Causality, computing, and complexity

Russ Abbott
Department of Computer Science
California State University, Los Angeles

Abstract. I discuss two categories of causal relationships: *primitive causal interactions* of the sort characterized by Phil Dowe and the more general *manipulable causal relationships* as defined by James Woodward. All primitive causal interactions are manipulable causal relationships, but there are manipulable causal relationships that are not primitive causal interactions. I'll call the latter *constructed causal relationships*, and I'll argue that constructed causal relationships serve as a foundation for both computing and complex systems.

Perhaps even more interesting are *autonomous causal relationships*. These are constructed causal relationships in which the causal mechanism resides primarily in the effect. A typical example is a software execution engine. Software execution engines are on the effect side of a cause-effect relationship in which software is the cause and the behavior of the execution engine is the effect. The mechanism responsible for that causal relationship resides in the execution engine.

Introduction

Causality has been a longstanding and controversial topic in philosophy. I think it's fair to say, though, that Phil Dowe's primitive causal interactions and James Woodward's manipulable causality currently reign as the primary ways to think about philosophical causality. See the section below on actions and events for some additional discussion.

Primitive causal interactions

Primitive causal interactions correspond to direct interactions—the sorts of interactions one associates with billiard balls hitting one another. (Dowe 2000) defines a [primitive[1]] *causal interaction* as follows.

> An *object* is anything found in the ontology of science (such as particles, waves or fields), or common sense (such as chairs, buildings, or people).
>
> A *world line* is the collection of points on a space-time (Minkowski) diagram which represents the history of an object.
>
> A *causal interaction* is an intersection of world lines which involves [an] exchange of a conserved quantity.

For our purposes one can think of a primitive causal interaction as occurring at a single moment in time and at a single point in space.[2]

Manipulable causal relationships

A *manipulable causal relationship* is defined by Woodward (2003, 2012/forthcoming), as follows.[3]

---

[1] Dowe uses the term *causal interaction* rather than *primitive causal interaction*. He entitles his book *Physical Causality* reflecting his commitment to fundamental physical interactions. Hence my use of the term *primitive*.

[2] Quantum interactions do not occur at a single moment in time or at a single point in space. They occur through an exchange of virtual particles. Emission and absorption occur at different moments and different locations. But this issue will not be relevant for this paper, which will be concerned with macro-level interactions.

[3] Woodward's general definition is expressed in terms of probability distributions rather than just values.

> *X* causes *Y* iff (i) there is a possible intervention[4] that changes the value of *X* which is such that (ii) under this intervention the value of *Y* changes.

The motto characterizing manipulable causal relationships is: when you wiggle *X*, *Y* wiggles. The idea is that *X* provides access to *Y* for the purpose of change.

The more common term for Woodward-style causality is *interventionist causality*. I am using the term *manipulable* instead because it seems to reflect Woodward's intent more accurately. Woodward (2012/forthcoming) says that his approach serves our human desire to manipulate in the world. "Even if one wants to consign the notion of cause to the rubbish heap, one needs some way [to identify] relationships [that can be exploited] for manipulation and control."[5]

Philosophical causality can be a hall of mirrors

Consider the following example: Bob killed the plant by not watering it. No conserved quantity is transferred between Bob and the plant. Yet from a manipulability perspective Bob's not watering the plant caused it to die. (We are assuming that it was Bob's practice to water the plant.) Changing Bob's behavior from watering the plant to not watering the plant changed the plant's fate from living to dying. So Bob's killing the plant by not watering it is a manipulable causal relationship that is not a primitive causal interaction.

One can construct arbitrarily complex—and perhaps arbitrarily foolish and arbitrarily vexing—examples. What if Alice, noticing that Bob was no longer watering the plant, did so herself, thereby keeping it alive? Did Bob's not watering the plant cause it to stay alive—because his not watering it caused Alice to water it, which caused it to stay alive? I do not intend to be drawn into issues of this sort. See Paul and Hall (2012) for an extended discussion of examples like this.

Constructed causal relationships

I'll use the term *constructed causal relationships* for a manipulable causal relationship that is not a primitive causal interaction. Bob's killing the plant (or not killing the plant depending on whether Alice takes over the job) by not watering it are two examples. For another, consider a light switch or a water faucet. What happens when one turns the switch or faucet from *Off* to *On*? First of all, light starts shining or water starts flowing. And this happens so reliably that it makes sense to say that manipulable causal relationships exist between the position of the switch/faucet and the flowing or not flowing of the light/water.[6]

What role do actions and events play in manipulable causal relationships?

