Technical Report No. 2020-650
A MAP OF ENGLAND, THE SIMULATOR SIMULATED, AND NONUNIVERSALITY IN COMPUTATION

Selim G. Akl
School of Computing, Queen's University
Kingston, Ontario K7L 3N6, Canada
akl@cs.queensu.ca

April 30, 2020

Abstract

A *reductio ad absurdum* argument is used to establish that a universal computer cannot exist. The proof uses a recursive process whereby an assumed universal computer attempting to simulate itself descends into an infinite regress. Appealing to a more powerful computer to do the simulation leads to the same unavoidable conclusion.

Keywords and phrases: Church-Turing thesis, simulation, universal computer, nonuniversality in computation, map within a map, proof by contradiction, proof by counterexample, recursion, self-reference, infinite regress.

1 Introduction

In 1899, American philosopher Josiah Royce (1855-1916) proposed an interesting thought experiment [13]. Imagine, he suggested, we could draw a detailed map of England. The map would be so precise as to contain every road, every river, every hill, every plain, and so on. A flat terrain will be chosen and the map inscribed on the surface of the English countryside. But then something unexpected would happen. Since the map is now part of the landscape, in order for it to be exact, it would necessarily have to contain a copy of itself. That copy would therefore inevitably contain a copy of itself as well, and the copy of the copy a copy, ..., the process will continue indefinitely. This self-referential map-within-a map construction is known as the map of England problem and is sometimes used in studies of consciousness.

This paper takes the map of England problem as inspiration to address the question of universality in computation. Specifically, a recursive approach is used to prove the non-existence of a universal computer. We begin by assuming that a universal computer exists capable of simulating any possible computation. We then show that this assumption leads to a non-terminating computation. Contradiction thus establishes that the assumption was false and that a universal computer is impossible.

2 There is no universal computer

The *principle of simulation* in computer science states that any computation by a general-purpose computer can be simulated by any other general-purpose computer (regardless of any architectural differences between the simulating and simulated computers, or how efficient or inefficient is the simulation). This principle holds in theory for such models of computation as the Turing Machine, the Random Access Machine, the Cellular Automaton, and so on, as well as in practice for physical computing devices such as desktops,

laptops, and so on. An immediate consequence of the principle of simulation is the *principle of universality*, which essentially states that there exists a universal computer capable of simulating all other computers. Effectively, any general purpose computer can serve as a universal computer [8, 9, 10, 11, 12, 14].

We prove in what follows that such a universal computer cannot be.

Theorem. A universal computer does not exist.

Proof: Let us assume, as is commonly the case in computer science, that there exists a universal computer U, capable of simulating any possible computation C, that runs on a computer M with an input I. We write $U(M, I)$ to express the fact that U takes M and I as input and simulates the computation C by performing the actions of M as it executes a program on I.

In what follows, let $C = (M, I)$ be a terminating computation, meaning that computer M executes a program on input I and halts in a finite number of computational steps. We write $U[C]$ as a shorthand for $U(M, I)$, the simulation of C by U, the latter also a terminating computation.

It is evident, from the principle of simulation, that the actions of U itself should be possible to simulate. The question is: 'Who' is to simulate a computation performed by a universal computer? Specifically, how are the actions of U–as it simulates C, that is, $U[C]$–*themselves* to be simulated?

There are two options.

Option 1. U simulates itself. We write $U[U[C]]$ to indicate that U is simulating $U[C]$. This means that U is executing the actions of itself (U, the computer), on the simulation ($U[C]$, the input); we write:

$$U[U[C]] = U(U, U[C]).$$

The right hand side of the above expression, has three Us: the first is the simulator, the second is the computer being simulated, and the third, $U[C]$, is the input. We therefore have:

$$\begin{aligned} U[U[C]] &= U(U, U[C]) \\ &= U(U[U[C]]) \\ &= U(U(U, U[C])) \\ &= U(U(U[U[C]])) \\ &= U(U(U(U, U[C]))) \\ &\vdots \end{aligned} \tag{1}$$

This leads to a *self-referential* infinite regress, with U simulating itself, simulating itself, simulating itself, ... The computation (1) just described is a non-terminating process (one could say that it is not even a computation, by definition, since it does not halt, but that is not important for our purpose). It follows that U has failed to simulate itself, while executing a *terminating* computation C.

