EXCERPTED FROM

STEPHEN WOLFRAM

A NEW KIND OF SCIENCE

SECTION 12.3

The Content of the Principle

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.

But the discussion of universality in the previous chapter already suggests that this is not the case. For it implies that at least across the kinds of systems that we considered in that chapter there is in fact an upper limit on the sophistication of computations that can be done.

For as we discussed, once one has a universal system such a system can emulate any of the kinds of systems that we considered—even ones whose construction is more complicated than its own. So this means that whatever kinds of computations can be done by the universal system, none of the other systems will ever be able to do computations that have any higher level of sophistication.

And as a result it has often seemed reasonable to define what one means by a computation as being precisely something that can be done by a universal system of the kind we discussed in the previous chapter.

But despite this, at an abstract level one can always imagine having systems that do computations beyond what any of the cellular automata, Turing machines or other types of systems in the previous chapter can do. For as soon as one identifies any such class of computations, one can imagine setting up a system which includes an infinite table of their results.

But even though one can perfectly well imagine such a system, the Principle of Computational Equivalence makes the assertion that no such system could ever in fact be constructed in our actual universe.

In essence, therefore, the Principle of Computational Equivalence introduces a new law of nature to the effect that no system can ever carry out explicit computations that are more sophisticated than those carried out by systems like cellular automata and Turing machines.

So what might make one think that this is true? One important piece of evidence is the success of the various models of natural systems that I have discussed in this book based on systems like cellular automata. But despite these successes, one might still imagine that other systems could exist in nature that are based, say, on continuous mathematics, and which would allow computations more sophisticated than those in systems like cellular automata to be done.

Needless to say, I do not believe that this is the case, and in fact if one could find a truly fundamental theory of physics along the lines I

discussed in Chapter 9 it would actually be possible to establish this with complete certainty. For such a theory would have the feature that it could be emulated by a universal system of the type I discussed in the previous chapter—with the result that nowhere in our universe could computations ever occur that are more sophisticated than those carried out by the universal systems we have discussed.

So what about computations that we perform abstractly with computers or in our brains? Can these perhaps be more sophisticated? Presumably they cannot, at least if we want actual results, and not just generalities. For if a computation is to be carried out explicitly, then it must ultimately be implemented as a physical process, and must therefore be subject to the same limitations as any such process.

But as I discussed in the previous section, beyond asserting that there is an upper limit to computational sophistication, the Principle of Computational Equivalence also makes the much stronger statement that almost all processes except those that are obviously simple actually achieve this limit.

And this is related to what I believe is a very fundamental abstract fact: that among all possible systems with behavior that is not obviously simple an overwhelming fraction are universal.

So what would be involved in establishing this fact?

One could imagine doing much as I did early in this book and successively looking at every possible rule for some type of system like a cellular automaton. And if one did this what one would find is that many of the rules exhibit obviously simple repetitive or nested behavior. But as I discovered early in this book, many also do not, and instead exhibit behavior that is often vastly more complex.

And what the Principle of Computational Equivalence then asserts is that the vast majority of such rules will be universal.

If one starts from scratch then it is not particularly difficult to construct rules—though usually fairly complicated ones—that one knows are universal. And from the result in the previous chapter that rule 110 is universal it follows for example that any rule containing this one must also be universal. But if one is just given an arbitrary rule—

and especially a simple one—then it can be extremely difficult to determine whether or not the rule is universal.

As we discussed in the previous chapter, the usual way to demonstrate that a rule is universal is to find a scheme for setting up initial conditions and for decoding output that makes the rule emulate some other rule that is already known to be universal.

But the problem is that in any particular case there is almost no limit on how complicated such a scheme might need to be. In fact, about the only restriction is that the scheme itself should not exhibit universality just in setting up initial conditions and decoding output.

And indeed it is almost inevitable that the scheme will have to be at least somewhat complicated: for if a system is to be universal then it must be able to emulate any of the huge range of other systems that are universal—with the result that specifying which particular such system it is going to emulate for the purposes of a proof will typically require giving a fair amount of information, all of which must somehow be part of the encoding scheme.

It is often even more difficult to prove that a system is not universal than to prove that it is. For what one needs to show is that no possible scheme can be devised that will allow the system to emulate any other universal system. And usually the only way to be sure of this is to have a more or less complete analysis of all possible behavior that the system can exhibit.

If this behavior always has an obvious repetitive or nested form then it will often be quite straightforward to analyze. But as we saw in Chapter 10, in almost no other case do standard methods of perception and analysis allow one to make much progress at all.

