



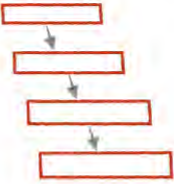
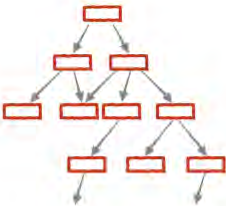
Contents

= **STEPHEN WOLFRAM** WRITINGS

RECENT | CATEGORIES | Q

# Even beyond Physics: Introducing Multicomputation as a Fourth General Paradigm for Theoretical Science

September 9, 2021

 <p><b>structural</b> (antiquity)</p> <p>e.g. geometrical elements</p> <p>explicit time not considered</p> <p>static facts deduced by reasoning</p>	 <p><b>mathematical</b> (1600s)</p> <p>e.g. differential equations</p> <p>time as mathematical coordinate</p> <p>find behavior at any time from formula</p>	 <p><b>computational</b> (1980s)</p> <p>e.g. cellular automata</p> <p>time as progress of computation</p> <p>determine future only by running program</p>	 <p><b>multicomputational</b> (2020s)</p> <p>e.g. multiway systems</p> <p>many computational threads of time</p> <p>need model of observer to determine state</p>
--	--	---	--

## *The Path to a New Paradigm*

One might have thought it was already exciting enough for our [Physics Project](#) to be showing a [path to a fundamental theory of physics](#) and a fundamental description of how our physical universe works. But what I've [increasingly been realizing](#) is that actually it's showing us something even bigger and deeper: a whole fundamentally new paradigm for making models and in general for doing theoretical science. And I fully expect that this new paradigm will give us ways to address a remarkable range of longstanding central problems in all sorts of areas of science—as well as suggesting whole new areas and new directions to pursue.

modeling paradigms that have been developed over the course of scientific history—each of them leading to dramatic progress. The first, originating in antiquity, one might call the “structural paradigm”. Its key idea is to think of things in the world as being constructed from some kind of simple-to-describe elements—say geometrical objects—and then to use something like logical reasoning to work out what will happen with them. Typically this paradigm has no explicit notion of time or dynamical change, though in its modern forms it often involves making descriptions of structures of relationships, usually built from logical or “flowchart-like” elements.

Many would say that modern exact science was launched in the 1600s with the introduction of what we can call the “[mathematical paradigm](#)”: the idea that things in the world can be described by mathematical equations—and that their behavior can be determined by finding solutions to these equations. It’s common in this paradigm to discuss time—but normally it’s just treated as a variable in the equations, and one hopes that to find out what will happen at some arbitrary time one can just substitute the appropriate value for that variable into some formula derived by solving the equations.

For three hundred years the mathematical paradigm was the state of the art in theoretical science—and immense progress was made using it. But there remained plenty of phenomena—particularly associated with complexity—that this paradigm seemed to have little to say about. But then—basically starting in the early 1980s—there was a burst of progress based on a new idea (of which, yes, I seem to have ultimately been the primary initiator): the [idea of using simple programs](#), rather than mathematical equations, as the basis for models of things in nature and elsewhere.

Part of what this achieves is to generalize beyond traditional mathematics the kind of constructs that can appear in models. But there is something else too—and it’s from this that the full computational paradigm emerges. In the mathematical paradigm one imagines having a mathematical equation and then separately somehow solving it. But if one has a program one can imagine just directly taking it and running it to find out what it does. And this is the essence of the computational paradigm: to define a model using computational rules (say, for a [cellular automaton](#)) and then explicitly be able to run these to work out their consequences.

intrinsic. In the mathematical paradigm it's in effect just the arbitrary value of a variable. But in the computational paradigm it's a direct reflection of the actual process of applying computational rules in a model—or in other words in this paradigm the passage of time corresponds to the actual progress of computation.

A major discovery is that in the computational universe of possible programs even ones with very simple rules **can show immensely complex behavior**. And this points the way—through the **Principle of Computational Equivalence**—to **computational irreducibility**: the phenomenon that there may be no faster way to find out what a system will do than just to trace each of its computational steps. Or, in other words, that the passage of time can be an irreducible process, and it can take an irreducible amount of computational work to predict what a system will do at some particular time in the future. (Yes, this is closely related not only to things like undecidability, but also to things like the **Second Law of Thermodynamics**.)

In the full arc of scientific history, the computational paradigm is very new. But in the past couple of decades, it's seen rapid and dramatic success—and by now it's **significantly overtaken the mathematical paradigm** as the most common source for new models of things. Despite this, however, **fundamental physics always seemed to resist its advance**. And now, from our Physics Project, we can see why.

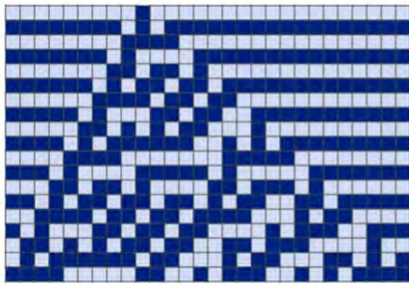
Because at the core of our Physics Project is actually a new paradigm that goes beyond the computational one: a fourth paradigm for theoretical science that I'm calling the multicomputational paradigm. There've been hints of this paradigm before—some even going back a century. But it's only as a result of our Physics Project that we've been able to start to see its full depth and structure. And to understand that it really is a fundamentally new paradigm—that transcends physics and applies quite generally as the foundation for a new and broadly applicable methodology for making models in theoretical science.

## *Multway Systems and the Concept of the Multicomputation*

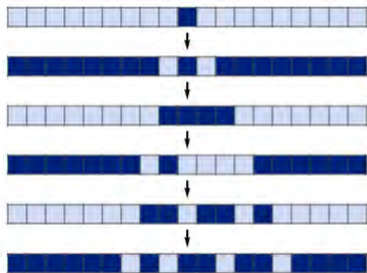
In the ordinary computational paradigm the typical setup is to have a system that evolves in a series of steps by repeatedly applying some particular rule. **Cellular automata** are a quintessential example. Given a rule like



Contents



as corresponding to a sequence of states of the cellular automaton:

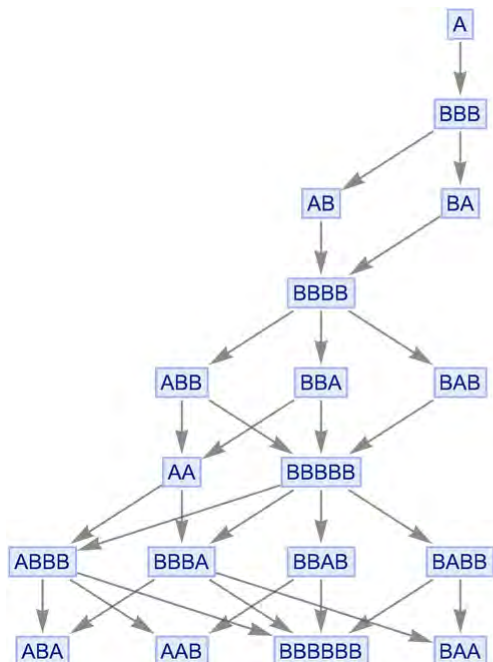


The essence of the multicomputational paradigm is to generalize beyond just having simple linear sequences of states, and in effect to allow multiple interwoven threads of history.

Consider as an example a system defined by the string rewrite rules:

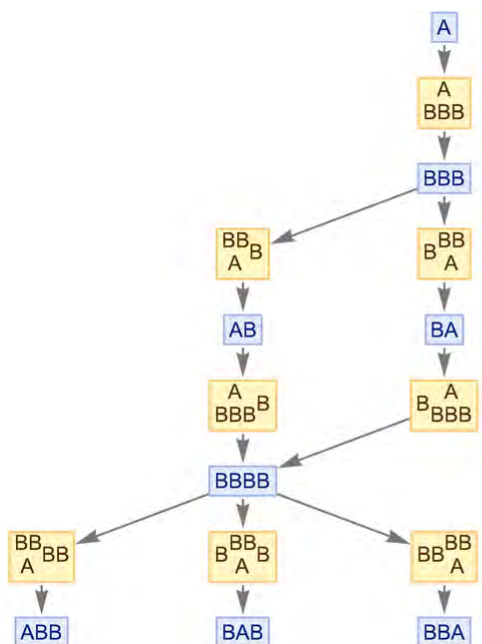
$$\{A \rightarrow BBB, BB \rightarrow A\}$$

Starting from A, the next state has to be BBB. But now there are two possible ways to apply the rules, one generating AB and the other BA. And if we trace both possibilities we get what I call a **multiway system**—whose behavior we can represent using a **multiway graph**:



## Contents

application as an “updating event”. And then the point is that even within a single string multiple updating events (shown here in yellow) may be possible—leading to multiple branches in the multiway graph:



At first, one might want to say that while many branches are in principle possible, the system must somehow in any particular case always choose (even if perhaps “non-deterministically”) a single branch, and therefore a particular history. But a key to the multicomputational paradigm is not to do this, and instead to say that “what the system does” is defined by the whole multiway graph, with all its branches.

In the ordinary computational paradigm, time in effect progresses in a linear way, corresponding to the successive computation of the next state of the system from the previous one. But in the multicomputational paradigm there is no longer just a single thread of time; instead one can think of every possible path through the multiway system as defining a different interwoven thread of time.

If we look at the four paradigms for theoretical science that we’ve identified we can now see that they involve successively more complicated views of time. The structural paradigm doesn’t directly talk about time at all. The mathematical paradigm does consider time, but treats it as a mathematical variable whose value can in a sense be arbitrarily chosen. The computational paradigm treats time as reflecting the progression of a computation. And now the multicomputational paradigm treats time as something multithreaded, reflecting the interwoven progression of multiple threads of computation.

There are multiway systems based on rewriting not only strings, but also [trees](#), [graphs](#) or [hypergraphs](#). There are also multiway systems that [work just with numbers](#). It's even possible (though not especially natural) to define [multiway cellular automata](#). And in fact, whenever there's a system where a single state can be updated in multiple ways, one's led to a multiway system. (Examples include [games where multiple moves are possible at each turn](#), and computer systems with asynchronous or distributed elements that operate independently.)

And once one has the idea of multiway systems it's amazing how often they end up being the most natural models for things. And indeed one can see them as minimal models pretty much whenever there's no rigid built-in notion of time, and no predefined specification of “when things happen” in a system.

But right now the “killer app” for multiway systems is our Physics Project. Because what we seem to be learning is that in fact our whole universe is operating as a giant multiway system. And it's the limiting properties of that multiway system that give us space and time and relativity and quantum mechanics.

### *Observers, Reference Frames and Emergent Laws*

In the mathematical paradigm one expects to immediately “read off” from a model what happens at a particular time. In the computational paradigm one might have to run an irreducible computation, but then one can still “read off” what happens after a certain time. But in the multicomputational paradigm, it's more complicated—because now there are multiple threads of time, with no intrinsic way to line up “what happens when” across different threads.

But imagine you're trying to see what's going on in a multicomputational system. In principle you could keep track of the behaviors on all the threads as well as the complicated interweavings between them. But a crucial fact about us as observers is that we don't normally do that. Instead, we normally combine things so that we can describe the system as somehow just progressively “evolving through time”.

There might in principle be some alien intelligence that routinely keeps track of all the different threads. But we humans—and the descriptions we use of the world—[always tend to sequentialize things](#). In other words, in order to understand “what's happening in the



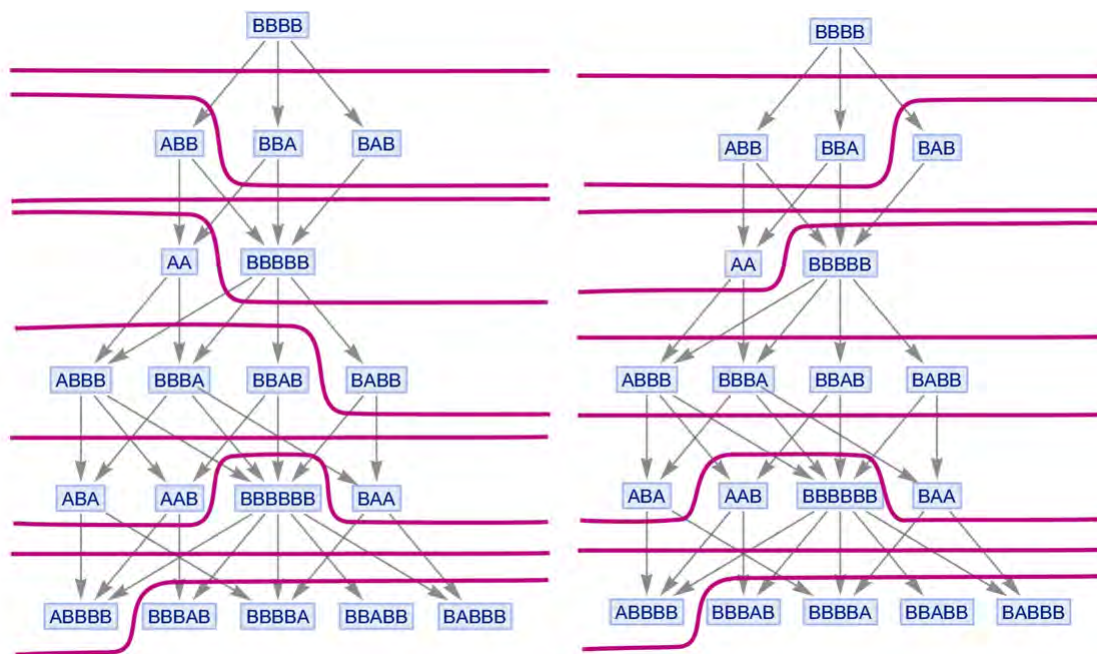
## Contents

that is “merely computational”. Instead of following lots of different “local times” on different threads, we try to think about things in terms of a single “global time”.

And this isn’t just something we do “for convenience”; the tendency to “sequentialize” like this is directly related to our perception that we have a single thread of experience, which seems to be a key **defining feature of our notion of consciousness** and our general way of relating to the world.

But how should we line up different threads of time in a multicomputational system? A crucial point is that there typically isn’t just “one natural way” to do it. Instead, there are many possible choices. And it’s “up to the observer” which one to use—and therefore “how to parse” the behavior of the multicomputational system.

The underlying structure of the multiway system puts constraints on what’s possible, but typically there are **many ways of choosing a sequence of “time slices”** that successively sample the behavior of the system. Here are two choices of how to do this for the multiway system above:



In both cases the underlying multicomputational behavior is the same. But the “experience” of the observer is different. And—taking a term used in relativity theory that we’ll later see captures exactly the same idea—we can consider the different choices of time slices as different “reference frames” from which to view what’s going on.

Contents

---

(though the system does put constraints on what reference frames are possible). Instead, the reference frame is just something the observer “uses to understand the system”. But as soon as an observer sequentializes time—as I **believe we characteristically do**—then essentially by definition they must be using some reference frame.

In the ordinary computational paradigm there are fundamental limits on our prediction or understanding of the behavior of systems, associated with the phenomenon of **computational irreducibility**. And things get even more difficult when it comes to multicomputational systems—where not only can individual threads of history show computational irreducibility, but also these threads can interweave in computationally irreducible ways.

But what will an observer with a certain reference frame perceive about the multicomputational system? Well, it depends on the reference frame. And for example one might imagine that one could have a very elaborate reference frame that somehow “untangles” the computational irreducibility associated with the weaving of different threads and delivers some arbitrarily different “perception” of what’s going on.

But now there’s another crucial point: actual observers such as us don’t use arbitrary reference frames; they only use computationally bounded ones. In other words, there’s a limit to how complicated the reference frame can be, and how much computation it can effectively serve to “decode”.

If the observer is somehow embedded inside the multicomputational system (as must be the case if, for example, the system corresponds to the fundamental physics of our whole universe), then it’s necessary and inevitable that the observer (being a subpart of the whole system)—and the reference frames they use—must be computationally bounded. But the notion of a computationally bounded observer is actually something much more general—and as we’ll see in a series of examples later—it’s a central part of multicomputational models for all sorts of systems.

By the way, we’ve discussed sequentialization in time separately from computational boundedness. But in some sense sequentialization in time is actually just a particular example of computational boundedness that happens to be very obvious and significant for us humans. And potentially some alien intelligence could act as a computationally bounded observer with some other way of “simplifying time”.



irreducible way. And we have a computationally bounded observer who's "parsing" the multicomputational system using particular reference frames. What will that observer perceive about the behavior of the system?

Well, here's the crucial and surprising thing that's emerged from our Physics Project: with the setup for multicomputational systems that we've described, the observer will almost inevitably perceive the system to follow laws that are simple enough to be captured by mathematical equations. And in the case of physics these laws basically [correspond in different situations to general relativity and to quantum mechanics](#).

In other words, despite the complexity of the underlying behavior of the multicomputational system, the comparative simplicity of the observer makes them inevitably sample only certain "simple aspects" of the whole behavior of the multicomputational system. In computational terms, the observer is perceiving a computationally reducible slice of the whole computationally irreducible behavior of the system.

