

# Lampport's Unfinished Revolution

## Time, Clocks and the *Reordering* of Events

Papers We Love Too - San Francisco

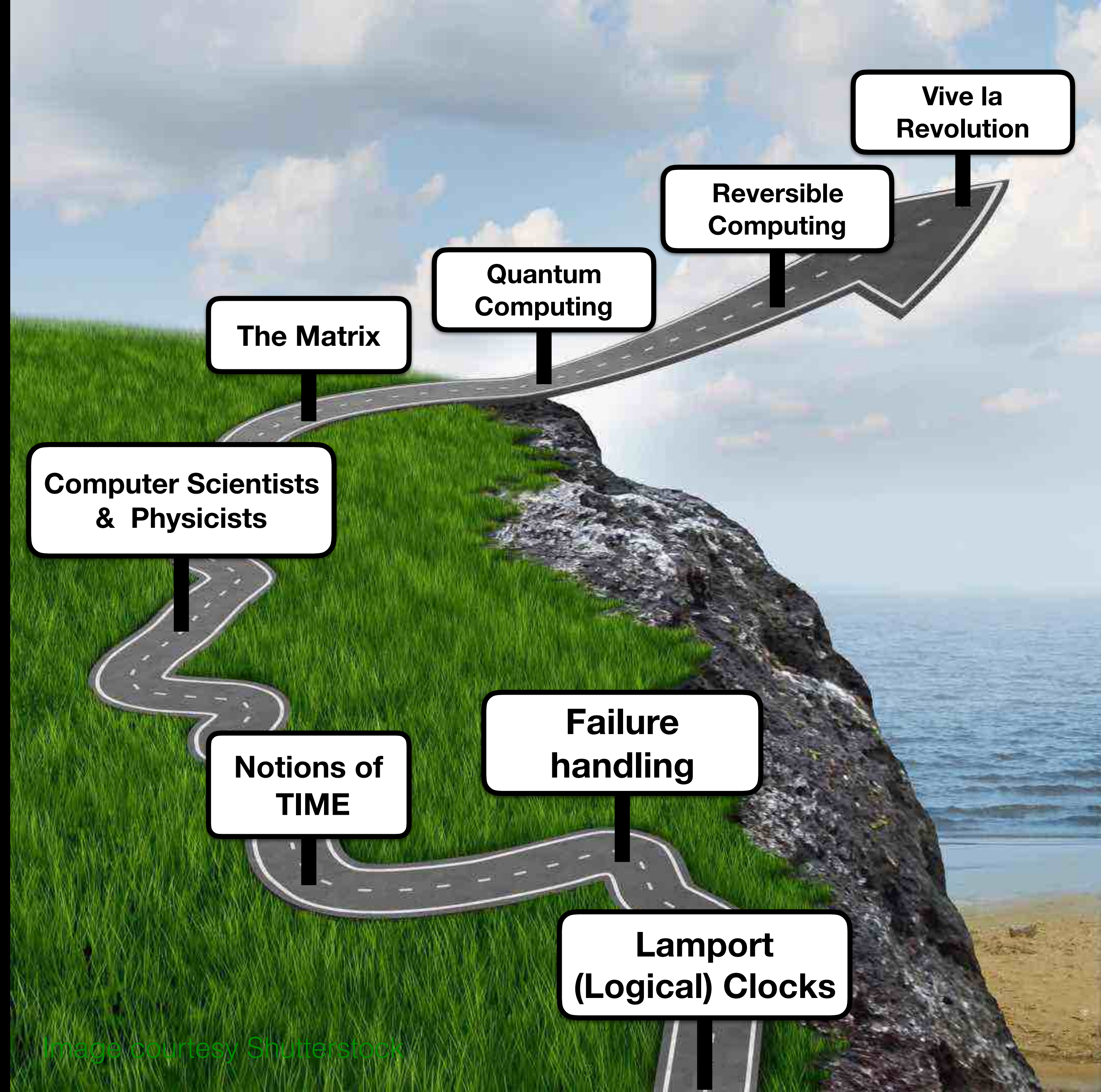
02016-7-14 18:30

***Paul Borrill, EARTH Computing, Inc***

***@plborrill [paul@borrill.com](mailto:paul@borrill.com)***



# Lamport's Unfinished Revolution





Operating  
Systems

R. Stockton Gaines  
Editor

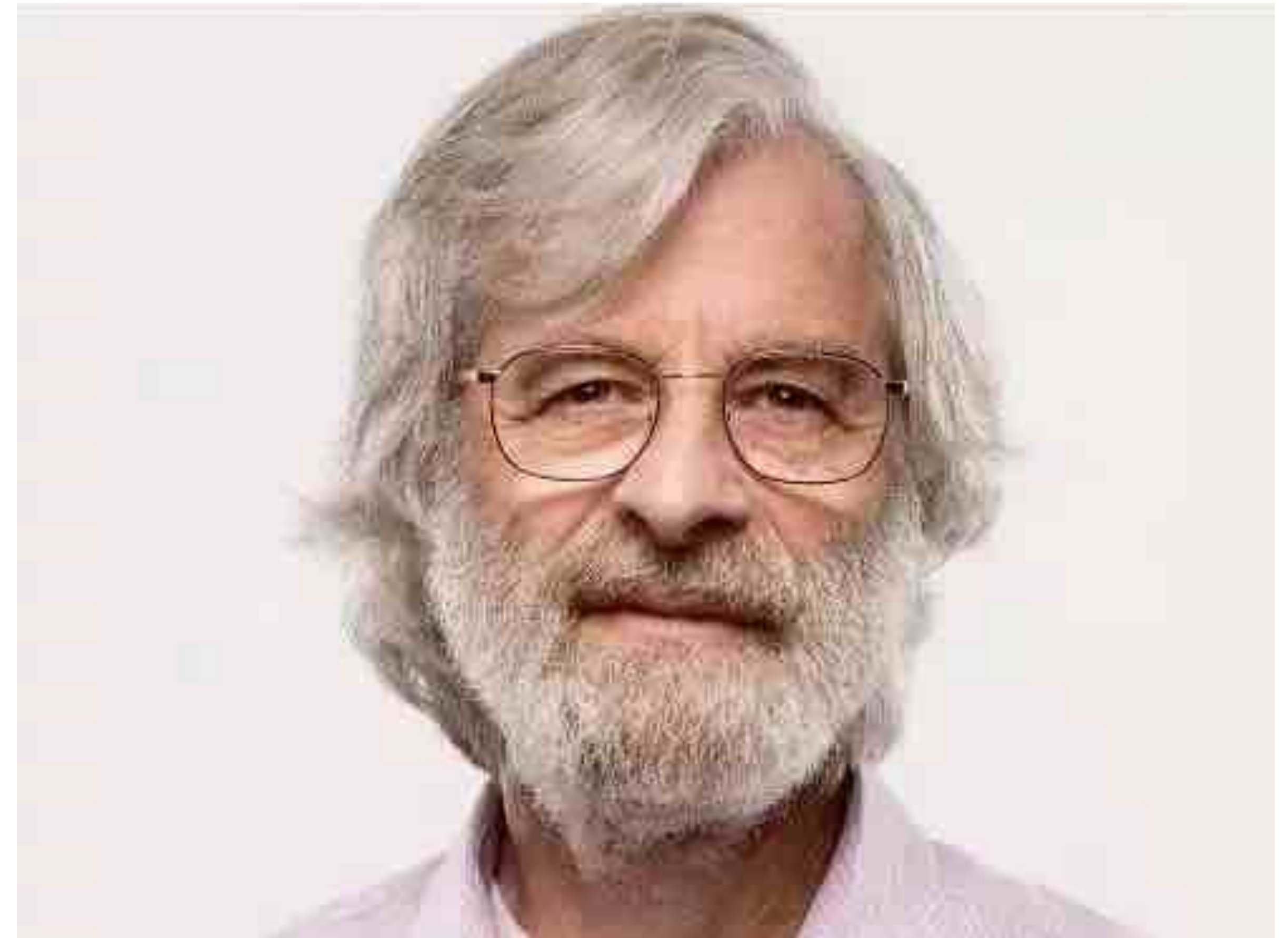
## Time, Clocks, and the Ordering of Events in a Distributed System

Leslie Lamport  
Massachusetts Computer Associates, Inc.

**The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.**

**Key Words and Phrases:** distributed systems, computer networks, clock synchronization, multiprocess systems

**CR Categories:** 4.32, 5.29



**Introduced:**

- Logical Clocks (a distributed algorithm for synchronizing a system of logical clocks which can be used to TOTALLY order events)
- A time bound on the synchronization of physical clocks (*This algorithm depends heavily on there being no faults in the system, and is not used by practitioners*)

**Lamport Clocks are all about assigning labels to events, and that those assignments must be *causally* related**



## Abstract

**The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events.**

A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems

concept of one event happening  
before another in a distributed  
system is ... is shown to define a  
**partial ordering** of the events.



## Introduction

The concept of time is fundamental to our way of thinking. It is derived from the more basic concept of the order in which events occur. We say that something happened at 3:15 if it occurred after our clock read 3:15 and before it read 3:16. The concept of the temporal ordering of events pervades our thinking about systems. For example, in an airline reservation system we specify that a request for a reservation should be granted if it is made before the flight is filled. However, we will see that this concept must be carefully reexamined when considering events in a distributed system.

A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages.

A network of interconnected computers, such as the ARPA net, is a distributed system. A single computer can also be viewed as a distributed system in which the central control unit, the memory units, and the input-output channels are separate processes. A system is distributed if the message transmission delay is not negligible compared to the time between events in a single process.

The concept of **time** is fundamental to our way of thinking. It is derived from the **more basic concept of the order** in which events occur.

A distributed system consists of a collection of distinct processes which are spatially separated, and which communicate with one another by exchanging messages.



We will concern ourselves primarily with systems of spatially separated computers. However, many of our remarks will apply more generally. In particular, a multiprocessing system on a single computer involves problems similar to those of a distributed system because of the unpredictable order in which certain events can occur.

In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation "happened before" is therefore only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.

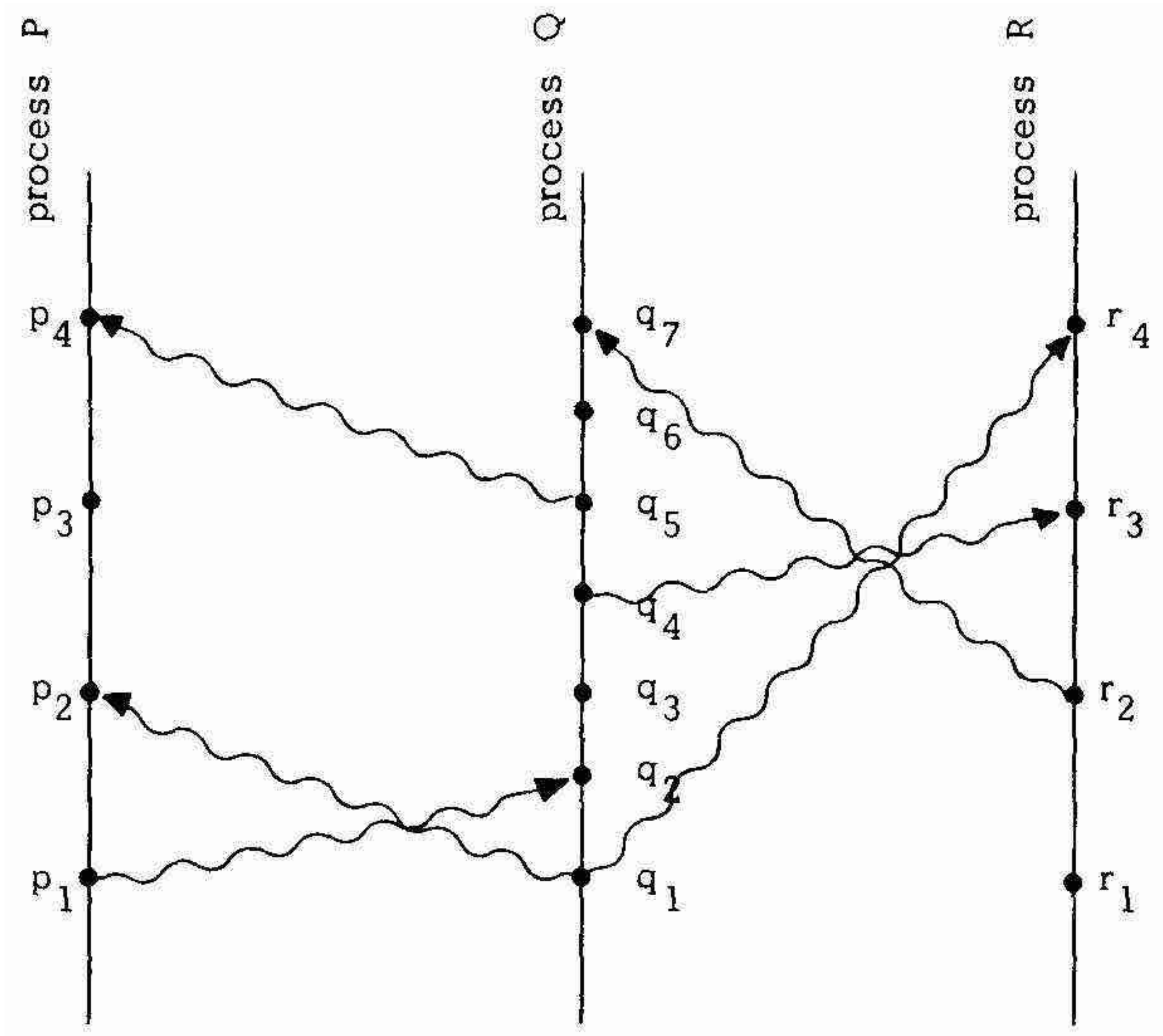
In this paper, we discuss the partial ordering defined by the "happened before" relation, and give a distributed algorithm for extending it to a consistent total ordering of all the events. This algorithm can provide a useful mechanism for implementing a distributed system. We illustrate its use with a simple method for solving synchronization problems. Unexpected, anomalous behavior can occur if the ordering obtained by this algorithm differs from that perceived by the user. This can be avoided by introducing real, physical clocks. We describe a simple method for synchronizing these clocks, and derive an upper bound on how far out of synchrony they can drift.

Introduces **"happened before"** relation

**"happened before"** is meaningless unless intimately associated with **"happened where"**

Well articulated by Lamport, but frequently misunderstood by readers



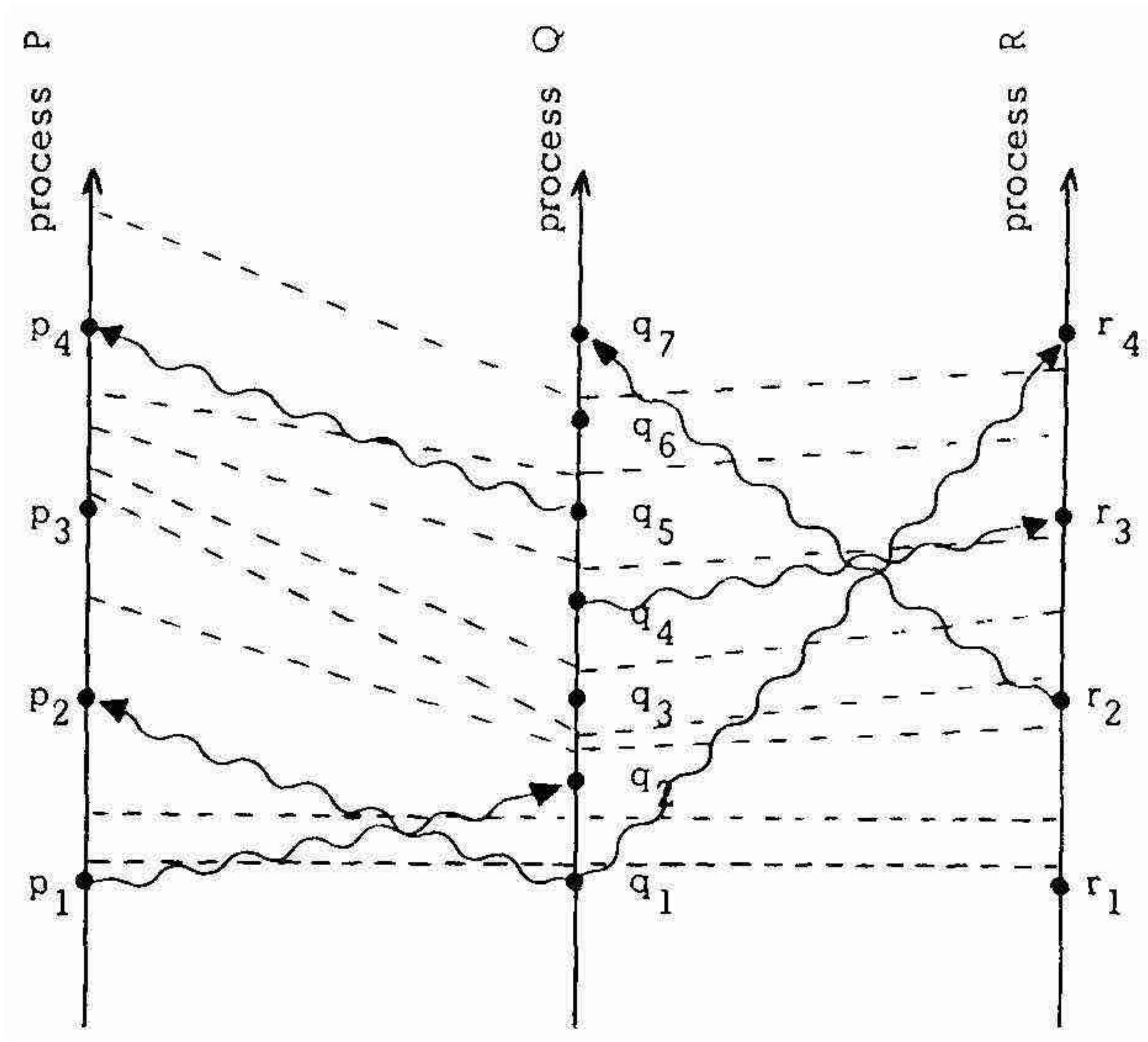


## Space-time diagram

The horizontal direction represents space, and the vertical direction represents time—later times being higher than earlier ones. The dots denote events, the vertical lines denote processes, and the wavy lines denote messages

**Basic Space-Time Diagram,  
Processes each along their own “timeline”**

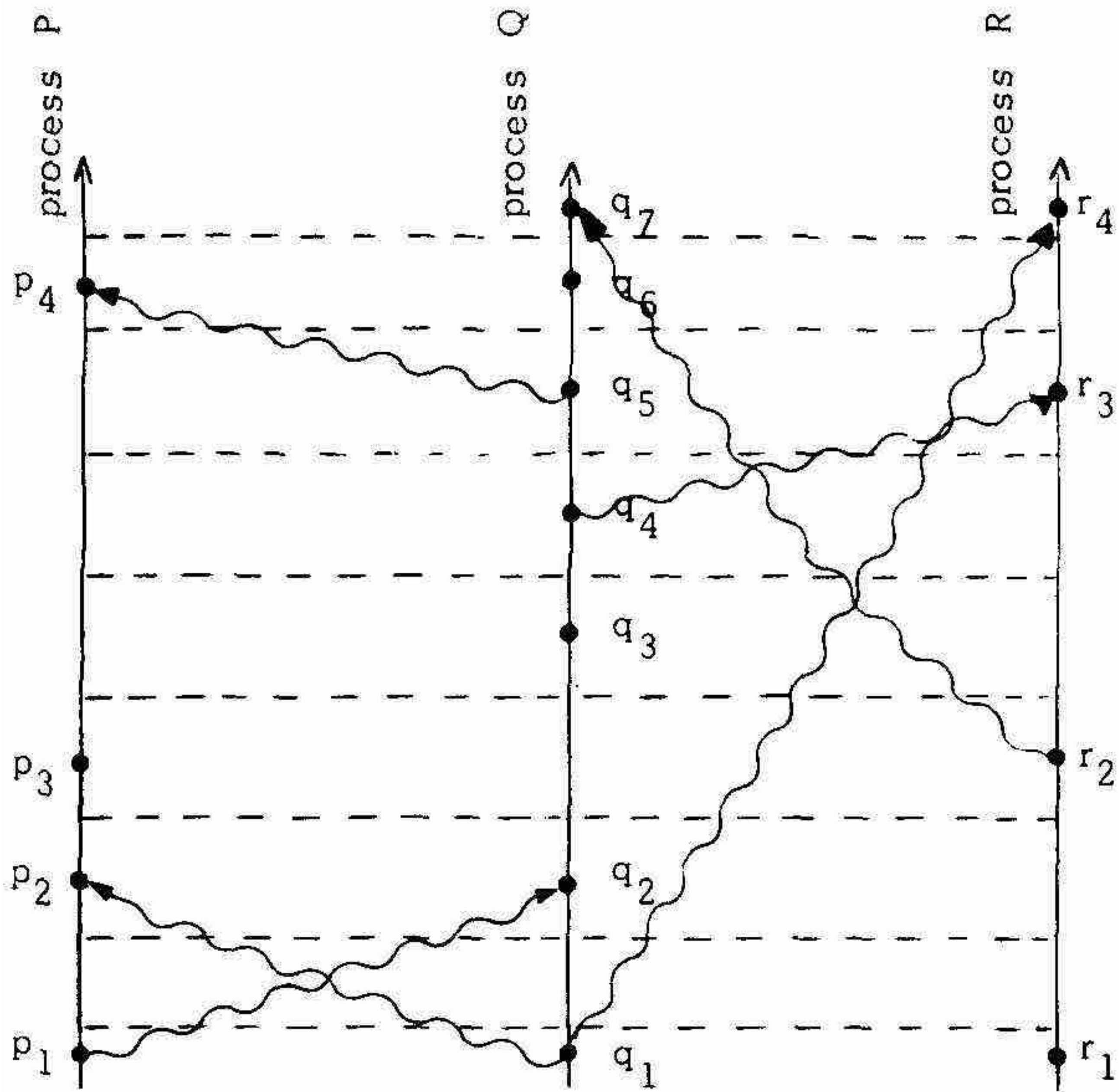




## Space-time diagram

The horizontal direction represents space, and the vertical direction represents time—later times being higher than earlier ones. The dots denote events, the vertical lines denote processes, and the wavy lines denote messages





## Space-time diagram

The horizontal direction represents space, and the vertical direction represents time—later times being higher than earlier ones. The dots denote events, the vertical lines denote processes, and the wavy lines denote messages

