

Lectures on Philosophy and Computation

Cristian S. Calude

Semester 1, 2016

Sans les mathématiques on ne pénètre point au fond de la philosophie.

Sans la philosophie on ne pénètre point au fond des mathématiques.

Sans les deux on ne pénètre point au fond de rien.

Without mathematics one cannot understand the fundamentals of philosophy.

Without philosophy we cannot reach the foundation of mathematics.

Without both (mathematics and philosophy) one cannot reach anything that is fundamental.

Bibliography (books)

- ▶ J. Barrow. *Impossibility—The Limits of Science and the Science of Limits*, Oxford University Press, Oxford, 1998.
- ▶ J. M. Borwein, D. Bailey, R. Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery*, A.K. Peters, Natick, MA, 2004.
- ▶ C. S. Calude. *Information and Randomness – An Algorithmic Perspective*, Springer-Verlag, 2002.

The halting problem

The problem of determining, from a description of an arbitrary computer program and an input, whether the program will finish running or continue to run forever is called the **halting problem**.

Alan Turing proved in 1936 that **no algorithm can solve the halting problem for all possible program-input pairs: the halting problem is undecidable**.

In what follows we will give an information-theoretic proof using the informal notion of “program”.

Without restricting the generality of the proof we assume that all programs incorporate inputs—which are coded as natural numbers. A program may run forever or may just eventually stop, in which case it prints a natural number.

A troubling program

Assume that there exists a **halting program** deciding whether an arbitrary program eventually halts. Construct the following program $\text{Trouble}(N)$:

1. read a natural N ;
2. generate all programs up to N bits in size;
3. use the **halting program** to check for each generated program whether it halts; remove non-halting programs;
4. simulate the running of the above generated programs, and
5. compute the biggest value output by these programs, say o , and output $2o + 1$.

The proof

- ▶ The program $\text{Trouble}(N)$ **halts** for every natural N .
- ▶ The program $\text{Trouble}(N)$ about $\log N$ bits. Reason: to know N we need $\log_2 N$ bits (in binary); the rest is a constant, so $\text{Trouble}(N)$ is $\log N + O(1)$ bits.
- ▶ For large enough N , $\text{Trouble}(N)$ **belongs** to the set of programs having less than N bits (because $\log N + O(1) < N$).
- ▶ $\text{Trouble}(N)$ **generates itself** at some stage of the computation.
- ▶ **Contradiction!**. Reason: on one hand, $\text{Trouble}(N)$ outputs a natural number no larger than o , but, on the other hand, it outputs $2o + 1$!

In early times, the concepts of chance and randomness were intertwined with that of fate.

Ancient peoples threw dice to determine fate.

The Chinese formalised odds and chance about 3,000 years ago.

The Greek philosophers discussed randomness at length, but only in non-quantitative forms.

It was only in the sixteenth century that Italian mathematicians began to formalise the odds associated with various games of chance.

What is randomness?

Randomness means lack of pattern or predictability; it suggests a non-order or non-coherence in a sequence of symbols or steps, such that there is no intelligible pattern or combination.

Symptoms of randomness:

- ▶ typicality,
- ▶ lack of patterns,
- ▶ unpredictability,
- ▶ incompressibility,
- ▶ ...

Two examples

The string

```
000000000000000000000000000000000000
```

is clearly non-random.

The string

```
00111100010001101101010001000010
```

seems random.

More examples

Which strings are random?

01100110011001100110011001100110

00111100010001100011110001000110

0000000000000000000011110001000110

00001010101000000010000000101000

00100100010000101001010001000110

The paradox of randomness

The French mathematician Émile Borel, a pioneer of probability theory, argued that there is no way to formalise in an acceptable way the concept of randomness. His argument is based only one symptom of randomness, **typicality** and is known today as the **randomness paradox**.

A random bit-string should be “typical”: it should not stand out from the crowd of other bit-strings.

Assume that there is a precise way to distinguish between “random bit-strings” and bit-strings which are “non-random”. **It does not matter how this criterion was found or operates.**

Operationally, we have a precise **criterion** which can be applied to any bit-string and once a bit-string is given, we can say whether the bit-string is *random* or *non-random*.

Can the adopted **criterion** be consistent? The answer is *negative*.

Indeed, choose the first bit-string which **criterion** asserts it is random. This particular bit-string is

the first bit-string satisfying the property of being random,

a property making it atypical, so non-random!

What are we left with?

- ▶ randomness does not exist,
- ▶ randomness is a matter of degree: we have a hierarchy of definitions of randomness,
- ▶ use a different paradox,
- ▶ ...

Name numbers in English.

Zero is the first number.

One is the first non-zero number.

One is the second number.

Find the value of the number defined by:

The first number not nameable in under ten words.

What is the reason of the “paradox”?

Combining strings

Let x, y be two binary strings. Can we combine them into a single string z in such a way that both x and y can be retrieved algorithmically from z ?

Here is a solution: use the prefix-free encoding of strings. Given $x = x_1x_2 \dots x_n$ and y ,

$$z = \langle x, y \rangle = x_1x_1x_2x_2 \dots x_nx_n01y.$$

It is seen that the length of $|\langle x, y \rangle|$ is $2|x| + 2 + |y|$.

In what follows we use algorithms (for example, TM machines) to describe strings.

Algorithms will be written in binary, i.e. they will be represented by binary strings denoted M, N, \dots

We describe a string x with an algorithm M and a binary input w such that

$$M(w) = x.$$

The length of the description is the combined length of M and w :

$$|\langle M, w \rangle|.$$

Note that from $\langle M, w \rangle$ we can algorithmically extract the unique components M and w , and $|\langle M, w \rangle| = 2|M| + |w| + 2$.

Let x be a binary string. The **minimal description** of x , written

$$d(x),$$

is the shortest string $\langle M, w \rangle$, where M is an algorithm which on input w halts and produces x . If several such strings exist, we select the lexicographically first among them.

The **descriptive complexity** or **Kolmogorov-Chaitin complexity** of x , written $K(x)$, is

$$K(x) = |d(x)|.$$

Theorem. $\exists c \forall x [K(x) \leq |x| + c]$.

Proof. Consider an algorithm that computes the identity function, $M(x) = x$.

A description of x is simply $\langle M, x \rangle$ which has the length:

$$|\langle M, w \rangle| = 2|M| + |x| + 2 = |x| + c,$$

where $c = 2|M| + 2$ is a fixed constant depending only on M (and not on x).

Theorem. The function K is unbounded, that is,

$$\forall M > 0 \exists x [K(x) > M].$$

Proof. Assume for a contradiction that K is bounded, that is, there exists $M_0 > 0$ such that for every x we have $K(x) \leq M_0$. As the set

$$\{x \mid K(x) \leq M_0\}$$

is **finite** (why?), we have obtained a contradiction.

Theorem. The function K is not computable.

Proof. Assume for a contradiction that K is computable. Fix $m \geq 0$ and define

$$f_m = \min[x \mid K(x) > m],$$

where \min is taken according to the lexicographical order. As K is unbounded, for every M there exists a string x such that

$$K(x) > M. \tag{1}$$

Because K is computable, f_m is also computable, say, by an algorithm R . Clearly,

$$|R| \leq \log_2(M) + \text{constant} < (M - 2)/2,$$

for large enough M and $\langle R, \varepsilon \rangle$ is a description of a string x with $K(x) > M$, a contradiction with (1).

