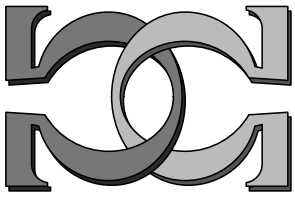
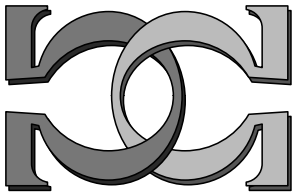
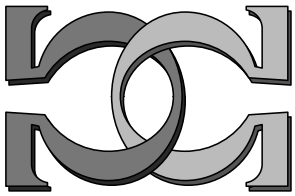
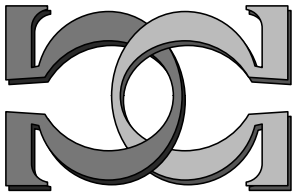


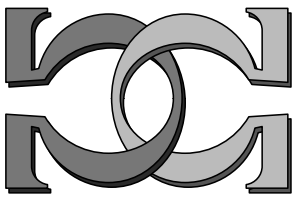




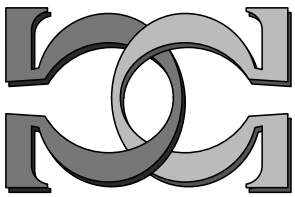
CDMTCS
Research
Report
Series



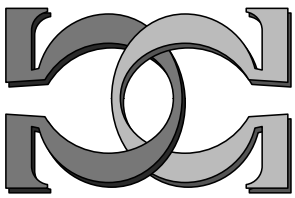
Fermat's Last Theorem and Chaoticity



Elena Calude
Massey University, NZ



CDMTCS-383
June 2010



Centre for Discrete Mathematics and
Theoretical Computer Science

Fermat's Last Theorem and Chaoticity

Elena Calude

Institute of Information and Mathematical Sciences, Massey University at Albany, Private Bag 102-904, North Shore MSC New Zealand

Abstract

Proving that a dynamical system is chaotic is a central problem in chaos theory [11]. In this note we apply the computational method developed in [4, 2, 3] to show that Fermat's last theorem is in the lowest complexity class $\mathfrak{C}_{U,1}$. Using this result we prove the existence of a two-dimensional Hamiltonian system for which the proof that the system has a Smale horseshoe is in the class $\mathfrak{C}_{U,1}$, i.e. it is not too complex.

1. Introduction

A system is chaotic if small differences in initial conditions could yield widely diverging outcomes; for such a system long-term prediction is in general impossible. Even deterministic systems whose dynamics are fully determined by their initial conditions, and no random elements are involved, can be chaotic [13, 8].

There are only few "bridges" between chaotic dynamical systems and complexity theories, in particular algorithmic information theory. Recently, [9] showed that in classical chaotic dynamics, typicality corresponds exactly to Schnorr randomness; this means that a chaotic system may produce a computable sequence of bits provided the initial point is suitable chosen, but this event has probability zero (the set of initial points can be infinite).

Virtually any "interesting" question about non-trivial dynamical systems is undecidable. Undecidability does not imply the impossibility to prove non-trivial properties of dynamical systems, in particular, chaoticity: it says that there is no general method, new specifically designed methods are required for different problems.

Email address: e.calude@massey.ac.nz (Elena Calude)

How difficult is to prove chaoticity? Building on results in [15, 12] in [7] a two-dimensional Hamiltonian system $\mathcal{H}$ was constructed with the property that in Zermelo-Fraenkel set theory with the Axiom of Choice (ZFC) proving the existence of a Smale horseshoe in $\mathcal{H}$ is equivalent to proving Fermat's last theorem. We say that "ZFC proves s" in case there is a proof in ZFC for s. We can now state more precisely the result described above:

Theorem 1. [7] *Assume* ZFC *is arithmetically sound (that is, any theorem of arithmetic proved by* ZFC *is true). Then, one can effectively construct in the formal language of* ZFC *the expression describing a two-dimensional Hamiltonian system* $\mathcal{H}$ *such that* ZFC *proves that* $\mathcal{H}$ *has a Smale horseshoe iff* ZFC *proves Fermat's last theorem.*

The choice of the Fermat's last theorem in [7] was motivated by the contrast between the short length of this elementary statement and the belief that any proof of the theorem has to be very complicated; this belief was indeed confirmed by the proof in [17].