When thinking about causal relationships it's important not to rely too heavily on the intuitive meaning of the verb *to cause*. One common understanding of *to cause* is *to make happen*: "*X* causes *Y*" is often taken to mean that *X* makes *Y* happen or *X* brings *Y* about.

To cause is an active transitive verb: a subject does something to an object. To a great extent this is what the conserved-quantity-transfer notion of *cause* captures. Transferring a conserved quantity to or from an object typically makes something happen to that object. In particular the cause and the effect are both events. (This is the position that Paul and Hall take.) Schaeffer (2014) puts it this way.

---

[4] An "intervention" is an idealized unconfounded manipulation of *X* that changes *Y*, if at all, only through *X*.

[5] Although Woodward's approach may be motivated by a human goal of manipulation, it is not a commitment to a causality dependent on human manipulations. An intervention simply requires that something change, not that a human being change it. See, for example, (Woodward 2013, section 9) for a discussion.

[6] More precisely and as discussed in the next section, the relationship between the position of the switch/faucet and the flow of light/water is persistent under the *Off*/*On* intervention in just those situations in which (perhaps somewhat circularly) the intervention results in water/light flowing or not.

> The standard view of the causal relata is that they are of the category of event, and that their number is *two*, in the roles of *cause* and *effect*. So on the standard view, when the cue ball knocks the nine ball into the corner pocket, there is said to be an (actual) event e_1 of the cue ball striking the nine ball, and an (actual, distinct) event e_2 of the nine ball sinking into the corner pocket, such that e_1 is cause and e_2 effect. …
>
> The standard view has, of course, been disputed on all counts. As to category, while the standard view casts the causal relata as events, one also finds considerable support for *facts*, and occasional support for such other entities as *features*, *tropes*, *states of affairs*, *situations*, and *aspects*. Allegiances are further complicated by disagreements over what events, facts, and these other creatures are.

In the perspective taken here a manipulable causal relationship need *not* involve either an action or an event. Recall that for Woodward a manipulable causal relationship exists between *X* and *Y* (to simply somewhat) if changing *X*—i.e., intervening on *X*—results in a consistent change to *Y*. But *X* and *Y* themselves need not be actions or events. They can be static properties. All that is required is that the value(s) for *Y* must be determined for any given value(s) for *X*.[7]

A more straightforward way to put this is that a manipulable causal relationship is a functional dependency as defined in relational database theory.[8] For example, in US addresses there is a manipulable causal relationship between ZIP code and state.[9] Given a ZIP code, the state is determined. Furthermore, the *ZIP code → state* dependency is invariant under an intervention that changes ZIP code from one value to another.

- If in a database system, one changes a person's ZIP code, then that person's state also changes—and in a manner consistent with other ZIP codes.
- If an attempt is made to enter data showing two people with the same ZIP code but different states the system rejects that data.

Another way to think about a manipulable causal relationship between *X* and *Y* is as if *X* were a remote control for *Y*: one can change *Y* by changing *X*. Although a nice metaphor and not a bad thought aid, this metaphor is misleading since most remote controls are linked to their controlled devices via a physical signal. As our database example shows no such direct physical link need necessarily exist.

Woodward refers to the link between cause and effect as a generalization that is invariant under an intervention.

> Suppose there is a function *f* relating values of *X* to values of *Y*: $Y = f(X)$. This relation is invariant under intervention *I* if and only if were *X* set through intervention *I* to some value x_0, *Y* would then have the value $f(x_0)$.[10]

Woodward is careful not to require that the function *f* hold for every change to *X*. In this way he distinguishes a law of nature from a more limited generalization that holds under certain but not all conditions. For example, in our ZIP code example, it's possible that one may know a person's state but not a person's ZIP code. Furthermore there may be multiple people with unknown ZIP codes—each

---

[7] *X* and *Y* may be collections of properties.

[8] Wikipedia/Functional dependency has a good explanation.

[9] This isn't a law of nature; it just happens to be the way ZIP codes are assigned. No ZIP code crosses a state border; every ZIP code is completely contained within some state.

[10] I have been unable to find a clear direct statement by Woodward. The preceding was paraphrased from Hiddleston 2005.