A **general description language** is a computable function $p : \Sigma^* \rightarrow \Sigma^*$; the **minimal description** of x with respect to p , written $d_p(x)$, is the lexicographically shortest string s where $p(s) = x$ and

$$K_p(x) = |d_p(x)|.$$

For example, for LISP (encoded into binary) as the description language, $d_{\text{LISP}}(x)$ is the minimal LISP program that outputs x , and $K_{\text{LISP}}(x)$ is the length of the minimal program producing x .

Theorem. For any descriptive language p , there exists a constant c (that depends only on p) such that

$$\forall x [K(x) \leq K_p(x) + c].$$

Proof. Let p be a description language and consider the algorithm M such that $M(w) = p(w)$, for all w .

Then $\langle M, d_p(x) \rangle$ is a description of x and

$$|\langle M, d_p(x) \rangle| = 2|M| + K_p(x) + 2 = K_p(x) + c,$$

where $c = 2|M| + 2$.

A string's minimal description is never much longer than the string itself. Of course for some strings, the minimal description may be much shorter if the information in the string appears sparsely or redundantly.

But, are strings with no short descriptions?

A string x is c -compressible ($1 \leq c < |x|$) if $K(x) \leq |x| - c$.
If x is not c -compressible, we say that x is incompressible by c .
If x is incompressible by 1, we say that x is incompressible.

Do incompressible strings exist?

Theorem. Incompressible strings of every length exist.

Proof. Each description is a binary string, so the number of descriptions of length less than n is at most the sum of the number of strings length $0, 1, \dots$ up to $n - 1$:

$$1 + 2 + 4 + \dots + 2^{n-1} = 2^n - 1.$$

The number of short descriptions is less than the number of strings of length n —which is 2^n , so at least one string of length n is incompressible.

Most strings are incompressible

Corollary. At least $2^n - 2^{n-c+1} + 1$ strings of length n are incompressible by c .

Proof. As in the proof above, at most $2^{n-c+1} - 1$ strings of length n are c -compressible (at most that many descriptions of length at most $n - c$ exist), hence the remaining

$$2^n - (2^{n-c+1} - 1)$$

strings of length n are incompressible by c .

Incompressible strings have many properties of “random strings”, i.e. in a long enough incompressible string the numbers of 0s and 1s are roughly equal, the length of its longest run of 0s is about the log of the string’s length and the set of incompressible strings is not computable.

Theorem. For every $b > 0$, the set $R_b = \{x \mid K(x) > |x| - b\}$ is incomputable.

Proof. Obviously, R_b is infinite (why?). Assume for the sake of a contradiction that R_b is computable by M .

Construct the function (defined on a non-negative integer represented in binary):

$$p(n) = \min[x \in R_b \mid n - b < |x|],$$

where \min is taken according to the fixed enumeration given by M .

Proof continued.

1. For each n , $p(n) \in R_b$ and from the definition of p we have:

$$K(p(n)) > |p(n)| - b > n - b.$$

2. There exists a constant d such that for every n we have:

$$K_p(p(n)) \leq \log_2(n) + d.$$

3. In view of the optimality of K , there exists constant c such that for all n :

$$n - b < K(p(n)) \leq K_p(p(n)) + c \leq \log_2(n) + c + d,$$

a contradiction.

Russell's paradox, discovered by Bertrand Russell in 1901, showed that the naive set theory created by Georg Cantor leads to a contradiction. This generated a crisis in the foundations of mathematics.

According to naive set theory, any definable collection is a set.

Let R be the set of all sets that are not members of themselves.

If R is a member of itself, it would contradict its own definition as the set containing all sets that are not members of themselves.

If R is not a member of itself, it would qualify as a member of itself by the same definition, again a contradiction.

Give examples of:

1. sets which are not members of themselves,
2. sets which are members of themselves.

The main goal of Hilbert's program was to provide **secure** foundations for all mathematics. This includes:

- ▶ A formalisation of mathematics: all mathematical statements should be written in a precise formal language, and manipulated according to well defined rules (formal system).
- ▶ Completeness: a proof that all **true** mathematical statements can be proved in the adopted formal system.
- ▶ Consistency: a proof—preferably reasoning about finite mathematical objects only—that no contradiction can be obtained in the formal system.

- ▶ Decidability: construction of an algorithm for deciding, in the formal system, the truth or falsity of any mathematical statement.
- ▶ Conservation: a proof that any result about “real objects” obtained using “ideal objects” (such as uncountable sets) can be restated without using ideal objects.

*Is the axiom of solvability of every problem a peculiar characteristic of mathematical thought alone, or is it possibly a general law inherent in the nature of the mind, that all questions which it asks must be answerable?
... This conviction of the solvability of every mathematical problem is a powerful incentive to the worker. We hear within us the perpetual call: There is the problem. Seek its solution. You can find it by pure reason, for in mathematics there is no igorabimus.*

By $N(P, v)$ we mean that the program P will *never* halt when begun with input v .

For any particular program P and input v , $N(P, v)$ is a perfectly definite statement which is either true (in case P will never halt in the described situation) or false (in case P will eventually halt).

Incompleteness: a difficulty

When $N(P, v)$ is false, this fact can always be demonstrated by running P on v .

No amount of computation will suffice to demonstrate the fact that $N(P, v)$ is true. We may still be able to prove that a particular $N(P, v)$ is true by a logical analysis of P 's behaviour, but, as we proved in the undecidability of the halting problem, no such method works correctly in all cases.

What is a *proof*?

Suppose that certain strings of symbols (possibly paragraphs of a natural language (English, for example)) have been singled out—typically with the help of axioms and rules of inference—as proofs of particular statements of the form $N(P, v)$.

Operationally, we assume that we have an algorithm called **syntactic test** that can test an alleged proof Π that $N(P, v)$ is true and determine whether Π is or is not actually such a proof.

There are two basic requirements which it is natural to demand of our supposed rules of proof:

- ▶ *Soundness*: If there is a proof Π that $N(P, v)$ is true, then P will in fact never halt when begun with v on its tape.
- ▶ *Completeness*: If P will never halt when begun with v on its tape, then there is a proof Π that $N(P, v)$ is true.

Theorem [Gödel's incompleteness theorem]

No rules of proof can be both sound and complete.

Obviously, soundness cannot be sacrificed.

If a given set of rules of proof is sound, then, according to Gödel's incompleteness theorem, there is some true statement $N(P, v)$ which has no proof Π according to the given rules of proof.

Such a true unprovable statement is called *undecidable* since it will surely not be disprovable.

Suppose we had found rules of proof which were both sound and complete.

Suppose that the “proofs” according to these rules were particular strings of symbols on some specific finite alphabet.

Let $\Pi_1, \Pi_2, \Pi_3, \dots$ be the quasi-lexicographic computable enumeration of all finite strings on this alphabet. This sequence includes all possible proofs, as well as a lot of other things (including a high percentage of total nonsense). But, hidden among the nonsense, are all possible proofs.

Now we show how we can use our supposed rules of proof to solve the halting problem—an impossibility.

So, we are given a program P and an input v and wish to test whether or not P will eventually halt when begun on v .

We run in parallel two computations:

- ▶ the computation of P on v ,
- ▶ we generate the sequence $\Pi_1, \Pi_2, \Pi_3, \dots$ of possible proofs and, as each Π_i is generated, we use the **syntactic test** to determine whether or not Π_i is a proof of $N(P, v)$.

Incompleteness: the proof (cont.)

If P will eventually halt on v we will find out in finite time.