Is the excruciating long proof of the Fermat's last theorem [17] a good indication that any proof that the corresponding two-dimensional Hamiltonian system is chaotic should be very complex, hence proving chaoticity is a difficult problem?

First, the fact that the known proof is complex is not a proof that every proof for Fermat's last theorem is complex.

Secondly, the result proven in [10]—which shows that in ZFC one can (effectively) find infinite sets of trivially true theorems which require as long proofs as the hardest theorems—indicates that the length of a proof may not be an adequate complexity measure for how complicated/deep the theorem is. In the words of Hartmanis [10]:

> In every formalization, infinite sets of trivial theorems will require very long proofs. ... It also gives a warning that a necessarily long proof in a formal system does not certify that the result is non-trivial.

Using the method developed in [4, 2, 3] we prove that Fermat's last theorem is in the class $\mathfrak{C}_{U,1}$, hence from this point of view its complexity is low.

The paper is organised as follows. In the next section we present a short proof for Theorem 1; in section 3 we briefly describe the complexity measure;

in section 4 we use this measure to obtain an upper bound on the complexity of Fermat's last theorem which shows that this statement is in the class $\mathfrak{C}_{U,1}$ and this bound is transferred to the statement regarding the chaoticity of a specific two-dimensional Hamiltonian system via the equivalence in Theorem 1; in section 5 we present some conclusions and an open problem.

2. A proof for Theorem 1

Following Richardson [15], Caviness [5] and Wang [16] we fix a positive integer n and denote by $\mathcal{E}_n$ a set of expressions representing real valued, partially defined functions of real variables and by $F(\mathcal{E}_n)$ the set of functions represented by the expressions in $\mathcal{E}_n$. By $e(x_1, x_2, \ldots, x_n)$ we denote the function represented by the expression $e \in \mathcal{E}_n$. We assume that $\mathcal{E}_n$ is generated by: i) the rational numbers and π as constant functions, ii) variables $x_1, x_2, \ldots, x_n$, iii) the functions $\sin, \mu, \mathfrak{s}$, and iv) the operations of addition, subtraction, multiplication and composition.[1] Here μ and $\mathfrak{s}$ are expressions denoting two unary functions such that $\mu(x) = |x|$, $\mathfrak{s}(x) = 1$, for all $x \neq 0$.

Let $\mathcal{P}_n$ be the set of exponential polynomials with integral coefficients in the variables $x_1, x_2, \ldots, x_n$.

Lemma 2. [15, 5, 16] *For every polynomial $P(x_1, x_2, \ldots, x_n) \in \mathcal{P}_n$ there exists $f_P(x_1, x_2, \ldots, x_n) \in F(\mathcal{E}_n)$ such that the following conditions are equivalent.*

1. *There are naturals $x_1, x_2, \ldots, x_n$ such that $P(x_1, x_2, \ldots, x_n) = 0$.*
2. *There are reals $x_1, x_2, \ldots, x_n$ such that $f_P(x_1, x_2, \ldots, x_n) = 0$.*
3. *There are reals $x_1, x_2, \ldots, x_n$ such that $f_P(x_1, x_2, \ldots, x_n) \leq 1$.*

We can now present the proof of Theorem 1.

Proof. Let $P(m, x, y, z) = (x+1)^{m+3} + (y+1)^{m+3} - (z+1)^{m+3}$. In view of Lemma 2 we have the following sequence of equivalences:

1. for every natural m, x, y, z, $P(m, x, y, z) \neq 0$,
2. for every natural m, $\neg(\exists$ natural x, y, z such that $P(m, x, y, z) = 0)$,
3. for every natural m, $\neg(\exists$ real x, y, z such that $f_P(m, x, y, z) = 0)$,

[1] In fact we can omit subtraction and π, cf. [16, 14].

4. for every natural m, $\neg(\exists$ real x, y, z such that $f_P(m, x, y, z) \leq 1)$,
5. for every natural m, $\neg(\exists$ real x, y, z such that $F_P(m, x, y, z) \leq 1)$,
6. for every natural m, $\neg(\exists$ real x, y, z such that $F_P(m, x, y, z) = 0)$,

where

$$F_P(m, x, y, z) = \mathfrak{s}(\mu(f_P(m, x, y, z) - 1) + f_P(m, x, y, z) - 1),$$

and

$$F_P(m, x, y, z) = 0 \text{ iff } f_P(m, x, y, z) \leq 1.$$

Further on, observe that $F_P(m, x, y, z) \in \{0, 1\}$, so if we denote by h, k the Hamiltonian for the free particle and the Hamiltonian for the two-dimensional system with a horseshoe in [12] (Example 4) then one can prove in ZFC that the Hamiltonian

$$\mathcal{H} = F_P \cdot h + (1 - F_P) \cdot k, \tag{1}$$

has a horseshoe iff one can prove in ZFC the condition "for every natural m, x, y, z, $P(m, x, y, z) \neq 0$", i.e. ZFC proves Fermat's last theorem.

