Evaluating the Complexity of Mathematical Problems. Part 1

C. S. Calude[1] and Elena Calude[2]
[1]University of Auckland, NZ
[2]Massey University at Albany, NZ

CDMTCS-343
December 2008

Evaluating the Complexity of Mathematical Problems. Part 1*

Cristian S. Calude[1], **Elena Calude**[2]
[1]University of Auckland, New Zealand
`www.cs.auckland.ac.nz/~cristian`
[2]Massey University at Albany, New Zealand
`http://www.massey.ac.nz/~ecalude`

May 18, 2009

Abstract

In this paper we provide a computational method for evaluating in a uniform way the complexity of a large class of mathematical problems. The method, which is inspired by NKS[1], is based on the possibility to completely describe complex mathematical problems, like the Riemann hypothesis, in terms of (very) simple programs. The method is illustrated on a variety of examples coming from different areas of mathematics and its power and limits are studied.

1 Introduction

Evaluating (or even guessing) the degree of difficulty of an open problem (or of a solved problem but before seeing its solution) is notoriously hard not only for beginners, but also for the most experienced mathematicians.

Can one develop a (uniform) method to evaluate, in some objective way, the difficulty of a mathematical problem? The question is not trivial because mathematical problems can be so diverse—the Mathematics Subject Classification (MSC2000), based on two databases, Mathematical Reviews and Zentralblatt MATH, contains over 5,000 two-, three-, and five-digit classifications, cf. [33]—and

*This work was supported in part by The Andrea von Braun Foundation, Munich, under the grant for "Artistic Forms and Complexity".

[1]New Kind of Science [32].

there is no clear indication that all, or most, or even a large part of mathematical problems have a kind of "commonality" allowing a uniform evaluation of their complexity. How could one compare a problem in number theory with a problem in complex analysis or a problem in algebraic topology?

Surprisingly enough, such a "commonality" exists for many mathematical problems and one of them (which will be discussed in this paper) is based on the possibility of expressing the problem in terms of (very) simple programs reducible to a (natural) question in theoretical computer science, the so-called *halting problem* [7]. As a consequence, a uniform approach for evaluating the complexity of a large class of mathematical problems can be (and was) developed.

The paper is structured as follows. In the next section a series of interesting mathematical problems will be presented and analysed in order to find their "commonality": the infinity of primes, Goldbach's conjecture and other problems in number theory, the pigeonhole principle, Hilbert's tenth problem, the four colour theorem, Riemann hypothesis, Collatz and palindrome conjectures. We will show that all these problems are closely related to the halting problem. The third section discusses the most (in)famous problem in theoretical computer science, the halting problem. Section 4 presents the method for evaluating the complexity. Section 5 gives an example of a class of problems to which the proposed method applies, namely the class of finitely refutable problems, and section 6 examines the power and limits of the proposed method. We finish the paper with a few concluding remarks.

2 Some interesting mathematical problems: what do they have in common?

In this section we discuss some interesting, solved or open, mathematical problems searching for their possible "common computational structure".

2.1 The infinity of primes

Euclid is credited with the first proof that the set of primes is infinite: there is no largest prime as much as there is no largest natural number. The typical argument—a reasoning by absurdity—runs as follows. Let us suppose that the set of primes is finite, say $\{p_1, p_2, p_3, \ldots, p_n\}$. Construct the number $q = p_1 p_2 p_3 \cdots p_n + 1$. If q is a prime, then we have a new prime as $q > p_i$ for all $1 \leq i \leq n$. If q is not a prime, then (by the prime factoring theorem) it must be divisible by a prime $r < q$. But r cannot be any p_i in our original exhaustive list of primes because dividing q by p_i produces the remainder 1. As a consequence, r is a new prime as well. In both cases we have found a prime not in the original list, a contradiction.

Wittgenstein[2] criticised Euclid's proof on its "totality" nature because:[3]

> *In mathematics we must always be dealing with systems, and not with totalities.*

Instead of a proof of the existence of an infinity of primes based on deduction from certain formal or informal assumptions, Wittgenstein called for the construction of "a formal expression that proves the infinity of primes by its syntactical features."

Such a proof can be easily obtained by rephrasing Euclid's argument in terms of a formal or informal (computer) program Π_{primes} which generates the sequence of primes in increasing order: the infinity of primes is equivalent with the property of Π_{primes} to continue indefinitely (never stop).[4]

2.2 Goldbach's conjecture and other problems in number theory

Goldbach's conjecture, which is part of Hilbert's eighth problem [16], states that

> *all positive even integers greater than two can be expressed as the sum of two primes.*

The conjecture was tested up to 10^{18}, cf. [23].

Applying the same idea as for the infinity of primes one can write a (computer) program Π_{Goldbach} which just enumerates all positive even integers greater than two and for each of them checks the required property; the program Π_{Goldbach} stops if and only if it finds a counter-example for Goldbach's conjecture. Re-phrased: Π_{Goldbach} never stops if and only if Goldbach's conjecture is true.

The same approach works for many problems in number theory, in particular for Fermat's last theorem: the program Π_{Fermat} systematically generates all 4-tuples of positive integers greater than 3, (n, x, y, z), and stops when it finds the first 4-tuple for which $x^n + y^n = z^n$.

Can the same method be applied to the conjecture that

> *there are infinitely many Mersenne[5] primes?*

---

[2]Who advocated a form of anti-platonism by rejecting the interpretation of mathematical propositions in terms of propositions which are capable of being true or false in correspondence to reality, cf. [20].

