Complexity Beyond Computation

Liu Yuting

December 15, 2025

1 About Complexity

To appreciate the study of complexity, we may start with what Feynman held as the most important knowledge humans possess: Everything is made of atoms. Atoms form molecules. Molecules form biological cells. Biological cells form organs. Organs form the human body. The human body consists of vast amount of atoms working magically together. The study of complexity is to explain that magic.

So far, no scientific results fully achieve the goal set in the opening paragraph. Although there are claims that some notion solves the entire mystery of complexity, all are baseless and we will refute one such claim in this manuscript.

1.1 Observations

Although the simplicity of modern user interfaces hides great amount of complexities from us, we sometimes still experience the consequences of complexity. Weather can be unpredictable. Financial markets can be volatile. The fate of the world can be uncertain.

Sciences have found ways to cope with complexities. Axiomatic systems derive complex statements from simple axioms. Complex motions may be reduced to a set of physical laws. Darwinian evolution is said to account for the complexities of life. These explanations, though successful, all suffer deficiencies. Gödel proved the limitation of axiomatic systems in deriving all mathematical truths in second order arithmetic. Quantum mechanics and general relativity can not easily reach coherence with each other. The similarity between certain biological forms and automata seems to suggest life has a algorithmic aspect not explored by Darwin.

It's against this unsatisfactory background that modern complexity studies emerged. Compared with scheme theory in algebraic geometry, complexity studies feel like mud. It's not because the subject matter isn't attractive, but we don't have very great tools at hand.

1.2 Modern Complexity Studies

Because there is no consensus how to define complexity, modern complexity studies diverge in subject matter as well as methodology. For some, complexity means randomness. For others, it means logical depth. The lack of agreement on the definition of complexity makes summarizing the literature difficult, so here only a few well-established approaches are described.

1.2.1 Chaos

The study of chaotic dynamics features unpredictability and randomness in solutions to differential equations or iterative maps, that their trajectories are highly sensitive to initial conditions. It's often said that the long-term behavior of chaotic orbits is unpredictable, but there are many possibilities what 'long-term' means. From time to time, it may be 'eventually' or 'exponentially' depending on the dynamical system under consideration.

Although chaotic orbits may seem a mess, hope to gauge their evolution isn't lost. Occasionally, the existence of bounds, topologies, or geometries can be proven to describe the long-term stability of chaotic orbits, that is, they may be unpredictable, but won't fall apart.

A interesting question in chaos theory is when and how a system transitions from order to chaos. Although it's hard to answer such questions, measures of order/disorder can often be found so that we can see a transition indeed occurs.

Probably one of the most astonishing results in chaos theory is Sharkovskii's Theorem, that there is a hierarchy of periods in chaotic maps. Despite being in disorder, there is still structure!

Overall, chaos theory's contribution to complexity study is how unpredictability happens. There is no simple way to accurately pin down the long-term behavior of chaotic orbits.

1.2.2 Computational Complexity

The main measure of computational complexity is logical depth, that a complex procedure requires more logical steps than a simple one. Although the definition may seem elementary, the strength of computational complexity is greatly amplified by the existence of universal Turing machines. Since universal Turing machines can simulate all computations, there is a unified method to generate all complexities in this characterization.

A obvious application of computational complexity is cryptography. A encryption method is good if it requires astronomical logical steps to break it.

But, the computational approach runs much deeper into sciences. Take biology for example. Since genetic materials are discrete data structures, many biological patterns found in nature resemble the result of a algorithm applied to genetic data. In fact, thinking in terms of code has been hugely successful in molecular biology, that DNA codifies proteins.

Measuring in logical depth, the computational approach allows a systematic classification of complexity in terms of *P*, *NP*, *PSPACE*, etc. The class *NP* is particularly interesting as there are a plethora of *NP*-complete problems that if one of them falls in *P*, the entire *NP* is equal to *P*.

1.2.3 PDE

The PDE approach to complexity study is often based on physical laws. Whether it's linear Schrödinger equation, or nonlinear Navier–Stokes equations, PDE gives a concise rule how such systems assemble, interact, and evolve.

Contrary to the computational approach, one significant drawback of the PDE approach is that there is no unified machinery in finding solutions. However, whenever applicable, like in the case of Schrödinger equation, the solution often provides great insights into how complexity arises. Indeed, the whole study of chemistry may be put on a solid foundation of quantum mechanics.

The difficulty in solving PDEs indicates that nature appears to be infinitely subtle, capable of effortlessly generating solutions with high accuracy. Why can nature solve PDEs that beset mathematicians?

1.3 Sense of Complexity

Since there is no agreement what complexity is, it's impossible to satisfactorily categorize the concept. Instead, there are sensible features when complexity is encountered. Here, a few such features are discussed.