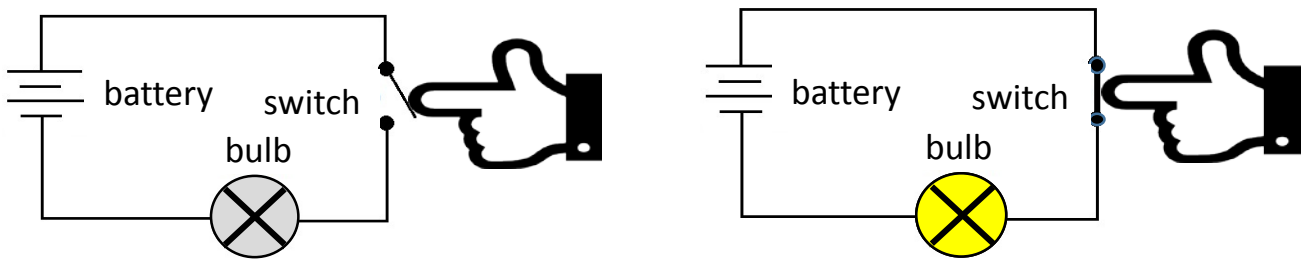
residing in a different state. Setting the ZIP code for these people to *unknown* does not guarantee that they all have the same value for state. The only intervention under which the *ZIP code → state* dependency holds is one that sets ZIP code to some valid[11] value.

Causal mechanisms

Earlier we said that manipulable causal relationships don't necessarily involve specific causal mechanisms. But they aren't magic either. For any manipulable causal relationship there must be some means whereby the dependency between cause and effect is maintained.

For example, suppose you take a picture of yourself in a mirror, i.e., a selfie. There is a manipulable causal relationship between the visual presentation projected toward the mirror, the cause, and the image captured by the camera, the effect. Change, i.e., manipulate, your expression, and the captured image will change. The mirror provides a significant part of the causal mechanism linking cause to effect.

Let's return to our switch example. What happens when we turn the switch from *Off* to *On*?



The key to most constructed causal relationships is that a change to the cause changes the environment in such a way that the effect occurs. For example, flipping a light switch closes a circuit so that under normal circumstances—in which a voltage is available in the circuit, the rest of the circuit is unbroken, the light bulb is working properly, etc.—the light bulb produces light. A change to the cause results in a change to the effect even though the change to the cause does not transfer a conserved quantity to the effect: the kinetic energy involved in closing the switch was not transferred to the light bulb.

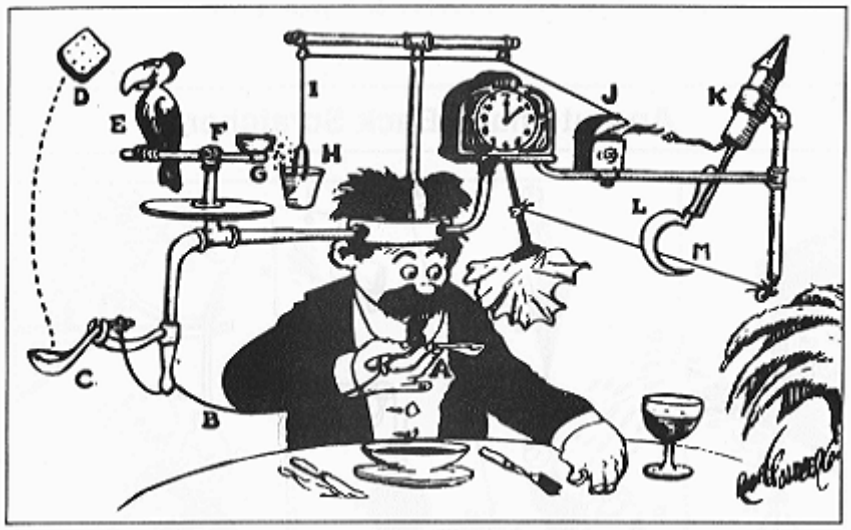
An even simpler story applies to a water faucet. Turning it to *On* allows water to flow through the tap. On its own, turning the faucet to *On* does nothing to push water through the tap. But water flows through the tap only when the system has water and sufficient pressure. In addition, there must not be either a hole or a blockage in the pipe between the faucet and the tap. Furthermore, the pipe must not be made of sugar, which will dissolve as water flows through it.

