International Journal of Theoretical Physics, Vol. 23, No. 10, 1984

Nondeterminism Can Be Causal

Michael J. Manthey

Computer Science Department, The University of New Mexico, Albuquerque, New Mexico 87131

Received January 13, 1984

It is shown how a new type of model, based in the abstract properties of computational processes, provides both a mechanism and a novel *causal* explanation for the nondeterminism observed in particle interactions. Related work shows that hidden variable theories are at the same time unacceptable.

1. INTRODUCTION

Despite the astounding success of quantum mechanics in predicting actual experimental outcomes, there are grounds for unease:

a. There is no unified theory of this level of reality.

b. There are mathematical problems, e.g., singularities and divergences, which can only sometimes be avoided by mathematical manipulation. The "reality" of the description is tenuous.

c. We cannot *explain* what is "going on" except in terms of the mathematical formalism. (I am here distinguishing between "understanding" and "mastery of the formalism".) If one cannot explain something in common language, one does *not* understand it.

d. Paradoxes (affronts to "common sense") abound. I do not accept that the quantum world's behavior is by nature counterintuitive and totally unlike our everyday macroscopic world.

I believe that items (c) and (d) are in large part traceable to an outmoded fashion which treats causality and nondeterminism as incompatible concepts; this paper proposes a novel computer-based *mechanism* which resolves this incompatibility. It is this basis in the abstract properties of computations that distinguishes our approach from the more common statistical ones (e.g., Mehlberg, 1980; Everett, 1957; Ballantine, 1970).[1] It is

[1] The Appendix compares these approaches briefly.

0020-7748/84/1000-0929$03.50/0

my hope that by getting this part right, progress could be made on (a) and (b). For example, Manthey and Moret (1983) present some basic results in what might be called information dynamics, as well as a new type of abstraction hierarchy which naturally excludes hidden variable theories (Rohrlich, 1983); the latter is discussed at greater length in Manthey (unpublished). Even if such a computational approach does not affect the formalism in its daily application—and why should one seek to supplant such a successful and highly developed tool?—it makes it easier to teach and disseminate the theories and concepts to a wider audience.

It is appropriate at this point to discuss the relevance of computer science to physics for purposes other than grinding numbers. The title of "computer scientist" does not mean some sort of superprogrammer. Over the last two decades, a large body of theoretical material with considerable intrinsic interest has been developed; after all, many of the original faculty of computer science departments were mathematicians. Beckman (1981) gives one possible overview of some of this material. Computer scientists have been studying and manipulating the concept of a "process" for some time. Given that we neither have to rely on Nature to supply sample processes for study nor are bound to what is "natural," we have had an unprecedented opportunity to study and manipulate them in the abstract. The material presented in this paper thus builds on several decades of such research.

The developments presented here are based on the idea of a *computational process* used in the study of concurrent computational systems, which are systems containing >1 interacting processes. For this reason we call this proposal "the computational metaphor." The essential idea is that what appears to observers as "random" behavior is a property of the time frames-of-reference of the observer and other processes in the system under study. Thus from the frame of reference of one process, the "time of arrival" of another process is not commensurate, and hence can only be considered random. This results, as will be shown, in an indeterminate future for at least one of the two interacting processes.

Before this argument, which is extremely simple in its essence, can be presented more precisely, we find it necessary to introduce some computational definitions (Section 2), out of which we build the standard particle physicist's definitions of such things as events and processes (Section 3). Finally in Section 4 we present the above argument.

The principal deviations of the computational metaphor from the usual models of quantum mechanics are as follows:

1. An actual *mechanism* is proposed; and therefore

2. Causality is not identified with determinism, nor is lack of causality identified with nondeterminism. Rather, our definition of causality applies to both outcome regimes.

