
Are the Foundations of Computer Science Logic-Dependent?

WALTER A. CARNIELLI AND FRANCISCO A. DORIA

ABSTRACT. We formalize computer science (Turing machine theory) within a standard axiomatic framework like Peano Arithmetic or Zermelo–Fraenkel set theory and argue that the manifold examples of unprovable sentences that appear in those axiomatizations show that such an axiomatics, or any similar kind of axiomatics, is unable to prove some naïvely simple facts in the theory. We then change the logical background to a nonclassical, paraconsistent one, and obtain a theory which is close to quantum computing. We conclude with an evaluation of possible “natural” frameworks for the axiomatics of computer science.

1 Prologue

Gödel’s incompleteness phenomenon is Janus-faced. One of its aspects is a negative one: there are many nontrivial mathematical sentences that turn out to be undecidable with respect to strong axiomatic systems, like *ZFC* (Zermelo–Fraenkel theory, plus the Axiom of Choice) or even *ZFC* plus some large cardinal axiom. On the other side, the Gödel phenomenon opens up the unsuspected wealth and weirdness of the world of nonstandard models or of forcing-extended models — just as a brief aside, which should be the meaning of a non-Cantorian, exotic (“fake”) 4-space? (A recent reference on fake 4-manifolds is Scorpan [33].)

How does the Gödel phenomenon affect computer science? In order to examine that question, we will focus on two approaches.

We will first deal here with examples of the incompleteness phenomenon for versions of computer science which are axiomatized within classical theories, that is to say, first-order theories which include arithmetic and are based on the classical first-order predicate calculus. We then add some extra spice to our considerations, and consider a non-classical, paraconsistent, axiomatic framework for computer science.

What do we learn? By far, the most damaging difficulty for axiomatics of computer science which are based on first-order theories like Peano Arithmetic (*PA*) or *ZFC* is the fact that all such theories do not recognize naïvely total recursive functions beyond a certain growth rate. Moreover, as we show in Section 6 this

inability is tied to fundamental, inner workings of the theory itself, since if such a theory were to prove that a certain (partial recursive) function we have noted F is total (see Definition 2.6) then it would prove its own consistency. On the other hand, when we change the logical framework where we are doing our axiomatization (Section 7), we may get a theory which is very close to quantum computation, as in the example that we discuss.

We may summarize our results in this paper in a slogan:

Axiomatized computer science is logic-dependent.

2 A “natural” axiomatics for computer science

We begin with some comments that motivate our main concepts. Our goal here is to discuss possible axiomatic treatments for computer science, as embodied in the theory of Turing machines. (For a background on Turing machines and computability see [22].)

Remark 2.1. Our main tool will be axiomatic systems like the one described below: these will be consistent axiomatic theories (noted S or T) based on a first-order classical predicate calculus that moreover

- have a recursively enumerable set of theorems
- include Peano Arithmetic
- have a model where the arithmetic segment is standard. \square

We will deviate as needed from this blueprint; exceptions will be made fully explicit whenever we change the above conditions.

Example 2.2. Suppose that there is a Turing machine that enumerates all theorems in a theory S . Since S includes Peano Arithmetic, its set of theorems will be recursively enumerable but not recursive, due to Gödel’s incompleteness. Start our enumerating machine and gradually build up a list of theorems of S . Separate those that formally assert “recursive function f_e , which has Gödel number e , is total”. Place them in a second list. These are the S -provably total recursive functions.

Remark 2.3. It is possible to have a well-defined function which however isn’t recursive. The best example is the Busy Beaver function; another example is the counterexample function to the $P = NP$ hypothesis [15]. \square

Now construct by the usual diagonalization procedure a function F over that second list. That function is immediately seen to be recursive (the diagonal procedure over that second listing gives a simple algorithm for it) and total. Yet it cannot be S -provably total.

That it to say, no theorem of S has the form “ F is total”. Also, as S is sound — a fact that is a consequence of its arithmetic portion having an interpretation in the standard model — the negation of that sentence, “ F isn’t total” cannot appear among the theorems of S . Therefore such a sentence is independent of the axioms of S . \square

(On that function F see Definition 2.6 and Section 6.) Let us clarify a few points in our example:

- A Turing machine M_S that enumerates all theorems of S can operate as follows. First, start a listing of all well-formed formulae in the language of S . Then consider each integer, $1, 2, 3, \dots, k, \dots$ and see if each such integer is the Gödel number of the proof of some well-formed formula in the language of S .
- If so, pick it up from the previous listing and place it in a new list, which will be the enumeration of theorems of S .

“Reasonable” axiomatics

Quotation marks are required here, as this is one of the many possibilities, albeit a very reasonable one. We can take as a starting point the following: Turing machine theory can be seen as arithmetic under disguise. Specifically, it can be seen as a privileged domain within arithmetic: the theory of Diophantine equations. So, we “translate” Turing machine theory into the theory of Diophantine equations [18] in order to axiomatize it.

Of course we always keep in mind the usual picture: Turing machines [32] are introduced via their tables which specify elementary operations like moving the head to the left or to the right, erasing or printing a symbol on the tape’s square under the head, and so on. However it isn’t clear how to reduce that naïve, even if fertile, picture, to an axiomatic treatment that encompasses all of its clear-cut features, and so we will try at first to give an alternative, strictly formal, definition for Turing machines via Diophantine equations, and we will use that picture to stress that we are dealing with a formal construction that moreover has a clearcut, naïve interpretation.

Let $p_U(k, x_0, x_1, x_2, \dots, x_k)$ be a universal Diophantine polynomial [18, 36] which we suppose to be fixed. We define the partial recursive function $\{e\}$ of Gödel number e that acts on natural number m as its input and has natural number n as its output as:

Definition 2.4.

$$[\{e\}(m) = n] \leftrightarrow_{\text{Def}} [\exists x_0, \dots, x_k \in \omega p_U(\langle e, m, n \rangle, x_0, x_1, \dots, x_k) = 0]. \square$$

(ω is the set of natural numbers, $\langle x, y, z \rangle = \langle x, \langle y, z \rangle \rangle$ and $\langle \dots \rangle$ is the usual

pairing function; for the computation of e and the construction of p_U see [18, 32, 36].)

Partial recursive function (or the corresponding Turing machine) $\{e\}$ is given by the preceding definition. Of course there is a relation between (the abstract objects) Turing machines and concrete objects of our real world such as computers (which can be best seen as realizations of finite automata) but we restrict our attention to the mathematical object characterized above.