## Key Assumptions:

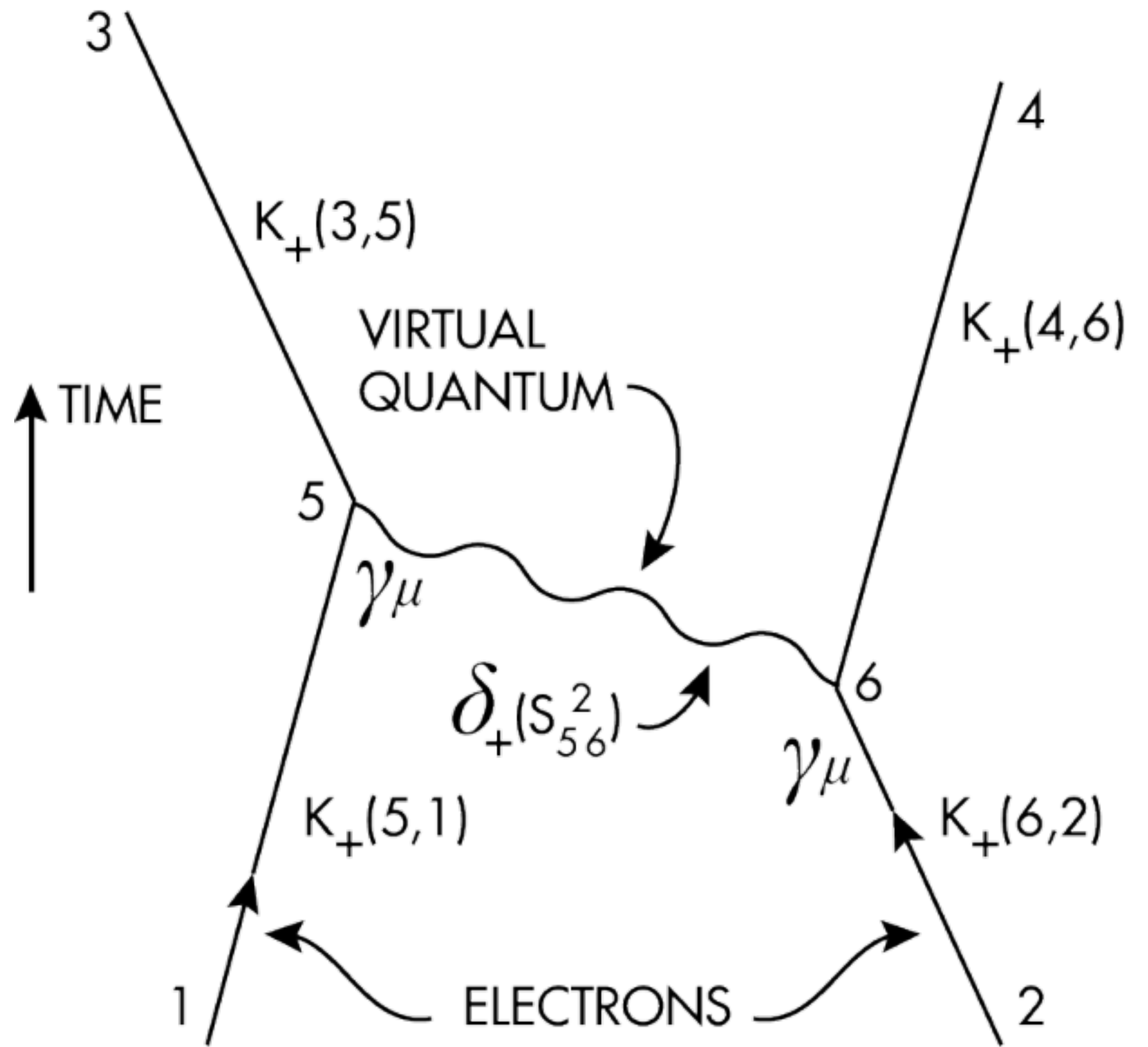
Events not Durations

Continuous Physical Time Background

$\therefore$  Irreversible Time & Messages

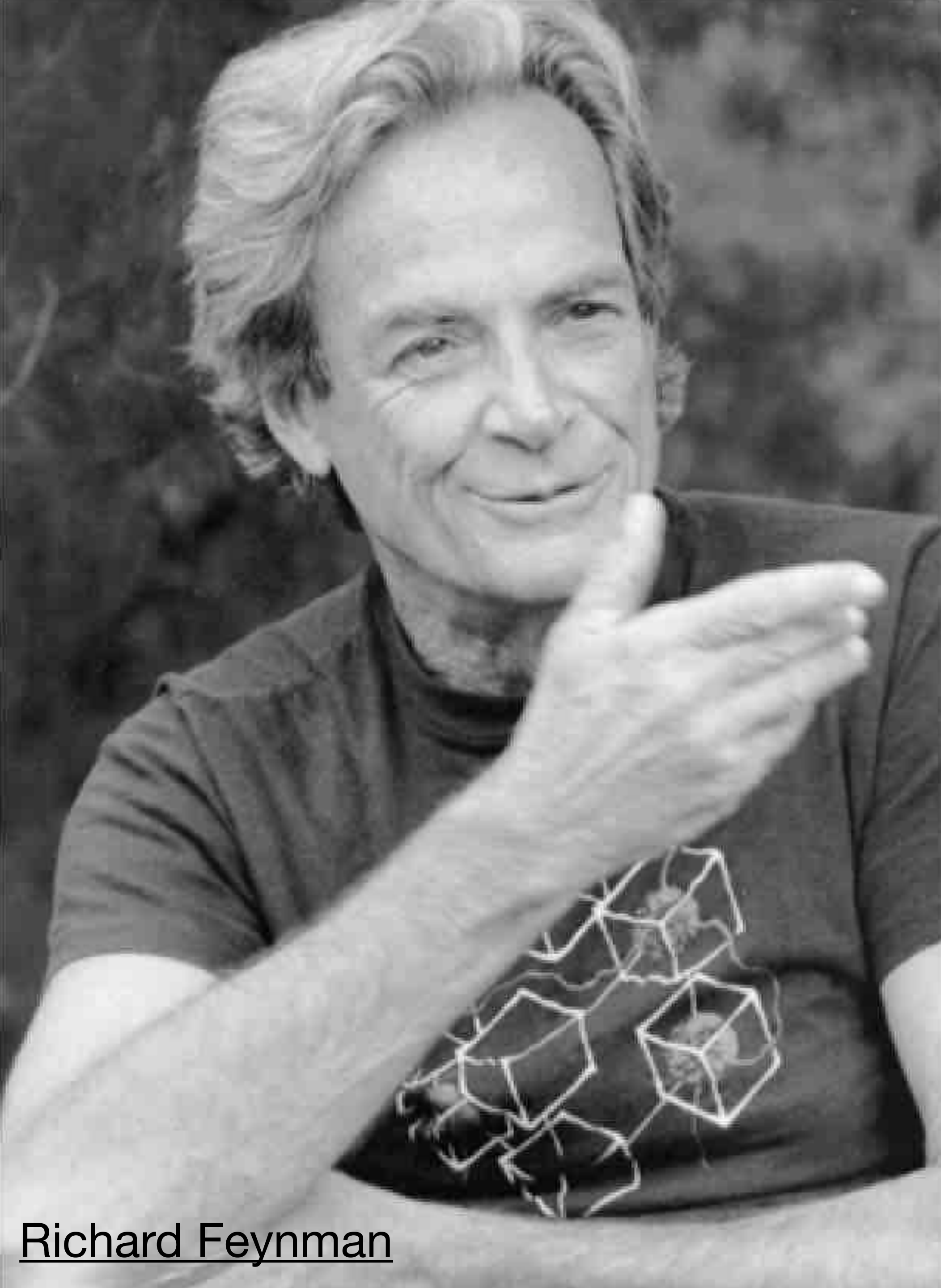
$\therefore$  Timestamps are Monotonic



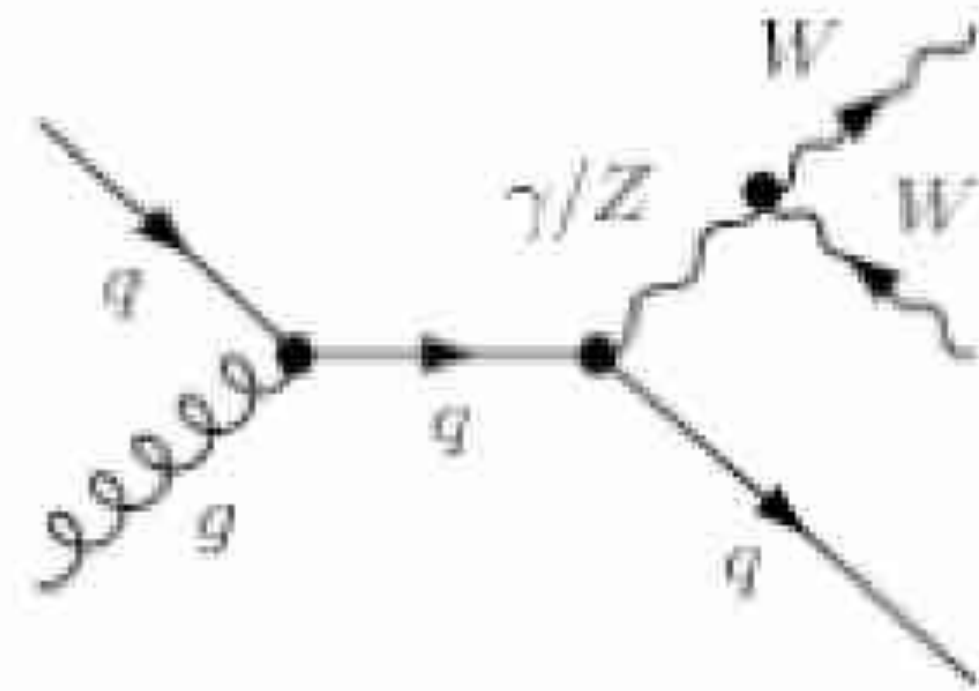


Richard Feynman

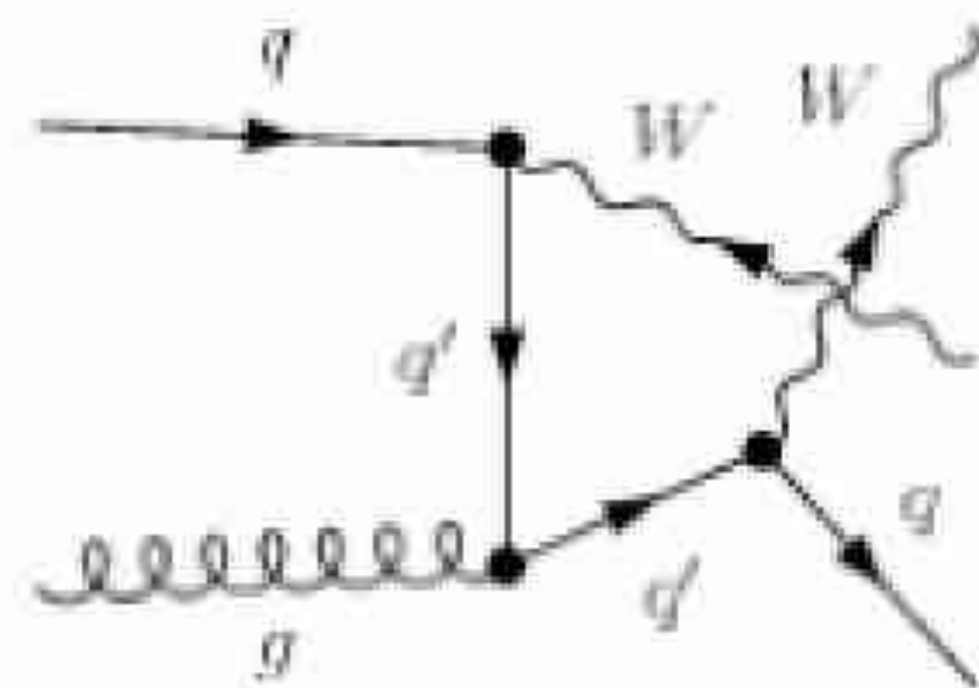




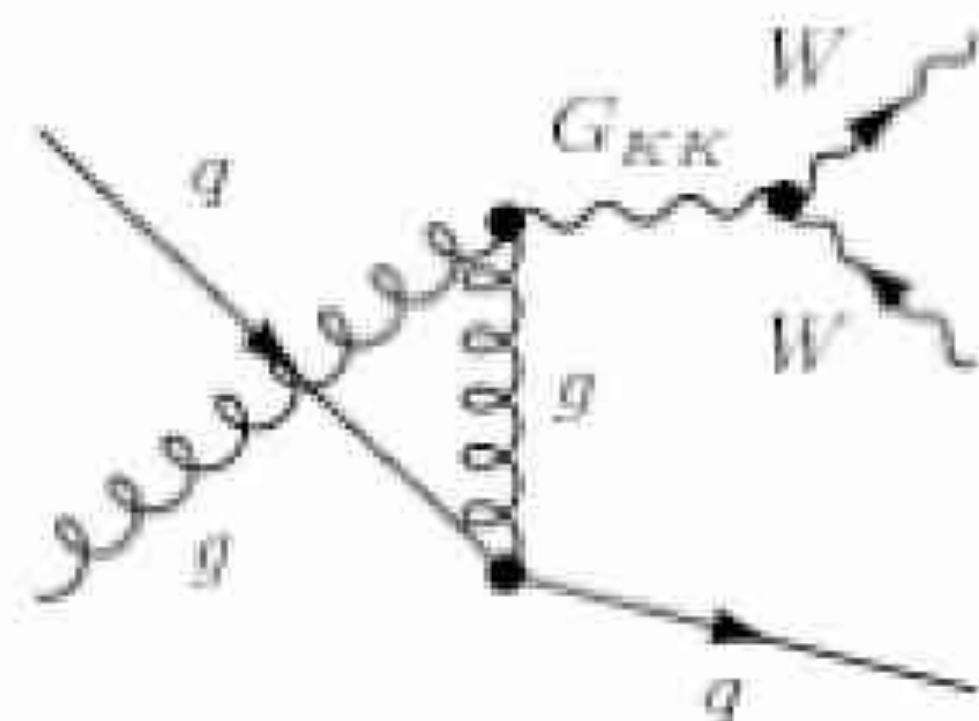
Richard Feynman



(1)



(4)



(7)



Leslie Lamport



event b. Two events are concurrent if neither can causally affect the other. For example, events p<sub>3</sub> and q<sub>3</sub> of Figure 1 are concurrent. Even though we have drawn the diagram to imply that q<sub>3</sub> occurs at an earlier physical time than p<sub>3</sub>, process P cannot know what process Q did at q<sub>3</sub> until it receives the message at p<sub>4</sub>. (Before event p<sub>4</sub>, P could at most know what Q was planning to do at q<sub>3</sub>.)

This definition will appear quite natural to the reader familiar with the invariant space-time formulation of special relativity, as described for example in [1] or the first chapter of [2]. In relativity, the ordering of events is defined in terms of messages that could be sent. However, we have taken the more pragmatic approach of only considering messages that actually are sent. We should be able to determine if a system performed correctly by knowing only those events which did occur, without knowing which events could have occurred.

This definition will appear quite natural to the reader familiar with the invariant space-time formulation of **special relativity** ... we have taken the more pragmatic approach of only considering messages that actually are sent. We should be able to determine if a system performed correctly by knowing only those events which did occur, without knowing which events could have occurred.



$N \leq i$ , and  $b \Rightarrow a$  otherwise; where  $N$  is the total number of processes.]

The ordering  $\Rightarrow$  depends upon the system of clocks  $C_i$ , and is not unique. Different choices of clocks which satisfy the Clock Condition yield different relations  $\Rightarrow$ . Given any total ordering relation  $\Rightarrow$  which extends  $\rightarrow$ , there is a system of clocks satisfying the Clock Condition which yields that relation. It is only the partial ordering which is uniquely determined by the system of events.

Being able to totally order the events can be very useful in implementing a distributed system. In fact, the reason for implementing a correct system of logical clocks is to obtain such a total ordering. We will illustrate the use of this total ordering of events by solving the following version of the mutual exclusion problem. Consider a system composed of a fixed collection of processes which share a single resource. Only one process can use the resource at a time, so the processes must synchronize themselves to avoid conflict. We wish to find an algorithm for granting the resource to a process which satisfies the following three conditions: (I) A process which has been granted the resource must release it before it can be granted to another process. (II) Different requests for the resource must be granted in

The ordering  $\Rightarrow$  depends upon the system of clocks  $C_i$ , and is not unique. **Different choices of clocks which satisfy the Clock Condition yield different relations  $\Rightarrow$ .**

Given any **total ordering** relation  $\Rightarrow$  which extends  $\rightarrow$ , there is a system of clocks satisfying the Clock Condition which yields that relation. **It is only the partial ordering which is uniquely determined by the system of events**

making a system become a single node and instead it just involves making sure that each process learns about all other processes' operations.

To simplify the problem, we make some assumptions. They are not essential, but they are introduced to avoid distracting implementation details. We assume first of all that for any two processes  $P_i$  and  $P_j$ , the messages sent from  $P_i$  to  $P_j$  are received in the same order as they are sent. Moreover, **we assume that every message is eventually received.** (These assumptions can be avoided by introducing message numbers and message acknowledgment protocols.) We also assume that a process can send messages directly to every other process.

Each process maintains its own request queue which is never seen by any other process. We assume that the request queues initially contain the single message  $To:Po$  requests resource, where  $P_o$  is the process initially granted the resource and  $To$  is less than the initial value of any clock.

***we assume that every message is eventually received.*** (These assumptions can be avoided by introducing message numbers ***and message acknowledgment protocols.***) We also assume that a process can send messages directly to every other process



Each process independently simulates the execution of the State Machine, using the commands issued by all the processes. Synchronization is achieved because all processes order the commands according to their timestamps (using the relation  $\Rightarrow$ ), so each process uses the same sequence of commands. A process can execute a command timestamped  $T$  when it has learned of all commands issued by all other processes with timestamps less than or equal to  $T$ . The precise algorithm is straightforward, and we will not bother to describe it.

This method allows one to implement any desired form of multiprocess synchronization in a distributed system. However, the resulting algorithm requires the active participation of all the processes. A process must know all the commands issued by other processes, so that the failure of a single process will make it impossible for any other process to execute State Machine commands, thereby halting the system.

**Introduces the state machine: This is the genesis of the Paxos Consensus Algorithm**

“The precise algorithm is straightforward and we will not bother to describe it”



The **problem of failure** is a difficult one, and it is beyond the scope of this paper to discuss it in any detail. We will just observe that **the entire concept of failure is only meaningful in the context of physical time. Without physical time, there is no way to distinguish a failed process from one which is just pausing between events.** A user can tell that a system has "crashed" only because he has been waiting too long for a response. A method which works despite the failure of individual processes or communication lines is described in [3].

The **problem of failure** is a difficult one, and it is beyond the scope of this paper to discuss it in any detail.

**the entire concept of failure is only meaningful in the context of physical time**

**Without physical time, there is no way to distinguish a failed process from one which is just pausing between events**

This significantly pre-dates the FLP result in 1985





**Physical Clocks**

Let us introduce a physical time coordinate into our space-time picture, and let  $C_i(t)$  denote the reading of the clock  $C_i$  at physical time  $t$ .

[Footnote 8: We will assume a Newtonian space-time. If the relative motion of the clocks or gravitational effects are not negligible, then  $C_i(t)$  must be deduced from the actual clock reading by transforming from proper time to the arbitrarily chosen time coordinate.]

For mathematical convenience, we assume that the clocks run continuously rather than in discrete "ticks." (A discrete clock can be thought of as a continuous one in which there is an error of up to  $1/2$  "tick" in reading it.) More precisely, we assume that  $C_i(t)$  is a continuous, differentiable function of  $t$  except for isolated jump discontinuities where the clock is reset. Then  $dC_i(t)/dt$  represents the rate at which the clock is running at time  $t$ .

In order for the clock  $C_i$  to be a true physical clock, it must run at approximately the correct rate. That is, we must have  $dC_i(t)/dt \approx 1$  for all  $t$ . More precisely, we will assume that the following condition is satisfied:

PC<sub>1</sub>. There exists a constant  $\kappa \ll 1$   
such that for all  $i$ :  $|dC_i(t)/dt - 1| < \kappa$

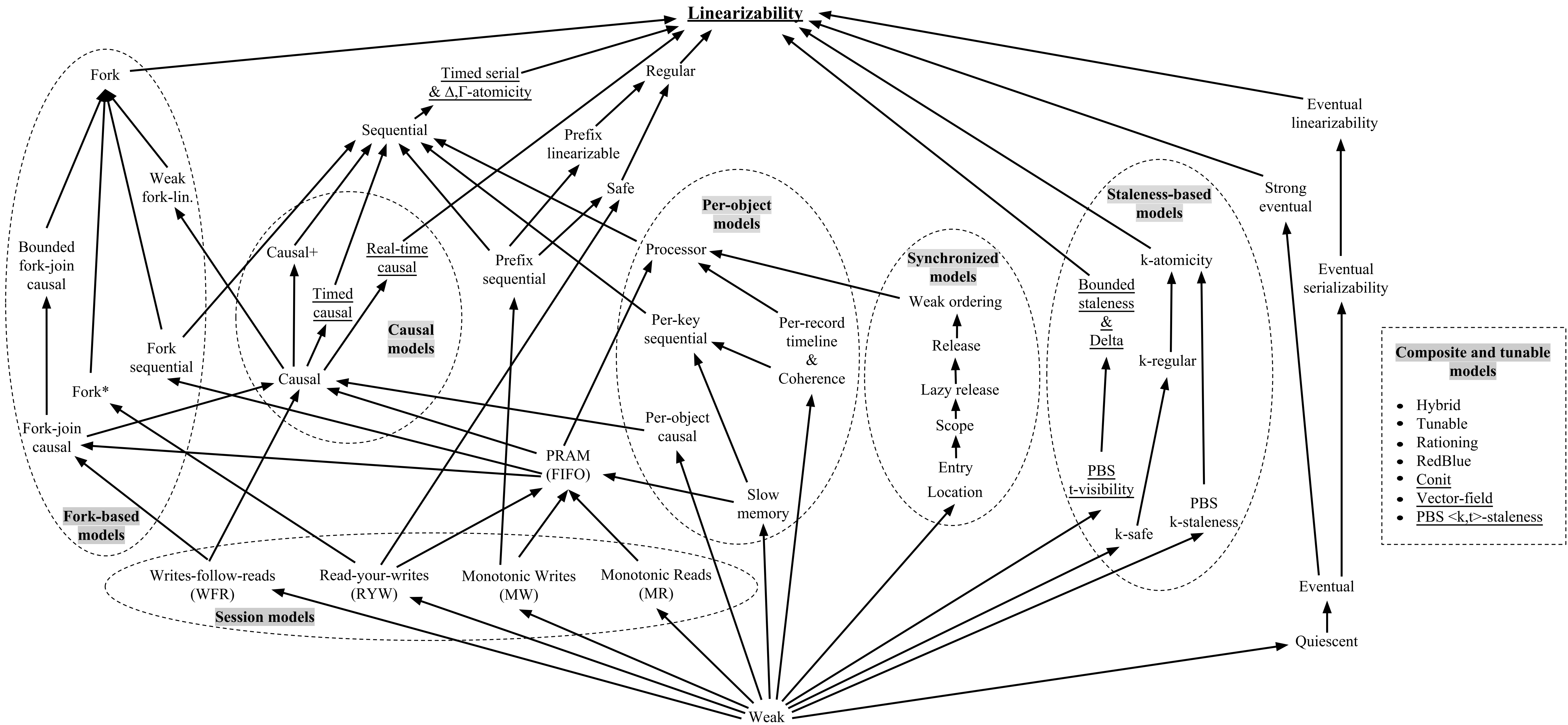
For typical crystal controlled clocks,  $\kappa \leq 10^{-6}$ .

*[Footnote 8: We will assume a Newtonian space-time. If the relative motion of the clocks or gravitational effects are not negligible, then  $C_i(t)$  must be deduced from the actual clock reading by transforming from proper time to the arbitrarily chosen time coordinate.]*

The infamous footnote 8 ....



Principal assumption: a smooth  
"background" of Minkowski spacetime

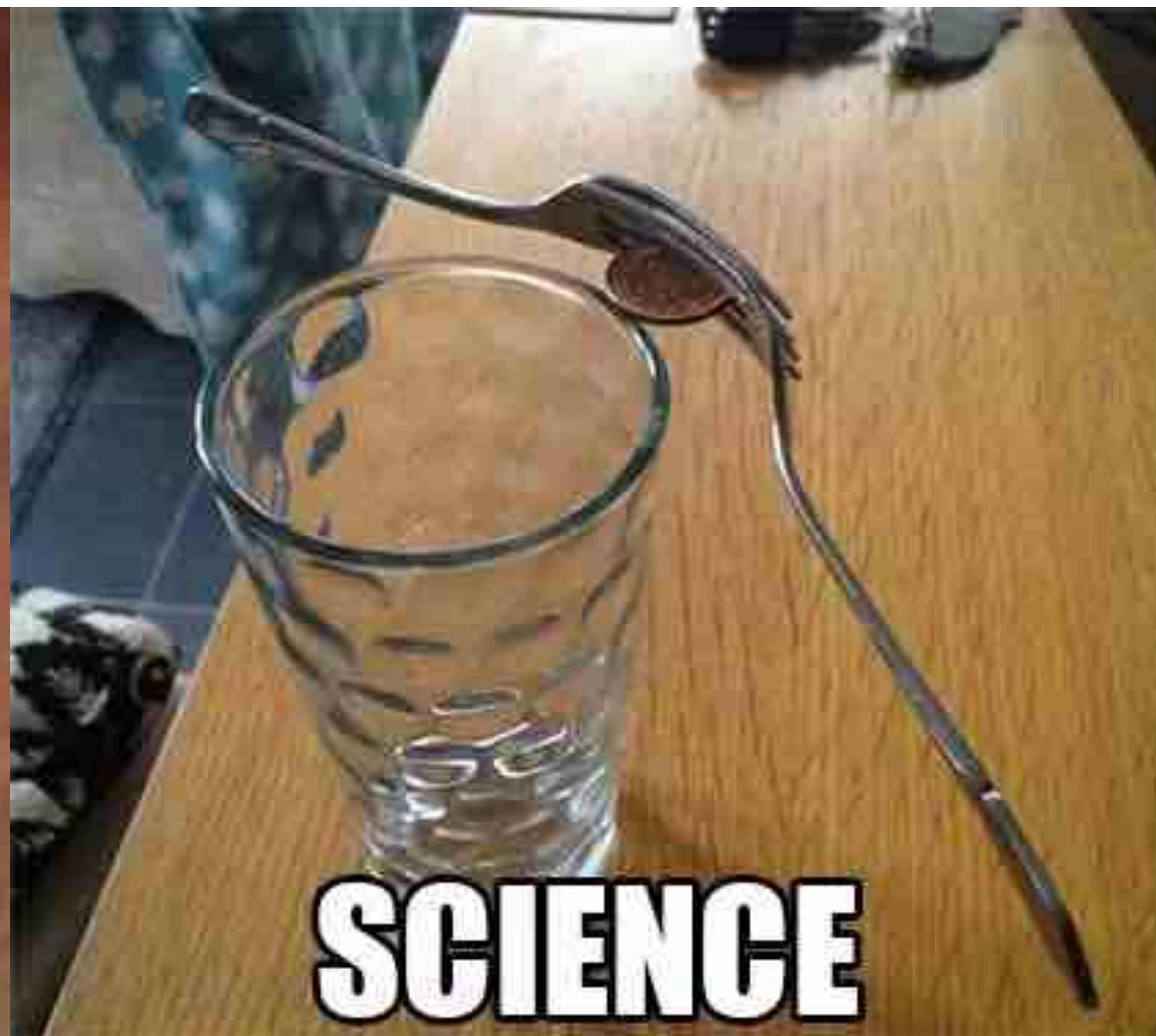


**All Roads Lead To Linearizability**





**BL: Before Lamport**



**AL: After Lamport**

**Who will finish the revolution  
started by Leslie Lamport?**

***A General Theory of Concurrency?***



# Epicycles?

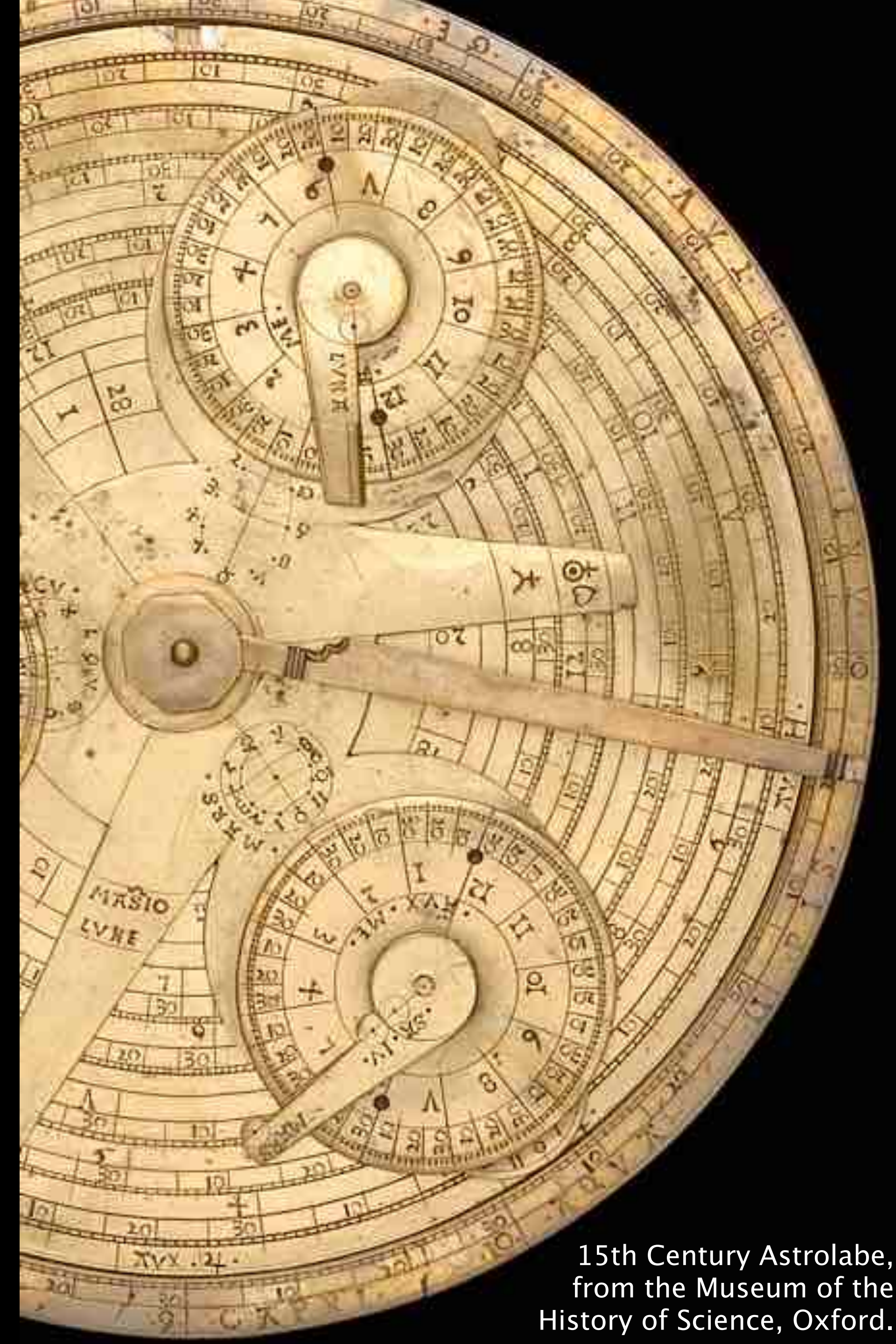
Epicycles rotated with a period of a Earth year, they were nothing but the shadow of Earth's motion. Other adjustments required still more circles; it took fifty-five circles to get it all to work. By assigning the right periods to each of the big circles, Ptolemy calibrated the model to a remarkable degree of accuracy.

