SJÄLVSTÄNDIGA ARBETEN I MATEMATIK

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET

Chaitin's incompleteness theorem

av

Erik Thormarker

2015 - No 11

MATEMATISKA INSTITUTIONEN, STOCKHOLMS UNIVERSITET, 106 91 STOCKHOLM

Chaitin's incompleteness theorem

Erik Thormarker

Självständigt arbete i matematik 15 högskolepoäng, grundnivå

Handledare: Erik Palmgren

2015

Abstract

In this thesis we look at an incompleteness result by Gregory Chaitin. Roughly Chaitin's result tells us that under certain assumptions on a formal system there exists a constant c such that no statements of the form "$C(n) > c$", where $C(n)$ is the *Kolmogorov complexity* of a natural number n, are provable in that formal system. After Chaitin's result there has been a discussion concerning what the theorem actually implies, we will give a summary of this discussion. We also compare Chaitin's result to Gödel's famous incompleteness theorems and discuss if Chaitin's result can be said to be as strong as Gödel's result. Finally we will look at some developments in recent years with a connection to Chaitin's result.

Contents

1 Introduction

Chaitin's incompleteness theorem is an incompleteness result in logic similar to Gödel's famous incompleteness theorem. Much like in Gödel's original proof Chaitin formalizes a paradox to obtain his result. To formalize the paradox Chaitin uses Kolmogorov complexity, which is the idea of measuring how complex a string is by the length of the shortest description that produces the string when the description is used as input to a Turing machine. Chaitin's result is that for a formal system there is a constant c such that no statements of the form "$C(n) > c$", where $C(n)$ is the Kolmogorov complexity of n, are provable in the system for any string n. However by an elementary counting argument we see that such strings must in fact exist. In fact it is even provable in the system that such strings must exist, but by Chaitin's result we can not prove for any specific string that it has such high complexity. Thus this is an incompleteness result in the sense that there are true statements that are not provable in the system.

In Section 2 we look at some preliminaries needed for discussing Chaitin's result. These include recursive functions, Turing machines and some basic results about Kolmogorov complexity. We also state Gödel's two incompleteness theorems as we will compare Chaitin's theorem with Gödel's result later in the text.

In Section 3 we state and prove the main result, Chaitin's incompleteness theorem. After this we summarize a discussion that started officially, as far as this author knows, with van Lambalgen (1989) that criticized the principle

> A set of axioms of complexity N cannot yield a theorem of complexity substantially greater than N

that had been mentioned in connection with Chaitin's result. We end the section by comparing Chaitin's result to Gödel's two incompleteness theorems, asking whether Chaitin's result is as strong as Gödel's in the sense that we can prove statements equivalent to Gödel's using Chaitin's theorem. We also look at other similarities and differences between the two results.

In Section 4 we look at some recent results that are connected to Chaitin's result in some way. Calude and Jürgensen (2005) suggests a new complexity measure that satisfies the principle discussed in Section 3, however as we shall see there are some problems with this approach. In Ibuka et al. (2011) we look at some details of results that were suggested during the same discussion. Finally in Bienvenu et al. (2014) the authors study the strength of the theory we get if we add all true previously mentioned statements of the form "$C(n) > c$" as axioms of the theory. We will see that this new theory proves whether any given Turing machine halts or not.

2 Preliminaries

We will follow Boolos et al. (2007) for most of the material in these preliminaries.

2.1 Arithmetical theories

The *language of arithmetic* consists of a constant 0, the binary relation $<$, the unary function s and the two binary functions $\cdot$ and $+$. The function s is sometimes called the successor function. If we interpret the language of arithmetic as we do in ordinary mathematics, where $s(0) = 1$, $s(1) = 2$ and so on, we get the *standard interpretation*. When we say a sentence in the language of arithmetic is *true* this means that it is true in the standard interpretation. We will tacitly assume throughout this text that the formulas mentioned are in the language of arithmetic. By $\overline{n}$ for a natural number n we mean the numeral of n which is shorthand for n applications of the function s to 0. For simplicity we will usually omit the numeral notation, for instance if we say $\vdash_{\mathbf{T}} \varphi(n)$ for some natural number n, then we mean the well defined $\vdash_{\mathbf{T}} \varphi(\overline{n})$. The axioms of minimal arithmetic **Q** are (letting x' be shorthand for $s(x)$)

$$\forall x(0 \neq x')$$
$$\forall x \forall y(x' = y' \rightarrow x = y)$$
$$\forall x(x + 0 = x)$$
$$\forall x \forall y(x + y' = (x + y)')$$
$$\forall x(x \cdot 0 = 0$$
$$\forall x \forall y(x \cdot y' = (x \cdot y) + x)$$
$$\forall x(\neg x < 0)$$
$$\forall x \forall y(x < y' \leftrightarrow (x < y \vee x = y))$$
$$\forall x(0 < x \leftrightarrow x \neq 0)$$
$$\forall x \forall y(x' < y \leftrightarrow (x < y \wedge y \neq x'))$$

The axioms of Peano arithmetic **PA** are those of **Q** together with the induction scheme

$$\varphi(0) \wedge \forall x(\varphi(x) \rightarrow \varphi(x')) \rightarrow \forall x \varphi(x)$$

where $\varphi(x)$ is any formula. The standard interpretation is a model of **PA** and is often referred to as the *standard model*.

We say a quantifier is bounded if it occurs as "$\exists x < t(\varphi(x))$" or "$\forall x < t(\varphi(x))$" for a term t, with $\exists x < t(\varphi(x))$ being shorthand for $\exists x(x < t \wedge \varphi(x))$ and $\forall x < t(\varphi(x))$ shorthand for $\forall(x < t \rightarrow \varphi(x))$.

Definition 2.1.1. We call a formula of the form $\exists x\varphi(x)$, where any quantifier in φ is bounded, a Σ_1-*formula*.

Definition 2.1.2. We call a formula of the form $\forall x\varphi(x)$, where any quantifier in φ is bounded, a Π_1-*formula*.

Theorem 2.1.3. *A Σ_1-sentence is provable in* $\mathbf{Q}$ *if and only if it is true.*

Proof. See Boolos et al. (2007) Theorem 16.13. □

2.2 Recursive functions

We can encode the natural numbers as $0, 0', 0'', 0''', ...$ For any such encoded natural numbers $n_0, ..., n_m$ we define the following functions:
the zero function

$$z(n_0) = 0$$

the successor function

$$s(n_0) = n_0'$$

and the class of projection functions

$$\text{proj}_i^m(n_0, ..., n_i, ..., n_m) = n_i$$

Definition 2.2.1. We call the zero function, the successor function and the class of projection functions the *basic functions*.

Definition 2.2.2. For functions $h, g_0, ...g_k$ we define *composition* as defining a function f by

$$f(n_0, ..., n_m) = h(g_0(n_0, ..., n_m), ..., g_k(n_0, ..., n_m))$$

Definition 2.2.3. For functions h, g we define *primitive recursion* as defining a function f by

$$f(n_0, 0) = h(n_0), \quad f(n_0, n_1') = g(n_0, n_1, f(n_0, n_1))$$

Definition 2.2.4. We call the functions definable through the basic functions, composition and primitive recursion the *primitive recursive* functions.

We note that a primitive recursive function is a total function from the natural numbers into the natural numbers.

Definition 2.2.5. For a function $f(n_0, ..., n_m, n_{m+1})$ we define *minimization* as

$$\min[f](n_0, ..., n_m) = \begin{cases} x \quad \text{, if } f(n_0, ..., n_m, x) = 0 \text{ and } t < x \Rightarrow f(n_0, ..., n_m, t) > 0 \\ \text{undefined , if no such } x \text{ exist} \end{cases}$$

Note that $f(n_0, ..., n_m, t) > 0$ implies $f(n_0, ..., n_m, t)$ is defined.

Definition 2.2.6. We call the functions definable through the basic functions, composition, primitive recursion and minimization the *recursive* functions.

Unlike our primitive recursive functions, the recursive functions are partial functions from the natural numbers into the natural numbers. This is because the use of minimization might make a recursive function undefined for some n.

Definition 2.2.7. A set $S \subset \mathbb{N}$ is *recursively enumerable* if it is the domain of some recursive function.

Definition 2.2.8. A function $f : \mathbb{N} \to \{0, 1\}$ is the *characteristic function* of a set $S \subset \mathbb{N}$ if

$$n \in S \Rightarrow f(n) = 1$$
$$n \notin S \Rightarrow f(n) = 0$$

Definition 2.2.9. A set $S \subset \mathbb{N}$ is *recursive* if it has a recursive characteristic function.

Definition 2.2.10. We call a function $f(n)$ representable in a theory $\mathbf{T}$ if there exists a formula $\varphi(x, y)$ such that $\vdash_{\mathbf{T}} \forall y(\varphi(n, y) \leftrightarrow y = m)$ if and only if $f(n) = m$. We then say that f is representable in $\mathbf{T}$ by φ.

Theorem 2.2.11. *Every recursive function is representable in* $\mathbf{Q}$ *by a* Σ_1*-formula.*

Proof. See Boolos et al. (2007) Theorem 16.14. □

2.3 Provability

A Gödel numbering is a way of encoding the formulas in the language of arithmetic as natural numbers. We will not go in to details on this and refer to Boolos et al. (2007) for a more detailed treatment. With modern computers encoding complex information as binary numbers the claim that

Gödel numberings are possible to construct is hardly implausible. We denote the Gödel number of a formula φ by $\ulcorner\varphi\urcorner$. We also assume we can encode derivations in a theory into the natural numbers using a Gödel numbering.