Axiomatic treatment

We use a Suppes predicate to axiomatize the theory of Diophantine equations within some formal system that includes Peano Arithmetic, such as Peano Arithmetic (PA) itself, or Zermelo–Fraenkel set theory with or without the Axiom of Choice (ZFC , ZF). For the explicit construction see [12]; other examples of the application of Suppes predicates in the axiomatic treatment of several theories are to be found in [11].

A first caveat

The next discussion and examples [14, 17] ponder well-known phenomena from the theory of fast-growing recursive functions in the light of our main theme, that is, axiomatization of CS (computer science) within ZFC or any other standard axiomatics for set theory, based on the classical predicate calculus, with a recursively enumerable set of theorems, and which extend enough of arithmetic to adequately handle Definition 2.4.

The first example we now give starts from a well-known property of intuitively total recursive functions when framed within usual axiomatic systems: *not every intuitively total recursive function can be proved so within a given axiomatic framework*. That is to say, the axiomatic system isn't able to clearly "see" as total functions some recursive functions that are intuitively total.

The point is [13, 21]:

Remark 2.5. Suppose that we are given a prescription so that, for any integer n , we can compute a finite set of numbers S_n . Then put: $F^*(n) = \max S_n + 1$. Most professional mathematicians would immediately agree that F^* is both computable and total. But is that really so? \square

Can we construct that function within ZFC ? We surely can give a Turing program, or a recursive-function definition for it relative to the recursivity of S_n — but we won't then be able to prove that similar functions are total, in the general case, as it is well-known.

Let's take a closer look at a function F as the ones naïvely described in Remark 2.5:

Definition 2.6. For each n , $F(n)$ is the sup of those $\{e\}(k)$ such that:

1. $k \leq n$.
2. $[\text{Pr}_{\text{ZFC}}([\forall x \exists z T(e, x, z)])] \leq n$.

That is, there is a proof of $\{e\}$ is total in ZFC whose Gödel number is $\leq n$. (For sentence φ , $[\varphi]$ is its Gödel number; T is Kleene's predicate.) \square

F^* in Remark 2.5 is always intuitively total, and so is the particular version F in Definition 2.6. The fact that F is intuitively total follows from the formal Löwenheim–Skolem theorem (or “reflection theorem” in the sense of set theory — [28], p. 133ff — not to be confused with Feferman's Reflection Principles; see [7, 23, 24]), which ensures that, for some particular value of n we will be able to construct a model for a collection of cardinality $\leq n$ of sentences of ZFC , and so to compute $F(n)$. The analogy is: we can *individually* check that, for each natural number n , $F(n)$ exists and can be computed. However we cannot “join” all those results together to show that F is total.

Granted those remarks:

Proposition 2.7. We can explicitly compute a Gödel number e_F so that $\{e_F\} = F$. \square

F is such that:

Proposition 2.8. The formal sentence $\forall x \exists z T(e_F, x, z)$ cannot be proved or disproved from the axioms of ZFC , supposed consistent, that is,

$$\forall x \exists z T(e_F, x, z)$$

is independent of the ZFC axioms. \square

(We can naïvely suggest that Proposition 2.8 results from the same kind of obstacle that hinders the extension of the Löwenheim–Skolem theorem, from finite subsets of axioms of ZFC to the whole theory.) The preceding propositions for ZFC can be extended to any theory like our S .

So F cannot be proved to be total in ZFC . In other words: this means that even if F is intuitively total (that is to say, it holds of the standard model for arithmetic), there must exist a model for ZFC with a nonstandard part where $\{F\}$ isn't total is verified. (This result originates in Kleene's 1936 paper [26]; see also [27], p. 257.)

We will now require Feferman's Σ_1 -soundness reflection principle. Naïvely, a system S is Σ_1 -sound if, given a proof of $\exists x P(x)$, where P is a predicate with values in the natural numbers, then there is some natural number n so that $P(n)$ is true. That concept was formalized by Feferman as one of his reflection principles [23]. Then there is an important result that connects F to Feferman's Σ_1 -reflection principles [23, 24]:

Recall that a reflection principle in Feferman's sense [7, 23, 24] has (roughly) two aspects:

- If we prove in S that there is the Gödel number of a proof of φ , then φ .
- If we obtain for each n , $\text{Pr}_S[\varphi(n)] \rightarrow \varphi(n)$, then we can collect all those under a universal quantifier, that is, there is a form of the ω -rule at work here.

Σ_1 -soundness is such a principle restricted to Σ_1 -sentences φ . The chief result is:

Proposition 2.9. $S \vdash [\text{F is total}] \leftrightarrow [S \text{ is } \Sigma_1\text{-sound}]$. \square

For the proof see Section 6.

3 Can we handle arbitrary infinite sets of poly machines in ZFC ?

The next two examples have been first discussed in [14, 17]. We may argue that the results about F deal with fast-growing functions, which are objects quite far from the everyday realm of programs and concrete computers. So we move on to another example, that bears on concepts related to the P vs. NP question.

The main point here is: the concept of an *infinite set of poly machines* may be quite difficult to handle within an axiomatic system, even a strong one like ZFC .

A *poly machine*, or a polynomially time-bounded Turing machine is a total Turing machine M_m that on a binary input x of length $|x|$ outputs a binary word y after less than $p_m(|x|)$ operation steps, p_m a polynomial kept fixed for M_m . Consider the following example, which is an application of the previous results (the trick used is due to [6]):

Example 3.1. Let $\langle m, a, b \rangle$ denote a Turing machine M_m coupled to a clock $C_{(a,b)}$ — another Turing machine — that stops M_m after it executes $|x|^a + b$ cycles over input x of length $|x|$. We note that pair $\langle M_m, C_{(a,b)} \rangle$. We agree that if clock $C_{(a,b)}$ interrupts the operation of M_m then M_m outputs 0 and stops.

Consider the set $A = \{(m, F(a), F(b)) : m, a, b \in \omega\}$, F as in Definition 2.6. Each individual machine $\langle m, F(a), F(b) \rangle$, m, a, b integers like $0, 1, 2, \dots$, in it is certainly a poly machine, but we cannot prove in ZFC that the whole set only contains poly machines. It is in fact undecidable: “ A is a set of poly machines” holds of the standard integers, but doesn’t hold in some models for ZFC with a nonstandard arithmetic part. \square

We have shown:

Proposition 3.2. The sentence “ A is a set of poly machines” is independent of ZFC , supposed consistent. \square

Now we may ask: does it help if we add some strong axiom — here noted X — to ZFC , so that the resulting theory $ZFC + X$, supposed consistent, proves the

consistency of *ZFC* itself? (Think of X , say, as some large cardinal axiom, for instance.) No. For there is a set A' so that:

Proposition 3.3. The sentence “ A' is a set of poly machines” is independent of $ZFC + X$, supposed consistent.

