Computational Indeterminacy: what is your computer doing?

Originally written as a blogpost for the Intercontinental Academia 4 Project on Intelligence and Artificial Intelligence.

Henry Taylor

j.h.taylor.1@bham.ac.uk

Today we plunged into the philosophy of computation, with two philosophers of cognitive science (Jack Copeland and Oron Shagrir). We focussed on the notion of indeterminacy in computation. In this blog post, we'll first look at what computational indeterminacy is, and then look at its applications and philosophical questions.

What is computational indeterminacy?

Suppose you have a computer, carrying out all sorts of tasks. Electrical signals are pulsing through it, flowing across its circuits, inputs flowing in, outputs flowing out. Now, we can ask the question: *what* is the computer doing? Or, more precisely: which computations is it actually carrying out?

This may seem easy to answer: you simply look at the machine and find out which computations it's carrying out, surely? Well, no. Here is where the issue of computational indeterminacy comes in, and to understand it we'll need a tiny bit of logic.

Computers make use of what are called *logic gates*, or simply *gates.* Each gate takes a certain set of inputs, then (based on the rules that govern the gate), it gives a certain output. So let's take a really simple gate, with two inputs, and one output. Inputs and outputs are in the form of electrical pulses (so it can take in up to two electrical pulses as inputs, and give up to one electrical pulse as an output). Furthermore, the *absence* of a pulse can also be an output or an input as well.

Let's say the gate follows this rule:

GATE-RULE:
IF I RECEIVE TWO ELECTRICAL PULSES, THEN I WILL OUTPUT ONE PULSE. IF I DO NOT RECEIVE TWO ELECTRICAL PULSES, I WILL NOT OUTPUT A PULSE.

More simply, the gate's job is to only output a pulse if it receives two pulses as input. Otherwise, it stays totally silent. So far so good.

Now the core question: what is the gate computing? Someone with a teeny bit of undergraduate logic may think the answer is obvious: it's computing AND. AND one of the most basic logical functions, which is characterised by the following rule:

AND-RULE:
IF I RECEIVE TWO INPUTS OF 'TRUE' (OR '1') THEN I WILL OUTPUT 'TRUE' (OR '1'). OTHERWISE, I WILL OUTPUT 'FALSE' (OR '0').

It initially looks pretty clear that the gate is computing AND. If you compare the AND-RULE to the GATE-RULE, they look pretty much the same. After all, to get the AND-RULE from the GATE-RULE, you just need to substitute an electrical pulse for 'TRUE' or '1', and the absence of a pulse for 'FALSE' or '0'.

So what’s the problem? It stems from the way we map pulses to ‘TRUE’ (or ‘1’) and non-pulses to ‘FALSE’ (or ‘0’). If we interpret a pulse as TRUE, and a non-pulse as FALSE, then the gate will indeed compute the function AND (characterised by the AND-RULE, above). But that’s just a choice we made, to map pulses to TRUE and non-pulses to FALSE in these ways.

What if we did it the other way round? What if we switched the mappings round? What if we interpret a pulse as FALSE, and non-pulse as TRUE? What then? The gate will still output a pulse when (and only when) it receives two pulses. But now (since we’ve switched the mappings around), what it’s doing is outputting a ‘FALSE’ only when it receives two ‘FALSE’ inputs. In fact, what it’s now doing (as any quick-witted first year logic student will tell you) is computing OR. That is, it is computing this:

OR-RULE
IF I RECEIVE TWO INPUTS OF ‘FALSE’ (OR ‘0’) THEN I WILL OUTPUT ‘FALSE’ (OR ‘0’). OTHERWISE, I WILL OUTPUT ‘TRUE’ (OR ‘1’).

So this is pretty weird. Whether the gate is computing AND or OR is a matter of how we interpret it. It’s a matter of how we map electrical pulses to TRUE/1, or FALSE/0. There’s no correct answer to which computation the gate itself is doing, that’s a matter of how we interpret it. That’s computational indeterminacy.

