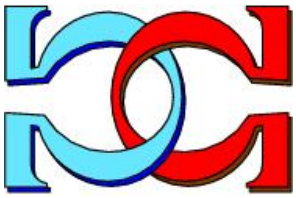

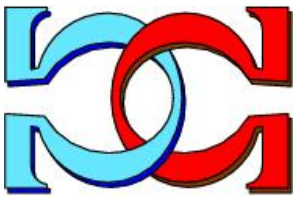

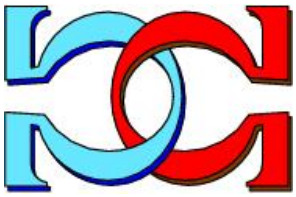

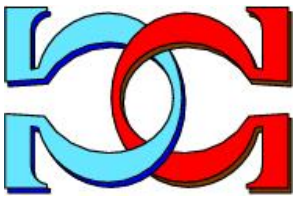

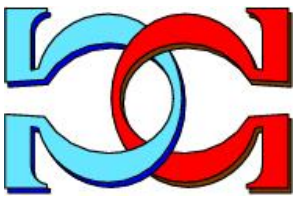

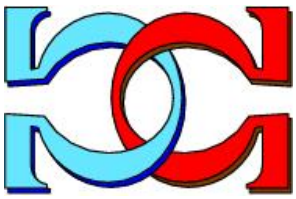

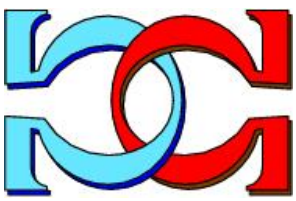

CDMTCS
Research
Report
Series

Incompleteness and the Halting Problem

C. S. Calude

University of Auckland, New Zealand

CDMTCS-546
August 2020/December 2020

Centre for Discrete Mathematics and
Theoretical Computer Science

Cristian S. Calude

Incompleteness and the Halting Problem

Abstract. We present an abstract framework in which we give simple proofs for Gödel's First and Second Incompleteness Theorems and obtain, as consequences, Davis', Chaitin's and Kritchman-Raz's Theorems.

Keywords: Gödel's Incompleteness Theorems, the Halting Problem

1. Introduction

Gödel's Incompleteness Theorems are arguably the most important results in mathematical logic; they have deep implications not only in mathematics and logic, but also in other fields like philosophy and computer science.

Gödel's First Incompleteness Theorem states that in every computably enumerable rich enough consistent formal theory, there exists a sentence that cannot be proved nor disproved within the theory. This result is purely syntactic; if the formal theory has a sound semantics, then from the Gödel's First Incompleteness Theorem one can deduce the existence of a true, but unprovable in the theory, sentence. The Second Incompleteness Theorem states that the consistency of a computably enumerable rich enough consistent formal theory cannot be proved within the theory.

In this paper we give simple proofs, in an abstract framework, for both Incompleteness Theorems based on the undecidability of the Halting Problem. The undecidable sentences have a "relatively simple structure", lowest in Gödel's hierarchy [8]. From these results we deduce the First Incompleteness Theorem in Davis' version [10] and Chaitin's version [5, 6], as well as Kritchman and Raz's version [17] of the Second Incompleteness Theorem.

2. Framework

We assume elementary knowledge of computability theory [21] and algorithmic information theory [2]. In particular, we use an (abstract) universal programming language operating with binary strings. Such a language is equivalent to Turing machines, or register machines, see [21, 11].