But what exactly will they perceive? And how much does it depend on the details of the underlying computationally irreducible behavior? Well, here's something very crucial—and surprising—about multicomputational systems: there's a lot that can be said quite generically about what observers will perceive, largely independent of the details of underlying computationally irreducible behavior.

It's deeply related to (but more general than) the result in thermodynamics and statistical physics that there are generic laws for, say, the perceived behavior of gases. At an underlying level, gases consist of large numbers of molecules with complicated and computationally irreducible patterns of motion. But a computationally bounded observer perceives only certain "coarse-grained" features—which don't depend on the underlying properties of the molecules, and instead correspond to the familiar generic laws for gases.

And so it is in general with multicomputational systems: that quite independent of the details of underlying computationally irreducible behavior there are generic ("computationally reducible") laws that computationally bounded observers will perceive. The specifics of those laws will depend on aspects of the observer (like their sequentialization of time). But the fact that there will be such laws seems to be an essentially inevitable consequence of the core structure of multicomputational systems.

## Contents

~~AS SOON AS ONE IMAGINES THAT EVENTS CAN OCCUR WHEREVER AND WHATEVER RULES ALLOW, THIS~~  
inevitably leads to a kind of inexorable combinatorial structure of interwoven “threads of time” that necessarily leads to certain “generic perceptions” by computationally bounded observers. There can be great complexity in the underlying behavior of multicomputational systems. But there’s a certain inevitable overall structure that gets revealed when observers sample the systems. And that inevitable structure can manifest itself in fairly simple laws for certain aspects of the system.

A characteristic feature of systems based on the ordinary computational paradigm is the appearance of computational irreducibility and complex behavior. And with such systems it’s perfectly possible to have computationally bounded observers who sample this complex behavior and **reduce it to rather simple features**. But what tends to happen is that rather little is left; the observer has in a sense crushed everything out. (Imagine, say, an observer averaging the colors of a complex-enough-to-seem-random sequence of black and white cells to a simple uniform gray.)

But with a multicomputational system, things work differently. Because there’s enough inevitable structure in the fundamental multicomputational setup of the system that even when it’s sampled by a somewhat arbitrary observer there are still nontrivial effective laws that remain. And in the case of fundamental physics we can identify these laws as general relativity and quantum mechanics.

But the point is that because these laws depend only on the fundamental setup of the system, and on certain basic properties of the observer, we can expect that they will apply quite generally to multicomputational systems. Or, in other words, that we can expect to identify overall laws in basically any multicomputational system—and those laws will in effect be direct analogs of general relativity and quantum mechanics.

In ordinary computational systems there’s a very powerful general result: the **Principle of Computational Equivalence**, which leads to computational irreducibility. And this result also carries over to multicomputational systems. But in multicomputational systems—which basically inevitably have to be sampled by an observer—there’s an additional result: that from the fundamental structure of the system (and the observer) there’s a certain amount of computational reducibility, which leads to certain specific overall laws of behavior.

complicated—going from the ordinary computational paradigm to the multicomputational one—we’d inevitably have less to say about how systems generally behave. But actually—basically because of the interplay of the observer with the fundamental structure of the system—it’s the exact opposite. And that’s very important when it comes to theoretical science. Because it means that systems that seemed like they would show only unreachably complex behavior can actually have features that are described by definite overall laws that are potentially within reach even of the mathematical paradigm.

Or, in other words, if one analyzes things correctly through the multicomputational paradigm, it’s potentially possible to find overall laws even in situations and fields where this seemed hopeless before.

### *Leveraging Ideas from Physics*

The multicomputational paradigm is something that’s emerging from our [Physics Project](#), and from [thinking about fundamental physics](#). But one of the most powerful things about having a general paradigm for theoretical science is that it implies a certain unity across different areas of science—and by providing a common framework it allows results and intuitions developed in one area to be transferred to others.

So with its roots in fundamental physics the multicomputational paradigm immediately gets to leverage the ideas and successes of physics—and in effect use them to illuminate other areas.

But just how does the multicomputational paradigm work in physics? And how did it even arise there? Well, it’s not something that the traditional mathematical approach to physics would readily lead one to. And instead what basically happened is that having seen how successful the computational paradigm was in studying lots of kinds of systems [I started wondering whether something like it](#) might apply to fundamental physics.

It was fairly clear, though, that the ordinary computational paradigm—especially with its “global” view of time—wasn’t a great match for what we already knew about things like relativity in physics. But the pivotal idea that eventually led inexorably to the multicomputational paradigm was a hint from the computational paradigm about the nature of space.

## Contents

just as a kind of “mathematical source of coordinates”. But in the computational paradigm one tends to imagine that everything is ultimately made of discrete computational elements. So in particular I began to think that this might be true of space.

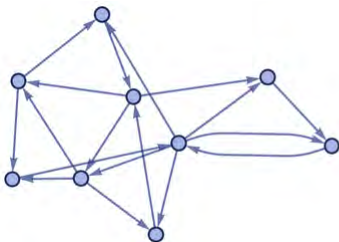
But how would the elements of space behave? The computational approach would suggest that there must be “finitely specifiable” rules—that effectively define “update events” involving limited numbers of elements of space. But right here is where the multicomputational idea comes in. Because inevitably—across all the elements of space in our universe—there must be a huge number of different ways these updating events could be applied. And the result is that there is no just one unique “computational history”—but instead a whole multiway system with different threads of history for different sequences of updating events.

As we’ll discuss later, it’s the updating events—and the relations between them—that are in a sense really the most fundamental things in the multicomputational paradigm. But in understanding the multicomputational paradigm, and its way of representing fundamental physics, it’s helpful to start instead by thinking about what the updating events act on, or, in effect, the “data structure of the universe”.

A convenient way to set this up is to imagine that the universe—or, in particular, space and everything in it—is **defined by a large number of relations between the elements of space**. Representing each element of space by an integer, one might have a collection of (in this case, binary) relations like

$\{\{2, 3\}, \{0, 4\}, \{5, 0\}, \{0, 2\}, \{2, 5\}, \{3, 5\}, \{6, 1\}, \{1, 2\}, \{2, 6\}, \{0, 6\}, \{7, 1\}, \{1, 3\}, \{3, 7\}, \{0, 7\}, \{8, 4\}, \{4, 0\}, \{0, 8\}, \{1, 8\}\}$

which can in turn be thought of as defining a graph (or, in general a hypergraph):



But now imagine a rule like

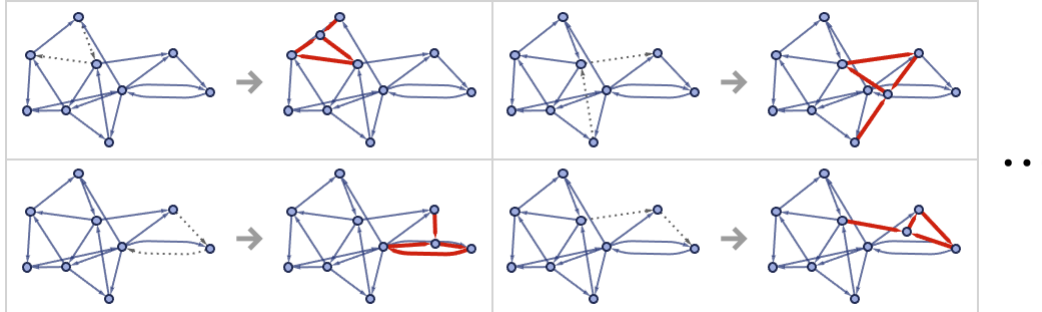
$\{\{x, y\}, \{y, z\}\} \rightarrow \{\{w, y\}, \{y, z\}, \{z, w\}, \{x, w\}\}$

or, stated pictorially,

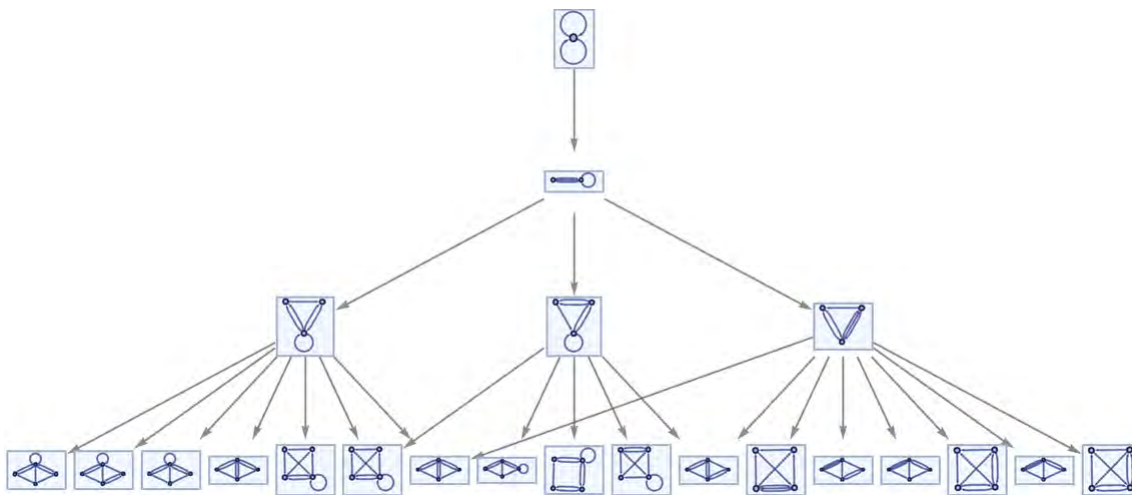
Contents



that specifies what updating events should occur. There are in general many different places where a rule like this can be applied to a given hypergraph:



So—in a multicomputational fashion—we can define a multiway graph to represent all the possibilities (here starting from  $\{\{0,0\},\{0,0\}\}$ ):

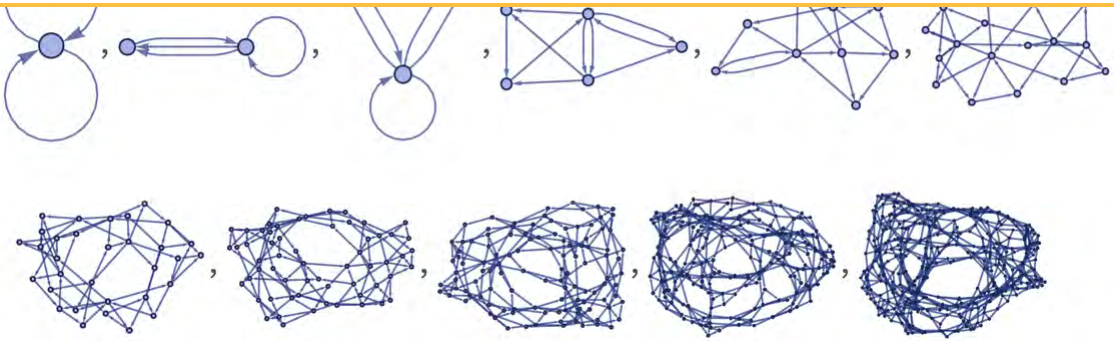


In our model of fundamental physics, the presence of many different branching and merging paths is a reflection of quantum mechanics—with every path in effect representing a history for the universe.

But to get at least some idea of “what the universe does” we can imagine following a particular path, and seeing what hypergraphs are generated:

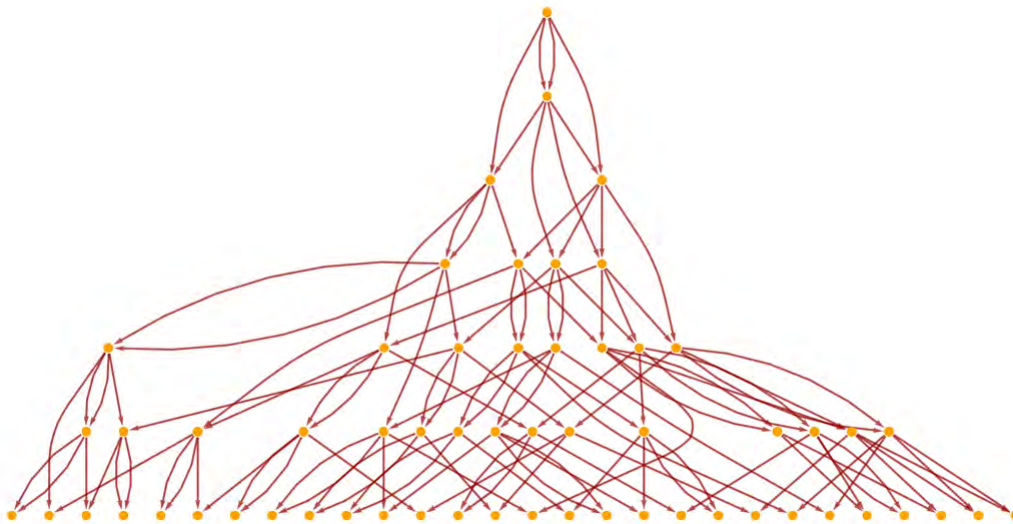


## Contents



And the concept is that after a large number of steps of such a process we'll get a recognizable representation of an “instantaneous state of space” in the universe.

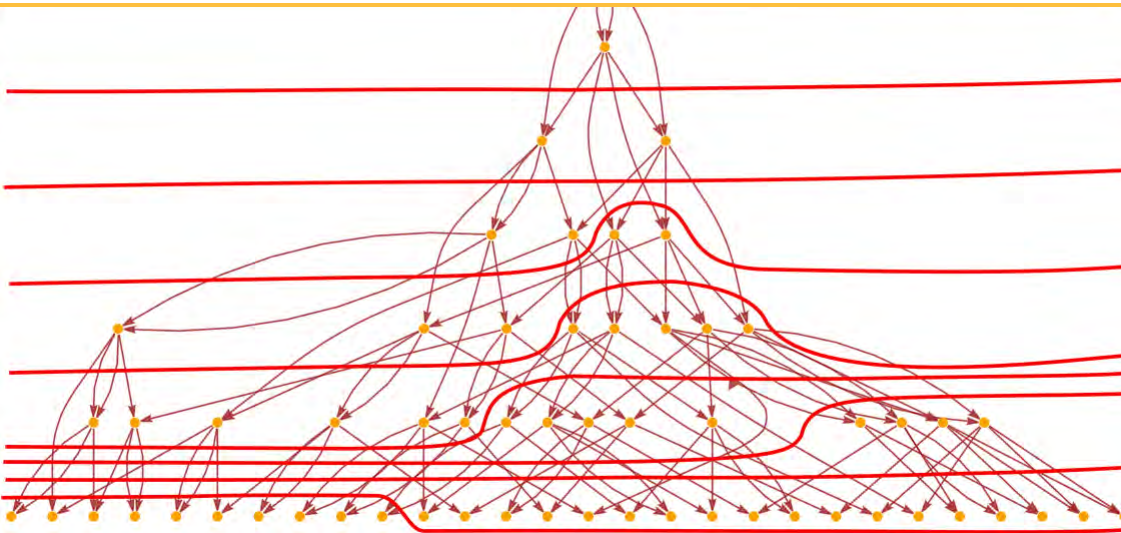
But what about time? Ultimately it's the individual updating events that define the progress of time. Representing updating events by nodes, we can now draw a **causal graph** that shows the “causal relationships” between these updating events—with each edge representing the fact that the “output” from one event is being “consumed” as “input” by another event:



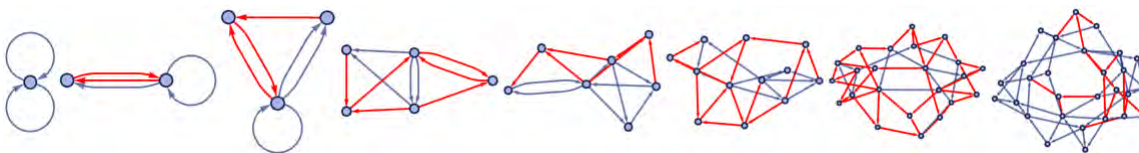
And as is characteristic of the multicomputational paradigm this causal graph reflects the fact that there are many possible sequences in which updating events can occur. But how does this jibe with our everyday impression that a definite sequence of things happen in the universe?

