Simple Theoretically Practical Complexity Theory

Ari Cohn
acohn@andrew.cmu.edu

Abstract

Complexity theory is complex. Let's make it simpler.

1 Introduction

There is an inherent dichotomy between the theoretical and the practical, the former describing abstractly what may be done in a universe of our own design and the latter describing what is possible under the bounds of the world we were given. Like its cousin mathematics, computer science often lies in the realm of theory. However, computers themselves are very practical devices and the theory of computer science exists to inform practical implementation of its concepts. Therefore, I would like to propose a new model for looking at one of the core sub-fields of computer science — complexity theory. As its name suggests, complexity theory is both complex and theoretical. But why does it have to be so? Why not create a new model that is simple and practical, a model that more closely resembles the world in which we live? Introducing: The Simple Theoretically Practical Complexity Theory Model.

2 The SToP Model

In this section, I will describe the Simple Theoretically Practical Complexity Theory Model, or SToP model for short. The goal of the SToP model is to provide a new way of looking at complexity theory that is practical, but only in theory.

2.1 Assumptions

In order to make computation practical, we must write algorithms and programs on physical computers available for human use. So, let us first assume that the resources that are and ever will be available to humans for use in constructing computers is finitely bounded, i.e. by the size of the observable universe. We can then define a constant ϕ, the number of particles "available" to humans, which is estimated to be approximately 3.28×10^{80} [1]. Now, assume all possible computation exists within this physical universe. Therefore, under the SToP model there cannot exist data that exceeds ϕ in size.

2.2 One Complexity Class to Rule Them All

It follows directly from the assumptions that any halting program will terminate in $O(1)$ steps. Therefore, the SToP model contains exactly one complexity class: constant time.

Proof. Consider the set of all computer programs. This set is necessarily finite, because all programs must be stored on physical computers using at most ϕ particles. Therefore, the subset of all halting computer programs is also finite. Denote this set H, and define the function $T_p(n)$ as the number of steps a program p takes to terminate on an input of size n, where $0 \leq n \leq \phi$. Observe that there exists a constant c such that

$$c = \max_{p \in H, n} T_p(n)$$

where c is a constant upper bound on all possible computer programs' runtime. Thus, we can conclude that all (halting) computer programs are $O(c) = O(1)$.

Figure 1: Like a picture of a clock, all programs are constant time. (source: en.wikipedia.org)

3 Applications

3.1 P vs. NP

The most notable application of the SToP model is a clean, simple solution to the infamous P vs. NP problem. The answer, of course, is that P must be equal to NP.

Proof. It is well known that P is a subset of NP, so it will suffice to show that NP is a subset of P. Consider any problem p in NP. By definition, this problem can be solved by some algorithm A. Since A can be written as a computer program, $A \in O(1)$. So, $p \in$ P. Thus, P = NP.

3.2 How Hard is CMU?

Let CMU_Student be any computable function defining how to graduate from Carnegie Mellon. Such a function is known to exist since it is indeed purportedly possible for a human to graduate, and one can construct a computer program that exactly follows what such a human would do. Under SToP, we can conclude that CMU_Student is $O(1)$ and therefore CMU_Student performs constant work.

4 Conclusion

In an effort to make complexity theory simpler and more approachable, the SToP model provides a theoretically practical view of complexity theory that is both simple and easy to understand. Under the model, all computation can be completed in constant time, meaning every computer program has $O(1)$ runtime. Theoretically, this result makes no sense, and practically it has no value. But hey, that's why its called theoretically practical.

References

[1] Jay Bennett. 2017. *How Many Particles Are in the Observable Universe?* https://www.popularmechanics.com/space/a27259/how-many-particles-are-in-the-entire-universe/