Presented by **Daniele Mundici**; *Received* December 1, 2020

The Halting Problem [9, p. 70–71] requires a decider program which can determine, given any pair (program, input), whether the program *will* eventually *halt* when run on input. The future tense indicates that the decision has to be made "in advance". There are no resource limitations on the amount of memory or time required for the decider program's execution. The decider program should stop in *finite time* and give the *correct answer for all* possible pairs (program, input). Arguably the most important result in computability theory is the undecidability of the Halting Problem.[1]

THEOREM 1 (Halting Theorem). *No program can solve the Halting Problem for a universal programming language.*

COROLLARY 1. *There exists a computably enumerable set which is not computable.*

The Halting Theorem fails to apply when *we know the number of inputs of given lengths that stop on a given program.*

LEMMA 1. *Consider a universal programming language having as inputs binary strings. Fix an $n > 0$ and a program P. If we know the number of inputs v of length less than or equal to n for which $P(v)$ stops, then the Halting Problem for the set $\{(P, v) \mid |v| \leq n\}$ is decidable.*

Indeed, if we know the number N of programs $|v| \leq n$ for which $P(v)$ stops, then we can run in parallel $P(v)$ for all programs $|v| \leq n$ till the N halting inputs show up and stop; all other programs of length less than or equal to n never stop on P.

A formula (sentence) S is provable in a formal theory $\mathbf{T}$ if there exists a proof π in $\mathbf{T}$ for S; in this case we write $\vdash S$. The formula $\neg S$ is the negation of S.

The following properties of a formal theory $\mathbf{T}$ will be used in what follows:

- $\mathbf{T}$ is *computably enumerable* if the set of proofs (hence, theorems) in $\mathbf{T}$ is computably enumerable.
- $\mathbf{T}$ is *rich enough*[2] if a certain amount of elementary arithmetic can be carried out in it.

[1] According to Copeland [7, p. 40], the "Halting Problem was so named (and, it appears, first stated) by Martin Davis" [9, p. 70–71]. Davis credited Kleene's monograph [16, p. 382] with an informal proof.

[2] The minimal amount of arithmetic required will be clear in each case.

- **T** is *consistent* if there is no sentence S in **T** such that $\vdash S$ and $\vdash \neg S$.
- **T** is *syntactically complete* if for every sentence S in **T** we have $\vdash S$ or $\vdash \neg S$.

Gödel's First Incompleteness Theorem is syntactic:

> *In every computably enumerable, rich enough and consistent formal theory, there exists a sentence that cannot be proved nor disproved within the theory.*

From the First Incompleteness Theorem one can deduce a semantic, more intuitive, variant:

> *In every computably enumerable, rich enough and consistent formal theory, there exists a true, but unprovable sentence.*

But, what semantics should one choose for **T**? According to [18, p. 18]

> Gödel never used, neither in his statements nor in his proofs, the notion of "truth", which is not a formal concept. It is necessary to stress this point, because in current readings of this theorem, it is often too hastily said that it shows the existence of "statements that are true but unprovable" in Arithmetics. "True" statements? But where, how, according to which notion of truth?

This question will be discussed in Section 3.

Gödel's Second Incompleteness Theorem states that

> *The consistency of a computably enumerable, rich enough and consistent formal theory cannot be proved within the theory.*

For more details see [22]. The paper [18] contains a deep analysis of the incompleteness phenomenon and, in particular, of the distinction between the syntactic and semantic variants.

3. First Incompleteness Theorems

In this section we present an abstract form of the First Incompleteness Theorem and, as consequences, we deduce syntactic variants of Davis' and Chaitin's Theorems. Next we introduce an abstract form of semantics and prove a semantic version of the First Incompleteness Theorem from which

we deduce Davis' and Chaitin's Theorems.

The framework developed in this paper is justified by Gödel's view of a formal system as just a mechanical procedure for generating "provable formulas" [14]:

> ... due to A. M. Turing's work, a precise and unquestionably adequate definition of the general concept of formal system can now be given, the existence of undecidable arithmetical propositions and the non-demonstrability of the consistency of a system in the same system can now be proved rigorously for every consistent formal system containing a certain amount of finitary number theory.
> Turing's work gives an analysis of the concept of "mechanical procedure" (alias "algorithm" or "computation procedure" or "finite combinatorial procedure"). This concept is shown to be equivalent with that of a "Turing machine". ... *A formal system can simply be defined to be any mechanical procedure for producing formulas, called provable formulas.*[3]

In what follows we will work with a computably enumerable and rich enough formal theory (shortly, formal theory) **T** *"for almost all mathematics".* An example is **ZFC**, Zermelo–Fraenkel set theory with the axiom of choice, the standard axiomatic set theory for mathematics. This means that virtually every mathematical proof, and in particular the proofs in this paper, can be carried out in **T**.