The basic point is that we don't perceive the whole causal graph in all its detail. Instead, as computationally bounded observers, we just pick some particular reference frame from which to perceive what's going on. And this reference frame defines a sequence of global “time slices” such as:

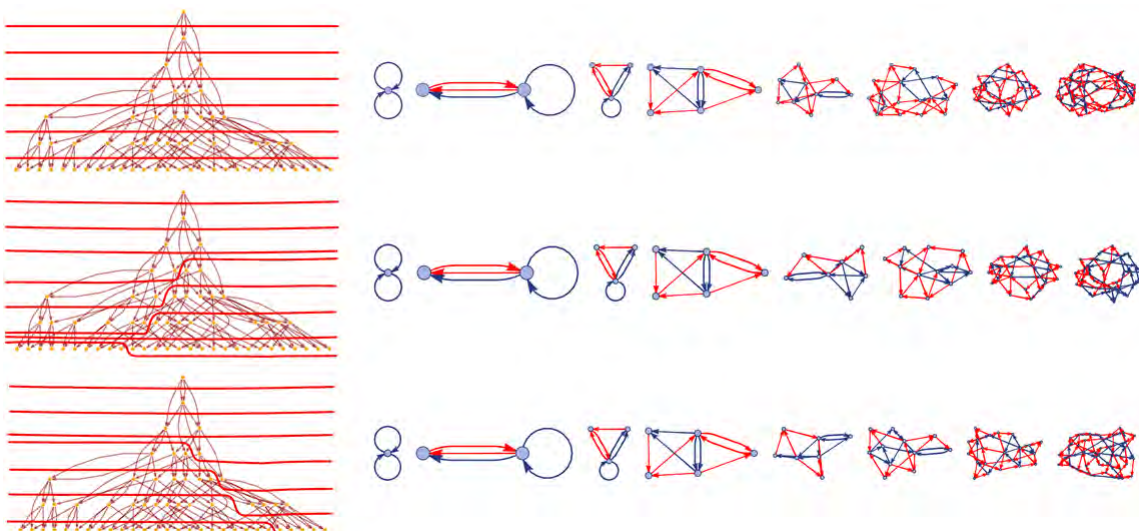
Contents



Each “time slice” contains a collection of events that—with our reference frame—we take to be “happening simultaneously”. And we can then trace the “steps in the evolution of the universe” by seeing the results of all updating events in successive time slices:



But how do we determine what reference frame to use? The underlying rule determines the structure of the causal graph, and what event can follow what other one. But it still allows huge freedom in the choice of reference frame—in effect imposing only the constraint that if one event follows another, then these events must appear in that order in the time slices defined by the reference frame:



In general each of these different choices of reference frame will lead to a different sequence of “instantaneous states of space”. And in principle one could imagine that some elaborately

practice there is an important constraint on possible reference frames: as computationally bounded observers we are limited in the amount of computational effort that we can put into the construction of the reference frame.

And in general to achieve a “[pathological result](#)” we’ll typically have to “reverse engineer” the underlying computational irreducibility of the system—which we won’t be able to do with a reference frame constructed by a computationally bounded observer. (This is directly analogous to the result in the ordinary computational paradigm that computationally bounded observers effectively can’t avoid perceiving the validity of the [Second Law of Thermodynamics](#).)

So, OK, what then will an observer perceive in a system like the one we’ve defined? With a variety of caveats the basic answer is that in the limit of a “sufficiently large universe” they’ll perceive average behavior that’s simple enough to describe mathematically, and specifically to describe as [following Einstein’s equations from general relativity](#). And the key point is that this is in a sense a generic result (a bit like the gas laws in thermodynamics) that’s independent of the details of the underlying rule.

But there’s more to this story. We’ll talk about it a bit more formally in the next section. But the basic point is that so far we’ve just talked about picking reference frames in a “spacetime causal graph”. But ultimately we have to consider the whole multiway graph of all possible sequences of update events. And then we have to figure out how an observer can set up some kind of reference frame to give them a perception of what’s going on.

At the core of the concept of a reference frame is the idea of being able to treat certain things (typically events) as somehow “equivalent”. In the case of the causal graphs we’ve discussed so far, what we’re doing is to treat certain events as equivalent in the sense that they can be viewed as happening “in the same time slice” or effectively “simultaneously”. But if we just pick two events at random, there’s no guarantee that it’ll be consistent to consider them to be in the same time slice.

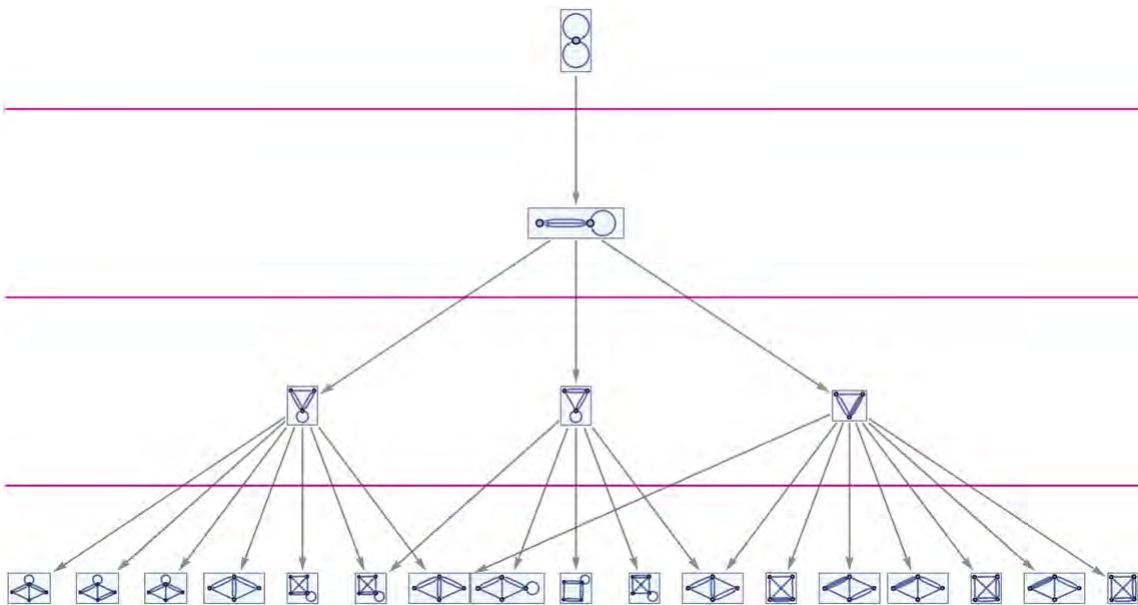
In particular, if one event causally depends on another (in the sense that its input requires output from the other), then it can only occur in a later time slice. And in this situation (which corresponds to one event being reachable from the other by following directed edges in the causal graph) we can say that these events are “timelike separated”. Similarly, if two events can occur in the same time slice, we can say that they are “spacelike separated”. And

spacetime—or at least discrete analogs of them.

So what about with the full multiway graph? We can look at every event that occurs in every state in the multiway graph. And there are then basically three kinds of separation between events. There can be timelike separation, in the sense that one event causally depends on another. There can be spacelike separation, in the sense that different events occur in different parts of space that are not causally connected. And then there's a third case, which is that different events can occur on different branches of the multiway graph—in which case we say that they're branchlike separated.

And in general when we pick a reference frame in the full multiway system, we can have time slices that contain both spacelike- and branchlike-separated events. What's the significance of this? Basically, just as spacelike separation is associated with the concept of ordinary space, branchlike separation is associated with a different kind of space, that we call **branchial space**.

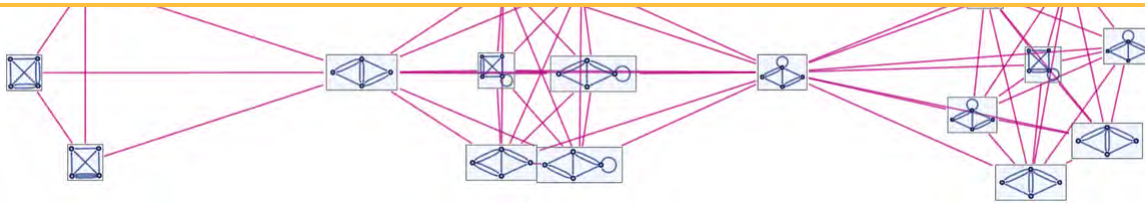
With a multiway graph of the kind we've drawn above (in which every node represents a possible “complete state of the universe”), we can investigate “pure branchial space” by looking at time slices in the graph:



For example, we can construct “**branchial graphs**” by looking at which states are connected by having immediate common ancestors. And in effect these branchial graphs are the branchial-space analogs of the hypergraphs we've constructed to represent the instantaneous state of ordinary space:



## Contents



But now, instead of representing ordinary space—with features like general relativity and gravity—they represent something different: they represent a space of quantum states, with the branchial graph effectively being a map of quantum entanglements.

But to define a branchial graph, we have to pick the analog of a reference frame: we have to say what branchlike-separated events we consider to happen “at the same time”. In the case of spacelike-separated events it’s fairly easy to interpret that what we get from a reference frame is a view of what’s happening throughout space at a particular time. But what’s the analog for branchlike-separated events?

In effect what we’re doing when we make a reference frame is to treat as equivalent events that are happening “on different branches of history”. At first, that may seem like a very odd thing to do. But the thing to understand is that as entities embedded in the same universe that’s generating all these different branches of history, we too are branching. So it’s really a question of how a “branching brain” will perceive a “branching universe”. And that depends on what reference frame (or “quantum observation frame”) we pick. But **as soon as we insist** that we maintain a single thread of experience, or, equivalently, that we sequentialize time, then—together with computational boundedness—this puts all sorts of constraints on the reference frames we pick.

And just as in the case of ordinary space, the result is that it ultimately seems to be possible to give a fairly simple—and essentially mathematical—description of what the observer will perceive. And the answer is that it basically appears to correspond to quantum mechanics.

But there’s actually more to it. What we get is a kind of generic multicomputational result—that doesn’t depend on the details of underlying rules or particular choices of reference frames. Structurally it’s basically the same result as for ordinary space. But now it’s interpreted in terms of branchial space, quantum states, and so on. And what was interpreted as the geodesic equation of general relativity **now essentially gets interpreted as** the path integral of quantum mechanics.



physics that quantum mechanics is the same theory as general relativity—though operating in branchial space rather than ordinary space.

There are important implications here for physics. But there are also general implications for all multicomputational systems. Because the sophisticated definitions and phenomena of both general relativity and quantum mechanics we can now expect will have analogs in any system that can be modeled in a multicomputational way, whatever field of science it may come from.

So, later, when we talk about the application of the multicomputational paradigm to other fields, we can expect to talk and reason in terms of things we know from physics. So we'll be able to bring in light cones, inertial frames, time dilation, black holes, the uncertainty principle, and much more. In effect, the common use of the multicomputational paradigm will allow us to leverage the development of physics—and its status as the most advanced current area of theoretical science—to “physicalize” all sorts of other areas, and shed new light on them. As well, of course, as taking ideas and intuition from other areas (including ones much closer to everyday experience) and “applying them back” to physics.

### *The Formal Structure of Multicomputation*

In the previous section, we discussed how the multicomputational paradigm plays out in the particular case of fundamental physics. And in many ways, physics is probably a fairly typical application of the paradigm. But there are some special features it has that add complication, though also add concreteness.

So what are the ultimate foundations of the multicomputational paradigm? At its core, I think it's fair to say that the paradigm is about events and their relationships—where the events are defined by some kind of rules.

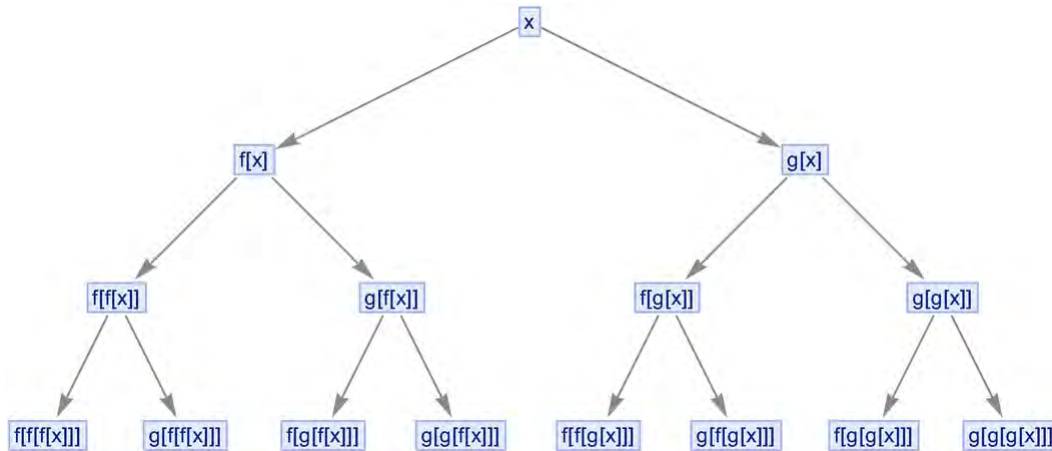
What happens in an event? It's a bit like the application of a function. It takes some set of “expressions” or “tokens” as input, and returns some other set as output.

In a simple ordinary computational system there might just be one input and one output expression for each event, as in  $x \mapsto f[x]$ , leading to a trivial graph for the sequence of states reached in the evolution of the system:



## Contents

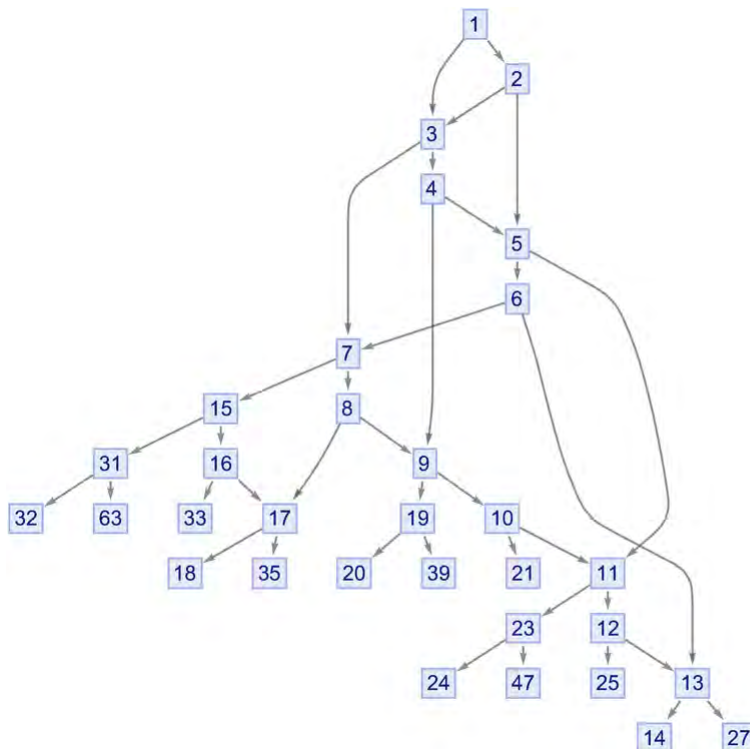
$g[x]$ . Then the evolution of the system can instead be represented by the tree:



But what starts making a nontrivial multiway graph is when some states that are generated in different ways end up being the same, so that they get merged in the graph. As an example, consider the rule

$$n \mapsto \{n+1, 2n+1\}$$

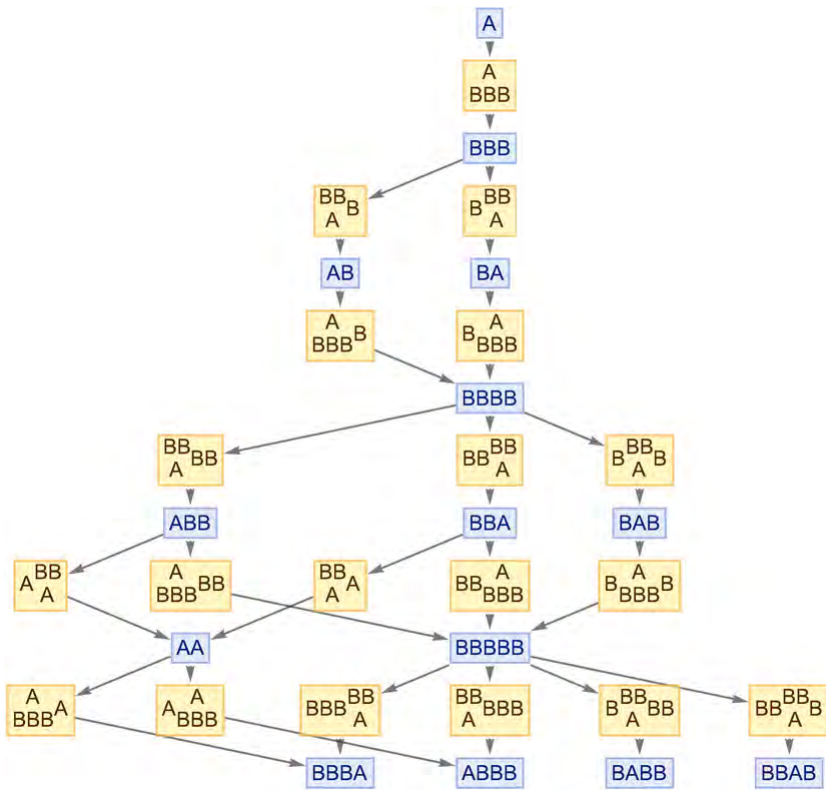
The multiway graph produced in this case is



where now we see merging even at the top of the graph, associated, for example, with the equivalence of  $1 + 1 + 1$  and  $2 \times 1 + 1$ . And as we can see, even with this very simple rule, the multiway graph is not so simple.

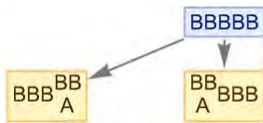
Contents

affects causal dependence: in all its events complete states (here integers) are used as input and output. But in a string-based system (say with a rule like  $A \rightarrow BBB$ ,  $BB \rightarrow A$ ) the situation is different. Because now the events can act on just part of the string:



And it's the same when we use hypergraphs—as in our models of fundamental physics. The events don't typically apply to complete hypergraphs, but instead to subhypergraphs within them.