If P will never halt on v , since our rules of proof are assumed to be complete, there will be a proof Π_i of $N(P, v)$ which we will **discover** via the enumeration $\Pi_1, \Pi_2, \Pi_3, \dots$ of possible proofs.

Having obtained this Π_i we will be sure (because of soundness) that P will indeed never halt.

Thus, we have described an algorithm which would solve the halting problem, a contradiction!

Gödel's theorem does not indicate any particular pair P, v for which we will never be able to convince ourselves that $N(P, v)$ is true!

It says that for any given sound rules of proof, there will be a pair P, v for which $N(P, v)$ is true, but not provable *using the given rules*.

There may well be, and in fact there always are, other sound rules which decide this “undecidable” statement. But these other rules will in turn have their own undecidabilities.

Incompleteness and its consequences showed that most of the goals of Hilbert's program are impossible to achieve, at least if interpreted in the "most obvious way".

However, much of it can be salvaged by changing its goals slightly, and with the following modifications some of it was successfully completed:

- ▶ Although it is not possible to formalise all mathematics, it is possible to formalise essentially all the mathematics that "anyone uses". Zermelo-Fraenkel set theory, combined with first-order logic, gives a satisfactory and generally accepted formalism for essentially all current mathematics.

- ▶ Although it is not possible to prove completeness for systems at least as powerful as Peano arithmetic (if they have a computable set of axioms), it is possible to prove forms of completeness for many interesting systems. Gödel proved the completeness theorem for first-order logic before he proved the incompleteness theorems.

Hilbert's programme after incompleteness (cont.)

- ▶ The theory of algebraically closed fields of given characteristic is complete.
- ▶ Although there is no algorithm for deciding the truth of statements in Peano arithmetic, Tarski found an algorithm that can decide the truth of any statement in analytic geometry (more precisely, the theory of real closed fields is decidable). Given the Cantor-Dedekind axiom, this algorithm can decide the truth of any statement in Euclidean geometry.

Objective vs. subjective mathematics

- ▶ Objective mathematics consists of the body of mathematical propositions, constructive or not, which hold true in an absolute sense. Peano Arithmetic or Zermelo-Fraenkel set theory are parts of it.
- ▶ Subjective mathematics consists of all mathematical truths *humanly demonstrable* (or *provable* or *knowable*), in a constructive manner or not.

Does objective mathematics coincide with subjective mathematics?

Gödel **accepted** Hilbert's rejection of the existence of absolutely unsolvable problems because otherwise,

it would mean that human reason is utterly irrational by asking questions it cannot answer, while asserting emphatically that only reason can answer them

but found Turing's argument **inconclusive**:

Turing gives an argument which is supposed to show that mental procedures cannot go beyond mechanical procedures. However, this argument is inconclusive. What Turing disregards completely is the fact that mind, in its use, is not static, but constantly developing, i.e., we understand abstract terms more and more precisely as we go on using them . . . though at each stage the number and precision of the abstract terms at our disposal may be finite, both . . . may converge toward infinity . . .

Does objective mathematics coincide with subjective mathematics?

Gödel's answer (Gibbs lecture "Some Basic Theorems on the Foundations of Mathematics and their Implications", [2]) based on his incompleteness theorem is a **disjunctive conclusion**:

*Either mathematics is incompletable in this sense, that its evident axioms can never be comprised in a finite rule, that is to say, **the human mind (even within the realm of pure mathematics) infinitely surpasses the powers of any finite machine, or else there exist absolutely unsolvable diophantine problems of the type specified.***

Does objective mathematics coincide with subjective mathematics?

Martin-Löf's answer based on a *constructive* interpretation of the notions of “true”, “false” and “can be known” [1]:

There are no propositions which can neither be known to be true nor be known to be false.

For the non-constructive mathematician:

No propositions can be effectively produced (i.e. by an algorithm) of which it can be shown that they can neither be proved constructively nor disproved constructively. There may be absolutely unsolvable problems, but one cannot effectively produce one for which one can show that it is unsolvable.

Guided by Post [2, p. 200],[3]

A fundamental problem is the question of the existence of absolutely undecidable propositions, that is, propositions which in some a priori fashion can be said to have a determined truth-value, and yet cannot be proved or disproved by any valid logic

we will only require that the objective mathematics contains the subjective mathematics.

Furthermore, in contrast with Feferman [1], we will include in subjective mathematics all statements provable by any methods, axiomatic (dynamic, not only static), constructive, computational or by methods currently not yet discovered.

Constructive logical interpretations:

- ▶ The proposition A can be known to be true if we have a proof for A .
- ▶ The proposition $A \vee B$ can be known to be true if we have a proof for A or we have a proof for B .
- ▶ The proposition $A \wedge B$ can be known to be true if we have a proof for A and we have a proof for B .
- ▶ The proposition $A \rightarrow B$ can be known to be true if we have an algorithm which converts any proof for A into a proof for B .
- ▶ The proposition $\neg A$ can be known to be true if we have a proof for $A \rightarrow (0 = 1)$.

Martin-Löf's argument (cont.)

- ▶ The proposition A **can be known to be false** if we have a proof for $\neg A$.
- ▶ The proposition A **cannot be known to be true** if we have an algorithm which tests and rejects any given 'proof' which purports to demonstrate A .
- ▶ If the proposition A can be known to be true, then A is true.
- ▶ Martin-Löf's notions of **can be known to be true/false** are not related to any fixed formal system.

Fact 1. [Unknowability of truth entails knowability of falsity] *If the proposition A cannot be known to be true, then A can be known to be false.*

Proof. To prove that A can be known to be false we have to show that $\neg A = A \rightarrow (0 = 1)$ can be known to be true. To this aim we need an algorithm \mathcal{B} to convert any proof of A into a proof of $(0 = 1)$. The algorithm \mathcal{B} returns anything output by the algorithm \mathcal{A} provided by the hypothesis, i.e. noting: vacuously, the implication holds.

Comment: The proof constructively produces positive information from negative information.

Fact 2. *If A can be known to be true and B can be known to be true, then $A \wedge B$ can be known to be true.*

Fact 3. [Absolute consistency] *The proposition $(0 = 1)$ cannot to be known to be true.*

Proof. The proposition $\neg(0 = 1)$ can be known to be true because $(0 = 1) \rightarrow (0 = 1)$ is provable using the identity algorithm, so $(0 = 1)$ can be known to be false, i.e. it is false. No proof can demonstrate $(0 = 1)$ because otherwise it would be true: the algorithm rejects any proof candidate.

Fact 4. [Law of contradiction] *One and the same proposition A cannot both be known to be true and be known to be false.*

Proof. By hypothesis we have a proof demonstrating A and a proof demonstrating $\neg A = A \rightarrow (0 = 1)$. Then we can demonstrate $(0 = 1)$, contradicting Fact 3.

Fact 5. [Law of excluded middle] *There is no proposition which can neither be known to be true nor be known to be false, i.e. there is no absolutely unprovable proposition.*

Proof. If A is a proposition which cannot be known to be true, then by Fact 1, A can be known to be false, a contradiction.



Solomon Feferman.

Are there absolutely unsolvable problems? Gödel's dichotomy.
Philosophia Mathematica, 14(2):134–152, 2006.



Kurt Gödel.

Some basic theorems on the foundations of mathematics and their implications.

In S. Feferman, J. W. Dawson, Jr., W. Goldfarb, C. Parsons, and R. M. Solovay, editors, *Collected Works. Unpublished Essays and Lectures. Volume III*, pages 304–323. Oxford University Press, 1995.