We say a theory is *recursively enumerable* if the set consisting of the Gödel numbers of the theorems of the theory is recursively enumerable. Both our theories **Q** and **PA** are recursively enumerable since one can show that it is possible to define a recursive function that is defined for the natural number n if and only if n is the Gödel number of a theorem for which there exists a proof in natural deduction whose only undischarged assumptions are among the axioms or instances of axiom schemes of the theory. One way to find such proofs is to in increasing order check all natural numbers and see if we can find a natural number that is the Gödel number of a proof satisfying our requirements. If the reader finds the existence of the function described hard to believe without seeing a detailed proof, then let us note that another way to see why the function exists is through the notion of effectively computable functions and Church's thesis, both concepts will be explained in Section 2.4.

Given a recursively enumerable theory **T** extending **Q** we will for any sentence φ assume that we can define a Σ_1-sentence $\mathrm{Pr}_{\mathbf{T}}(\ulcorner\varphi\urcorner)$ that expresses "there exists an y that is the Gödel number of a derivation of φ in **T**". A theory is incomplete if there exists a sentence expressible in the language of the theory which is neither provable nor disprovable in the theory. We say a sentence of the language of a theory is undecidable in the theory if it is neither provable nor disprovable.

Theorem 2.3.1. (Gödel's first incompleteness theorem) *Suppose* **T** *is a consistent and recursively enumerable extension of* **Q**, *then* **T** *is incomplete.*

Proof. See Boolos et al. (2007) Theorem 17.7. □

Originally Gödel proved his first incompleteness theorem under the stronger assumption of ω-consistency, which means that if a theory proves $\varphi(0)$, $\varphi(0')$, $\varphi(0'')$..., then the theory does not prove $\exists x\neg\varphi(x)$. Gödel formalized a version of the liar paradox "This sentence is false." to obtain his result. The form stated in Theorem 2.3.1 is due to Rosser. We should mention that the proof of Theorem 2.3.1 actually produces an explicit sentence that is in fact undecidable in the theory.

Theorem 2.3.2. (The Hilbert-Bernays-Löb provability conditions) *Suppose* **T** *is a recursively enumerable extension of* **PA** *and let* φ *and* ψ *be any sentences, then the following conditions hold*

1. If $\vdash_{\mathbf{T}} \varphi$, *then* $\vdash_{\mathbf{T}} Pr_{\mathbf{T}}(\ulcorner\varphi\urcorner)$

2. $\vdash_{\mathbf{T}} Pr_{\mathbf{T}}(\ulcorner \varphi \to \psi \urcorner) \to (Pr_{\mathbf{T}}(\ulcorner \varphi \urcorner) \to Pr_{\mathbf{T}}(\ulcorner \psi \urcorner))$

3. $\vdash_{\mathbf{T}} Pr_{\mathbf{T}}(\ulcorner \varphi \urcorner) \to Pr_{\mathbf{T}}(\ulcorner Pr_{\mathbf{T}}(\ulcorner \varphi \urcorner) \urcorner)$

Proof. See Boolos et al. (2007) Lemma 18.2. □

In Gödel's second incompleteness theorem we talk about the consistency of a theory $\mathbf{T}$ within $\mathbf{T}$. We do this through the consistency sentence $\mathrm{Con}(\mathbf{T})$ for $\mathbf{T}$, which is usually taken to be

$$\neg \mathrm{Pr}_{\mathbf{T}}(\ulcorner \bot \urcorner)$$

Theorem 2.3.3. (Gödel's second incompleteness theorem) *Suppose* $\mathbf{T}$ *is a consistent and recursively enumerable extension of* $\mathbf{PA}$*, then* $\nvdash_{\mathbf{T}} \mathrm{Con}(\mathbf{T})$.

Proof. See Boolos et al. (2007) Theorem 18.3. □

We will not look at the modern standard proofs of Gödel's theorems in this text, we will however look at alternative proofs that are related to Chaitin's theorem.

2.4 Turing machines

The Turing machine is a type of computing device with an infinite memory. Informally it can be thought of as a box moving over an infinite one dimensional tape. The tape consists of cells and each cell contains either the symbol 0, the symbol 1 or the delimiter symbol #. The box can move over the tape, scanning one cell at a time and after the box has scanned a cell it does the following:

1. Sets the current cell content to 0,1 or #
2. Changes state (in the sense that the new state could be the same as the current)
3. Moves one step left or right on the tape

Exactly which actions that are performed depends on the content of the current cell and which one of the box's finitely many different states the box is currently in.

More formally, for a Turing machine M we have a finite state space $Q = \{q_0, q_1, ..., q_n\}$ with q_0 being the start state, an alphabet $S = \{1, 0, \#\}$ and a partial function $\delta : Q \times S \to Q \times S \times \{\mathrm{L,R}\}$. if δ is undefined for some $(q_i, S_i) \in Q \times S$ we say that T *halts* if it scans S_i while in state q_i. M starts

in state q_0 on a cell c_0 of the tape. The tape continues infinitely in both directions from c_0. The input, consisting of 1's and 0's, is in the, possibly empty, consecutive finite sequence of cells from c_0 to the last cell c_i with $i \geq 0$ that does not contain #. All cells that do not contain the finite input instead contains # when M starts. The output is in the, possibly empty, consecutive finite sequence of cells from c_0 to the last cell c_i with $i \geq 0$ that does not contain # when M halts. One step of computation consists of one cell scanning along with actions 1 to 3 described above. By $\text{halt}(M(n))$ we denote the number of steps before M halts on input n. If M does not halt on input n we say that M is undefined for n and let $\text{halt}(M(n)) = \infty$. We sometimes say that M loops if M does not halt. By $M(n) \downarrow m$ we mean that M halts on input n with output m. By $M(n) \uparrow$ we mean that M does not halt on input n. Following Li and Vitányi (2008) we will throughout this text identify the natural numbers with binary strings under the following bijection

$$(\epsilon, 0), (0, 1), (1, 2), (00, 3), (01, 4), (10, 5), (11, 6), (000, 7), (001, 8), (010, 9), ...$$

i.e. the n:th natural number is identified with the n:th binary string of the lexicographical order displayed above, letting the empty string ϵ count as the first binary string. We let $|n|$ for a natural number n denote the length of the binary string that we identify with n. It can be shown that

$$|n| = \lfloor \log_2(n + 1) \rfloor$$

Under this bijection we can think of Turing machines as partial functions from the natural numbers to the natural numbers. We say a function $f : \mathbb{N} \to \mathbb{N}$ is Turing computable if there exists a Turing machine M such that $f(n) = m$ if and only if $M(n) \downarrow m$ and $f(n)$ is undefined if and only if $M(n) \uparrow$.

We will also use the following notation for strings: 0^e and 1^e are the binary strings consisting of e 0's and e 1's respectively. Unless otherwise explicitly stated we will always mean the string of e 1's when we say 1^e and not the natural number 1 to the power of e, which is identified with the string 0. We will also concatenate strings, by for instance $1^3 0n$ for $n = 6$ we mean the string 111011.

From the above description of Turing machines we see that we can describe a Turing machine M completely by writing down the sequence of tuples that represent the transition function δ of M. Much like we can create a Gödel numbering for formulas we can then create a Gödel numbering for the countable number of sequences of tuples that represent Turing machines. As is done in for instance Rogers (1987) Section 1.8 we now simply take one such listing by lexicographical order and we call this the standard Gödel

numbering of Turing machines. What is important is that finding the Turing machine with the standard Gödel number n is effectively computable in the sense defined below. After explaining what effectively computable means it will be clear that if we use a lexicographical order, then our standard Gödel numbering will satisfy this requirement.

We introduce the notation $\simeq$ for Turing machines as: for two Turing machines M_1 and M_2 we have $M_1 \simeq M_2$ if and only if $M_1(n) \downarrow m \Leftrightarrow M_1(n) \downarrow m$ and $M_1(n) \uparrow \Leftrightarrow M_2(n) \uparrow$ hold for any natural numbers n and m. We will also apply $\simeq$ to recursive functions by an analogue definition.

Definition 2.4.1. A Gödel numbering Φ of Turing machines is *acceptable* if there exist recursive functions f and g such that $\Phi_e \simeq \Psi_{f(e)}$ and $\Psi_e \simeq \Phi_{g(e)}$ for all natural numbers e, where Ψ is the standard Gödel numbering.

One of the most important theorems in the theory of Turing machines is the existence of a universal Turing machine, meaning a Turing machine that can simulate any other Turing machine.

Theorem 2.4.2. *For any acceptable Gödel numbering of Turing machines Φ there exists a Turing machine U such that $U(0^e 1 n) \simeq \Phi_e(n)$ for any natural numbers n and e. We say that U is a universal Turing machine (with respect to Φ).*

Proof. See Li and Vitányi (2008) Example 1.7.4. □

We will sometimes refer to an input p of a universal Turing machine U as a *program*.

We say a function f is *effectively computable* if we can determine $f(n)$ for any natural number n using only explicit mechanic computations following a list of precise instructions, no ingenuity or information except the list of instructions should be required. Note that we allow for f to be a partial function in the sense that our mechanical computations for computing $f(n)$ are not limited to a finite amount of steps, they may continue forever and in this case $f(n)$ is undefined. It is clear that any Turing computable function is also effectively computable. Turing's thesis says that the converse holds, that any effectively computable function is also Turing computable. Since we can not give a precise mathematical definition of what an effectively computable function is we can not prove Turing's thesis. Church's thesis is the corresponding unprovable claim for recursive functions, namely that any effectively computable function is recursive. Recursive functions and Turing machines are two independently developed models of computation and therefore, as stressed in Boolos et al. (2007), the following theorem is a strong indication both claims are correct.