[3]Cited from [20], p. 64.

[4]Although we do not subscribe to Wittgenstein's philosophy of mathematics we acknowledge the importance of his preference for process vs. function.

[5]Mersenne numbers are numbers of the form $2^n - 1$. Currently only 46 Mersenne primes are known and the largest is $2^{43112609} - 1$. The conjecture is believed to be true because the harmonic series diverges.

Or to the twin prime conjecture[6]

> *there are infinitely many primes p such that $p+2$ is also prime?*

It is clear that Goldbach's conjecture and Fermat's last theorem are statements of the form $(\forall n)\, P(n)$, where P is a computable predicate. The last two conjectures have a more complicated structure, i.e. they can be written in the form $(\forall N)\, (\exists n > N)\, P'(n)$, where P' is a computable predicate. A program generating more and more natural numbers satisfying the twin prime conjecture may not stop either because there are infinitely many pairs of primes $p, p+2$ or because there are only finitely many primes p such that $p+2$ is also prime!

So, directly, the last two conjectures cannot be represented by the halting property of an associated program. It is an *open question* whether the last conjectures can be represented in the form $(\forall n)\, P(n)$, where P is a computable predicate. Still, can they be described in terms of the halting status of some program? A positive answer will be given in section 6.

2.3 The pigeonhole principle

The statement

> *if $n > m$ pigeons are put into m pigeonholes, there's a hole with more than one pigeon*

is called the pigeonhole principle or the Dirichlet principle.

A more formal statement is the following:

> *every function from a set of n elements into a set with $m < n$ elements is not injective.*

A program $\Pi_{\text{pigeonholeprinciple}}$ which for all n generates all functions from $\{1, 2, \ldots, n\}$ into $\{1, 2, \ldots, m\}$, for all $m < n$, and for each of them checks whether the function is injective will either find an injective function and stop, or will continue for ever. The validity of the pigeonhole principle is equivalent to the fact that the program $\Pi_{\text{pigeonholeprinciple}}$ never stops.

2.4 Hilbert's tenth problem

Solving algebraic equations using integer (or rational) constants in the domain of (positive) integers is an old mathematical activity. Some of these equations do not have solutions at all; others have finitely many solutions or infinitely many

[6]Believed to be true because of the probabilistic distribution of primes.

solutions. The equation $2x - 2y = 1$ has no integer solutions, the equation $5x = 10$ has a unique integer solution while the equation $7x - 17y = 1$ has infinitely many integer solutions.

A Diophantine equation[7] is an equation of the form $P = 0$ where P is a polynomial with integer coefficients. Fermat's equations $x^n + y^n = z^n$ for $n = 1, 2, \ldots$ are all Diophantine. To solve a given Diophantine equation $P = 0$ one has to determine whether the equation has solutions in the domain of (positive) integers, and, if it has, to find all of them.

The original formulation of Hilbert's tenth problem is:[8]

> 10. *Determining the solvability of a Diophantine equation. Given a Diophantine equation with any number of unknowns and with rational integer coefficients: devise a process, which could determine by a finite number of operations whether the equation is solvable in rational integers.*

Consider a parametric Diophantine equation

$$P(a_1, a_2, \ldots, a_n, x_1, x_2, \ldots, x_{m+1}) = 0, \tag{1}$$

where $a_1, a_2, \ldots, a_n$ are parameters and $x_1, x_2, \ldots, x_{m+1}$ are unknowns. Fixing values for parameters results in a particular Diophantine equation. For example, in the parametric Diophantine equation $(a_1 - a_2)^2 - x_1 - 1 = 0$ the parameters are a_1, a_2 and x_1 is the only unknown. If we put $a_1 = 1, a_2 = 0$ we get the Diophantine equation $x_1 = 0$; if we take $a_1 = a_2 = 0$ we get the Diophantine equation $x_1 + 1 = 0$.

Given a parametric Diophantine equation (1) one can construct a program Π_{P} which, beginning with the input $a_1, a_2, \ldots, a_n$, will eventually halt if and only if the equation (1) has a solution in the unknowns $x_1, x_2, \ldots, x_{m+1}$. The program Π_{P} systematically generates all vectors with $m + 1$ integers $(i_1, i_2, \ldots, i_{m+1})$, checks for each of them whether $P(a_1, a_2, \ldots, a_n, i_1, i_2, \ldots, i_{m+1}) = 0$ and stops when the first solution is found.

Can we make the program Π_{P} "independent" of P in the sense that P appears as an input of the program? The answer is affirmative: one can construct a program Π_{H10P} which, given an arbitrary Diophantine equation (without parameters, i.e. $n = 0$) $P = 0$, eventually stops if and only if the equation $P = 0$ has a solution. Can we decide in a finite amount of time whether the program Π_{H10P} eventually halts? The answer is negative as it is well-known, cf. [21]. The core argument is based on the fact that every computably enumerable set of natural numbers can be represented in the form $\{n : P(n, x_1, x_2, \ldots, x_m) = 0$ has a solution in the non-negative integers unknown $x_1, x_2, \ldots, x_m\}$, cf. [12].

[7]Named after Diophantus of Alexandria (III century AD).

[8]For the original statement in German see [34].