Similarly with the mirror example. The cause and the effect are in a manipulable causal relationship only if there is a source of light, the camera is working, there is room on the camera's memory card, nothing blocks the flow of light from cause to effect, etc.

As these examples illustrate one can multiply almost endlessly conditions that must hold for an effect to occur in a constructed causal relationship. They also suggest a motivation for using the term *constructed* for these types of causal relationships. Such relationships depend on mechanisms in the environment. An electric switch lever, a faucet knob, and a mirror do nothing unless embedded within a larger construction.

[11] According to the United States Postal Service there are somewhat more than 42,000 ZIP codes. Since there are 100,000 five-digits strings, fewer than half of all five-digit strings are valid ZIP codes.

Cause-and-effect in software



Software is text—or when compiled some other static representation of information. On its own software does nothing. For software to have any dynamic properties it must be executed by a software execution engine. Let's compare a software execution engine to a Rube Goldberg machine such as the Self-Operating Napkin[12] shown above. The two differ in two important ways.

1. A software execution engine follows directions encoded symbolically. A Rube Goldberg machine consists of a series of primitive causal interactions. There are no symbolic instructions.
2. A software execution engine runs on an independent energy supply—usually electricity from an external source. A Rube Goldberg machine is powered by multiple integrated sources of energy—usually the conversion of gravitational potential energy into kinetic energy although sometimes energy released through the action of living organisms or chemical energy as in combustion. The charm of a Rube Goldberg machine inheres in the mechanisms through which the release of energy from one source triggers the release of energy from the next. Without this cascading sequence of energy releases the machine would stop. Energetics, a primary concern for Rube Goldberg machines, is irrelevant in software execution engines.

What are the causal relationships in software? Software is in a constructed causal relationship with its execution engine. The software is the cause; the actions of the engine are the effects. Software execution is something like a railroad train moving along its track—or multiple tracks if the software is multiply threaded. The software is the track; the execution engine is the train. Changing the software results in a change in the behavior of the execution engine.

In many ways the cause and effect relationship between software and its execution engine resembles the way Laplace used the terms *cause* and *effect* when describing his demon. "We may regard the present state of the universe as the *effect* of its past and the *cause* of its future." [Emphasis added][13]

---

[12] The Self-Operating Napkin is activated when soup spoon (A) is raised to mouth, pulling string (B) and thereby jerking ladle (C), which throws cracker (D) past parrot (E). Parrot jumps after cracker and perch (F) tilts, upsetting seeds (G) into pail (H). Extra weight in pail pulls cord (I), which opens and lights automatic cigar lighter (J), setting off skyrocket (K) which causes sickle (L) to cut string (M) and allow pendulum with attached napkin to swing back and forth, thereby wiping chin. (From Wikipedia 2015.)

[13] The rest of the description is as follows. "An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest

In using the terms *cause* and *effect* Laplace was not intending to add anything to physics. All he was saying is that the laws of physics (as understood at the time[14]) were sufficient to enable one to see the world in terms of cause and effect. Laplace didn't say explicitly how cause and effect are related to the laws of physics. Dowe's primitive causal interaction is an attempt to make the means of mapping the laws of physics onto the terms *cause* and *effect* more precise.

Aaronson (2014) recently made a point similar to that of Laplace.

> The laws of physics aren't "suggestions," which particles obey when they feel like it but ignore when Uri Geller picks up a spoon.[15] … The world "obeys" them in much the same sense that a computer obeys its code.

Hoefer (2015) characterizes this way of thinking as follows.

> The laws of nature are assumed to be *pushy explainers*. They *make things happen in certain ways*. [The laws of nature] are implicitly thought of as the *cause* of everything that happens.

Hoefer goes on to deprecate this way of thinking—as if nature could be distinguished from her laws.

> Talk of laws "governing" … is so commonplace that it takes an effort of will to see it as metaphorical.

In contrast, in a world in which a software execution engine is running some software there are as Aaronson suggests two distinct things: the software and the execution engine. The software plays the role of pushy laws of nature—and hence is the cause of everything that happens. The execution engine plays the role of the material world, which "obeys" those laws. The software governs the engine; there is no metaphor. As indicated above, though, there are fundamental differences.

