's Incompleteness Theorem

Gödel's Incompleteness Theorem – Russell's Paradox
Chaition's Incompleteness Theorem – Berry Paradox
But somehow, they're closely related.

The following algorithm seems to generate a proof of $K(n) \geq c$ in T: list all strings s with length less than c as input, and check whether we can prove the universal Turing machine halts with output n.
However, the algorithm does not work because we cannot effectively decide whether a Turing machine halts. Moreover, since now we're given only finitely many strings, there has to be one particular string s_0, such that we cannot decide the halting problem with s_0 as input.
Question: What is that particular input s_0?

Gödel's Incompleteness Theorem – Russell's Paradox
Chaition's Incompleteness Theorem – Berry Paradox
But somehow, they're closely related.

The following algorithm seems to generate a proof of $K(n) \geq c$ in T: list all strings s with length less than c as input, and check whether we can prove the universal Turing machine halts with output n.
However, the algorithm does not work because we cannot effectively decide whether a Turing machine halts. Moreover, since now we're given only finitely many strings, there has to be one particular string s_0, such that we cannot decide the halting problem with s_0 as input.
Question: What is that particular input s_0?

The answer lies in the proof of Chaitin's Incompleteness Theorem.
We constructed a Turing machine that enumerates all theorems of T to see whether $K(n) \geq c$ is provable. With little modification, we can also construct a Turing machine that checks whether $0 \neq 0$ is a theorem of T. We've supposed T is consistent, and thus such a Turing machine does not halt. On the other hand, by Gödel's incompleteness theorem, T cannot prove that this Turing machine does not halt. This is the input we're looking for.