Option 2. We can stipulate the existence of a more powerful universal computer U_1 that simulates $U[C]$. Again, this leads, in turn, to an *ascending* infinite regress, featuring a sequence of ever more powerful computers U_1, U_2, U_3, ... , U_n, U_{n+1}, ..., performing simulations $C_1, C_2, C_3, \ldots, C_n, C_{n+1}, \ldots$. Formally,

$$\begin{aligned} C_1 &= U_1[U[C]], \\ C_2 &= U_2[U_1[U[C]]], \\ C_3 &= U_3[U_2[U_1[U[C]]]], \\ &\vdots \\ C_{n+1} &= U_{n+1}[U_n \ldots U_3[U_2[U_1[U[C]]]] \ldots], \\ &\vdots \end{aligned} \tag{2}$$

and so on *ad infinitum*. In the unending sequence of computations described by (2), computer U_{n+1} is needed to simulate U_n, for $n = 1, 2, 3, \ldots$, given that computer U_n cannot simulate itself (as shown in Option 1 for U).

Both Option 1 and Option 2 are computationally absurd, leading to one conclusion: the assumption regarding the existence of a universal computer U is false. No computer is universal. ■

3 Conclusion

The idea of nonuniversality in computation is not a new one. Several proofs by counterexample of the non-existence of a universal computer have been presented since early in this century. They described different computational paradigms that falsify the idea of a universal computer. Examples of such paradigms include computations with: time varying variables, time varying computational complexity, rank varying computational complexity, interacting variables, uncertain time constraints, mathematical constraints, global variables, and so on [1, 2, 3, 4, 5, 6, 7]. The proof by contradiction presented here, however, is to our knowledge, the first of its kind.

We note in closing that nonuniversality in computation applies to all known computational models and existing conventional computers, both sequential and parallel, as well as applying to all future unconventional computers, including quantum computers, biomolecular computers, chemical computers, and so on.

References

[1] Akl, S.G., Three counterexamples to dispel the myth of the universal computer, *Parallel Processing Letters*, Vol. 16, No. 3, September 2006, pp. 381– 403.

[2] Akl, S.G., Even accelerating machines are not universal, *International Journal of Unconventional Computing*, Vol. 3, No. 2, 2007, pp. 105–121.

[3] Akl, S.G., Evolving computational systems, Chapter One in: *Handbook of Parallel Computing: Models, Algorithms, and Applications*, Rajasekaran, S. and Reif, J.H., Eds., Taylor & Francis, CRC Press, Boca Raton, Florida, 2008, pp. 1–22.

[4] Akl, S.G., Nonuniversality in computation: Fifteen misconceptions rectified, Chapter One in: *Advances in Unconventional Computing*, Adamatzky, A., Ed., Springer, Cham, Switzerland, 2017, pp. 1–31.

[5] Akl, S.G., Unconventional computational problems, in: *Encyclopedia of Complexity and Systems Science*, Springer, New York, 2018, pp. 631–639.

[6] Akl, S.G., Unconventional wisdom: Superlinear speedup and inherently parallel computations, Chapter Sixteen in: *From Parallel to Emergent Computing*, Adamatzky, A. et al., Eds., Taylor & Francis, CRC Press, Boca Raton, Florida, 2019, pp. 347–366.

[7] Akl, S.G. and Salay, N., On computable numbers, nonuniversality, and the genuine power of parallelism, *International Journal of Unconventional Computing*, Vol. 11, Nos. 3–4, 2015, pp. 283–297.

[8] Davis, M., *The Universal Computer*, W.W. Norton, 2000.

[9] Deutsch D., *The Fabric of Reality*, Penguin Books, London, United Kingdom, 1997.

[10] Hillis, D., *The Pattern on the Stone*, Basic Books, New York, 1998.

[11] Hopcroft, J.E. and Ullman, J.D., *Formal Languages and their Relations to Automata*, Addison-Wesley, Reading, Massachusetts, 1969.

[12] Lewis, H.R. and Papadimitriou, C.H., *Elements of the Theory of Computation*, Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[13] Royce, J., *The World and The Individual, First Series: The Four Historical Conceptions of Being*, Dover Publications, New York, 1959, pp. 504–505.

[14] Sipser, M., *Introduction to the Theory of Computation*, PWS Publishing Company, Boston, Massachusetts, 1997.