As mentioned in Chapter 10, however, I do know of a few systems based on numbers for which a fairly complete analysis can be given even though the overall behavior is not repetitive or nested or otherwise obviously simple. And no doubt some other examples like this do exist. But it is my strong belief—as embodied in the Principle of Computational Equivalence—that in the end the vast majority of systems whose behavior is not obviously simple will turn out to be universal.

If one tries to use some kind of systematic procedure to test whether systems are universal then inevitably there will be three types of outcomes. Sometimes the procedure will successfully prove that a system is universal, and sometimes it will prove that it is not. But very often the procedure will simply come to no definite conclusion, even after spending a large amount of effort.

Yet in almost all such cases the Principle of Computational Equivalence asserts that the systems are in fact universal. And although almost inevitably it will never be easy to prove this in any great generality, my guess is that, as the decades go by, more and more specific rules will end up being proved to exhibit universality.

But even if one becomes convinced of the abstract fact that out of all possible rules that do not yield obviously simple behavior the vast majority are universal, this still does not quite establish the assertion made by the Principle of Computational Equivalence that rules of this kind that appear in nature and elsewhere are almost always universal.

For it could still be that the particular rules that appear are somehow specially selected to be ones that are not universal. And certainly there are all sorts of situations in which rules are constrained to have behavior that is too simple to support universality. Thus, for example, in most kinds of engineering one tends to pick rules whose behavior is simple enough that one can readily predict it. And as I discussed in Chapter 8, something similar seems to happen with rules in biology that are determined by natural selection.

But when there are no constraints that force simple overall behavior, my guess is that most rules that appear in nature can be viewed as being selected in no special way—save perhaps for the fact that the structure of the rules themselves tends to be fairly simple.

And what this means is that such rules will typically show the same features as rules chosen at random from all possibilities—with the result that presumably they do in the end exhibit universality in almost all cases where their overall behavior is not obviously simple.

But even if a wide range of systems can indeed be shown to be universal this is still not enough to establish the full Principle of Computational Equivalence. For the Principle of Computational

Equivalence is concerned not only with the computational sophistication of complete systems but also with the computational sophistication of specific processes that occur within systems.

And when one says that a particular system is universal what one means is that it is possible by choosing appropriate initial conditions to make the system perform computations of essentially any sophistication. But from this there is no guarantee that the vast majority of initial conditions—including perhaps all those that could readily arise in nature—will not just yield behavior that corresponds only to very simple computations.

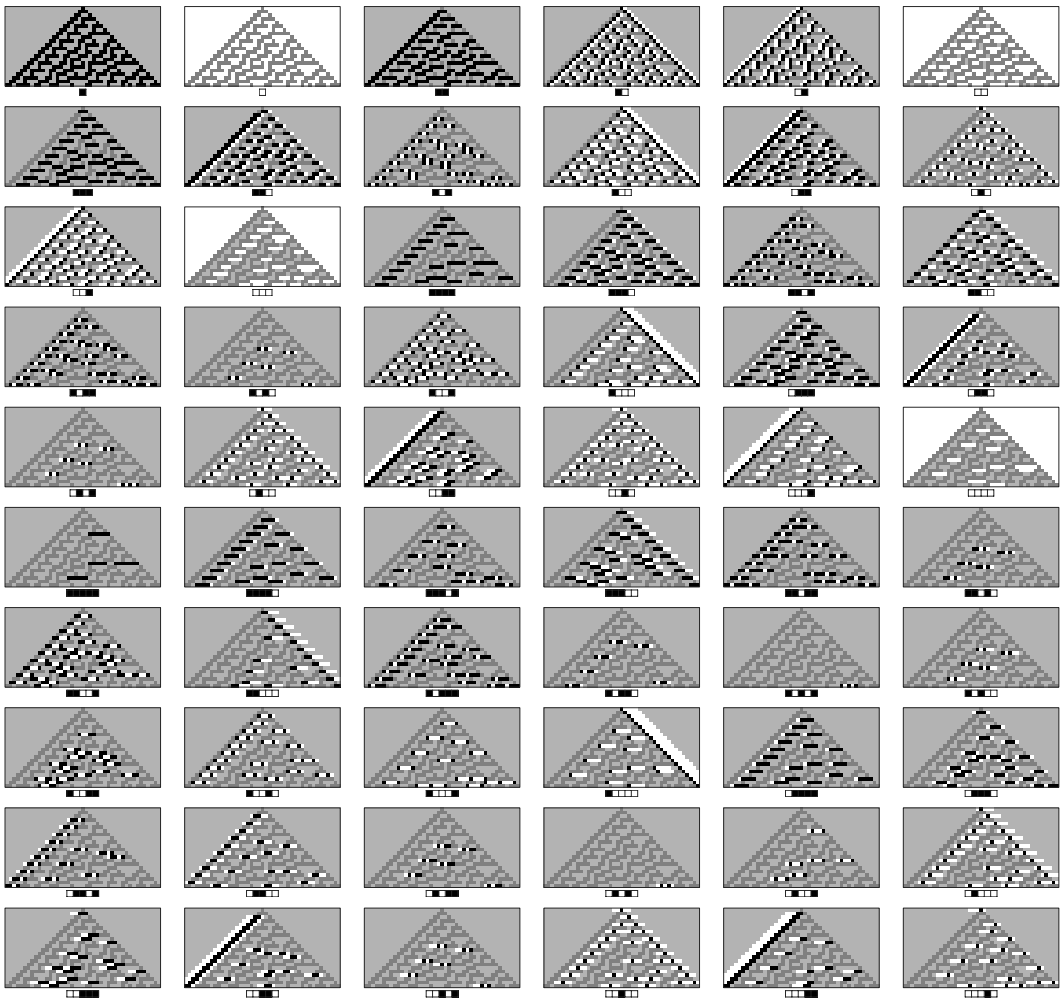
And indeed in the proof of the universality of rule 110 in the previous chapter extremely complicated initial conditions were used to perform even rather simple computations.