Theorem 2.4.3. *A function is Turing computable if and only if it is recursive.*

Proof. See Boolos et al. (2007) Theorem 8.2. □

Since Theorem 2.4.3 shows that Turing's and Church's thesis are equivalent they are sometimes referred to instead as the Church-Turing thesis. Theorem 2.4.3 also shows that whether we choose to argue using recursive functions or Turing machines is just a matter of personal preference. There also exist many other equivalent models of computation. One direction in the proof of Theorem 2.4.3 in Boolos et al. (2007) is to for any given Turing machine construct a primitive recursive function that expresses the "state of the computation of the Turing machine" after t steps of computation. One then applies minimization to find the least t such that the Turing machine halts. Using Theorem 2.2.11 one can show that this means a Turing machine M is representable by a Σ_1-sentence in $\mathbf{Q}$. Note that this also means that if a Turing machine halts, then this is provable in $\mathbf{Q}$. These facts will be used often in this text.

2.5 Kolmogorov complexity

Using our bijection from the last section between binary strings and natural numbers we make the following definition given a universal Turing machine

Definition 2.5.1. The *Kolmogorov complexity* of a natural number n denoted by $C(n)$ is the length of shortest binary string p such that $U(p) = n$.

Note that our choice of bijection between the natural numbers and binary strings also effects the complexity of a natural number. If a string n satisfies $C(n) \geq |n|$ we say that n is random, thus Kolmogorov complexity captures the intuitive idea that the shortest description of a random string is the string itself, we can find no pattern that might enable a shorter description. Kolmogorov complexity was independently developed in the 1960's by Solomonoff, Kolmogorov and Chaitin. We refer to Li and Vitányi (2008) for a detailed historical overview.

Theorem 2.4.2 shows that two Kolmogorov complexity measures defined with respect to Gödel numberings Φ and Ψ respectively differ only by at most a constant that is independent of the strings measured, this is because U_Φ can run U_Ψ at constant cost and vice versa. Note that it also follows from Theorem 2.4.2 that

$$C(n) \leq |n| + e$$

where e is a constant independent of n. This is because we can create a Turing machine Φ_e, which given n as input outputs n.

It is important to note that one can show by arguments similar to those in our discussion in the end of the previous section that in fact $C(n) \leq w$ is equivalent to a Σ_1-statement. Roughly this is because we saw that for our universal Turing machine U we could express that U halts on input p with output n by a Σ_1-sentence and thus we can also express that U halts on some input p of length less than or equal to w with output n by a Σ_1-sentence. From this we also see that $C(n) > w$ is equivalent to a Π_1-statement. We state these facts as propositions to emphasize their importance.

Proposition 2.5.2. *$C(n) \leq w$ is equivalent to a Σ_1-statement.*

Proposition 2.5.3. *$C(n) > w$ is equivalent to a Π_1-statement.*

3 Chaitin's incompleteness theorem

3.1 Main result

Chaitin has given many versions[1] of his incompleteness theorem, here we roughly follow a version given in Chaitin (1992a). We tacitly assume **T**, unless otherwise stated, is an extension of **Q** for the remainder of this section. Also note that if we have a finite amount of axioms, then we can always create one single equivalent axiom through conjunction.

Theorem 3.1.1. (Chaitin's incompleteness theorem) *Assume the theory* **T** *is finitely axiomatizable by an axiom A and also assume all statements of the form "$C(n) > w$" that are provable from A are true, then there exists a constant c such that* **T** *contains no theorems of the form "$C(n) > C(\ulcorner A \urcorner) + c$".*

Proof. We can create a Turing machine M that takes as input a concatenation $0^k 1 \ulcorner A \urcorner$. M now, with the use of A, begins listing **T**. This could for instance be done by in increasing order checking natural numbers m to see if m is the Gödel number of a proof in which the only undischarged assumption is A. If this is the case, then the conclusion is a theorem of **T**. If **T** contains a theorem of the form "$C(n) > |\ulcorner A \urcorner| + 2k$", then M stops when finding the first such theorem and outputs n.

Now assume towards a contradiction that M halts with output n. Then for some constant c_M we have $U(c_M 0^k 1 \ulcorner A \urcorner) = M(0^k 1 \ulcorner A \urcorner) = n$ and therefore

$$C(n) \leq |c_M| + |\ulcorner A \urcorner| + k + 1$$

But by our soundness assumption $|\ulcorner A \urcorner| + 2k < C(n)$ also holds so we have

$$|\ulcorner A \urcorner| + 2k \leq |c_M| + |\ulcorner A \urcorner| + k + 1$$

This is a contradiction for sufficiently large k and thus **T** can contain no theorem of the form "$C(n) > C(\ulcorner A \urcorner) + c$" for $c = 2k$ and k sufficiently large. □

Remarks:

(i) It is important to note that while there are 2^n binary strings of length n, there are only $2^n - 1$ binary strings of length strictly less than n. This shows that there exists random strings of any length. Thus Theorem 3.1.1 proves that there exists true statements that are not provable in the system. We will discuss this further in Section 3.4.

[1] See for instance Chaitin (1974) or Chaitin (1992b).

(ii) Chaitin (1992a) chooses to prove Theorem 3.1.1 for a prefix Kolmogorov complexity measure (see Section 4.1), however such implementation details will not matter for the discussion that follows in Section 3.2.

(iii) In Theorem 3.1.1 we assume that our theory is finitely axiomatizable, this is quite a limitation. However we could for a stronger theorem assume just that our theory is recursively enumerable, then there exists a Turing machine M', which given the input 0^k1 searches the specific theory $\mathbf{T}$ for the first theorem of the form described in the proof. In this case we have "hard coded" the choice of theory into our Turing machine. For any recursively enumerable theory we can create such a Turing machine. When proving the theorem in this manner an alternative formulation of the theorem would be "there exists a constant c such that T contains no theorems of the form "$C(n) > c$", this is indeed a common way to state the theorem, see for instance Calude and Jürgensen (2005). This alternative proof hints that the connection between the complexity of the axioms and the constant c in our alternative formulation might be weak, this is something we will discuss in the following section.

3.2 Discussion

What has been controversial in the past is not the validity of Chaitin's theorem, but how to interpret it. In Theorem 3.1.1 the Kolmogorov complexity of the Gödel number of the axiom of the system appears as a rough (if not to say very rough as the constant c of the proof may be a very large number) upper bound for how great we can prove the Kolmogorov complexity of any given string n to be. In what follows we will say "the complexity of the axioms" instead of "the Kolmogorov complexity of the Gödel number of the axiom" as the detail how we identify a sentence with a binary string is usually not explicitly discussed in the papers we will cite.

In Chaitin (1992b) Chaitin explains that what he the calls the spirit of Theorem 3.1.1 has been expressed as:

> A set of axioms of complexity N cannot yield a theorem of complexity substantially greater than N

He writes that this description originated with the discussion in Chaitin (1974):

> We shall see that there are circumstances in which one only gets out of a set of axioms what one puts in, and in which it is possible to reason in the following manner. If a set of theorems constitutes

> t bits of information, and a set of axioms contains less than t bits of information, then it is impossible to deduce these theorems from these axioms.

In Chaitin (1982) Chaitin also expresses ambitions of a similar kind:

> In contrast I would like to measure the power of a set of axioms and rules of inference. I would like to [be] able to say that if one has ten pounds of axioms and a twenty-pound theorem, then that theorem cannot be derived from those axioms.

In van Lambalgen (1989) the author explains that the purpose of his paper is to show that Chaitin's mathematics does not support his philosophical claims. To illustrate that one can prove $\nvdash_{\mathbf{T}} C(n) > c$ for a constant c without there being any explicit connection between c and the Kolmogorov complexity of the axioms of $\mathbf{T}$ van Lambalgen[2] sketches the following proof.

Theorem 3.2.1. *Suppose* $\mathbf{T}$ *is as in Theorem 3.1.1, then there exists a constant* c *such* $\nvdash_{\mathbf{T}} C(n) > c$ *for any string* n*.*

Proof. We create a Turing machine Φ_e as follows: By listing[3] the theorems of $\mathbf{T}$, Φ_e on input m finds the first theorem in $\mathbf{T}$ of the form "$\Phi_m(m) \not\downarrow n$", when such a theorem is found Φ_e halts with output n. We now see that $\Phi_e(e)$ is undefined, i.e. $\Phi_e(e) \uparrow$, since otherwise we would be able to prove both $\Phi_e(e) \downarrow n$ (if $\Phi_e(e) \downarrow n$ is true, then it is a true Σ_1-sentence) and $\Phi_e(e) \not\downarrow n$ in $\mathbf{T}$, which contradicts our soundness assumption about $\mathbf{T}$. This shows that $\nvdash_{\mathbf{T}} \Phi_e(e) \not\downarrow n$ for any n, so for any n we see that $\mathbf{T} + \Phi_e(e) \downarrow n$ is consistent. But from $\mathbf{T} + \Phi_e(e) \downarrow n$ we can prove $U(0^e 1 e) \downarrow n$, from which we can prove $C(n) \leq e + 1 + |e|$, and therefore $C(n) > e + 1 + |e|$ can not be provable in $\mathbf{T}$ for any string n. □

Note in the proof above that we could have weakened our soundness assumption on $\mathbf{T}$ to a consistency assumption, as we shall see in the proof of Theorem 3.4.1 this is actually the case with Chaitin's proof as well when we assume $\mathbf{T}$ is an extension of $\mathbf{Q}$. Since van Lambalgen can prove Chaitin's theorem without referring to the complexity of the axiom he questions whether there in fact is any interesting connection between the complexity of the axioms of $\mathbf{T}$ and the minimal constant $c_{\mathbf{T}}$ such that $\nvdash_{\mathbf{T}} C(n) > c_{\mathbf{T}}$ for

[2] van Lambalgen credits Albert Visser and Dick de Jongh for the idea.

