On the Relativistic Structure of Logical Time in Distributed Systems*

Friedemann Mattern
FB Informatik, Universität des Saarlandes, D 6600 Saarbrücken, Germany
mattern@cs.uni-sb.de

Abstract

A distributed system, where processes communicate via messages with unpredictable transmission times, is characterized by the fact that no process has an up-to-date and consistent view of the system's global state. Furthermore, the notion of global time does not exist *a priori* within such a system. However, by defining time to be a partially ordered set of vectors forming a lattice structure, one gets a notion of logical time which is realizable in such a system and which represents the causality structure of events in an isomorphic way. With this definition, the relations "later" and "at the same time" get a new and generalized but adequate interpretation. Interestingly, it turns out that this notion of time is similar in structure to Minkowski's relativistic space-time model. We motivate the concept of "vector time", elaborate its properties, and discuss the analogy to relativistic space-time.

1 Distributed Computations and Time Diagrams

A distributed system consists of sequential processes which communicate solely via messages. We assume that message transmission times are unpredictable and that the processes don't have access to a global clock or to perfectly synchronized local clocks. With those assumptions, no process has an up-to-date and consistent view of the global state. This is the cause for many interesting phenomena and problems which characterize distributed systems and which are non-trivial to solve [11, 12, 14, 19].

*This paper is based on the two earlier papers by the author "Virtual Time and Global States of Distributed Systems" and "Über die relativistische Struktur logischer Zeit in verteilten Systemen".

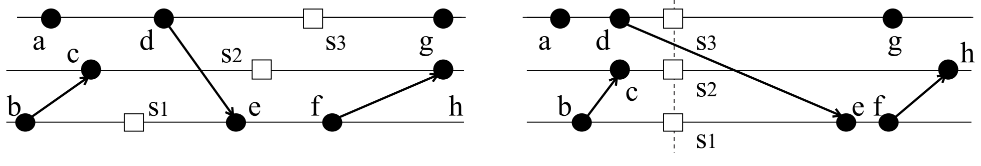

a
d
s3
g
c
s2
b
s1
e
f
h
a
d
s3
g
c
s2
h
b
e
f
s1

Figure 1: Two equivalent time diagrams.

The behavior of each process in the system is governed by an algorithm which determines the sequence of local actions and the reaction of the process to incoming messages. The concurrent execution of the local algorithms, which is coordinated by messages, forms a *distributed computation*. Formally, the execution of actions can be viewed as atomic *events* which are usually classified into *send events*, *receive events*, and *internal events*. Such an abstract distributed computation can be depicted with the help of a *time diagram* (cf. Fig. 1) where time moves from left to right.[1] Messages are drawn as arrows which go from left to right, and events (which represent atomic actions) are depicted by dots.

In a time diagram, the distributed computation is represented in a way an idealized external observer would see it, if the observer is instantaneously informed about the occurrence of an event. It seems to be clear, however, that the exact global time at which an event happens is of no concern provided that the local sequence of events is not changed and message arrows do always go from left to right. Hence, a time diagram represents a whole class of executions which are equivalent in a way that will become clear further down. In that sense the left time diagram of Figure 1 is equivalent to the right time diagram. Informally, the left diagram can be transformed into the right diagram by stretching and compressing the horizontal process axes. Note that in the right time diagram events s_1, s_2, s_3 are aligned vertically and connected by a straight vertical line. Such "cut lines" will play an important role further down.

In order to give precise meanings to the notions, we now provide some definitions. We assume that an abstract distributed computation with a corresponding time diagram is given, and that E denotes the set of its events which happen at processes $P_1, ..., P_n$.