1.3.1 Hierarchy of Hardness

There is a hierarchy of hardness in machineries. If a machine A can do whatever another machine B can do and more, then A is more complex than B. A example is the progression from linear bounded automata to Turing machines. A partial order system is naturally formed according to the functionalities of machines.

1.3.2 Nonlinearity

Nonlinearity is a very broad term. Usually, nonlinear systems are complex because the superposition of states isn't valid. These systems can generate complex output from simple input. A particular example is cellular automaton. There are many patterns in nature that resemble cellular automata.

1.3.3 Unpredictability and Randomness

The logistic map exhibits period doubling to chaos. The increase in unpredictability and randomness signals increase in complexity. Shannon entropy and Kolmogorov complexity are useful measures of randomness.

2 The Computational Paradigm

There are many approaches to complexity study. However, one approach, that of computation, finds power to greatly expand into other areas.

2.1 Advantages of Computational Complexity

To understand the success of the computational paradigm, it requires a lot of technical work. Here a brief explanation is provided to clarify how two previously mentioned approaches to complexity study, chaos and PDE, may be covered by computation.

At first glance, the divergent behavior paramount in chaos seems to prohibit all sorts of computational approach. However, with the aid of topological estimates like the shadowing lemma, one may prove that although the true trajectory of a given initial condition may not be computed, a computed trajectory may still be very close to a true trajectory of a very close initial condition. Thus, computation tells something about the system.

As for PDE, the success of computational fluid dynamics not only find applications in vehicle designs, but more importantly, demonstrates the need for more powerful computers to solve more complex CFD problems. There seems to be a correspondence between computational power and complexity of the problem under consideration.

Overall, computation provided concrete solutions to abstract theories. In this way, it gradually became the primary tool for complex study.

2.2 Principle of Computational Equivalence

With the establishment of the computational paradigm, radical ideas began to emerge. One such idea is Stephen Wolfram's Principle of Computational Equivalence.

While traditional sciences view computation as a approximation to a specified model, Wolfram views computation as essential, and traditional models are approximations. The tremendous power of this formulation lies in the fact that there are universal Turing machines that can simulate all computations. Therefore, there is a unified mechanism to generate all complex phenomena.

However, this seems too good to be true, and it's the deficiencies in Wolfram's Principle of Computational Equivalence that lead to considerations beyond the computational paradigm.

3 Beyond the Computational Paradigm

To put it simply, if complexities are computations, then life can be generated on a ink-and-tape Turing machine, which is absurd. Here technical reasons why Principle of Computational Equivalence is wrong are given.

3.1 Deficiencies of the Computational Paradigm

Pretty much of the deficiencies of the computational paradigm can be justified by common sense. Though, it's not clear what's the path forward for the definition of complexity.

3.1.1 Cracks from Within

The computational paradigm highlights the importance of Turing-completeness. However, Turing-completeness alone suffers scale problems. For example, both ink-and-tape Turing machines and sophisticated semiconductor chips can perform identical computations, but semiconductor chips are a lot more complex and much faster. The computational paradigm doesn't address the discrepancy. There are serious implications. If computers were all in the form of ink-and-tape Turing machines, there would be no iPhone.

Quantum computation provides another critique of the computational paradigm. Although both classical and quantum computers can be Turing-complete, there are quantum algorithms that run much faster than classical algorithms. That is, time complexity much depends on architecture.

3.1.2 Cracks from Without

Modern spacecrafts have computer chips. However, it's a fallacy to declare that these spacecrafts are simply computers. Ordinary computers don't go to Mars. Therefore, although ordinary computers and spacecrafts may possess identical computational power, their complexities can be quite different. While ordinary people can perform computations like computers, albeit inefficiently, most people don't have the knowledge of designing complex spacecrafts for Mars.

Specifically, Principle of Computational Equivalence ignores statistical concepts like evidence and geometrical concepts like dimension. It claims that they are irrelevant if complexity is the main concern. Going to Mars is simply as complex as pencil-and-paper computation, which is absurd.

In fact, recent research showed that knots may be put into a partial order system according to some definition of complexity called ribbon concordance. It uses basic tools like Morse theory. It's hard to dismiss such results as being irrelevant.

3.2 A Application

While the computational paradigm provides complexity hierarchy according to computational power, it says nothing about the possible interaction between computers of different capabilities. Such interactions are possible because of computers can be topologically separated. It's these interactions that made the design of Apple's Secure Enclave, and similar constructs, possible. Therefore, going beyond the computational paradigm can be fruitful.

3.3 Summary of the Status Quo

The status of complexity study is still kind of muddy. No universal definition of complexity can be given, no great tools can be applied, and no approach is satisfactory. But if we are going to understand how the world works as a whole, complexity study must be undertaken.