A few centuries later, Islamic astronomers fine-tuned the Ptolemaic model, and in Tycho's time it predicted the positions of the planets, the sun, and moon to an accuracy of 1 part in 1,000—good enough to agree with most of Tycho's observations.

Ptolemy's model was beautiful mathematically, and its success convinced astronomers and theologians for more than a millennium that its premises were correct. And how could they be wrong? After all, the model had been confirmed by observation.\*

***Then along came Copernicus ...***

*\*FROM Smolin, Lee. "Time Reborn" (2013).*



15th Century Astrolabe,  
from the Museum of the  
History of Science, Oxford.



Ptolemyi

*or*

Copernici?





The Mutual Exclusion Problem  
Part I: A Theory of Interprocess Communication

L. Lamport<sup>1</sup>  
Digital Equipment Corporation

6 October 1980

Revised:  
1 February 1983  
1 May 1984  
27 February 1985  
June 26, 2000

To appear in *Journal of the ACM*

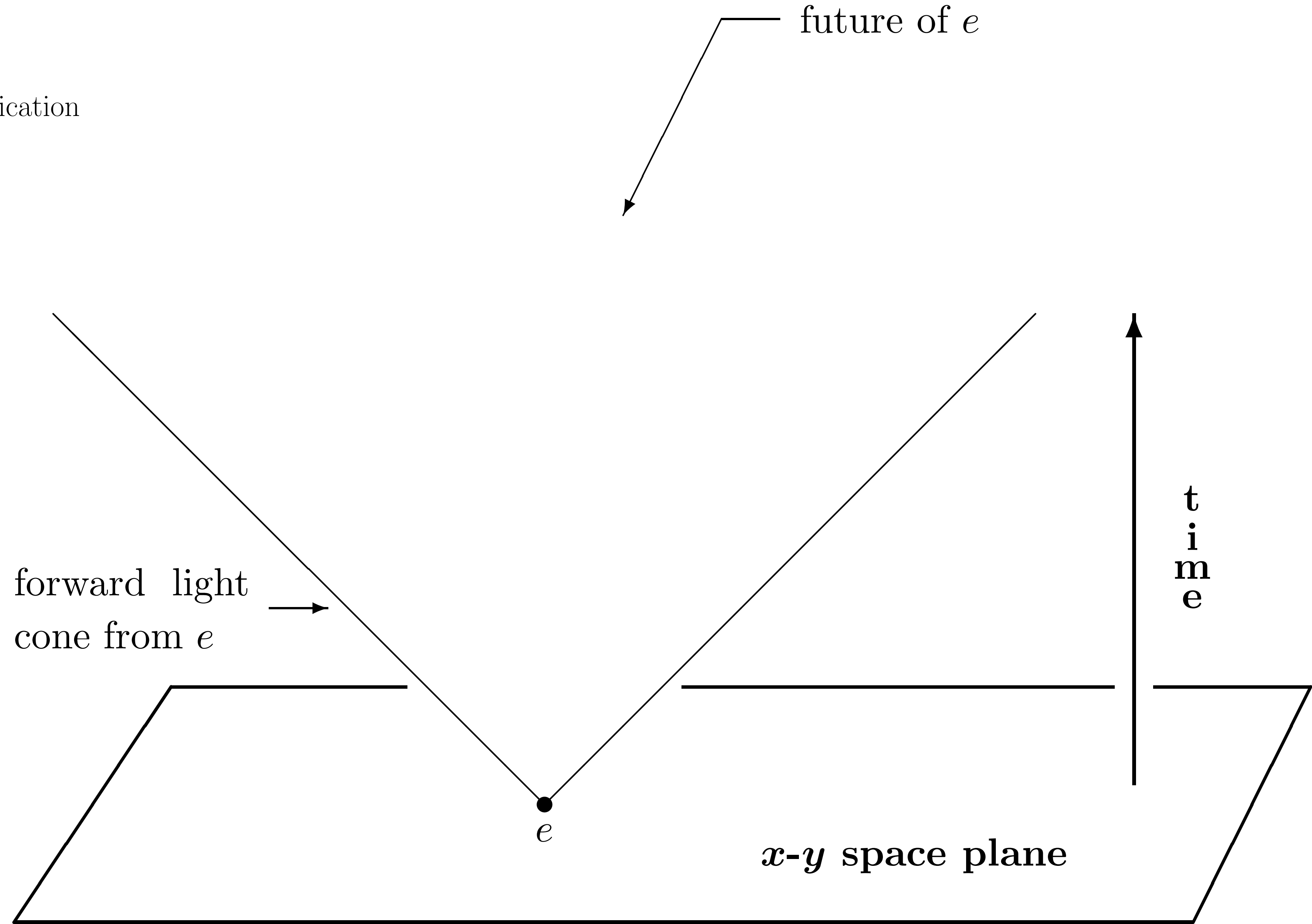
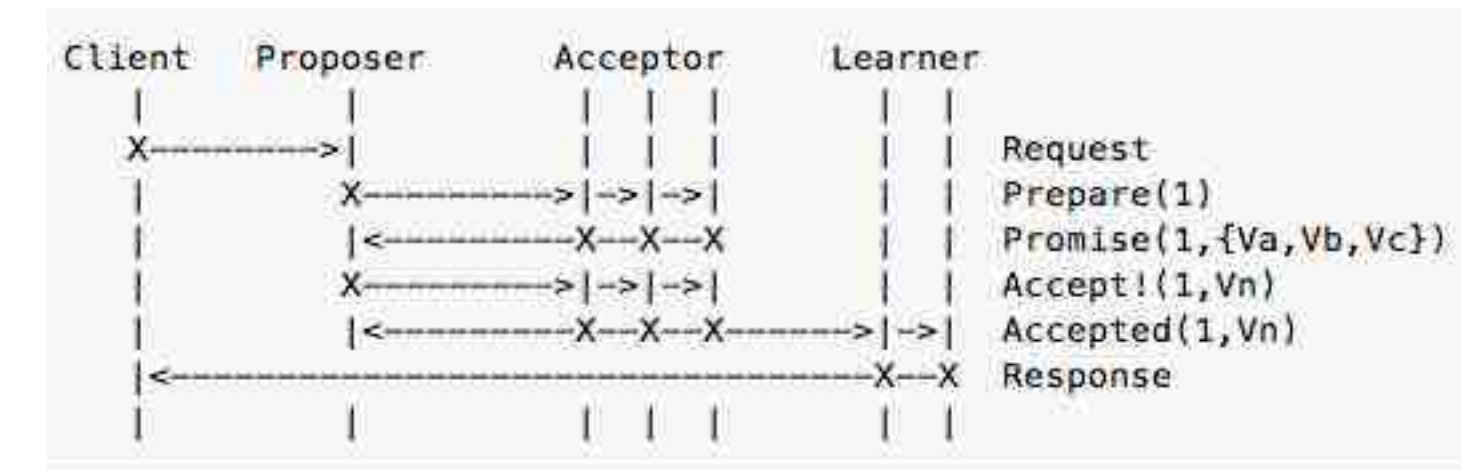
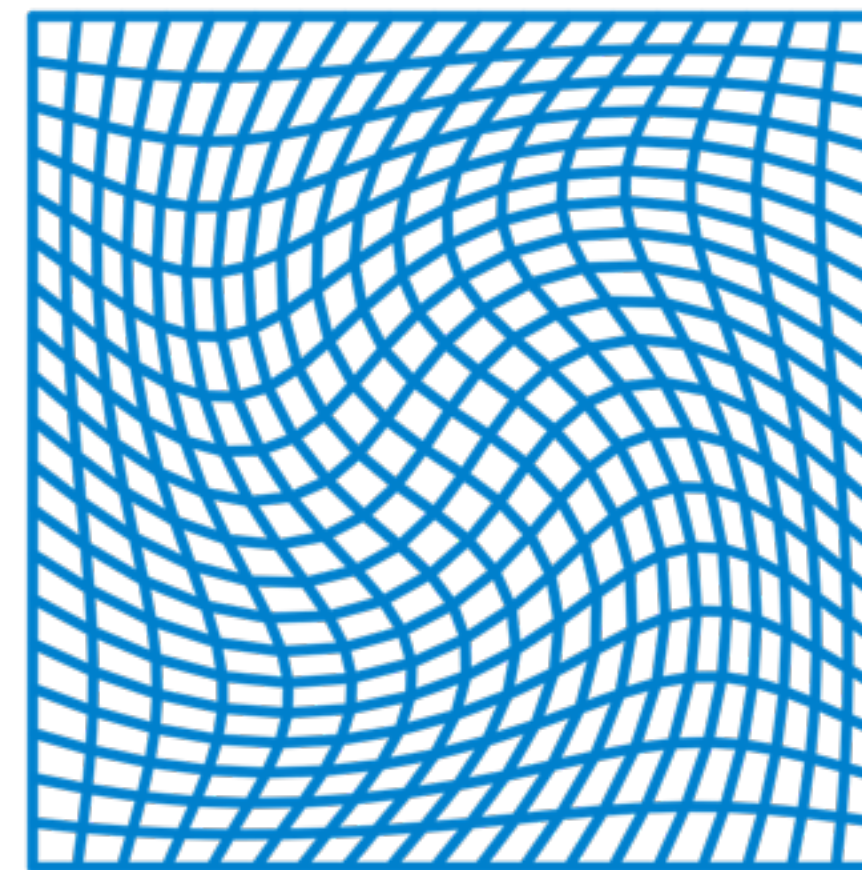


Figure 1: Space-Time

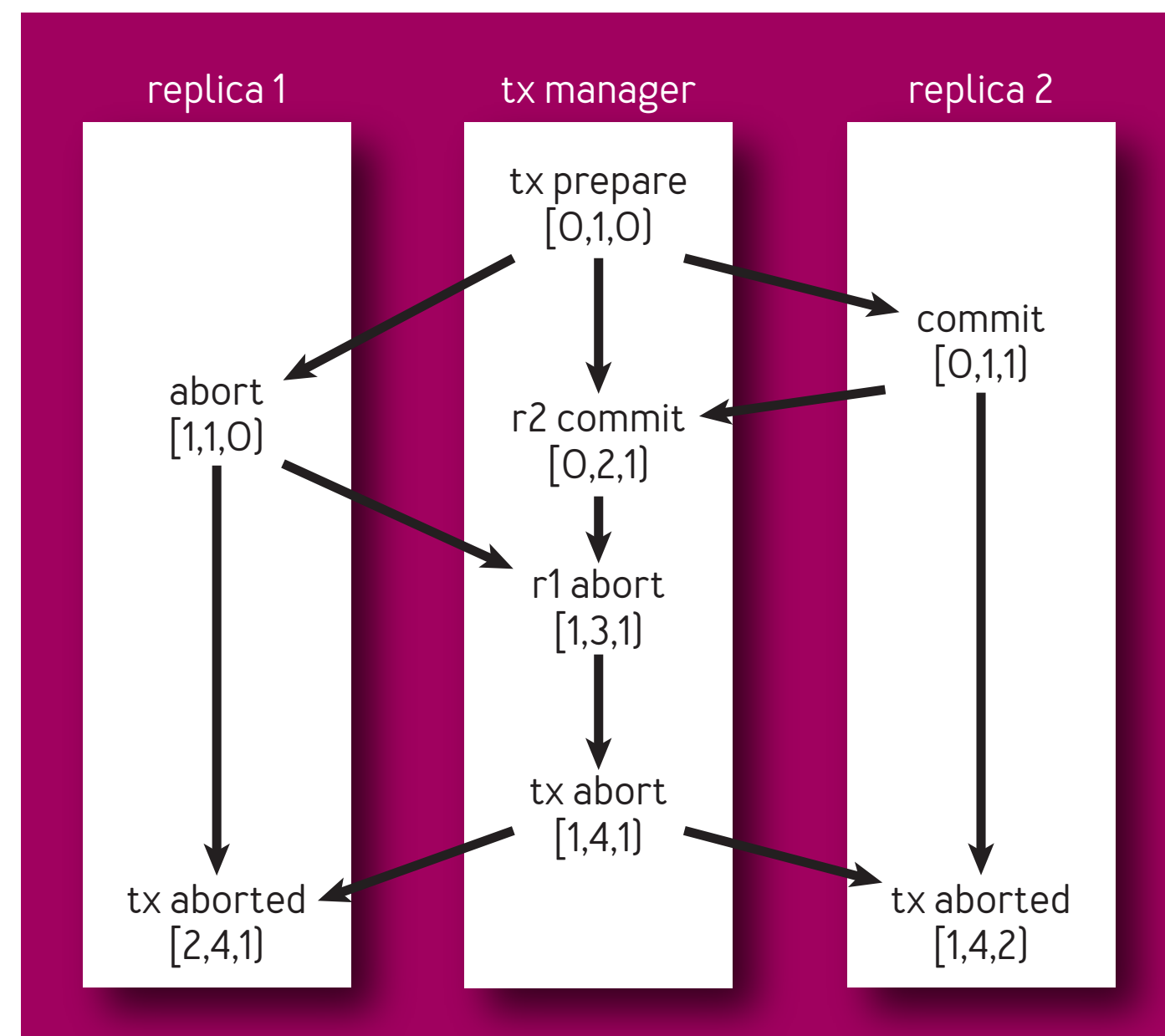




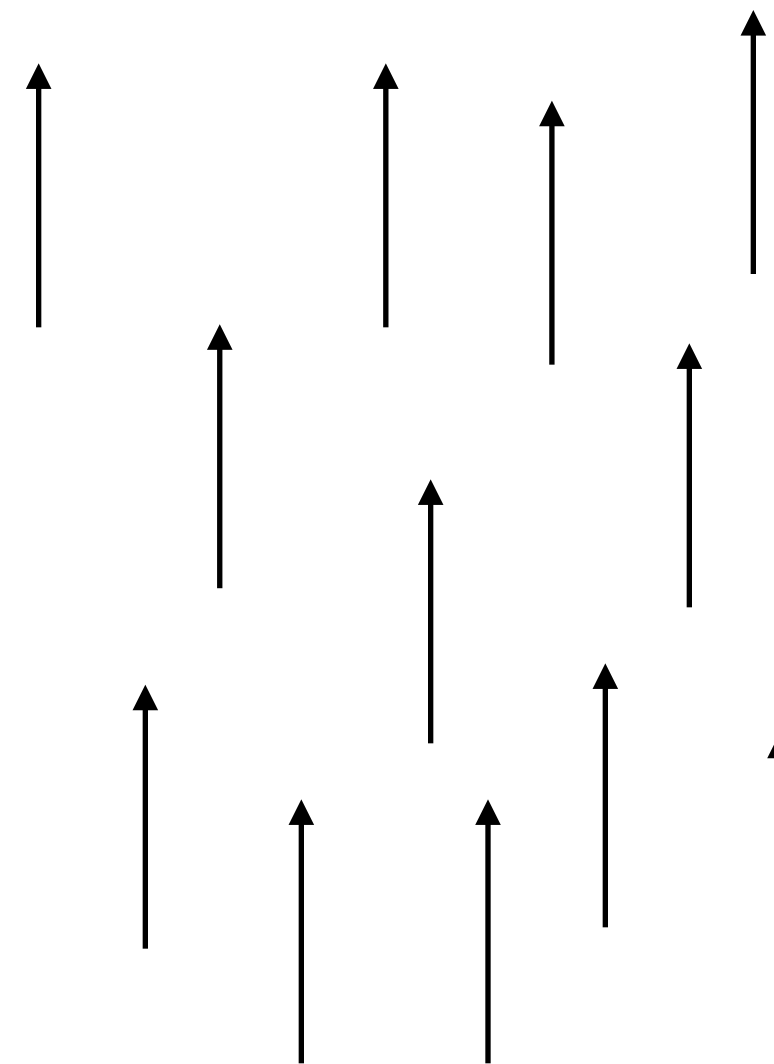
(from Carlos)