[3] We can use a method such as that described in the proof of Theorem 3.1.1 to list the theorems, as previously we could just as well hard code our choice of theory in the Turing machine, meaning we do not need to use the Gödel number of a finite axiomatization as input.

any string n. As van Lambalgen points out we for instance have no idea if $c_{\mathbf{ZFC}} > c_{\mathbf{PA}}$, where **ZFC** is Zermelo-Fraenkel set theory with the axiom of choice which is strictly stronger than **PA**. van Lambalgen also comments on the basic fact that Theorem 3.1.1 speaks about the unprovability of certain formulas asserting that the Kolmogorov complexity of a string is greater than some constant, the theorem does not explicitly make a claim about general formulas being unprovable due to the formulas themselves having too high Kolmogorov complexity.

In Chaitin (1992b) Chaitin comments on what he calls the "heuristic principle" described in the quote above from Chaitin (1974). He notes that any set of axioms from which we can derive an infinite set of theorems must violate this principle since only finitely many strings can have complexity below any constant c. As a simple example he mentions the infinitely many trivial theorems of the form $n = n$ derivable from $\forall x(x = x)$. Chaitin suggests rephrasing the principle as:

> A set of axioms of complexity N cannot yield a theorem that asserts that a specific object is of complexity substantially greater than N.

and he goes on to explain:

> It was removing the words "asserts that a specific object" that yielded the slightly overly-simplified version of the principle that we discussed above

Chaitin's rephrasing changes the meaning of the principle, it is now a trivial corollary of Theorem 3.1.1. Chaitin is quoted in Svozil (1993) on a similar rephrasing:

> A more technical statement is "You can't prove a 10 pound theorem from a 5 pound set of axioms AND KNOW THAT YOU HAVE DONE IT."(I.e., know that you've proven a theorem with a particular complexity that substantially exceeds that of the axioms.) Restated in this slightly more careful fashion, it is now obvious that my assertion is an immediate corollary of my basic theorem that one can prove "$H(s) > n$"[4] only if $n < H(A)+O(1)$.

As Fallis (1996) comments, the claim "You can't prove a 10 pound theorem from a 5 pound set of axioms AND KNOW THAT YOU HAVE DONE IT" is true, but also slightly misleading since Theorem 3.1.1 speaks about a general

[4]By $H(x)$ Chaitin means prefix Kolmogorov complexity, see Remark (ii) after Theorem 3.1.1.

string n, whether that string is identified with a formula which happens to be a theorem of the theory or not is irrelevant.

Raatikainen (1998) extends van Lambalgen's criticism by showing that the minimal constant c such that $\mathbf{T}$ proves no sentences of the form $C(n) > c$ for any string n can be made close to 0 regardless of the strength of $\mathbf{T}$. There is a small mistake[5] in Raatikainen's paper and the argument is extended in Ibuka et al. (2011) so we follow the proof given there instead. It should be noted that Ibuka et al. (2011) works with Turing machines that accepts no input, in that case we define the Kolmogorov complexity of a string n as the least e such that $\Phi_e \downarrow n$ with Φ being our Gödel numbering of Turing machines. Here we have however adapted the result to Turing machines that do accept input (our machine $A_{\mathbf{S},\mathbf{T}}$ soon to be defined just happens to ignore any input) and therefore everything is as before.

Given an acceptable Gödel numbering Φ of Turing machines and a Kolmogorov complexity $C(\cdot)$ with respect to Φ we make the following definition for any recursively enumerable and consistent theory $\mathbf{T}$

Definition 3.2.2. Define $c_{\mathbf{T}}^{\Phi}$ as the least c such that $\mathbf{T}$ proves no statements of the form $C(n) > c$ for any string n.

We have here adapted the notation of Ibuka et al. (2011) to index $c_{\mathbf{T}}^{\Phi}$ by the Gödel numbering Φ used in order to make things a little clearer. Note that it would not be wrong for our purposes in this text to index the Kolmogorov complexity $C(.)$ by Φ as well.

Lemma 3.2.3. *Suppose* $\mathbf{S}$ *and* $\mathbf{T}$ *are two recursively enumerable and consistent theories, then there exists a Turing machine* $A_{\mathbf{S},\mathbf{T}}$ *such that for any* n

$$\mathbf{S} \nvdash A_{\mathbf{S},\mathbf{T}} \not\downarrow n$$

and

$$\mathbf{T} \nvdash A_{\mathbf{S},\mathbf{T}} \not\downarrow n$$

Proof. (Proof idea) Let $A_{\mathbf{S},\mathbf{T}}$ be a Turing machine that ignores any input given to it and instead runs $\Phi_e(e)$, where Φ_e is the Turing machine from the proof of Theorem 3.2.1. Only difference from Theorem 3.2.1 is that we alter Φ_e so that it in parallel list the theorems of two theories instead of just one. We refer to Ibuka et al. (2011) for an alternative proof. □

We followed Ibuka et al. (2011) and stated Lemma 3.2.3 for two theories since we will return to it in Section 4.2 and there we will discuss two theories instead of just one. It is clear from our proof idea how one would define a similar machine $A_{\mathbf{T}}$ for just one theory.

[5] http://mathforum.org/kb/plaintext.jspa?messageID=570656

Proposition 3.2.4. *There exists an acceptable Gödel numbering* Φ *such that* $c_{\mathbf{T}}^{\Phi} \leq 1$.

Proof. Put $\Phi_0 = A_{\mathbf{T}}$. Since we somewhere in the standard Gödel numbering Φ find $A_{\mathbf{T}}$, it is clear from Definition 2.4.1 that we can define an acceptable Gödel numbering by interchanging the index of the first Turing machine in Φ with the index of $A_{\mathbf{T}}$ in Φ. Since $\mathbf{T} \nvdash A_{\mathbf{T}} \not\downarrow n$ for any string n, $\mathbf{T}$ can not prove that $U(\epsilon 0 \epsilon)$[6] does not halt with output n for any string n and thus it can not prove $C(n) > 1$ for any n. Note that by a small adjustment in how our universal machine takes input (we could for instance let U run Φ_0 with empty input when U is given empty input) we could just as well have proved $c_{\mathbf{T}}^{\Phi} \leq 0$. □

By Proposition 3.2.4 we see that if we are allowed to manipulate the Gödel numbering used by our universal Turing machine, then the minimal constant c such that $\mathbf{T}$ proves no sentences of the form $C(n) > c$ for any string n can be made less than or equal to 1 regardless of the strength or complexity of $\mathbf{T}$ or it's axioms. As Raatikainen (1998) comments, this shows that van Lambalgen's question how for instance $c_{\mathbf{ZFC}}$ relates to $c_{\mathbf{PA}}$ is not well defined. One could argue that we can make it well defined by choosing a fixed universal Turing machine U and then compare the constants with respect to the complexity induced by U. One could also argue that the trick used by Ibuka et al. (2011) and Raatikainen (1998) results in an acceptable but very artificial Gödel numbering of Turing machines. Raatikainen (1998) argues in response to this that it would be very hard to settle on one "natural" Gödel numbering of Turing machines (we would also need to decide on implementation details of the universal Turing machine U), let alone even which computational model to use among the many known equivalent ones (Turing machines, recursive functions, URM etc). Raatikainen (1998) also argues that the goal here is not to create odd Gödel numberings, but to illustrate that the constant $c_{\mathbf{T}}$ is effected by more or less accidental coding choices. Let us also just note that it was the false principle

> A set of axioms of complexity N cannot yield a theorem of complexity substantially greater than N

that sparked our interest in these constants to begin with.

[6]Recall Definition 2.4.2 to understand why the input string is $\epsilon 0 \epsilon$, by that definition running Φ_0 with empty input on our universal Turing machine is done by giving the input string $\epsilon 0 \epsilon$.

3.3 Summary of discussion

It is clear that any principle which claims

> A set of axioms of complexity N cannot yield a theorem of complexity substantially greater than N

is trivially false, at least under the complexity measure we discussed above. Chaitin's result on the other hand can be informally stated as

> A set of axioms of complexity N cannot yield a theorem that asserts that a specific object is of complexity substantially greater than N

this is a fundamentally different statement and, depending on how interested one is in proving statements about complexity of strings, a less useful one. Had the former principle been true it would have indeed been an interesting addition to our knowledge about sources of incompleteness in formal systems. As Chaitin himself notes in Chaitin (1974), what he has done in his incompleteness theorem is that he has successfully formalised Berry's paradox in a wide range of formal systems. Berry's paradox[7] is that with "the smallest natural number not describable by fewer than a 100 words" we have just described the number in less than a 100 words. To escape the paradox our consistent formal systems must be unable to prove that a number is "not describable by fewer than a 100 words". Specifying that we look for "the smallest natural number" in Berry's paradox is of course just a way of making a unique selection among candidate numbers which are not describable by fewer than a 100 words, in our proof of Theorem 3.1.1 we do this unique selection by extracting the number from the first statement of the correct form that we find as we list the theorems of our recursively enumerable theory $\mathbf{T}$. What Chaitin's theorem shows is that for some sufficiently large constant c we find no theorems of the form "$C(n) > c$" in $\mathbf{T}$, it makes no claim about what the minimal such c is. van Lambalgen gives a fundamentally different approach to proving the same theorem, he instead uses the well known fact that we can not determine if every Turing machine halts and notes that this undecidability transfers to a complexity measure which use the Turing machines for measurement. As illustrated by van Lambalgen one way to find

[7] At http://en.wikipedia.org/wiki/Berry_paradox we find:

> Bertrand Russell, the first to discuss the paradox in print, attributed it to G. G. Berry (1867-1928), a junior librarian at Oxford's Bodleian library, who had suggested the more limited paradox arising from the expression "the first undefinable ordinal".

the index of such an unpredictable Turing machine is that if we are given a recursively enumerable theory such as $\mathbf{T}$ we can always create a Turing machine that searches for a contradictory proof about it's own output among the theorems of the theory. Raatikainen then notes that by manipulating the Gödel numbering used by our complexity measure we can make the constant c arbitrarily small, regardless of any property of $\mathbf{T}$. This illustrates that we can not give too much meaning to specific results obtained under some certain universal Turing machine. To give concrete examples: which exact strings that are random will vary under different universal Turing machines, but intuitive ideas such as "the least string of length of L is not random" will be true for any universal Turing machine U_Φ (U_Φ denotes a universal Turing machine with respect to an acceptable Gödel numbering Φ) for sufficiently (depending on U_Φ) large L. This is simply because for any Φ there exists an e such that Φ_e is the Turing machine which given L as input outputs the least string of length L and as mentioned in Section 2.4 $|L| = \lfloor \log_2(L+1) \rfloor$.