Consider a formal theory **T** using sentences over an alphabet Σ containing the symbol $\neg$. In **T** we fix two sets: P_1 is a non-empty computable set of sentences over $\Sigma \setminus \{\neg\}$ and $P_2 = \{\neg S \mid S \in P_1\}$. We also define two sets of sentences in P_1 provable in **T**:

$$\mathrm{Prov}_1 = \{S \in P_1 \mid \vdash S\}, \mathrm{Prov}_2 = \{S \in P_1 \mid \vdash \neg S\}.$$

THEOREM 2 (First incompleteness Theorem). *Let* **T** *be a consistent formal theory such that* Prov_1 *is not computable. Then, there exists a sentence in* P_1 *such that neither itself nor its negation are provable in* **T**.

PROOF. From consistency it follows that $\mathrm{Prov}_1 \cap \mathrm{Prov}_2 = \emptyset$. Assume by absurdity that for every $S \in P_1$ we have either $\vdash S$ or $\vdash \neg S$. Fix $S \in P_1$ and

[3] My italics.

computably enumerate all proofs $\pi_i, i = 1, 2, \ldots$ in $\mathbf{T}$; from the absurdity assumption it follows that eventually either a proof π_i for S or a proof π_j for $\neg S$ will show up in the enumeration. In the first case $\vdash S$, so $S \in \text{Prov}_1$; in the second case $\vdash \neg S$, so $S \in \text{Prov}_2$, hence by consistency, $S \notin \text{Prov}_1$. This proves that Prov_1 is computable, a contradiction. ■

DEFINITION 1. *A sentence that neither itself nor its negation is provable in* $\mathbf{T}$ *is called* undecidable (*in* $\mathbf{T}$).

In the context of Theorem 2, if S is undecidable in $\mathbf{T}$, then the set $P_1 \setminus \{S\}$ is computable and the set $\text{Prov}_1 \setminus \{S\}$ is not computable, hence the theorem can be applied again to $P_1 \setminus \{S\}$.

COROLLARY 2. *Let* $\mathbf{T}$ *be a consistent formal theory such that* Prov_1 *is not computable. Then, there exist infinitely many undecidable sentences in* $\mathbf{T}$.

Consider now a universal programming language like the Turing machine language [21] or Turing-Post programming language [10]. The sentence "$N(P, v)$" says that the program P never halts on input v. So, for every program P and string v, "$N(P, v)$" is a perfectly definite sentence which is either true (if P never halts) or false (if P eventually halts). The falsity of "$N(P, v)$" can always be proved by exhibiting the sequence of program instructions run by P on v which leads to termination. However, when "$N(P, v)$" is true, no finite sequence of instructions suffices to demonstrate it.

The sentences "$N(P, v)$" can be formalised in a formal theory $\mathbf{T}$ and they form a computable set P_1. As a consequence of the undecidability of the Halting Problem the set Prov_1 is computably enumerable but not computable, so we get

COROLLARY 3 (Syntactic version of Davis' Theorem [10]). *Let* $\mathbf{T}$ *be a consistent formal theory. There exist infinitely many undecidable sentences "*$N(P, v)$*" in* $\mathbf{T}$.

We may still be able to prove that a particular "$N(P, v)$" is provable in $\mathbf{T}$ by a logical analysis of P's behaviour on v and, indeed, there are infinitely many such cases.