It is interesting to note that the counterpart of Hilbert's tenth problem for real unknowns, that is, given an equation of the form $P(x_1, x_2, \ldots, x_m) = 0$ where P is a polynomial with integer coefficients (like in the classical case) but $x_1, x_2, \ldots, x_m$ are real unknowns, is decidable: there is a program which decides in a finite amount of time whether the equation has a solution in the domain of reals. Indeed, the decision problem is solved by Sturm method [30] for $m = 1$; Tarski's method [29] works for any number of unknowns.[9] Of course, no program can in general compute exactly some solutions, even if their number is known, because solutions can be irrational.[10] This leads us to the following problem for standard Diophantine equations.

Fix a Diophantine equation

$$D(n, x_1, x_2, \ldots, x_m) = 0, \tag{2}$$

and then consider the following two questions:

- for a fixed $n = n_0$, does the equation $D(n_0, x_1, x_2, \ldots, x_m) = 0$ have a solution?
- for a fixed $n = n_0$, does the equation $D(n_0, x_1, x_2, \ldots, x_m) = 0$ have an infinity of solutions?

Both questions are undecidable for *some* equations (2). For *each* equation (2) the information contained in the sequence of k answers to the yes/no questions "does the equation $D(n_0, x_1, x_2, \ldots, x_m) = 0$ have a solution, for $n = 1, 2, \ldots, k$?" contains only $\log k$ bits of information (knowing how many equations have solutions is enough to determine exactly which equations have solutions). This information can be substantially compressed. However, for *some* equations (2) the information contained in the sequence of k answers to the yes/no questions "does the equation $D(n_0, x_1, x_2, \ldots, x_m) = 0$ have infinitely many solution, for $n = 1, 2, \ldots, k$?" contains about k bits of information, i.e. this information cannot be algorithmically compressed. See more in [6].

2.5 The four colour theorem

The four colour theorem—first conjectured in 1853 by Francis Guthrie—states that every plane separated into regions may be coloured using no more than four colours in such a way that no two adjacent regions receive the same colour. Two regions are called adjacent if they share a border segment, not just a point; regions must be contiguous, i.e. the plan has no exclaves.

[9] This shows that extrapolating computational facts from positive integers to reals is not always possible.

[10] However, solutions can be effectively approximated up to any precision.

In graph-theoretical terms, the four colour theorem states that the vertices of every planar graph can be coloured with at most four colours so that no two adjacent vertices receive the same colour. Shortly, every planar graph is four-colourable.

The theorem was proved in 1977 [1, 2] (see also [31]) using a computer-assisted proof which consists in constructing a finite set of "configurations", and then prove that each of them is "reducible"—which implies that no configuration with this property can appear in a minimal counterexample to the theorem. Checking the correctness of the original proof is a very difficult task: it implies, among other things, checking the inputting of the descriptions of 1476 graphs, checking the correctness of the programs, proving the correctness of the compiler used to compile the programs, checking the degree of reliability of the hardware used to ran the programs.[11] Various partial independent verifications have been obtained[12] culminating with the formal confirmation announced in [28] which uses the equational logic program Coq.[13] The following part of the concluding discussion in [28] is relevant for the current status of the proof:

> *However, an argument can be made that our 'proof' is not a proof in the traditional sense, because it contains steps that can never be verified by humans. In particular, we have not proved the correctness of the compiler we compiled our programs on, nor have we proved the infallibility of the hardware we ran our programs on. These have to be taken on faith, and are conceivably a source of error. ... Apart from this hypothetical possibility of a computer consistently giving an incorrect answer, the rest of our proof can be verified in the same way as traditional mathematical proofs. We concede, however, that* verifying a computer program is much more difficult than checking a mathematical proof of the same length.[14]

A program $\Pi_{\text{fourcolourtheorem}}$ which systematically generates all planar graphs and checks for each of them whether it is colourable with four colours and stops when the first counter-example is found will never halt if and only if the theorem is true. However, this program will be quite long because testing the planarity of a graph is difficult. A better solution is to use the Diophantine representation of the four colour theorem proposed in [12]: a Diophantine equation

$$F(n, t, a, \ldots) = 0, \tag{3}$$

[11]This computer-assisted proof generated lots of mathematical and philosophical discussions around the notion of acceptable mathematical proof, see for example [5, 8, 9].

[12]It appears that there is no verification in its entirety.

[13]See [14] for a recent presentation of the formal proof.

[14]Our emphasis.

such that the equation (3) has no solution if and only if every planar graph can be coloured with at most four colours so that no two adjacent vertices receive the same colour. Based on the equation (3) one can write the program Π_{F} as in the previous section which can be taken as $\Pi_{\mathrm{fourcolourtheorem}}$.

Actually, it is better to use a pre-Diophantine representation given by the following conditions. Without restricting the generality we consider the maps T_n consisting of the points (x, y) such that $J(x, y) \leq Q = (n^2 + 3n)/2$, where J is Cantor's bijection $J(x, y) = ((x + y)^2 + 3x + y)/2$. Given a 4-colouring of T_n, $t_0, t_1, \ldots, t_Q$ there exist (and can be effectively computed) s, t such that for all $0 \leq i \leq Q$:[15]

$$t_i = rem(t, 1 + s(i + 1)).$$

In other words, the sequence $t_0, t_1, \ldots, t_Q$ can be coded by s and t.