3. On the basis of the two preceding points, the computational metaphor is "realistic" rather than "idealistic." The common Copenhagen interpretation is idealistic, in that it attributes *no* reality to any process which is not actually being "observed." That is, the idealist interpretation holds that one cannot speak of a system's being in *any* state between consecutive observations; said states obtain *only* when the wave function describing the possible outcomes collapses in the act of measurement. We do however agree that what one sees depends heavily on how one observes.

Mehlberg (1980) arrives at much the same conclusions with a comprehensive and careful analysis, although he proposes no mechanism.

2. COMPUTATIONAL DEFINITIONS

The computational metaphor is stated in terms of Hewitt's actor model of computation (Hewitt and Baker, 1977, 1978; Baker, 1978; MacQueen, 1979), which is an operational model. We prefer this model precisely because it is operational, and therefore *a priori* close to the models found in quantum mechanics. Algebraic computational models of concurrency (e.g., Peterson, 1977; Milner, 1978; MacQueen, 1979), in contrast, are not amenable to the actual calculation of numerical results.

Definition. An *actor* is an active entity, analogous to a von-Neumann-type sequential deterministic computer. An actor may have any given arbitrary but fixed behavior (e.g., specified by a program).

Definition A *message* is a finite string of bits of nonzero length.

Actors receive messages serially and undergo internal state transformations as they route messages to other actors, transform message content, and the like.

Definition. An *arc* is a directed communication link between two actors with the property that a message placed on an arc always arrives intact at the other end. The arrival order of consecutive messages sent by one actor to another is not necessarily the transmission order, even over a single arc.

We assume here that any actor is directly connected (i.e. with a single arc) to only a finite number of other actors.

Definition. A *system* is a directed graph whose nodes are actors and whose communication links are arcs.

Definition. A *c event* (computational event) is the receipt of a message by an actor.

We assume that if an arc exists, then the c event associated with its (arrow-)head *can* occur. We are after all interested in all possible c events. If an arc is in fact untraversable, it plays no role whatsoever, and so can be eliminated. We speak of "c events" rather than "events" because the usual particle physicist's definition of this term is quite different (see "p event" below).

Definition. A *c process* is any sequence of c events formed by a walk of a system graph. (As in the preceding paragraph, we assume that a walk represents a path a message *could* follow.)

The *state* of a c process at some instant is the message content at that instant.[2]

Definition. One c event e_1 at actor a_1 is the *direct cause* of another c event e_2 at actor a_2 if a_1 and a_2 are directly connected (i.e., by a single arc.) Two c events are *causally related* if there is a c process that includes both of them. [Note that being causally related does not necessarily imply a deterministic relationship; this should become clearer later.]

We now define a set of primitive actors with the properties that (1) each examines or changes at most one bit of a message, (2) no primitive actor can be expressed in terms of the others, (3) all effective computations can be modeled (i.e., the set has power equivalent to a Turing machine), and (4) each primitive processes arriving messages serially. [Another way of stating (4) is that an actor is either busy processing a single message, or idle, awaiting a message.]

The primitive actors are:

Source—an actor which receives no messages, and produces messages acausally. One can postulate a source which produces messages of any given (effectively computable) content, and even in any particular order, but never with any statement of timing (e.g., interarrival time). See Figure 1a.

Sink—an actor which consumes all arriving messages and produces none. See Figure 1b.

Decider—an actor which routes messages arriving on its single in-leg to one of two out-legs (labeled 0 and 1) based on the value of the first bit of the message. The message is passed unchanged. See Figure 1c.

Arbiter—an actor which routes messages arriving on either of its two in-legs (labeled 0 and (1) to its single outleg. In the case where a message arrived at each in-leg simultaneously (from the time frame of the arbiter),

[2]This use of the word "state" should be distinguished from that of Ballantine (1970), which refers to ensembles. See also the Conclusion.