But the Principle of Computational Equivalence asserts that in fact even if it comes from simple initial conditions almost all behavior that is not obviously simple will in the end correspond to computations of equivalent sophistication.

And certainly there are all sorts of pictures in this book that lend support to this idea. For over and over again we have seen that simple initial conditions are quite sufficient to produce behavior of immense complexity, and that making the initial conditions more complicated typically does not lead to behavior that looks any different.

Quite often part of the reason for this, as illustrated in the pictures on the facing page, is that even with a single very simple initial condition the actual evolution of a system will generate blocks that correspond to essentially all possible initial conditions. And this means that whatever behavior would be seen with a given overall initial condition, that same behavior will also be seen at appropriate places in the single pattern generated from a specific initial condition.

So this suggests a way of having something analogous to universality in a single pattern instead of in a complete system. The idea would be that a pattern that is universal could serve as a kind of directory of possible computations—with different regions in the pattern giving results for all possible different initial conditions.



Occurrences of progressively longer blocks in the pattern generated by rule 30 starting from a single black cell. So far as I can tell, all possible blocks eventually appear, potentially letting the pattern serve as a kind of directory of all possible computations.

So as a simple example one could imagine having a pattern laid out on a three-dimensional array with each successive vertical plane giving the evolution of some one-dimensional universal system from each of its successive possible initial conditions. And with this setup any computation, regardless of its sophistication, must appear somewhere in the pattern.

In a pattern like the one obtained from rule 30 above different computations are presumably not arranged in any such straightforward way. But I strongly suspect that even though it may be quite impractical to find particular computations that one wants, it is still the case that essentially any possible computation exists somewhere in the pattern.

Much as in the case of universality for complete systems, however, the Principle of Computational Equivalence does not just say that a sophisticated computation will be found somewhere in a pattern produced by a system like rule 30. Rather, it asserts that unless it is obviously simple essentially any behavior that one sees should correspond to a computation of equivalent sophistication.

And in a sense this can be viewed as providing a new way to define the very notion of computation. For it implies that essentially any piece of complex behavior that we see corresponds to a kind of lump of computation that is at some level equivalent.

It is a little like what happens in thermodynamics, where all sorts of complicated microscopic motions are identified as corresponding in some uniform way to a notion of heat.

But computation is both a much more general and much more powerful notion than heat. And as a result, the Principle of Computational Equivalence has vastly richer implications than the laws of thermodynamics—or for that matter, than essentially any single collection of laws in science.

The Validity of the Principle

With the intuition of traditional science the Principle of Computational Equivalence—and particularly many of its implications—might seem almost absurd. But as I have developed more and more new intuition from the discoveries in this book so I have become more and more certain that the Principle of Computational Equivalence must be valid.

But like any principle in science with real content it could in the future always be found that at least some aspect of the Principle of Computational Equivalence is not valid. For as a law of nature the principle could turn out to disagree with what is observed in our