# Marc Shapiro

FIGURE 1: **TIME-SPACE DIAGRAM OF AN EXECUTION WITH THREE NODES**

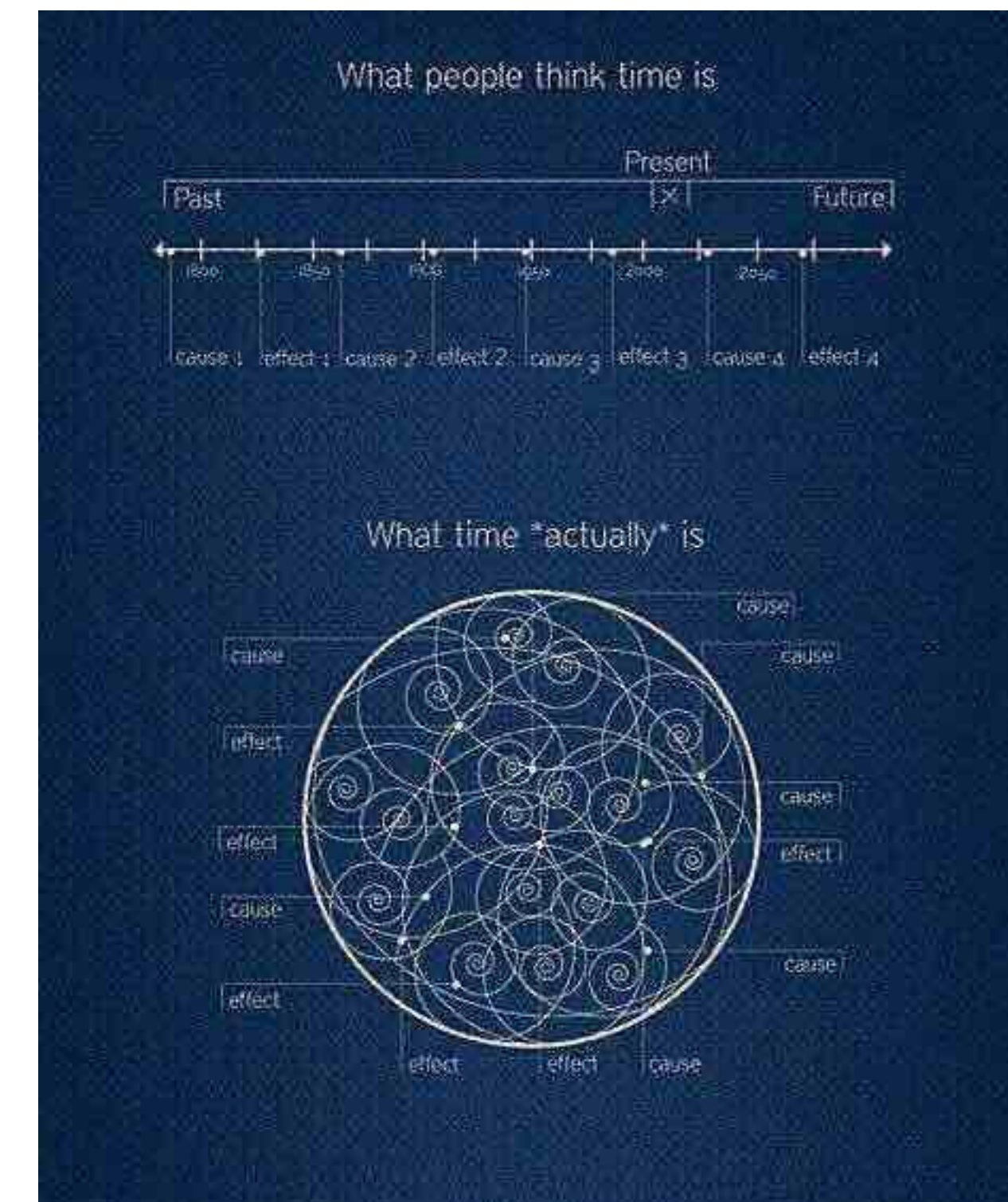


# Einstein



# Newton

Also, Herlihy & Shavit



# Carlos Bacquero



What people think time is



What time \*actually\* is







**Logical Clocks are easy  
Sometimes all you need  
is the right language**

# The trouble with timestamps





**Master Slave is a  
Solution**

**Oh wait...**

**Immutability  
changes everything**

**Oh wait...**

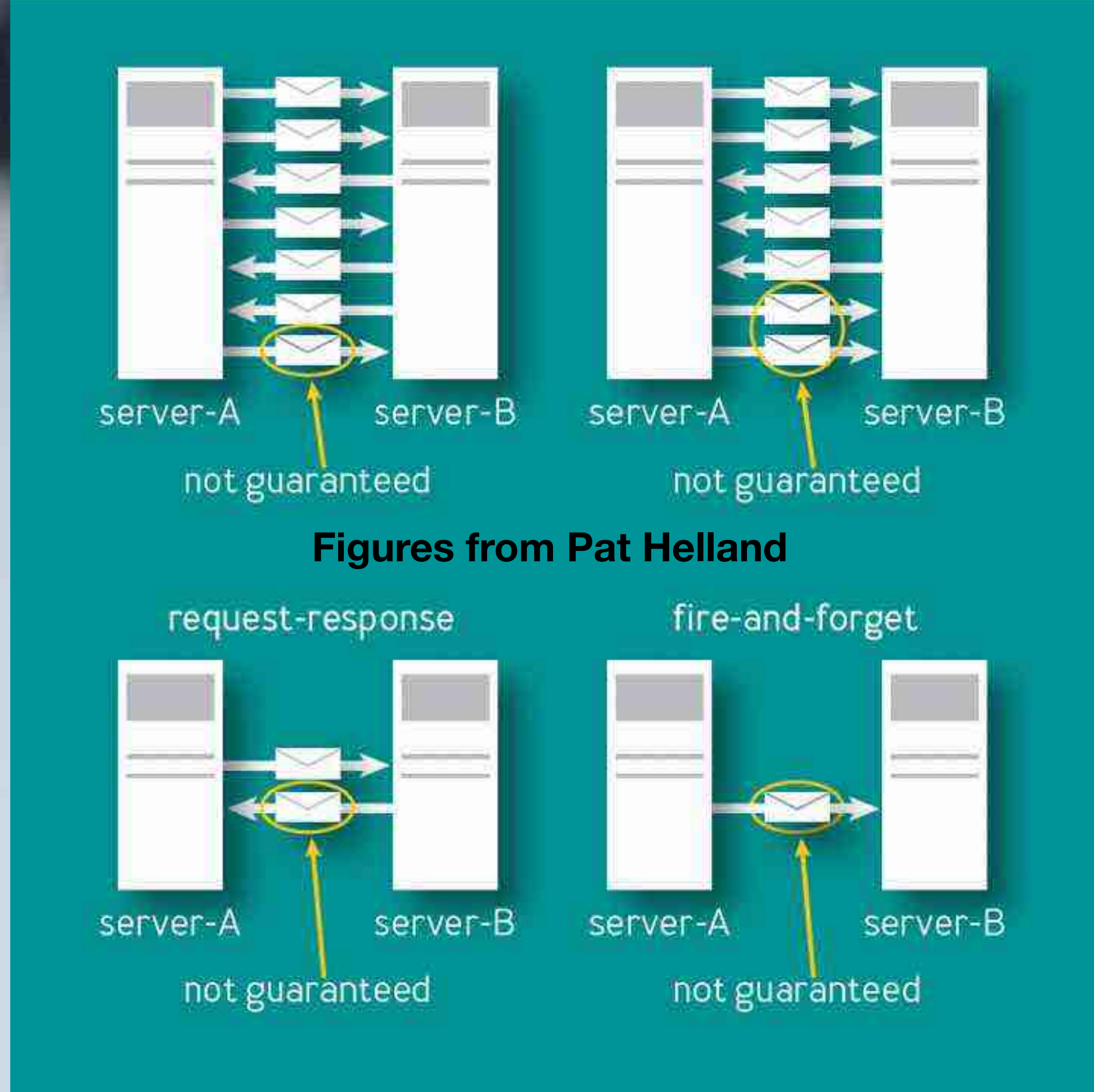
**Its all about Heisenberg**







Sean Cribbs

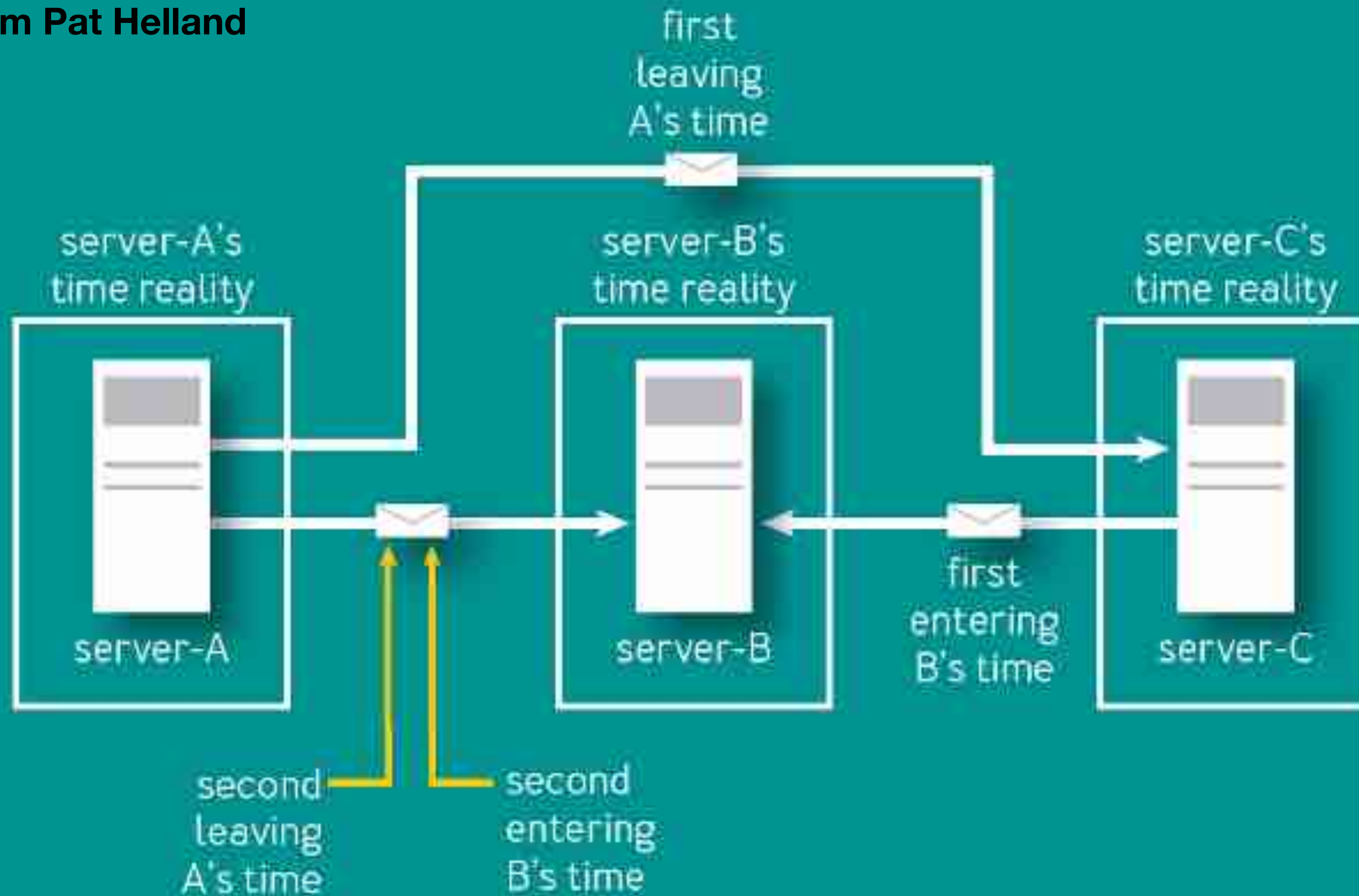


**Figures from Pat Helland**

**Darn it, that last write just got lost again!**



# From Pat Helland



# Disconnected Time May Be Slower or Faster Than Expected

Coordination Avoidance in Distributed Databases

By

Peter David Bailis

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

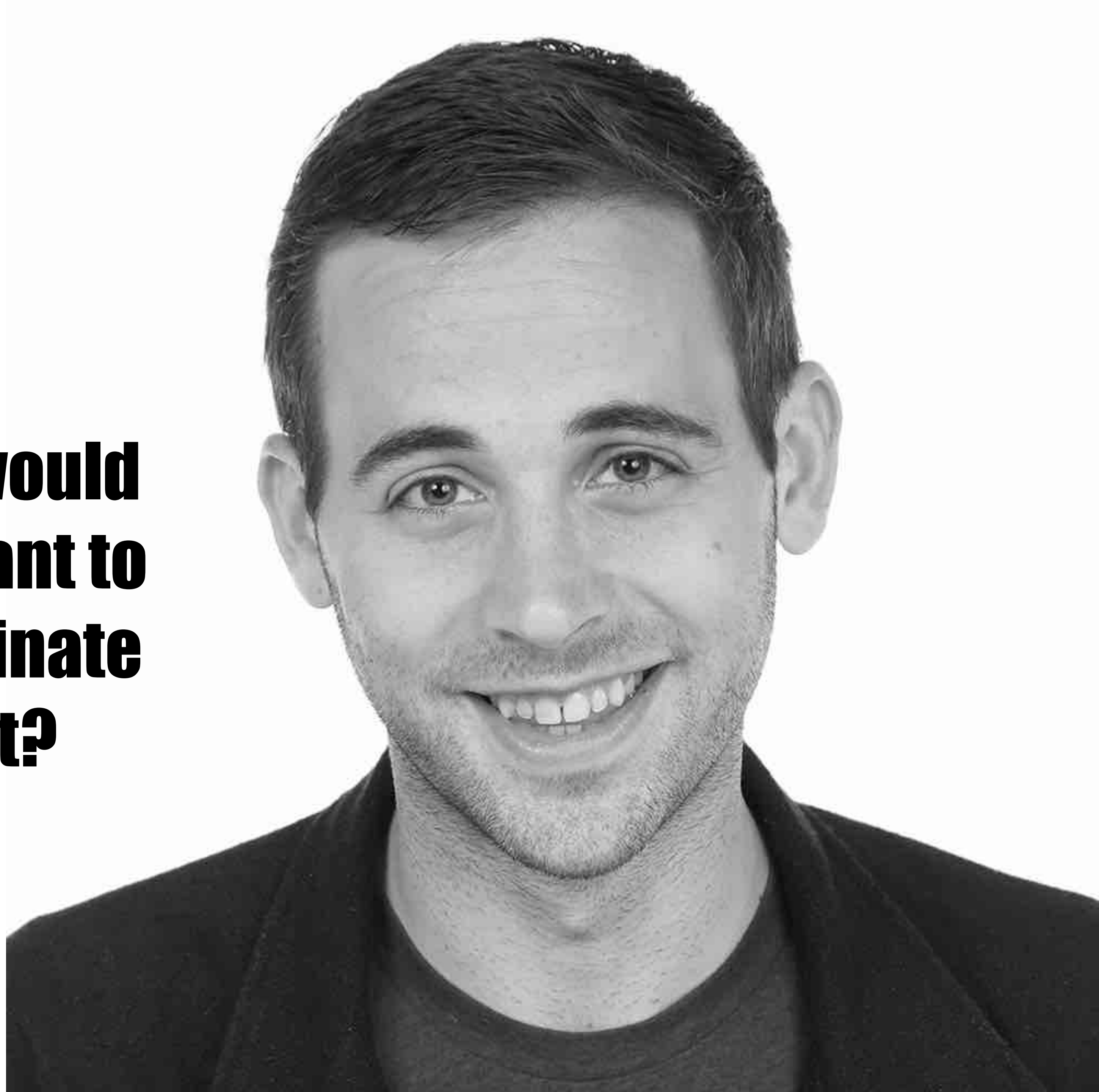
Professor Joseph M. Hellerstein, Co-Chair

Professor Ion Stoica, Co-Chair

Professor Ali Ghodsi

Professor Tapan Parikh

**Why would  
you want to  
coordinate  
that?**





Exceptions  
Hierarchical  
Tree  
Causality  
Dotted  
Version  
Plausible  
History  
Dot  
Matrix  
Time  
Last  
Stamp  
Time-Stamp  
Wins  
Dependency  
Logical  
Lamport  
Causal  
Interval  
Happens-Before

# Clock

## Vector



## Timestamps in Message-Passing Systems That Preserve the Partial Ordering

Colin J. Fidge  
 Department of Computer Science, Australian National University, Canberra, ACT.

### ABSTRACT

Timestamping is a common method of totally ordering events in concurrent programs. However, for applications requiring access to the global state, a total ordering is inappropriate. This paper presents algorithms for timestamping events in both synchronous and asynchronous message-passing programs that allow for access to the partial ordering inherent in a parallel system. The algorithms do not change the communications graph or require a central timestamp issuing authority.

**Keywords and phrases:** concurrent programming, message-passing, timestamps, logical clocks  
**CR categories:** D.1.3

### INTRODUCTION

A fundamental problem in concurrent programming is determining the order in which events in different processes occurred. An obvious solution is to attach a number representing the current time to a permanent record of the execution of each event. This assumes that each process can access an accurate clock, but practical parallel systems, by their very nature, make it difficult to ensure consistency among the processes.

There are two solutions to this problem. Firstly, have a central process to issue timestamps, i.e. provide the system with a global clock. In practice this has the major disadvantage of needing communication links from all processes to the central clock.

More acceptable are separate clocks in each process that are kept synchronised as much as necessary to ensure that the timestamps represent, at the very least, a *possible* ordering of events (in light of the vagaries of distributed scheduling). Lamport (1978) describes just such a scheme of logical clocks that can be used to totally order events, without the need to introduce extra communication links.

However this only yields one of the many possible, and equally valid, event orderings defined by a particular distributed computation. For problems concerned with the global program state it is far more useful to have access to the entire *partial* ordering, which defines the set of consistent “slices” of the global state at any arbitrary moment in time.

This paper presents an implementation of the partially ordered relation “happened before” that is true for two given events iff the first could causally affect the second in all possible interleavings of events. This allows access to *all* possible global states for a particular distributed computation, rather than a single, arbitrarily selected ordering. Lamport’s totally ordered relation is used as a starting point. The algorithm is first defined for the asynchronous case, and then extended to cater for concurrent programs using synchronous message-passing.

## Virtual Time and Global States of Distributed Systems \*

Friedemann Mattern †

Department of Computer Science, University of Kaiserslautern  
 D 6750 Kaiserslautern, Germany

### Abstract

*A distributed system can be characterized by the fact that the global state is distributed and that a common time base does not exist. However, the notion of time is an important concept in every day life of our decentralized “real world” and helps to solve problems like getting a consistent population census or determining the potential causality between events. We argue that a linearly ordered structure of time is not (always) adequate for distributed systems and propose a generalized non-standard model of time which consists of vectors of clocks. These clock-vectors are partially ordered and form a lattice. By using timestamps and a simple clock update mechanism the structure of causality is represented in an isomorphic way. The new model of time has a close analogy to Minkowski’s relativistic space-time and leads among others to an interesting characterization of the global state problem. Finally, we present a new algorithm to compute a consistent global snapshot of a distributed system where messages may be received out of order.*

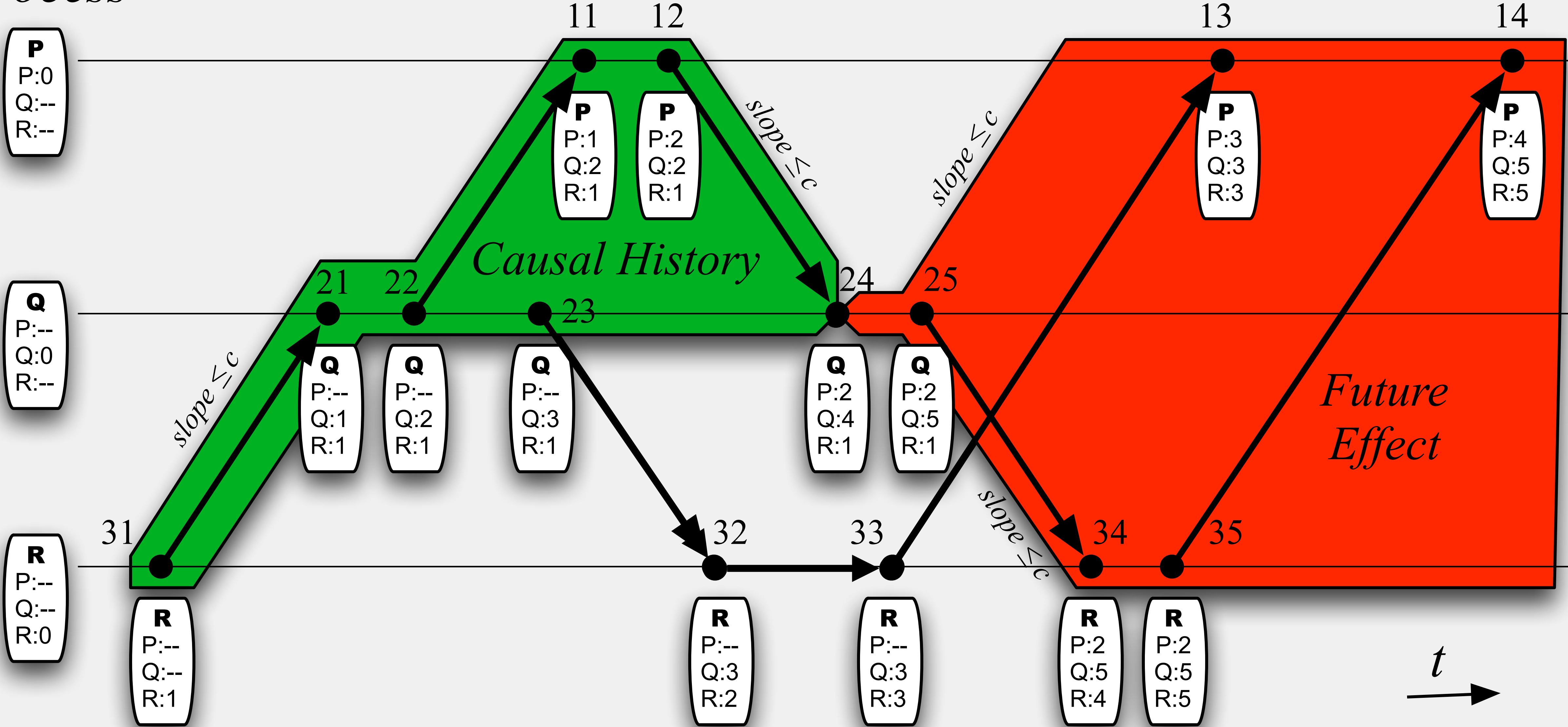
view of an idealized external observer having immediate access to all processes.

The fact that *a priori* no process has a consistent view of the global state and a common time base does not exist is the cause for most typical problems of distributed systems. Control tasks of operating systems and database systems like *mutual exclusion*, *deadlock detection*, and *concurrency control* are much more difficult to solve in a distributed environment than in a classical centralized environment, and a rather large number of distributed control algorithms for those problems has found to be wrong. *New problems* which do not exist in centralized systems or in parallel systems with common memory also emerge in distributed systems. Among the most prominent of these problems are *distributed agreement*, *distributed termination detection*, and the *symmetry breaking* or *election problem*. The great diversity of the solutions to these problems—some of them being really beautiful and elegant—is truly amazing and exemplifies many principles of distributed computing to cope with the absence of global state and time.

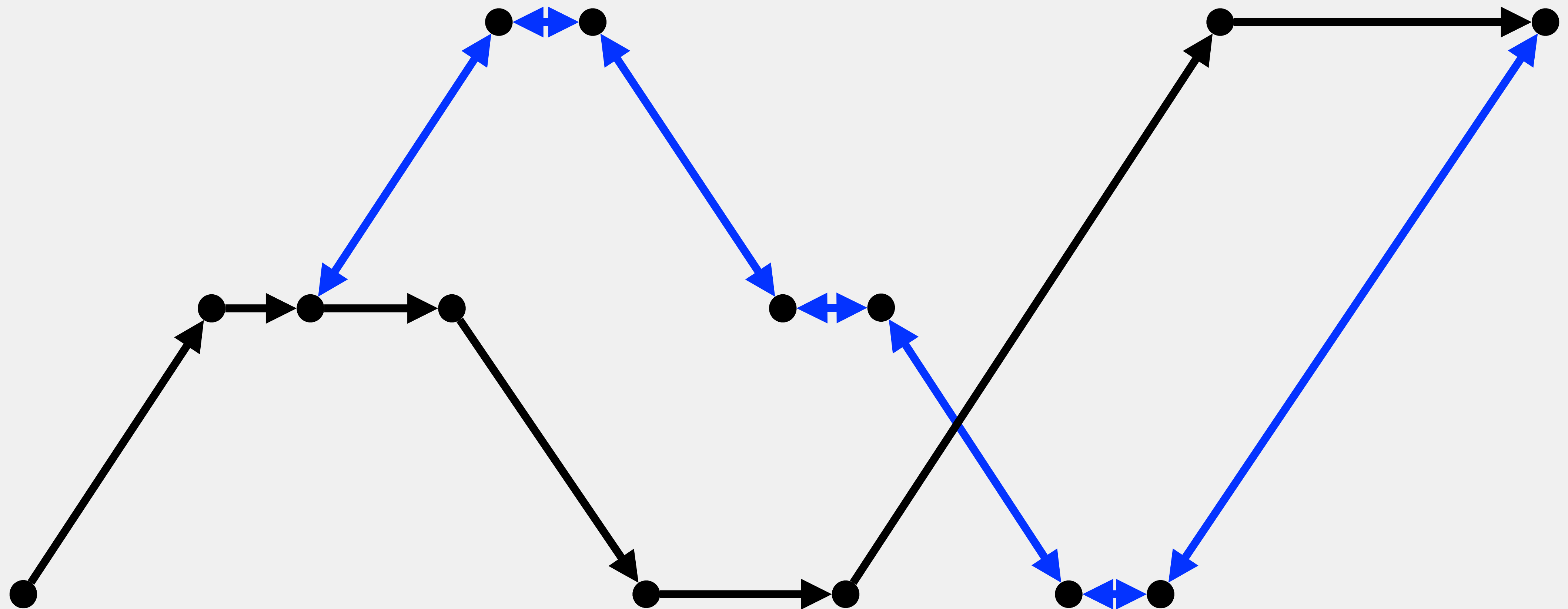
Since the design, verification, and analysis of algorithms for asynchronous systems is difficult and error-



*Process*







**Why Can't Links Be Reversible?**



'91

## Concerning the size of clocks

Bernadette Charron-Bost

I.N.T., 9 rue C. Fourier, 91011 Evry

L.R.I. Université Paris XI, 91405 Orsay

France

## 1 Introduction

Distributed systems with no known bounds on relative processor speed and transmission delay are called *asynchronous*. In such a system coordination and synchronization between processes are difficult to achieve. So the design and the proof of distributed algorithms for asynchronous systems are much subtle than for a classical centralized environment.

These difficulties vanish if the processes have a common time base, *i.e.* have access to perfectly synchronized clocks. But in asynchronous systems, such common clock cannot be achieved.

In [8] Lamport shows how to simulate a global clock by a clock that just captures causality. Such clocks are called *logical clocks* and are sufficient for instance to solve the mutual exclusion problem or for achieving a snapshot.

However with a logical clock we loose some informations about the causality relation which are crucial for implementing causal ordering (*cf.* [4]), debugging distributed systems (*cf.* [5]) or for assessing concurrency (*cf.* [1]). In [6] and [9] Fidge and Mattern independently improve Lamport's virtual time with a clock that entirely reflects the partial order defined by the causality relation. The dates assigned to the events are vectors of  $\mathbb{R}^n$  where  $n$  is the number of processes and the use of such vectors may seem very heavy as soon as one is concerned with a distributed system on a large number of processes.

In this paper by constructing an appropriate distributed computation we prove that smaller clocks do not work if one wants to characterize causality. Then we use classical theorems of the theory of partially ordered sets to give a mathematical interpretation of this result.

'94

### Detecting Causal Relationships in Distributed Computations: In Search of the Holy Grail

Reinhard Schwarz

Department of Computer Science, University of Kaiserslautern,

P.O. Box 3049, D - 67653 Kaiserslautern, Germany

schwarz@informatik.uni-kl.de

Friedemann Mattern

Department of Computer Science, University of Saarland

Im Stadtwald 36, D - 66041 Saarbrücken, Germany

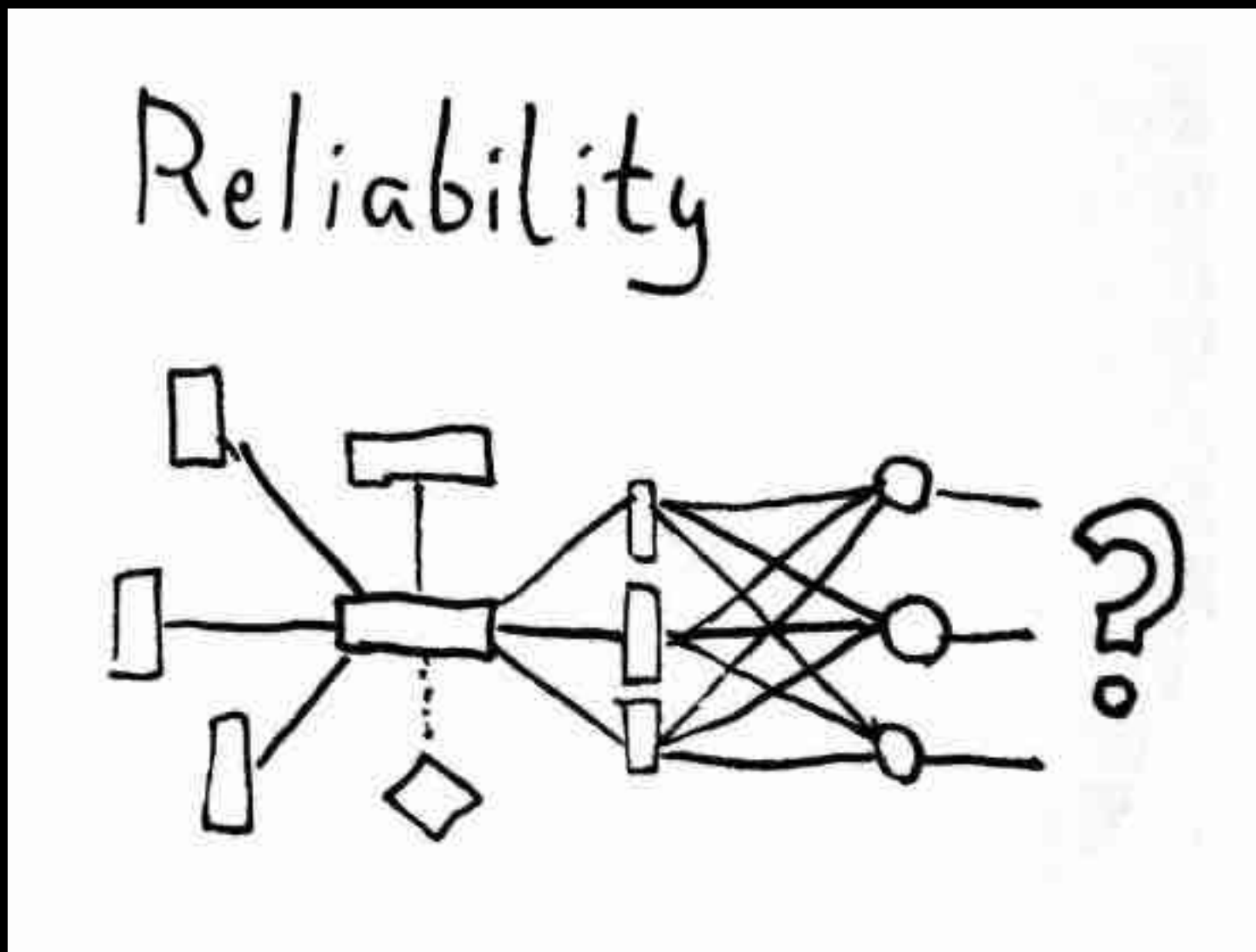
mattern@cs.uni-sb.de

**Abstract:** The paper shows that characterizing the causal relationship between significant events is an important but non-trivial aspect for understanding the behavior of distributed programs. An introduction to the notion of causality and its relation to logical time is given; some fundamental results concerning the characterization of causality are presented. Recent work on the detection of causal relationships in distributed computations is surveyed. The issue of observing distributed computations in a causally consistent way and the basic problems of detecting global predicates are discussed. To illustrate the major difficulties, some typical monitoring and debugging approaches are assessed, and it is demonstrated how their feasibility is severely limited by the fundamental problem to master the complexity of causal relationships.

**Keywords:** Distributed Computation, Causality, Distributed System, Causal Ordering, Logical Time, Vector Time, Global Predicate Detection, Distributed Debugging, Timestamps



**So your packets can be  
dropped, delayed,  
duplicated, reordered  
or just plain f\*\*ked**





# Summary

- Defined “**happened before**” *relation*: a **partial order**
- Defined “**logical timestamps**” which forms an *arbitrary total order*, restricting the available concurrency of a system (i.e. algorithm proceeds no faster than a single thread execution)
- This “concurrency efficiency loss” gets worse as:
  - We add more nodes to a distributed system
  - These nodes become more spatially separated
  - Our processors and networks get faster
  - Our processors are comprised of more cores

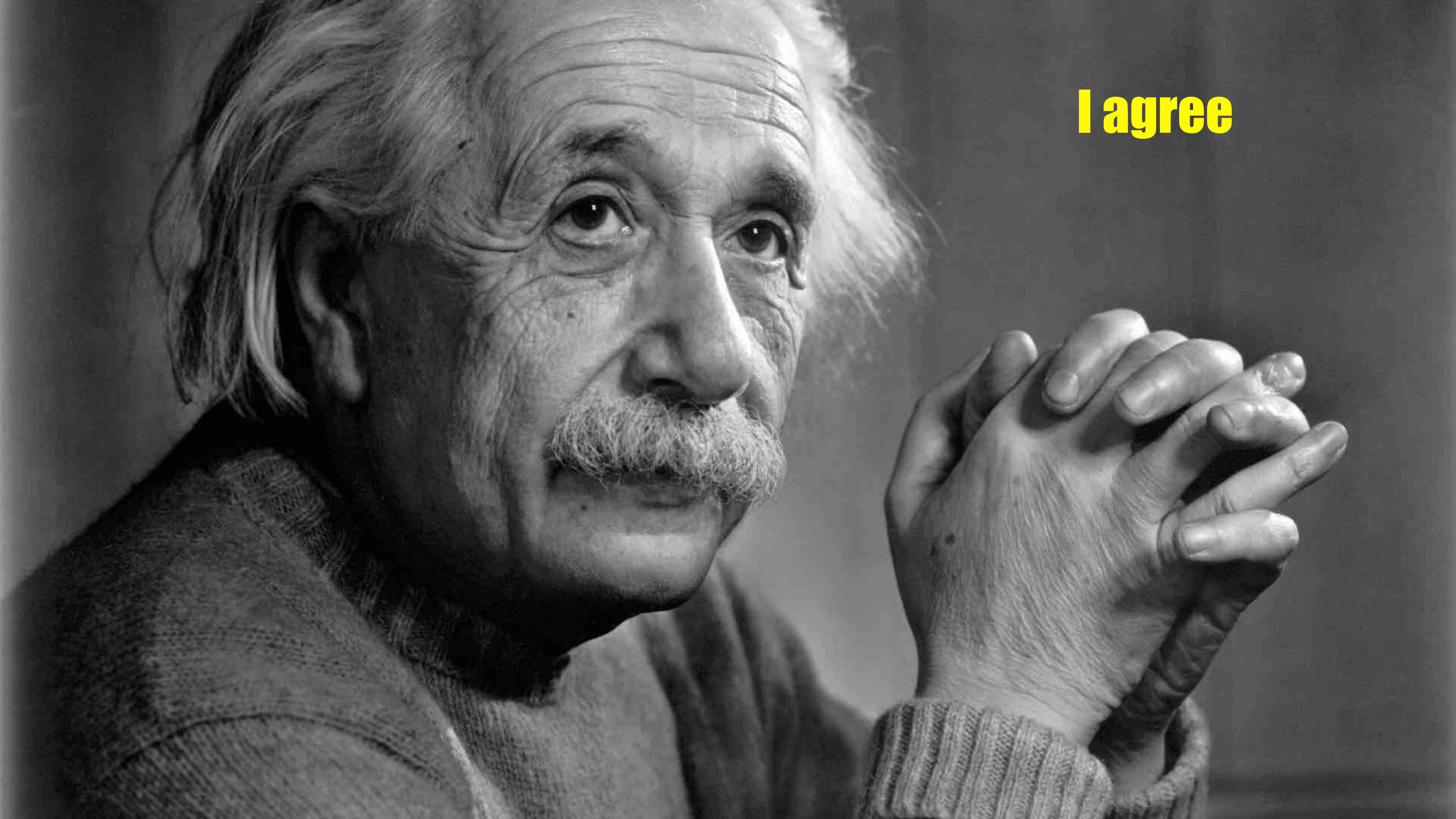




**There is no now**



**I agree**





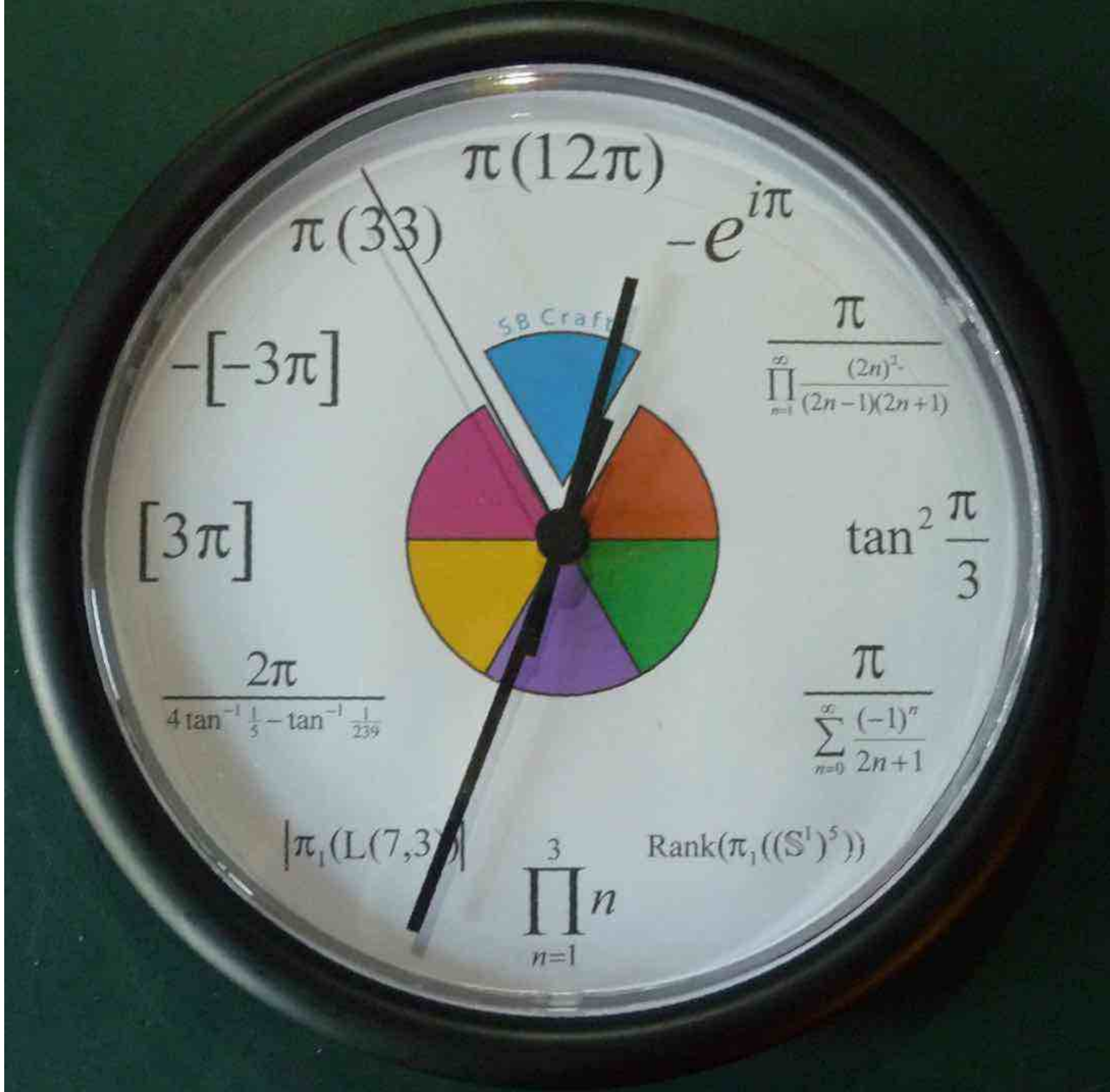
# Simultaneity is a Myth

“A circular argument:

*To determine the simultaneity of distant events we need to know a velocity, and to measure a velocity we require knowledge of the simultaneity of distant events” \**

\*Quoting Reichenbach, in: “Concepts of Simultaneity. From Antiquity to Einstein and Beyond.” Max Jammer( 2006)







2022

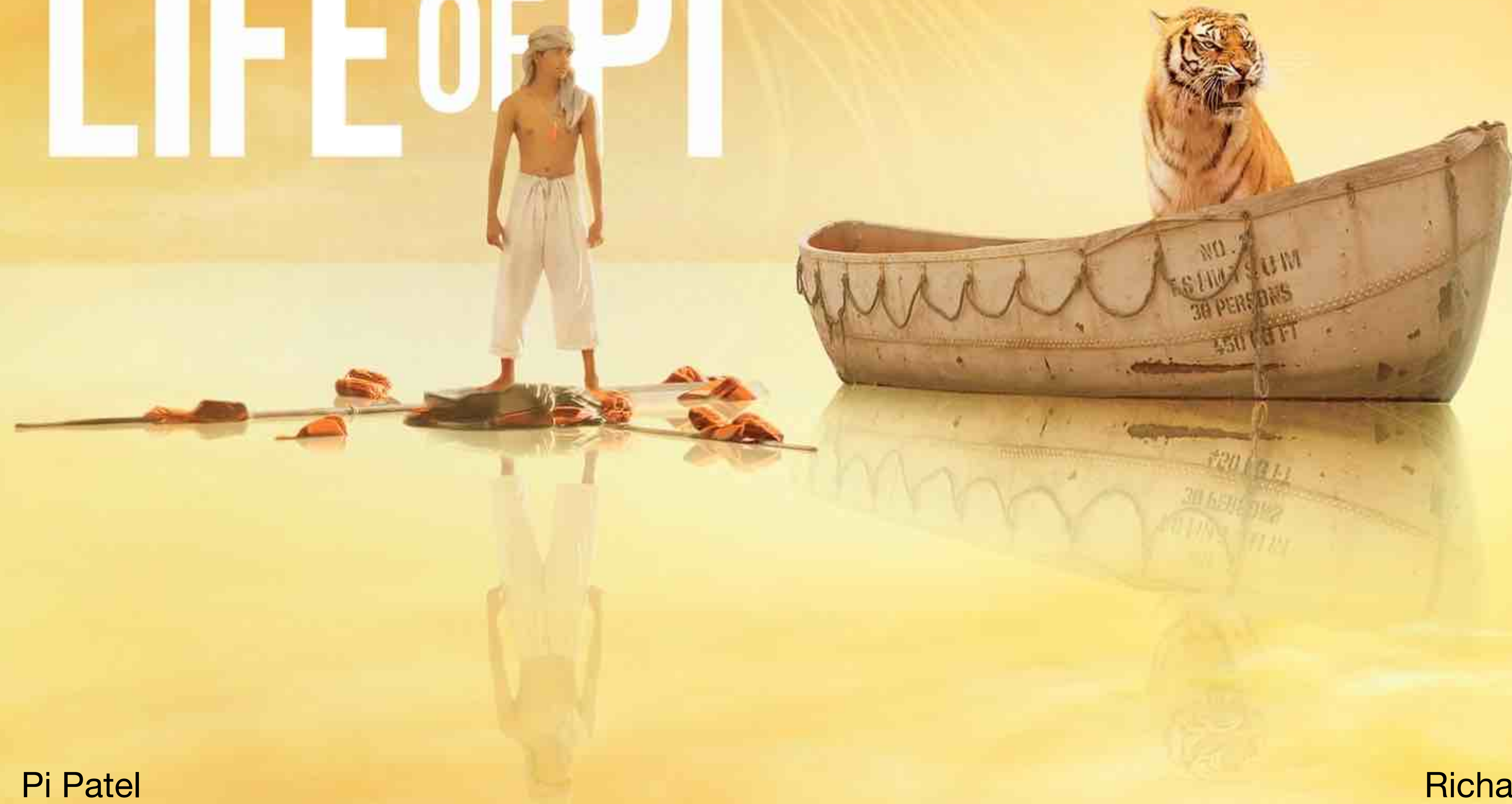


$$e^{i\pi} + 1 = 0$$

**Euler's Identity**



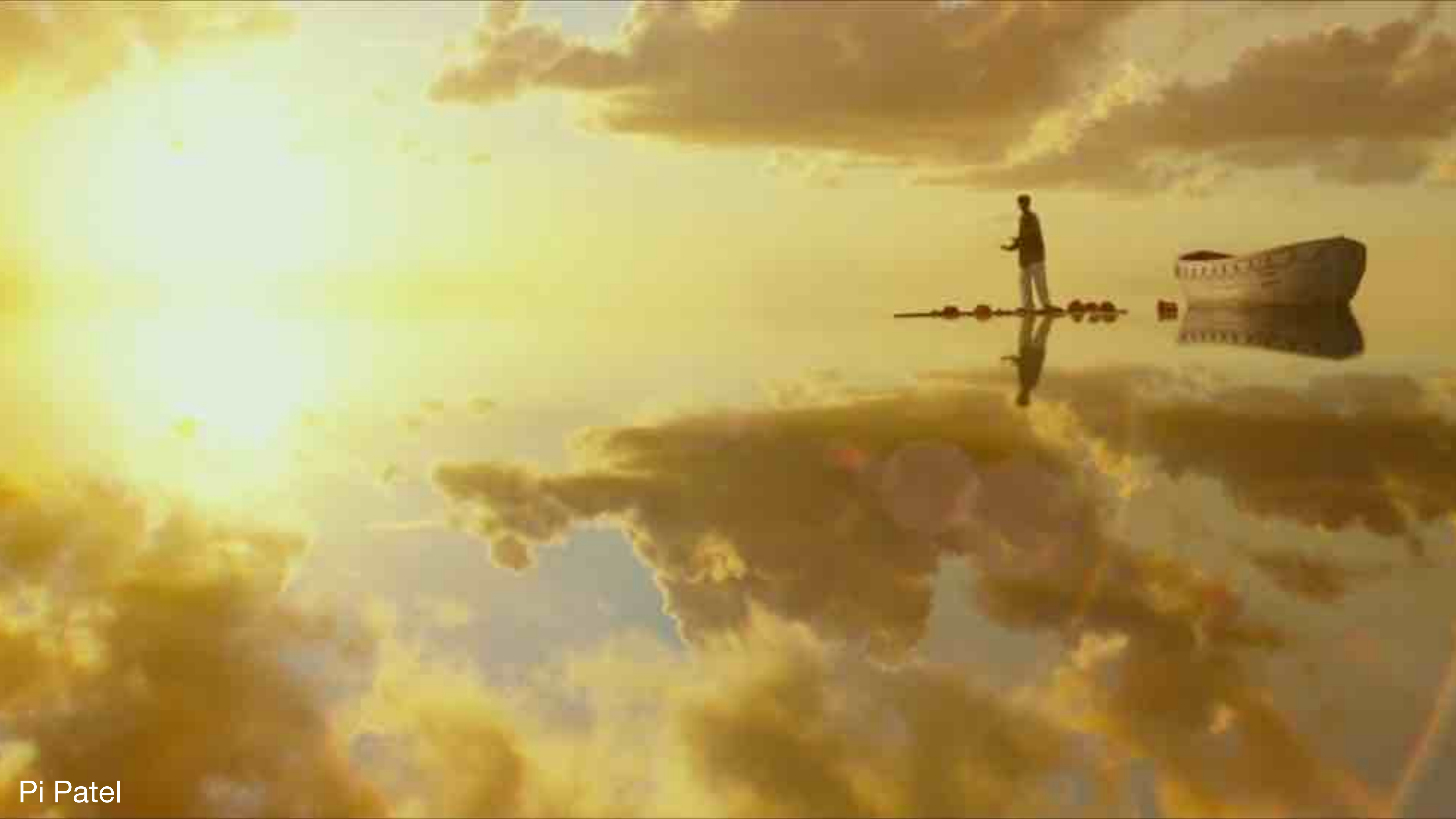
# LIFE of PI



Pi Patel

Richard Parker







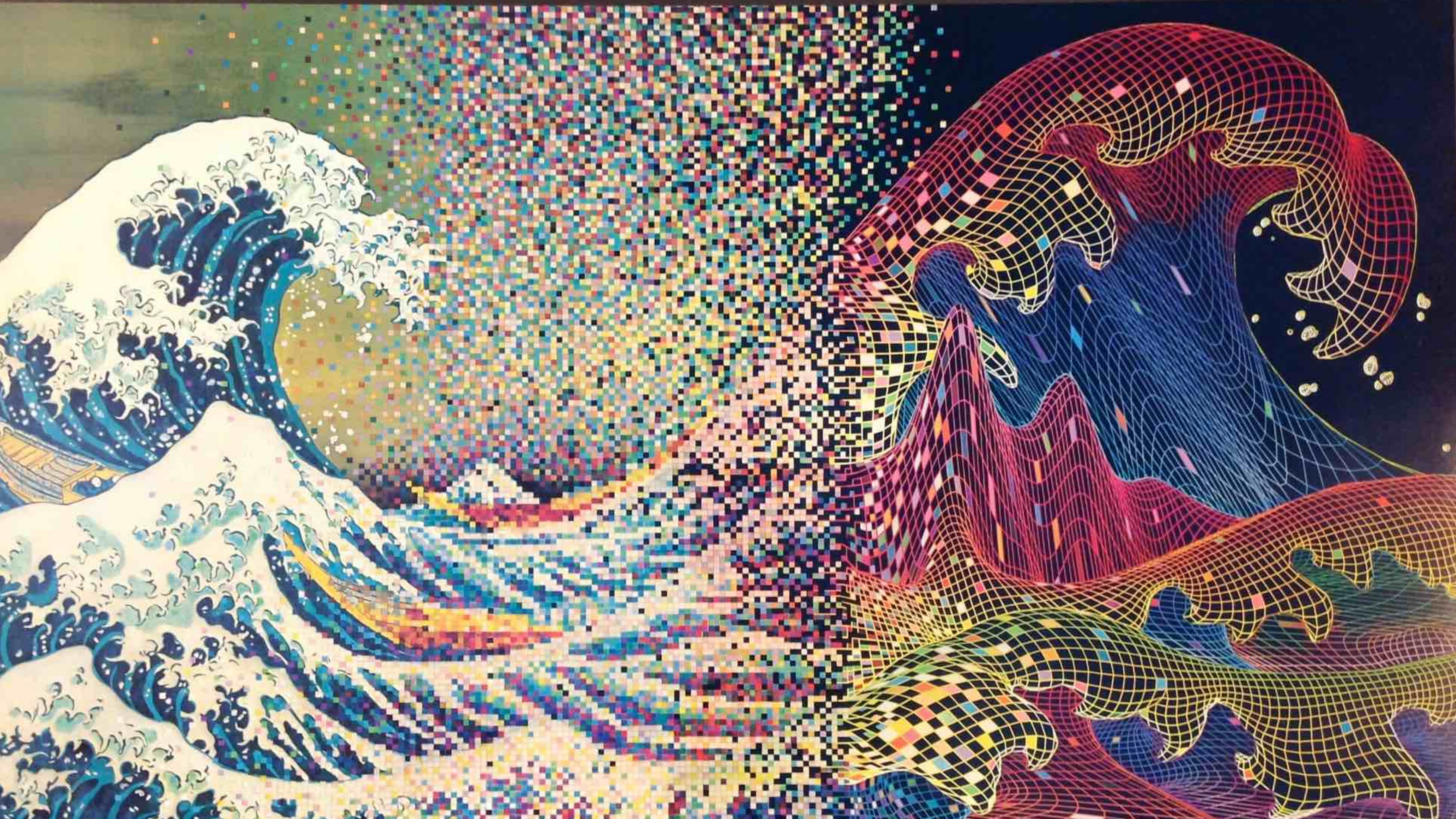


A smooth background of spacetime?













Richard Parker



# What is Time?

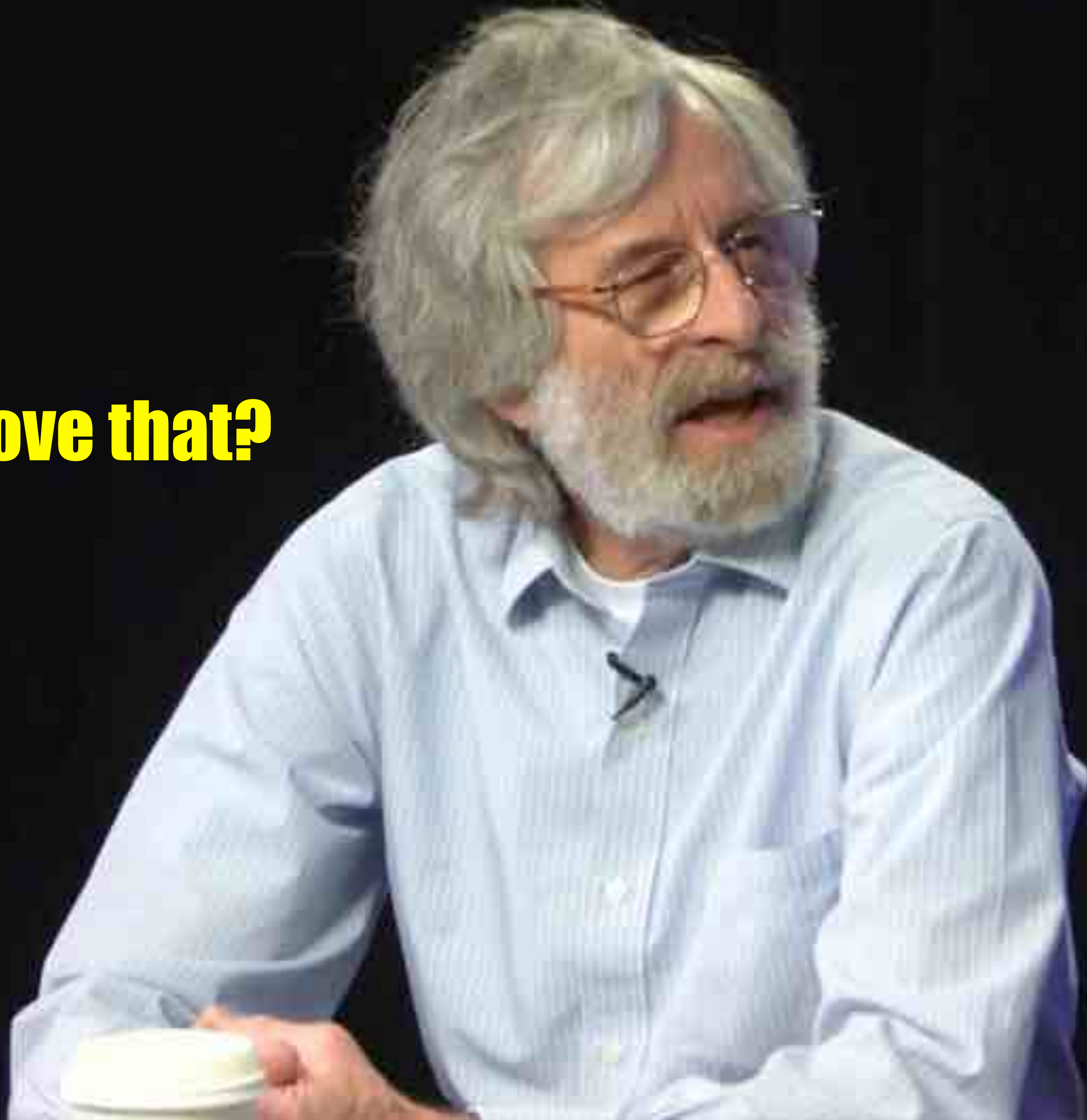
- *Time is change that we can count*
- *All change is part of a tree; pick your root*
- *Entanglements are roots of irreversible change*
- *Anything that can happen can unhappen*
- *Messages that can be sent can be unsent*



**ER=EPR**



**Can you prove that?**



The man himself ...



### III.5 ON THE EINSTEIN PODOLSKY ROSEN PARADOX\*

JOHN S. BELL†

#### I. Introduction

THE paradox of Einstein, Podolsky and Rosen [1] was advanced as an argument that quantum mechanics could not be a complete theory but should be supplemented by additional variables. These additional variables were to restore to the theory causality and locality [2]. In this note that idea will be formulated mathematically and shown to be incompatible with the statistical predictions of quantum mechanics. It is the requirement of locality, or more precisely that the result of a measurement on one system be unaffected by operations on a distant system with which it has interacted in the past, that creates the essential difficulty. There have been attempts [3] to show that even without such a separability or locality requirement no “hidden variable” interpretation of quantum mechanics is possible. These attempts have been examined elsewhere [4] and found wanting. Moreover, a hidden variable interpretation of elementary quantum theory [5] has been explicitly constructed. That particular interpretation has indeed a grossly non-local structure. This is characteristic, according to the result to be proved here, of any such theory which reproduces exactly the quantum mechanical predictions.

#### II. Formulation

With the example advocated by Bohm and Aharonov [6], the EPR argument is the following. Consider a pair of spin one-half particles formed somehow in the singlet spin state and moving freely in opposite directions. Measurements can be made, say by Stern-Gerlach magnets, on selected components of the spins  $\vec{\sigma}_1$  and  $\vec{\sigma}_2$ . If measurement of the component  $\vec{\sigma}_1 \cdot \vec{a}$ , where  $\vec{a}$  is some unit vector, yields the value  $+1$  then, according to quantum mechanics, measurement of  $\vec{\sigma}_2 \cdot \vec{a}$  must yield the value  $-1$  and vice versa. Now we make the hypothesis [2], and it seems one at least worth considering, that if the two measurements are made at places remote from one another the orientation of one magnet does not influence the result obtained with the other. Since we can predict in advance the result of measuring any chosen component of  $\vec{\sigma}_2$ , by previously measuring the same component of  $\vec{\sigma}_1$ , it follows that the result of any such measurement must actually be predetermined. Since the initial quantum mechanical wave function does *not* determine the result of an individual measurement, this predetermination implies the possibility of a more complete specification of the state.

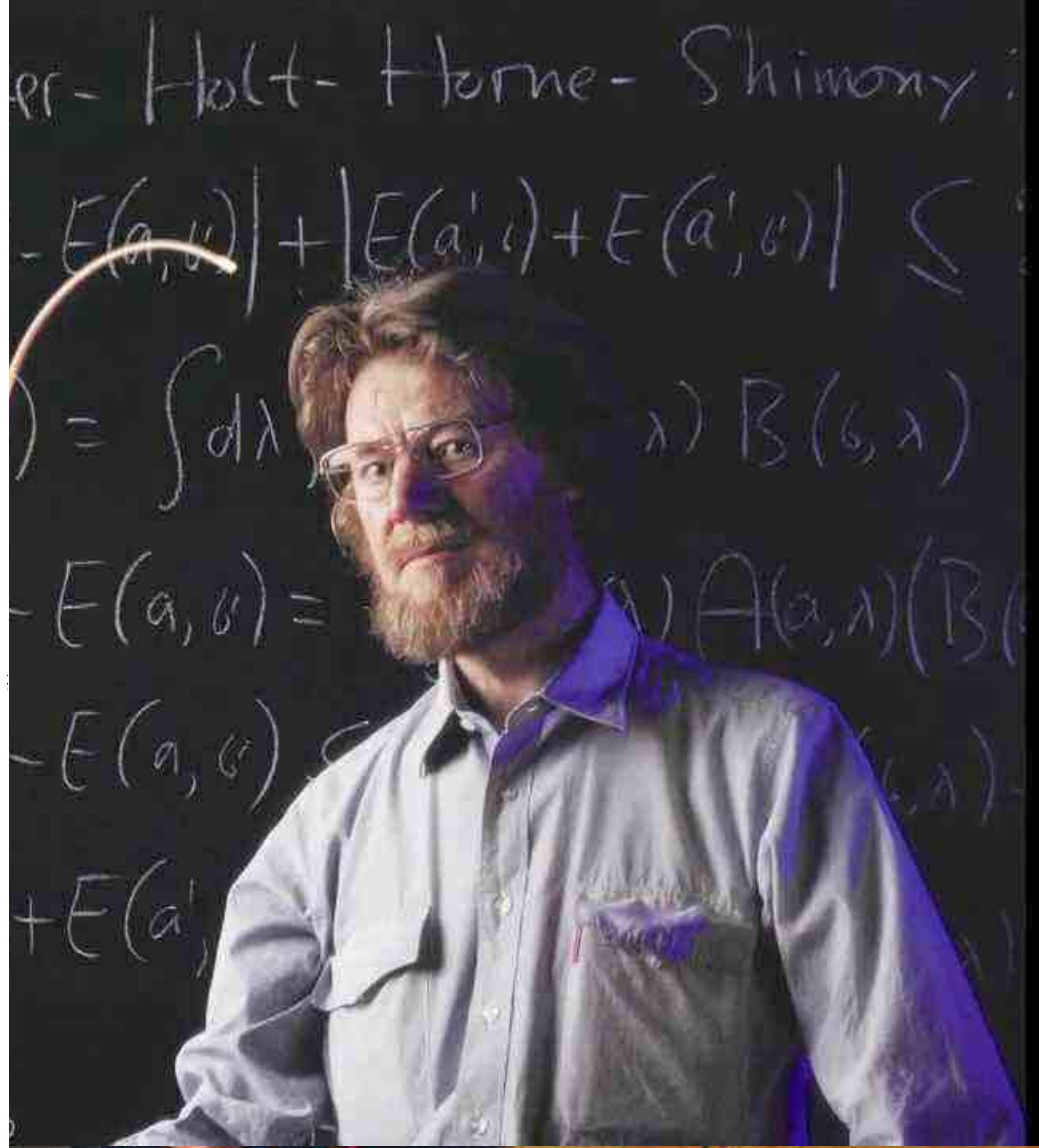
Let this more complete specification be effected by means of parameters  $\lambda$ . It is a matter of indifference in the following whether  $\lambda$  denotes a single variable or a set, or even a set of functions, and whether the variables are discrete or continuous. However, we write as if  $\lambda$  were a single continuous parameter. The result  $A$  of measuring  $\vec{\sigma}_1 \cdot \vec{a}$  is then determined by  $\vec{a}$  and  $\lambda$ , and the result  $B$  of measuring  $\vec{\sigma}_2 \cdot \vec{b}$  in the same instance is determined by  $\vec{b}$  and  $\lambda$ , and