But let's look a bit more carefully at the string case above. When we see different update events for a given string, we can identify two different cases. The first is a case like



where the updates don't overlap, and the second is a case like

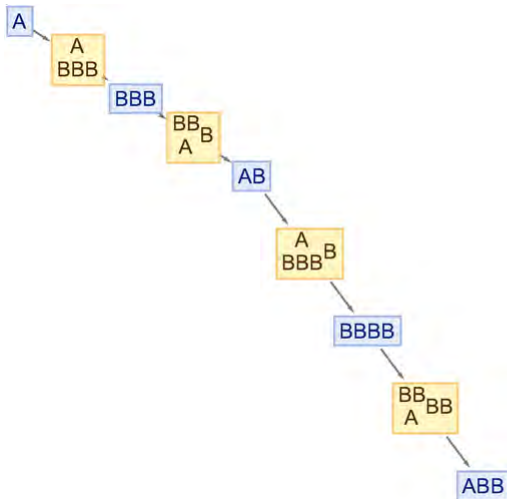


where they do. And what we'll find is that in the first case we can consider the events to be purely spacelike separated, while in the second they are also branchlike separated.

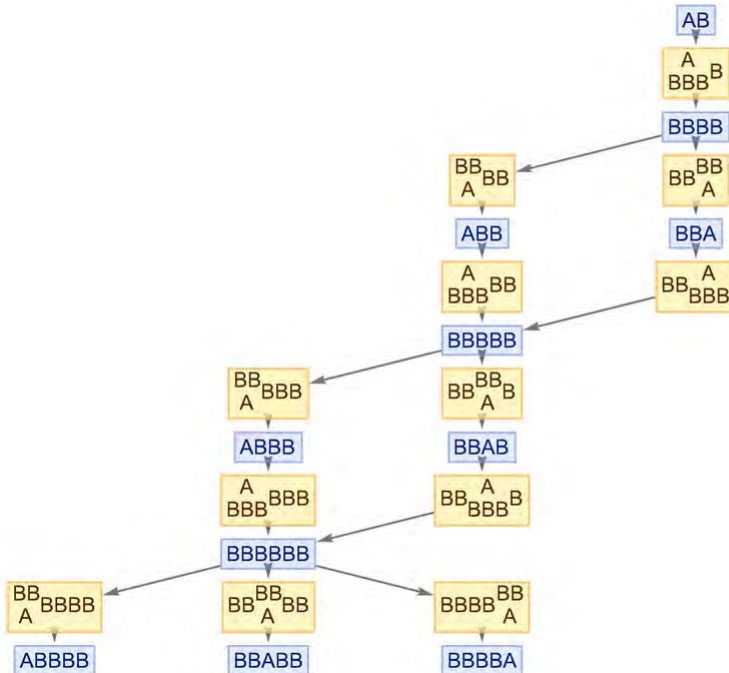
## Contents

obtained by running all possible update events on each state (i.e. string) that is generated.

But what if we choose for example just to use the first update event found in a left-to-right scan of each state (so we've got a "sequential substitution system")? Then we'd get a "single-way" graph with no branching:



As another "evaluation strategy" we could scan the string at each step, applying all updates that don't overlap:

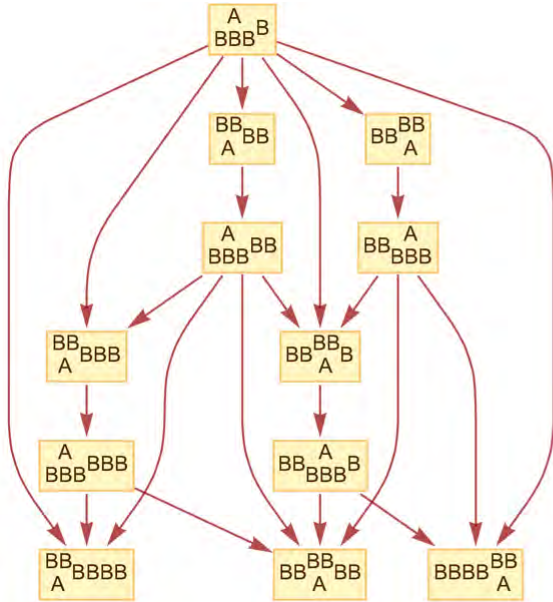


Both the results we've just got are subgraphs of the full multiway graph. But they both have the feature that they effectively just yield a single sequence of states for the system. In the first case this is obvious. In the second case there are little "temporary branchings" but they always merge back into a single state. And the reason for this is that with the evaluation

## Contents

multiway branches”, as would be generated by branchlike-separated events.

But even though ultimately there’s just a “single branch of history” there’s a “shadow” of the presence of other branches visible in the nontrivial causal graph that shows causal relationships between updating events:



So what about our models of physics? In exploring them it’s often convenient not to track the whole multiway system but instead just to look at the results from a particular “evaluation strategy”. In a hypergraph there’s no obvious “scan order”, but we still often use a strategy that—like our second string strategy above—attempts to “do whatever can be done in parallel”.

Inevitably, though, there’s a certain arbitrariness to this. But it turns out there’s a **more “principled” way to set things up**. The basic idea is not to think about complete states (like strings or hypergraphs) but instead just to think separately about the “tokens” that will be used as inputs and outputs in events.

And—giving yet more evidence that even though hypergraphs may be hard for us humans to handle, there’s great naturalness to them—it’s considerably easier to do this for hypergraphs than for strings. And the basic reason is that in a hypergraph each token automatically comes “appropriately labeled”.

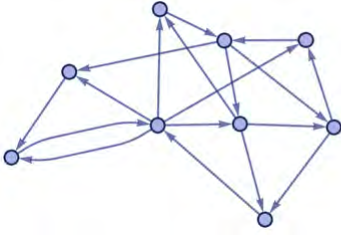
The relevant tokens for hypergraphs are hyperedges. And in a rule like

$$\{\{x, y\}, \{y, z\}\} \rightarrow \{\{w, y\}, \{y, z\}, \{z, w\}, \{x, w\}\}$$



## Contents

generated as output. And the point is that when we have a hypergraph like



that corresponds for example to

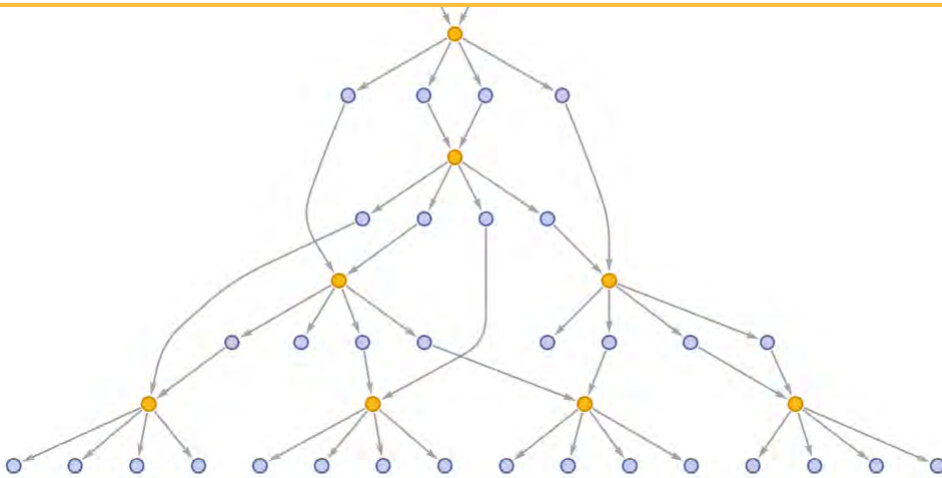
$\{\{2, 3\}, \{0, 4\}, \{5, 0\}, \{0, 2\}, \{2, 5\}, \{3, 5\}, \{6, 1\}, \{1, 2\}, \{2, 6\}, \{0, 6\}, \{7, 1\}, \{1, 3\}, \{3, 7\}, \{0, 7\}, \{8, 4\}, \{4, 0\}, \{0, 8\}, \{1, 8\}\}$

we can just think of this as an unordered “sea” (or multiset) of hyperedges, where each event will just pick up some pair of hyperedges that match the pattern  $\{\{x, y\}, \{y, z\}\}$ . But what does it mean to “match the pattern”?  $x, y, z$  can each correspond to any node of the hypergraph (i.e. for our model of physics, any atom of space). But the key constraint is that the two instances of  $y$  must refer to the same node.

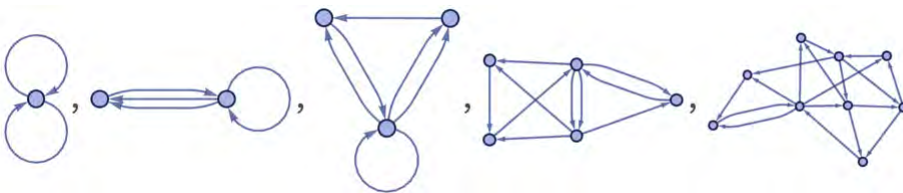
If we tried to do the same thing for strings, it’d be **considerably more difficult**. Because then the relevant tokens would be individual characters in the string. But whereas in a hypergraph every token is a hyperedge that can be identified from the uniquely named nodes it contains, every A or every B in a string is usually thought of as just being the same—giving us no immediate way to identify distinct tokens in our system.

But assuming we have a way to identify distinct tokens, we can consider representing the evolution of our system just in terms of events applied to tokens (or what we can call a “token-event graph”). This is going to get a bit complicated. But here’s an example of how it works for the hypergraph system we’ve just shown. Each blue node is a token (i.e. a hyperedge) and each yellow node is an event:

Contents

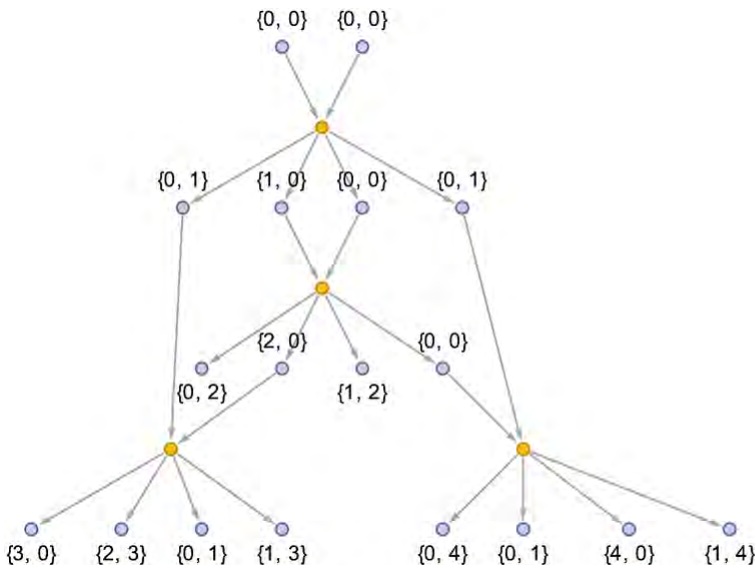


What’s going on here? At each step shown, there’s a token (i.e. blue node) for every hyperedge generated at that step. Let’s compare with the overall sequence of hypergraphs:



The initial state contains two hyperedges, so there are two tokens at the top of the token-event graph. Both these hyperedges are “consumed” by the event associated with applying the rule—and out of that event come four new hyperedges, represented by the four tokens on the next step.

Let’s look in a little more detail at what’s happening. Here is the beginning of the token-event graph, annotated with the actual hyperedges represented by each token (the numbers in the hyperedges are the “IDs” assigned to the “atoms of space” they involve):

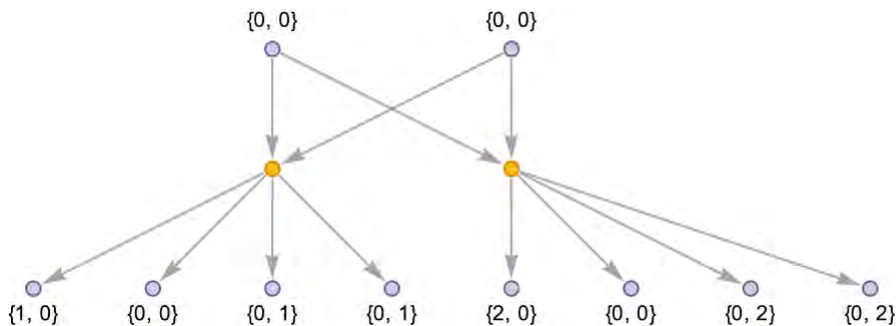


$\{0,0\}$  and  $\{0,0\}$  and generates four new hyperedges. (Note that the  $\{0,0\}$  on step 2 is considered a “new hyperedge”, even though it happens to have the same content as both hyperedges on step 1; more about this later.) At the second step, the rule consumes  $\{1,0\}$  and  $\{0,0\}$ . And at the third step, there are two invocations of the rule (i.e. two events), each consuming two hyperedges, and generating four new ones.

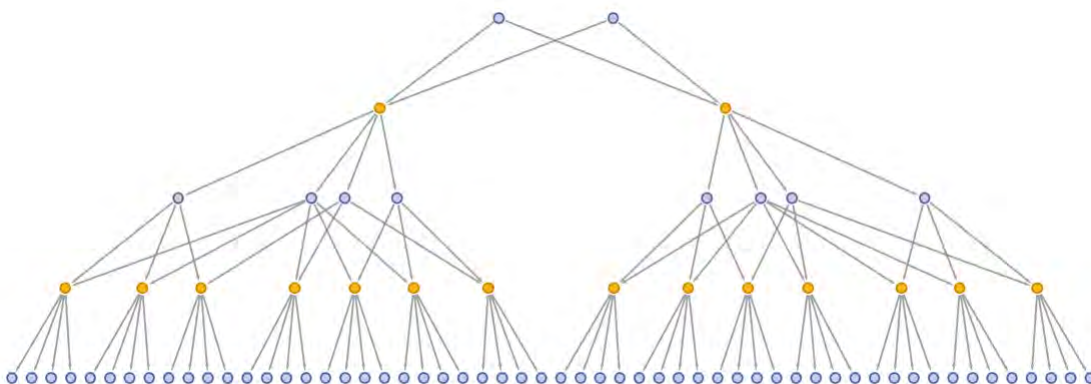
And looking at this one might ask “Why did the second event consume  $\{1,0\}$  and  $\{0,0\}$  rather than, say,  $\{1,0\}$  and one of the  $\{0,1\}$  tokens?” Well, the answer is that the token-event graph we’re showing is just for a specific possible history, obtained with a particular “evaluation strategy”—and this is what that strategy picked to do.

But it’s possible to extend our token-event graph to show not just what can happen for a particular history, but for all possible histories. In effect what we’re getting is a finer-grained version of our multiway graph, where now the (blue) nodes are not whole states (i.e. hypergraphs) but instead just individual tokens (i.e. hyperedges) from inside those states.

Here’s the result for a single step:

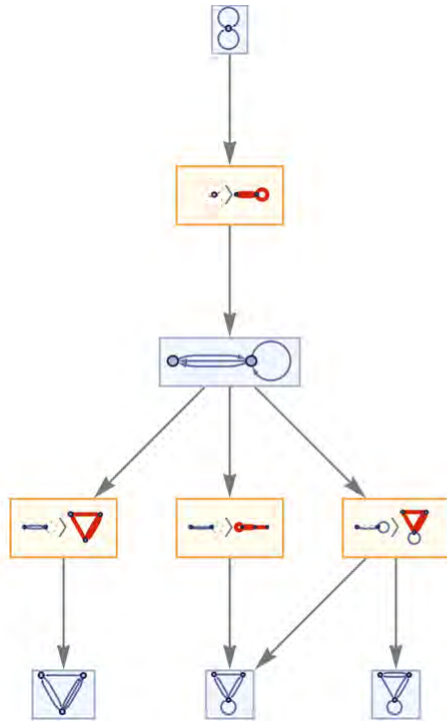


There are two possible events because the two initial hyperedges given here can in effect be consumed in two different orders. Continuing even one more step things rapidly get significantly more complicated:



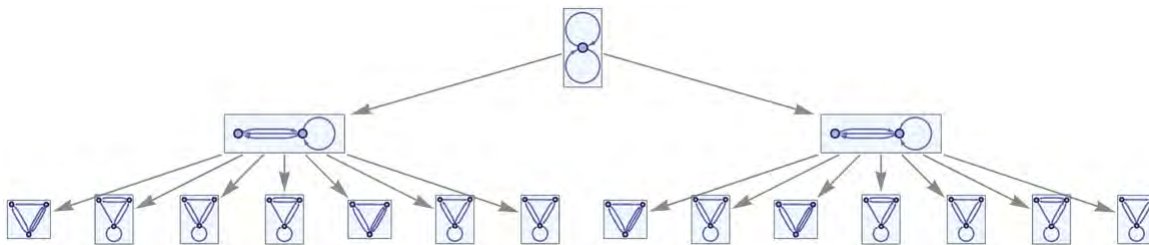
## Contents

system:



Why is this so much simpler? First, it's because we've collected the individual tokens (i.e. hyperedges) into complete states (i.e. hypergraphs)—“knitting them together” by seeing which atoms they have in common. But we're doing something else as well: even if hypergraphs are generated in different ways, we're conflating them whenever they're “the same”. And for hypergraphs our [definition of “being the same”](#) is that they're isomorphic, in the sense that we can transform them into each other just by permuting node labels.