□

3. A complexity measure

In this section we present a complexity measure [4, 2, 3] forΠ_1-statements (i.e. statements of the form "$\forall n\, Pred(n)$", where $Pred$ is a computable predicate) defined by means of register machine programs.

We use a fixed "universal formalism" for programs, more precisely, a universal self-delimiting Turing machine U. The machine U (which is fully described below) has to be *minimal* in the sense that none of its instructions can be simulated by a program for U written with the remaining instructions.

To everyΠ_1–problem $\sigma = \forall m P(m)$ we associate the algorithm$\Pi_P = \inf\{n : P(n) =$ false$\}$ which systematically searches for a counter-example for σ. There are many programs (for U) which implementΠ_P; without loss of generality, any such program will be denoted also byΠ_P. Note that σ is true iff $U(\Pi_P)$ never halts.

The complexity (with respect to U) of a Π_1–problem σ is defined by the length of the smallest-length program (for U) Π_P—defined as above—

where minimisation is calculated for all possible representations of σ as $\sigma = \forall n P(n)$:[2]

$$C_U(\sigma) = \min\{|\Pi_P| \, : \, \sigma = \forall n P(n)\}.$$

Because the complexity C_U is incomputable, we can work only with upper bounds for C_U. As the exact value of C_U is not important, following [3] we classifyΠ $_1$–problems into the following classes:

$$\mathfrak{C}_{U,n} = \{\sigma \, : \, \sigma \text{ is a}\Pi \, _1\text{–problem}, C_U(\sigma) \leq n \text{ kbit}^3\}.$$

We briefly describe the syntax and the semantics of a register machine language which implements a (natural) minimal universal prefix-free binary Turing machine U used for evaluating the complexity of Fermat's last theorem, aΠ $_1$–problem.

Any register program (machine) uses a finite number of registers, each of which may contain an arbitrarily large non-negative integer.

By default, all registers, named with a string of lower or upper case letters, are initialised to 0. Instructions are labeled by default with 0,1,2,...

The register machine instructions are listed below. Note that in all cases R2 and R3 denote either a register or a non-negative integer, while R1 must be a register. When referring to R we use, depending upon the context, either the name of register R or the non-negative integer stored in R.

=R1,R2,R3

If the contents of R1 and R2 are equal, then the execution continues at the R3-th instruction of the program. If the contents of R1 and R2 are not equal, then execution continues with the next instruction in sequence. If the content of R3 is outside the scope of the program, then we have an illegal branch error.

&R1,R2

The contents of register R1 is replaced by R2.

[2]For C_U it is irrelevant whether σ is known to be true or false. In particular, the program containing the single instruction halt is not aΠ $_P$ program, for any P.

[3]A kilobit (kbit or kb) is equal to 2^{10} bits.

+R1,R2

The contents of register R1 is replaced by the sum of the contents of R1 and R2.

!R1

One bit is read into the register R1, so the contents of R1 becomes either 0 or 1. Any attempt to read past the last data-bit results in a run-time error.

%

This is the last instruction for each register machine program before the input data. It halts the execution in two possible states: either successfully halts or it halts with an under-read error.

A *register machine program* consists of a finite list of labeled instructions from the above list, with the restriction that the halt instruction appears only once, as the last instruction of the list. The input data (a binary string) follows immediately after the halt instruction. A program not reading the whole data or attempting to read past the last data-bit results in a run-time error. Some programs (as the ones presented in this paper) have no input data; these programs cannot halt with an under-read error.

The instruction `=R,R,n` is used for the unconditional jump to the n-th instruction of the program. For Boolean data types we use integers 0 = `false` and 1 = `true`.

For longer programs it is convenient to distinguish between the main program and some sets of instructions called "routines" which perform specific tasks for another routine or the main program. The call and call-back of a routine are executed with unconditional jumps.