Per Martin-Löf.

Verification then and now.

In W. De Pauli-Schimanovich, E. Koehler, and F. Stadler, editors, *The Foundational Debate, Complexity and Constructivity in Mathematics and Physics*, pages 187–196. Kluwer Academic Publishers Dordrecht, 1995.



Emil L. Post.

Formal reductions of the general combinatorial decision problem.

American Journal of Mathematics, 65:197–215, 1943.



Alasdair Urquhart.

Emil Post.

In Dov M. Gabbay and John Woods, editors, *Logic from Russell to Church. Handbook of the History of Logic 5*, pages 617–666. Elsevier Science B.V., 2009.

Proof: 1. a fact or thing that shows or helps to show that something is true or exists; 2. a demonstration of the truth of something, “in proof of my statement”; 3. the process of testing whether something is true or good or valid, “put it to the proof”. To prove: to give or be proof of; to establish the validity of; to be found to be, “it proved to be a good theory”; to test or stay out. To argue: 1. to express disagreement, to exchange angry words; 2. to give reasons for or against something, to debate; 3. to persuade by talking, “argued him into going”; 4. to indicate, “their style of living argues that they are well off”. Argument: 1. a discussion involving disagreement, a quarrel; 2. a reason put forward; 3. a theme or chain of reasoning. (Oxford American Dictionary)

A critique of the definition of proof

In all these statements, nothing is said about the means used “to show or help to show that something is true or exists”, about the means used “in the process of testing whether something is true or good or valid”.

In argumentation theory, various ways to argue are discussed, deductive reasoning being only one of them. We use suggestions, impressions, emotions, logic, gestures, mimicry, etc.

An *informal* (pen-on-paper) *proof* is a rigorous argument expressed in a mixture of natural language and formulae (for some mathematicians an equal mixture is the best proportion) that is intended to convince a knowledgeable mathematician of the truth of a statement, the theorem. Routine logical inferences are omitted. “Folklore” results are used without proof. Depending on the area, arguments may rely on intuition. Informal proofs are the standard of presentation of mathematics in textbooks, journals, classrooms, and conferences. They are the product of a social process.

A *formal proof* is written in a formal language consisting of certain strings of symbols from a fixed alphabet. Formal proofs are precisely specified without any ambiguity because all notions are explicitly defined, no steps (no matter how small) are omitted, no appeal to any kind of intuition is made. They satisfy Hilbert's criterion of mechanical testing:

The rules should be so clear, that if somebody gives you what they claim is a proof, there is a mechanical procedure that will check whether the proof is correct or not, whether it obeys the rules or not.

By making sure that every step is correct, one can tell once and for all whether a proof is correct or not, i.e. whether a theorem has been proved.

Formal proof cannot be found in mathematical articles or books (except for a few simple examples).

However, most mathematicians believe that almost all “real” proofs, published in articles and books, can, with tedious work, be transformed into Hilbertian proofs. Why?

Because “real” proofs look *convincing* for the mathematical community. Going further, DeMillo, Lipton and Perlis argued that “real proofs” should be highly non-monolithic because they aim to be heard, read, assimilated, discussed, used and generalised by mathematicians—they are part of a social process.

Programming is the activity of solving problems with computers. It includes the following steps:

- a) developing the *algorithm* to solve the problem,
- b) writing the algorithm in a specific programming language (that is, coding the algorithm into a program),
- c) assembling or compiling the program to turn it into machine language,
- d) testing and debugging the program,
- e) preparing the necessary documentation.

Ideally, at d) one should have written:

d) *proving the correctness of the algorithm*, testing and debugging the program.

We said, ideally, because correctness, although desired, is only practised in very few instances (for example, the programs involved in the proof of the Four-Color Theorem were not proved correct!)

It makes sense to prove the correctness of an algorithm, but *not*, the correctness of a program. Programs are analogues of mathematical models, they may be more or less adequate to code algorithms.

Adequacy is a property which depends on many factors, from pure formal/coding ones to physical and engineering ones.

One can even argue that a “correctness proof” for a program, if one could imagine such a thing, adds very little to the confidence in the program. In Knuth’s words:

Beware of bugs in the above code: I have only proved it correct, not tried it.

The role of proof in mathematical modelling is very small: *adequacy is the main issue!* Here is an illustration from Jack Schwartz:

... it may come as a shock to the mathematician to learn that the Schrödinger equation for the hydrogen atom ... is not a literally correct description of this atom, but only an approximation to a somewhat more correct equation taking account of spin, magnetic dipole, and relativistic effects; that this corrected equation is itself only an ill-understood approximation to an infinite set of quantum field-theoretical equations; and finally that the quantum field theory besides diverging, neglects a myriad of strange-particle interactions whose strength and form are largely unknown. ...

The modelling component of mathematics appears not only in applications, but also in the way mathematics develops new concepts.

Many important notions in mathematics reached an accepted definition only after a long process of modelling, from an intuitive, pre-mathematical notion to a more precisely defined, formal one. In the end, the accepted definition is adopted as a “thesis” claiming its adequacy.

For example, “Weierstrass’ thesis” is the statement that the intuitive notion of continuity is extensionally equivalent to the notions yielded by the now standard definitions of continuous function.

Other examples include:

- ▶ “The function thesis”: identification of a function with a set of ordered pairs,
- ▶ “Tarski’s thesis”: identification of Tarski’s definition of truth in a formalised language with the intuitive notion of truth,
- ▶ “Church-Turing thesis”.

None of these “theses” can be *proved*, but various analyses can conclude their degrees of plausibility/adequacy/applicability. Mathematics in both its practice and development is an “open-texture” .

There are many “new” types of proofs, probabilistic, experimental or hybrid proofs (computation plus theoretical arguments).

Zeilberger has argued in favour of the transition from rigorous proofs to an “age of *semi-rigorous* mathematics, in which identities (and perhaps other kinds of theorems) will carry price tags” measured in computer and human resources necessary to prove them with a certain degree of confidence.

The real work of us mathematicians, from now until, roughly, fifty years from now, when computers won't need us anymore, is to make the transition from human-centric math to machine-centric math as smooth and efficient as possible.

No theorem is validated before it is “communicated” to the mathematical community (orally and, eventually, in writing).

Manin:

Proof is not just an argument convincing an imaginary opponent. Not at all. Proof is the way we communicate mathematical truth.

However, as Rota pointed out:

One must guard, however, against confusing the presentation of mathematics with the content of mathematics.

Proofs have to be written on paper, which means proofs are *physical*. From this perspective, proofs depend upon the physical universe (Calude and Chaitin).

Standards of rigour have changed throughout the history of mathematics and not necessarily from less rigour to more rigour.

B. Russell:

I wanted certainty in the kind of way in which people want religious faith.

J.-P. Serre was quoted saying that mathematics is the only producer of “totally reliable and verifiable” truths.

D. Knuth:

... programming demands a significantly higher standard of accuracy. Things don't simply have to make sense to another human being, they must make sense to a computer.

W. P. Thurston:

The standard of correctness and completeness necessary to get a program to work at all is a couple of orders of magnitude higher than the mathematical community's standard of valid proofs.

When one considers how hard it is to write a computer program even approaching the intellectual scope of a good mathematical paper, and how much greater time and effort have to be put into it to make it "almost" formally correct, it is preposterous to claim that mathematics as we practice it is anywhere near formally correct.