Proof: It suffices to obtain the function F' for $ZFC + X$, according to the blueprint in Definition 2.6. \square

This example is crucial, because among other consequences, any (tentative) proof of the hypothesis $P < NP$ must include a step that says, “for every poly machine, it will give a wrong answer at least once”, and a sentence like “ A is a set of poly machines” is undecidable within consistent *ZFC*, even if we can pick up infinitely many of its elements and individually show each of them to be a poly machine. This raises the following question: will any system like the S we have been considering here (see Remark 2.1), be able to prove $P < NP$? The preceding discussion is quite sobering.

4 More examples of incompleteness for computer science in S

We now quote two recent versions of some results by Hartmanis and Hopcroft [17, 25]. They start from a formal theory that:

- includes set theory (more precisely, they ask that the theory be of “sufficient power to prove the basic theorems of set theory”).
- Also the theory must allow for predicate symbols P, Q, \dots
- It has a recursively enumerable set of theorems.
- Its theorems are “intuitively true”. This is too strong and also vague for the whole of set theory with the axiom of choice — for instance, is the Banach–Tarski theorem intuitively true? So, we take this third requirement to be the *arithmetically soundness* condition, that is, S must have a model with standard arithmetic.

We can agree that S (see Definition 2.1) adequately fits the picture. We now endow this version of S with the ι -symbol.

Remark 4.1. We notice here the importance of theories like S : most results like those described here extend to any such theory, or, in other words, are sort of insensitive to the axiomatic framework where they are constructed, but for the fact that one requires arithmetic to formulate them. \square

According to Hartmanis and Hopcroft [25]:

- The first undecidability result in [25] has to do with the BGS relativization result [6]. The BGS result says that there are recursive oracles $A, B, A \neq B$, so that one has (for the relativized versions) $P^A = NP^A$ and $P^B < NP^B$. Hartmanis and Hopcroft show that there is an oracle C so that the assertion $P^C = NP^C$ is undecidable with respect to the axioms of S . Their proof is by a diagonal argument; we have used here an alternative, quite general argument.
- Then they show that there is an algorithm A (a Turing machine) of which it is true that for input x it runs in time x^2 , but so that the formal version of the sentence “ $A(x)$ runs in time $t_A < 2^x$ ” is undecidable in S .

We present here a general argument for these results which is based on a version of Rice’s theorem in fragments of set theory with the ι -symbol (which we suppose to be available) and which stems from an idea first used in da Costa and Doria [11, 16]. More precisely:

Remark 4.2. For consistent S , let $\text{Consis } S$ be the usual formal sentence that asserts the consistency of S ; $S \not\vdash \text{Consis } S$ and $S \not\vdash \neg \text{Consis } S$. Let ξ, ζ be terms in the language of S , so that for some predicate P in the language of S , $S \vdash P(\xi)$ while $S \vdash \neg P(\zeta)$. Then:

$$\lambda = \iota_x \{ [\text{Consis } S \wedge x = \xi] \vee [\neg \text{Consis } S \wedge x = \zeta] \}.$$

$S \not\vdash \lambda = \xi$ and $S \not\vdash \lambda = \zeta$, but if $\mathbf{N} \models S$ and \mathbf{N} has a standard arithmetic part, then $\mathbf{N} \models \lambda = \xi$. Moreover, $S \not\vdash P(\lambda)$ and $S \not\vdash \neg P(\lambda)$, while $\mathbf{N} \models P(\lambda)$. \square

- For the first result, put oracles A, B as $\xi = A$ and $\zeta = B$. Then C :

$$C = \iota_x \{ [\text{Consis } S \wedge x = A] \vee [\neg \text{Consis } S \wedge x = B] \}$$

is proved to be a recursive oracle in S , but $S \not\vdash C = A$ and $S \not\vdash C = B$. So, $S \not\vdash P^C = NP^C$ and $S \not\vdash P^C < NP^C$.

- For the second result, if P is a polynomial Turing machine, and E is an exponential Turing machine, then:

$$M = \iota_x \{ [\text{Consis } S \wedge x = P] \vee [\neg \text{Consis } S \wedge x = E] \}$$

is such that S proves M to be a total Turing machine which has an exponential time bound which cannot be improved in S , but such that it is true of \mathbf{N} that it is time-polynomial.

We may also use the term:

$$\lambda = \iota_x \{ [x = \xi \wedge \beta = 0] \vee [x = \zeta \wedge \beta = 1] \}.$$

β [16] is an algebraic expression which can be explicitly constructed such that $S \vDash \beta = 0$ and $S \vDash \beta = 1$, while $\beta = 0$ holds of the standard model for arithmetic; see the references for details.

Remark 4.3. The second result can be easily extended to a result by Loo [29]: that there is a Turing machine of arbitrarily large complexity when “seen” from within the standard model, but which is only polynomial in an adequate nonstandard model. An analogous example was discussed in 2002 by one of the authors [20]. An alternative argument that clarifies the above discussion stems from the following observation: consider

$$p(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots + \frac{x^K}{K!},$$

where K is a nonstandard positive integer.

If $\mathbf{N} \models S$, where \mathbf{N} has a standard arithmetic part, while also $\mathbf{M} \models S$, and \mathbf{M} has a nonstandard arithmetic part, with $K \in \mathbf{M}$, K nonstandard, then we notice that the restriction $p(x)|\mathbf{N}$ is an exponential, while $p(x) \in \mathbf{M}$ is a polynomial. \square

5 Function F and function G

Now recall that a consistent system S is ω -consistent if S doesn't simultaneously prove $\exists x P(x)$ and $\neg P(0), \neg P(1), \neg P(2), \dots$. ω -consistency implies consistency, while the converse isn't true. Consider PA which is supposed consistent, and add a new symbol ζ to its alphabet, with the (new) predicate $N(x)$ which should intuitively mean, “ x is a natural number”.