\*Work supported in part by the U.S. Atomic Energy Commission

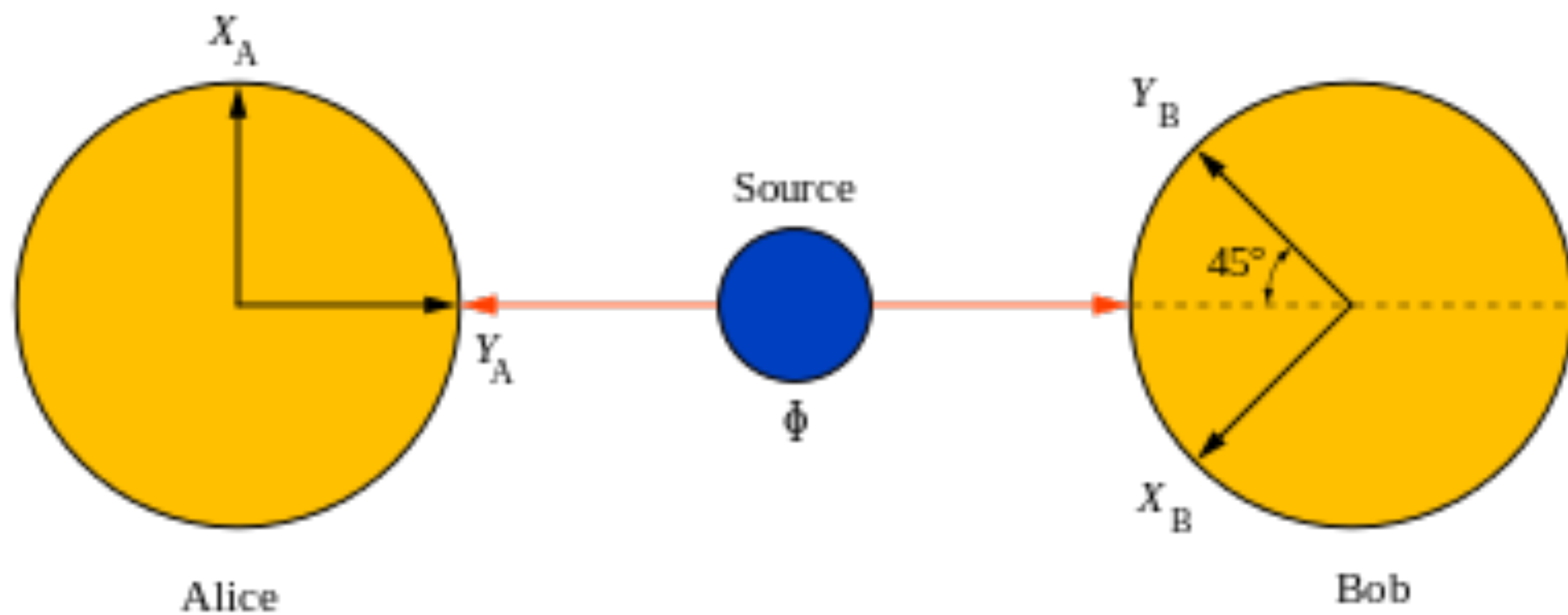
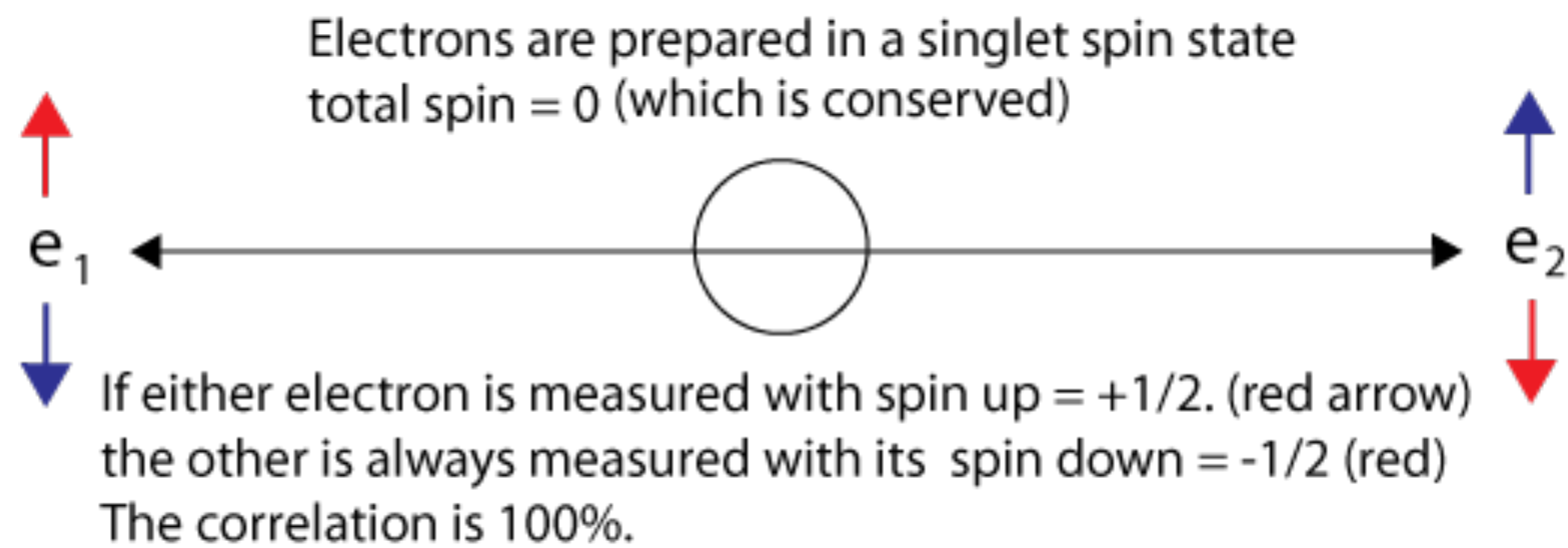
†On leave of absence from SLAC and CERN

Originally published in *Physics*, 1, 195-200 (1964).

John S. Bell

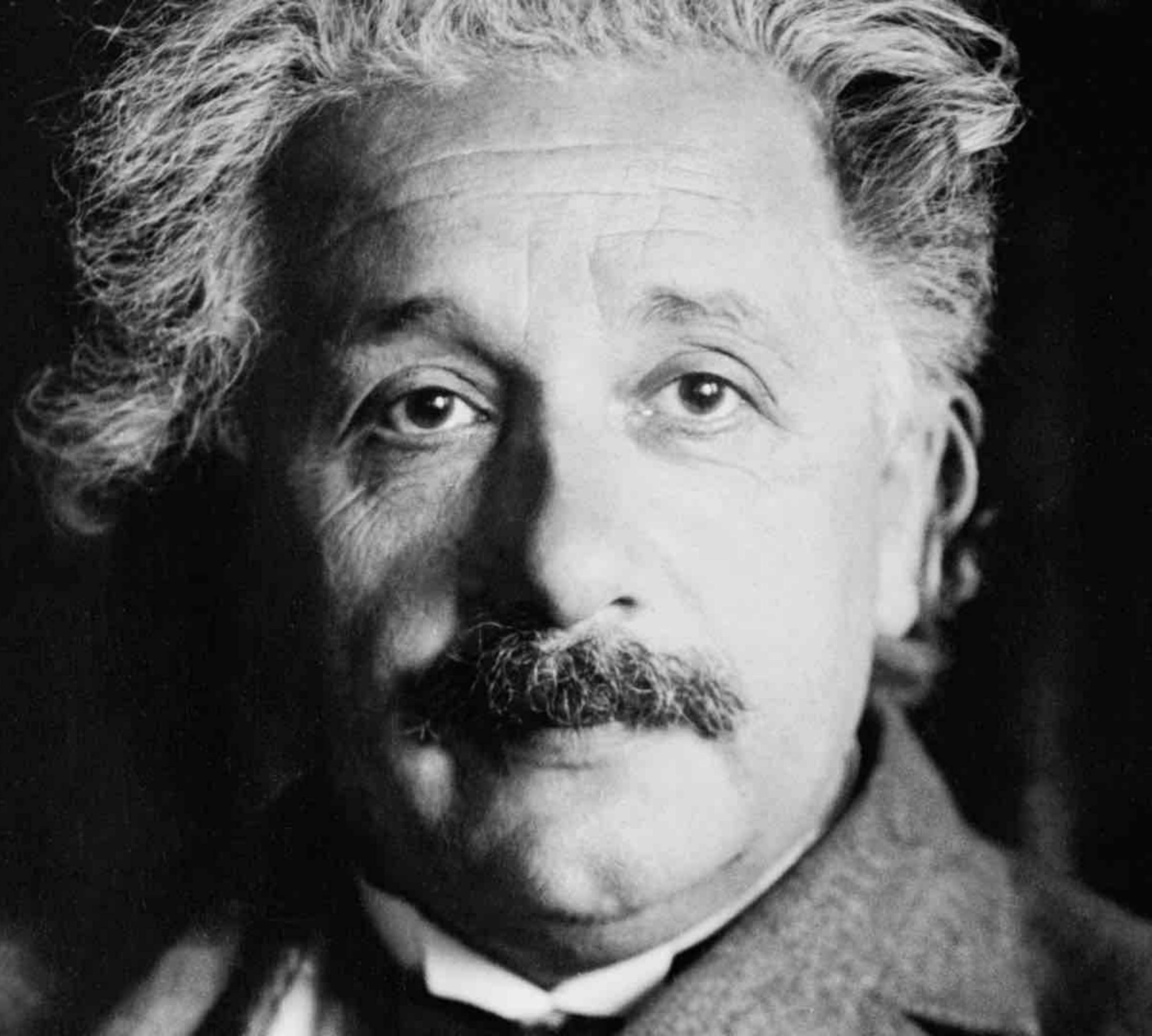






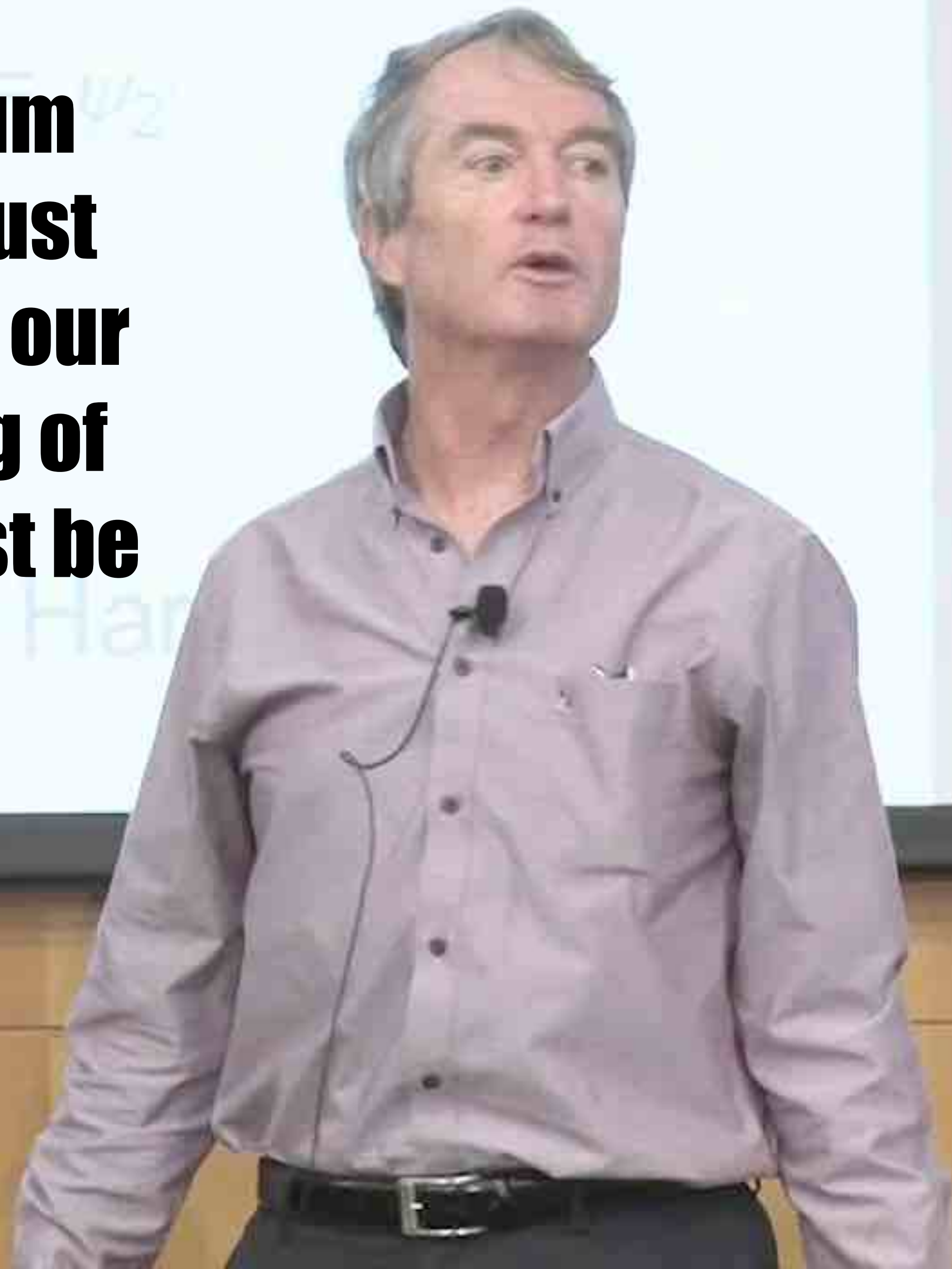


**“We have to bear in mind that all our propositions involving time are always propositions about simultaneous events”**



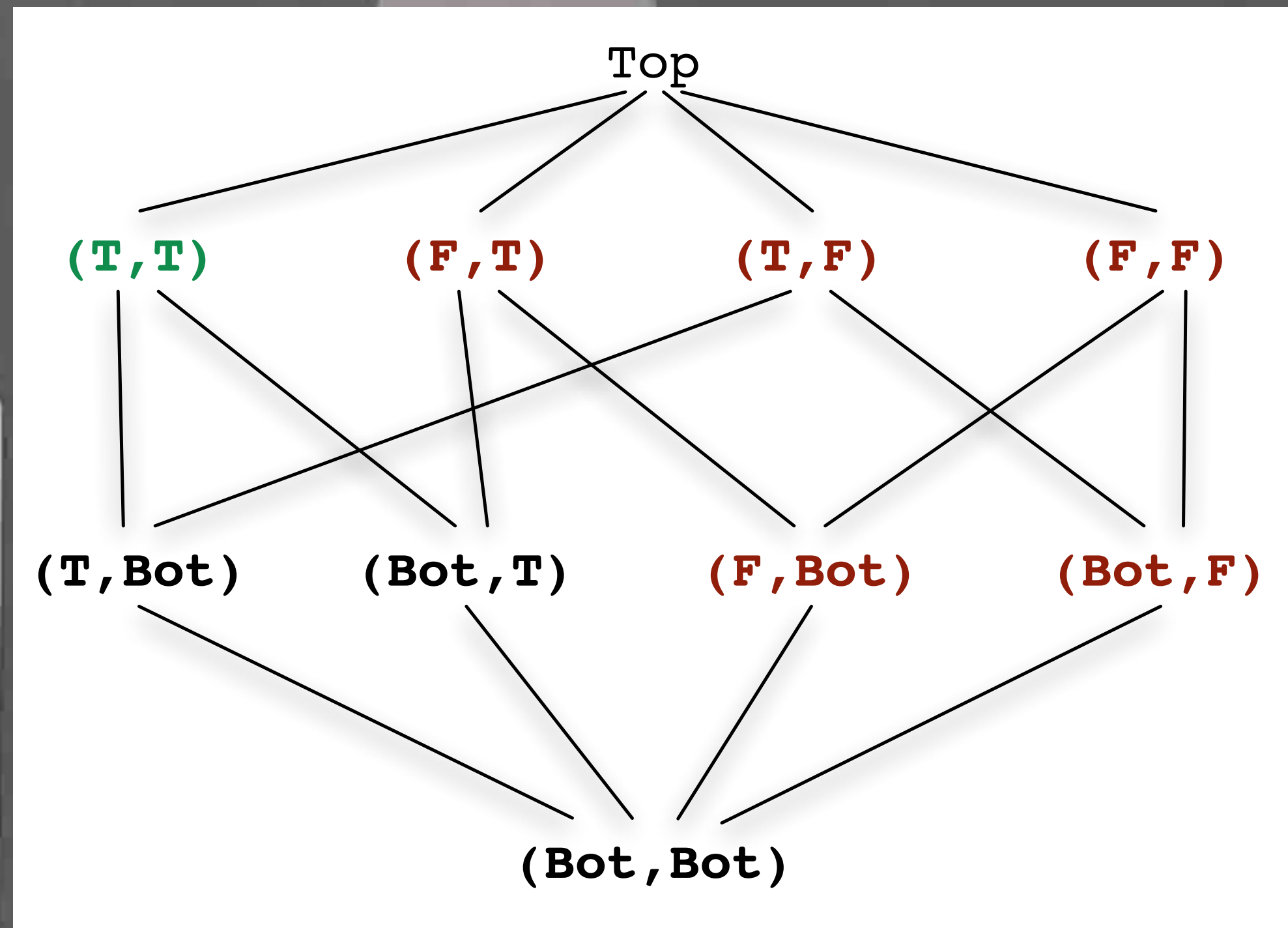


**either quantum  
mechanics must  
break down, or our  
understanding of  
spacetime must be  
wrong**





# What about lattice variables



to capture causality?



**Ta da!**

**Spacetime is built from  
entanglement**

Mark van Raamsdonk



# Order and causality expressed as lattices



## Lattice Example

### Lattice Example

```
{ok, ObjectSetStream} = derflow:declare(),
{ok, ObjectSetId} = derflow:declare(riak_dt_gset),
ObjectSetFun = fun(X) ->
    {ok, Set0, _} = derflow:read(ObjectSetId),
    {ok, Set} = riak_dt_gset:update({add, X},
        undefined, Set0),
    {ok, _} = derflow:bind(ObjectSetId, Set),
    Set
end,
derflow:thread(?MODULE,
    consumer,
    [ObjectStream,
    ObjectSetFun,
    ObjectSetStream]),
```



**Entanglement is  
transferable  
and it's  
universal**



**What I've learned about  
time is that we still don't  
fully understand it**



Max Tegmark



# Simultaneity is a Myth

Maurice Herlihy and Nir Shavit:

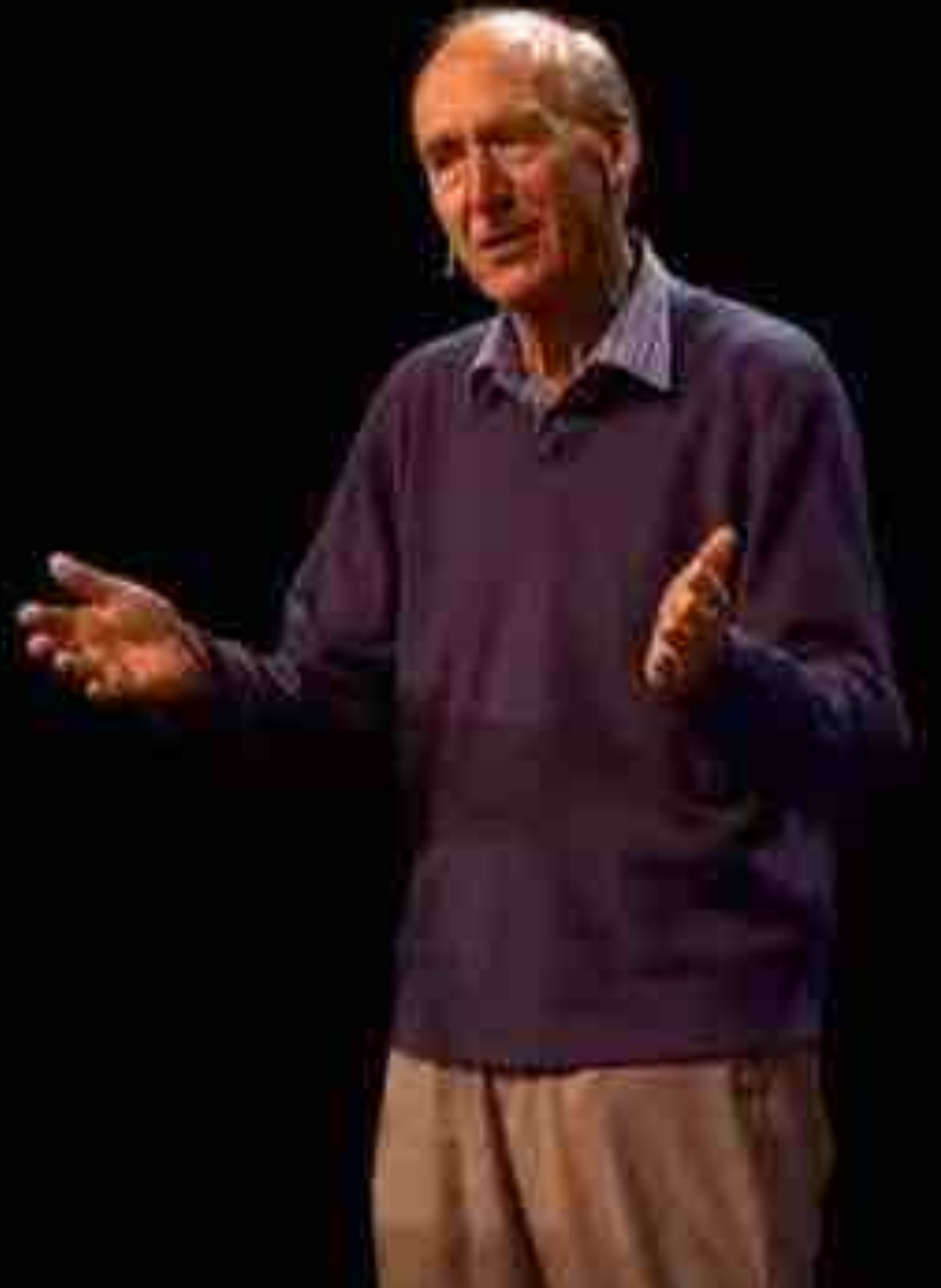
The Art of Multiprocessor Programming [2008]:

*"In 1689, Isaac Newton stated 'absolute, true and mathematical time, of itself and from its own nature, flows equably without relation to anything external.'" "We endorse his notion of time"*

*A notion of time proven incorrect over a hundred years ago ...*



**if nothing were to  
change we could not  
say that time passes**





**Ta da!**

**a subsystem of an entangled  
state works as a "clock" of  
another subsystem**





# The Arrow of Time Dilemma\*

- The laws of physics are invariant for time inversion. The phenomena we see everyday are not (entropy increases)
- Within a quantum mechanical framework, all phenomena which leave a trail of information behind (and hence can be studied by physics) are those where entropy necessarily increases or remains constant
- All phenomena where the entropy decreases must not leave any information of their having happened. This situation is completely indistinguishable from their not having happened at all
- The second law of thermodynamics is reduced to a tautology: physics cannot study those processes where entropy has decreased, even if they were commonplace– because the evidence has been erased

\*Lorenzo Maccone. "Quantum Solution to the Arrow-of-Time Dilemma." *Physical Review Letters* 103, no. 8 (2009)



**Shh ...  
don't tell that  
Schwinger fellow,  
but it's really all  
particles ...**





**Oh shit ...**



**That means time goes backwards for positrons ...**



# A Myth: Common Error

In reality, a distributed program runs on multiple nodes; with multiple CPUs and multiple streams of operations coming in. You can still assign a total order, but it requires either **accurate clocks** or some form of communication. You could timestamp each operation using a **completely accurate clock then use that to figure out the total order**. Or you might have some kind of communication system that makes it possible to assign sequential numbers as in a total order.

- *Not even wrong*
- *So what if you did it?*



# General Theory of Concurrency

Physicists and computer scientists are talking past each other when they talk about time

If we could resolve that we might make progress on a general theory of concurrency



# 8th Conference on Reversible Computation (RC)

July 7<sup>th</sup> -8<sup>th</sup> 2016, Bologna, Italy

[Home](#)
[Program](#)
[Registration](#)
[Call for Papers](#)
[Program Committee](#)
[Invited Talks](#)
[Location](#)
[Social Event](#)
[Accommodation](#)
[Steering Committee](#)
[Previous Editions](#)
[Contact](#)
[Home](#)

## Welcome to the 8th Conference on Reversible Computation

July 7<sup>th</sup>-8<sup>th</sup>, 2016, Bologna, Italy

The Conference on Reversible Computation will bring together researchers from computer science, mathematics, engineering, and physics to discuss new developments and directions for future research in the emerging area of Reversible Computation. This particularly includes applications of reversibility in quantum computation. Research papers, tutorials, tool demonstrations, and work-in-progress reports are within the scope of the conference.



### Important Dates:

Abstract Submission:  
Sun, February 7<sup>th</sup>, 2016

Submission Deadline:  
Sun, February 14<sup>th</sup>, 2016

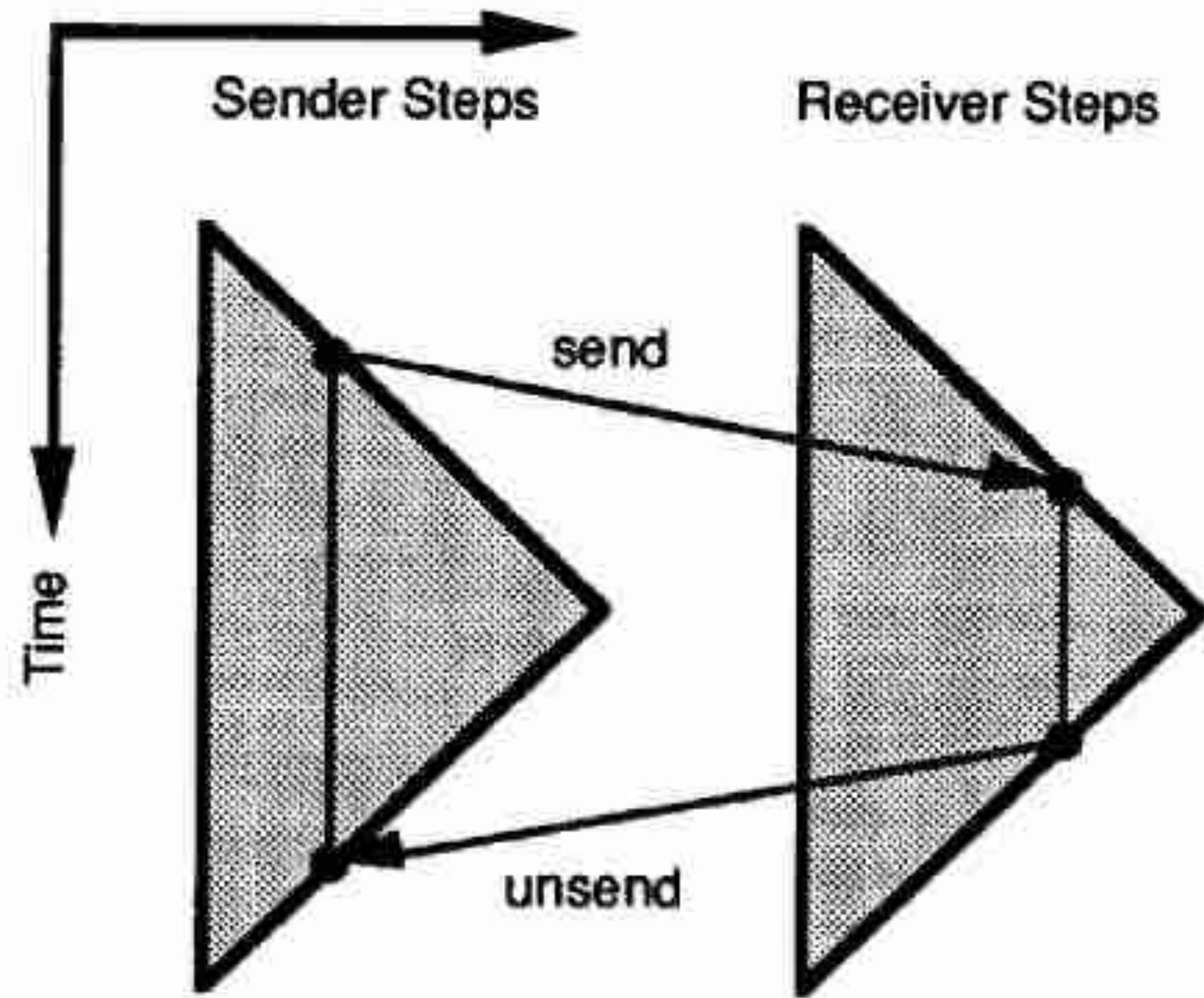
Notification to Authors:  
Mon, March 21<sup>st</sup>, 2016

Final Version:  
Sun, April 10<sup>th</sup>, 2016

Early Registration:  
Thu, June 16<sup>th</sup>, 2016

Conference:  
Thu-Fri, July 7<sup>th</sup> and 8<sup>th</sup>, 2016





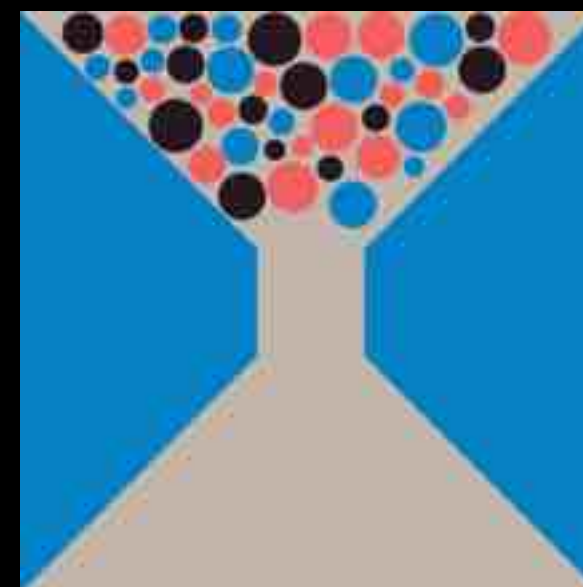
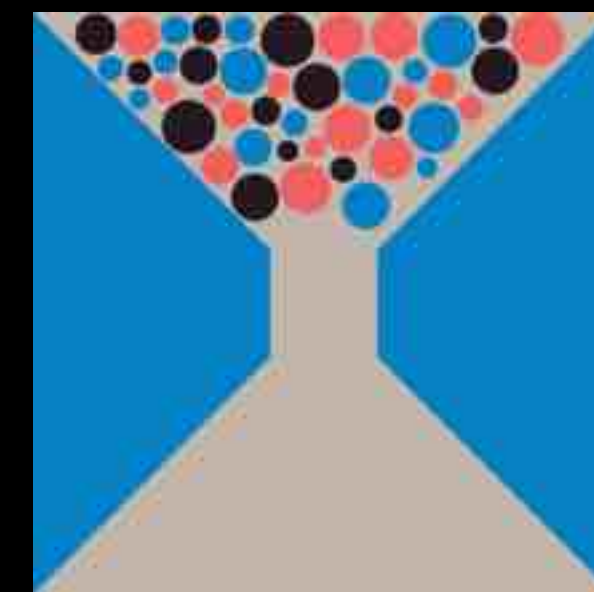
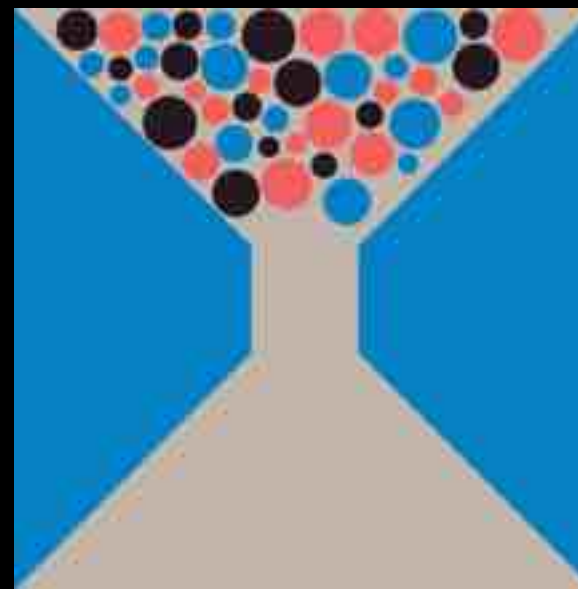
**Figure 3** When Cached, Talk Is Cheap

A computer's task is often taken to be that of starting with some input, grinding for a while, and eventually returning an output. Remarkably, all such tasks can be accomplished "reversibly", with an arbitrarily low intrinsic entropy cost, and in reasonable space and time relative to irreversible approaches.