A conceptual period reaches its maturity under the form of an operational one, which, in its turn, is looking for a new level of conceptual attitude.

The whole treatise of Bourbaki is a conceptual reaction to an operational approach. Dirichlet's slogan asking to replace calculations with ideas should be supplemented with another, complementary slogan, requiring to detect an algorithmic level of concepts.

Can we expect a similar alternation of attitudes in respect to programming? Perhaps it is too early to answer, taking into account that the whole field is still too young. The question is not only academical as the project Flyspeck (to produce a formal proof of the Kepler Conjecture [completed in 2014](#)) reminds us.

In theory, each informal proof can be converted into a formal proof. However, this is rarely, almost never, done in practice.

A proof assistant (interactive theorem prover) is a software tool to assist with the development of formal proofs by man-machine collaboration.

Proof-assistants can be used not only to check the validity of a formalised proof of a known mathematical result, but also to interactively help to “prove” new theorems.

Hilbert's standard of proof is practicable, it's becoming reality

An impressive record of deep mathematical theorems formally proved:

- ▶ Gödel Incompleteness Theorem (1986),
- ▶ the Fundamental Theorem of Calculus (1996),
- ▶ the Fundamental Theorem of Algebra (2000),
- ▶ the Four Colour Theorem (2004),
- ▶ Jordan's Curve Theorem (2005),
- ▶ the Prime Number Theorem (2008).

The December 2008 issue of the *Notices of AMS* includes four papers on formal proof.

Produced by INRIA, free distributed under the GNU Lesser General Public Licence, <http://coq.inria.fr/>, Coq is a formal proof management system providing a formal language to write mathematical definitions, executable algorithms and theorems together with an environment for semi-interactive development of machine-checked proofs.

Typical applications include the formalisation of programming languages semantics, formalisation of mathematics (e.g. Four Colour Theorem) and teaching.

Isabelle is a generic proof assistant, free distributed under the BSD license, <http://www.cl.cam.ac.uk/research/hvg/Isabelle/>, allowing mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus.

Isabelle is developed at the University of Cambridge, Technische Universität München and Université Paris-Sud.

The first formal proofs in algorithmic information theory have been developed in Isabelle at UoA (C. Calude and N. Hay).

How many naturals are even?

As you have learnt, the set of set of naturals has the same “number of elements” as the set of all even naturals.

However, we all “know” that, **about half of naturals are even.**

Can this statement be rigorously proved?

A **numeral** is a symbol or group of symbols that represents a number.

A number is a concept that a numeral expresses.

A numeral can be written down, erased, copied. A number can be expressed by different numerals. For example, the numerals

'10', 'ten', '1010', 'X'

represent the same number.

There exist various numeral systems, Babylonian, Roman, Arabic, Chinese, etc. We now use a positional numeral system using the base or radius 10, or 2, or 3, etc. Set theory deals with cardinal numbers, which can be finite or infinite.

Pirahã numeral system

A primitive tribe living in Amazon use a numeral system consisting of three numerals:

one, two, many.

In this system we cannot talk about 3,7 or 9. The only rules are:

many + one = many, many + two = many. (2)

Imagine we are in a granary and we are asked to count how much grains is inside. We can

1. count the grain seed by seed, or
2. use sacks, fill them with seeds, and count the number of sacks.

The first method is clearly impractical. The second one, which consists in counting the number of number of sacks and the remaining seeds (not enough to complete a sack) is approximate, but gives a good estimation.

3. If the granary is very large, wagons could be used first, then sacks and seeds.

A new infinite unit of measure expressed by the numeral

①

called *grossone* is introduced as the number of elements of the set of positive integers:

$$\mathbb{N} = \{1, 2, 3, \dots\}. \quad (3)$$

1. Infinity: For every $n \in \mathbb{N}$, $n < \textcircled{1}$.
2. Identity:
 $0 \cdot \textcircled{1} = \textcircled{1} \cdot 0 = 0$, $\textcircled{1} - \textcircled{1} = 0$, $\frac{\textcircled{1}}{\textcircled{1}} = 1$, $1^{\textcircled{1}} = 1$, $0^{\textcircled{1}} = 0$.
3. Divisibility: For every $n \in \mathbb{N}$, $1 \leq i \leq n$, the n th parts of $\textcircled{1}$ are

$$\frac{\textcircled{1}}{n} : \{i, n + i, 2n + i, \dots\}.$$

$$\textcircled{1} : \{1, 2, 3, \dots\}$$

$$\frac{\textcircled{1}}{2} : \{1, 3, 5, \dots\},$$

$$\frac{\textcircled{1}}{2} : \{2, 4, 6, \dots\},$$

$$\frac{\textcircled{1}}{3} : \{1, 4, 7, \dots\},$$

$$\frac{\textcircled{1}}{3} : \{2, 5, 8, \dots\}$$

$$\frac{\textcircled{1}}{n} : \{1, n + 1, 2n + 1, \dots\}.$$

Sets expressed with grossone

$$\textcircled{1} + 1,$$

$$\textcircled{1} - 2,$$

$$3\textcircled{1} + 45,$$

$$6\textcircled{1}^2 - 3\textcircled{1} + 14.$$

The set of positive integers with grossone

Recall that for every positive k , the set $\{1, 2, \dots, k\}$ has exactly k elements and k is the largest element. This happens for \mathbb{N} as well:

$$\mathbb{N} = \{1, 2, 3, \dots, \textcircled{1} - 2, \textcircled{1} - 1, \textcircled{1}\}.$$

Infinite natural numbers that are “invisible” for the traditional numeral system can now be “seen” with $\textcircled{1}$:

$$\dots, \frac{\textcircled{1}}{2} - 2, \frac{\textcircled{1}}{2} - 1, \frac{\textcircled{1}}{2}, \frac{\textcircled{1}}{2} + 1, \frac{\textcircled{1}}{2} + 2, \dots, \textcircled{1} - 2, \textcircled{1} - 1, \textcircled{1}.$$

The set of positive integers with grossone

We have:

$$\mathbb{N} = \{1, 2, \dots, \frac{\textcircled{1}}{2} - 2, \frac{\textcircled{1}}{2} - 1, \frac{\textcircled{1}}{2}, \frac{\textcircled{1}}{2} + 1, \frac{\textcircled{1}}{2} + 2, \dots, \textcircled{1} - 2, \textcircled{1} - 1, \textcircled{1}\}. \quad (4)$$

The traditional representation $\mathbb{N} = \{1, 2, \dots\}$ and (4) refer to the same set—the set of positive integers—they are both correct and do not contradict each other. **We have the same object—the set \mathbb{N} —that can be “observed” with different instruments (numeral systems) with different accuracies.**

Similarly, Pirahã are not able to see the finite natural numbers 3, 4, and 5; they reveal to a more powerful numeral system.