4 Relative Complexity Theory

Here we introduce relative complexity theory that overcomes the defects of computational complexity theory. It's summarized in a post on Glacier Studio Blog.

4.1 Blog Post

Complexity Theory, April 13, 2024 Our goal is to produce a complexity theory that properly addresses the role played by computation, statistics, and geometry. It's not a trivial task. As a first step, a outline of relative complexity is provided here to generalize Turing machines, so that computational complexity becomes a special case.

We postulate that complexity depends on what cognitive instrument is at work. A task that appears complex to a Turing machine may not be complex to a probability process at all. A task that can not be performed by a machinery is called too complex with respect to it. We always speak of complexity with respect to a machinery.

Machineries may be put into a partial order system according to what tasks they can perform. So, we can say that a Turing machine with a stochastic register is more complex than the Turing machine part, because there are tasks beyond computation.

Logical depth can be put into this picture by considering performing computation on a Turing machine within certain number of steps. Kolmogorov complexity can be put into this picture by considering what output a Turing machine can generate given certain length of input.

This theory can handle the following important case. Theories with different geometrical structures may be Turing equivalent, but some are chiral, while others are not. If we make the distinction between these machineries, we may say chirality is too complex for a simple universal non-chiral cellular automaton.

This amounts to the fact that Wolfram's classification of processes into computational machineries with equivalent computational power is too crude. They tell no difference between chiral and non-chiral theories.

Furthermore, we define easy as simple to do, hard as complex to do. A icon simple to recognize may be complex to produce, which reflects the more vague experience that simplicity isn't always easy to achieve. Clearly, we need more dimension theory to account for it in our complexity theory.

All the essential materials here are put forward in a form or another long ago. Since recently there is renewed public interest in complexity theory, we say it again so that the topic may be more broadly understood and confusion may be avoided. We are tired of unnecessary and unjust damages from public misunderstanding.

4.2 Elaboration

We explain what the blog post means.

4.2.1 Machineries and Tasks

A machinery is a well-defined object, like a stochastic register or a Turing machine.

A task is a true or false statement about a machinery. A task a machinery can perform is a true statement about the machinery. A task a machinery can not perform is a false statement about it, in which case, we also say that the task is too complex for the machinery.

If we select a certain set of tasks, there may be additional structure like poset. For example, a machinery that can perform task A and task B is said to be more complex than a machinery that can perform task A but not task B, with respect to the task structure. In this way, we say a Turing machine with a stochastic register is more complex than the Turing machine part.

4.2.2 Reproducing Computational Complexity

Let M be a universal Turing machine, and $M(n)$ be M running in less than or equal to n steps. Tasks M can perform are computable functions. Logical depth is reproduced when we consider $M(n)$. For input size m, there is a minimal n_m such that $M(n_m)$ can perform the desired function, which means M can perform the function in $O(n_m)$ steps.

4.2.3 Geometry

Some machineries are endowed with geometric structures in addition to computational structures, like cellular automata.

Chirality, March 19, 2025 To explain chirality, just notice that a trefoil is not isotopic to its reflection. In the case of cellular automata, non-chirality can be defined as the condition that the rules are invariant under reflection, so for all configurations, the reflection of a evolution is still a evolution of the same rule. A chiral cellular automaton is not non-chiral. Game of Life is non-chiral. Rule 110 is chiral. By appropriate embedding, one can construct higher dimensional chiral cellular automata, as well as non-chiral ones, from lower dimensional rules. Since both chiral and non-chiral cellular automata can be Turing-complete, chirality is not explained by computation. We can say chirality is too complex for Game of Life. One of the consequences of our new relative complexity theory is that sciences like medicine are not completely explained by computation. For example, reflection of a molecule may no longer be effective medical treatment. Of course, computation is useful, but Wolfram is wrong to say it's everything.

Chirality finds application in physics. The Standard Model is a chiral theory. While one may use non-chiral theories like electromagnetism and chiral theories like electroweak interactions to perform universal computation, our relative complexity theory can tell the difference in chirality, but Wolfram's classification can not.

4.3 Applications

We outline several applications of relative complexity theory.

4.3.1 Beyond Generative Grammar

Gödel's incompleteness theorem may be reformulated in relative complexity theory, that is, if a effectively axiomatized theory is consistent and contains Peano arithmetic, proving its own consistency is too complex for the theory. This led to a refutation of generative grammar.

Beyond Generative Grammar, September 15, 2018 The complexity of human language together with the success of modern computational complexity research seduced many scientists into believing that there is a computationally enumerable generative grammar for human language, even innate, up to convention. However, ordinary human mind, given infinite logical depth, is Turing-complete, complex enough to simulate all Turing machines. Thus, the generative grammar doctrine demands that a particular computable enumeration is favored by nature at the expense of all other computable enumerations. Does nature embrace grammatical favoritism?