Note that if we don't conflate isomorphic hypergraphs the multiway graph we get is



which corresponds much more obviously to the token-event graph above.

When we think about multicomputational systems in general, the conflation of “identical” (say by isomorphism) states is in a sense the “lowest-level act” of an observer. The “true underlying system” might in some sense “actually” be generating lots of separate, identical states. But if the observer can't tell them apart then we might as well say they're all “the same state”. (Of course, when there are different numbers of paths that lead to different

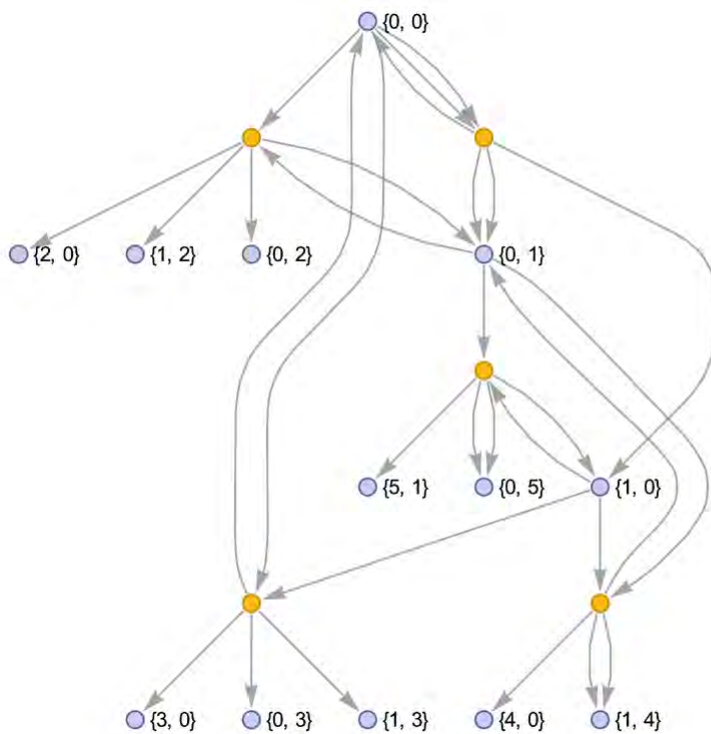
physics this is where the different magnitudes of quantum amplitudes for different states come from.)

It seems natural and obvious to conflate hypergraphs if they're isomorphic. But actual observers (say humans observing the physical universe) typically conflate much, much more than that. And indeed when we say that we're operating in some particular reference frame we're basically defining potentially huge collections of states to conflate.

But there's actually also a much lower level at which we can do conflation. In the token-event graphs that we've looked at so far, every token generated by every event is shown as a separate node. But—as the labeled versions of these graphs make clear—many of these tokens are actually identically the same, in the sense that they're just direct copies created by our way of computing (and rendering) the token-event graph.

So what about conflating all of these—or in effect “deduplicating” tokens so that we have just one unique shared representation of every token, regardless of how many times or where it appears in the original graph?

Here's the result after doing this for the 2-step version of the token-event graph above:



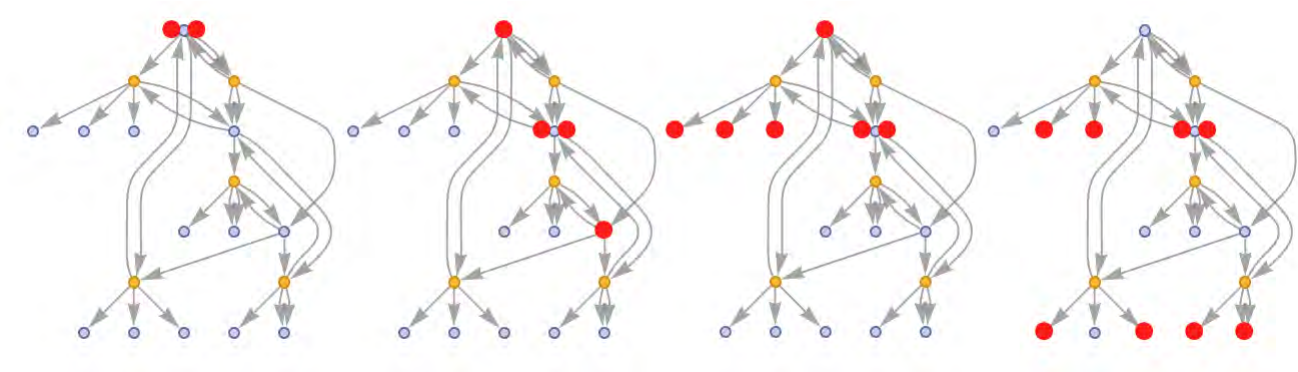
This deduplicated-token-event graph in essence records every “combinatorially possible” “distinct event” that yields a “transition” between distinct tokens. But while sharing the



## Contents

has a definite notion of “progress in time”: there are edges “going both up and down”, sometimes even forming loops (i.e. “closed timelike curves”).

So how can this graph represent the actual evolution of the original system with a particular evaluation strategy (or, equivalently, as “viewed in a particular reference frame”)? Basically what we need are some kind of “markers” that move around the graph from “step to step” to indicate which tokens are “reached” at each step. And doing this, this is what we get for the “standard evaluation strategy” above:

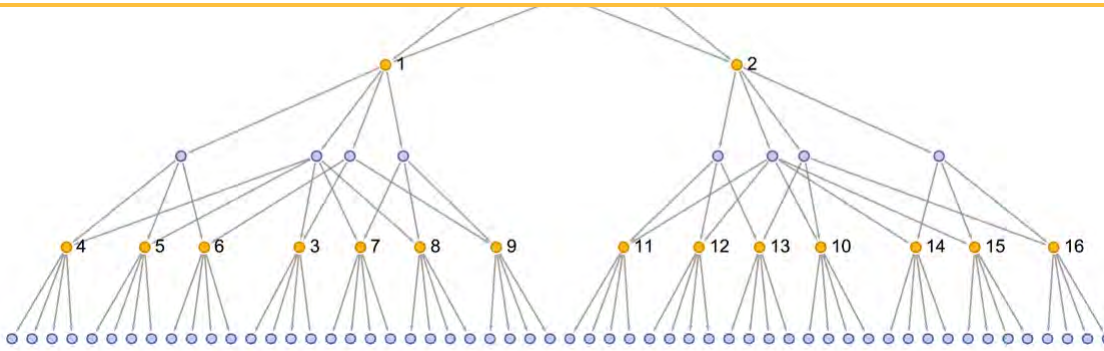


In a sense a deduplicated token-event graph is the ultimate minimal invariant representation of a multicomputational system. (Note that in physics and elsewhere the graph is typically infinite.) But any particular observer will effectively make some kind of sampling of this graph. And in working out this sampling we’re going to encounter issues about knitting together states and reference frames—that are ultimately equivalent to what we saw before from our earlier perspective on multiway systems.

(Note that if we had a deduplicated token-event graph with markers that is finite then this would basically be a [Petri net](#), with decidable reachability results, etc. But in most relevant cases, our graphs won’t be finite.)

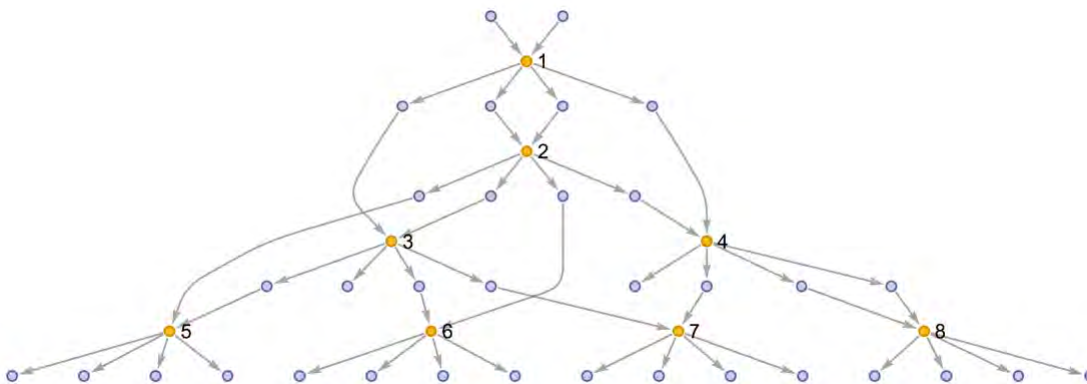
So although token-event graphs—even in deduplicated form—don’t ultimately avoid the complexities of other representations of multicomputational systems, they do make some things easier to discuss. For example, in a token-event graph there’s a straightforward way to read off whether two events are branchlike or spacelike separated. Consider the “all histories” token-event graph we had before:

## Contents



To figure out the type of separation between events, we just need to look at their first common ancestor. If it's a token, that means the events are branchlike separated (because there must have been some “ambiguity” in how the token was transformed). But if the common ancestor is an event, that means the events we're looking at are spacelike separated. So, for example, events 1 and 2 here are branchlike separated, as are 4 and 5. But events 4 and 9 are spacelike separated.

Note that if instead of looking at an “all histories” token-event graph, we restrict to a single history then there will only be spacelike- (and timelike-) separated events:



A token-event graph is in a sense a lowest-level representation of a multicomputational system. But when an observer tries to “see what's going on” in the system, they inevitably conflate things together, effectively perceiving only certain equivalence classes of the lowest-level elements. Thus, for example, an observer might tend to “knit together” tokens into states, and pick out particular histories or particular sequences of time slices—corresponding to using what we can call a certain “reference frame”. (In mathematical terms, we can think of particular histories as like fibrations, and sequences of time slices as like foliations.)

of reference frames are “reasonable” based on some general model for the observer. And one almost inevitable constraint is that it should only require bounded computational effort to construct the reference frame.

Our Physics Project then suggests that in appropriate large-scale limits specific structures like general relativity and quantum mechanics should then emerge. And it seems likely that this is a general and very powerful result that’s essentially inexorably true about the limiting behavior of any multicomputational system.

But if so, the result represents an elaborate—and unprecedented—interweaving of fundamentally computational and fundamentally mathematical concepts. Maybe it’ll be possible to use a generalization of category theory as a bridge. But it’s going to involve not only discussing ways in which operations can be applied and composed, but also what the computational costs and constraints are. And in the end computational concepts like computational reducibility are going to have to be related to mathematical concepts like continuity—I suspect shedding important new light all around.

Before closing our discussion of the formalism of multicomputation there’s something perhaps still more abstract to discuss—that we can call “[rulial multicomputation](#)”. In ordinary multicomputation we’re interested in seeing what happens when we follow certain rules in all possible ways. But in rulial multicomputation we go a step further, and also ask about following all possible rules.

One might think that following all possible rules would just “make every possible thing happen”—so there wouldn’t be much to say. But the crucial point is that in a (rulial) multiway system, different rules can lead to equivalent results, yielding a whole elaborately entangled structure of possible states and events.

But at the level of the token-event formalism we’ve discussed above, rulial multicomputation in some sense just “makes events diverse”, and more like tokens. For in a rulial multiway system, there are lots of different kinds of events (representing different rules)—much as there are different kinds of tokens (containing, for example, different underlying elements or atoms).

And if we look at different events in a rulial multiway system, there is now another possible form of separation between them: in addition to being timelike, spacelike or branchlike

once again we can ask about an observer “parsing” the (rulial) multiway system, and defining a reference frame that can, for example, treat events in a single “rulelike hypersurface” as equivalent.

I’ve [discussed elsewhere](#) the rather remarkable implications of rulial multiway systems for our understanding of the physical (and mathematical) universe, and its fundamental formal necessity. But here the main point to make is that the presence of many possible rules doesn’t fundamentally affect the formalism for multicomputational systems; it just requires the observer to define yet more equivalences to reduce the “raw multiway behavior” to something computationally simple enough to parse.

And although one might have thought that adding the concept of multiple rules would just make everything more complicated, I won’t be surprised if in the end the greater “separation” between “raw behavior” and what the observer can perceive will actually make it easier to derive robust general conclusions about overall behavior at the level of the observer.

### *Physicalized Concepts in Multicomputation*

From the basic definitions of multicomputation it’s hard to have much intuition about how multicomputational systems will work. But knowing how multicomputation works in our model of fundamental physics immediately gives us not only powerful intuition, but also all sorts of metaphors and language for describing multicomputation in pretty much any possible setting.

As we saw in the previous section, at the lowest level in any multicomputational system there are what we can call (in correspondence with physics) “events”—that represent individual applications of rules, “progressing through time”. We can think of these rules as operating on “tokens” of some kind (and, yes, that’s a term from computation, not physics). And what these tokens represent will depend on what the multicomputational system is supposed to be modeling. Often (as in our Physics Project) the tokens will consist of combinations of elements—where the same elements can be shared across different tokens. (In the case of our Physics Project, we view the elements as “atoms of space”, with the tokens representing connections among them.)

together” to define something like space. But there is another, more robust way as well: whenever a single event produces multiple tokens it effectively defines a relation between those tokens. And the map of all such relations—which is essentially the token-event graph—defines the way that different tokens are knitted together into some kind of generalized spacetime structure.

At the level of individual events, ideas from the theory and practice of computation are useful. Events are like functions, whose “arguments” are incoming tokens, and whose output is one or more outgoing tokens. The tokens that exist in a certain “time slice” of the token-event graph together effectively represent the “data structure” on which the functions are acting. (Unlike in basic sequential programming, however, the functions can act in parallel on different parts of the data structure.) The whole token-event graph then gives the complete “execution history” of how the functions act on the data structure. (In an ordinary computational system, this “execution history” would essentially form a single chain; a defining feature of a true multicomputational system is that this history instead forms a nontrivial graph.)

In understanding the analogy with our everyday experience with physics, we’re immediately led to ask what aspect of the token-event graph corresponds to ordinary, physical space. But as we’ve discussed, the answer is slightly complicated. As soon as we set up a foliation of the token-event graph, effectively dividing it into a sequence of time slices, we can say that the tokens on each slice correspond to a certain kind of space, “knitted together” by the entanglements of the tokens defined by their common ancestry in events.

But the kind of space we get is in general something beyond ordinary physical space—effectively something we can call “[multispace](#)”. In the specific setup of our Physics Project, however, it’s possible to define at least in certain limits a decomposition of this space into two components: one that corresponds to ordinary physical space, and one that corresponds to what we call branchial space, that is effectively a map of entangled possible quantum states. In multicomputational systems set up in different ways, this kind of decomposition may work differently. But given our everyday intuition—and mathematical physics knowledge—about ordinary physical space it’s convenient first to focus on this in describing the general “physicalization” of multicomputational systems.

limit of slices of the token–event graph that can be represented as “spatial hypergraphs”. In the Physics Project we imagine that at least in the current universe, the [effective dimension of the spatial hypergraph \(measured, for example, through growth rates of geodesic balls\)](#) corresponds to the observed 3 dimensions of physical space. But it’s important to realize that the underlying structure of multicomputation doesn’t in any way require such a “tame” limiting form for space—and in other settings (even branchial space in physics) things may be much wilder, and much less amenable to present-day mathematical characterization.

But the picture in our Physics Project is that even though there is all sorts of computationally irreducible—and seemingly random—underlying behavior, physical space still has an identifiable large-scale limiting structure. Needless to say, as soon as we talk about “identifiable structure” we’re implicitly assuming something about the observer who’s perceiving it. And in seeing how to leverage intuition from physics, it’s useful to discuss what we can view as the [simpler case of thermodynamics and statistical mechanics](#).

At the lowest level something like a gas consists of large numbers of discrete molecules interacting according to certain rules. And it’s almost inevitable that the detailed behavior of these molecules will show computational irreducibility—and great complexity. But to an observer who just looks at things like average densities of molecules the story will be different—and the observer will just perceive [simple laws like diffusion](#).

And in fact it’s the very complexity of the underlying behavior that leads to this apparent simplicity. Because a computationally bounded observer (like one who just looks at average densities) won’t be able to do more than just read the underlying computational irreducibility as being like “simple randomness”. And this means that for such an observer it’s going to be reasonable to model the overall behavior by using mathematical concepts like statistical averaging, and—at least at the level of that observer—to describe the system as showing computationally reducible behavior represented, say, by the diffusion equation.

It’s interesting to note that the emergence of something like diffusion depends on the presence of certain (identifiable) underlying constraints in the system—like conservation of the number of molecules. Without such constraints, the underlying computational irreducibility would lead to “pure randomness”—and no recognizable larger-scale structure. And in the end it’s the interplay of identifiable underlying constraints with identifiable features of the observer that leads to identifiable emergent computational reducibility.



Contents

---

the “identifiable constraints” are much more abstract ones having to do with the fundamental structure of multicomputation. But much as we can say that the detailed computationally irreducible behavior of underlying molecules leads to things like large-scale fluid mechanics at the level of practical (“coarse-grained”) observers, so also we can say that the detailed computationally irreducible behavior of the hypergraph that represents space leads to the large-scale structure of space, and things like Einstein’s equations.