Extending \mathbb{N} with grossone

$$\tilde{\mathbb{N}} = \{1, 2, 3, \dots, \textcircled{1} - 1, \textcircled{1}, \textcircled{1} + 1, \dots, \textcircled{1}^2 - 1, \textcircled{1}^2, \textcircled{1}^2 + 1, \dots\}.$$

Measuring infinite sets

Description of sets	Cardinality	Number of elements
\mathbb{N}	countable, \aleph_0	$\textcircled{1}$
$\mathbb{N} \cup \{0\}$	countable, \aleph_0	$\textcircled{1}+1$
$\mathbb{N} \setminus \{3, 5, 10, 23, 114\}$	countable, \aleph_0	$\textcircled{1}-5$
the set of even numbers \mathbb{E}	countable, \aleph_0	$\frac{\textcircled{1}}{2}$
the set of odd numbers \mathbb{O}	countable, \aleph_0	$\frac{\textcircled{1}}{2}$
the set of integers \mathbb{Z}	countable, \aleph_0	$2\textcircled{1}+1$
$\mathbb{Z} \setminus \{0\}$	countable, \aleph_0	$2\textcircled{1}$
$\mathbb{G} = \{x \mid x = n^2, x \in \mathbb{N}, n \in \mathbb{N}\}$	countable, \aleph_0	$\lfloor \sqrt{\textcircled{1}} \rfloor$
$\mathbb{P} = \{(p, q) \mid p \in \mathbb{N}, q \in \mathbb{N}\}$	countable, \aleph_0	$\textcircled{1}^2$
$\mathbb{Q}' = \{-\frac{p}{q}, \frac{p}{q} \mid p \in \mathbb{N}, q \in \mathbb{N}\}$	countable, \aleph_0	$2\textcircled{1}^2$
$\mathbb{Q} = \{0, -\frac{p}{q}, \frac{p}{q} \mid p \in \mathbb{N}, q \in \mathbb{N}\}$	countable, \aleph_0	$2\textcircled{1}^2 + 1$
the set of numbers $x \in [0, 1)$ expressed in the binary	continuum, \mathcal{C}	$2^{\textcircled{1}}$

A full hotel has no room left. However, a full infinite hotel has vacancies.

Indeed, we first note that the infinite hotel has a countable set of rooms, as each room has a number (positive integer).

Suppose that the infinite hotel is full and a new guest arrives. To free a room for the guest we need to move the occupant of room 1 to room 2, then move the guest in room 2 to room 3, ... In this way the room 1 is free for the new guest.




Is this solution logically correct?

Recall that $\mathbb{N} = \{1, 2, 3, \dots, \textcircled{1}\}$.

In the grossone terminology, the hotel has $\textcircled{1}$ rooms.

When the new guest arrives, we move the occupant of room 1 to room 2, move the guest in room 2 to room 3, aso. In particular, the guest in room $\textcircled{1}$ has to move in room $\textcircled{1} + 1 \neq \textcircled{1}$, a contradiction. As in the case of a finite hotel, a full hotel has no vacancy!

References

-  Y. D. Sergeyev. *Arithmetic of Infinity*, Edizioni Orizzonti Meridionali, 2003.
-  Y. D. Sergeyev. Computations with Grossone-based infinities, in C. S. Calude, M. J. Dinneen (eds.). *Proc. 14th International Conference Unconventional Computation and Natural Computation*, Lecture Notes Comput. Sci. 9252, Springer, Heidelberg, 2015, 89–106.
-  L. H. Kauffman. Infinite computations and the generic finite, *Applied Mathematics and Computation* 255 (2015), 25–35.

George Box: All models are wrong, but some are useful.

Models have been able to consistently, if imperfectly, explain the world around us.

Is there any choice?

Sixty years ago, digital computers made information readable.

Twenty years ago, the Internet made it reachable.

Ten years ago, the first search engine crawlers made it a single database.

Kilobytes are stored on floppy disks, megabytes are stored on hard disks, terabytes are stored in disk arrays, and petabytes are stored in the cloud. We leave in the *Petabyte Age*.

Every day, we create 2.5 quintillion bytes of data – so much that 90% of the data in the world today has been created in the last two years alone. This data comes from everywhere: sensors used to gather climate information, posts to social media sites, digital pictures and videos, purchase transaction records, and cell phone GPS signals to name a few. This data is big data. [IBM: What is big data?](#)

We don't know why this page is better than that one. If the statistics of incoming links say it is, *that's good enough*. *No semantic or causal analysis is required*.

Operationalising this philosophy, Google

- ▶ can match ads to content without any knowledge or assumptions about the ads or the content;
- ▶ can translate languages without actually “knowing” them; it can translate Maori into Farsi as easily as it can translate French into English provided it has equal corpus data (see [WIKI-LINKS](#)).

According to [C. ANDERSON](#), *Wired Magazine*, Google's research director Peter Norvig offered an update to George Box's dictum:

All models are wrong, and increasingly you can succeed without them.

Unfortunately, Norvig [denies](#):

That's a silly statement, I didn't say it, and I disagree with it.

The big target here is science. The scientific method is built around testable hypotheses. The models are then tested, and experiments confirm or falsify theoretical models of how the world works. This is the way science has worked for hundreds of years.

THE END OF THEORY: THE DATA DELUGE MAKES THE
SCIENTIFIC METHOD OBSOLETE.

Welcome to data science! Long live the dead science!

Operationalising the idea

THE END OF THEORY: THE DATA DELUGE MAKES THE SCIENTIFIC METHOD OBSOLETE.

In short, the more we learn about [[biology]], the further we find ourselves from a model that can explain it.

There is now a better way. Petabytes allow us to say: "Correlation is enough." We can stop looking for models. We can analyze the data without hypotheses about what it might show. We can throw the numbers into the biggest computing clusters the world has ever seen and let statistical algorithms find patterns where science cannot.

NSF : *Computational and Data-Enabled Science and Engineering (CDS & E) is a new program. CDS & E is now clearly recognizable as a distinct intellectual and technological discipline lying at the intersection of applied mathematics, statistics, computer science, core science and engineering disciplines. . . . We regard CDS & E as explicitly recognizing the importance of data-enabled, data-intensive, and data centric science. CDS & E broadly interpreted now affects virtually every area of science and technology, revolutionizing the way science and engineering are done. Theory and experimentation have for centuries been regarded as two fundamental pillars of science. It is now widely recognized that computational and data-enabled science forms a critical third pillar.*

The “wave of the future”—as it was called—disposes of experiment and theory in science. It affects everything from astronomy to zoology, from medical sciences to social sciences.

History reminds us of similar exaggerations (recall chaos theory or catastrophe theory in mathematics): the “wave of the future” often washes away a number of worthy things and leaves a number of questionable items littering the shore.

In June 2012, Google demonstrated the power of “data-oriented deep learning” with one of the largest neural networks containing more than a billion connections.

A Stanford University–Google team (lead by Andrew Ng and Jeff Dean) showed the system images from 10 million randomly selected YouTube videos. One simulated neuron in the model fixated on images of **cats**. Others focused on **human faces**, **yellow flowers**, and other discrete objects.

The model identified reasonable well these discrete objects even though no humans had ever defined or labeled them.

Data science: translating is't easy...

How computers translate human language

Does grammar matter?

Data science: Deep learning...

in April 2013 [Jeff Dean presentation](#)

in April 2016 [Google DeepMind's artificial intelligence program AlphaGo seals 4-1 victory over Go grandmaster Lee Sedol](#)

Imagine there are two papers somewhere in the literature, one of which says that A implies B , and another that says B implies C . With the incredible growth of the scientific literature, it is likely that these two papers remain unrelated.

A program can find a way to stitch these two papers together, showing that A implies C , potentially an important discovery.

Is there a price in this finding? Cornell University program [EUREQA](#) is a free tool for detecting equations and hidden mathematical relationships in data with the goal “to identify the simplest mathematical formulas which could describe the underlying mechanisms that produced the data”.

A theorem may be proven in this way, but no one person actually may understand the proof, though there may be reasons to believe it is correct.