Robin Hanson, 1992



# Reversible Computing





## Summary / Lecture (Quantum Behaviour)

1. The probability of an event (in ideal experiment - no uncertain external disturbance) is absolute square of complex quantity called probability amplitude.
2. When event can occur in several alternative ways, the prob. amp. is the sum of a prob. amp. for each way considered separately.
3. If an experiment, capable of determining which alternative is actually taken, is performed, the interference is lost & the prob. becomes the sum of the prob. for each alternative.

Call Prob. Amp.  $\phi$ :

Call Prob.  $P$ .

1.  $\text{Prob} = |\phi|^2$

2. If two alternatives  $\phi = \phi_1 + \phi_2$  so  $P = |\phi_1 + \phi_2|^2$

3. In the case 3. above:  $P = |\phi_1|^2 + |\phi_2|^2 = P_1 + P_2$

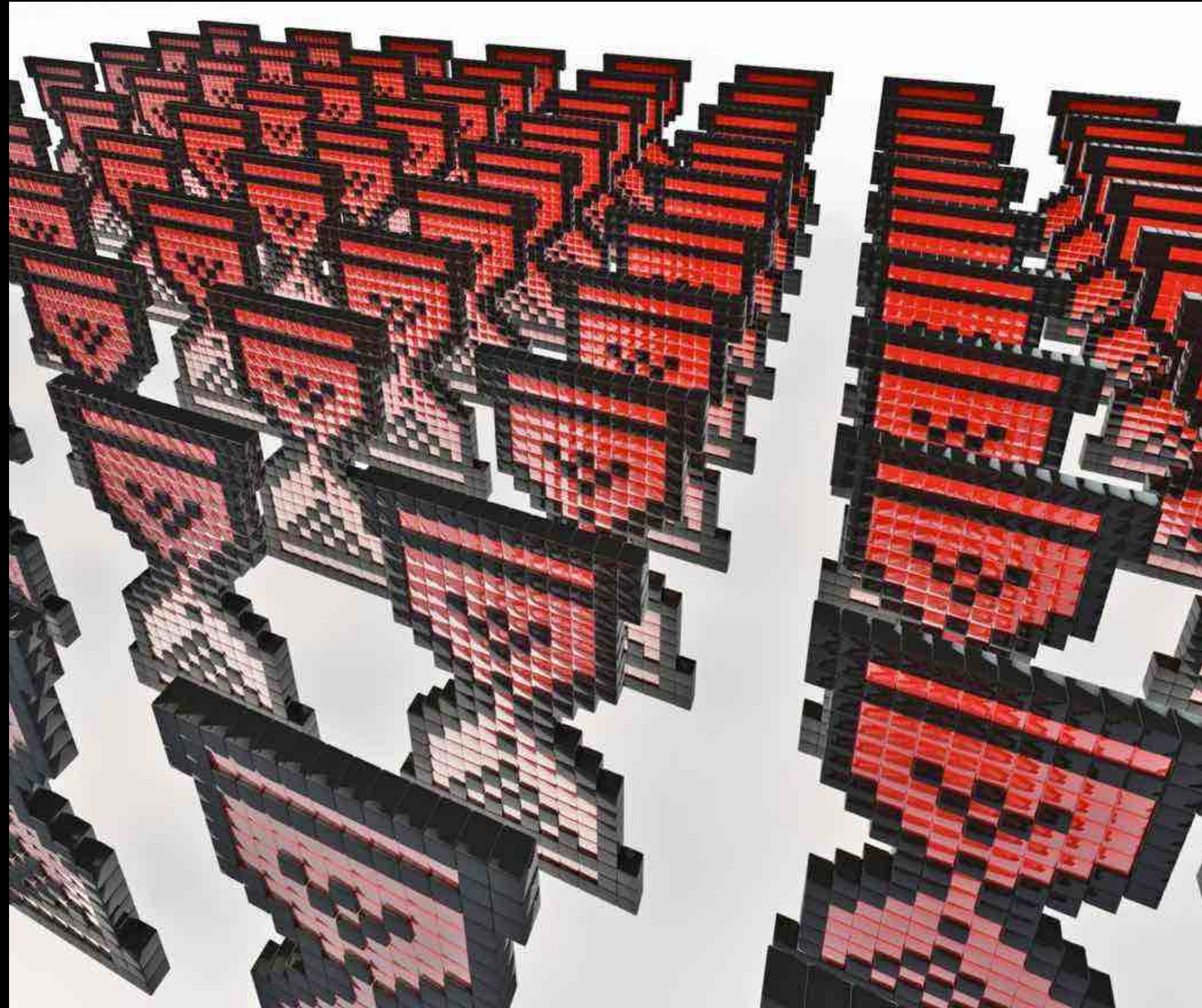




# Reversible Time: Secret to *Concurrency*

- *Google* created the first WAN scale SQL in *Spanner*, by redefining the time API:
  - Uses GPS Clocks
  - Time is no longer a single scalar, it is now an “*interval bounded by events*”, testable through an API
- Distributed systems today use *timestamps* as a crutch
- What happens when they go backwards?

Image courtesy Shutterstock





**entanglement  
might explain the  
arrow of time**



Seth Lloyd



**You know, you  
really ought to  
use formal  
methods for that!**



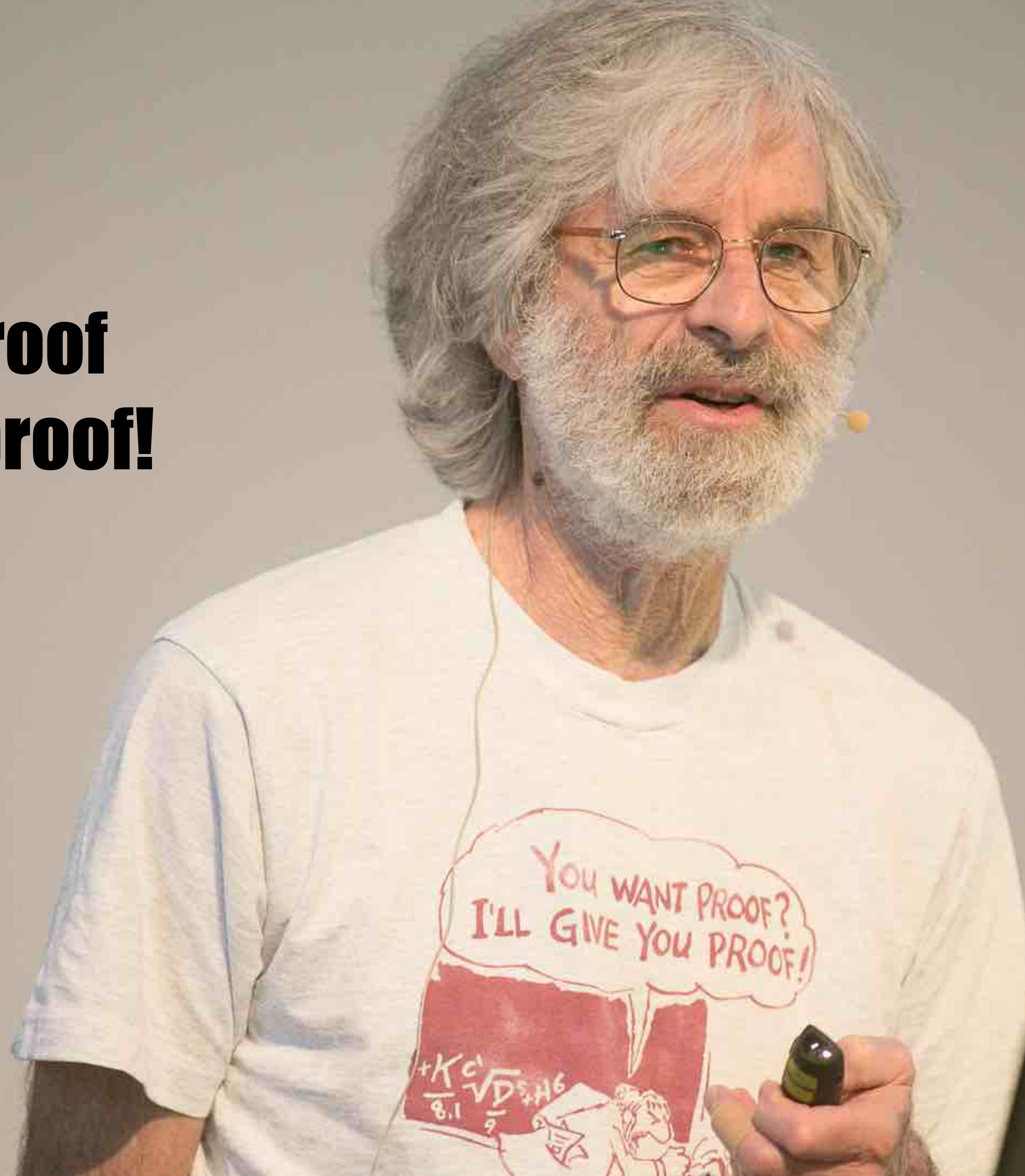


**Spacetime is  
doomed, and  
something has  
to replace it**





**You want proof  
I'll give you proof!**





**Is Quantum  
Computing  
speedup real or  
an illusion?**





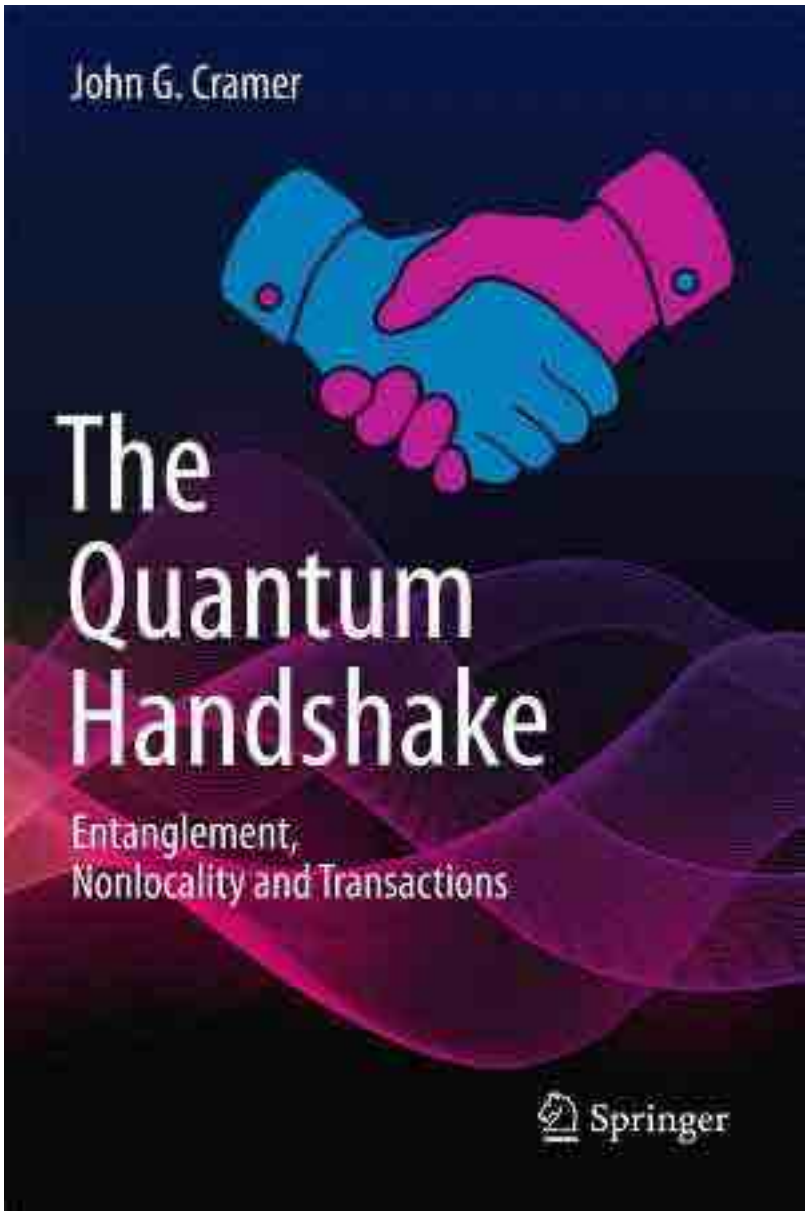
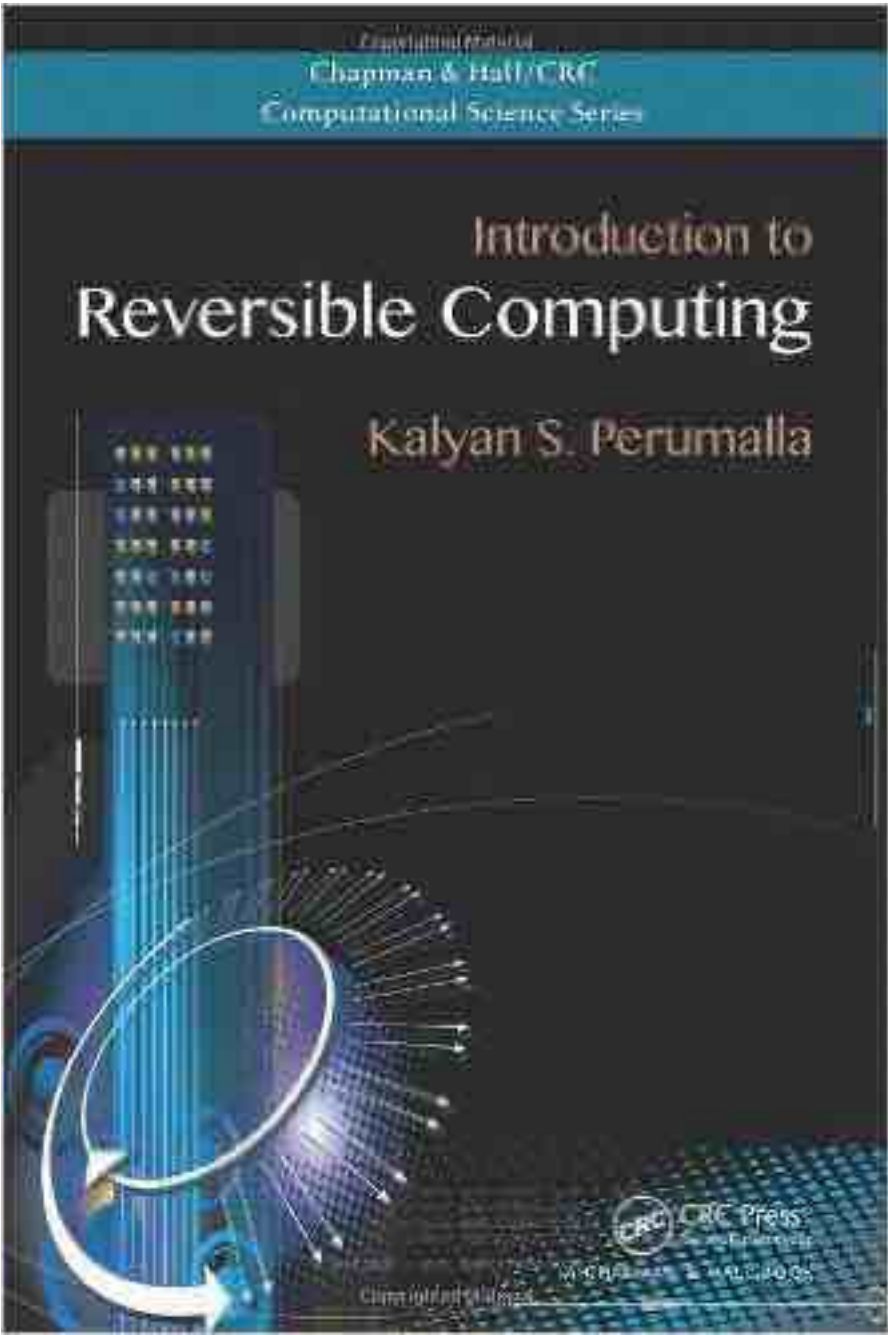
Imperial College London  
Department of Physics

# Negative Probabilities in Physics: a Review

Adam C. Levy

September 2015

Submitted in part fulfilment of the requirements for the degree of  
Master of Science in Physics of Imperial College London



## Interpretations of Negative Probabilities

M. Burgin  
Department of Mathematics  
*University of California, Los Angeles*  
405 Hilgard Ave.  
Los Angeles, CA 90095

### Abstract

In this paper, we give a frequency interpretation of negative probability, as well as for extended probability, demonstrating that to a great extent these new types of probabilities, behave as conventional probabilities. Extended probability comprises both conventional probability and negative probability. The frequency interpretation of negative probabilities gives supportive evidence to the axiomatic system built in (Burgin, 2009) for extended probability as it is demonstrated in this paper that frequency probabilities satisfy all axioms of extended probability.

*Keywords:* probability; negative probability; extended probability; axiom; relative frequency; random experiment; random event



# Time and Computer Science

Simultaneity is a Myth

“at the same time” is like asking what’s north of the north pole

Negative probability is just as real as positive probability

Just with **before** and **after** *substituted*

In quantum mechanics, all probabilities are complex

Time is change, and change can be represented as a tree,  
be careful what to pick for a *root*



**The  
universe  
is like a  
box of  
chocolates**



Lee Smolin



# A Potential Insight: The Subtime Conjecture

*“We must, therefore, be prepared to find that further advance into this region will require a still more extensive renunciation of features which we are accustomed to demand of the space time mode of description”*

~ Niels Bohr



**Genius or lunatic?**



Rasputin






Morpheus





**Would you like to take the red pill or the blue pill?**



A close-up, low-angle shot of Morpheus from the movie The Matrix. He is wearing his signature dark sunglasses and a dark suit. His expression is serious and contemplative as he looks slightly off-camera. His hands are visible in the foreground, holding a small object. The background is dark and out of focus, suggesting an indoor setting with wooden paneling.

**I mean, like,  
you **really** want  
to take the red  
pill?**

Morpheus



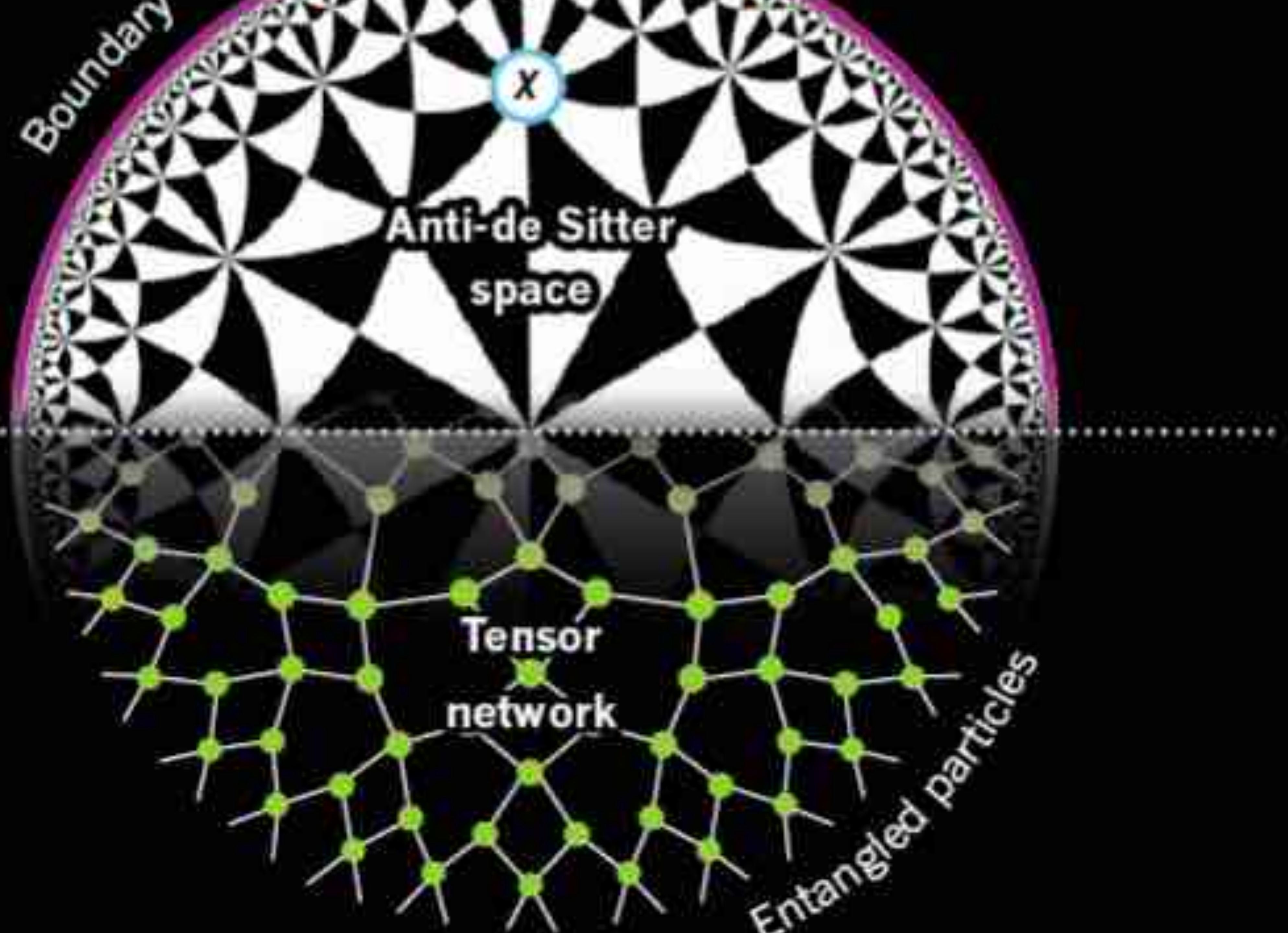
# *THE ENTANGLEMENT CONNECTION*

The ghostly quantum phenomenon of entanglement may be what knits space-time into a smooth whole.

In an infinite model universe known as anti-de Sitter space, the effects of gravity at any point  $x$  in the interior are mathematically equivalent to a quantum field theory on its boundary. This universe can be visualized in 2D by filling it with imaginary triangles. Although the triangles are identical, they look increasingly distorted as they approach the boundary.