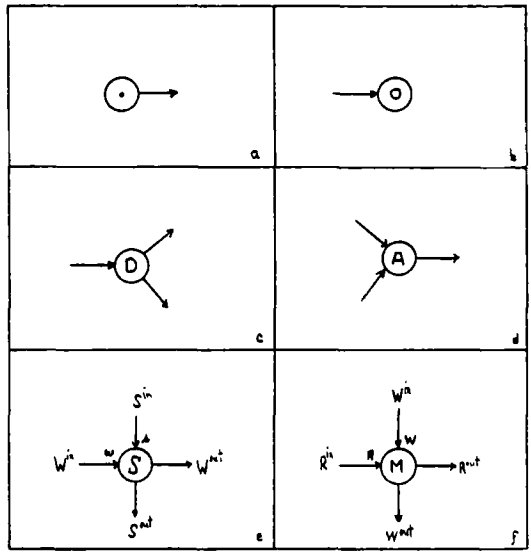

Fig. 1. The six one-bit primitive actors: (a) source, (b) sink, (c) decider, (d) arbiter, (e) synchronizer, (f) memory.

then the order of the two messages on the out-leg is unpredictable. The message(s) is (are) left unchanged. See Figure 1d.

These four primitives are said to be *pure* because their behavior is independent of their activation history, or alternatively, that their only persistent state is busy/idle.

Synchronizer—an actor with two in-legs [labeled W (wait) and S (signal)] and two out-legs, each associated exclusively with its corresponding in-leg. Messages arriving on the W-inleg are passed through one for one with messages arriving on the S-inleg, i.e., no waiting message may pass through without a corresponding signal message. Both types of messages are otherwise unaffected by passing through a synchronizer. See Figure 1e.

Memory—an actor with two in-legs [labeled W (write) and R (read)] and two out-legs, each associated exclusively with its corresponding in-leg. The front bit of a message arriving on the W-inleg is removed and becomes a part of the persistent internal state of the memory actor. A message arriving on the R-inleg has appended to its front the bit (if any) acquired by the memory actor from the previous write message, whereafter the persistent internal state of the memory actor is now "#" (no value, as opposed to 1 or 0). A Write attempt on a memory actor which already has a bit (1 or 0) results in that bit being "kicked out" the Read out-leg, after which the Write proceeds as described earlier. See Figure 1f.

Both of these actors are *impure* since their behavior *is* affected by their activation history, although the delays introduced by synchronizer a cannot be observed in principle.

Several general comments are in order. First, there is nothing unique about the particular set of primitives presented above. The most likely variation is in the definition of the memory primitive, which conserves bits strictly in our version; we have chosen a conservative form for the sake of simplicity in our subsequent analyses. The only nonconservation is at sources and sinks.

Second, and more important, is that actors and their states cannot be observed directly, but only by their effects on messages. Since the bits in a message are the current state of some process, this says that *one can actually only observe* (*c*) *processes*, not actors or *c* events. Indeed, in this lies the germ of the comptutational version of the uncertainty principle: In learning the state of a *c* process, by "stealing" a bit or two from it, one inevitably disturbs its state.

3. PARTICLE PHYSICISTS' DEFINITIONS REVISITED

With the definitions above, the *c* process concept is at a higher level of abstraction than an actor. To be precise, the *c*-process concept is built by functionally composing an actor-level concept (*c* events). A particularly important consequence of this is that while actors communicate via messages, *c* processes can communicate *only* via shared memory. In the minimal case, *c* process P_1 Writes a bit to memory actor M which is subsequently Read by *c* process P_2. Such an interaction corresponds to the physical notion of a collision of two particles, *which in turn is what physicists call an event*. Thus, what we have called a *c* process is what physicists call a particle.

Definition. A *p event* (particle physicist's event) is the bit exchange via a common memory actor by two *c* processes, one a Reader and the other a Writer.

Given this definition, we can now define the sole "undefined concept" of Mehlberg's axiomatization of quantum mechanics in terms of our lower level concepts. As mentioned earlier, this axiomatization does not deny causality. It is tempting to conclude that since the computational metaphor enters below the level of Mehlberg's axioms and definitions, it (the metaphor) includes quantum mechanics automatically, but we are not (yet) claiming such a sweeping conclusion.