Arithmetic statements may be formulated with human language notation. A true arithmetic statement may be considered grammatical under the context of number theory. It follows that the generative grammar doctrine entails a computable enumeration E for entire human knowledge about arithmetic.

Suppose that computational logical depth is not a material constraint. All human knowledge about arithmetic may be enumerated by E. Because E is composed of true arithmetic statements, E can not include the statement G that E is consistent, due to Gödel's incompleteness theorem for second order arithmetic. The generative grammar doctrine declares that G can not be learned, which sounds dubious enough, for readers already learned it here.

It's natural to speculate that E doesn't exist under the context of number theory, provided with sufficient computational resource and logical depth, or reasonable size limit on E being humanly, not astronomical.

Generative grammar scientists, with luck, might discover sorts of statement generators resembling true or false arithmetic statements and allow us to run them in order to determine E.

Suppose we articulate and learn that the statement generator is consistent, before the determination of E, even though we don't know whether the statement generator is actually consistent or not.

If the statement generator is inconsistent, it will prove both its own consistency and inconsistency in finite logical depth. It follows that the statement generator is not E. On the other hand, if the statement generator is consistent and generative grammar scientists identify it with E, E will not prove its own consistency in finite logical depth, according to Gödel's incompleteness theorem. Thus, the learned statement G, that the statement generator is consistent, can not be derived from E. Generative grammar scientists claim that the successfully learned statement G is not learnable, which is absurd and proves that E doesn't exist under the context of number theory.

To be fair, there may exist humans of limited mental capacity that G is not learnable. However, the existence of these humans deals another fatal blow to generative grammar scientists who claim that humans share a mother tongue. If G is learnable for some, but not others, G can not possibly be a shared feature derived from common human genetic factors.

Generative grammar scientists may try to exclude arithmetic from human language, but since arithmetic is expressible with ordinary human language, the exclusion elicits generative grammar scientists' explicit denigration of human creativity.

4.3.2 Computability

Orbits and Computation, May 28, 2025 Demis Hassabis suggested that simply because you can approximate nature with computation, it's evidence that nature is computational. However, every good student who studied dynamical systems knew that's wrong.

Shadowing lemma allows you to approximate a orbit computationally, but Smale's Horseshoe generates orbits corresponding to real numbers of which computables are measure zero, that is, generic orbits aren't computable.

4.3.3 Quantum Computers

Conventional quantum computers only use quantum mechanics to speed up computation. Thus, the results about computability still hold for quantum computers.

4.3.4 Ribbon Concordance

As a element in a poset can be defined by the set of elements less than or equal to the element, ribbon concordance can be reformulated in relative complexity theory by considering the collection of tasks for a knot K that there exists a ribbon concordance from knot K' to K.

Generalized Algorithms, August 30, 2024 People have been using the word algorithm vaguely for centuries. Turing defined it as a Turing machine that halts on all inputs. The entirety of computational complexity theory was built upon it.

It's therefore natural that our relative complexity theory requires a generalization of algorithms. While exact definitions are more precise, it's better just to think of it as a recipe for operating a machinery here.

The generalization is not without past attempts. Since the arrival of parallel computing, it became clear that thinking in terms of a Turing machine was not enough. But the proposed definitions were all related to Turing machines, and not very general.

A benefit of thinking in terms of generalized algorithms is that geometrical processes find natural expressions. Ribbon concordance is a machinery to produce more and more complex knots, for example. A algorithm can mean to derive a knot with a Morse function.

As we have seen in relative complexity theory, computational complexity is not a satisfactory characterization of real world complexities. By introducing generalized algorithms, we can talk about the complexity of real world processes in very specific language. It's a rich new field of study.

5 Technicals

A few technical statements used in the previous sections are listed here.

5.1 Logic

In the discussion of machineries and tasks, we said a task is a true or false statement about a machinery. To be precise, it's a predicate $P(x)$ for machinery X, such that $P(X)$ is true or false.

By second order arithmetic, we mean the induction schema in Peano arithmetic is a second order axiom, rather than using the full machinery of second order theories.

5.2 Dynamical Systems

In the case of Smale's Horseshoe, shadowing lemma is not needed to prove computational approximations. It's simply the approximation of a real number by a rational number.

5.3 Cellular Automata

A discussion of cellular automata may be found in *A New Kind of Science* written by Stephen Wolfram.

5.4 Ribbon Concordance

Here our definition is the reverse of the usual definition, but since the reverse of a Morse function is still a Morse function, we simply change the index and all arguments follow through.