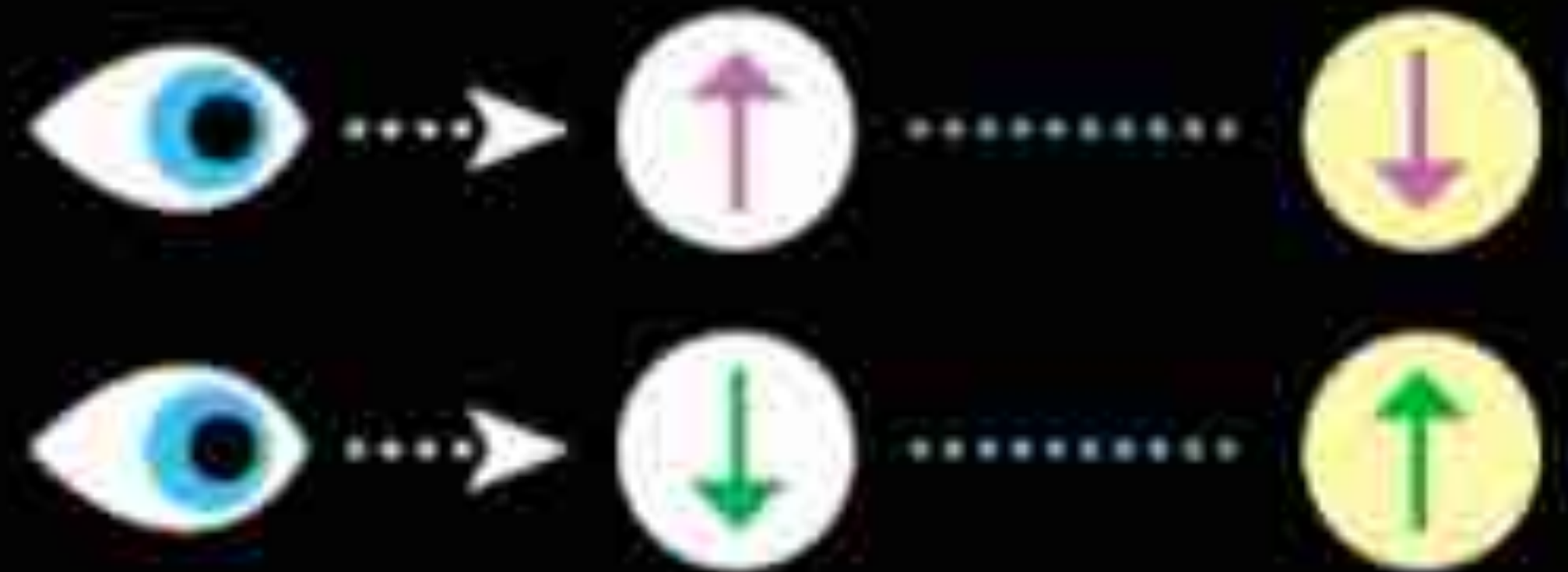
# ***What is quantum entanglement?***

In 1935, Albert Einstein, Boris Podolsky and Nathan Rosen (EPR) pointed out that a connection can exist between widely separated quantum systems: a measurement of one will determine the state of the other.

## **EXAMPLE**



The particles  
are separated.



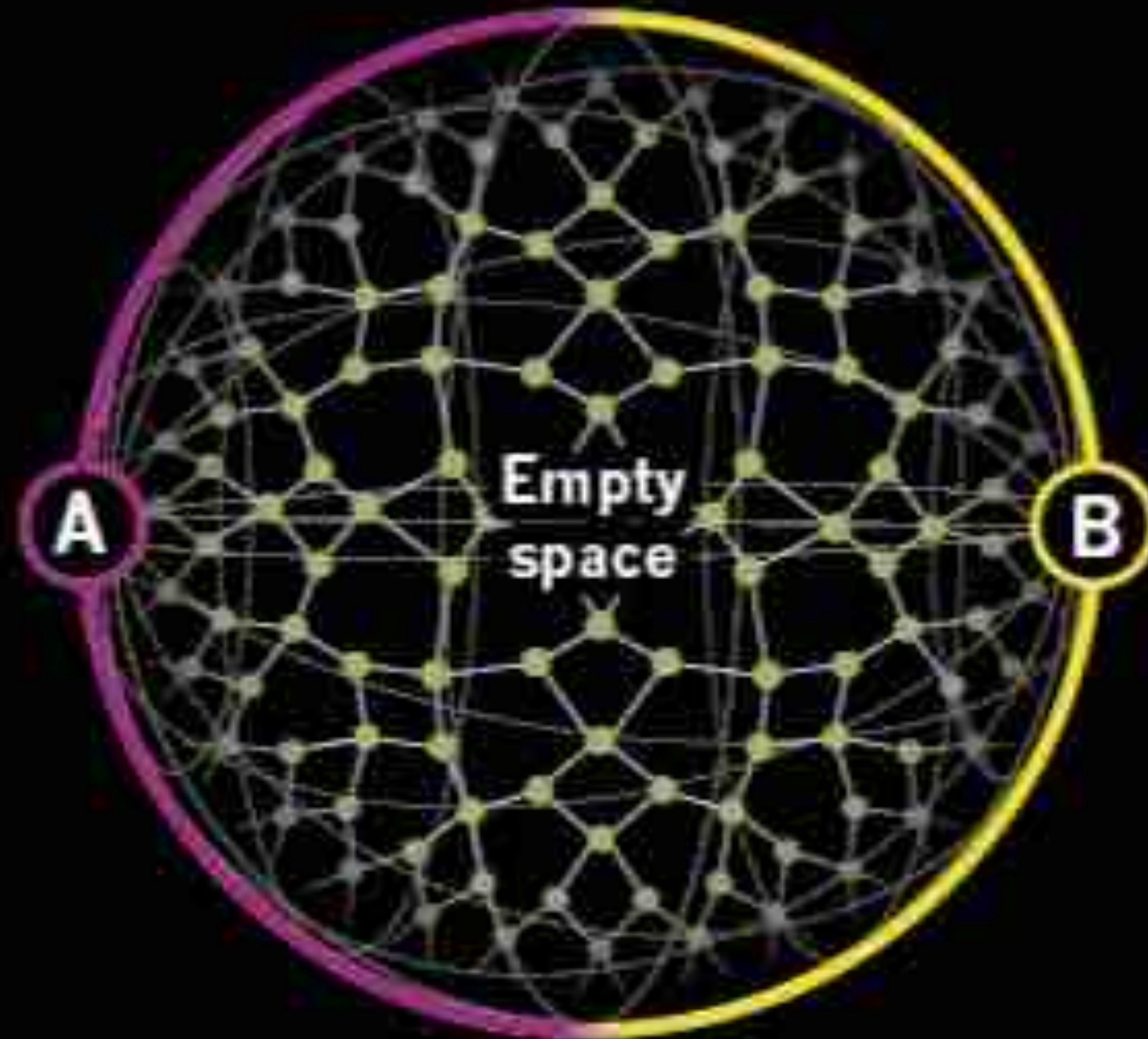
Entangled spins: if one particle is spinning up, the other spins down, and vice versa.

Observation of one particle instantaneously reveals the state of the other.



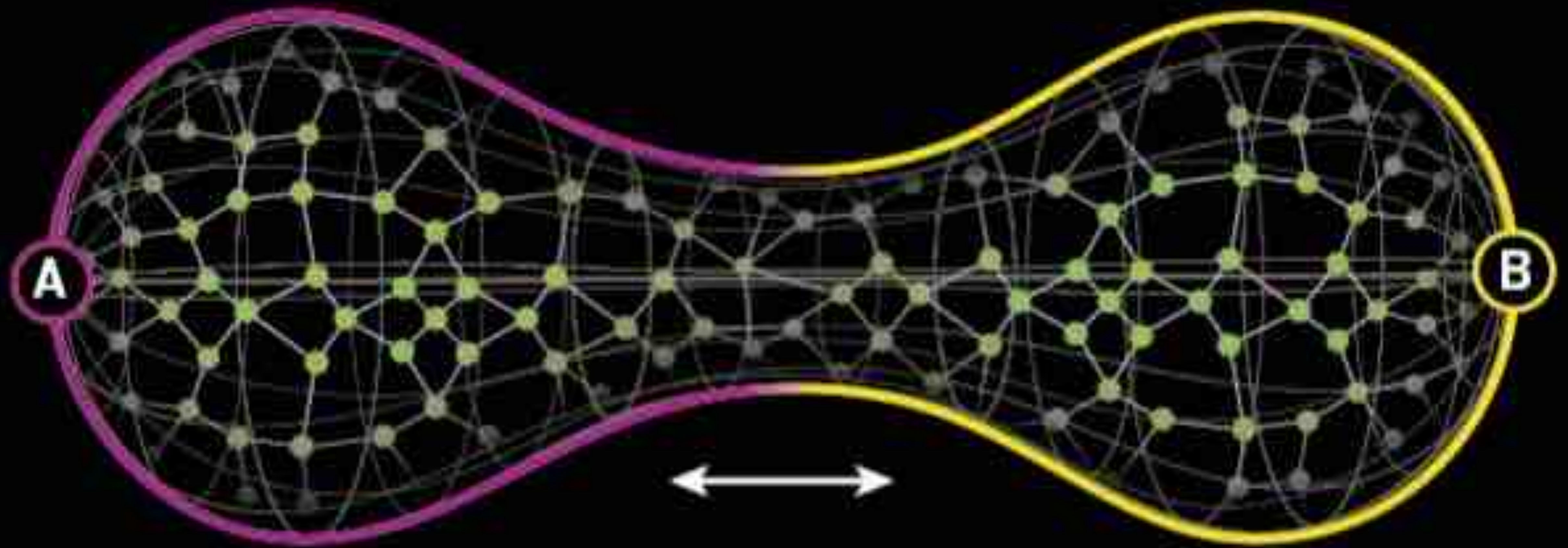
# DISENTANGLEMENT

The bulk–boundary correspondence implies that space on the inside is built from quantum entanglement around the outside.





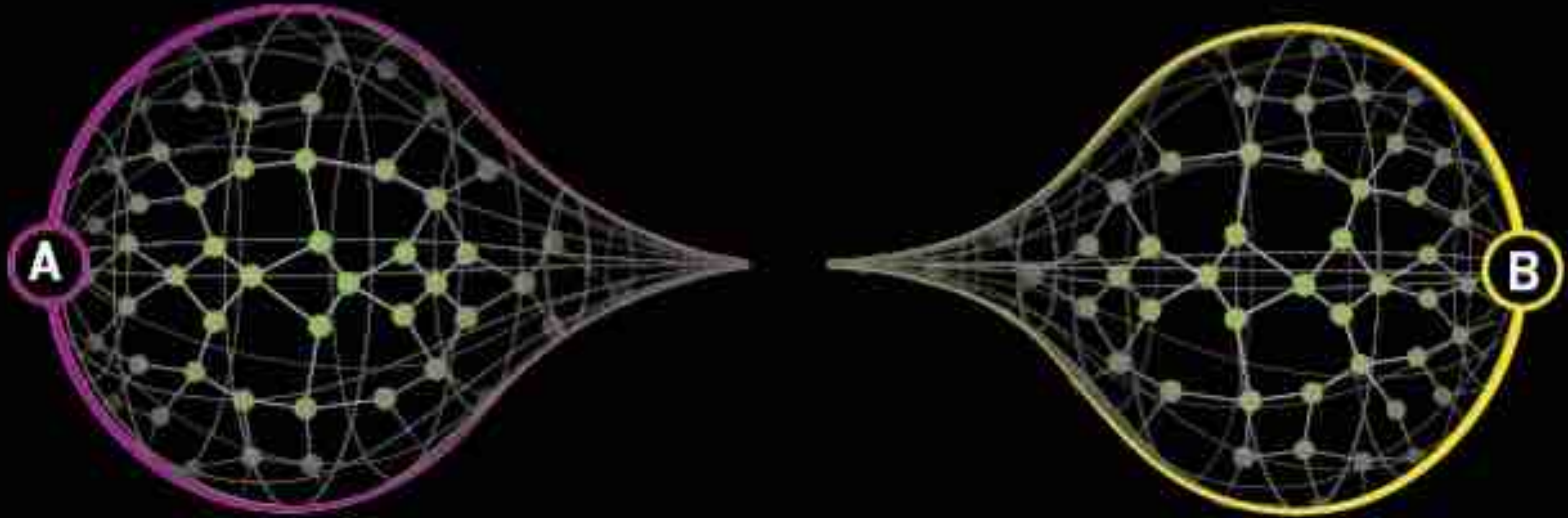
Even when the bulk universe is empty, the quantum fields in any two regions of the boundary (A and B) are heavily entangled with one another.



If the entanglement between these regions is reduced, the bulk universe ***starts pulling apart.***



*starts pulling apart.*

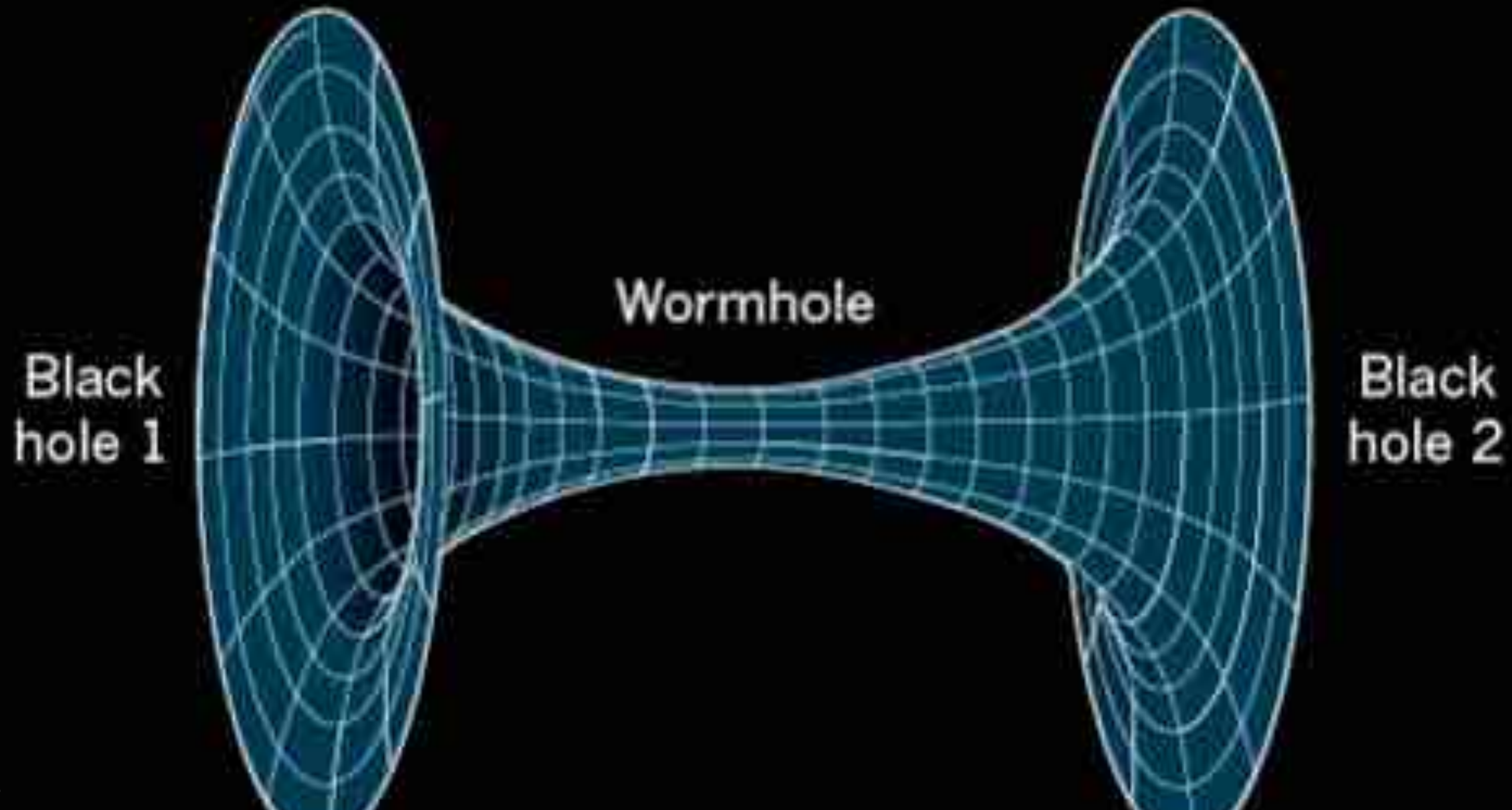


When the entanglement is reduced to zero,  
the bulk universe splits in two —  
*showing that entanglement is  
necessary for space to exist.*

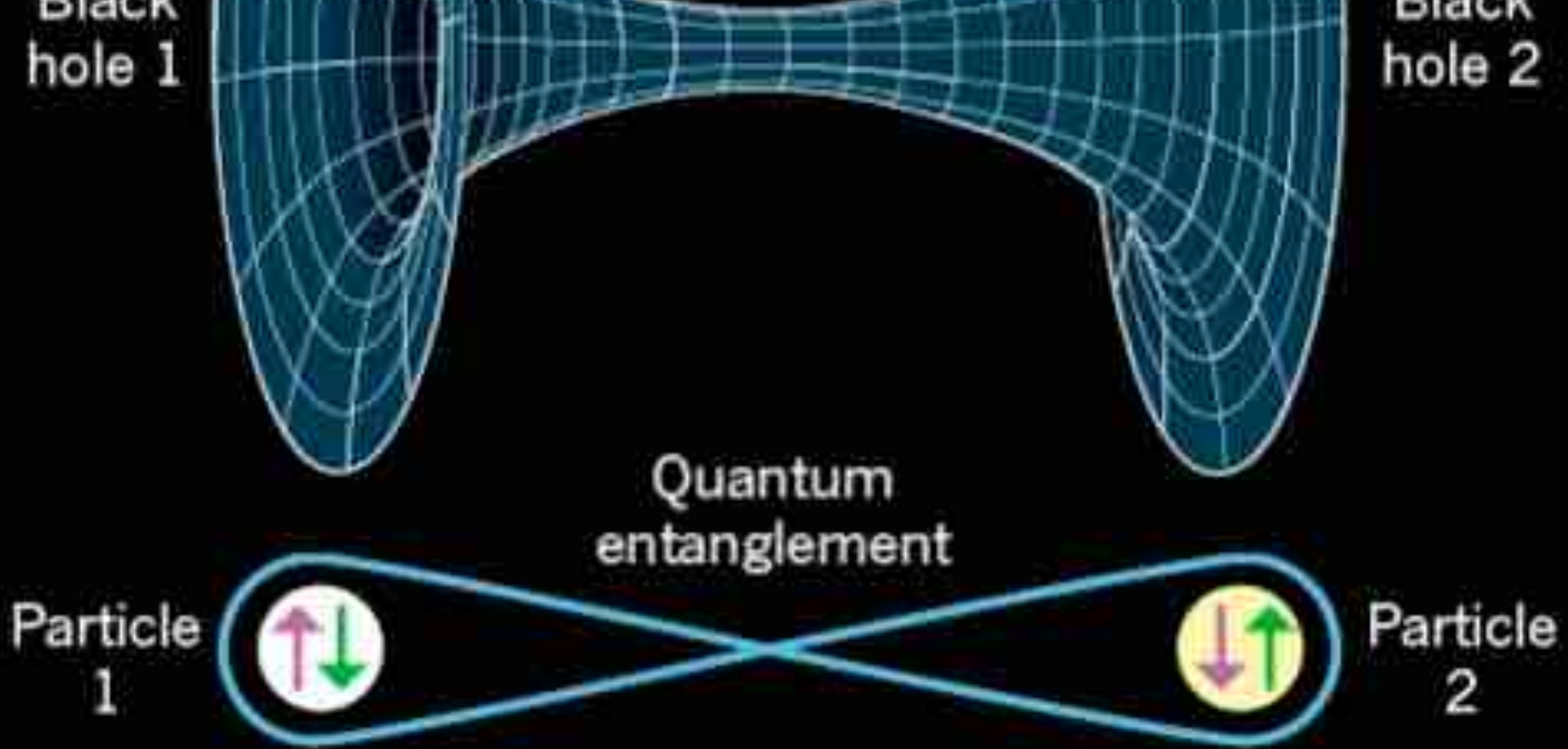


# ER = EPR

Also in 1935, Einstein and Rosen (ER) showed that widely separated black holes can be connected by a tunnel through space-time now often known as a wormhole.







Physicists suspect that the connection in a wormhole  
and the connection in quantum entanglement  
***are the same thing, just on a vastly different scale.***  
Aside from their size there is no fundamental difference.



# Computer Science Driving?

## **DEMONIC programming: a computational language for single-particle equilibrium thermodynamics, and its formal semantics**

Samson Abramsky

Dominic Horsman

Department of Computer Science, University of Oxford, Parks Road, Oxford, OX1 3QD, UK

`{samson.abramsky,clare.horsman}@cs.ox.ac.uk`

Maxwell's Demon, 'a being whose faculties are so sharpened that he can follow every molecule in its course', has been the centre of much debate about its abilities to violate the second law of thermodynamics. Landauer's hypothesis, that the Demon must erase its memory and incur a thermodynamic cost, has become the standard response to Maxwell's dilemma, and its implications for the thermodynamics of computation reach into many areas of quantum and classical computing. It remains, however, still a hypothesis. Debate has often centred around simple toy models of a single particle in a box. Despite their simplicity, the ability of these systems to accurately represent thermodynamics (specifically to satisfy the second law) and whether or not they display Landauer Erasure, has been a matter of ongoing argument. The recent Norton-Ladyman controversy is one such example.

In this paper we introduce a programming language to describe these simple thermodynamic processes, and give a formal operational semantics and program logic as a basis for formal reasoning about thermodynamic systems. We formalise the basic single-particle operations as statements in the language, and then show that the second law must be satisfied by any composition of these basic operations. This is done by finding a computational invariant of the system. We show, furthermore, that this invariant requires an erasure cost to exist within the system, equal to  $kT \ln 2$  for a bit of information: Landauer Erasure becomes a theorem of the formal system. The Norton-Ladyman controversy can therefore be resolved in a rigorous fashion, and moreover the formalism we introduce gives a set of reasoning tools for further analysis of Landauer erasure, which are provably consistent with the second law of thermodynamics.

**The issue with Maxwell's  
Demon is now resolved,  
thanks to formal methods  
from computer science**

Samson Abramsky



**I told you you  
should use  
formal methods  
for that!**





**Who will finish the revolution  
Lamport started?**

***A General Theory of Concurrency***

**\*\*All of You\*\***



Leslie was right in the first place,  
it's not about time, it's about  
events, and in introducing  
**“happened before”**

I would like to introduce:  
***“unhappened”*** **“before”**  
**“happened before”**



# Take-aways

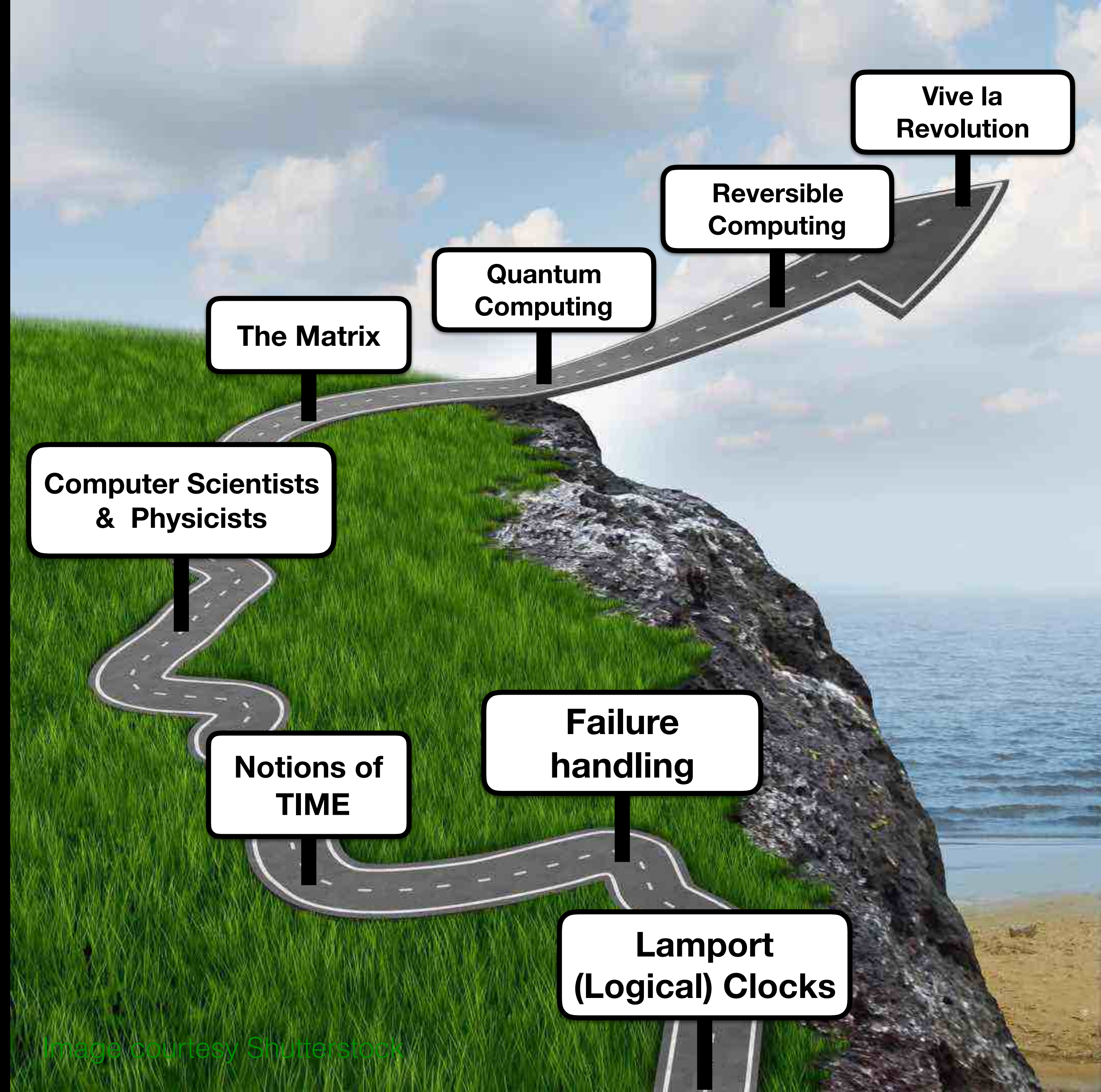
- **“Instantaneous” has no meaning: simultaneity is a myth**
- **Entanglement: Once I measure my one of the entangled particles, I know what you would measure or will measure; our actions are uncoordinated**
- **Entanglement is monogamous**
- **Spacetime is doomed**
- **Time is change that we can count**



**Questions?**



# Lamport's Unfinished Revolution





# References

## Nima Arkani Hamed:

Science Museum Interview, London: <https://www.youtube.com/watch?v=pup3s86oJXU>

Cornell Lecture: [Space-time is doomed. What replaces it?](#)

Perimeter Lecture: [A 21st-century discourse on quantum mechanics and space-time.](#)

## Lorenzo Maccone

Physics ArXiv Blog: [How Time Emerges from Entanglement.](#)

Original ArXiv Paper: [A quantum solution to the arrow of time dilemma.](#)

Experiment: Time from quantum entanglement: [an experimental illustration](#)

## Leonard Susskind

Stanford: [Entanglement builds spacetime](#) The ER=EPR argument from [Juan Maldacena](#).

[ER=EPR but Entanglement is Not Enough](#) (With a connection to *complexity theory*).

Cornell: [Entanglement and the Hooks that Hold Space Together.](#)

There are many others, e.g:

[How Spacetime is built by Quantum Entanglement: New Insight into Unification of General Relativity and Quantum Mechanics.](#)

[New Scientist. Entanglement is the thread that binds spacetime together.](#)

[Seth Lloyd](#), [Brian Swingle](#), [Van Raamsdonk](#), [Sean Carroll](#).

By this Author: Paul Borrill.

@plborrill [paul@borrill.com](mailto:paul@borrill.com)

[Stanford EE380 Seminar on Time in Computer Science](#)

Youtube Video: [Stanford Seminar.](#)

***Special thanks to: Leslie Lamport for his inspiration, Ines Sombra for fabulous organization, and João Taveira for the Keynote Template***