Definition 1.1 (global event order '$\prec$').
Let '$\prec$' denote the smallest transitive relation on the set of events E, such that $e \prec e'$ for $e, e' \in E$ if

(1) e and e' happen at the same process and e is the immediate predecessor of e' or

[1] "*It is true that there are certain implicit dangers in using such graphical representations, because in every geometrical diagram time appears to be misleadingly spatialized. On the other hand, such diagrams, provided we do not forget their symbolic nature, have a definite advantage...*" (From Milič Čapek's philosophical critique of Minkowski's space-time concept [20] and its simplified models. Further down we will see that our time diagrams are indeed very similar to the diagrams which are used to represent Minkowski's space-time model.)

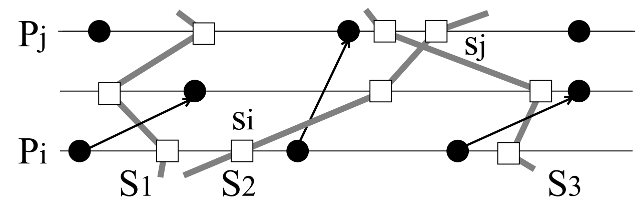

Pj
Pi
sj
si
S1
S2
S3

Figure 2: Consistent and inconsistent cuts.

(2) e' is the receipt of a message which was sent by event e (i.e., e' is the send event that corresponds *to the receive event e).*

The reflexive closure of $\prec$ will be denoted by $\preceq$. Because we assume that in a time diagram message arrows do only go from left to right, the '$\prec$'-relation contains no cycles. The relation was called "happens before" by Lamport [9], because $e \prec e'$ signifies that e happens before e' – in the sense that e is drawn to the left of e', (i.e., that e happens earlier than e' in global time), but also in the sense that e causally precedes e'. Note that the converse is not true; $e \prec e'$ does not necessarily hold if e is left of e' in a time diagram (cf. events a and c in Figure 1). Obviously, the partial order '$\prec$' abstracts from the exact place in the time diagram where an event is located; the two diagrams of Figure 1 therefore depict the *same* partial order. The '$\prec$'-relation can be viewed as (potential) *causality*; it can be depicted on time diagrams by paths which go from left to right:

Corollary 1.2 (causal path).
The relation $e \prec e'$ holds between two events e, e' of a computation iff there exists a directed path (consisting of message arrows and fragments of process lines traversed from left to right) from e to e' on a time diagram corresponding to the computation.

The corollary follows from Definition 1.1; intuitively its correctness becomes evident if one reads $e \prec e'$ as "e may influence e'". Formally, it can be proved by induction on the length of the path and by making use of the transitivity of the '$\prec$'-relation. For example, in Figure 1 $b \prec c$ (b is a direct predecessor of c) and $a \prec h$ (there exists a path from a via d, e, f to h). Hence, a path in a time diagram is merely a graphical representation of a *causal chain.*

Later, we also need the symmetric relation "causally independent". We define it as follows:

Definition 1.3 ('$||$'-relation).
For two events $e, e' \in E$ the relation $e||e'$ holds iff $\neg(e \prec e') \wedge \neg(e' \prec e)$.

Obviously, for two events e, e' one has $e \prec e'$ or $e' \prec e$ or $e||e'$. Up to now, we did not formally define the notion of a distributed computation. We shall now do so in a way which conforms to the informal view given above.

Definition 1.4 (distributed computation).
An (n-fold) distributed computation *over an event set E is an n-tuple $(E_1, ..., E_n)$ with a relation $\Gamma \subseteq S \times R$ of corresponding send events $S \subseteq E$ and receive events $R \subseteq E$ $(S \cap R = \emptyset)$, such that each $E_i \subseteq E$ is linearly ordered by a relation $\prec_i$ and the following three conditions hold:*

1. *The event sets $E_1, ..., E_n$ are pairwise disjoint.*
2. *Γ is left-unique and right-unique.*
3. *The smallest transitive relation $\prec$ which fulfills the two axioms*
   A1. $a \prec_i b \;\Rightarrow\; a \prec b$
   A2. $(a, b) \in \Gamma \;\Rightarrow\; a \prec b$
   is an (irreflexive) partial order.