To compute an upper bound on the complexity of the Fermat last theorem we need to compute the size in bits of the programΠ_{Fermat}, so we need to uniquely code in binary the programs for U. To this aim we use a prefix-free coding as follows.

The binary coding of special characters (instructions and comma) is the following (ε is the empty string):

special characters	code	instruction	code
,	ε	+	111
&	01	!	110
=	00	%	100

Table 1

For registers we use the prefix-free regular code $\text{code}_1 = \{0^{|x|}1x \mid x \in \{0,1\}^*\}$. Here are the codes of the first 14 registers:[4]

register	code_1	register	code_1
R_1	010	R_8	0001001
R_2	011	R_9	0001010
R_3	00100	R_{10}	0001011
R_4	00101	R_{11}	0001100
R_5	00110	R_{12}	0001101
R_6	00111	R_{13}	0001110
R_7	0001000	R_{14}	0001111

Table 2

For non-negative integers we use the prefix-free regular code $\text{code}_2 = \{1^{|x|}0x \mid x \in \{0,1\}^*\}$. Here are the codes of the first 16 non-negative integers:

integer	code_2	integer	code_2	integer	code_2	integer	code_2
0	100	4	11010	8	1110010	12	1110110
1	101	5	11011	9	1110011	13	1110111
2	11000	6	1110000	10	1110100	14	111100000
3	11001	7	1110001	11	1110101	15	111100001

Table 3

The instructions are coded by self-delimiting binary strings as follows:

1. `&R1,R2` is coded in two different ways depending on R2:[5]

$$01\text{code}_1(\text{R1})\text{code}_i(\text{R2}),$$

where $i = 1$ if R2 is a register and $i = 2$ if R2 is an integer.

[4]The register names are chosen to optimise the length of the program, i.e. the most frequent registers have the smallest code_1 length.

[5]As $x\varepsilon = \varepsilon x = x$, for every string $x \in \{0,1\}^*$, in what follows we omit ε.

2. `+R1,R2` is coded in two different ways depending on R2:

$$111\text{code}_1(\text{R1})\text{code}_i(\text{R2}),$$

where $i = 1$ if R2 is a register and $i = 2$ if R2 is a non-negative integer.

3. `=R1,R2,R3` is coded in four different ways depending on the data types of R2 and R3:

$$00\text{code}_1(\text{R1})\text{code}_i(\text{R2})\text{code}_j(\text{R3}),$$

where $i = 1$ if R2 is a register and $i = 2$ if R2 is a non-negative integer, $j = 1$ if R3 is a register and $j = 2$ if R3 is a non-negative integer.

4. `!R1` is coded by

$$110\text{code}_1(\text{R1}).$$

5. `%` is coded by

$$100.$$

4. The complexity of Fermat's last theorem

Fermat's last theorem is one of the most famous theorems in the history of mathematics. It states that there are no positive integers x, y, z satisfying the equation $x^n + y^n = z^n$, for any integer value $n > 2$. The result was conjectured by Pierre de Fermat in 1637, and it was proven only in 1995 by A. Wiles [17] (see also [1]). Many illustrious mathematicians failed to proved it, but their efforts stimulated the development of algebraic number theory.

The register machine program presented below uses the integer $B \geq 5$ to enumerate all 4-tuples of integers (x, y, z, n) with $z \leq B, x, y < z, n \leq B$ for which the equality $x^n + y^n = z^n$ is tested.

The register machine program for Fermat's last theorem is:

```
0. =a,a,14
1. &e,0     //===a^b
2. &d,1
3. +e,1
4. &f,0
5. &g,0
6. +f,1
7. +g,a
8. =f,d,10
```

```
 9. =a,a,6
10. &d,g     //g = a*d
11. =e,b,13
12. =a,a,3
13. =a,a,c   //d = a^b
14. &B,4     //===main program
15. +B,1
16. &n,3
17. +n,1
18. =n,B,15
19. &z,3
20. +z,1
21. =z,B,17
22. &x,3
23. +x,1
24. =x,z,20
25. &y,3
26. +y,1
27. =y,z,23
28. &b,n
29. &a,x
30. &c,32
31. =a,a,1   //d = x^n
32. &E,d
33. &a,y
34. +c,4     //c = 36
35. =a,a,1   //d = y^n
36. +E,d     //E = x^n + y^n
37. &a,z
38. +c,4     //c = 40
39. =a,a,1   //d = z^n
40. =E,a,42  //x^n + y^n = z^n
41. =a,a,26  //x^n + y^n =/= z^n
42. %        //Fermat Theorem is false
```