Then, by the usual compactness argument the system $\text{PA} + \zeta \neq 0 + \zeta \neq 1 + \dots + N(\zeta)$ is consistent. Also, from $N(\zeta) \rightarrow \exists x N(x)$ one proves $\exists x N(x)$. We then get the ω -inconsistent system $\text{PA} + \zeta \neq 0 + \zeta \neq 1 + \dots + \exists x N(x)$, which however is consistent.

Our F gives rise to such a situation:

Remark 5.1. Notice that given function F as above in Definition 2.6, we can see that $[\text{F is total}]$ is independent of S , but the structure of that function for the nonstandard models where $\neg[\text{F is total}]$ holds isn't clear at all. We immediately get an ω -inconsistency result, for

$$S + \neg[\text{F is total}] + \exists y (F(0) = y) + \exists y (F(1) = y) + \dots$$

is a consistent theory. Thus Σ_1 -unsoundness (or equivalently $\neg[\text{F is total}]$) implies ω -inconsistency. \square

We can say that for very large values of its argument, F ceases to be defined in such ω -inconsistent systems.

Construction of function \mathbf{G}

But nevertheless we can ask: is there a partial recursive function \mathbf{G} with a clearly infinite domain and yet with a behavior similar to that of \mathbf{F} ? We answer this question in the affirmative through the following:

Proposition 5.2. There is a partial recursive function \mathbf{G} so that:

1. If $\mathbf{N} \models S$ and has standard arithmetic, then $\mathbf{N} \models \mathbf{F} = \mathbf{G}$.
2. $S + [\mathbf{F} \text{ is total}] \vdash [\text{Domain } \mathbf{G} \text{ is infinite and } \mathbf{G} \text{ is increasing}]$.
3. $S + [\mathbf{F} \text{ is total}] \not\vdash [\mathbf{G} \text{ is total}]$ and $S + [\mathbf{F} \text{ is total}] \not\vdash \neg[\mathbf{G} \text{ is total}]$.
4. If $S_{(\alpha)}$, $\alpha > 1$ an ordinal, is in the Turing–Feferman hierarchy over S extended by Σ_1 –soundness reflection principles, then we can choose α as high as we wish in that hierarchy $\leq \omega$, so that $S_{(\alpha)} \vdash [\mathbf{G} \text{ is total}]$, but such that $S_{(\beta)} \not\vdash [\mathbf{G} \text{ is total}]$, $\beta < \alpha$.

Proof: As we have supposed that S has a model \mathbf{N} with standard arithmetic, then so does theory $S' = S + [\mathbf{F} \text{ is total}]$:

- We can explicitly obtain a Diophantine polynomial $p(x_1, \dots, x_k)$ so that:
 1. $S' \not\vdash [\forall x_1, \dots, p(x_1, \dots) > 0]$ while $\forall x_1, \dots, p(x_1, \dots) > 0$ holds of \mathbf{N} .
 2. $S' \not\vdash [\exists x_1, \dots, p(x_1, \dots) = 0]$ and for some model \mathbf{M} with nonstandard arithmetic, $\mathbf{M} \models \exists x_1, \dots, p(x_1, \dots) = 0$.
- Define: $\zeta = a_1$, given that for some (necessarily nonstandard) model \mathbf{M} ,

$$\mathbf{M} \models [p(a_1, a_2, \dots, a_k) = 0].$$

(We must impose some uniqueness condition on $\langle a_1, \dots, a_k \rangle$.)

By construction, for all models of S' , $[\mathbf{F} \text{ is total}]$.

Remark 5.3. We now informally describe an algorithm for the function we are looking for. Let \mathbf{h} be such that S proves \mathbf{h} to be total and strictly increasing. Put, for \mathbf{G} :

- If $m < \zeta$, $\mathbf{G}(m) = \mathbf{F}(m)$.
- If $m > \zeta$ and $n = \mathbf{h}(m)$, then $\mathbf{G}(n)$ is undefined.
- If $m > \zeta$ and $n \neq \mathbf{h}(m)$, then $\mathbf{G}(n) = \mathbf{F}(n)$.

More precisely: If $m \in \{x \mid x \in \mathfrak{I}(h) \text{ and } x \geq \zeta\}$, then $G(n)$ is undefined. Or, If $m \in \{x \mid x \notin \mathfrak{I}(h) \text{ and } x \geq \zeta\}$, then $G(n) = F(n)$. \mathfrak{I} is of course the image of ...

So \mathbf{G} will always have an infinite domain whenever $[\mathbf{F} \text{ is total}]$ holds. And due to the dependence of \mathbf{G} on ζ , it cannot be proved total even in a strong theory such as $S + [\mathbf{F} \text{ is total}]$, that is, $S + [S \text{ is } \Sigma_1\text{-sound}]$. For the last assertion, given the hierarchy over S plus Σ_1 -sound reflection principles, it suffices to choose an adequate p so that $\forall x, \dots p(x, \dots) > 0$ is only proved by $S_{(\alpha)}$, but not by any $S_{(\beta)}$, $\beta < \alpha \leq \omega$. (For the Turing-Feferman theorem see [7, 23, 24].) \square

We can give a formal expression for \mathbf{G} with the help of the ι -symbol, which we then add to our formal background — S and the required extensions.

- We can write down the algorithm for a Turing machine that never stops over any input n . Let $\{e_0\}$ be that machine.
- Consider \mathbf{h} as in Remark 5.3. \mathbf{h} is a S -provably total recursive function. Then write:

$$\begin{aligned} \mathbf{H}(n) = \iota_x \{ & [(x = \mathbf{F}(n)) \wedge (n \notin \text{Image}(\mathbf{h}))] \vee \\ & \vee [(x = \{e_0\}(n)) \wedge (n \in \text{Image}(\mathbf{h}))] \}. \end{aligned}$$

Therefore S proves that such a function equals \mathbf{F} (which is total for all models of S), but for the values of \mathbf{h} , where it is undefined. As the image of \mathbf{h} doesn't exhaust all of ω , its complement is infinite, and \mathbf{H} will have an infinite domain.