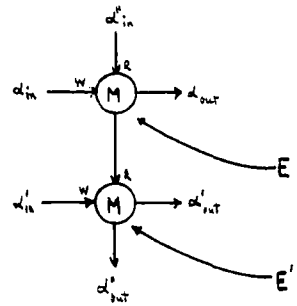

Fig. 2. Collision connectibility. (The W/R labels on the memories are arbitrary.)

Definition (after Mehlberg). *Collision connectibility* is the dyadic relation between any two p events E and E′ which obtains whenever particle α'', distinct from particles α and α' in which the events E and E′ occur, respectively, collides at different times with the particles α and α'. [With the understanding that E (or E′) occurs on α (or α') when α (or α') collides with α''.]

In terms of our primitive actors, Figure 2 illustrates the definition of collision connectibility.

Definition. A *p process* is an ordered sequence of collision connectible p events.

We can now demonstrate a non-determinism similar to that of quantum physics. In order to do this, we must back up briefly and consider the nature of time at the level of c processes.

4. TIME AND RANDOMNESS

Axiom. Time in a c process is discrete and marked in instants. These instants correspond 1-1 with successive changes in the state of the c process (= changes in message content, which by definition occur only in memory actors).[3]

In Figure 3, consider c process P_1 approaching memory actor M on its Read-leg, and c process P_2 approaching on M's Write-leg. If P_1 reaches M

[3]See Manthey and Moret (1983) for a slightly different primitive set, and corresponding slight difference in this definition, viz. instants 1-1 with all c events except synchronizers.

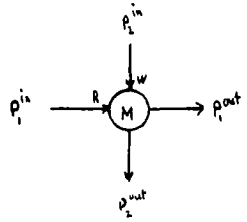

Fig. 3. Processes P_1 and P_2 are asynchronous, and the outcome of their interaction is nondeterministic.

before P_2, then it will get whatever bit is there (if any at all). If P_2 arrives first, then in general the bit in M will be different, and P_1's future correspondingly different. From P_1's local time frame, no statement about P_2's arrival "time" can be made, and vice versa. So if for example P_1 is our observer process, the outcome of the observation (i.e., the state of P_1), while consisting of different discrete possibilities ([M = #, M ≠ #]×[P_1 first, P_2 first]), will within these possibilities *appear* random. This phenomenon could be termed "the randomness born of asynchronous communication." In other words, owing to the incommensurability of the local time frames of the two c processes, the arrival time of one with respect to the other is in principle unknowable! Since one of these c processes is the observer process, its (P_1's) future of will be unpredictable, *although at no point is the definition of causality violated.*

Thus, even though P_1 and P_2 might individually be deterministic, the outcome of their interaction is not. More precisely, P_2's (the Writer's) behavior is (locally, at least) deterministic—it *always* writes its first bit, whereas shown earlier, P_1's future is always *non*deterministic. (Note however that in many cases, P_2's state is a function of the entire system, and in any event is unknowable owing to the computational metaphor's version of the uncertainty principle.) Notice finally the intrinsic noncommutativity of the operations Read and Write.

The above example with just two c processes (P_1 and P_2), where the Reader is the "observer" and the Writer the "observed particle," is somewhat like shooting fish at night in a big lake. Experimentalists prefer to shoot their fish in barrels, which looks like Figure 4. Here, P_1 is the "particle" of interest, P_2 is the experimental probing process that perturbs the particle, and P_3 is the "observer" process that samples P_1's state, and whose (P_3's) behavior is probabilistically linked to P_1's initial state. P_3's state is of course indeterminate (*but very real*). This abstract experimental arrangement is analogous to the Stern–Gerlach experiment, where P_1 is the

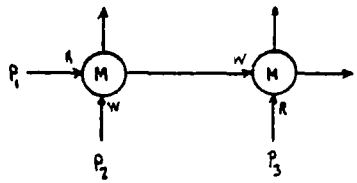

Fig. 4. A typical experiment, where P_1 is the particle of interest, P_2 the experimental probing process, and P_3 the observation of P_1's final state.

