Proceeding Paper

Undecidability and Complexity for Super-Turing Models of Computation †

Eugene Eberbach

Department of Engineering and Science, Rensselaer Polytechnic Institute, Hartford, CT 12180, USA; eeberbach@gmail.com

† Presented at the Conference on Theoretical and Foundational Problems in Information Studies, IS4SI Summit 2021, online, 12–19 September 2021.

Abstract: It seems that intelligent complex systems will require formalisms having richer behavior than Turing machines. Very little is known about the relations (e.g., the expressiveness and/or effectiveness) between new super-Turing models of computation. The objective of this paper is an attempt to establish a hierarchy of expressiveness of super-Turing models. Truly, a new theory of undecidability and complexity for super-Turing models has to be developed. Some preliminary steps have been done in this paper by introducing a-decidable and i-decidable algorithms and U-complete, D-complete, and H-complete complexity classes that were inspired by NP-complete and PSPACE-complete classes for intractable problems. This paper should be understood as a preliminary step leading to feasible approximate solutions of Turing machine undecidable problems, in a similar way as approximate, randomized, and parallel algorithms allow for feasible solutions for intractable problems.

Keywords: undecidability; intractability; super-Turing computation; super-recursive algorithms; a-decidability; i-decidability; reduction techniques; U-completeness; D-completeness; H-completeness

check for updates

Citation: Eberbach, E. Undecidability and Complexity for Super-Turing Models of Computation. *Proceedings* **2022**, *81*, 123. https://doi.org/10.3390/proceedings2022081123

Academic Editor: Mark Burgin

Published: 30 March 2022

1. Introduction: Beyond Turing Machines and Recursive Algorithms

Turing machines [1,2] and algorithms are two fundamental concepts of computer science and problem-solving. Turing machines describe the limits of problem-solving using conventional recursive algorithms and laid the foundation of current computer science in the 1960s.

Note that there are several other models of algorithms, called super-recursive algorithms that can compute more than Turing machines, using hypercomputational (also called super-Turing) models of computation [3–6].

It turns out that (TM) undecidable problems cannot be solved by TMs and intractable problems are solvable but require too many resources (e.g., steps or memory). For undecidable problems, effective recipes do not exist, problems are called nonalgorithmic or nonrecursive. On the other hand, for intractable problems, algorithms exist, but running them on a deterministic Turing machine requires an exponential amount of time (the number of elementary moves of the TM) as a function of the TM input.

2. Super-Turing Models of Computation

We use the simplicity of the TM model to formally prove that there are specific problems (languages) that the TM cannot solve [7]. Solving the problem is equivalent to deciding whether a string belongs to the language. A problem that cannot be solved by a computer (Turing machine) is called undecidable (TM undecidable). Unfortunately, the Turing machine model is not sufficient for many domains. In solving real problems, we work with computational models (or invent new ones) that are more convenient and appropriate for the domain. Such new types of computation and computational models are often called hypercomputational or super-Turing models of computation.

Definition 1. (On hypercomputation/super-Turing computation) *By super-Turing computation (also called hypercomputation), we mean any computation that cannot be carried out by a Turing machine as well as any (algorithmic) computation carried out by a Turing machine.*

The above definition is consistent with Wikipedia's definition that states that hypercomputation or super-Turing computation refers to models of computation that go beyond, or are incomparable to, Turing computability (see also [3–6]).

Super-Turing models derive their higher than the TM expressiveness using three principles: Interaction, evolution, or infinity:

- In the interaction principle, the model becomes open and the agent interacts with either a more expressive component or with infinitely many components.
- In the evolution principle, the model can evolve to a more expressive one using nonrecursive variation operators.
- In the infinity principle, models can use unbounded resources: Time, memory, the number of computational elements, an unbounded initial configuration, an infinite alphabet, etc.

The details can be found in [4,5].