So what does this all mean for the future of truth?

Is it possible for something to be true but not understandable? Is this bad?

Believing without finding reasons is controversial. However, if these findings motivate the search for more elegantly constructed, human-understandable, versions of these proofs, then they are good.

Data science: The signal problem

Data are assumed to accurately reflect the “real world”; however, significant gaps, with little or no signal coming from particular parts may exist.

Boston has a problem with potholes, patching approximately 20,000 every year. **STREETBUMP** smartphone app passively detects and instantly reports potholes to the City. *A clever approach which has a signal problem.* People in lower income groups are less likely to have smartphones, or to use StreetBump and this is particularly true of older residents, where smartphone penetration is as low as 16%.

Smartphone data sets miss inputs from significant parts of the population.

Ramsey theory, named after the British mathematician, logician and philosopher Frank P. Ramsey, is a branch of mathematics that studies the conditions under which order **must** appear.

Problems in Ramsey theory typically ask questions of the form:

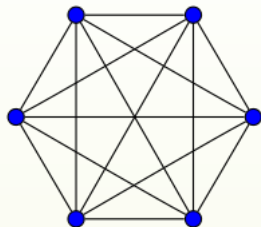
How many elements of some structure must there be to guarantee that a particular property will hold?

Suppose a party has six people. Consider any two of them.

They might be meeting for the first time—in which case we will call them *mutual strangers*; or they might have met before—in which case we will call them *mutual acquaintances*.

Theorem (Paul Erdős, Alfréd Rényi, Vera T. Sós): *In any party of six people either*

- ▶ *at least three of them are (pairwise) mutual strangers or*
- ▶ *at least three of them are (pairwise) mutual acquaintances.*



Party graph

This is the complete graph with six vertices in which every pair of vertices is joined by an edge. Every colouring of edges with red and blue, cannot avoid having either a red triangle or a blue triangle.

Proof. Choose any one vertex; call it P . There are five edges leaving P . They are each coloured red or blue. The *pigeonhole principle* says that at least three of them must be of the same colour.

Let A, B, C be the other ends of these three edges, all of the same colour, say blue. If any one of AB, BC, CA is blue, then that edge together with the two edges from P to the edge's endpoints forms a blue triangle. If none of AB, BC, CA is blue, then all three edges are red and we have a red triangle, namely, ABC .

When a set system is “quite big”?

Given a set X , a collection of subsets $S \subset 2^X$ is called *partition regular* if every set $A \in S$ has the property that, no matter how A is partitioned into finitely many subsets $A = C_1 \cup C_2 \cup \dots \cup C_n$, at least one of the subsets C_i must belong to the collection S .

The infinite pigeonhole principle. Partition regularity asserts that every finite partition of an infinite set contains an infinite set.

Proof: take S the collection of all infinite subsets of the infinite set.

Ramsey theory proves that

Complete disorder is an impossibility. Every large set of numbers, points or objects necessarily contains a highly regular pattern.

R. Graham, J. H. Spencer. [Ramsey Theory](#), *Scientific American* 262 no. 7 (1990), 112–117.

- ▶ How to distinguish correlation from causation?
- ▶ How to distinguish content-correlations from Ramsey-type correlations?

A growing list of over 24,000 spurious correlations in real databases:

- ▶ Total revenue generated by arcades (US) correlates with Computer science doctorates awarded (US). Correlation: 0.985065
- ▶ Per capita consumption of mozzarella cheese (US) correlates with Civil engineering doctorates awarded (US). Correlation: 0.958648
- ▶ Precipitation in Yakima County, WA correlates with Apple iPhone sales. Correlation: 0.993657
- ▶ Divorce rate in Maine correlates with Per capita consumption of margarine (US). Correlation: 0.992558
- ▶ People who drowned after falling out of a fishing boat correlates with Marriage rate in Kentucky. Correlation: 0.952407




Using classical results from ergodic theory, Ramsey theory and algorithmic information theory, the paper



C. S. Calude, G. Longo. The deluge of spurious correlations in big data, *Foundations of Science*, 2016, DOI [10.1007/s10699-016-9489-4](https://doi.org/10.1007/s10699-016-9489-4); also at <http://goo.gl/MHdRMJ>

proves that very large databases contain arbitrary correlations due to the size, not the nature, of data. They can be found in “randomly” generated, large enough databases, which implies that a) **they are spurious**, and b) **most correlations are spurious**.

Too much information tends to behave like very little information.
The scientific method can be enriched by computer mining in immense databases, but not replaced by it.

-  Jeffrey M. Stanton. *Introduction to Data Science*, 2012.
-  *Data Science: An Introduction*, wikibook.
-  *The Data Science Handbook*, 2015.

It is frequently claimed that randomness conflicts with free will because

[i]f our actions are caused by chance we lack control

and

[r]andomness, the operation of mere chance, clearly excludes control.

How “powerful” is this type of argument?

An **agent** is an entity that can make a choice (decision) in some context.

By **choice** we mean that the agent is able to pick an element from an abstract set of objects.

The **context** \mathcal{C} gives the environment and the relevant constraints for the choice to be made. A context may include the position in space and time where an agent must make a choice, as well as various constraints the objects should satisfy. The object that the agent picks is called the **chosen object in the context** \mathcal{C} .

Consider first a girl at a pet store looking to buy a pet. Here the agent can be taken to be the girl, the objects are the pets available for purchase in the shop, and the context is the shop at some time; the girl can choose any pet.

On the other hand, an individual animal in the shop can be an agent in the same context: the objects are the foods the animal can eat and a choice consists in selecting one of the available foods.

In the same context, the same animal can be an agent for the set of actions {eat, not eat}, where the choice consists in deciding on an action in that set.

The idea of context is crucially related to the idea of **possibility**. The choices the agent can make are assumed to be contextually possible, i.e. **possible in the given context**.

They may or may not be considered rational, morally justified or politically acceptable.

Contextual possibility should also be distinguished from other notions of possibility; for instance, an agent may have the ability to scuba dive, so that it is possible for the agent, given her abilities, to scuba dive, even though it is not possible in a given context because the agent is not close to water.

Contextual possibility is also different from a kind of intentional contextual possibility in which the agent has sufficient control over her ability to deliberately select certain outcomes (an ability Alfred Mele calls **intentional**).

Intentional contextual possibility implies contextual possibility, but the converse does not hold. For example, an agent can intentionally flip a coin, but while it is possible in the context for her to get heads she cannot intentionally get heads. It then also follows that she cannot freely choose to get heads. To understand why, we need to address the constraints needed in an account of free choice.

In the vast literature on free will a lot of misunderstanding comes from the fact that free will has been understood in numerous different ways.

In what follows we try to avoid confusion by working with a **simple, two-stage, contextual (not absolute) definition of free will.**

This is **not** the uniquely right definition: its goal is mainly to propose a more precise and detailed framework for studying the relation between free will, determinism, and randomness.

A definition of free will: intuition

The definition of free will models the idea that an agent has free will in some context if it has the ability to make a decision that is not completely determined by or the result of prior events.

An agent A acted freely in the context \mathcal{C} with respect to the set of objects $O_{\mathcal{C}}(A)$ if A could have acted differently to the way it did and A had full control over the outcome of the choice. This means that in the context \mathcal{C} :

- ▶ **Possibility Assumption (P)** The set of objects $O_{\mathcal{C}}(A)$ available to A contains at least two elements and every choice for A is possible in the given context.
- ▶ **Choice Assumption (C)** The agent A has full control over which of the objects in $O_{\mathcal{C}}(A)$ to choose.