- Now consider the first Diophantine polynomial p above together with $\zeta = a_1$. If we write $\forall x p > 0$ for $[\forall x_1, \dots p(x_1, \dots) > 0]$, and similarly $\exists x p = 0$, then consider the next expression that can be seen to be in the language of S (possibly extended for definitions):

$$\begin{aligned} \mathbf{G}(n) = \iota_x \{ & [(\forall x p > 0) \wedge (x = \mathbf{F}(n))] \vee \\ & \vee [(\exists x p = 0) \wedge [(n < \zeta) \wedge (x = \mathbf{F}(n))] \vee \\ & \vee [(n \geq \zeta) \wedge (x = \mathbf{H}(n))] \}. \end{aligned}$$

ζ is as above. This construction originates in the (symbolic) form $(1 - \beta)\mathbf{X} + \beta\mathbf{X}'$, which was used in [11]; for the expression above see the construction in [16], p. 34. It is also akin to the construction in the result known as Kreisel's Lemma [17]. \square

6 Σ_1 -soundness and F

The next results show that F sort of codes, or represents, deep facts about the structure of the axiomatic system where it is defined. Namely, if it is total, then the axiomatic system T w.r.t. which F is defined, is consistent. We also show that F is total if and only if T is Σ_1 -sound.

Remark 6.1. The *local reflection principle* $\text{Rfn}(T)$ for theory T is:

$$\text{Pr}_T([\varphi]) \rightarrow \varphi,$$

(that is, if there is a proof for φ , we actually find it in T); and the (first) *uniform reflection principle*, $\text{RFN}(T)$ is:

$$[\forall x \text{Pr}_T([\varphi(\dot{x})])] \rightarrow [\forall x \varphi(x)],$$

all φ with only x free, and \dot{x} standing for the x that can be represented by actual constants in T . This means that once one can list instances $\varphi(0), \varphi(1), \dots$ (which are derivable due to the first supposition in the Reflection Principle) for all nameable n , then a restricted application of the ω -rule leads to the principle.

For Σ_1 -soundness one restricts φ to all $\exists x \varphi(x)$, φ primitive recursive; the corresponding restricted reflection principle is noted $\text{RFN}_{\Sigma_1}(T)$. \square

For T as PA, ZFC, and first-order extensions as considered here:

Corollary 6.2. $T \vdash \text{RFN}_{\Sigma_1}(T) \rightarrow \text{Consis}(T)$.

Proof:

- Suppose $T \vdash \neg(0 = 0)$.
- Therefore, $T \vdash \text{Pr}_T[\neg(0 = 0)]$.
- Given Σ_1 -soundness, as this is a trivial Σ_1 -sentence,

$$\text{Pr}_T[\neg(0 = 0)] \rightarrow [\neg(0 = 0)].$$

- By contraposition,

$$\{\neg[\neg(0 = 0)]\} \rightarrow \neg\text{Pr}_T[\neg(0 = 0)].$$

- Or, $(0 = 0) \rightarrow \neg\text{Pr}_T[\neg(0 = 0)]$.
- However, $T \vdash (0 = 0)$. Then follows:
- $\neg\text{Pr}_T[\neg(0 = 0)]$. A contradiction.

(This proof is due to N. C. A. da Costa.) \square

Σ_1 -soundness is equivalent to [F is total]

We argue here for *ZFC*, since we want to have as much “elbow room” as possible. However we could have argued for any theory like the one we have denoted by *S*. If we allow for the abuse of language that subsumes the infinitely many sentences of the Reflection Principle in our formulation as a single one in the statement of our result:

Lemma 6.3. $ZFC \vdash [\text{F is total}] \leftrightarrow [ZFC \text{ is } \Sigma_1\text{-sound}]$.

Proof: Recall that $[\text{F is total}] \leftrightarrow \forall x \exists z T(e_F, x, z)$, where — here — T is Kleene’s T predicate and e_F is a Gödel number for F .

(\Leftarrow). We first prove: assuming $RFN_{\Sigma_1}(ZFC)$, then $\forall x \exists z T(e_F, x, z)$. Given the (recursively enumerable) infinite set of conditions $RFN_{\Sigma_1}(ZFC)$, for T we get:

$$[\forall x \text{Pr}_{ZFC}([\exists z T(e_F, \dot{x}, z)])] \rightarrow [\forall x \exists z T(e_F, x, z)].$$

Now, for each *ZFC* constant \dot{n} we have that:

$$ZFC \vdash [\exists z T(e_F, \dot{n}, z)].$$

Then there are proofs of each of these sentences, for each \dot{n} . Therefore we conclude: $[\forall x \text{Pr}_{ZFC}([\exists z T(e_F, \dot{x}, z)])]$, as \dot{x} only ranges over the constants. From the corresponding restriction of $RFN_{\Sigma_1}(ZFC)$ we conclude that:

$$[\forall x \exists z T(e_F, x, z)]$$

holds, by modus ponens.

(\Rightarrow). For the converse: given $\forall x \exists z T(e_F, x, z)$, we deduce $RFN_{\Sigma_1}(ZFC)$. (We will have to deduce each instance of the Reflection Principle, for each 1-variable $\exists z \psi(z, x)$, ψ primitive recursive — for the meaning of \mathbf{z} see below.)

Recall that given each e we can explicitly construct a Diophantine polynomial

$$p_e(\langle x, y \rangle, z_1, z_2, \dots, z_m)$$

so that:

$$[\forall x \exists z T(e, x, z)] \leftrightarrow [\forall x \exists y, z_1, \dots, z_m p_e(\langle x, y \rangle, z_1, \dots, z_m) = 0].$$

(Of course $[\exists z_1, \dots, z_m p_e(\langle x, y \rangle, z_1, \dots, z_m) = 0] \leftrightarrow [y = \{e\}(x)]$. Since we aren’t using a universal equation, m may depend on e .) We will abbreviate the z_1, \dots, z_m by \mathbf{z} .

1. Now, if $[F \text{ is total}]$, then, for each $n \in \omega$ we have that $F(n)$ is the sup of all $\{e\}(k)$ so that $k \leq n$ and:

$$[\text{Pr}_{\text{ZFC}}([\forall x \exists z T(e, x, z)])] \leq n.$$

2. Since n is explicitly given, it is a bound on the Gödel number of the proof. Therefore we can also obtain a $n' > n$ so that:

$$[\text{Pr}_{\text{ZFC}}([\exists z T(e, x, z)])] \leq n'.$$

3. Or, for another n'' , $[\text{Pr}_{\text{ZFC}}([\exists y, \mathbf{z} p_e(\langle x, y \rangle, \mathbf{z}) = 0])] \leq n''$.

This follows from:

- Every recursive function $\{e\}(n) = m$ can be represented by a predicate $F_e(n, m)$.
(The algorithm to produce F_e given e is in Machtey and Young [30], p. 126ff.)
- Given $F_e(n, m)$ we can use the procedure described in Davis' paper on Hilbert's 10th Problem [18] to get a polynomial p_e out of F_e .