Every sequence $u_0, u_1, \ldots, u_Q$ with $u_i < 4$ can be represented by some $u \leq R = (1 + 4(Q + 2)!)^{Q+1}$ such that

$$u_i = rem(u, 1 + 4(Q + 2)!(i + 1)).$$

Finally, there is a map (say T_n) which cannot be coloured in 4 colours if and only if the following condition is satisfied:

$$(\exists n, t, s)(\forall u \leq R)(\exists x, y)(x + y \leq n)[A \vee B],$$

where

$$A = u_{J(x,y)} \geq 4,$$

$$\begin{aligned} B = [&(t_{J(x,y)} = t_{J(x+1,y)} \wedge u_{J(x,y)} \neq u_{J(x+1,y)}) \\ &\vee (t_{J(x,y)} \neq t_{J(x+1,y)} \wedge u_{J(x,y)} = u_{J(x+1,y)}) \\ &\vee (t_{J(x,y)} = t_{J(x,y+1)} \wedge u_{J(x,y)} \neq u_{J(x,y+1)}) \\ &\vee (t_{J(x,y)} \neq t_{J(x,y+1)} \wedge u_{J(x,y)} = u_{J(x,y+1)})]. \end{aligned}$$

A simple inspection shows that the above condition is computable, so the four colour theorem is of the form $(\forall n)P(n)$, where P is a computable predicate.

[15]The integer remainder function is denoted by *rem*.

2.6 The Riemann hypothesis

The Riemann hypothesis is probably the most famous/important conjecture in mathematics. It appears in Hilbert's eighth problem [16]: the non-trivial complex zeros of Riemann's zeta function, which is defined for $Re(s) > 1$ by

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s},$$

lie exactly on the line $Re(s) = 1/2$.

According to Matiyasevich [21], p. 119–121, the negation of the Riemann hypothesis is equivalent to the existence of positive integers k, l, m, n satisfying the following six conditions (here $x \mid z$ means "x divides z"):

1. $n \geq 600$,
2. $\forall y < n\, [(y + 1) \mid m]$,
3. $m > 0\, \&\, \forall y < m\, [y = m \vee \exists x < n\, [\neg\, [(x + 1) \mid y]]]$,
4. $explog(m - 1, l)$,
5. $explog(n - 1, k)$,
6. $(l - n)^2 > 4n^2k^4$,

and $explog(a, b)$ denotes the predicate

$$\exists x\, [x > b + 1\, \&\, (1 + 1/x)^{xb} \leq a + 1 < 4(1 + 1/x)^{xb}].$$

An inspection of the above conditions shows that the Riemann hypothesis is of the form $\forall n, R(n)$, where R is a computable predicate. Hence, one can write a program Π_{Riemann} such that the Riemann hypothesis is false if and only if Π_{Riemann} halts.

2.7 Collatz and palindrome conjectures

When he was a student L. Collatz posed the following problem:[16] given any integer seed a_1 there exists a natural N such that $a_N = 1$, where

$$a_{n+1} = \begin{cases} a_n/2, & \text{if } a_n \text{ is even,} \\ 3a_n + 1, & \text{otherwise}. \end{cases}$$

There is a huge literature on this problem and various natural generalisations: see [19, 15, 11]. Erdös has said (cf. [19]) that

[16]Known as Collatz's conjecture, the Syracuse conjecture, the $3x + 1$ problem, Kakutani's problem, Hasse algorithm, or Ulam's problem.

Mathematics may not be ready for such problems.

Does there exist a program Π_{Collatz} such that Collatz's conjecture is false if and only if Π_{Collatz} halts?

First we note that a brute-force tester, i.e. the program which will enumerate all seeds and for each of them will try to find an iteration equal to 1, may never stop in two different cases: a) because the conjecture is indeed true, b) because for some specific seed a_1 there is no N such that $a_N = 1$. It is not clear how to differentiate these cases; even worse, it is not clear how to refute b) by a brute-force tester.

A simple non-constructive argument answering in the affirmative our question appears in [7]. Indeed, observe first that the set

$$Collatz = \{a_1 \,:\, a_N = 1, \text{ for some } N \geq 1\}$$

is computably enumerable. Collatz's conjecture requires to prove that *Collatz* contains indeed all positive integers.

If *Collatz* is not computable, then the conjecture is false, and any program which eventually halts can be taken as Π_{Collatz} as a) is ruled out. If *Collatz* is computable, then we can write a program Π_{Collatz} to find an integer not in *Collatz*: the conjecture is true if and only if Π_{Collatz} never stops.

Next we present the palindrome conjecture. The reverse (mirror) of a number is the number formed with the same decimal digits but written in the opposite order. For example, the mirror of 12 is 21, the mirror of 131072 is 270131, etc. Start with the decimal representation of a natural a, reverse the digits and add the constructed number to a; iterate this process till the result is a palindrome. Following [13] the palindrome conjecture states that for every natural a, a palindrome number will be obtained after finitely many iterations of the above procedure.

The same non-constructive argument used for Collatz's conjecture applies also to the palindrome conjecture: there exists a program $\Pi_{\mathrm{palindrome}}$ such that the palindrome conjecture is true if and only if $\Pi_{\mathrm{palindrome}}$ never stops.

Collatz and palindrome conjectures have the following general form. Let $a \in \mathbf{N}$ and let T be a computable function from naturals to naturals. The conjecture associated to (a, T) is: "for each $x \in \mathbf{N}, T^i(x) = a,$ for some $i > 0$".

Considering the set

$$B(a,T) = \{x \in \mathbf{N} \,:\, T^i(x) = a, \text{ for some } i > 0\}, \tag{4}$$

the conjecture associated to (a, T) becomes equivalent with the equality $B(a, T) = \mathbf{N}$. The argument used for Collatz's conjecture applies to this general case too, so one can prove in a non-constructive way the existence of a program $\Pi_{(\mathrm{a,T})}$ such that the conjecture associated to (a, T) is true if and only if $\Pi_{(\mathrm{a,T})}$ never stops.

