EXCERPTED FROM

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE

SECTION 12.2

Outline of the Principle

But what about processes in nature? Can these also be viewed as computations? Or does the notion of computation somehow apply only to systems with abstract elements like, say, the black and white cells in a cellular automaton?

Before the advent of modern computer applications one might have assumed that it did. But now every day we see computations being done with a vast range of different kinds of data—from numbers to text to images to almost anything else. And what this suggests is that it is possible to think of any process that follows definite rules as being a computation—regardless of the kinds of elements it involves.

So in particular this implies that it should be possible to think of processes in nature as computations. And indeed in the end the only unfamiliar aspect of this is that the rules such processes follow are defined not by some computer program that we as humans construct but rather by the basic laws of nature.

But whatever the details of the rules involved the crucial point is that it is possible to view every process that occurs in nature or elsewhere as a computation. And it is this remarkable uniformity that makes it possible to formulate a principle as broad and powerful as the Principle of Computational Equivalence.

Outline of the Principle

Across all the vastly different processes that we see in nature and in systems that we construct one might at first think that there could be very little in common. But the idea that any process whatsoever can be viewed as a computation immediately provides at least a uniform framework in which to discuss different processes.

And it is by using this framework that the Principle of Computational Equivalence is formulated. For what the principle does is to assert that when viewed in computational terms there is a fundamental equivalence between many different kinds of processes.

There are various ways to state the Principle of Computational Equivalence, but probably the most general is just to say that almost all

processes that are not obviously simple can be viewed as computations of equivalent sophistication.

And although at first this statement might seem vague and perhaps almost inconsequential, we will see in the course of this chapter that in fact it has many very specific and dramatic implications.

One might have assumed that among different processes there would be a vast range of different levels of computational sophistication. But the remarkable assertion that the Principle of Computational Equivalence makes is that in practice this is not the case, and that instead there is essentially just one highest level of computational sophistication, and this is achieved by almost all processes that do not seem obviously simple.

So what might lead one to this rather surprising idea? An important clue comes from the phenomenon of universality that I discussed in the previous chapter and that has been responsible for much of the success of modern computer technology. For the essence of this phenomenon is that it is possible to construct universal systems that can perform essentially any computation—and which must therefore all in a sense be capable of exhibiting the highest level of computational sophistication.

The most familiar examples of universal systems today are practical computers and general-purpose computer languages. But in the fifty or so years since the phenomenon of universality was first identified, all sorts of types of systems have been found to be able to exhibit universality. Indeed, as I showed in the previous chapter, it is possible for example to get universality in cellular automata, Turing machines, register machines—or in fact in practically every kind of system that I have considered in this book.

So this implies that from a computational point of view even systems with quite different underlying structures will still usually show a certain kind of equivalence, in that rules can be found for them that achieve universality—and that therefore can always exhibit the same level of computational sophistication.

But while this is already a remarkable result, it represents only a first step in the direction of the Principle of Computational

Equivalence. For what the result implies is that in many kinds of systems particular rules can be found that achieve universality and thus show the same level of computational sophistication. But the result says nothing about whether such rules are somehow typical, or are instead very rare and special.

And in practice, almost without exception, the actual rules that have been established to be universal have tended to be quite complex. Indeed, most often they have in effect been engineered out of all sorts of components that are direct idealizations of various elaborate structures that exist in practical digital electronic computers.

And on the basis of traditional intuition it has almost always been assumed that this is somehow inevitable, and that in order to get something as sophisticated as universality there must be no choice but to set up rules that are themselves special and sophisticated.

One of the dramatic discoveries of this book, however, is that this is not the case, and that in fact even extremely simple rules can be universal. Indeed, from our discussion in the previous chapter, we already know that among the 256 very simplest possible cellular automaton rules at least rule 110 and three others like it are universal.

And my strong suspicion is that this is just the beginning, and that in time a fair fraction of other simple rules will also be shown to be universal. For one of the implications of the Principle of Computational Equivalence is that almost any rule whose behavior is not obviously simple should ultimately be capable of achieving the same level of computational sophistication and should thus in effect be universal.

So far from universality being some rare and special property that exists only in systems that have carefully been built to exhibit it, the Principle of Computational Equivalence implies that instead this property should be extremely common. And among other things this means that universality can be expected to occur not only in many kinds of abstract systems but also in all sorts of systems in nature.

And as we shall see in this chapter, this idea already has many important and surprising consequences. But still it is far short of what the full Principle of Computational Equivalence has to say.

For knowing that a particular rule is universal just tells one that it is possible to set up initial conditions that will cause a sophisticated computation to occur. But it does not tell one what will happen if, for example, one starts from typical simple initial conditions.

Yet the Principle of Computational Equivalence asserts that even in such a case, whenever the behavior one sees is not obviously simple, it will almost always correspond to a computation of equivalent sophistication.

So what this means is that even, say, in cellular automata that start from very simple initial conditions, one can expect that those aspects of their behavior that do not look obviously simple will usually correspond to computations of equivalent sophistication.

According to the Principle of Computational Equivalence therefore it does not matter how simple or complicated either the rules or the initial conditions for a process are: so long as the process itself does not look obviously simple, then it will almost always correspond to a computation of equivalent sophistication.

And what this suggests is that a fundamental unity exists across a vast range of processes in nature and elsewhere: despite all their detailed differences every process can be viewed as corresponding to a computation that is ultimately equivalent in its sophistication.

The Content of the Principle

Like many other fundamental principles in science, the Principle of Computational Equivalence can be viewed in part as a new law of nature, in part as an abstract fact and in part as a definition. For in one sense it tells us what kinds of computations can and cannot happen in our universe, yet it also summarizes purely abstract deductions about possible computations, and provides foundations for more general definitions of the very concept of computation.

Without the Principle of Computational Equivalence one might assume that different systems would always be able to perform completely different computations, and that in particular there would be no upper limit on the sophistication of computations that systems with sufficiently complicated structures would be able to perform.