Obviously, the totally ordered sets E_i represent the local computations, Γ represents the set of messages sent and received in the computation, and '$\prec$' is the above-mentioned "happens before" relation which represents causality. Condition 3 of the definition requires the relation to be cycle free. Intuitively, this is an obvious requirement; it guarantees that for each distributed computation it is possible to draw a time diagram for which—as it is required by definition—all message arrows go from left to right. If global time does also go from left to right[2] and if the events on a process line are sorted according to the '$\prec_i$'-relation, this means that the future cannot influence the past – a property which every sensible notion of time should have of course!

2 Cuts and Lattices

We aim at an adequate notion of time for distributed systems. Although we assume that global time is not available[3] within a distributed system, we can use global time as a starting point and assume that it "exists" for an idealized observer who sees the whole time diagram of a computation at once. In such a diagram, instants in global time are represented by vertical lines which cut all process axes "simultaneously". Since there is no canonical diagram for a given distributed computation, this constructive definition of a global instant seems to be problematic at first sight. However, all time diagrams which represent the same computation are equivalent in the sense that they can be mutually transformed by compressing and expanding the process lines. These "rubber band transformations" are exactly those transformations on time diagrams which leave the causality relation invariant. In that way, a vertical line cutting the process lines at special additional cut events $s_1, ..., s_n$ (cf. Fig. 1) may be transformed in a

[2] *"Everybody knows that time flows from left to right unless you are left-handed".* (Anonymous posting on the usenet electronic bulletin board.)

[3] Be it whether we deny an absolute time reference for philosophical reasons, or whether we argue more technically that local clocks cannot be perfectly synchronized.

zigzag line. Independently of its shape, such a cut line separates the set of events E in two disjoint sets, namely the "past" (i.e., all those events located to the left of the line) and the "future" (the remaining events). In that sense, cut lines qualify as a substitute for instants in global time. This motivates the definition of a *cut* of a distributed computation:

Definition 2.1 (local event order '$\prec_l$').
The local event order *'$\prec_l$' is defined by $e \prec_l e'$ iff $e \prec e'$ and e, e' are events of the same process.*

Note that '$\prec_l$' is the union of all '$\prec_i$' defined on E_i.

Definition 2.2 (cut).
A finite subset $S \subseteq E$ is a cut *of E if $(e \in S \wedge e' \prec_l e) \Rightarrow e' \in S$.*

That is, cuts are left-closed subsets with respect to '$\prec_l$'. One should observe that a cut line is merely a geometrical object that cuts a diagram into two parts, whereas a cut is a mathematical object, namely a set of events. To each cut line, however, one can associate the set of the events to its left as a cut, and often it is convenient to represent cuts by lines in a time diagram. A particular class of cuts is of great interest:

Definition 2.3 (consistent cut).
A finite subset $S \subseteq E$ is a consistent cut *of E if $(e \in S \wedge e' \prec e) \Rightarrow e' \in S$.*

Because of $\prec_l \subseteq \prec$, every consistent cut is a cut according to Definition 2.2. Cuts which are not consistent are called *inconsistent.*

Figure 2 shows cut lines of consistent (S_1, S_3) and of inconsistent (S_2) cuts. Because by definition a send event is part of a consistent cut if the corresponding receive event is part of the cut, an inconsistent cut line is characterized by a "message from the future" (i.e., a message whose send event is to the right of the cut line although its receive event is to the left). Such messages, which traverse the cut line "in the wrong sense", are responsible for the fact that at least two cut events s_i, s_j are causally related in an inconsistent cut – obviously there exists a causal chain from cut event s_i of the sending process to the cut event s_j of the receiving process (see Figure 2). For consistent cuts this is not the case, there all cut events are pairwise causally independent.

Intuitively, vertical cut lines should be the adequate substitutes for instants in global time. Hence, cuts with cut lines that can be made vertical are of particular interest. The following theorem characterizes those cuts:

Theorem 2.4 (rubber band consistency criterion).
A cut line of a time diagram represents a consistent cut iff it can be transformed into a vertical line by the rubber band transformation.