3 The halting problem

In the previous sections the halting property of various programs repeatedly appeared. It is time to ask the question: can the halting problem be solved by a program? As all our programs have a very specific form—they have no input and each of them either stops (in which case the output is a natural number) or never stops—we will show (with a simple argument) that the halting problem, the problem whether such a program eventually stops, is unsolvable by any program. This means that there is no program **H** having the following three properties:

1. **H** accepts as input any program of the above type,
2. **H** eventually halts, and
3. **H** produces the output one if the input-program eventually stops or zero in case the input-program never stops.

Here is an information-theoretic analysis of the existence of the hypothetical program **H**. Assume that there exists a halting program **H** having the above three properties. Using **H** we construct the following (legitimate) program **P**:

1. read a natural N;
2. generate all programs up to N bits in size;
3. use **H** to check for each generated program whether it halts;
4. simulate the running of the remaining programs, and
5. output one plus the biggest value output by these programs.

The program **P** halts for every natural N. Indeed, the number of programs of less than N bits in size is finite, so by the assumption that **H** can decide the halting status of every program in a finite amount of time one can filter out all non-halting programs. The remaining ones (certainly, finitely many) can be run and after a finite (maybe very long) computation they will all halt and each will produce a natural number as output. Did we obtain a contradiction?

To answer the question we have to ask the right auxiliary question: How long is **P**? Answer: **P** is about $\log_2 N$ bits. Indeed, we need about $\log_2 N$ bits to code N in binary, and the rest of the program **P** is a constant, say c. Hence, the length of **P** is $\log_2 N + c$ bits.

Now observe that there is a big difference between the size of **P** and the size of the output produced by **P**. Indeed, for large enough N, **P** belongs to the set of programs having less than N bits because $\log_2 N + \mathrm{O}(1) < N$. Hence, in this case, **P** generates itself at some stage of the computation. As **P** always halts, the program **H** decides that **P** stops, so **P** is run as a part of the simulation (inside the computation of **P**) and produces a natural number as a result. But the program **P** itself will output a natural number which is different from every output produced by a simulated computation, in particular from the output produced by **P** itself, a contradiction.

The above proof (see [6] for more details) shows that in general there is no method, no uniform procedure, to test whether an arbitrary program eventually stops or not. Of course, this does not imply that for some (infinite) class of programs we cannot find a program deciding the halting status of those programs.[17]

What is the situation with the programs associated to the problems discussed before? Can we hope to decide for each of them whether it halts or not? For some programs, like Π_{Fermat}, we know the answer: the program never stops as certified by A. Wiles' proof of the Fermat's last theorem. For the program Π_{Riemann} the answer is not known. We [currently] cannot even write explicitly the program Π_{Collatz}.[18] For some programs Π the statement "Π halts" is independent of ZFC.[19]

4 A computational method for evaluating the complexity of mathematical problems

Let us return to Fermat's last theorem and to the fact that the theorem is equivalent with the statement "Π_{Fermat} never halts". We do not propose to prove Fermat's last theorem by showing that Π_{Fermat} never halts, but *to use the program* Π_{Fermat} *as a measure of complexity of Fermat's last theorem.*

How? Simply by counting the number of bits necessary to specify Π_{Fermat} in some fixed "universal formalism" (like a universal self-delimiting Turing machine [6]). Of course, there are many programs equivalent to Π_{Fermat}, so a natural way to evaluate the complexity is to consider the *smallest* such program.

The choice of the universal formalism used to code programs is irrelevant up to an additive constant, so if a problem is significantly more complex in some fixed formalism than another one, then it will continue to be more complex in any other formalism. However, the proposed measure is *uncomputable* (see [6]), so

[17] Actually, there are infinite sets of programs for which the halting problem is decidable, e.g. the class of primitive recursive programs (which are all total).

[18] We conjecture that the statement "Π_{Collatz} never stops" is *independent* of ZFC.

[19] Such a statement has to be true. A proof of independence is an alternative proof—admittedly not usual—of the truth of the statement.

we have to work with an upper bound on the size of a program "describing" the conjecture/problem/theorem.

In practice, to evaluate the complexity of a problem P we need to obtain effectively the program Π_{P} and to compute its size in bits: this gives an upper bound on the complexity of Π_{P}, hence on the difficulty of P. But, as we have seen with Collatz and palindrome conjectures, even this type of approximation may not be achievable in all cases: we cannot evaluate a bound on the difficulty of a problem P if we do not know at least one "explicit" program Π_{P}.

5 Finitely refutable problems

It is time to ask the question: what is the class of problems whose complexity/difficulty can be evaluated with the method proposed in the preceding section? We will not answer this question—which is *open*—but give an example of a large class of problems to which the method applies. With Pythagoras' dictum "all is number" as a guiding principle we will look at finite numerical tests.

Let $\mathbf{N}$ denote the set of positive integers and for every $k \in \mathbf{N}$ consider a predicate P on $\mathbf{N}$.