Definition 2. (On time/space complexity) *We can define four classes of problems/languages to decide strings in the language in the order of increasing complexity:*

1. *The number of steps/memory cells is polynomial in the problem size and problems will be called polynomial decidable (p-decidable).*
2. *The number of steps/memory cells is exponential in the problem size and problems will be called exponentially decidable (e-decidable).*
3. *The number of steps/memory cells is infinite but computed in a finite time/finite number of cells, i.e., asymptotically decidable in limit (a-decidable) (analogy: Convergent infinite series, a mathematical induction, computing an infinite sum in a definite integral).*
4. *The number of steps/memory cells is infinite and requires an infinite time/infinite number of cells to decide strings in the problem size, i.e., infinitely decidable (i-decidable) (undecidable in the finite sense).*

The classical complexity theory usually covers classes (1) as easy/tractable and class (2) as intractable problems (but both decidable). Class (3) is an intermediate class because although technically it requires an infinite number of steps (or memory cells for space complexity), we can find a solution in the limit using finite resources. Its infinite computations are decided in a finite number of steps or require a finite amount of memory and are represented by inductive Turing machines, accelerating Turing machines, persistent Turing machines, inductive machine learning, anytime algorithms, evolutionary algorithms, neural networks or $-calculus. Class (4) requires infinite resources and is unsolvable by Turing machine (but solvable by hypercomputers using infinite resources). Classes (2), (3), and (4) cover nonpolynomial algorithms, classes (3) and (4) belong to super-recursive algorithms [4] (see also Wikipedia). Class (1) and (2) cover recursive algorithms.

Obviously, undecidable problems are characterized by computations growing faster than exponentially (i.e., hyper-exponentially) or they may have even an infinite computational complexity (an enumerable or real numbers infinity type). For relations between undecidability and complexity, look, for instance, to [8,9].

In [4,5], several super-Turing models have been discussed and overviewed. An incomplete list includes:

- Turing's o-machines, c-machines, and u-machines (Turing, A.). They use the help of Oracle (o-machines) or human operator (c-machines), or they form an unorganized network that may evolve by genetic algorithms or reinforcement learning (u-machines),
- Cellular automata (von Neumann, J.), an infinite number of discrete finite automata cells in a regular grid,

- Discrete and analog neural networks (Garzon, M.; Siegelmann, H.), a potentially infinite number of discrete neurons or neurons with true real-valued inputs/outputs,
- Interaction machines (Wegner, P.). They interact with other machines sequentially or in parallel by infinite multiple streams of inputs and outputs,
- Persistent Turing machines (Goldin, D.). They preserve contents of memory tape from computation to computation,
- Site and Internet machines (van Leeuwen, J.; Wiedermann, J.). They have input/output ports that allow to interact with an environment or Oracle and communicate by infinite streams of messages,
- The π-calculus (Milner, R.) potentially an infinite number of agents interacting in parallel by message-passing,
- The $-calculus (Eberbach, E.), potentially an infinite number of agents interacting in parallel by message-passing and searching for solutions by built-in kΩ-optimization meta-search that may evolve,
- Inductive Turing machines (Burgin, M.). They may continue computation after providing the results in a finite time,
- Infinite time Turing machines (Hamkins, J.D.). They allow an infinite number of computational steps,
- Accelerating Turing machines (Copeland, B.J.). Each instruction requires half of the time of its predecessor's time forming a geometric convergent series,
- Evolutionary Turing machines (Eberbach, E.) and evolutionary automata (Eberbach, E.; Burgin, M.). They use an infinite chain of abstract automata that may evolve in successive generations and communicate by message-passing (an output becomes an input to a next generation).

3. Three New Classes of TM Undecidable Problems

We introduce three new classes of TM undecidable problems, inspired by the Cook/Levin NP-complete class definition [7]. The new complexity classes will be defined in the order of growing undecidability based on [10], however, using new modified definitions.

Examples of typical unsolvable problems [7] include the universal language L_u (of the Universal Turing Machine), the diagonalization language L_d, nonenpty L_{ne} and empty L_e TM langugaes, Post Correspondence Problem (PCP), Busy Beaver Problem (BBP), the Economy Collapse Problem (ECP), the Economy Immortality Problem (EIP).

