EXCERPTED FROM

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE

SECTION 11.3

The Phenomenon of Universality

The Phenomenon of Universality

In the previous section we saw that it is possible to get cellular automata to perform some fairly sophisticated computations. But for each specific computation we wanted to do, we always set up a cellular automaton with a different set of underlying rules. And indeed our everyday experience with mechanical and other devices might lead us to assume that in general in order to perform different kinds of tasks we must always use systems that have different underlying constructions.

But the remarkable discovery that launched the computer revolution is that this is not in fact the case. And instead, it is possible to build universal systems whose underlying construction remains fixed, but which can be made to perform different tasks just by being programmed in different ways.

And indeed, this is exactly how practical computers work: the hardware of the computer remains fixed, but the computer can be programmed for different tasks by loading different pieces of software.

The idea of universality is also the basis for computer languages. For in each language, there are a certain set of primitive operations, which are then strung together in different ways to create programs for different tasks.

The details of a particular computer system or computer language will certainly affect how easy it is to perform a particular task. But the crucial fact that is by now a matter of common knowledge is that with appropriate programming any computer system or computer language can ultimately be made to perform exactly the same set of tasks.

One way to see that this must be true is to note that any particular computer system or computer language can always be set up by appropriate programming to emulate any other one.

Typically the way this is done is by having each individual action in the system that is to be emulated be reproduced by some sequence of actions in the other system. And indeed this is ultimately how, for example, *Mathematica* works. For when one enters a command such as Log[15], what actually happens is that the program which implements the *Mathematica* language interprets this command by executing the appropriate sequence of machine instructions on whatever computer system one is using.

And having now identified the phenomenon of universality in the context of practical computing, one can immediately see various analogs of it in other areas of common experience. Human languages provide an example. For one knows that given a single fixed underlying language, it is possible to describe an almost arbitrarily wide range of things. And given any two languages, it is for the most part always possible to translate between them.

So what about natural science? Is the phenomenon of universality also relevant there? Despite its great importance in computing and elsewhere, it turns out that universality has in the past never been considered seriously in relation to natural science.

But what I will show in this chapter and the next is that in fact universality is for example quite crucial in finding general ways to characterize and understand the complexity we see in natural systems.

The basic point is that if a system is universal, then it must effectively be capable of emulating any other system, and as a result it must be able to produce behavior that is as complex as the behavior of any other system. So knowing that a particular system is universal thus immediately implies that the system can produce behavior that is in a sense arbitrarily complex.

But now the question is what kinds of systems are in fact universal.

Most present-day mechanical devices, for example, are built only for rather specific tasks, and are not universal. And among electronic devices there are examples such as simple calculators and electronic address books that are not universal. But by now the vast majority of practical electronic devices, despite all their apparent differences, are based on computers that are universal.

At some level, however, these computers tend to be extremely similar. Indeed, essentially all of them are based on the same kinds of logic circuits, the same basic layout of data paths, and so on. And knowing this, one might conclude that any system which was universal must include direct analogs of these specific elements. But from experience with computer languages, there is already an indication that the range of systems that are universal might be somewhat broader.

Indeed, *Mathematica* turns out to be a particularly good example, in which one can pick very different sets of operations to use, and yet still be able to implement exactly the same kinds of programs.

So what about cellular automata and other systems with simple rules? Is it possible for these kinds of systems to be universal?

At first, it seems quite implausible that they could be. For the intuition that one gets from practical computers and computer languages seems to suggest that to achieve universality there must be some fundamentally fairly sophisticated elements present.

But just as we found that the intuition which suggests that simple rules cannot lead to complex behavior is wrong, so also the intuition that simple rules cannot be universal also turns out to be wrong. And indeed, later in this chapter, I will show an example of a cellular automaton with an extremely simple underlying rule that can nevertheless in the end be seen to be universal.

In the past it has tended to be assumed that universality is somehow a rare and special quality, usually possessed only by systems that are specifically constructed to have it. But one of the results of this chapter is that in fact universality is a much more widespread phenomenon. And in the next chapter I will argue that for example it also occurs in a wide range of important systems that we see in nature.

A Universal Cellular Automaton

As our first specific example of a system that exhibits universality, I discuss in this section a particular universal cellular automaton that has been set up to make its operation as easy to follow as possible.

The rules for this cellular automaton itself are always the same. But the fact that it is universal means that if it is given appropriate initial conditions it can effectively be programmed to emulate for example any possible cellular automaton—with any set of rules.

The next three pages show three examples of this.