3.4 Chaitin's theorem in relation to Gödel's theorems

As we have seen both Chaitin's and Gödel's theorems make use of paradoxes, in the case of Chaitin's result it is Berry's paradox and in Gödel's case it is the liar paradox. van Lambalgen (1989) remarks that while Chaitin's theorem, much like Gödel's first theorem, proves the existence of unprovable sentences in a formal system, it does not give us an explicit construction of such a sentence the way Gödel's result does. Of course, if we allow ourselves to manipulate the Gödel numbering used by our fixed universal Turing machine, we can as Raatikainen (1998) points out make $\Phi_0 \not\downarrow n$ unprovable for any n and thereby, under some assumptions about the operation of our universal Turing machine, make for instance $C(0) > 1$ unprovable. Chaitin's theorem does however not in general give us an explicit example of an undecidable sentence.

It should be noted that Theorem 3.1.1 (Chaitin's theorem) makes a soundness assumption, we assume that all statements of the form "$K(n) > w$" provable in our formal system are true, Theorem 2.3.1 (Gödel's first theorem) on the other hand only make the weaker assumption of consistency. It is under this soundness assumption Chaitin's result is usually proved[8]. We could however make do with just the assumptions of Theorem 2.3.1 when proving Theorem 3.1.1, namely that $\mathbf{T}$ is a recursively enumerable and consistent extension of $\mathbf{Q}$, this is the way it is done in for instance Kritchman and Raz (2010).

[8]This is discussed by Carl Mummert at http://math.stackexchange.com/a/1002601

Theorem 3.4.1. *Suppose* $\mathbf{T}$ *is a consistent and recursively enumerable extension of* $\mathbf{Q}$*, then there exists a true sentence that is not provable* $\mathbf{T}$

Proof. First we need to show that the soundness assumption of Chaitin's theorem (Theorem 3.1.1) can be replaced by a consistency assumption. To see why this works, assume towards a contradiction that $\mathbf{T}$ proves a false statement of the form "$C(n) > w$". Since this means that the Σ_1-formula $C(n) \leq w$ is true we see that $\mathbf{T}$ also proves $C(n) \leq w$, which means that $\mathbf{T}$ is inconsistent.

Now, as noted in Section 2.5, since there exists random strings of any length it follows from Chaitin's theorem that there exists an c such that true sentences of the form $C(n) > c$ are not provable in $\mathbf{T}$. Note here that we have not proved that $\mathbf{T}$ is incomplete. Since $\mathbf{T}$ is an extension of $\mathbf{Q}$ we can not appeal to Theorem 2.1.3 and say that a Σ_1-sentence $C(n) \leq c$ is provable in $\mathbf{T}$ only if $C(n) \leq c$ is true. We could however add an assumption that $\mathbf{T}$ is ω-consistent, from this it follows that $C(n) \leq c$ is provable in $\mathbf{T}$ only if $C(n) \leq c$ is true and then we have in fact that $\mathbf{T}$ is incomplete.

Finally note that any extension of $\mathbf{Q}$ as noted in Section 2.4 is strong enough to formalize Turing machines and therefore also strong enough to formalize Kolmogorov complexity. □

We saw above that Chaitin's result is as strong as the original version of Gödel's first incompleteness theorem in the sense that it shows under an assumption of ω-consistency that recursively enumerable theories whose arithmetic fragment is as strong as $\mathbf{Q}$ are incomplete. If we only want to use an assumption of consistency we can prove that there exists true unprovable sentences. We can also note that roughly the same amount of theory is needed regardless if we prove incompleteness through Gödel's or Chaitin's approach, usually that theory needed is recursive functions and a Gödel numbering of derivations.

One can now ask if Chaitin's result implies Gödel's second theorem. Kritchman and Raz (2010) showed that this in fact is the case.

Theorem 3.4.2. *Suppose* $\mathbf{T}$ *is a consistent and recursively enumerable extension of* $\mathbf{PA}$*, then* $\nvdash_{\mathbf{T}} \mathrm{Con}(\mathbf{T})$.

Proof. (Informal proof idea) Let c be the constant such that $\mathbf{T}$ contains no theorem of the form "$C(n) > c$" for any n, this is the c guaranteed to exist by Chaitin's theorem. As noted earlier we see by a simple counting argument that there are $2^{c+1} - 1$ binary strings of length at most c. We now let m be the number of natural numbers n such that $0 \leq n \leq 2^{c+1} - 1$ and $C(n) > c$, we also note that $1 \leq m \leq 2^{c+1}$ follows from our counting argument above and we assume this conclusion is provable in $\mathbf{T}$.

Now assume $m = 1$. Since the statement "$C(n) \leq c$" is a true Σ_1-statement for all except one n with $0 \leq n \leq 2^{c+1} - 1$ we can prove $C(n) \leq c$ for all except one n in $\mathbf{T}$. But we can also prove $m \geq 1$ in $\mathbf{T}$ and thus we can prove $C(n) > c$ for one concrete n. By Chaitin's theorem this contradicts the consistency of $\mathbf{T}$. Hence we must have $m \geq 2$ and what is done then in the formal proof is to show that by assuming the consistency sentence of $\mathbf{T}$ in $\mathbf{T}$ we can prove $m \geq 2$ in $\mathbf{T}$. Thus we can repeat the steps above to prove $m \geq 3$ in $\mathbf{T}$ and so on, this eventually yields a contradiction since we could also prove $m \leq 2^{c+1}$ in $\mathbf{T}$. We omit the formal proof which is an exercise in using the Hilbert-Bernays-Löb provability conditions and the following formal version of Chaitin's theorem from Kikuchi (1997)

$$\vdash_{\mathbf{T}} \mathrm{Con}(\mathbf{T}) \rightarrow \forall x(\neg \mathrm{Pr}_{\mathbf{T}}(\ulcorner C(x) > c \urcorner))$$

□

Finally let us also mention that there are other alternative proofs of Gödel's theorems which also utilize Berry's paradox. Extending work in Boolos (1989), Kikuchi et al. (2012) proves Gödel's two theorems for recursively enumerable extensions of **PA**. To do this Kikuchi et al. (2012) defines the notion of formulas *naming* natural numbers. One can then talk about the least natural number not named by any formula of at most a certain length. It is interesting to note that Kikuchi et al. (2012) also require an assumption of ω-consistency to prove incompleteness. Under an assumption of consistency Kikuchi et al. (2012) can prove the existence of a true unprovable sentence, so we see that this is a similar situation to that we had when using Chaitin's result to prove incompleteness. Kikuchi et al. (2012) tries to make the comparison between the two approaches more concrete by defining the concept of a Kolmogorov complexity based on provability to show that the approach in Kikuchi et al. (2012) to proving Gödel's second incompleteness theorem is a special case of Chaitin's result that we can not prove $C(n) > c$ for arbitrarily large c. We refer to Kikuchi et al. (2012) for a full discussion with the necessary definitions.

4 Recent developments

In this section we look at some recent results that have a connection to Chaitin's theorem.

4.1 Calude and Jürgensen (2005)

Let us take a look at Calude and Jürgensen (2005). The main purpose of the paper is to prove that for a certain complexity measure the previously discussed principle

> The theorems of a finitely specified[9] theory can not be significantly more complex than the theory itself

holds, Calude and Jürgensen (2005) refer to this as Chaitin's heuristic principle. We will mainly follow Grenet (2010) here as one of it's purposes is to prove the same thing, but with some corrections. To start with we need to define prefix Kolmogorov complexity. Let an alphabet X_i of i symbols be given and let X_i^* denote the set of all finite strings on this alphabet.

Definition 4.1.1. Suppose x, z are strings in X_i^*, then we say that x is a prefix of z if $z = xy$ for some non empty y in X_i^*.

Definition 4.1.2. We say a subset S of X_i^* is *prefix free* if there are no two elements x and y in S such that x is a prefix of y.

Definition 4.1.3. We say a Turing machine M on an alphabet X_i is *prefix free* if the set $S = \{x \in X_i^* : M(x)\downarrow\}$ is prefix free.

Definition 4.1.4. Given a universal prefix free Turing machine U with domain $S \subset X_i^*$, we define the *prefix Kolmogorov complexity* $K_i(.)$ with respect to U for x in X_i^* as

$$K_i(x) = \min_{\substack{u \in S \\ U(u)\downarrow x}} |u|_i$$

$|x|_i$ is simply the length of a string x on an alphabet of i symbols, we add an index i as this notation is used in Grenet (2010). For a proof of the existence of the above mentioned prefix free universal Turing machine U and more information about prefix Kolmogorov complexity we refer to Li and Vitányi (2008).