Consider the formula

$$f = Q_1 n_1\, Q_2 n_2\, \ldots\, Q_k n_k\, P(n_1, n_2, \ldots, n_k)$$

where $Q_1, Q_2, \ldots, Q_k \in \{\forall, \exists\}$ are quantifier symbols. In analogy with the arithmetic classes, we say that f is in the class $\hat{\Pi}_s$ or $\hat{\Sigma}_s$ if the quantifier prefix of f starts with $\forall$ or $\exists$, respectively, and contains $s-1$ alternations of quantifier symbols. When P is computable, then f is in Π_s or Σ_s, respectively. It is sufficient to consider only such formulæ f in which no two consecutive quantifier symbols are the same; in the sequel we make this assumption without special mention. With f as above, one has $s = k$.

As usual, with P as above, we write $P(n_1, \ldots, n_k)$ instead of $P(n_1, \ldots, n_k) = 1$ when $n_1, \ldots, n_k$ are elements of $\mathbf{N}$. Thus, $\neg P(n_1, \ldots, n_k)$ if and only if $P(n_1, \ldots, n_k) = 0$. Moreover, since we consider variable symbols only in the domain $\mathbf{N}$, if f is any formula in first-order logic, we write f is true instead of f is true in $\mathbf{N}$.

Let Γ_s be one of the classes $\hat{\Pi}_s$, $\hat{\Sigma}_s$, Π_s, and Σ_s. We refer to the task of proving or refuting a first-order logic formula as a *problem* and, in particular, to problems expressed by formulæ in Γ_s as *Γ_s–problems.*

We say that a problem is being *solved* if the corresponding formula is proved or disproved to be true, that is, if the truth value of the formula is determined. A problem is said to be *finitely solvable* if it can be solved by examining finitely many cases.

For example, consider the predicate

$$P(n) = \begin{cases} 1, & \text{if } n \text{ is even or } n = 1 \text{ or } n \text{ is a prime,} \\ 0, & \text{otherwise,} \end{cases}$$

that is, $P(n) = 0$ if and only if n is an odd number greater than 1 which is not a prime. Then the conjecture expressed by the formula $(\forall n)\, P(n)$ is finitely solvable; indeed, it is sufficient to check all n up to 10 to refute this conjecture.

Goldbach's conjecture is a Π_1–problem. To express it let $P_{\text{Goldbach}} : \mathbf{N} \to \{0, 1\}$ be such that

$$P_{\text{Goldbach}}(n) = \begin{cases} 1, & \text{if } n \text{ is odd or } (n \text{ is even and } n \text{ is the sum of two primes),} \\ 0, & \text{otherwise.} \end{cases}$$

Thus, $f_{\text{Goldbach}} = (\forall n)\ P_{\text{Goldbach}}(n)$ is true if and only if Goldbach's conjecture is true.

Similarly, the Riemann hypothesis is a Π_1–problem. By a result of [12], the Riemann hypothesis can be expressed in terms of the function $\delta_{\text{Riemann}} : \mathbf{N} \to \mathbf{R}$ defined by

$$\delta_{\text{Riemann}}(k) = \prod_{n<k} \prod_{j \le n} \eta_{\text{Riemann}}(j),$$

where

$$\eta_{\text{Riemann}}(j) = \begin{cases} p, & \text{if } j = p^r \text{ for some prime } p \text{ and some } r \in \mathbf{N}, \\ 1, & \text{otherwise.} \end{cases}$$

The Riemann hypothesis is equivalent with the assertion that for all $n \in \mathbf{N}$

$$\left(\sum_{k \le \delta_{\text{Riemann}}(n)} \frac{1}{k} - \frac{n^2}{2} \right)^2 < 36n^3.$$

If we set

$$P_{\text{Riemann}}(n) = \begin{cases} 1, & \text{if } \left(\sum_{k \le \delta_{\text{Riemann}}(n)} \frac{1}{k} - \frac{n^2}{2} \right)^2 < 36n^3, \\ 0, & \text{otherwise.} \end{cases}$$

then, $f_{\text{Riemann}} = (\forall n)\ P_{\text{Riemann}}(n)$ is true if and only if the Riemann hypothesis is true. Clearly, P_{Riemann} is decidable, therefore, the Riemann hypothesis is a Π_1–problem.

What is the "commonality" of all problems in classes $\hat{\Pi}_s$ and $\hat{\Sigma}_s$?

For $s \in \mathbf{N}$, let $\hat{\Gamma}_s$ denote any of $\hat{\Pi}_s$ and $\hat{\Sigma}_s$, and let Γ_s denote any of Π_s and Σ_s. Let

$$f = Q_1 n_1\, Q_2 n_2\, \ldots Q_s n_s\, P(n_1, n_2, \ldots, n_s)$$

with $s \in \mathbf{N}$, where $Q_1, Q_2, \ldots, Q_s$ are alternating quantifier symbols.

Following [10], we define a test set for f to be a set $T \subseteq \mathbf{N}^s$ such that f is true in $\mathbf{N}^s$ if and only if it is true in T. The problem f is finitely solvable if there is a finite test set for f. In [10] the following result was proved:

Every $f \in \hat{\Gamma}_s$ is finitely solvable.

In other words, a solution to each mass problem[20] in the above classes can be obtained by inspecting only finitely many instances of the problem. As we might expect, this fact cannot be used to obtain a uniform algorithmical way to solve these type of problems because the finite test set cannot be computed even for all problems in the class Π_1:

There is no constructive proof showing that every $f \in \Pi_1$ has a finite test set.