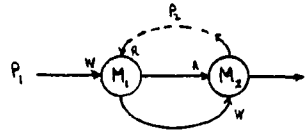

Fig. 5. Process P_1 can exchange a bit with itself if the timing is right.

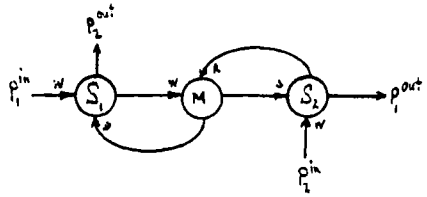

Fig. 6. Using synchronizers to remove nondeterministic outcomes.

electron, P_2 the magnetic field (photons), and P_3 the final spin observation. Notice that this arrangement is identical to that of the definition of collision-connectibility, and illuminates the origins of the definition of a *p* event in particle physics.

Another primitive form of interaction is shown in Figure 5. If the information obtained by P_2 and M_1 from P_1 is deposited in M_2 before P_1 arrives at M_2, then P_1 is unaffected by the whole interaction. This is reminiscent of virtual particle emission and absorption[4] (Rohrlich, 1983).

In our model, synchronization can only be accomplished using synchronization actors (Section 2). The purpose of synchronization is to reduce or eliminate the randomness of asynchronous interactions by causally linking a *particular c* event in one *c* process with a *particular c* event in a second *c* process. See Figure 6. With the initial conditions that M = #,

[4] Manthey and Moret (1983) shows that computational momentum is conserved in a W/R over a memory cell.

S_1 = open, S_2 = closed, this arrangement guarantees that P_2 will not attempt to Read M before P_1 has put a bit there, and that P_1 cannot Write a new bit into M before P_2 has read the previous one. Thus, if P_1 and P_2 are otherwise deterministic c processes, then so are both their futures.

Nature, unlike system programmers, has no vested interest in guaranteeing a particular outcome, and hence synchronizers would not be expected to play any deep role in the computational metaphor's models.

5. CONCLUSION

We have shown how the particle physicists' concept of an "event" can be defined in the simpler terms of computational events and processes. Besides the intellectual satisfaction of reducing some traditional terms to still simpler ones, the approach shows how one can retain the principle of cause and effect even in the face of nondeterminism. This of course conflicts in numerous ways with Bohr's "Copenhagen interpretation," but the point of view being advanced here is that that interpretation was saddled with the view that nondeterminism is incompatible with causality. Perhaps this mistake is understandable in view of the success of the determinism of Newtonian physics, where causality and deterministic outcome are essentially identified with each other. In having a realistic model and actual mechanism, the computational metaphor has the advantage of neatly sidestepping many of the usual paradoxes, e.g., "collapse of the wave function."

Can one reformulate quantum physics on a computational foundation? Can theoretical computability results be applied to physics? Can physics' mathematical methods be transported to problems in computer science? The answer to these and similar questions is uncertain. In Manthey and Moret (1983) we show that communicating c processes conserve (computational) momentum, and reveal the logical origin of quantum numbers in our human perceptual/intellectual habits. But much remains, e.g., showing the *necessity* of the appearance of wavelike phenomena (via ensembles of c processes) and demonstrating formally the existence of a Heisenberg-type uncertainty principle based on complex amplitudes.[5] We are currently working on a matrix mechanical (i.e., operational) semantics for actor networks. A model for quantum mechanical phenomena grounded in the abstract properties of concurrent computational systems could well bear such fruit.

[5] Mehlberg: "The individuals of the atomic world are not sometimes particles and sometimes waves—they are always particles... In other words, it is not the case that the same entity is both a particle and a wave. The waves are statistical properties of particles" (1980, pp. 28 and 29).