And the important point is that because the “constraints” in multicomputational systems are generic features of the basic abstract structure of multicomputation, the emergent laws like Einstein’s equations can also be expected to be generic, and to apply with appropriate translation to all multicomputational systems perceived by observers that operate at least somewhat like the way we operate in perceiving the physical universe.

Any system in which the same rules get applied many times must have a certain ultimate uniformity to its behavior, manifest, for example, in the “same laws” applying “all over the system”. And that’s why, for example, we’re not surprised that physical space seems to work the same throughout the physical universe. But given this uniformity, how do there come to be any identifiable features or “places” in the universe, or, for that matter, in other kinds of systems that are constructed in similar ways?

One possibility is just that the observer can choose to name things: “I’ll call this token ‘Tokie’ and then I’ll trace what happens, and describe the behavior of the universe in terms of the ‘adventures of Tokie’”. But as such, this approach will inevitably be quite limited. Because a feature of multicomputational systems is events are continually happening, consuming existing tokens and creating new ones. In physics terms, there is nothing fundamentally constant in the universe: everything in it (including space itself) is being continually recreated.

So how come we have any perception of permanence in physics? The answer is that even though individual tokens are continually being created and destroyed, there are overall patterns that are persistent. Much like vortices in a fluid, there can for example be essentially topological phenomena whose overall structure is preserved even though their specific component parts are continually changed.

[elementary particles](#), with all their various elaborate symmetries. And it's not clear how much of this structure will carry over to other multicomputational systems, but we can expect that there will be some kinds of persistent “objects”—corresponding to certain pockets of local computational reducibility.

An important idea in physics is the concept of “pure motion”: that “objects” can “move around in space” and somehow maintain their character. And once again the possibility of this depends on the observer, and on what it means that their “character is maintained”. But we can expect that as soon as there is a concept of space in a multicomputational system there will also be a concept of motion.

What can we say about motion? In physics, we can discuss how it will be perceived in different reference frames—and for example we define inertial frames that explore space and time differently precisely so as to “cancel out motion”. This leads to phenomena like time dilation, which we can view as a reflection of the fact that if an object is “using its computational resources to move in space” then it has less to devote to its evolution in time—so it will “evolve less in a certain time” than if it wasn't moving.

So if we can identify things like motion (and, to make it as simple as possible, things like inertial frames) in any multicomputational system, we can expect to see phenomena like time dilation—though potentially translated into quite different terms.

What about phenomena like gravity? In physics, energy (and mass) act as a “source of gravity”. But in our models of physics, [energy has a rather simple \(and generic\) interpretation](#): it is effectively just the “density of activity” in the multicomputational system—or the number of events in a certain “region of space”.

Imagine that we pick a token in a multicomputational system. One question we can ask is: what is the shortest path through the token-event graph to get to some specific other token? There'll be a “light cone” that defines “how far in space” we can get in a certain time. But in general in physics terms we can view the shortest path as defining a spacetime geodesic. And now there's a crucial—but essentially structural—fact: the presence of activity in the token-event graph inevitably effectively “deflects” the geodesic.

And at least with the particular setup of our Physics Project it appears that that deflection can be described (in some appropriate limit) by Einstein's equations, or in other words, that

## Contents

---

assuming there is any similar kind of notion of space, or similar character of the observer—a phenomenon like gravity will also show up in other multicomputational systems.

Once we have gravity, what about phenomena like black holes? The [concept of an event horizon](#) is immediately something quite generic: it is just associated with disconnection in the causal graph, which can potentially occur in basically any multicomputational system.

What about a spacetime singularity? In the most familiar kind of singularity in physics (a “spacelike singularity” of the kind that appears at the center of a non-rotating black hole spacetime), what fundamentally happens is that there is a piece of the token-event graph to which no rules apply—so that in essence “time ends” there. And once again, we can expect that this will be a generic phenomenon in multicomputational systems.

But there’s more to say about this. In general relativity, the singularity theorems say that when there’s “enough energy or mass” it’s inevitable that a singularity will be formed. And we can expect that the same kind of thing will happen in any multicomputational system, though potentially it’ll be interpreted in very different terms. (By the way, the singularity theorems implicitly depend on assumptions about the observer and about what “states of the universe” they can prepare, and these may be different for other kinds of multicomputational systems.)

It’s worth mentioning that when it comes to singularities, there’s a computational characterization that may be more familiar than the physics one (not least since, after all, we don’t have direct experience of black holes). We can think of the progress of a multicomputational system through time as being like a [process of evaluation](#) in which rules are repeatedly applied to transform whatever “input” was given. In the most familiar case in physics, this process will just keep going forever. But in the more familiar case in practical computing, it will eventually reach a fixed point representing the “result of the computation”. And this fixed point is the direct analog of a “time ends” singularity in physics.

When we have a large multicomputational system we can expect that—like in physics—it will seem (at least to appropriate observers, etc.) like an approximate continuum of some kind. And then it’s essentially inevitable that there will be a whole collection of rather general “local” statements about the behavior that can be made. But what if we look at the multicomputational system as a whole? This is the analog of studying cosmology in physics.

conditions for the multicomputational system playing the role of the Big Bang in physics.

In the history of physics over the past century or so three great theoretical frameworks have emerged: statistical mechanics, general relativity and quantum mechanics. And when we look at multicomputational systems we can expect to get intuition—and results—from all three of these.

So what about quantum mechanics? As I mentioned above, quantum mechanics is—in our model of physics—basically just like general relativity, except played out not in ordinary physical space, but instead in branchial space. And in many ways, branchial space is a more immediate kind of space to appear in multicomputational systems than physical space. But unlike physical space, it's not something about which we have everyday experience, and instead to think about it we tend to have to rely on the somewhat elaborate traditional formalism of quantum mechanics.

A key question about branchial space both in physics and in other multicomputational systems is how it can be coordinatized (and, yes, that's inevitably a question about observers). In general the issue of how to put meaningful “numerical” coordinates on a very “non-numerical space” (where the “points of the space” are for example tokens corresponding to strings or hyperedges or whatever) is a difficult one. But the formalism of quantum mechanics makes for example the suggestion of thinking in terms of complex numbers and phases.

The spaces that arise in multicomputational systems can be very complicated, but it's rather typical that they can be thought of as somehow “curved”, so that, for example, “parallel” lines (i.e. geodesics) don't stay a fixed distance apart, and that “squares” drawn out of geodesics won't close. And in our model of physics, this kind of phenomenon not only yields gravity in physical space, but also yields things like the uncertainty principle when applied to branchial space.

We might at first have imagined that a theory of physics would be specific to physics. But as soon as we imagine that physics is multicomputational then that fact alone leads to a robust and inexorable structure that should appear in any other multicomputational system. It may be complicated to know quite what the detailed translations and interpretations are for other multicomputational systems. But we can expect that the core phenomena we've identified in physics will somehow be reflected there. So that through the common thread

sorts of systems in all sorts of fields.

## *Potential Application Areas*

metamathematics • chemistry • molecular biology  
evolutionary biology • neuroscience • immunology • linguistics  
economics • machine learning • distributed computing

We've talked about the nature of the multicomputational paradigm, and about its application in physics. But where else can it be applied? Already in the short time I've been thinking directly about this, I've identified a remarkable range of fields that seem to have great potential for the multicomputational paradigm, and where in fact it seems quite conceivable that by using the paradigm one might be able to successfully unlock what have been long-unresolved foundational problems.

Beginning a few decades ago, the [computational paradigm also helped shed new light](#) on the foundations of all sorts of fields. But typically its most important message has been: “Great and irreducible complexity arises here—and limits what we can expect to predict or describe”. But what's particularly exciting about the multicomputational paradigm is that it potentially delivers a quite different message. It says that even though the underlying behavior of a system may be mired in irreducible complexity, it's still the case that those aspects of the system perceived by an observer can show predictable and reducible behavior. Or, in other words, that at the level of what the observer perceives, the system will seem to follow definite and understandable laws.

But that's not all. As soon as one assumes that the observer in a multicomputational system is enough [“like us” to be computationally bounded and to “sequentialize time”](#) then it follows that the laws they will perceive will inevitably be some kind of translated version of the laws we've already identified in fundamental physics.

Physics has always been a standout field in its ability to deliver laws that have a rich (typically mathematical) structure that we can readily work with. But with the multicomputational paradigm there's now the remarkable possibility that this feature of physics could be transported to many other fields—and could deliver there what's in many cases been seen as a “holy grail” of finding “physics-like” laws.



Contents

---

“reduction” to an accurate model of the primitive parts of the system. But actually what the multicomputational paradigm indicates is that there’s a certain inexorability to what happens, independent of those details. The challenge, though, is to figure out what an “observer” of a certain kind of system will actually perceive. In other words, successfully finding overall laws isn’t so much about applying reductionism to the system; it’s more about understanding how observers fit together the details of the system to synthesize their perception of it.

So what kinds of systems can we expect to describe in multicomputational terms? Basically any kind of system where there are many component parts that somehow “operate independently in parallel”—interacting only through certain “events”. And the key idea is that there are many possible detailed histories for the system—but in the multicomputational paradigm we look at all of them together, thereby building up a structure with inexorable properties, at least as perceived by certain general kinds of observers.

In areas like statistical physics it’s been common for a century to think about “ensembles of possible states” for a system. But what’s different about the multicomputational paradigm is that it’s not just looking “statically” at “possible states”; instead it’s “taking a bigger gulp”, and looking at all possible whole histories for the system, essentially developing through time. And, yes, a slice at a particular time will show some ensemble of possible states—but they’re states generated by the entangled possible histories of the system, not just states “statically” and combinatorially generated from the possible configurations of the system.

OK, so what are some areas to which the multicomputational paradigm can potentially be applied? There are many. But among the examples I’ve at least begun to investigate are metamathematics, molecular biology, evolutionary biology, molecular computing, neuroscience, machine learning, immunology, linguistics, economics and distributed computing.

So how can one start developing a multicomputational model in a particular area? Ultimately one wants to see how the structure and behavior of the system can be broken down into elementary “tokens” and “events”. The network of events will define some way in which the histories of tokens are entangled, and in which the tokens are effectively “knitted together” to define something that in some limiting sense can be interpreted as some kind

## Contents

the things one identifies as tokens and events—and the emergent space may seem more familiar, as it does in the case of physical space in our model of physics.

OK, so what might the tokens and events be in particular areas? I'm not yet sure about most of these. But here are a few preliminary thoughts:

<i>field</i>	<i>tokens</i>	<i>events</i>
<b>metamathematics</b>	mathematical statements	application of laws of inference
<b>chemistry / molecular biology</b>	molecules of different types	reactions between molecules
<b>evolutionary biology</b>	individual organisms	interactions between organisms
<b>neuroscience</b>	states of individual neurons	activation of neurons
<b>immunology</b>	molecular shapes	molecular recognition
<b>linguistics</b>	occurrences of words, etc.	individual utterances
<b>economics</b>	configurations of agents	transactions between agents
<b>machine learning</b>	things to learn about	learning rules
<b>distributed computing</b>	states of processors	computation & communication

It's important to emphasize that the multicomputational paradigm is at its core not about particular histories (say particular interactions between organisms or particular words spoken) but about the evolution of all possible histories. And in most cases it won't have things to say about particular histories. But instead what it will describe is what an observer sampling the whole multicomputational process will perceive.

And in a sense the nub of the effort of using the multicomputational paradigm to find new laws in new fields is to identify just what it is that one should be looking at, or in effect what one would think an observer should do.

Imagine one's looking at the behavior of a gas. Underneath there's all sorts of irreducible complexity in the particular motions of the molecules. But if we consider the "correct" kind of observer, we'll just sample the gas at a level where they'll perceive overall laws like the diffusion equation or the gas laws. And in the case of a gas we're immediately led to that "correct" kind of observer, because it's what we get with our usual human sensory perception.

metamathematics or linguistics might be. And if we can figure that out, we'll potentially have overall laws—like diffusion or fluid dynamics—that apply in these quite different fields.

## Metamathematics

Let's start by talking about perhaps the most abstract potential application area: [metamathematics](#). The individual “tokens of mathematics” can be mathematical statements, written in some symbolic form (as they would be in the [Wolfram Language](#)). In a sense these mathematical statements are like the hyperedges of our spatial hypergraph in physics: they define relations between elements, which in the case of physics are “atoms of space” but in the case of mathematics are “literal mathematical objects”, like the number 1 or the operation **Plus** (or at least single instances of them).

Now we can imagine that the “state of mathematics” at some particular time in its development consists of a large number of mathematical statements. Like the hyperedges in the spatial hypergraph for physics, these mathematical statements are knitted together through their common elements (two mathematical statements might both refer to **Plus**, just as two hyperedges might both refer to a particular atom of space).

What is the “evolution of mathematics” like? Basically we imagine that there are laws of inference that take, say, two mathematical statements and deduce another one from them, either using something like structural substitution, or using some (symbolically defined) [logical principle like modus ponens](#). The result of [repeatedly applying a law of inference in all possible ways is to build up a multiway graph](#) of—essentially—what statements imply what other ones, or in other words, what can be proved from what.

But what does a human mathematician perceive of all this? Most mathematicians don't operate at the level of the raw proof graph and individual raw formalized mathematical statements. Instead, they aggregate together the statements and their relationships to form more “human-level” mathematical concepts.

In effect that aggregation can be thought of as choosing some “mathematical reference frame”—a slice of metamathematical space that can successfully be “parsed” by a human “mathematical observer”. No doubt there will be certain typical features of that reference frame; for example it might be set up so that things are “sufficiently organized” that

between categories” while preserving structure.

There are both familiar and unfamiliar features of this emerging picture. There are the analog of light cones in “proof space” that define dependencies between mathematical results. There are geodesics that correspond to shortest derivations. There are regions of “metamathematical space” (the slices of proof space) that might have higher “densities of proofs” corresponding to more interconnected fields of mathematics—or more “metamathematical energy”. And as part of the generic behavior of multicomputational systems we can expect an analog of Einstein’s equations, and we can expect that “proof geodesics” will be “gravitationally attracted” to regions of higher “metamathematical energy”.

In most areas of metamathematical space there will be “proof paths” that go on forever, reflecting the fact that there may be no path of bounded length that will reach a given statement, so that the question of whether that statement is present at all can be considered undecidable. But in the presence of large amounts of “metamathematical energy” there’ll effectively be a metamathematical black hole formed. And where there’s a “singularity in metamathematical space” there’ll be a whole collection of proof paths that just end—effectively corresponding to a decidable area of mathematics.

Mathematics is normally done at the level of “specific mathematical concepts” (like, say, algebraic equations or hyperbolic geometry)—that are effectively the “populated places” (or “populated reference frames”) of metamathematical space. But by having a multicomputational model of the low-level “machine code of metamathematics” there’s the potential to make more general statements—and to identify what amount to general “bulk laws of metamathematics” that apply at least to the “metamathematical reference frames” used by human “mathematical observers”.

What might these laws tell us? Perhaps they’ll say something about the homogeneity of metamathematical space and explain why the same structures seem to show up so often in different areas of mathematics. Perhaps they’ll say something about why the “aggregated” mathematical concepts we humans usually talk about can be connected without infinite paths—and thus why undecidability is so comparatively uncommon in mathematics as it is normally done.

understand more about the ultimate foundations of mathematics, and what mathematics “really is”. It might seem a bit arbitrary to have mathematics be constructed according to some particular law of inference. But in direct analogy to our Physics Project, we can also consider the “[rulial multiway system](#)” that allows all possible laws of inference. And as I’ve [argued elsewhere](#), the limiting object that we get for mathematics will be the same as for physics, connecting the question of why the universe exists to the “Platonic” question of whether mathematics “exists”.

## Chemistry / Molecular Biology

In mathematics, the tokens of our multicomputational model are abstract mathematical statements, and the “events between them” represent the application of laws of inference. In thinking about chemistry we can make a much more concrete multicomputational model: the tokens are actual individual molecules (represented say in terms of bonds) and the events are reactions between them. The token-event graph is then a “molecular-dynamics-level” representation of a chemical process.

But what would a more macroscopic observer make of this? One “chemical reference frame” might combine all molecules of a particular chemical species at a given time. The result would be a fairly traditional “chemical reaction network”. (The choice of time slices might reflect external conditions; the number of microscopic paths might reflect chemical concentrations.) Within the chemical reaction network a “[synthesis pathway](#)” is much like a [proof in mathematics](#): a path that leads from certain “inputs” in the network to a particular output. (And, yes, one can imagine “chemical undecidability” where it’s not clear if there’s any path of any length to make “this from that”.)

A chemical reaction network is much like a multiway graph of the kind we’ve shown for string substitutions. And just as in that case we can define a branchial graph that describes relationships between chemical species associated with their “entanglement” through participation in reactions—and from which a kind of “chemical space” emerges in which different chemical species appear at different positions.

There’s plenty to study at this “species level”. (As a simple example, small loops represent equilibria but larger ones can reflect the effect of protecting groups, or give signatures of



underlying token-event graph.