6 The power and the limits of the method

Our analysis gives a new method of comparing the difficulties of two or more finitely refutable problem. *The main obstacle is the non-computability of the measure,* cf. [6]. However, working with upper bounds one can obtain a practical method for evaluating the complexity which allows a relative ranking of problems.[21]

The method can be applied to every Π_1–problem. All problems discussed in section 2 can be analysed with this method (see [7]). Trying to reduce the lengths of programs is in general possible; of course, proving minimality is, in general, impossible cf. [6].

The method proposed is, certainly, not universal. Let us discuss here the class of Π_1–problems. Not every mathematical statement is a Π_1–problem. For instance, the twin prime conjecture—discussed in section 2.2—is not a Π_1–problem. Writing

$$P_{\mathrm{TP}}(n,m) = \begin{cases} 1, & m > n \text{ and } m \text{ and } m+2 \text{ are primes,} \\ 0, & \text{otherwise,} \end{cases}$$

this conjecture can be stated as

$$f_{\mathrm{TP}} = \forall n \, \exists m \, P_{\mathrm{TP}}(n,m).$$

The formula f_{TP} is in the class Π_2. Bennett [3] conjectured that most mathematical conjectures can be settled indirectly by proving stronger conjectures. For the

[20]A problem having an infinite number of instances/cases.

[21]In weighting the importance of computing the *exact value* of the complexity measure one should recall Knuth [18]: "premature optimization is the root of all evil" and Rabin [27] "we should give up the attempt to derive results and answers with complete certainty".

twin prime conjecture a stronger Π_1–problem is obtained as follows. Consider the predicate

$$P'_{\mathrm{T}}(n) = \begin{cases} 1, & \text{if there is } m \text{ with } 10^{n-1} \leq m \leq 10^n,\ m \text{ and } m+2 \text{ primes}, \\ 0, & \text{otherwise.} \end{cases}$$

Let $f'_{\mathrm{T}} = (\forall n) P'_{\mathrm{T}}(n)$. Thus, f'_{T} gives rise to a Π_1–problem and, if f'_{T} is true, then f_{T} is also true (but the converse is not necessarily true).

However, *there exists a program* Π_{TP} *such that the twin prime conjecture is true if and only if* Π_{TP} *never halts.* As in Collatz's case the argument is non-constructive and is based on the fact that the set

$$TP = \{n : \forall n\, \exists m > n \text{ such that } m \text{ and } m+2 \text{ are primes}\}$$

is computably enumerable.

The method depends on the chosen universal Turing machine, our working framework. Changing the universal machine will result in the change of the value of the complexity, but not in relative comparison between problems. The choice of the machine is irrelevant up to an additive constant, so if a problem is significantly more complex than another one with respect to a fixed universal machine, then it will continue to be more complex for any other machine. The method was used by relativising to the halting problem. The same method can be used by relativising to other unsolvable problems different from the halting problem (e.g. the totality problem). Going from the halting problem to the totality problem we increase the power of expression. For example to the conjecture associated to (a, T) (see the equation (4)) we can associate the program $\Gamma_{a,T}(x)$ defined by

$$\Gamma_{a,T}(x) = \min_i [T^i(x) = a].$$

The conjecture associated to (a, T) is true iff $\Gamma_{a,T}(x)$ is total. Writing the program for $\Gamma_{a,T}(x)$ is simple, but in the special cases of the Collatz/palindrome/twin prime conjectures writing the corresponding $\Pi_{a,T}$ program is problematic (we were able only to prove its existence).

The paper [17] discusses the possibility that one of the recently solved problems in topology—see [24, 25, 26], the Poincaré conjecture, is equivalent to the unsolvability of a Diophantine equation (see also [22]). If this would be true then our method would offer, at least in principle, an indication of the difficulty of this problem too.

7 Conclusions

We have presented a computational method to evaluate the complexity of mathematical problems. The method, which is inspired by NKS [32], is based on the possibility to completely describe complex mathematical problems, like the Riemann hypothesis, in terms of (very) simple programs.

If a mathematical problem, irrespective of its nature, can be equivalently expressed in terms of the property that a certain (associated) program eventually halts then the proposed method applies. For example, the method applies to every Π_1–problem. Specific instances of such problems are, for example, the Fermat last theorem, the Goldbach conjecture, the four colour problem, the Riemann hypothesis, the Hilbert 10th problem, the Collatz problem, the palindrome conjecture and the twin prime conjecture. As an illustration, *according to this complexity complexity measure, the Riemann hypothesis is more difficult*[22] *than the Goldbach conjecture* [7]. Although the method applies to both Collatz and twin prime conjectures, it is an open question whether one can *effectively evaluate/rank* the complexities of these problems.

Our method, which is a refinement below the first Turing degree, provides a total order in the class of finitely refutable problems. The difficulty of the problem is additive (modulo the constants involved due to the choice of the universal Turing machine). The same method can be used by relativising to other unsolvable problems different from the halting problem (e.g. the totality problem).

The scalability of the measure, both in terms of ordering, the role of the additive constants involved, and its relativisation to various unsolvable problems are open questions.

In the second part of this study we will present a formalism for evaluating in a uniform way the complexities of the problems discussed in this paper and a ranking of those problems will be presented.

Acknowledgement

We thank Professor John Casti for discussions and comments regarding the topic of this paper which improved our presentation. We also thank the anonymous referee for very helpful comments and suggestions.

References

[1] K. Appel, W. Haken, J. Koch. Every planar map is four colourable, I: Discharging, *Illinois J. Math.* 21 (1977), 429–490.

[2] K. Appel, W. Haken. Every planar map is four-colorable, II: Reducibility, *Illinois J. Math.* 21 (1977), 491–567.