Obviously, a computer is more than just one gate. It’s many many gates. As you add more and more gates into the system, the indeterminacy is going to multiply and multiply. Just one gate can give you OR or AND, but when you put enough gates together (like there are in a modern computer) the indeterminacy is only going to get more complex.

What is it good for?

This phenomenon was discovered by Ralph Slutz, the 20th century computational engineer. However, its importance for philosophy and cognitive science is only now becoming apparent. Why is it so important? This is something Jack Copeland focussed on in his talk. Return to our example of a gate that could be OR or AND. Imagine you have a computer, and it has two different tasks to perform. Suppose that, in order to perform task 1, it will need an OR gate. In order to perform task 2, it needs an AND gate. Here’s the crucial question: will this computer need two different gates to do this?

No. As we saw above, whether a gate computes AND or OR is a matter of interpretation, so in a sense the gate performs both of them. You only need one gate, and it can compute both. All you need is two different systems that interpret the gate (these are what Jack Copeland calls ‘probes’). One probe interprets the gate as an AND gate, the other interprets it as an OR gate. So you only need one gate, and it can do the job of an AND gate, and an OR gate. Your computer can perform both tasks with only one gate. You get two gates for the price of one. Gates, it turns out, are great at multi-tasking.

This could be especially useful in reducing redundancy in machine design. Rather than using huge numbers of gates, each of which doing one job, we might hope to get each one to multi-task. Ultimately, we might hope to end up with one large central system, with many gates all doing their thing. We might then hope that there could be multiple probes, each of which interprets this system in a different way. Maybe one of them interprets the system in a way that is relevant for one task, another which is relevant for another, and so on. All of the tasks are performed by just one system, but it gets put to many different uses, by being interpreted in different ways by different probes. It's like getting many many different computations, all for the price of one.

Philosophical questions:

Let's leave machine design to one side, and concentrate on the philosophical implications of all of this. Oron Shagrir takes the indeterminacy one step further. Above, we used the example of just two possible states (pulse and non-pulse) and two interpretations (TRUE/1 and FALSE/0). But of course, in reality, there are many states. Imagine you have three possible states. For example, suppose that instead of taking just a pulse or a non-pulse as inputs, the gate can take 3 volts, 6 volts, and 10 volts as inputs. Now we have three possible input states. And you can map these to TRUE/1 and FALSE/0 in different ways too. You could interpret 3 volts as TRUE/1, and then interpret 6 and 10 volts each as FALSE/0. Or you could go the other way, and interpret 3 or 6 volts as TRUE/1, and 10 volts as FALSE/0 (obviously there will be many more possible interpretations).

Now, by making things just a tiny bit more complicated in this way (by scrapping just pulse and non-pulse and replacing them with 3, 6 and 10 voltages, and mapping them in different ways to TRUE/1, and FALSE/0), the number of computations that a gate can compute will go up. One gate could not just do OR or AND, but also other ones as well, such as XOR (XOR is 'exclusive-or': it will output 'TRUE' only in the case that *exactly one* of the inputs is 'TRUE').

Here we encounter a problem. As we have seen, one physical system can perform many different computations at once (in our first example, the really simple gate can compute OR and AND at the same time). But which physical systems can be correctly interpreted in these ways? Where does this end? Let's take the example of a rock. The rock is made up of atoms and molecules, each of which are engaged in complex physical activities. Could we interpret these complex physical interactions as performing computations, in the same way that we interpreted our really simple gate above as performing AND and OR? Could we end up attributing computations to a rock? Surely that feels wrong. Rocks are not computers!

These are called 'triviality results', first introduced by philosophers such as Hilary Putnam. These are cases where you basically end up saying that everything is a computer. But we don't want that. We want to say that laptops, smartphones (maybe brains) compute things. Rocks, sticks, stones, and sealing wax do not. We need to draw the line in the right place.

Oron's own proposal is that we use semantics to solve this problem. The really computational states are the ones that have semantic content. That is, the ones that carry *meanings*, that represent things in the world. Contrary to what a lot of philosophers have thought, computation may be a deeply semantic phenomenon.