To obtain Chaitin's Theorem we need to introduce the notion of Kolmogorov-Chaitin complexity, shortly, complexity. To define the complexity of a binary string w we need a binary encoding of programs P, see [21], i.e. an injective, computable function $P \to \mathit{code}(P)$, and a computable encoding $\langle\ ,\rangle$ of pairs of strings (from $z = \langle x, y\rangle$ we retrieve uniquely, in a computable

way, x, y).[4] A string x is described by a pair $\langle P, v\rangle$ such that $P(v) = x$. The length of the description is $|\langle P, v\rangle|$. A string x has complexity n, in writing $C(x) = n$, if

1) there is a program P and a string v such that $P(v) = x$ and $|\langle P, v\rangle| = n$, and
2) n is the smallest natural number for which 1) holds.

The sentences of the form "$C(x) > n$" can be formalised in a formal theory $\mathbf{T}$ and they form a computable set P_1. Again, as a consequence of the Halting Theorem, the complexity function C is incomputable [2], hence the set Prov_1 is computably enumerable but not computable, so we get

COROLLARY 4 (Syntactic version of Chaitin's Theorem [5]). *Let* $\mathbf{T}$ *be a consistent formal theory. There exist infinitely many undecidable sentences "*$C(v) > n$*" in* $\mathbf{T}$.

We now present a form of semantic incompleteness. To this aim we need a suitable formal definition of "true sentences" of $\mathbf{T}$. The completeness of propositional calculus and, more interestingly, of predicate calculus, offer solutions: a formula in such a calculus is true iff it is true under every interpretation [22]. In the first example the truth/validity of a formula can be decided with truth tables; in the second, more complex case, truth was defined by Gödel [12] as part of his proof of the Completeness Theorem.[5] In contrast with the case of propositional calculus, Church [20] proved that truth is not decidable in predicate calculus. These formulations of truth are not powerful enough for the complexity of the formal systems aiming to be foundations for the whole mathematics. Following [18, p. 24] we can ask:

> ... how can we state, in Mathematics, that an assertion is true without demonstrating it (or taking it for hypothesis)? The interlocutor must then produce a proof convincing us of the "(unprovable) truth" of G.[6] The hypothesis of consistency, he/she points out, implies that G is unprovable (first theorem). And since G "asserts" that it is not provable ... then it is true. This reasoning based on the "meaning" of G is informal, vague and unwritten. But once formalised, it is a

[4] An example is $\langle x_1x_2 \dots x_n, y\rangle = x_1 0 x_2 0 \dots x_n 1 y$.

[5] Which became the first important link between proof theory – that studies which sentences can be formally proven in formal theories – and model theory – that deals with what is true in different models of theories.

[6] This refers to Gödel's original proof [13].

semantic version of the rigorous formal implication $PA \vdash (Cons \to G)$ that constitutes the core of the second theorem.

To make matters worse we recall[7]

THEOREM 3 (Tarski's Undefinability Theorem [23]). *Arithmetical truth cannot be defined in Peano Arithmetic.*

Still, a painless formalisation of truth suited for our discussion can be provided. Motivated by the simple structure of sentences in Corollary 3 and Corollary 4 we introduce the following:

DEFINITION 2. By $True$ we denote a subset of $P_1 \cup P_2$ satisfying the property

- for every $S \in P_1, S \in True$ iff $\neg S \notin True$.

The elements S in $True$ are called *true sentences* and are denoted by $\models S$. True sentences satisfy a semantic form of consistency:

LEMMA 2. *There is no $S \in P_1$ such that $\models S$ and $\models \neg S$.*

Fix a set $True$. We say that

- $\mathbf{T}$ is *sound* for $True$ if for every sentence S in $\mathbf{T}$, if $\vdash S$, then $\models S$.
- $\mathbf{T}$ is *semantically complete* for $True$ if for every sentence S in $\mathbf{T}$, if $\models S$ then $\vdash S$.