4. Since n'' is explicitly given, we can then recover proofs in *ZFC* of:

$$[\exists y, \mathbf{z} p_e(\langle x, y \rangle, \mathbf{z}) = 0],$$

all e under the specified conditions.

5. We then establish that *ZFC* proves, for all such e ,

$$\text{Pr}_{\text{ZFC}}([\exists y, \mathbf{z} p_e(\langle x, y \rangle, \mathbf{z}) = 0]) \rightarrow [\exists y, \mathbf{z} p_e(\langle x, y \rangle, \mathbf{z}) = 0].$$

6. We now add the universal quantifier for x , and as it distributes over \rightarrow ,

$$[\forall x \text{Pr}_{\text{ZFC}}([\exists y, \mathbf{z} p_e(\langle x, y \rangle, \mathbf{z}) = 0])] \rightarrow [\forall x \exists y, \mathbf{z} p_e(\langle x, y \rangle, \mathbf{z}) = 0].$$

7. This will of course also hold for (due to the implication's properties):

$$[\forall x \text{Pr}_{\text{ZFC}}([\exists y, \mathbf{z} p_e(\langle x, y \rangle, \mathbf{z}) = 0])] \rightarrow [\forall x \exists y, \mathbf{z} p_e(\langle x, y \rangle, \mathbf{z}) = 0].$$

8. Finally the $\exists \mathbf{w} p_e$ provide an enumeration of all Σ_1 -relations in *ZFC* of interest for Σ_1 -soundness. Notice that in *ZFC*, f and F recursive,

$$[f \text{ is total}] \leftrightarrow \{[F \text{ is total}] \rightarrow [F \text{ dominates } f]\}.$$

9. To show that the enumeration is exhaustive: suppose that for some p.r. ψ one has:

$$\forall x \text{Pr}_{\text{ZFC}}(\exists y \psi(\dot{x}, y)),$$

and that moreover the following sentence is proved:

$$[\forall x \text{Pr}_{\text{ZFC}}(\exists y \psi(\dot{x}, y))] \rightarrow \forall x \exists y \psi(x, y).$$

10. Put $f_\psi(x) = \min_y \psi(x, y)$. Then:

$$[\forall x \exists y (f_\psi(x) = y)] \leftrightarrow [\forall x \exists y \psi(x, y)].$$

11. That is, f_ψ is *ZFC*-provably total recursive, and therefore falls into the preceding case. \square

Remark 6.4. The preceding result shows the essential inner connections between the inability of an axiomatic theory like S to “see” the totality of functions like F and beyond, and the usual formal sentences that assert the consistency of S itself. That is to say, the obstructions we have to face when trying to prove the totality of F in S have to do with the impossibility of proving the consistency of S itself with its own tools.

The proof of the preceding result for PA can be found in the original Paris–Harrington paper [31]. \square

7 Paraconsistent Turing machines and quantum computation

We will just sketch the main concepts and ideas here, and leave details to be checked in the references. The main contention of this paper is that the way we conceive computation theory, or perhaps more specifically, the way we conceive computation procedures, turns out to be logic-relative and logic-dependent. The preceding examples were intended to stress that fact. We now turn to a different, potentially fruitful, approach.

A new model of *paraconsistent Turing machines* (**PTMs**) and its relation to quantum computation was studied in [3], where one shows how the classical Turing machines model can be enhanced by the change of the underlying logic from classical logic to a paraconsistent one. By conveniently interpreting **PTMs** as standard models of quantum computation (quantum Turing machines and the equivalent quantum circuits), some algorithms surprisingly close in spirit to the classical Turing machine formulation, but possessing quantic efficiency, can be designed to solve the so-called *Deutsch’s problem* and *Deutsch–Jozsa problem*.

In a general form, such problems (firstly introduced in [19]) consist in determining, for a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ whether f is *constant* $f(x) = f(y)$ for all vectors $x, y \in \{0, 1\}^n$ or *balanced* (the number of vectors $x \in \{0, 1\}^n$ such that $f(x) = 0$ is equal to the number of vectors $x \in \{0, 1\}^n$ such that $f(x) = 1$).

In case $n = 2$ there are obviously two constant functions and two balanced functions; classically we need at least two steps to evaluate f two times (in the entry 0 and in the entry 1) to determine if f is constant or balanced. Quantically, however, we can simultaneously evaluate $f(0)$ and $f(1)$ by means of a single application of a quantum operator that evaluates the f , and determine if f is constant or balanced in a single step. (See e.g. [10], or [3] and [4] for detailed explanations.)

In the general case, a quantum circuit is able to solve deterministically the Deutsch-Jozsa problem by performing a single application of a quantum operator, while the classical view requires (in the worst case) $2^{n-1} + 1$ steps. The classical algorithm has exponential complexity, while the quantum algorithm to solve the same problem has polynomial complexity.

But what is crucial for our arguments here is that standard models of quantum computation (quantum Turing machines and quantum circuits) can be simulated by paraconsistent Turing machines, as we shall see in the next section.

From classical Turing machines to quantum Turing machines

The achievement of this insight into relating quantum computation with logic-dependent notions of computation as the **PTMs** inaugurated the main idea that computation may be advantageously seen as *logic-relative*.

Although the first definition of **PTMs** is due to [1] and [5] and has been better developed in logical terms in [2], the potentialities of paraconsistent models of computation as simulating quantum computing were only fully perceived in [3].

Already in the original definition of what became known as a Turing machine (see [37]), the idea of the distinction between *deterministic Turing machines (DTMs)* and *non-deterministic Turing machines (NDTMs)* was present.

However, outputs of **NDTMs** cannot subsist simultaneously without leading to deductive trivialization of the underlying theories, since classical deduction lumps together contradictions (spawned by the non-determinism) and deductive triviality.

This is unfortunate from the point of view of a serious theory of quantum computation, if it intends to be a theory of computation based upon basilar principles of quantum mechanics such as superposition of states, entangled states and interference.

In 1994, Peter Shor proposed a quantum algorithm (using the model of quantum circuits) devoted to factoring numbers in polynomial time (see [34] and [35]), a problem of crucial importance for cryptography for which no classical algorithm with polynomial complexity is known. Since Shor's algorithm the research in quantum computing grew drastically, and there is a widespread need for alternative models of quantum computation.