- The laws of nature are neither expressed symbolically nor executed by physical objects. The laws of nature are not distinct from nature itself. It is only metaphorically that the laws of nature direct the behavior of material objects. In contrast software is distinct from an execution engine. It makes sense to say that software directs the behavior of the engine.
- The physical world needs a source of energy if it is to "obey" the laws. More accurately, the laws of nature have built into them energetic requirements. That is not the case with a software execution engine. A software execution engine is assumed to be self-powered. Within a software-governed world, its energy source is not relevant. Software need not be concerned about whether the execution engine has sufficient energy to run.

Autonomous causal mechanisms

If software is the cause of the behavior of its software execution engine, what is the causal mechanism? The answer, of course, is that software execution engines are built to interpret software instructions. In

---

bodies of the universe and those of the tiniest atom; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes."

[14] We now know that Laplace was incorrect on at least three counts. (i) Quantum randomness make prediction impossible. Laplace could be forgiven for this mistake since he did not know about quantum mechanics. (ii) Chaos makes prediction impossible. We now know that it's impossible (a) to know the forces and positions of nature with sufficient precision and then (b) computationally to propagate those forces and positions in a way that retains that level of precision over an extended time span to achieve Laplace's intended result. Laplace was not aware of chaos theory either. (iii) Biological evolution depends on (random) mutation and recombination. Laplace did not know about evolution, and had he known it's not clear what role he would have assigned to randomness in mutation and recombination. Had he assumed they are truly random, he would have known that prediction is not possible. Had he assumed that there is a non-random mechanism behind them—such as chaotic or quantum phenomena—he would have been stuck with the problems of (i) and (ii).

[15] Uri Geller is a stage magician who "bends" spoons as part of his act. See, for example, (Randi 2006).

an inversion of the usual causal mechanism relationship—in which a cause produces an effect—software execution engines make themselves the effect of the software that serves as the cause. The cause-effect relationship still holds: the software is the cause; the behavior of the execution engine is the effect. It's just that the mechanism is backward. Instead of projecting cause to effect, the engine, whose behavior is the effect, maps the cause onto whatever effect it determines. Because the effect engine serves as the causal mechanism, I'll refer to this (perhaps paradoxically) as *autonomous causality* and to the effect as an *autonomous effect*.

Anything capable of following symbolic instructions illustrates autonomous causality. Besides software execution engines, examples include human beings when acting as drivers obeying traffic signals and as soldiers following orders.

Many entities that serve as autonomous effects can be programmed to provide effect components for multiple causal relationships. Examples include human beings, computers, trainable animals, and now robots. Because they can be programmed to provide the effect component for many different causal relationships, such entities are generally considered valuable resources.

Review

Let's review what we've said about causal relationships.

- We are considering two kinds of causal relationships.
    - Primitive causal interactions are based on an exchange of a conserved quantity.
    - Manipulable causal relationships reflect situations in which an effect can be manipulated via some intervention on the cause. The cause acts as something like a remote control for the effect.
- A manipulable causal relationship is formally equivalent to a functional dependency. As is the case with functional dependencies, manipulable causal relationships may connect sets of static properties as cause and effect.
- Neither the cause nor the effect of a manipulable causal relationship need be either an event or an action.
- All primitive causal interactions are manipulable causal relationships but not vice versa.
- I am calling manipulable causal relationships that are not primitive causal interactions constructed causal relationships.
- For every manipulable causal relationship, even those that connect static sets of properties, it is possible to identify some causal mechanism that brings about the causal relationship. There appear to be (at least) the following categories of causal mechanisms.
    - The causal mechanism may be a cascade of primitive causal interactions that begin at the cause and propagate to the effect thereby projecting changes in the cause onto changes to the effect. Think of a row of dominoes knocking each other over, the gears of a clock, a Rube Goldberg machine, chemical combustion, a nuclear reaction, or mechanisms that maintain homeostatic conditions such as a thermostat.
    - The causal mechanism may exist in the environment—and only under certain conditions—as in the example of an electric switch, a water faucet, or a mirror.
    - The causal mechanism may be built into the framework within which the relationship exists—as in the example of a functional dependency linking ZIP code to state.
    - the causal mechanism may be built into the effect—as in a software execution engine. I am calling this form of causality autonomous causality.

The Switch

The final major bullet in the preceding section tells us that for every causal relationship some mechanism (broadly understood[16]) links the cause to the effect. I'll use the term *Switch* for any causal mechanism not involving the exchange of a conserved quantity.