LEMMA 3. *Every formal theory $\mathbf{T}$ sound for $True$ is consistent.*

PROOF. If $\mathbf{T}$ is not consistent, then there exists an $S \in P_1$ such that $\vdash S$ and $\vdash \neg S$, so by soundness we get, $\models S$ and $\models \neg S$, contradicting Lemma 2. ■

REMARK 1. *If $\mathbf{T}$ is sound for $True$ and $Prov_1 \neq \emptyset$, then $P_1 \cap True \neq \emptyset$.*

THEOREM 4 (Semantic incompleteness Theorem). *Let $\mathbf{T}$ be a formal theory which is sound for a set $True$ and* Prov_1 *is not computable. Then, there exist in $True$ infinitely many undecidable sentences (in $\mathbf{T}$).*

PROOF. By hypothesis $\mathbf{T}$ is sound for $True$, so by Lemma 3, $\mathbf{T}$ is consistent; as Prov_1 is not computable, Theorem 2 applies, so we get a sentence $G \in P_1$ such that G and $\neg G$ are not provable in $\mathbf{T}$. From Lemma 2 one and only one of these two sentences is true. Finally, we use Corollary 2 to get infinitely many undecidable true sentences (in $\mathbf{T}$). ■

[7] See [1, p. 128]; Gödel discovered this result in 1930, while proving his First Incompleteness Theorem [13], well before 1933, the year of Tarski's publication, see [19].

COROLLARY 5 (Davis' Theorem [10]). *Let* $\mathbf{T}$ *be a formal theory which is sound for the set* $True$ *of all true sentences "*$N(P, v)$*". Then there exist infinitely many true sentences "*$N(P, v)$*" unprovable in* $\mathbf{T}$.

COROLLARY 6 (Chaitin's Theorem [5]). *Let* $\mathbf{T}$ *be a formal theory which is sound for the set* $True$ *of all true sentences "*$C(v) > n$*". Then there exist infinitely many true sentences "*$C(v) > n$*" unprovable in* $\mathbf{T}$.

The original form of Chaitin's Theorem is: *For any rich enough consistent formal theory* $\mathbf{T}$*, there exists a positive integer* L *such that for every binary string* v*, the statement "*$C(v) > L$*" cannot be proved in* $\mathbf{T}$. At a superficial level one may think that the statement says that Corollary 6 is true for almost all true sentences "$C(v) > n$", but this is false: there exist infinitely many true statements "$C(v) > n$" provable in $\mathbf{T}$.

4. Second Incompleteness Theorem

In this section we present an abstract form of the Second Incompleteness Theorem from which we deduce a variant based on the Halting Theorem and Kritchman-Raz Theorem.

DEFINITION 3. *A formal theory* $\mathbf{T}$ *with a set* $True$ *is*

- *reciprocally sound* for $True$ if for every $S \notin True$ we have $\vdash \neg S$.

THEOREM 5 (Second Incompleteness Theorem). *Let* $\mathbf{T}$ *be a consistent formal theory which is sound and reciprocally sound for a set* $True$ *and the set* Prov_1 *is not computable. Then,* $\mathbf{T}$ *cannot prove its consistency.*

PROOF. Assume that the computable set P_1 (which is infinite by Corollary 2) is injectively coded in the form $\{S(x) \mid x \in \{0,1\}^*\}$, where $\{0,1\}^*$ is the set of all binary strings. For each $n \geq 1$ consider the set

$$M_n = \{S(x) \in P_1 \mid S(x) \in True, |x| \leq n\}.$$

By Theorem 4 there exists a sentence $S(z) \in True \setminus Prov_1$, hence for every $n \geq |z|$, $S(z) \in M_n \setminus Prov_1$ and $1 \leq \#M_n \leq 2^{n+1} - 1$.