Now we are ready to introduce three new classes of TM undecidable problems.

Definition 3. (on U-complete languages) *We say a language L is U-complete (Universal Turing Machine complete) if*

1. *Any word w can be decided in a finite number of steps if $w \in L$, or it requires an infinite number of steps if $w \notin L$ (semi-decidability condition).*
2. *For any language L′ satisfying (1), there is p-decidable or e-decidable reduction of L′ to L (completeness condition).*

U-complete languages belong to the *RE-nonREC* class. Examples of U-complete languages include L_u (a basic representative to call the whole class), PCP, L_{ne}, BBP, ECP, planning problem, optimization problem. The **U-hard languages**, a superset of U-complete languages, satisfy only the completeness condition from the above definition.

Definition 4. (on D-complete languages) *We say a language L is D-complete (Diagonalization complete) if*

1. *Any word w from L cannot be decided in a finite number of steps (undecidability condition).*
2. *For any language satisfying (1), there is p-decidable, or e-decidable reduction of L′ to L (completeness condition).*

Examples of D-complete languages include L_d (a basic representative to call the whole class), L_e, EIP, complement of L_d, complement of L_u, complement of BBP. The D-hard languages, a superset of D-complete languages, satisfy only the completeness condition from the above definition.

Definition 5. (The hyper-diagonalization language) *The hyper-diagonalization language L_{hd} consists of all strings w such that TM M whose code is w will not accept even in an infinite number of steps when given w as input.*

Definition 6. (on H-complete languages) *We say a language L is H-complete (Hypercomputation complete) if*

1. *Any word w from or outside of L cannot be decided in a finite number of steps (undecidability condition).*
2. *For any language L′ satisfying (1), there is an a-decidable or i-decidable reduction of L′ to L (completeness condition).*

A canonical representative of this class is L_{hd}. The H-hard languages, a superset of H-complete languages, satisfy only the completeness condition from the above definition.

4. Terminal Languages and Expressiveness of Evolutionary Automata and Interaction Machines

Definition 7. *A word w is accepted in the terminal mode of the automaton E if given the word w as input to the automaton E, there is a number n such that the automaton A[n] from E comes to an accepting state.*

Definition 8. *The terminal language TL(E) of the automaton E is the set of all words accepted in the terminal mode of the automaton E.*

It has been proven that evolutionary automata (e.g., evolutionary Turing machines or evolutionary finite automata) and interaction machines accept arbitrary languages over a given alphabet [11,12]. From the above, we get Theorem 1:

Theorem 1. *Terminal languages of evolutionary automata and interaction machines coincide with the class of all languages in the alphabet X.*

5. Expressiveness of o-Machines, Site and Internet Machines, $-Calculus, π-Calculus, Cellular Automata, Neural Networks and u-Machines

We can safely assume that models based on Oracles, i.e., Turing o-machines [2] and site and Internet machines, can also accept arbitrary languages over a given alphabet.

Theorem 2. *Terminal languages of o-machines and site and Internet machines coincide with the class of all languages in the alphabet X.*

We believe that analogous proofs can be derived for $-calculus [13], π-calculus (pending that replication operator allows for infinity), cellular automata (extended to random automata networks, where each cell may represent a different finite state automaton), neural networks, Turing u-machines (pending that they allow for an infinite number of nodes), i.e., models where we can derive the sequence of components inheriting all needed information from their predecessors, i.e., we can repeat essentially the proofs for evolutionary automata and interaction machines. Thus, we will write, skipping the proofs, the following.

Conjecture 1. *Terminal languages for $-calculus, π-calculus, cellular automata generalized to random automata networks, neural networks and Turing u-machines coincide with the class of all languages in the alphabet*

From Theorems 1 and 2 and Conjecture 1, we can immediately derive Corollary 1.