A *Switch*: a means or mechanism whereby a cause is linked to an effect in such a way that a change to the cause results in a consistent change to the effect without the exchange of a conserved quantity.

Even though a Switch is intended to represent a physical realization of a constructed causal relationship, it is defined quite abstractly.

- In software terms, a Switch is polymorphic: the types of the values to be assigned to the cause and effect are not specified.
- Switch functionality is expressed abstractly: that a change to the cause results in a consistent change to the effect. But the specific change, e.g., the function that maps cause to effect and the mechanism that realizes that mapping are left open. These properties are left to the concrete realization of a Switch in any particular case.
- Switches tend to lack of proportionality. They often have the property known as non-linearity: the magnitude of the change in the cause generally has little or no relationship to the magnitude of the change in the effect. This is not the case for primitive causal interactions. Since these depend on an exchange of a conserved quantity, the magnitude of the change in the cause is linked directly to the magnitude of the change in the effect.

A Switch is a generic mechanisms for building manipulable causal relationships. In software terms a Switch is something like an abstract class; it can be understood as having an abstract method that maps cause values to effect values. To create a concrete manipulable causal relationship one (a) creates a non-abstract Switch subclass with a specific cause and effect and (b) builds a concrete implementation of the cause-to-effect mapping. As we'll see, switches are fundamental to both computing and complex systems.

I'll use the term *autonomous Switch* for the switch component of an autonomous causal relationship.

The Turing machine is an autonomous Switch

Earlier we discussed how the causal mechanism that links software to its execution engine resides in the execution engine. A software execution engine is able to follow instructions expressed symbolically: it reads software and does what the software instructs it to do. This is fundamentally different from most other causal mechanisms. We referred to this as autonomous causality. In this section I want to look at a Turing machine, which I will claim is an autonomous Switch.

A Turing machine is assumed to be in one of a finite number of pre-specified internal states. It is assumed to have a read/write head, which is positioned over a symbol from a finite alphabet—also fixed in advance—arranged in a finite but unbounded sequence. [17] As befits its name, the read/write head can

---

[16] Online definitions of *mechanism* include the following.

- a process or system used to produce a particular result (Merriam-Webster)
- the agency or means by which an effect is produced (Dictionary.com and FreeDictionary.com)
- a natural or established process by which something takes place or is brought about (Google.com and Oxford)
- the technical aspects of doing something (Wordnet.com, Wordnet.dk, and vocabulary.com)

See also the definitions at Onelook.com.

[17] The unboundedness of the symbol sequence does not affect the step-by-step operations the machine is assumed capable of performing. At each step the device reads a symbol and then acts as described.

read the symbol over which it is positioned. Based on its current state and the symbol it reads, the machine (a) writes a replacement symbol (perhaps the same as the original), (b) either does not move the read/write head or moves the read/write head left or right, and (c) either changes its internal state to some different state or remains in the same internal state.

Turing machines are frequently represented as collections of 5-tuples—often written as rules.

$$(\langle state_{pre} \rangle, \langle symbol_{pre} \rangle) \rightarrow (\langle state_{post} \rangle, \langle symbol_{post} \rangle, \langle move \rangle)$$

The *<state>* and *<symbol>* components are the values for the machine's internal state and the symbol being read/written before and after the rule is applied. The *<move>* value is one of *{left, right, no-move}*.

Following Woodward (2003) and Pearl (2009) cause and effect can be represented by variables that assume values from predefined domains. It will often be useful to think of these values as n-tuples. So a cause may have values $C = (c_1, c_2, \ldots, c_n)$, and an effect may have values $E = (e_1, e_2, \ldots, e_m)$. Modifying the cause from C_a to C_b results in a modification to the effect from E_a to E_b.

Given this convention a Turing machine is an abstract *Switch*: the *Cause* value is a *(<state>, <symbol>)* pair; the *Effect* value is a *(<state>, <symbol>, <move>)* triple. A Turing machine is an *abstract* Switch because Turing offered no physical instructions for building his machines. All we can say is that the computational capability of a Turing machine is essentially that of a Switch.

Turing machines are autonomous in that they provide the mechanism that links the cause to the effect. So Turing machines are autonomous Switches. In other words we can reframe the notion of a Turing machine as an autonomous Switch plus unbounded storage.