In standard chemistry, one typically characterizes the state of a chemical system at a particular time in terms of the concentrations of different chemical species. But ultimately there's much more information in the whole token-event graph—for example about the entangled histories of individual molecules and the causal relationships between events that produced them (which at a physical level might be manifest in things like correlations in orientations or momenta of molecules).

Does this matter, though? Perhaps not for chemistry as it's done today. But in thinking about molecular computing it may be crucial—and perhaps it's also necessary for understanding molecular biology. Processes in molecular biology today tend to be described—like chemical reactions—in terms of networks and concentrations of chemical species. (There are additional pieces having to do with the spatial structure of molecules and the possibility of “events at different places on a molecule”.) But maybe the whole “entanglement network” at the “token-event level” is also important in successfully capturing what amounts to the molecular-scale “chemical information processing” going on in molecular biology.

Just as in genetics in the 1950s there was a crucial realization that information could be stored not just, say, in concentrations of molecules, but in the structure of a single molecule, so perhaps now we need to consider that information can be stored—and processed—in a dynamic network of molecular interactions. And that in addition to seeing how things behave in “chemical species space”, one also needs to consider how they behave in branchial space. And in the end, maybe it just takes a different kind of “chemical observer” (and maybe one more embedded in the system and operating at a molecular scale) to be able to understand the “overall architecture” of many of the molecular computations that go on in biology.

(By the way, it's worth emphasizing that even though branchial space is what's associated with quantum mechanics in our model of fundamental physics we're not thinking about the “physical quantum mechanics” of molecules here. It's just that through the general structure of multicomputational models “quantum formalism” may end up being central to molecular computing and molecular biology even though—ironically enough—there doesn't have to be anything “physically quantum” about them.)

## Evolutionary Biology

---

What would it take to make a global theory of evolutionary biology? At a “local level” there’s natural selection. And there are plenty of “chemical-reaction-equation-like” (and even “reaction-diffusion-equation-like”) models for relations between the “concentrations” of small numbers of species. And, yes, there are global “developmental constraints”, that I for example have [studied quite extensively with the computational paradigm](#). But somehow the multicomputational paradigm seems to have the potential to make global “structural” statements about things like the whole “space of species” (and even why there are lots of species at all) just on the basis of the pure “combinatorial structure” of biological processes.

For example, one can imagine making a multicomputational model of “generalized evolutionary biology” in which the tokens are possible specific individual organisms, and the events are all their conceivable behaviors and interactions (e.g. two organisms mating in all possible ways to produce another). (An alternative approach would take the tokens to be genes.) The particular history of all life on Earth would correspond to sampling a particular path through this giant token-event graph of all possibilities. And in a sense the “fitness environment” would be encoded in the “reference frame” being used. The “biologist observer” might “coarse grain” the token-event graph by combining tokens that are considered to be the “same species”—potentially reducing the graph to some kind of phylogenetic tree.

But the overall question is whether—much like in fundamental physics—the underlying multicomputational structure (as sampled by some class of observers) might inexorably imply certain “general emergent laws of biological evolution”. One might imagine that the layout of organisms in “evolutionary space” at a particular time could be defined from a slice of a causal graph. And perhaps there is an analog of general relativity that exists when the “fitness environment reference frames” are “computationally tame enough” relative to the computational process of evolution. And maybe there are even analogs of the singularity theorems of general relativity, that would generically lead to the formation of event horizons—so that in some sense the distribution of species is like the distribution of black holes in a late-stage universe.

(There is a certain analogy with metamathematics here too: different organisms are like different mathematical statements, and finding a “paleontological connection” between them is like finding proofs in mathematics. Sometimes a particular evolutionary path might

representing an infinite future of “evolutionary innovations”.)

## Neuroscience

How do brains work? And how for example are “thoughts formed” out of the firings of lots of individual neurons? Maybe there’s an analog to how the coherent physical world we perceive is formed from the interactions of lots of individual atoms of space. And to explore this we might consider a multicomputational model of brains in which the tokens are individual neurons in particular states and events are possible interactions between them.

There’s a strange bit of circularity though. As I’ve [argued elsewhere](#), what’s key to deriving the perceived laws of physics is our particular way of parsing the world (which we may view as core to our [notion of consciousness](#))—in particular our concept that we have a single thread of experience, and thus “sequentialize time”. When applied to a multicomputational model of brains our core “brain-related” way of parsing the world suggests reference frames that again sequentialize time and turn all those parallel neuron firings into a sequence of coherent “thoughts”.

Just like in physics one can expect that there are many possible reference frames—and one might imagine that ultimate equivalence between them (which leads to relativity in physics) might lead to the ability of different brains to “think comparable thoughts”. Are there analogs of other physical phenomena? One might imagine that in addition to a main “thread of conscious thought” there might be alternate multiway paths whose presence would lead to “quantum-like effects” that would manifest as “influence of the unconscious” (making an analog of Planck’s constant a “measure of the importance of the unconscious”).

## Immunology

The immune system, like the brain, involves a large number of elements “doing different things”. In the brain, there are neurons in a definite physical arrangement that interact electrically. In the (adaptive) immune system there are things like white blood cells and antibodies that basically just “float around”, occasionally interacting though molecular-scale “shape-based” physical binding. It seems pretty natural to make a multicomputational model of this, in which individual immune system elements interact through all possible binding events. One can pick an “assay” reference frame in which one “coarse grains together”, say, all antibodies or all T-cell receptors that have a particular sequence.

approximately) a “summary graph” of interactions between types of antibodies, etc. Then much like we imagine physical space to be knitted together from atoms of space by their interactions, so also we can expect that the “shape space” of antibodies, etc. will also be defined by their interactions. Maybe “interactionally near” shapes will also be near in some simple sequence-based metric, but not necessarily. And for example there’ll be some analog of a light cone that governs any kind of “spreading of immunity” associated with an antigen “at a particular position in shape space”—and it’ll be defined by the causal graph of interactions between immune elements.

When it comes to understanding the “state of the immune system” we can expect—in a typical multicomputational way—that the whole dynamic network will be important. Indeed, perhaps for example “immune memory” is maintained as a “property of the network” even though individual immune elements are continually being created and destroyed—much as particles and objects in physics persist even though their constituent atoms of space are continually changing.

## Linguistics

Languages, like all the other kinds of things we’ve discussed, are fundamentally dynamic constructs. And to make a multicomputational model of them we can for example imagine every instance of every word (or even every conceivable word) being a token—with events being utterances that involve multiple words. The resulting token-event graph then defines relationships between instances of words (essentially by the “context of their usage”). And within any given time slice these relationships will imply a certain layout of word instances in what we can interpret as “meaning space”.

There’s absolutely no guarantee that “meaning space” will be anything like a manifold, and I expect that—like the emergent spaces from most token-event graphs we’ve generated—it’ll be considerably more complicated to “coordinatize”. Still, the expectation is that instances of a word with a given sense will appear nearby, as will synonymous words—while different senses of a given word will appear as separate clusters.

In this setup, the time evolution of everything would be based on there being a sequence of utterances that are effectively strung together by someone somehow hearing a given word in a certain utterance, then at some point later using that word in another utterance. What

nub of “defining the language”. As a rough approximation one could for example use some simple grammatical rule—in which case the possible events might themselves be determined by a multiway system. But the key point is that—like in physics—we may expect that there’ll be global laws quite independent of the “microscopic details” of what precise utterances are possible, just as a consequence of the whole multicomputational structure.

What might these “global laws” of language be like? Maybe they’ll tell us things about how languages evolve and “speciate”, with event horizons forming in the “meaning space of words”. Maybe they’ll tell us slightly smaller-scale things about the splitting and merging of different meanings for a single word. Maybe there’ll be an analog of gravity, in which the “geodesic” associated with the “etymological path” for a word will be “attracted” to some region of meaning space with large amounts of activity (or “energy”)—or in effect, if a concept is being “talked about a lot” then the meanings of words will tend to get closer to that.

By the way, picking a “reference frame for a language” is presumably about picking which utterances one’s effectively chosen to have heard by any given time, and thus which utterances one can use to build up one’s “sense of the meanings of words” at that time. And if the selection of utterances for the reference frame is sufficiently wild, then one won’t get a “coherent sense of meaning” for the language as a whole—making the “emergence of meaning” something that’s ultimately about what amounts to human choices.

## Economics

A basic way to imagine “microscopically” modeling an economic system is to have a token-event graph in which the tokens are something like “configurations of agents”, and the events are possible transactions between them. Much like the relations between atoms of space that are the tokens (represented by hyperedges) in our models of fundamental physics, the tokens we’re describing here as “configurations of agents” can more accurately be thought of as relations between elements that, say, represent economic agents, objects, goods, services, currency, etc. In a transaction we’re imagining that an “interaction” between such “relations” leads to new “relations”—say representing the act of exchanging something, making something, doing something, buying something, etc.



Contents

---

can imagine a setup (essentially a virtual token-event graph) where every conceivable transaction can in principle happen. The result will be a very complicated structure—though with certain inexorable features. But now consider how we would “observe” this system. Maybe there’d be a “from-the-outside” way to do this, but we could also just “be in the system” getting data through transactions that we’re involved in. But then we’re in a situation that’s pretty closely analogous to fundamental physics. And to make sense of what we observe, we’ll basically inevitably end up sampling the system by setting up some kind of reference frame.

But if this reference frame has typical “generalized human” characteristics such as computational boundedness it’ll end up weaving through all possible transactions to pick out slices that are “computationally simple to describe”. And this seems likely to be related to the origin of “value” in economics (or perhaps more so to the notion of a numéraire). Much like in physics, a reference frame can allow coordinates to be assigned. But the question is what reference frames will lead to coordinates that are somehow stable under the time evolution of the system. And in effect this is what general relativity tells us. And quite possibly there’s an analog of this in economic systems.

Why isn’t there just an immediate value for everything? In the model we’re discussing, all that’s defined is the network of transactions. But just seeing particular local transactions only tells us about things like “local value equivalences”. To say something more global requires the whole knitting together of “economic space” achieved by all the local transactions in the network. It’s very much like in the emergence of physical space. Underneath, there’s all sorts of complicated and computationally irreducible behavior. But if we look at the right things, we see computational reducibility, and something we can describe in the limit as continuum space. In economic systems, low-level transactions may show complicated and computationally irreducible behavior. But the point is that if we look at the right things, we again see something like continuum behavior, but now it corresponds to money and value. (And, yes, it is ironic that computational irreducibility is the basic phenomenon that seems to lead to a robust notion of value—even as it’s also what proof-of-work cryptocurrencies use to “mine” value.)

Like a changing metric etc. in spacetime, “value” can vary with position and time. And we can expect that there will be some general-relativity-like principles about how this works (perhaps with “curvature in economic space” allowing arbitrage etc.). There also might be

multiway graph. (In “quant finance”—which, yes, coincidentally sounds a bit like “quantum”—it’s for example common to estimate prices from looking at the effects of all possible paths, say approximated by Monte Carlo.)

At the outset, it’s not obvious that one can make any “economics-level” conclusion just based on thinking about what amount to arbitrary token-event graphs. But the remarkable thing about multicomputational models is that just from their general structure there are often inexorable quantitative laws that can be derived. And it’s conceivable that at least in the limit of a large economic system, it may finally be possible to do this.

### Machine Learning

One can think of **machine learning** as being about deducing models of what can happen in the world from collections of “training examples”. Often one imagines a collection of conceivable inputs (say, possible arrays of pixels corresponding to images), for which one wants to “learn a structure for the space”—in such a way that one can for example find a “**manifold-style**” “**coordinatization**” in terms of a **feature vector**.

How can one make a multicomputational model for this process? Imagine for example that one has a neural net with a certain architecture. The “state” of the net is defined by the values of a large number of weights. Then one can imagine a multiway graph in which each state can be updated according to a large number of different events. Each possible event might correspond to the incremental update of weights associated with back-propagating the effect of adding in a single new training example.

In present-day neural net training one normally follows a single path in which one applies a particular (perhaps randomly chosen) sequence of weight updates. But in principle there’s a whole multiway graph of possible training sequences. The branchial space associated with this graph in effect defines a space of possible models obtained after a certain amount of training—complete with a measure on models (derived from path weights), distances between models, etc.

But what about a token-event graph? Present-day neural nets—with their usual back-propagation methods—tend to show very little factorizability in the updating of weights. But if one could treat certain collections of weights as “independently updatable” then one

effects-in-the-net” space.

But if training is associated with the multiway (or token-event) graph, what is evaluation? One possible answer is that it’s basically associated with the reference frame that we pick for the net. Running the net might generate some collection of output numbers—but then we have to choose some way to organize these numbers to determine whether they mean, say, that an image is of a cat or a dog. And it is this choice that in effect corresponds to our “reference frame” for sampling the net.

What does this mean, for example, about what is learnable? Perhaps this is where the analog of Einstein’s equations comes in—defining the possible large-scale structure of the underlying space, and telling us what reference frames can be set up with computationally bounded effort?

## Distributed Computing

In the applications we’ve discussed so far, the multicomputational paradigm enters basically in a descriptive way, providing raw material from which models can be made. In distributed computing, the paradigm also plays a very powerful prescriptive role, suggesting new ways to do computations, and new kinds of computations to do.

One can think of traditional sequential computation as being based on a simple chain of “evaluation events”, with each event being essentially the evaluation of a function that transforms its input to produce output. The input and output that the functions deal with can involve many “parallel” elements (as, for example, in a cellular automaton) but there’s always just “one output” produced by a given function.

The multicomputational paradigm, however, suggests computations that involve not just chains of evaluation events, but more complicated graphs of them. And one way this can arise is through having “functions” (or “evaluation events”) that produce not just one but several “results” that can then independently be “consumed” by future evaluation events.

A feature of traditional sequential computations is that they’re immediately suitable for execution on a single processor that successively performs a chain of evaluations. But the multicomputational paradigm involves in a sense lots of “separate” evaluation events, that can potentially occur on a whole distributed collection of processors. There are definite

## total ordering.

Some events require as input the output from other events, and so have a definite relative ordering, making them—in physics terminology—“timelike separated”. Some events can be executed in parallel, essentially independently or “asynchronously” of each other—and so can be considered “spacelike separated”. And some events can be executed in different orders, but doing so can lead to different results—making these events (in our physics terminology) “branchlike separated”.

In practical distributed computing, there are usually great efforts made to avoid branchlike-separated events (or “race conditions”, as they’re commonly called). And if one can do this then one has a computation that—despite its distributed character—can still be interpreted in a fundamentally sequential way in time, with a succession of “definite results” being generated. And, yes, this is certainly a natural thing for us humans to try to do, because it’s what allows us to map the computation into our typical sequentialized “single thread of experience” that seems to be a fundamental feature of our basic notion of consciousness.

But what the multicomputational paradigm suggests is that this isn’t the only way to set up distributed computing. Instead, we just want to think about the progress of a computation in terms of the “bulk perception” of some observer. The observer may be able to pick many different reference frames, but each will represent some computation. Sometimes this computation will correspond to a distributed version of something we were already familiar with. But sometimes it’ll effectively be a new kind of computation.

It’s common to talk about nondeterministic computation in which many paths can be followed—but ultimately one picks out one particular path (say, one that successfully satisfies some condition one’s looking for). The multicomputational paradigm is about the rather different idea of actually treating the “answer” as corresponding to a whole bundle of paths that are combined or conflated through a choice of reference frame. And, yes, this type of thing is rather alien to our traditional “single-thread-of-time” experience of computing. But the point is that particularly through its use and interpretation in physics and so many other areas the multicomputational paradigm gives us a general way to think about—and harness—such things.

And potentially gives us a very different—and powerful—new approach to distributed computing, perhaps complete with very general physics-like “bulk” laws.

## Contents

---

OK, so what about other areas? There are quite a few more that I've thought at least a little about. Among them are ones like history, psychology and the general development of knowledge. And, yes, it might seem quite surprising that there could be anything scientific or systematic to say about such areas. But the remarkable thing is that by having a new paradigm for theoretical science—the multicomputational paradigm—it becomes conceivable to start bringing science to areas it's never been able to touch before.

But even in what I've said above, I've only just begun to sketch how the multicomputational paradigm might apply in different areas. In each case there's years of work to do in developing and refining things. But I think there's amazing promise in expanding the domain of theoretical science, and potentially bringing physics-like laws to fields that have long sought such things, but never been able to find them.

### *Some Backstory*

What's the backstory of what I'm calling the multicomputational paradigm? For me, the realization that one can formulate such a broad and general new paradigm is something that's emerged only over the past year or so. But given what we now know it's possible to go back and see a quite tangled web of indications and precursors of it stretching back decades and perhaps more than a century.

A key technical step in the development of the multicomputational paradigm was the idea of what I named “[multiway systems](#)”. I first used the term “multiway systems” in 1992 when I included a placeholder for a future section about them in an early draft of what would become my 2002 book *A New Kind of Science*.

A major theme of my work during the development of *A New Kind of Science* was exploring as broadly as possible the computational universe of simple programs. I had already [in the 1980s extensively studied cellular automata](#)—and discovered all sorts of interesting phenomena like computational irreducibility in them. But now I wanted to see what happened in other parts of the computational universe. So I started investigating—and inventing—different kinds of systems with different underlying structures.