Corollary 1. *Expressiveness of $-calculus, n-calculus, cellular automata, neural networks, Turing o-machines and u-machines, evolutionary automata, and interaction machines is the same and allows to accept all languages over a given finite alphabet.*

6. Expressiveness of Other Super-Turing Models and Relations with U-Complete, D-Complete and H-Complete Complexity Classes

It is not clear at this moment how to classify expressiveness of Infinite time Turing machines and accelerating Turing machines. Simply, the conditions of an infinite number of steps or doubling the speed of each successive step alone seem not to be sufficient to prove that those models can accept all languages over a given alphabet. Similarly, we do not have enough details on c-machines because they were only briefly mentioned in the original paper on Turing machines. Moreover, we cannot properly classify at this moment the expressiveness of inductive Turing machines and persistent Turing machines in the form of the stand-alone components. However, it is clear that they, as components of evolutionary automata or interaction machines, may achieve such enormous expressiveness of their hosts.

On the other hand, from Corollary 1, we can conclude the following.

Corollary 2. *Turing o-machines and u-machines, site and Internet machines, $-calculus, π-calculus, cellular automata, neural networks, evolutionary automata and interaction machines accept all U-complete, D-complete, and H-complete languages.*

7. Conclusions

We proposed three new definitions of complexity classes for TM undecidable problems: U-complete, D-complete, and H-complete languages. We also proved that several super-Turing models of computation can accept all languages (including those not accepted by Turing machines) over a given alphabet.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: Not applicable.

References

1. Turing, A. On Computable Numbers, with an Application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **1936**, *42-2*, 230–265; Correction in *ibid* **1937**, *43*, 544–546. [CrossRef]
2. Turing, A. Systems of Logic based on Ordinals. *Proc. Lond. Math. Soc. Ser.* **1939**, *45*, 161–228. [CrossRef]
3. Burgin, M. *Super-Recursive Algorithms*; Springer: New York, NY, USA, 2005.
4. Eberbach, E.; Wegner, P. Beyond Turing Machines. *Bull. Eur. Assoc. Theor. Comput. Sci. (EATCS Bull.)* **2003**, *81*, 279–304.
5. Eberbach, E.; Goldin, D.; Wegner, P. Turing's Ideas and Models of Computation. In *Alan Turing: Life and Legacy of a Great Thinker*; Teuscher, C., Ed.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 159–194.
6. Syropoulos, A. *Hypercomputation: Computing Beyond the Church-Turing Barrier*; Springer: Cham, Switzerland, 2008.
7. Hopcroft, J.E.; Motwani, R.; Ullman, J.D. *Introduction to Automata Theory, Languages, and Computation*, 3rd ed.; Addison-Wesley: Boston, MA, USA, 2007.
8. Burgin, M. Algorithmic complexity of recursive and inductive algorithms. *Theor. Comput. Sci.* **2004**, *317*, 31–60. [CrossRef]
9. Burgin, M. Algorithmic complexity as a criterion of unsolvability. *Theor. Comput. Sci.* **2007**, *383*, 244–259. [CrossRef]
10. Eberbach, E. On Hypercomputation, Universal and Diagonalization Complete Problems. *Fundam. Inform.* **2015**, *139*, 329–346. [CrossRef]
11. Burgin, M.; Eberbach, E. Evolutionary Automata: Expressiveness and Convergence of Evolutionary Computation. *Comput. J.* **2012**, *55*, 1023–1029. [CrossRef]
12. Wegner, P.; Eberbach, E.; Burgin, M. Computational Completeness of Interaction Machines and Turing Machines. In *Turing-100. The Alan Turing Centenary*; Voronkov, A., Ed.; Volume 10, pp. 405–414. Available online: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.432.2685&rep=rep1&type=pdf (accessed on 28 February 2022).
13. Eberbach, E. The $-Calculus Process Algebra for Problem Solving: A Paradigmatic Shift in Handling Hard Computational Problems. *Theor. Comput. Sci.* **2007**, *383*, 200–243. [CrossRef]