Causality and functionality

Most software has either a user manual or a functional specification (or both). Either one describes what's known as the software's functionality, i.e., what the software does when executed—assuming that what the software does when executed conforms to its manual or specification. Like a Turing machine, executing software implements an autonomous causal relationship: it's state/input is the cause; its behavior/output is the effect. The executing software itself provides the causal mechanism.

Notwithstanding the unremarkable place the notion of functionality holds in computer science, that same notion seems to raise difficulties for the philosophy of biology. In that context, functionality is typically understood in terms of the role a function plays in the larger context of an organism. For example, one may say that the function of the heart is to pump blood and note that the pumping of blood is necessary for the survival of the organism. When understood in those terms, the heart is implicitly assumed to have a purpose, to do its part in the overall design of the organism. But philosophers object that design and teleology have no place in modern science. Nature doesn't design; nature doesn't operate in terms of purposes. One way around this problem is known as naturalizing functionality. The function exists because it was selected by evolution as one of the mechanisms that resulted in the survival of the organism. From this perspective one would say that the function of the heart is to pump blood, but one wouldn't say that the function of the heart is to produce a semi-regular rhythmic beating sound—even though the heart does both.

From a causality perspective, the heart provides a causal mechanism that links various physiological states to both the circulation of blood and the generation of a beating sound. It seems to me that this is a simpler way of looking at it. Beating noises are a causal effect whether or not they are required for survival. So when speaking of functionality I will be referring to causal effects rather than to biological needs.

Causality and complex systems

Steven Weinberg is a committed reductionist. He adopts (1995) a position he calls *grand reductionism*.

> All of nature is the way it is (with certain qualifications about initial conditions and historical accidents) because of simple universal laws, to which all other scientific laws may in some sense be reduced.

Note how Weinberg brackets off "initial conditions and historical accidents." According to many biologists, much of biology is due to historical accidents and hence would not fall within Weinberg's grand reductionism perspective. As we will see, complex systems are built on the constructed causal relationships of historical accidents. I want to examine two examples.

The gene scwitch

Jacob and Monod (1961) won the Nobel prize for their discovery of what has come to be known as the gene switch. *E. coli* is capable of digesting lactose. To do so it produces an enzyme that helps in the digestion. But production of that enzyme uses cellular resources. Therefore it makes sense to produce the enzyme only when lactose is present.

To accomplish this, *E. coli* produces a protein—known as a repressor—that binds to DNA just "upstream" of the genes for the lactose enzyme. The repressor protein blocks transcription of the enzyme genes. But when lactose is present, a lactose derivative binds to the repressor pulling it off the DNA. This unblocks transcription, resulting in the production of the enzyme. This is a Switch. The presence or absence of lactose is the cause; production of the lactose enzyme is the effect.

Although the preceding sketches the essential causal process, as with most things in biology, the actual mechanism is more complex. Since Jacob and Monod's discovery, many examples of gene regulation processes have been observed. Savageau (2012) describes the situation like this.

> The rich variety of mechanisms governing gene expression is the result of historical accident. Why are there positive and negative regulators? God only knows. There is no design—what works, works; what doesn't is dead. The only rule is that there are no rules.

Plant defense against herbivores

As a second example consider how plants respond to herbivores. (Paraphrased from War et. al., 2012)

> Plants confront herbivores both directly—by producing specialized morphological structures or secondary metabolites and proteins that have toxic, repellent, and/or anti-nutritional effects on the herbivores—and indirectly. Indirect defenses are mediated by the release of a blend of Herbivore Induced Plant Volatiles (HIPVs) that attract natural enemies of the herbivores. For example, mechanically wounded cabbage leaves emit volatiles that attract parasitic wasps which prey on the herbivores.

What I find especially worth noting is the combination of historical accident and causality. Through an extended process of co-evolution—more than 1/3 billion years—plants and the enemies of the plant's enemies have "agreed" on signaling compounds. The particular chemicals don't matter. All that matters is (a) that the chemical can be produced and emitted by the plant, (b) that it can be carried by air currents, and (c) that herbivore predators can evolve to respond to the chemical as an attractant.

In causal terms, the attack of a herbivore is linked as a cause via a complex causal mechanism to an effect, which is a counter attack by a herbivore predator. In other words, the plant has turned an attack into a cause that produces a counter-attack effect. Such networks of contingently constructed causal relationships are typical of complex systems—which is one reason they are considered complex!