For our regular Kolmogorov complexity it is the case that there exists a constant e such that for all n we have $C(n) \leq |n| + e$. It is very important

[9] By this is meant that the axioms of the theory are recursively enumerable.

for what follows to know that this no longer holds for prefix Kolmogorov complexity.

Grenet (2010) now makes the following definition

Definition 4.1.5. For any string x in X_i^* we define the complexity measure

$$\delta_i(x) = K_i(x) - |x|_i$$

Actually another complexity measure $\delta_g(x)$, for any Gödel numbering g of X_i^*, is also introduced. The authors of Calude and Jürgensen (2005) and Grenet (2010) state that the second complexity measure is introduced in order to prove that the results obtained does not depend on any specific way of encoding theorems. However, for our discussion here it will be enough to work with $\delta_i(x)$. Note that this is also how many of the arguments in Grenet (2010) go, they are carried out for $\delta_i(x)$ and then translated to an arbitrary complexity measure $\delta_g(x)$. The key to this translation is Theorem 2 of Grenet (2010), or more precisely the remark after Theorem 2 that for any g there exists a constant c such that

$$|\delta_g(x) - \lceil \log_2 i \rceil \cdot \delta_i(x)| < c \tag{1}$$

for all x in X_i^*. Also note that any such Gödel numbering g has nothing to do with the Gödel numbering used by our universal Turing machine by which we measure K_i.

The main (in the part which is a correction of Calude and Jürgensen (2005)) result of Grenet (2010) is stated as

Theorem 4.1.6. *Suppose* $\mathbf{T}$ *is a finitely specified and arithmetically sound*[10] *theory expressed in the alphabet* X_i*. Then there exists a constant* N *such that for any theorem* x *in* $\mathbf{T}$ *we have*

$$\delta_i(x) < N$$

Proof. Follows directly from Lemma 4.1.7 below. □

Note that this theorem (Theorem 3 of Grenet (2010)) is expressed for $\delta_g(x)$ for an arbitrary Gödel numbering g in Grenet (2010), however as previously noted an argument about δ_i is at the core of the proof and then (1) is used to transfer the result to $\delta_g(x)$. It is not clear how the assumption that the theory is arithmetically sound is used in the proof in Grenet (2010). The following lemma is the upper bound of Lemma 1 in Grenet (2010).

[10] By this is meant that any arithmetical sentence proved by $\mathbf{T}$ is true.

Lemma 4.1.7. *Suppose* $\mathbf{T}$ *is a finitely specified and arithmetically sound theory expressed in the alphabet* X_i*. Then there exists a constant* N *such that for any theorem* x *of* $\mathbf{T}$ *we have*

$$K_i(x) < |x|_i + N$$

Proof. This is claimed without proof in Calude and Jürgensen (2005) by appealing to syntactical constraints. The argument in Grenet (2010) is the following: note that a theorem is a special case of a well formed formula expressed in the alphabet X_i, as is noted in Grenet (2010) this also means that the lemma applies to any well formed formula. We create the following prefix free Turing machine M. Given an input xy, where y is a fixed non well formed formula expressed in X_i (Grenet (2010) suggests using for instance "++") and x is a well formed formula in X_i^*, M halts with output x. For input of any other form M does not halt. Note that it is adding y to the end of x which makes the domain of M prefix free, this is because y can not be contained in any well formed formula. Since we can run M at a constant cost c_M on our fixed universal prefix free Turing machine U we have proved that there exists a constant N (independent of x) such that

$$K_i(x) < |x|_i + N$$

□

Grenet (2010) says Theorem 4.1.6 is the formal version of Chaitin's principle

> The theorems of a finitely specified theory can not be significantly more complex than the theory itself

First of all I think it is a bit misleading since we have not defined what the complexity of a theory means, note that finitely specified does not imply finitely axiomatizable so it is not clear how we would determine the complexity of our theory. Even if the complexity of our theory was well defined, we would still not have used it in any way to determine the constant N of Lemma 4.1.7. In fact our remark in Lemma 4.1.7 shows that the following alternative principle holds for the complexity measure δ_i

> The well formed formulas of a finitely specified theory can not be significantly more complex than the theory itself

From an incompleteness and provability perspective a complexity measure that satisfies Chaitin's principle as a result of satisfying the alternative principle is not interesting. To stress our point: our theory $\mathbf{T}$ can not prove

sentences of arbitrarily high δ_i-complexity simply because all well formed formulas have complexity below a finite constant N by Lemma 4.1.7. It should be noted that in the original paper Calude and Jürgensen (2005) on p. 9 claims (the claim is for $\delta_g(x)$ since they have used (1) to translate it, however by this same translation the claim here is equivalent) that there exist true sentences x expressible in X_i which have $\delta_i(x) > N$, where N is the constant from Theorem 4.1.6. Calude and Jürgensen (2005) does not justify why these sentences have sufficiently large δ_i-complexity, the claim also appears to be false since these true sentences of course are well formed formulas and Calude and Jürgensen (2005) at the bottom of p. 6 explicitly (just as noted by Grenet (2010)) says that Lemma 4.1.7 is true for any x in $\mathbf{T}$ just because x is a well formed formula. There is no mention of these sentences in Grenet (2010). Another claim in Calude and Jürgensen (2005) which supports that there in fact are true well formed formulas of arbitrarily large δ_i-complexity is the following limits (L is any integer)

$$\lim_{n\to\infty} i^{-n} \cdot |\{x \in X_i^* : \quad |x|_i = n,\, \delta_i(x) \leq L\}| = 0 \tag{2}$$

$$\lim_{n\to\infty} i^{-n} \cdot |\{x \in X_i^* : \quad |x|_i = n,\, x \text{ is true}\}| > 0 \tag{3}$$

Let us first note that these limits contradict each other by Lemma 4.1.7 since for sufficiently large L the set of true sentences is a subset of the set of well formed formulas which have bounded δ_i-complexity by Lemma 4.1.7. That being said the argument proving the first limit can be found in the proof of Proposition 5.1 of Calude and Jürgensen (2005). The argument proving the second limit is found in Theorem 5.2 of Calude and Jürgensen (2005) and is the following: Consider true sentences of the form "$K_i(x) > 1$" with x in X_i^*, then there exists an M such that for all x in X_i^* with $|x|_i > M$ it is true that $K_i(x) > 1$. Thus for $n > M + c$, where c is the length of encoding $K_i(.) > 1$ in X_i, we have

$$i^{-n} \cdot |\{x \in X_i^* : \quad |x|_i = n,\, x \text{ is true}\}| \geq i^{-n} \cdot |\{x \in X_i^* : \quad |x|_i = n - c\}| = i^{-c}$$

The problem is that if we accept statements of the form "$K_i(x) > 1$" for any x in X_i^* as well formed formulas, then the argument in the proof Lemma 4.1.7 no longer holds. In the proof we used a fixed ill formed formula y as delimiter to ensure the domain of M is prefix free. However if we can put any string x in the parenthesis of the expression "$K_i(.) > 1$", then with the knowledge how $K_i(.) > 1$ is expressed in X_i (i.e. we need the exact expression for $K_i(.)$) we can for any z in X_i^* choose x so that qy is a prefix of py where q is $K_i(z) > 1$, p is $K_i(x) > 1$ and y is our fixed ill formed formula. The rough idea would be to let x be the string z)>1y (how this

would be done exactly depends once again on how $K_i(.) > 1$ is expressed in X_i). Thus M is no longer prefix free and our argument no longer works. Another way to put this is: we can not put any arbitrary string x from X_i^* in the parenthesis of the expression "$K_i(.) > 1$" and still necessarily have a well formed formula. This explains how the limits (2) and (3) can contradict each other, limit (2) only counts well formed formulas while limit (3) also counts non well formed formulas as true sentences. One solution to this problem would be to express x using a proper subset of the alphabet X_i, this restriction corresponds to the ordinary situation in logic where we do not allow terms to contain for instance the symbol $\wedge$, however note that then the argument above for proving inequality (3) no longer holds. It is inequality (3) which justifies the claim that the probability that a sentence of length n is true is strictly positive[11] in Theorem 5.2 of Calude and Jürgensen (2005). It should also be noted that I am not the first to ask questions about Calude and Jürgensen (2005), in the comments of http://mathoverflow.net/a/7902 we find similar concerns raised.

We have seen that δ_i satisfies Chaitin's principle as a result of satisfying the alternative principle

> The well formed formulas of a recursively enumerable theory can not be significantly more complex than the theory itself

thus unfortunately making it an uninteresting example of a complexity measure satisfying Chaitin's principle. This makes the, in the introduction and Section 4 of Grenet (2010), described use of δ_i as a model for finding other complexity measures which satisfy Chaitin's principle questionable, since it appears that additional constraints are desirable.

4.2 Ibuka et al. (2011)

We now look at some more details of the extension of the ideas in Raatikainen (1998) given in Ibuka et al. (2011). In this paper, as previously mentioned, we work in a setting of Turing machines that accept no input. Note that the definition given below for $r_{\mathbf{T}}$ does not make sense as it stands for a Gödel numbering of Turing machines that do accept input. The results we state for $c_{\mathbf{T}}$ on the other hand, can in fact be translated in an obvious manner to

[11]This is in fact true by the simple argument illustrated in http://mathoverflow.net/a/7902 when we instead look at the ratio between true statements and well formed formulas (not between true statements and all strings expressible in the alphabet as in Calude and Jürgensen (2005)), however as shown there we then see that the ratio between provable statements and well formed formulas is also strictly positive, thus we do not get inequalities such as (2) and (3).

a Gödel numbering of Turing machines that accept input, to see this note that even if we work with a Gödel numbering of Turing machines that accept input, we are of course always free to create Turing machines that ignore any input given to them.