ACKNOWLEDGMENTS

I would like to thank Colston Chandler of the Department of Physics and Astronomy at the University of New Mexico for his comments and suggestions on this manuscript.

APPENDIX: A BRIEF COMPARISON OF APPROACHES

It seems reasonable, given the novelty of the computation metaphor compared to traditional particle physics, Ballantine (1970), and Everett (1957), to compare them briefly and qualitatively.

Everett's basic point is that the observer–system dichotomy can be very profitably replaced by a subsystem–subsystem dichotomy. That is, the observer and the observed are viewed as subsystems of a common supersystem. He concludes, among other things, that (1) the EPR paradox is vacuous, and (2) that the traditional placing of the observer outside the system to be observed leads only to conceptual difficulties (e.g., Bohr's idealistic interpretation). Wheeler (1957) agrees.

In the computational metaphor, Everett's common supersystem is represented by the directed system graph in which both the observer and observed processes by definition exist. Although we have not shown it, we believe that the two points of view are formally equivalent (as they must be if the computational metaphor indeed models QM). The metaphor had led us, in any event, to the same conclusions in ignorance of Everett's work.

Ballantine's task, rather different from Everett's, is to analyze ensemble versus individual process behavior in the context of hidden variable theories. While the approach is the traditional external observer and statistical one, the paper is uniformly illuminating. Ballantine's conclusion, that hidden variable theories are not viable (except at best in a very warped form), is the same as ours (Manthey and Moret, 1983; Manthey, unpublished), although the reasoning is very different.

Mehlberg's analysis (1980) is similar in its general approach to Ballantine's.

REFERENCES

Baker, H., Jr. (1978). Actor systems for real-time computation, TR-197, MIT/LCS, March 1978.

Ballantine, I. E. (1970). The statistical interpretation of quantum mechanics, *Rev. Mod. Phys.*, **42**, 358.

Beckman, F. S. (1981). *Mathematical Foundations of Programming*. Addison-Wesley, Reading, Massachusetts.

Everett, H. (III). (1957). "Relative state" formulation of quantum mechanics, *Rev. Mod. Phys.*, **29**, 454.

Hewitt, C., and Baker, H., Jr. (1977). Laws for communicating parallel processes, in *Information Processing 77*, B. Gilchrist, ed., pp. 987–992. North-Holland, Amsterdam.

Hewitt, C., and Baker, H. Jr. (1978). Actors and continuous functionals, in *Formal Descriptions of Programming Languages*, E. J. Neuhold, ed. North-Holland, Amsterdam.

MacQueen, D. B. (1979). Models for distributed computing, Institut de Recherche d'Information et d'Automatique (IRIA), Le Chesnay (France) Rapport de Recherche no. 351. April 1979. (This report, though hard to obtain, is excellent, and covers the following references and more: Hewitt and Baker, 1977; 1978; Baker, 1978.)

Manthey, M. J. (unpublished). Hierarchy in sequential and concurrent systems, TR 83-2, Computer Science Department, The University of New Mexico, submitted to *ACM Trans. Comp. Syst.*

Manthey, M. J., and Moret, B. M. E. (1983). The computational metaphor and quantum physics, *Commun. ACM.*, **26**, 2 (February 1983).

Mehlberg, H. (1980). *Time, Causality, and the Quantum Theory*, Vol. II, R. S. Cohen, ed. D. Reidel Publishing Co., Dordrecht (USA: Kluwer Boston Inc.).

Milner, R. (1978). Algebras for communicating systems, CSR-25-78, Computer Science Department, University of Edinburgh, June 1978.

Peterson, J. L. (1977). Petri nets, *ACM Computing Surv.*, **9**(3) 221–252 (September 1977).

Rohrlich, F. (1983). Facing quantum mechanical reality. *Science*, **221**, 1251 (September 23, 1983).

Wheeler, J. A. (1957). Assessment of Everett's "relative state" formulation of quantum theory, *Rev. Mod. Phys.*, **29**, p. 463.