Fix $n \geq |z|$. If M_n has one element, then all the remaining $2^{n+1} - 2$ sentences $S(y)$ with $|y| \leq n$ are not in M_n, hence not in $True$, so by reciprocal soundness, $\vdash \neg S(y)$. By computably enumerating the proofs $\pi_1, \pi_2, \ldots$ in $\mathbf{T}$ till all $2^{n+1}-2$ proofs for these sentences show up, we get all $2^{n+1}-2$ sentences $S(y)$ with $|y| \leq n$ and $S(y) \notin True$;[8] the remaining unique sentence $S(z)$

[8]This process is analogous to Lemma 1.

is in $M_n \setminus Prov_1$. However, for every $|y| \leq n, y \neq z$ we have $\vdash \neg S(y)$, and because $\mathbf{T}$ is rich enough, $\vdash S(z)$, in contradiction with Theorem 4. Using the consistency of $\mathbf{T}$ and the fact that $\mathbf{T}$ is rich enough, we deduce that $\mathbf{T}$ proves that M_n has at least 2 elements. The above reasoning can continue by assuming that M_n has exactly 2 elements and getting again a contradiction with Theorem 4. Step by step we will reach the stage when $\mathbf{T}$ proves that M_n has at least $2^{n+1} - 1$ elements, a contradiction.

■

COROLLARY 7. *Let* $\mathbf{T}$ *be a formal theory which is sound for the set of all true sentences "*$N(P, v)$*". Then* $\mathbf{T}$ *cannot prove its consistency.*

PROOF. The set $Prov_1$ is computably enumerable but not computable because of the Halting Theorem. The set $True = \{$"$N(P, v)$" $\mid P(v)$ does not stop$\}$ satisfies Definition 2; furthermore, it is reciprocally sound because if "$N(P, v)$" $\notin True$, then $P(v)$ stops, hence $\mathbf{T}$ can prove it by simply describing the running of P on v until it stops. Finally, in view of the consistency of $\mathbf{T}$ and Lemma 3, the conclusion follows from Theorem 5. ■

COROLLARY 8 (Kritchman-Raz Theorem [17]). *Let* $\mathbf{T}$ *be a formal theory which is sound for the set of true sentences "*$C(x) > n$*". Then* $\mathbf{T}$ *cannot prove its consistency.*

PROOF. As in the proof of Corollary 7, Prov_1 is computably enumerable but not computable. The set $True = \{$"$C(x) > n$" $\mid C(x) > n\}$ satisfies Definition 2 and it is reciprocally sound because if "$C(x) > n$" $\notin True$, then there exists a program P and a string v such that $P(v) = x$ and $|\langle P, v\rangle| \leq n$, hence $\mathbf{T}$ can prove it by giving P and v and describing the running of $P(v)$ until it stops and produces x. Again, by consistency of $\mathbf{T}$ and Lemma 3, the conclusion follows from Theorem 5.

■

5. Conclusions

In this paper we have used the Halting Theorem to prove an abstract form of Gödel's First Incompleteness Theorem (Theorem 2) from which we have derived a semantic version (Theorem 4), Gödel's Second Incompleteness Theorem (Theorem 8) as well Davis', Chaitin's Theorems (also in syntactic variants) and finally Kritchman-Raz's Theorem. Theorem 4 answers in the affirmative the open problem posed in [3], which referred to unprovable instances of every undecidable problem.

All undecidable sentences in this paper have the form "$\forall x[Pred(x, y)]$", where $Pred$ is a computable predicate. They have a "relatively simple structure", lowest in Gödel's hierarchy [8], in the sense that such a sentence can be refuted by a single counter-example.[9] However, this does not mean that they are simple or mathematically non-interesting. For example, Fermat's Last Theorem and Riemann's Hypothesis have each this form.

No discussion about incompleteness can avoid a philosophical comment. Hintikka [15, p. 35] stated that

> We are now in position to see that any particular proof given to Gödel's first incompleteness theorem is philosophically irrelevant. ... it has no philosophical significance whatsoever. It does not, because it cannot, show anything about the reasons why elementary arithmetic is incomplete.

Would the undecidability of the Halting Problem – the common argument in the unification proposed in this paper – count as a reason for incompleteness?