Paraconsistent machines

The idea of one such quantum device is thus simply to allow for ambiguous situations (perplexing as they can be from the classical standpoint) wherein the machine

simultaneously executes several instructions, engendering a multiplicity of states, a multiplicity of positions and a multiplicity of symbols in certain cells of the tape.

The strategy to avoid deductive collapsing consists in changing the underlying logic behind such theories from classical to paraconsistent logic (leaving the axioms intact), avoiding trivialization and leading to new Turing machine models able to interpret and to take profit of the consequences of such theories.

In principle many solutions can be chosen, selecting underlying logics from the wide family of propositional paraconsistent logics known as ‘logics of formal inconsistency’ (**LFIs**) (cf. [8]). **LFIs** are paraconsistent logics that internalize the metatheoretical notions of consistency and inconsistency at the object language level, and are also characterized by preserving the positive fragment of propositional classical logic.

In [3] the first-order paraconsistent logic $LF11^*$ was chosen (treated in detail in [9], see also [8]). Baked up by such a careful logic which, it is opportune to clarify here, provides robust reasoning over uncertain and contradictory theories but *does not* beget any logical contradiction, we may now define **PTMs**:

Definition 7.1. The **paraconsistent Turing machines** are Turing machines defined by means of finite collections of instructions such that:

- We allow for contradictory instructions (viz. instructions the same initial symbols $q_i s_j$ are allowed);
- The machine executes simultaneously all possible instructions in the face of ambiguity. That procedure engenders multiple states as well as a multiplicity of positions and as multiplicity of symbols in some cells of the tape;
- Outputs are to be kept stored for further consideration;
- If the computation halts, each tape cell may contain multiple symbols and any choice of these symbols represents an output of the computation.

Control of unicity versus multiplicity conditions are the essential ingredients in the simulation of the quantum algorithms that solve Deutsch’s and Deutsch–Jozsa’s problems via **PTMs**, which are shown to simulate quantum algorithms preserving efficiency.

The symbols $^\circ$ and $^\bullet$ in the language of the logic $LF11^*$ are used to indicate respectively unicity (consistency) and multiplicity (inconsistency) conditions. These symbols will be written after the first symbol of the instruction, if the condition is on the state, or must be written after the second symbol of the instruction, if the condition is on the reading symbol. For example, instructions of the form $i_k = q_1^\circ s_1^\bullet xy$ will be executed in situations where the machine is in precisely the state q_1 (unicity required), and where one of the reading symbols is s_1 (multiplicity permitted).

In this way **PTMs** are expressed within first-order paraconsistent theories, and the most relevant cases of quantum parallelism can be simulated by **PTMs** by interpreting current states, positions and symbols on the tape as observables in a quantum system. From this perspective, **PTMs** emulate quantum Turing machines (**QTM**s). Constructive proofs of the following theorem are given in [3] and [4]:

Proposition 7.2. There exist paraconsistent Turing machines that solve deterministically any instance of the Deutsch-Jozsa problem in polynomial time.

Nevertheless such **PTM** models do not simulate quantum Turing machines and quantum circuits in full, as they do not allow an utterly adequate simulation of entangled states. Indeed, the construction of different models of **PTMs** which are able to simulate entangled states is a challenging task. This can be done, but the task now requires a non-adjunctive first-order paraconsistent logic (i.e., a logic where the inference from α and β to $\alpha \wedge \beta$ is not guaranteed) rather than $LFI1^*$, as proposed in [3] and [4]. This logic feature would ensure that the execution of distinct simultaneous instructions will be kept apart, permitting a logic control on the entangled configurations of a **QTM**. This fact — together with our examples above which show that highly complex machines in the standard model become even polynomial in non-standard extensions — confirms the view that computational models may be profitably approached as logic-relative devices.

8 Comments

Which is the natural axiomatic setting for Computer Science? That setting is the non recursive theory of all true arithmetical sentences, noted **T**. We can build **T** in several reasonable ways:

- $\mathbf{T} = \text{PA} + \text{Shoenfield's recursive } \omega\text{-rule [23, 24].}$
- $\mathbf{T} = \text{PA} + \text{all naively total recursive functions.}$
- $\mathbf{T} = \text{PA} + \text{a succession of naively total recursive functions so that for any such function } \mathbf{g} \text{ there is an } \mathbf{f} \text{ in the succession so that } \mathbf{f} \text{ dominates } \mathbf{g}.$
- $\mathbf{T} = \cup_{\alpha} \text{PA}_{\alpha}$, where PA_{α} is an iterate in the Turing–Feferman [7, 23, 24] progression of theories.
- $\mathbf{T} = \cup_{\alpha} \text{PA}_{\alpha}$, where now each PA_{α} is as in Turing’s original progression of theories [23], plus some axiom that reproduces the jump [32] within each component theory.

(Roughly what happens here is: the Turing progression proves all true Π_1 -sentences. These include all non halting instances of the Halting Problem. If we add those as an oracle to PA, plus the jump, we get all true sentences in the arithmetic hierarchy.)

All theories in those examples are nonrecursive, but they can be seen from the perspective of their “component” or iterate theories, which are, each one, recursive theories. So, the fact that \mathbf{T} is the natural setting for \mathbf{CS} is not as bad as it seems — the whole domain is non-recursive, but we can split it up into an infinite staircase of well-behaved theories.

Nonstandard stuff

And how about nonstandard stuff, like those that appear in our examples? How about logic-dependent models of computation devices? Being nonstandard, like being imaginary or being non-Euclidean, is a relative, historical, context-dependent concept. Take imaginary and complex numbers: electrical engineering is unconceivable without them. Non-Euclidean geometry led to today’s gravitation theory. We are still waiting for interpretations of non-Cantorian set theories, and of “fake” [33] spacetimes, but there is no reason not to suppose that reasonable ones will eventually be given.

The same applies to all our exotic, non-orthodox theories of Turing machines, including those sketched in the present paper. For mathematics deals with concrete objects, even if under the disguise of high, lofty abstraction. Someday someone will find a sensible, reasonable, concrete interpretation for all the unorthodox examples we have offered here.

9 Acknowledgments

WAC acknowledges support from FAPESP Thematic Research Project ConsRel grant 2004/14107-2 and from a CNPq research grant, and also wishes to acknowledge discussions with J. C. Agudelo. FAD thanks support from the Academia Brasileira de Filosofia and its chairman Prof. J. R. Moderno. The second author also wishes to acknowledge discussions and exchanges on the matters dealt with N. C. A. da Costa and M. Guillaume; he however regretfully notes that no Brazilian federal research grant supported his research efforts.