Given any Turing machine M we make the following definitions

Definition 4.2.1. Define M^n as a Turing machine such that

$$M \uparrow \rightarrow M^n \uparrow$$

and

$$M \downarrow \rightarrow M^n \downarrow n$$

We see that M^n is a Turing machine that loops if M loops and halts with output n if M halts, regardless of the output of M.

Definition 4.2.2. Define $M^{(n)}$ as a Turing machine such that for any $m \neq n$

$$M \uparrow \rightarrow M^{(n)} \uparrow,$$

$$M \downarrow n \rightarrow M^{(n)} \downarrow n+1$$

and

$$M \downarrow m \rightarrow M^{(n)} \downarrow m, \qquad m \neq n$$

We see here that $M^{(n)}$ is a Turing machine that loops if M loops and halts with output $n+1$ if M halts with output n, if M on the other hand halts with output m for $m \neq n$, then $M^{(n)}$ halts with unaltered output m. Also note that $M^{(n)} \not\downarrow n$ for any Turing machine M and any number n, in what follows we will tacitly assume such simple facts like this and for instance $M \uparrow \leftrightarrow M^n \uparrow$ are provable in **PA**, see Ibuka et al. (2011) for a detailed argument.

We will take the following lemma from Ibuka et al. (2011) as an assumption, a way to prove it is to develop theory on the arithmetical hierarchy and Kleene's T predicate.

Lemma 4.2.3. *Assume τ is a Π_1 sentence, then there exists a Turing machine D_τ such that* $\mathbf{PA} \vdash (\tau \leftrightarrow D_\tau \uparrow) \wedge (\neg\tau \leftrightarrow D_\tau \downarrow 0)$.

Given theories **S** and **T** extending **PA** and an acceptable Gödel numbering Φ of Turing machines that accept no input we define the Kolmogorov complexity of a string n as the least e such that $\Phi_e \downarrow n$. We now remind the reader of the following definition and lemma from 3.2.

Definition. Define $c^{\Phi}_{\mathbf{T}}$ as the least c such that $\mathbf{T}$ proves no statements of the form $C(n) > c$ for any string n.

Lemma. *There exists a Turing machine $A_{\mathbf{S},\mathbf{T}}$ such that*

$$\mathbf{S} \nvdash A_{\mathbf{S},\mathbf{T}} \not\downarrow n$$

and

$$\mathbf{T} \nvdash A_{\mathbf{S},\mathbf{T}} \not\downarrow n$$

for any n.

We also make the following definition

Definition 4.2.4. Define $r^{\Phi}_{\mathbf{T}}$ as the least r such that $\mathbf{T}$ does not prove $\Phi_r \uparrow$

Note that it follows from the above lemma that neither $\mathbf{S}$ nor $\mathbf{T}$ can prove $A_{\mathbf{S},\mathbf{T}} \uparrow$ and thus the lemma implies that both $r_{\mathbf{T}}$ and $r_{\mathbf{S}}$ exist.

Theorem 4.2.5. *The following statements are equivalent*

a. There exists a Π_1-sentence τ which is provable in $\mathbf{T}$, but not in $\mathbf{S}$.

b. $r^{\Phi}_{\mathbf{S}} = r^{\Phi}_{\mathbf{T}}$ and $c^{\Phi}_{\mathbf{S}} < c^{\Phi}_{\mathbf{T}}$ under some Gödel numbering Φ of Turing machines

c. $r^{\Psi}_{\mathbf{S}} < r^{\Psi}_{\mathbf{T}}$ and $c^{\Psi}_{\mathbf{S}} < c^{\Psi}_{\mathbf{T}}$ under some Gödel numbering Ψ of Turing machines

d. $r^{\Omega}_{\mathbf{S}} < r^{\Omega}_{\mathbf{T}}$ and $c^{\Omega}_{\mathbf{S}} = c^{\Omega}_{\mathbf{T}}$ under some Gödel numbering Ω of Turing machines

Proof. $(a) \Rightarrow (b)$:

Put $\Phi_0 = A^{(0)}_{\mathbf{S},\mathbf{T}}$ and $\Phi_1 = D^0_\tau$. Neither $\mathbf{S}$ nor $\mathbf{T}$ can prove $A_{\mathbf{S},\mathbf{T}} \uparrow$ by the lemma above, thus $r_{\mathbf{S}} = r_{\mathbf{T}} = 0$ since $\Phi_0 = A^{(0)}_{\mathbf{S},\mathbf{T}}$ and $A_{\mathbf{S},\mathbf{T}} \uparrow \leftrightarrow A^{(0)}_{\mathbf{S},\mathbf{T}} \uparrow$ is provable in both $\mathbf{S}$ and $\mathbf{T}$.

Since

$$\mathbf{T} \vdash A^{(0)}_{\mathbf{S},\mathbf{T}} \not\downarrow 0$$

$$\mathbf{T} \vdash D_\tau \uparrow$$

$$\mathbf{T} \vdash D_\tau \uparrow \leftrightarrow D^0_\tau \uparrow$$

we can prove $C(0) > 1$ in $\mathbf{T}$ and therefore $c_{\mathbf{T}} > 1$.

Now we want to show $c_{\mathbf{S}} \leq 1$. Note that this is equivalent to that $\mathbf{S}$ proves no statements of the form "$C(n) > 1$" for any string n. Now assume towards a contradiction that $\mathbf{S} \vdash C(n) > 1$ for some n. Suppose $n = 0$, then we must have $\mathbf{S} \vdash D^0_\tau \uparrow$, but that means $\mathbf{S} \vdash D_\tau \uparrow$ and thus $\mathbf{S} \vdash \tau$, a contradiction. Suppose instead $n \neq 0$, then by the assumption $\mathbf{S} \vdash C(n) > 1$ we must have $\mathbf{S} \vdash A^{(0)}_{\mathbf{S},\mathbf{T}} \not\downarrow n$. Since $n \neq 0$, this means $\mathbf{S} \vdash A_{\mathbf{S},\mathbf{T}} \not\downarrow n$ for

some n which contradicts the lemma above. We conclude that $\mathbf{S}$ proves no statements of the form "$C(n) > 1$" for any string n, i.e. $c_{\mathbf{S}} \leq 1$, and thus $c_{\mathbf{S}} < c_{\mathbf{T}}$.

Similar arguments to the one above show that $(\Psi_0, \Psi_1) = (D^0_\tau, A^{(0)}_{\mathbf{S},\mathbf{T}})$ proves $(a) \Rightarrow (c)$ and $(\Omega_0, \Omega_1) = (D^{(0)}_\tau, A_{\mathbf{S},\mathbf{T}})$ proves $(a) \Rightarrow (d)$, see Ibuka et al. (2011) for details.

$(a) \Leftarrow (b)$ and $(a) \Leftarrow (c)$:

$c_{\mathbf{S}} < c_{\mathbf{T}}$ implies $\mathbf{T} \vdash C(n) > c_{\mathbf{S}}$ for some string n while by definition $\mathbf{S} \nvdash C(n) > c_{\mathbf{S}}$ for any n. This proves the implication since $C(n) > c_{\mathbf{S}}$ is a Π_1-statement.

$(a) \Leftarrow (d)$:

$r^{\Omega}_{\mathbf{S}} < r^{\Omega}_{\mathbf{T}}$ implies that the least r such that $\mathbf{S}$ does not prove $\Omega_r \uparrow$ is strictly less than the least r such that $\mathbf{T}$ does not prove $\Omega_r \uparrow$. This proves the implication since this means that for some r we have $\mathbf{T} \vdash \Omega_r \uparrow$ while $\mathbf{S} \nvdash \Omega_r \uparrow$ and $\Omega_r \uparrow$ is a Π_1-statement.

□

4.3 Bienvenu et al. (2014)

In Bienvenu et al. (2014) the authors study the strength of the theory one gets by adding all true statements of the form "$C(n) > w$" to **PA** and they prove among other things the following result

Theorem 4.3.1. *Suppose* $\mathbf{T}$ *is the theory obtained by adding all true statements of the form "*$C(n) > w$*" to* **PA***, then* $\mathbf{T}$ *proves the halting or non halting of any program.*

Proof. Define $C^t(n)$ as the complexity measure obtained by limiting our fixed universal Turing machine U_Φ to only running each program for t steps. We see that by making t sufficiently large we get $C(n) = C^t(n)$ for every string n in W, where W is any finite set of strings. Let W_L be the set of all strings of length L and denote the minimal t such that $C(n) = C^t(n)$ for every n in W_L by t_L.

We now claim that for any halting program p with

$$|p| < L - e - 1 - 2|L| - 1 \tag{4}$$

we must have $\text{halt}(p) < t_L$, here L is assumed to be taken sufficiently large for the expression to make sense and e is such that Φ_e is the Turing machine which when given the concatenated string $1^{|L|}0Lp$ performs the following steps:

1. Measure the number of steps t needed for $U(p)$ to halt. To see why this is possible note that we can modify a universal Turing machine to as it simulates another Turing machine M also measure the number of steps needed for M to halt, remember that we assume p is a halting program.

2. Run t steps of $U(l)$ for every string l of length strictly less than L.

3. Output the least string n of length L such that $C^t(n) \geq L$, i.e. a string that was not a result of a run in step 2, as noted earlier such a string always exists by a simple counting argument.

Assume towards a contradiction that $\text{halt}(p) \geq t_L$, then in step 3 $C(n) = C^t(n)$ and thus $C(n) \geq L$, however by (4) we also have

$$C(n) \leq e + 1 + C_{\Phi_e}(n) \leq e + 1 + |1^{|L|}0Lp| <$$

$$e + 1 + |L| + 1 + |L| + L - e - 1 - 2|L| - 1 = L$$

We therefore see that $\text{halt}(p) < t_L$ for any halting p.