[22] In fact, about twice as difficult.

[3] C. H. Bennett. Chaitin's Omega, in M. Gardner (ed.). *Fractal Music, Hypercards, and More ...*, W. H. Freeman, New York, 1992, 307–319.

[4] F. E. Browder (ed.). *Mathematical Developments Arising from Hilbert Problems*, Amer. Math. Soc., Providence, RI, 1976.

[5] A. S. Calude. The journey of the four colour theorem through time, *The NZ Math. Magazine* 38, 3 (2001), 2735.

[6] C. S. Calude. *Information and Randomness: An Algorithmic Perspective*, 2nd Edition, Revised and Extended, Springer-Verlag, Berlin, 2002.

[7] C. S. Calude, Elena Calude, M. J. Dinneen. A new measure of the difficulty of problems, *Journal for Multiple-Valued Logic and Soft Computing* 12 (2006), 285–307.

[8] C. S. Calude, Elena Calude, S. Marcus. Passages of proof, *Bull. EATCS* 84 (2004), 167–188.

[9] C. S. Calude, E. Calude, S. Marcus. Proving and Programming, in C. S. Calude (ed.). *Randomness & Complexity, from Leibniz to Chaitin*, World Scientific, Singapore, 2007, 310–321.

[10] C. S. Calude, H. Jürgensen, S. Legg. Solving finitely refutable mathematical problems, in C. S. Calude, G. Păun (eds.). *Finite Versus Infinite. Contributions to an Eternal Dilemma*, Springer-Verlag, London, 2000, 39–52.

[11] J. P. Davalan. $3x + 1$, Collatz, Syracuse problem, `http://pagesperso-orange.fr/jean-paul.davalan/liens/liens_syracuse.html`, (accessed on 30 November 2008).

[12] M. Davis, Y. V. Matijasevič, J. Robinson. Hilbert's tenth problem. Diophantine equations: Positive aspects of a negative solution, in F. E. Browder (ed.). *Mathematical Developments Arising from Hilbert Problems*, American Mathematical Society, Providence, RI, 1976, 323–378.

[13] J.-P. Delahaye. Déconcertantes conjectures, *Pour la science* 367, Mai (2008), 90–95.

[14] G. Gonthier. Formal proof—the four color theorem, *Notices of AMS* 55, 11 (2008), 1382–1393.

[15] R. K. Guy. Problem E16 in *Unsolved Problems in Number Theory.* Springer, New York, 2004 (third edition), 330–336.

[16] D. Hilbert. Mathematical Problems, *Bull. Amer. Math. Soc.* 8, 437–479, 1901–1902.

[17] M. Kim. *Why Everyone Should Know Number Theory*, manuscript `http://www.ucl.ac.uk/~ucahmki/numbers.pdf`, 1998 (accessed on 30 November 2008).

[18] D. E. Knuth. Structured programming with go to statements, *ACM Journal Computing Surveys*, Vol 6, No. 4, Dec. (1974), p.268.

[19] J. Lagarias. The $3x+1$ problem and its generalizations, *Amer. Math. Monthly* 92 (1985), 3–23.

[20] T. Lampert. Wittgenstein on the infinity of primes, *History and Philosophy of Logic* 29, 1 (2008), 63–81.

[21] Yu. V. Matiyasevich. *Hilbert's Tenth Problem*, MIT Press, Cambridge, MA, 1993.

[22] J. Manning. Algorithmic detection and description of hyperbolic structures on closed 3–manifolds with solvable word problem, *Geometry & Topology* 6 (2002), 1–26.

[23] T. Oliveira e Silva. Goldbach Conjecture verification, `http://www.ieeta.pt/~tos/goldbach.html`, 25 April 2008 (accessed on 30 November 2008).

[24] G. Perelman. Ricci Flow and Geometrization of Three-Manifolds, Massachusetts Institute of Technology, Department of Mathematics Simons Lecture Series, September 23, 2004 `http://www-math.mit.edu/conferences/simons` (accessed on 30 November 2008).

[25] G. Perelman. The Entropy Formula for the Ricci Flow and Its Geometric Application, November 11, 2002, `http://www.arxiv.org/abs/math.DG/0211159` (accessed on 30 November 2008).

[26] G. Perelman. Ricci Flow with Surgery on Three-Manifolds, March 10, 2003, `http://www.arxiv.org/abs/math.DG/0303109` (accessed on 30 November 2008).

[27] M. O. Rabin. in D. Sasha and C. Lazare (eds.). *Out of their Minds*, Copernicus, New York, 1995, p. 68.

[28] N. Robertson, D. P. Sanders, P. Seymour, R. Thomas. The Four Color Theorem, `http://www.math.gatech.edu/~thomas/FC/fourcolor.html` (accessed on 30 November 2008).

[29] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*, University of California Press, Berkeley and Los Angeles, 1951.

[30] E. W. Weisstein. Sturm function, *MathWorld–A Wolfram Web Resource*, `http://mathworld.wolfram.com/SturmFunction.html` (accessed on 30 November 2008).

[31] R. Wilson. *Four Colours Suffice*, Penguin, London, 2002.

[32] S. Wolfram. *A New Kind of Science*, Wolfram Research, 2002.

[33] 2000 Mathematics Subject Classification, MSC2000, `http://www.ams.org/msc`; MSC2000 is now being revised for use in 2010.

[34] `http://logic.pdmi.ras.ru/Hilbert10/stat/stat.htm` (accessed on 30 November 2008).