Acknowledgements

This paper was inspired by many discussions on Gödel's incompleteness with G. Longo and, particularly, his recent paper [18]. I am grateful to M. Davis, F. Kroon, L. Staiger, K. Svozil and A. Withy for comments which improved the paper.

References

[1] Avigad, Jeremy, 'Computability and Incompleteness – Lectures Notes, 2007, 128 pages, `https://www.andrew.cmu.edu/user/avigad/Teaching/candi_notes.pdf`', .

[2] Calude, Cristian, *Information and Randomness—An Algorithmic Perspective*, Springer, Berlin, 2002 (2nd ed.).

[3] Calude, Cristian, and Gheorghe Păun, 'Independent instances for some undecidable problems', *RAIRO. Informatique théorique*, 17 (1983), 1, 49–54.

[4] Calude, Cristian S, and Sergiu Rudeanu, 'Proving as a computable procedure', *Fundamenta Informaticae*, 64 (2005), 1-4, 43–52.

[5] Chaitin, G. J., 'Computational complexity and Gödel's incompleteness theorem', *ACM SIGACT News*, (1971), 9, 11–12.

[6] Chaitin, Gregory J., *Information-Theoretic Incompleteness*, World Scientific, Singapore, 1992.

[9] The paper [4] discusses degrees of incompleteness.

[7] Copeland, B. Jack, (ed.) *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus The Secrets of Enigma*, Oxford University Press, Oxford, 2004.
[8] Davis, Martin, 'Here There Be Monsters', Presentation at Courant Institute, NYU.
[9] Davis, Martin, *Computability and Unsolvability*, McGraw-Hill, New York, 1958.
[10] Davis, Martin, 'What is a computation?', in Cristian S. Calude, (ed.), *Randomness and Complexity. From Leibniz to Chaitin*, World Scientific Publishing, Singapore, 2007, pp. 89–113.
[11] Davis, Martin, *The Universal Computer: the Road from Leibniz to Turing*, W. W. Norton Company, New York, 2017.
[12] Gödel, Kurt, *Über die Vollständigkeit des Logikkalküls*, Ph.D. thesis, University of Vienna, 1929.
[13] Gödel, Kurt, 'Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I', *Monatshefte für Mathematik*, 38 (1931), 173–198.
[14] Gödel, Kurt, 'Postscriptum', in Martin Davis, (ed.), *The Undecidable. Basic Papers on Undecidable, Unsolvable Problems and Computable Functions*, Raven Press, Hewlett, N.Y., 1965, pp. 71–72.
[15] Hintikka, Jaakko, *On Gödel*, Wadsworth, 2000.
[16] Kleene, Stephen C., *Introduction to Metamathematics*, North-Holland, Amsterdam, 1952.
[17] Kritchman, Shira, and Ran Raz, 'The surprise examination paradox and the second incompleteness theorem', *Notices of the AMS*, 57 (2010), 11, 1454–1458.
[18] Longo, Giuseppe, 'Interfaces of incompleteness', in Gianfranco Minati, Mario R., and Abram Eliano Pessa, (eds.), *Systemics of Incompleteness and Quasi-Systems*, Springer Nature, 2019, pp. 3–55.
[19] Murawski, R., 'Undefinability of truth. the problem of the priority: Tarski vs. Gödel', *History and Philosophy of Logic*, 19 (1998), 153–160.
[20] Post, Emil L., 'Introduction to a general theory of elementary propositions', 43 (1921), 3, 163–185.
[21] Sipser, Michael, *Introduction to the Theory of Computation*, 1st edn., International Thomson Publishing, 2013 (3rd ed.).
[22] Smith, Peter, *An Introduction to Gödel's Theorems*, Cambridge University Press, Cambridge, UK, 2013.
[23] Tarski, A., *Pojęcie Prawdy w Językach Nauk Dedukcyjnych*, Nakładem Towarzystwa Naukowego Warszawskiego, 1933.

Cristian S. Calude
School of Computer Science
University of Auckland
Auckland, New Zealand
cristian@cs.auckland.ac.nz