A true statement of the form "$C(n) \leq w$" is a true Σ_1-statement and is thus provable in $\mathbf{T}$, but since also all true statements of the form "$C(n) > w$" are axioms of $\mathbf{T}$ we have proofs in $\mathbf{T}$ of the complexity of any string n. We can then prove the value of t_L for any L in $\mathbf{T}$.

From the argument above, which is formalizable in $\mathbf{T}$, we can prove in $\mathbf{T}$ that we only need to run a program p for at most t_L steps, where L depends on $|p|$, to know if p will ever halt and it is also provable in $\mathbf{T}$ whether a program halts within t_L steps or not. Finally let us just note that we only need the new axioms to prove non halting of programs, this is because if $U(p)$ halts, then this as previously noted is a true Σ_1-statement.

□

It should be noted that in the corresponding theorem of Bienvenu et al. (2014), Theorem 5 to be exact, the authors go further and use the provable non halting of programs to prove true Π_1-statements. We also see that the theory $\mathbf{T}$ obtained is not recursively enumerable, otherwise it would contradict Remark (iii) Theorem 3.1.1. Let us also explain what we hinted at in the introduction, that since for any Turing machine M and input n we can find a program p such that $M(n) \simeq U(p)$ we see that $\mathbf{T}$ in fact proves whether any given Turing machine on any given input halts or not.

Bienvenu et al. (2014) goes on to prove something similar to the following result, in which we as usual assume we have fixed a universal Turing machine

Theorem 4.3.2. *Suppose* $\mathbf{T}$ *is the theory obtained by adding* $C(n_L) \geq L$ *to* **PA**, *where* n_L *is the least string* n *with* $|n| = L$ *such that* $C(n) \geq L$ *is true. Then there exists a constant* c *(independent of* L*) such that* $\mathbf{T}$ *proves the halting or non halting of any program of length less than* $L - 2|L| - c$.

Proof. (Proof idea) The argument is similar to the one in the proof of Theorem 4.3.1. Note in that proof that in the description of Φ_e we only really need t to be so large that for all strings n of length L preceding n_L we have $C^t(n) < L$ in order to get a contradiction. Also note the minimal value of such a t is provable from the axiom $C(n_L) \geq L$ and the previously mentioned fact that for any string n of length L preceding n_L we have that $C(n) < L$ is a true Σ_1-statement. From the proof of Theorem 4.3.1 we see that setting $c = e + 2$ completes the proof. □

The two theorems above are a strong testament to the axiomatic strength of statements of the form "$C(n) > w$", however as commented by Christopher P. Porter[12] we should keep in mind that our use of them above lay in the fact that they contained information about the halting of Turing machines and are thus no more powerful than axioms of that form in this regard.

[12] http://www.cpporter.com/wp-content/uploads/2013/08/PorterCambridge2013.pdf

5 Conclusions

We have seen that Chaitin's theorem is an alternative to Gödel's first theorem for proving incompleteness in a theory **T**, where **T** is a recursively enumerable extension of **Q**. In Section 3.4 we saw that Chaitin's theorem is as strong as the original version of Gödel's first incompleteness theorem in the sense that we can use it to show that consistency implies the existence of true unprovable sentences in **T**. We noted however that, unlike the version of Gödel's first incompleteness theorem due to Rosser, Chaitin's result needs a stronger assumption such as ω-consistency to prove incompleteness in the strict sense of the word. We then looked at Kritchman and Raz (2010) that showed us how to prove a statement equivalent to Gödel's second incompleteness theorem from Chaitin's result. The proof used among other things our elementary observation that by Chaitin's theorem we can not prove statements of the form "$C(n) > w$ for arbitrarily large w, yet we can easily see that n such that those statements are true exist. We also noted that the proof of Chaitin's theorem is, unlike that of Gödel's first incompleteness theorem, not constructive in the sense that it does not supply us with an explicit unprovable sentence, it merely guarantees it's existence.

In Section 3.2 we gave an overview of the discussion surrounding the proposed principle

> A set of axioms of complexity N cannot yield a theorem of complexity substantially greater than N

After contributions from van Lambalgen, Raatikainen, Chaitin himself and others the matter is resolved with the conclusion that Chaitin's theorem does not motivate such a principle. In fact there are even some trivial counter examples to the principle when using the standard Kolmogorov complexity as complexity measure. The discussion did however lead Raatikainen and Ibuka et al. (2011) to show more precisely how we can manipulate our universal Turing machine to in a sense control when statements of the form "$K(n) > w$" are unprovable in our theory. Key to this argument was that for a recursively enumerable theory we can always create a Turing machine M that searches the theory for proofs about the Turing machine's own output and if such a proof is found then outputs something which contradicts the proof. The conclusion was that our theory could not prove any theorem about the output of M and thus could not prove any theorem of the form "$K(n) > e$", where e is the cost of running M on our fixed universal Turing machine, which would imply that the output of M is not n. One conclusion from this was that the connection between any property of **T** and the least c such that **T** does not prove any statements of the form "$C(n) > c$" is more

or less accidental and depends on what acceptable Gödel numbering we use for our Turing machines.

Calude and Jürgensen (2005) proposed a complexity measure δ_i that satisfies the principle discussed in Section 3.2. We criticized this approach since δ_i violates an implicit assumption when discussing the principle, namely that there should also exist well formed formulas that have substantially greater complexity than that of the axioms. As far as this author knows, there is no known interesting complexity measure that satisfies the mentioned principle. Finally we saw in Bienvenu et al. (2014) that true statements of the form "$K(n) > w$" encodes halting information about general programs of a universal Turing machine.

References

Bienvenu, Laurent, Andrei Romashchenko, Alexander Shen, Antoine Taveneaux, and Stijn Vermeeren (2014), "The axiomatic power of Kolmogorov complexity." *Ann. Pure Appl. Logic*, 165, 1380–1402, URL http://dx.doi.org/10.1016/j.apal.2014.04.009.

Boolos, George (1989), "A new proof of the gödel incompleteness theorem." *Notices of the American Mathematical Society*, 36, 388–390.

Boolos, George S., John P. Burgess, and Richard C. Jeffrey (2007), *Computability and logic*, fifth edition. Cambridge University Press, Cambridge, URL http://dx.doi.org/10.1017/CBO9780511804076.

Calude, Cristian S. and Helmut Jürgensen (2005), "Is complexity a source of incompleteness?" *Adv. in Appl. Math.*, 35, 1–15, URL http://dx.doi.org/10.1016/j.aam.2004.10.003.

Chaitin, G. J. (1992a), "Information-theoretic incompleteness." *Appl. Math. Comput.*, 52, 83–101, URL http://dx.doi.org/10.1016/0096-3003(92)90099-M.

Chaitin, G. J. (1992b), "LISP program-size complexity. II, III, IV." *Appl. Math. Comput.*, 52, 103–126, 127–139, 141–147, URL http://dx.doi.org/10.1016/0096-3003(92)90100-F.

Chaitin, Gregory J. (1974), "Information-theoretic limitations of formal systems." *J. Assoc. Comput. Mach.*, 21, 403–424.

Chaitin, Gregory J. (1982), "Gödel's theorem and information." *Internat. J. Theoret. Phys.*, 21, 941–954, URL http://dx.doi.org/10.1007/BF02084159.

Fallis, Don (1996), "The source of Chaitin's incorrectness." *Philos. Math. (3)*, 4, 261–269, URL http://dx.doi.org/10.1093/philmat/4.3.261.

Grenet, Bruno (2010), "Acceptable complexity measures of theorems." *Complex Systems*, 18, 403–425.

Ibuka, Shingo, Makoto Kikuchi, and Hirotaka Kikyo (2011), "Kolmogorov complexity and characteristic constants of formal theories of arithmetic." *MLQ Math. Log. Q.*, 57, 470–473, URL http://dx.doi.org/10.1002/malq.201010017.

Kikuchi, Makoto (1997), "Kolmogorov complexity and the second incompleteness theorem." *Arch. Math. Logic*, 36, 437–443, URL http://dx.doi.org/10.1007/s001530050074.

Kikuchi, Makoto, Taishi Kurahashi, and Hiroshi Sakai (2012), "On proofs of the incompleteness theorems based on Berry's paradox by Vopěnka, Chaitin, and Boolos." *MLQ Math. Log. Q.*, 58, 307–316, URL http://dx.doi.org/10.1002/malq.201110067.

Kritchman, Shira and Ran Raz (2010), "The surprise examination paradox and the second incompleteness theorem." *Notices Amer. Math. Soc.*, 57, 1454–1458.

Li, Ming and Paul Vitányi (2008), *An introduction to Kolmogorov complexity and its applications*, third edition. Texts in Computer Science, Springer, New York, URL http://dx.doi.org/10.1007/978-0-387-49820-1.

Raatikainen, Panu (1998), "On interpreting Chaitin's incompleteness theorem." *J. Philos. Logic*, 27, 569–586, URL http://dx.doi.org/10.1023/A:1004305315546.

Rogers, Hartley, Jr. (1987), *Theory of recursive functions and effective computability*, second edition. MIT Press, Cambridge, MA.

Svozil, Karl (1993), *Randomness & undecidability in physics*. World Scientific Publishing Co., Inc., River Edge, NJ, URL http://dx.doi.org/10.1142/1524.

van Lambalgen, Michiel (1989), "Algorithmic information theory." *J. Symbolic Logic*, 54, 1389–1400, URL http://dx.doi.org/10.2307/2274821.