BIBLIOGRAPHY

- [1] J. C. Agudelo, *Máquinas de Turing paraconsistentes: algunas posibles definiciones y consecuencias*, monografía de especialización en Lógica y Filosofía. Universidad EAFIT, Colombia (2003), available at <http://sigma.eafit.edu.co:90/asicard/archivos/mtps.ps.gz>.
- [2] J. C. Agudelo, *Da Computação Paraconsistente à Computação Quântica*, M. Sc. dissertation, Unicamp (2006).
- [3] J. C. Agudelo and W. A. Carnielli, “Quantum algorithms, paraconsistent computation and Deutsch’s Problem”, *Proceedings of the 2nd Indian International Conference on Artificial Intelligence*, ed. B. Prasad, 1609–1628 (2005).
- [4] J. C. Agudelo and W. A. Carnielli, “Quantum computation via paraconsistent computation”, arXiv:quant-ph/0607100 v1 (2006).
- [5] J. C. Agudelo and A. Sicard, “Máquinas de Turing paraconsistentes: una posible definición”, *Matemáticas: Enseñanza Universitaria* **XII**, 2, 37–51 (2004). Available at <http://revistaerm.univalle.edu.co/Enlaces/volXII2.html>.

- [6] T. Baker, J. Gill and R. Solovay, “Relativizations of the $P = ?NP$ question”, *SIAM J. Comp.* **4**, 431–442 (1975).
- [7] L. Beklemishev, “Provability and reflection”, Lecture Notes for ESSLLI’97 (1997).
- [8] W. A. Carnielli, M. E. Coniglio and J. Marcos, “Logics of formal inconsistency”, in D. Gabbay and F. Günthner, *Handbook of Philosophical Logic*, Kluwer (2005).
- [9] W. A. Carnielli, J. Marcos and S. de Amo, “Formal inconsistency and evolutionary databases”, *Logic and Logical Philosophy* 115–152 (2000).
- [10] I. L. Chuang and M. A. Nielsen, *Quantum Computation and Quantum Information*, Cambridge University Press (2000).
- [11] N. C. A. da Costa and F. A. Doria, “Undecidability and incompleteness in classical mechanics”, *Int. J. Theor. Phys.* **30**, 1041–1073 (1991).
- [12] N. C. A. da Costa and F. A. Doria, “Suppes predicates and the construction of unsolvable problems in the axiomatized sciences”, in P. Humphreys, ed., *Patrick Suppes, Scientific Philosopher*, II, 151–191 Kluwer (1994).
- [13] N. C. A. da Costa and F. A. Doria, “Consequences of an exotic formulation for $P = NP$ ”, *Applied Mathematics and Computation* **145**, 655–665 (2003); also “Addendum”, *Applied Mathematics and Computation* **172**, 1364–1367 (2006).
- [14] N. C. A. da Costa and F. A. Doria, “On set theory as a foundation for computer science”, *Bulletin of the Section of Logic*, (University of Łódź) **33**, 33–40 (2004).
- [15] N. C. A. da Costa, F. A. Doria and E. Bir, “On the metamathematics of P vs. NP ”, *Applied Mathematics and Computation* **189**, 1223–1240 (2007).
- [16] N. C. A. da Costa and F. A. Doria, “Computing the future”, in K. Vela Velupillai, ed., *Computability, Complexity and Constructivity in Economic Analysis*, Blackwell (2005).
- [17] N. C. A. da Costa, F. A. Doria and E. Bir, “On the metamathematics of the P vs. NP question”, to appear in *Applied Mathematics and Computation* (2007). (Available online from ScienceDirect.)
- [18] M. Davis, “Hilbert’s Tenth Problem is unsolvable”, in *Computability and Unsolvability*, Dover (1982).
- [19] D. Deutsch, “Quantum theory, the Church-Turing principle and the universal quantum computer”, *Proc. R. Soc. Lond-A* **400**, 97–117 (1985).
- [20] F. A. Doria, “Metamathematics of P vs. NP ”, talk at the Suppes Fest, Fed. University at Santa Catarina, Florianópolis (Brazil), April (2002).
- [21] F. A. Doria, “Informal vs. formal mathematics”, *Synthèse*, to appear (2007). (Available online from SpringerLink.)
- [22] R. L. Epstein and W. A. Carnielli, *Computability: Computable Functions, Logic and the Foundations of Mathematics, with the timeline Computability and Undecidability*, 2nd ed., Wadsworth/Thomson Learning, Belmont CA (2000).
- [23] S. Feferman, “Transfinite recursive progressions of axiomatic theories”, *J. Symbolic Logic* **27**, 259–316 (1962).
- [24] T. Franzen, “Transfinite progressions: a second look at completeness”, *Bull. Symbolic Logic* **10**, 367–389 (2004).
- [25] J. Hartmanis and J. Hopcroft, “Independence results in computer science”, *SIGACT News*, 13, Oct. Dec. (1976).
- [26] S. C. Kleene, “General recursive functions of natural numbers”, *Math. Annalen* **112**, 727–742 (1936).
- [27] S. C. Kleene, *Mathematical Logic*, Wiley (1967).
- [28] K. Kunen, *Set Theory*, North Holland (1983).
- [29] K. Loo, “Internal Turing machines”, arXiv:math-ph/0407056 v2 (2004).
- [30] M. Machtey and P. Young, *An Introduction to the General Theory of Algorithms*, Elsevier North-Holland (1978).
- [31] J. Paris and L. Harrington, “A mathematical incompleteness in Peano arithmetic”, in J. Barwise, ed., *Handbook of Mathematical Logic*, North-Holland (1977).
- [32] H. Rogers Jr., *Theory of Recursive Functions and of Effective Computability*, reprint, MIT Press (1992).
- [33] A. Scorpan, *The Wild World of 4-Manifolds*, AMS (2005).

- [34] P. W. Shor, “Algorithms for Quantum Computation: Discrete Log and Factoring”, *Proc. 35th Symposium on Foundations of Computer Science*, 124–134, IEEE Press (1994).
- [35] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”, *SIAM J. Comput.* **26**, 1484–1509 (1997).
- [36] C. Smorýnski, *Logical Number Theory, I*, Springer (1991).
- [37] A. M. Turing, “On computable numbers, with an application to the Entscheidungsproblem”, *Proc. London Math. Society* **50**, 230 (1937).