Bricolage

Evolution is one historical accident piled atop another. A mutation or recombination that happens to enhance survival or reproduction is preserved. Once it exists, other evolutionary accidents can build on it. In "Évolution et bricolage" Jacob (1977) described this phenomenon as follows (slightly edited).

> It is hard to realize that the living world as we know it is just one among many possibilities—that its actual structure results from the history of the earth. Yet living organisms are historical structures, literally creations of history. They represent, not a perfect product of engineering, but a patchwork of odd sets pieced together when and where opportunities arose. The opportunism of natural selection is not simply a matter of indifference to the structure and operation of its products. It reflects the very nature of a historical process, full of contingency.
>
> [Evolution] works like a tinkerer, a tinkerer who does not know exactly what he is going to produce but uses whatever odds and ends are at his disposal. What he ultimately produces is generally related to no special project. Each product can be used in a number of different ways. What they have in common is "it might be of some use." For what? That depends on future opportunities.

Yet each evolutionary product embodies a constructed causal relationship. The environment on which the causal mechanism depends is almost always the product of a history of historical accidents. Evolution thus reflects both causality in the sense of constructed causal relationships and randomness in the form of the accidental environments on which the causal relationships depend

Conclusion

Causality, which can be immensely confusing when one attempts to see it as a relationship between events makes much more sense when understood in terms of manipulable causal relationships. From that perspective causality can be seen as a foundation for both computing and complexity.

References

Aaronson, Scott (2014) This review of Max Tegmark's book also occurs infinitely often in the decimal expansion of π. *Shetl-Optimized.*

Dowe, Phil (2000) *Physical Causation*, Cambridge University Press.

Hiddleston, Eric (2005) Review of Woodward, Making things happen. *The Philosophical Review*, Vol. 114, No. 4 (Oct., 2005), pp. 545-547

Hoefer, Carl (2015, forthcoming) Causal Determinism. Zalta, Edward N. (ed), *The Stanford Encyclopedia of Philosophy*.

Imai, Kosuke, Dustin Tingley, and Teppei Yamamoto (2013) Experimental Designs for Identifying Causal Mechanisms. *Journal of the Royal Statistical Society: Series A (Statistics in Society)* 176, no. 1 (January 2013): 5–51.

Jacob, Francois (1977) Evolution and Tinkering. *Science*. Vol. 196, No. 4295. (Jun. 10, 1977), pp. 1161-1166.

Jacob, Francois and Jacques Monod (1961) Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology* 3: 318-56.

Little, Daniel (2004) Causal mechanism. Michael Lewis-Beck, Alan Bryman, and Tim Futing Liao (eds) *Encyclopedia of Social Science Research Methods*, Sage Publications.

Paul, Laurie and Ned Hall (2012) *Causation: a user's guide*. Oxford University Press.

Pearl, Judea (2009) *Causality* (2e). Cambridge University Press.

Randi, James (2006) James Randi exposes Uri Geller and Peter Popoff. *YouTube*.

Savageau, Michael A. (2012) Phenotypes and Design Principles in System Design Space. In Walhout, Marian, Marc Vidal, Job Dekker (eds) *Handbook of Systems Biology: Concepts and Insights*. Academic Press.

Schaeffer, Jonathan (2014) The Metaphysics of Causation. Zalta, Edward N. (ed.) *The Stanford Encyclopedia of Philosophy* (Summer 2014 Edition).

War, A. R., Paulraj, M. G., Ahmad, T., Buhroo, A. A., Hussain, B., Ignacimuthu, S., & Sharma, H. C. (2012). Mechanisms of plant defense against insect herbivores. *Plant Signaling & Behavior,* 7(10), 1306–1320. doi:10.4161/psb.21663

Weinberg, Steven (1995). Reductionism redux. *The New York Review of Books*.

Wikipedia (2015) Rube Goldberg machine. *Wikipedia*.

Woodward, James (2003) *Making thigs happen*. Oxford University Press.

Woodward, James (2013) Causation and Manipulability. Edward N. Zalta (ed.)*The Stanford Encyclopedia of Philosophy* (Winter 2013 Edition).

Woodward, James (2012/forthcoming) A Functional Account of Causation, Or, A Defense of the Legitimacy of Causal Thinking by Reference to the Only Standard that Matters—Usefulness (as Opposed to Metaphysics or Agreement with Intuitive Judgment). (2012 Philosophy of Science Association Presidential Address). Forthcoming in *Philosophy of Science*.