The main point of **(P)** is that in context \mathcal{C} there are at least two objects in $O_{\mathcal{C}}(A)$ available to A to choose from. **(P)** thereby guarantees that the act of choice is meaningful and can lead to different outcomes, something that is impossible if $O_{\mathcal{C}}(A)$ has fewer than two elements.

- ▶ $O_{\mathcal{C}}(A)$ depends on the context \mathcal{C} and can vary with \mathcal{C} – by changing the context \mathcal{C} some elements may disappear and new elements can be added to $O_{\mathcal{C}}(A)$.
- ▶ The identity of $O_{\mathcal{C}}(A)$ may not be completely known by A : that is, A may not exhaustively know what, in the context, is possible for her.

The assumption **(C)** supplements **(P)** by claiming that the agent has full control over which option to choose.

A bit of philosophy of randomness

- ▶ True randomness does not exist mathematically.
- ▶ There are no random events in nature. Randomness is only a theoretical concept which is defined and produced in deterministic ways: it is not a direct cause of actions.
- ▶ There is a **process conception of randomness** based on which random actions are chancy actions.
- ▶ There is a **product conception of randomness** based on which random happenings are haphazard, lacking any discernible or explicable pattern.
- ▶ As the idea of a random event as the outcome of a chancy process faces a number of problems (which?), the **focus should be on the latter**.

How some philosophers see chance and randomness

The 5th BCE philosopher Leucippus wrote that

“Nothing occurs by chance, but there is a reason and necessity in everything.”

Similarly, under the influence of mathematician A. de Moivre, Hume called chance a mere word:

“... there be no such thing as Chance in the world.”

Randomness is incompatible with freedom: pure randomness argument

Arguments against the compatibility between free will and randomness based on “pure/true randomness” are **unsound** as they rest on vacuous concepts. An example is Hume’s and Schlick’s ontological thesis according to which there is nothing intermediate between chance and determinism.

In Eddington’s words: *There is no half-way house between random and correlated behavior. Either the behavior is wholly a matter of chance, in which case the precise behavior within the Heisenberg limits of uncertainty depends on chance and not volition. Or it is not wholly a matter of chance, in which case the Heisenberg limits . . . are irrelevant.*

Popper disagreed: *Hume’s and Schlick’s ontological thesis . . . seems to me not only highly dogmatic (not to say doctrinaire) but clearly absurd.*

Randomness is incompatible with freedom: using randomness violates **(C)**

A more interesting, but still **unsound**, argument is the following: randomness exists (in various degrees), so if an agent's actions are caused by randomness, the agent lacks control, so the assumption **(C)** is violated.

1. Assume **(P)**.
2. An agent A has free will with respect to $O(A)$ in the context \mathcal{C} if the assumption **(C)** is satisfied.
3. So A has ultimate control of which object in $O(A)$ to choose.
4. If the object was chosen randomly (to some degree), then no one had full control of which object in $O(A)$ was chosen.
5. Hence, A cannot not have full control on which object in $O(A)$ to choose.
6. Therefore, A has no free will with respect to $O(A)$ in the context \mathcal{C} .

Randomness is incompatible with freedom: using randomness violates **(C)**

Unfortunately, this argument does not offer a clear explanation of how the object selected was chosen sufficiently randomly to prevent *A* having control over its decision.

Rather, it is often claimed that **(P)** is inherently able to provide this. However, randomness does not “float around”, and is not something that is somehow “imposed on the agent”. Randomness is just produced, and then used by the agent in coming to a decision. To make random decisions the agent needs to use a random generator, which is a device producing random bits of a certain quality (but never “truly random bits”); there is no alternative.

Randomness is incompatible with freedom: using randomness violates **(C)**

According to our definition of free will, the detailed process used by the agent A to choose an object in some context \mathcal{C} where more than one object is available for selection is irrelevant: with **(P)** in place, all that is needed for her to choose freely is satisfaction of condition **(C)**. For the purposes of making a decision, using a random generator is no different than using the advice of a friend or getting more information from Wikipedia!

For clarification we look at the process of choosing at random. We henceforth assume that **(P)** is satisfied and consider first how a random generator may interact with an agent's decision making process, and second how the quality of the randomness generated may affect the agent's freedom. With the former in mind, we discuss four possible cases of interactions between the agent and the random generator.

Four ways to use randomness

1. (0) A uses G which outputs x , but ignores the output and picks an element of $O(A)$.
2. (1) A uses G which outputs x , and, after first deciding whether to use x to pick an element of $O(A)$, A then picks an element of $O(A)$.
3. (2) A uses G which outputs x and continues as follows:
 - ▶ A uses x to determine whether to use G or not,
 - ▶ depending on x , A makes no use of G or uses G to produce another (independent) output y which becomes its decision (in the last case G actually takes the decision on A 's behalf).
4. (3) A uses G which outputs x and its decision is x (G is used to take the decision on A 's behalf).

Four ways to use randomness

In case (0) it is clear that A 's freedom is not hindered by randomness. Still it is worth pointing out that there was a random element in her decision process (though it did not impact A 's decision).

To show that case (1) does not undermine A 's freedom, we restate that the information which A uses to make a decision is irrelevant to A 's decision being free. A merely asks G for advice. In fact cases (0) and (1) are identical if A picks something other than G 's output. In either of these cases however, A can consistently choose any element in $O(A)$ and, regardless of what G outputs, A has the final say on the decision. Hence, neither of these cases will disturb A 's freedom.

Four ways to use randomness

Case (2) is a hybrid between the earlier cases and the substantially more severe case (3). In (2) A operates G to generate a random bit. Depending on what this generated bit was, G either stops (leaving A to make the decision) or uses G to generate another random element of $O(A)$ and chooses this element on A 's behalf. Thus (2) will either reduce to (0) or (3) depending on the result of the first computation.

The point is that in cases (2) and (3), once A starts G , which object is chosen may potentially be decided by G rather than A . Is **(C)** fulfilled in these cases? Does the quality of random bits matter (e.g. if they form an incomputable sequence)?

Four ways to use randomness

The reason these cases seem to violate assumption **(C)** is because the agent gives up its final decision, not because the agent gives up its final decision to a random process. Asking another agent to make a decision on its behalf is no different than asking a random generator. Notice that whether A retains its freedom in asking another agent B to make its decision, is a delicate issue, one which is, in practice, judged on a case by case basis. This shows that from the point of view of free choice, the role of B is as detrimental to A 's freedom as the role of G .

Giving up freedom to randomness is as harmful as giving up freedom to any other agent: A retains its freedom when it gives up its decision to B if and only if A retains its freedom when it gives up its decision to G .

Four ways to use randomness

Similarly, even if A does not voluntarily give up its decision, another agent B choosing an element of $O(A)$ is no different than some random generator G doing the same. Thus, the fact that indeterminism allows for randomness should not lead us to conclude that free will is impossible any more than the fact that there are other free agents which are capable of choosing on behalf of others.

Randomness is compatible with free will

We showed that **randomness is compatible with free will so long as free will is itself metaphysically possible.**

Our arguments **are relative and do not answer** the main philosophical question about free will, namely does free will exist and, indeed, can it exist?

For more details see: C.S. Calude, F. Kroon, N. Poznanović. **Free Will Is Compatible With Randomness**, *CDMTCS Research Report* 461, 2015.