The register machine program for Fermat's last theorem has 43 instructions. Its size is 597 bits[6], hence the Fermat's last theorem is in $\mathfrak{C}_{U,1}$. Ac-

[6]We use: R_1 = a, R_2 = d, R_3 = z, R_4 = c, R_5 = B, R_6 = x, R_7 = n, R_8 = y, R_9 = e,

cording to Theorem 1 we obtain:

Theorem 3. *Assume* ZFC *is arithmetically sound. Then, one can effectively construct in the formal language of* ZFC *the expression describing a two-dimensional Hamiltonian system* $\mathcal{H}$ *such that* ZFC *proves that* $\mathcal{H}$ *has a Smale horseshoe iff there exists a* Π_1*-statement* $\sigma \in \mathfrak{C}_{U,1}$ *such that* ZFC *proves* σ*.*

5. Conclusions

Using the computational method in [4, 2, 3] we have shown that the problem of proving the existence of a Smale horseshoe in a two-dimensional Hamiltonian system is in the class $\mathfrak{C}_{U,1}$, i.e. it has low complexity according to our complexity measure. The specific pair of two-dimensional Hamiltonians used in the proof of Theorem 1 plays no specific role: any pair of Hamiltonians, one for a dynamics displaying chaotic behaviour and one for a smooth dynamics, will be equally useful in Eq (1).

It will be interesting to investigate whether the results presented in this note for Fermat's last theorem can be generalised for anyΠ_1-statement (in [7] it is claimed that Theorem 1 is true for a couple of otherΠ_1-statements).

References

[1] A. Aczel. *Fermat's Last Theorem: Unlocking the Secret of an Ancient Mathematical Problem*, Dell Publishing, New York, 1996.

[2] C. S. Calude, E. Calude. Evaluating the Complexity of Mathematical Problems. Part 1 *Complex Systems*, 19 (2009), 267–285.

[3] C.S. Calude, E. Calude. Evaluating the Complexity of Mathematical Problems. Part 2, *Complex Systems* 18 (2010), 387–401.

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

[5] B. F. Caviness. On canonical forms and simplification, *Journal of the Association for Computing Machinery* 17, 2 (1970), 385–396.

[6] G. J. Chaitin. *Algorithmic Information Theory*, Cambridge University Press, Cambridge, 1987. (third printing 1990)

R_{10} = f, R_{11} = g, R_{12} = E, R_{13} = b.

[7] N. C. A. da Costa, F. A. Doria, A. F. Furtado do Amaral. Dynamical system where proving chaos is equivalent to proving Fermat's conjecture, *International Journal of Theoretical Physics* 32, 11 (1993), 2187–2206.

[8] R. L. Devaney. *An Introduction to Chaotic Dynamical Systems*, 2nd ed. Westview Press, 2003.

[9] P. Gács, M. Hoyrup, C. Rojas. Randomness on computable probability spaces. A dynamical point of view, *Symposium on Theoretical Aspects of Computer Science 2009* (Freiburg), pp. 469–480.

[10] J. Hartmanis. On effective speed-up and long proofs of trivial theorems in formal theories, *Informatique Théorique et Applications* 10 (1976), 29–38.

[11] M. Hirsch. The chaos of dynamical systems, in P. Fisher, W. R. Smith (eds.). *Chaos, Fractals and Dynamics*, Marcel Dekker, 1985, 189–195.

[12] P. J. Holmes, J. E. Marsden. Horseshoes in perturbations of Hamiltonian systems with two degrees of freedom, *Communications in Mathematical Physics* 82 (1982), 523–544.

[13] S. H. Kellert. *In the Wake of Chaos: Unpredictable Order in Dynamical Systems*, University of Chicago Press, 1993.

[14] M. Laczkovich. The removal of π from some undecidable problems involving elementary functions, *Proceedings of the American Mathematical Society* 131, 7 (2002), 2235–2240.

[15] D. Richardson. Some unsolvable problems involving elementary functions of a real variable, *Journal of Symbolic Logic* 33 (1968), 514–520.

[16] P. Wang. The undecidability of the existence of zeros of real elementary functions, *Journal of the Association for Computing Machinery* 21 4, (1974), 586–589.

[17] A. Wiles. Modular elliptic curves and Fermat's Last Theorem, *Annals of Mathematics* 141 (3) (1995), 443–551.