And there, in [Chapter 5](#), sandwiched between a section on [Network Systems](#) (one of the precursors of the [discrete model of space](#) in our [Physics Project](#)) and [Systems Based on Constraints](#), is a section entitled [Multiway Systems](#). The basic thrust is already the core

dimensional arrangement of states in time”:

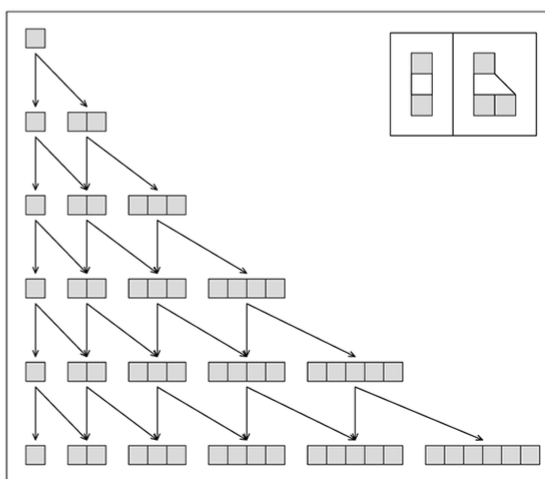
## Multiway Systems

The network systems that we discussed in the previous section do not have any underlying grid of elements in space. But they still in a sense have a simple one-dimensional arrangement of states in time. And in fact, all the systems that we have considered so far in this book can be thought of as having the same simple structure in time. For all of them are ultimately set up just to evolve progressively from one state to the next.

Multiway systems, however, are defined so that they can have not just a single state, but a whole collection of possible states at any given step.

The picture below shows a very simple example of such a system.

A very simple multiway system in which one element in each sequence is replaced at each step by either one or two elements. The main feature of multiway systems is that all the distinct sequences that result are kept.



I studied multiway systems as abstract systems. Later in the book I studied them as [idealizations of mathematical proofs](#). And I [mentioned them as a possible](#) (but as I thought then, rather unsatisfactory) underpinning for quantum mechanics.

I came back to multiway systems quite a few times over the years. But it was only when [we started the Wolfram Physics Project](#) in the latter half of 2019 that it began to become clear (particularly through an insight of [Jonathan Gorard's](#)) just how central multiway systems would end up being to fundamental physics.

The basic idea of multiway systems is in a sense so straightforward and seemingly obvious that one might assume it had arisen many times. And [in a sense it has](#), at least in special



names—though realistically in many cases we can only see the “multiway system character” when we look back from what we now know.

The rather trivial barely-a-real-multiway-system case of pure trees no doubt arose for example with the construction of family trees, presumably already in antiquity. Another special and rather trivial case arose in the construction of [Pascal’s triangle](#), which one can think of as working just like the “very simple multiway system” shown above. (In Pascal’s triangle, of course, the emphasis is not on the pattern of states, but on the path weights, which are binomial coefficients.)

Another very early—if implicit—use of multiway systems was in recreational mathematics puzzles and games. Perhaps in antiquity, and certainly by 800 AD, there was for example the wolf-goat-cabbage river crossing problem, whose possible histories form a multiway system. Mechanical puzzles like Chinese rings are perhaps even older. And games like [Mancala](#) and [Three Men’s Morris](#) might date from early antiquity—even though the explicit “mathematical” concept of “[game trees](#)” (which are typically multiway graphs that include merging) seems to have only arisen (stimulated by discussions of chess strategies) as an “application of set theory” by [Ernst Zermelo](#) in 1912.

In mathematics, multiway systems seem to have first appeared—again somewhat implicitly—in [connection with groups](#). Given words in a group, written out as sequences of generators, successive application of relations in the group essentially yield multiway string substitution systems—and in 1878 [Arthur Cayley](#) drew a kind of dual of this to give what’s now called a [Cayley graph](#).

But the first more-readily-recognizable examples of multiway systems seem to have appeared in the early 1900s in connection with efforts to find [minimal representations of axiomatic mathematics](#). The basic idea—which arose several times—was to think of the process of mathematical deduction as consisting of the progressive transformation of something like sequences of symbols according to certain rules. (And, yes, this basic idea is also what the [Wolfram Language](#) now uses for representing general computational processes.) The notion was that any given deduction (or proof) would correspond to a particular sequence of transformations. But if one looks at all possible sequences what [one has is a multiway system](#).

## Contents

paper entitled “Problems Concerning the Transformation of Symbol Sequences According to Given Rules”) what are essentially string equivalences (two-way string transformations) and discussed what paths could exist between strings—with the result that string substitution systems are now sometimes called “semi-Thue systems”. In 1921 [Emil Post](#) then considered (one-way) string transformations (or, as he called them, “productions”). But, like Thue, he focused on the “decision problem” of whether one string was reachable from another—and never seems to have considered the overall multiway structure.

Thue and Post (and later [Markov](#) with his so-called “normal algorithms”) considered strings. In 1920 [Moses Schönfinkel](#) introduced his [S, K combinators](#) and defined transformations between what amount to symbolic expressions. And here again it turns out there can be ambiguity in how the transformations are applied, leading to what we’d now consider a multiway system. And the same issue arose for [Alonzo Church’s lambda calculus](#) (introduced around 1930). But in 1936 Church and Rosser showed that at least for lambda calculus and combinators the multiway structure basically doesn’t matter so long as the transformations terminate: the final result is always the same. (Our “causal invariance” is a more general version of this kind of property.)

Even in antiquity the idea seems to have existed that sentences in languages were constructed from words according to certain grammatical rules, that could (in modern terms) perhaps be recursively applied. For a long time this wasn’t particularly formalized (except conceivably for Sanskrit). But finally in 1956—following work on string substitution systems—there emerged the concept of generative grammars. And while the focus was on things like what sentences could be generated, the underlying representation of the process of generating all possible sentences from grammatical rules can again be thought of as a multiway system.

In a related but somewhat different direction, the development of various kinds of (often switch-like) devices with discrete states had led by the 1940s to the fairly formal notion of a [finite automaton](#). In many engineering setups one wants just one “deterministic” path to be followed between the states of the automaton. But by 1959 there was explicit discussion of [nondeterministic finite automata](#), in which many paths could be followed. But while in principle tracing all these paths would have yielded a multiway system it was quickly discovered that as far as the set of possible strings generated or recognized was concerned, there was always an equivalent deterministic finite automaton.

nondeterminism”—in which there are multiple outcomes possible at a sequence of steps. And over the course of the past century or so quite a few different kinds of systems based on repeated nondeterminism have arisen—a simple example being a random walk, investigated since the late 1800s. Usually in studying systems like this, however, one’s either interested only in single “random instances”, or in some kind of “overall probability distribution”—and not the more detailed “map of possible histories” defined by a multiway system.

Multiway systems are in a sense specifically about the structure of “progressive evolution” (normally in time). But given what amount to rules for a multiway system one can also ask essentially combinatorial questions about what the distribution of all possible states is. And one place where this has been done for over a century is in estimating equilibrium properties of systems in statistical physics—as summarized by the so-called [partition function](#). Even after all these years there is, however, much less development of any general formalism for non-equilibrium statistical mechanics—though some diagrammatic techniques are perhaps reminiscent of multiway systems (and our full multicomputational approach might now make great progress).

Yet another place where multiway systems have implicitly appeared is in studying systems where asynchronous events occur. The systems can be based on Boolean algebra, database updating or other kinds of ultimately computational rules. And in making proofs about whether systems are for example “correct” or “safe” one needs to consider all possible sequences of asynchronous updates. Normally this is done using various optimized implicit methods, but ultimately there’s always effectively a multiway system underneath.

One class of models that’s been rediscovered multiple times since 1939—particularly in various systems engineering contexts—are so-called [Petri nets](#). Basically these generalize finite automata by defining rules for multiple “markers” to move around a graph—and once again if one were to make the trace of all possible histories it would be a multiway system, so that for example “reachability” questions amount to path finding. (Note that—as we saw earlier—a token-event graph can be “rolled up” into what amounts to a Petri net by completely deduplicating all instances of equivalent tokens.)

In the development of computer science, particularly beginning in the 1970s, there were various investigations of parallel and distributed computer systems where different

Contents

---

Concepts like channels, message-passing and coroutines were developed—and formal models like [Communicating Sequential Processes](#) were constructed. Once again the set of possible histories can be thought of as a multiway system, and methods like [process algebra](#) in effect provide a formalism for describing certain aspects of what can happen.

I know of a few perhaps-closer approaches to our conception of multiway systems. One is so-called [Böhm trees](#). In studying “term rewriting” systems like [combinators](#), lambda calculus and their generalization the initial focus was on sequences of transformations that [terminate in some kind of “answer”](#) (or “normal form”). But starting in the late 1970s a small amount of work was done on the [non-terminating case](#) and on what we would [now call multiway graphs](#) describing it.

We usually think of multiway graphs as being generated progressively by following certain rules. But if we just look at the final graphs, they (often) have the mathematical structure of [Hasse diagrams for partial orderings](#). Most posets that get constructed in mathematics don’t have particularly convenient interpretations in terms of interesting multiway systems (especially when the posets are finite). But for example the “partially ordered set of finite causets” (or “poscau”) generated by all possible sequential growth paths for causal sets (and studied since about 2000) can be thought of as a multiway system.

Beginning around the 1960s, the general idea of studying systems based on repeated rewriting—of strings or other “terms”—developed essentially into its own field, though with connections to formal language theory, automated theorem proving, combinatorics on words and other areas. For the most part what was studied were questions of termination, decidability and various kinds of path equivalence—but apparently not what we would now consider “full multiway structure”.

Already in the late 1960s there started to be discussion of [rewriting systems \(and grammars\) based on graphs](#). But unlike for strings and trees, defining how rewriting might structurally work was already complicated for graphs (leading to concepts like double-pushout rewriting). And in systematically organizing this structure (as well as those for other diagrammatic calculi), connections were made with category theory—and this in turn led to connections with formalizations of distributed computing such as process algebra and  $\pi$  calculus. The concept that rewritings can occur “in parallel” led to use of monoidal

abstract) perspective on what we now call multiway systems.

(It might be worth mentioning that I studied graph rewriting in the 1990s in connection with possible models of fundamental physics, but was somewhat unhappy with its structural complexity—which is what led me finally in 2018 to start studying hypergraph rewriting, and to develop the foundations of our Physics Project. Back in the 1990s I did consider the possibility of multiway graph rewriting—but it took the whole development of our Physics Project for me to understand its potential significance.)

An important feature of the multicomputational paradigm is the role of the observer and the concept of sampling multiway systems, for example in “slices” corresponding to certain “reference frames”. And here again there are historical precursors.

The term “reference frame” was introduced in the 1800s to organize ideas about characterizing motion—and was generalized by the introduction of special relativity in 1905 and further by general relativity in 1915. And when we talk about slices of multiway systems they’re structurally very much like discrete analogs of sequences of spacelike hypersurfaces in continuum spacetime in relativity. There’s a difference of interpretation, though: in relativity one’s dealing specifically with physical space, whereas in our multiway systems we’re dealing first and foremost with branchial space.

But beyond the specifics of relativity and spacetime, starting in the 1940s there emerged in mathematics—especially in dynamical systems theory—the general notion of foliations, which are basically the continuous analogs of our “slices” in multiway systems.

We can think of slices of multiway systems as defining a certain ordering in which we (or an observer) “scans” the multiway system. In mathematics starting about a century ago there were various situations in which different scanning orders for discrete sets were considered—notably in connection with diagonalization arguments, space-filling curves, etc. But the notion of scanning orders became more prominent through the development of practical algorithms for computers.

The basic point is that any nontrivial recursive algorithm has multiple possible branches for recursion (i.e. it defines a multiway system), and in running the algorithm one has to decide in what order to follow these. A typical example involves scanning a tree, and deciding in what order to visit the nodes. One possibility is to go “depth first”, visiting nodes all the way

mazes before 1900. But by the end of the 1950s, in the course of actual computer implementations, it was noticed that one could also go “[breadth first](#)”.

Algorithms for things like searching are an important use case for different scanning orders (or in our way of describing things, different reference frames). But another use case intimately tied into things like term rewriting is for [evaluation orders](#). And indeed—though I didn’t recognize it at the time—my own work on evaluation orders in symbolic computation around 1980 is quite related to what we’re now doing with multicomputation. (Evaluation orders are also related to lazy evaluation and to recent ideas like [CRDTs](#).)

The most typical “workflow” for a computation is a direct one (corresponding to what I call here the computational paradigm): start with an “input” and progressively operate on it to generate “output”. But another “workflow” is effectively to define some goal, and then to try to find a path that achieves it. Early examples around [automated theorem proving](#) were already implemented in the 1950s. By the 1970s the approach was used in practice in logic programming, and was also at a theoretical level formalized in [nondeterministic Turing machines](#) and [NP problems](#). At an underlying level, the setup was “very multiway”. But the focus was almost always just on finding particular “winning” paths—and not on [looking at the whole multiway structure](#). Slight exceptions from the 1980s were various studies of “distributions of difficulty” in NP problems, and [early considerations of “quantum Turing machines”](#) in which superpositions of possible paths were considered.

But as so often happens in the history of theoretical science, it was only through the development of a new conceptual framework—around our Physics Project—that the whole structure of multiway systems was able to emerge, complete with ideas like causal graphs, branchial space, etc.

Beyond the formal structure of multiway systems, etc., another important aspect of the multicomputational paradigm is the central role of the observer. And in a sense it might seem antithetical to “objective” theoretical science to even have to discuss the observer. But the development of relativity and quantum mechanics (as well as statistical mechanics and the concept of entropy) in the early 1900s was predicated on being “more realistic” about observers. And indeed what we now see is that the role of observers in these theories is deeply connected to their fundamentally multicomputational character.



## Contents

---

arguably for centuries, particularly in the philosophy of physics. But for the most part there hasn't been much intersection with computational ideas—with exceptions including [Heinz von Foerster's](#) “Second-Order Cybernetics” from the late 1960s, [John Wheeler's](#) “It from Bit” ideas, and to some extent [my own investigations about the origins of perceived complexity](#).

In some respects it might seem surprising that there's such a long and tangled backstory of “almost sightings” of multiway systems and the multicomputational paradigm. But in a sense this is just a sign of the fact that the multicomputational paradigm really is a new paradigm: it's something that requires a new conceptual framework, without which it really can't be grasped. And it's a tribute to how fundamental the multicomputational paradigm is that there have been so many “shadows” of it over the years.

But now that the paradigm is “out” I'm excited to see what it leads to—and I fully expect this new fourth paradigm for theoretical science to be at least as important and productive as the three we already have.

## *Thanks*

Thanks to Xerxes Arsiwalla, Hatem Elshatlawy, Jonathan Gorard, Nik Murzin, Max Piskunov and Christopher Wolfram for discussions and help with the general paradigm, and Etienne Bernard, Taliesin Beynon, James Boyd, Elise Cawley, Greg Chaitin, Jean du Plessis, Roger Germundsson, Fred Meinberg, Bob Nachbar, Terry Sejnowski and others for discussions about specific potential applications and history. Thanks also to the attendees of the [2021 Wolfram Summer School](#) for stimulating me to formulate more generally the whole concept of multicomputation.

### TOOLS:

Some software tools for working in the multicomputational paradigm are already [available from the Wolfram Physics Project](#). Going forward, we'll be [working on many additional tools](#).

---

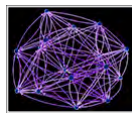
Posted in: [Big Picture](#), [Computational Science](#), [Computational Thinking](#), [Future Perspectives](#), [Life Science](#), [Mathematics](#), [New Kind of Science](#), [Philosophy](#), [Physics](#), [Ruliology](#)

---

*Join the discussion*

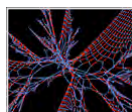
## Contents

## Related Writings



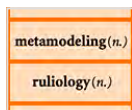
## The Concept of the Ruliad

November 10, 2021



## Multicomputation with Numbers: The Case of Simple Multiway Systems

October 7, 2021



## Charting a Course for “Complexity”: Metamodeling, Ruliology and More

September 23, 2021



## Even beyond Physics: Introducing Multicomputation as a Fourth General Paradigm for Theoretical Science

September 9, 2021

## Popular Categories

Artificial Intelligence

Big Picture

Companies and Business

Computational Science

Computational Thinking

Data Science

Education

Future Perspectives

Historical Perspectives

Language and Communication

Life and Times

Life Science

Mathematica

Mathematics

New Kind of Science

New Technology

Personal Analytics

Philosophy

Physics

Ruliology

Software Design

Wolfram|Alpha

Wolfram|One

Wolfram Language

Other

Contents

---

[2021](#) | [2020](#) | [2019](#) | [2018](#) | [2017](#) | [2016](#) | [2015](#) | [2014](#) | [2013](#) | [2012](#) | [2011](#) | [2010](#) |  
[2009](#) | [2008](#) | [2007](#) | [2006](#) | [2004](#) | [2003](#) | [All](#)

© Stephen Wolfram, LLC | [Terms](#) | [RSS](#)