The following sketch shows how the proof can be constructed in a do-it-yourself way: Cut a given time diagram with a consistent cut line along that line in two parts. Then move the right part to the right until the two parts do no longer overlap and are separated by a (vertical) gap. Message arrows that have been cut must then be repaired by bridging the gap. They still go from the left to the right because only the receive events have been moved to the right. (This is a consequence of the fact that the cut is consistent.) The cut line can then be drawn as a vertical line within the gap. Conversely, it is obvious that a message which is responsible for the inconsistency of the cut (i.e., which traverses the cut line from the right to the left) cannot be drawn as an arrow going from the left to the right in a time diagram with a vertical cut line.

Consistent cuts can be viewed as sets of events which did already happen. Because with this temporal interpretation there are usually several events to the right of a cut line which might be executed next in the current global state, the "next cut" is usually not unique. It is possible, however, to define a *partial* order "later" on the set of all cuts. Informally, a cut S_2 is *later* than a cut S_1 if the cut line of S_2 lies on the right of the cut line of cut S_1 (see Figure 2).

Definition 2.5 (later cut).
A cut S_2 is later *than a cut S_1 if $S_1 \subseteq S_2$.*

With the set theoretic definition of cuts according to Definition 2.2 we obtain the following theorem:

Theorem 2.6
With respect to "later" the cuts of a time diagram form a lattice.

Proof. Follows directly from the definition. A lattice is a partially ordered set where any two if its elements have a greatest lower bound inf and a least upper bound sup. Obviously, inf $= S_1 \cap S_2$ and sup $= S_1 \cup S_2$ for any two cuts S_1, S_2. (Note that inf and sup are left-closed with respect to $\prec_l$.) □

Only the consistent cuts are relevant for the definition of a notion of time. They do also form a lattice:

Theorem 2.7 (sublattice of consistent cuts).
For a given computation, the set of consistent cuts is a sublattice of the set of all cuts.

Proof. One has to show that the consistent cuts of the computation are closed under $\cup$ and $\cap$.

(1) Let S_1, S_2 be two consistent cuts. Furthermore, let $x \in S_1 \cap S_2$ and $y \prec x$. Then $y \in S_1$ because S_1 is consistent, and $y \in S_2$ because S_2 is consistent. Hence $y \in S_1 \cap S_2$, i.e., $S_1 \cap S_2$ is consistent.

(2) Let $x \in S_1 \cup S_2$ and $y \prec x$. Consider two cases:

  (a) $x \in S_1$. Then $y \in S_1$ because S_1 is consistent.

  (b) $x \in S_2$. Then $y \in S_2$ because S_2 is consistent.

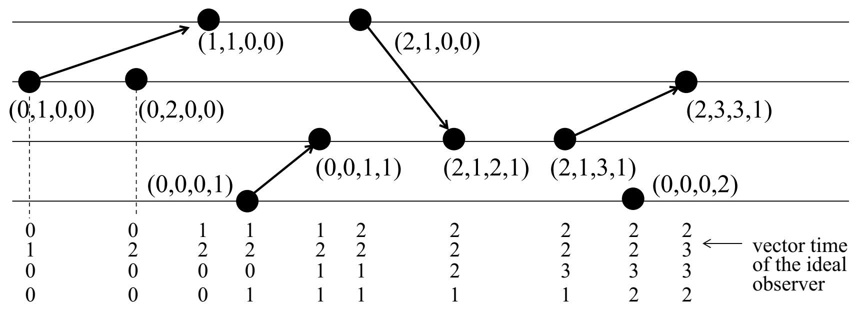
Hence $y \in S_1 \cup S_2$, i.e., $S_1 \cup S_2$ is consistent. $\Box$

The lattice structure of consistent cuts guarantees that for any two consistent cuts S_1 and S_2 there is always a consistent cut later than both of them and a consistent cut earlier than both of them. This can be extended to a finite set of consistent cuts: $\sup(S_1, ..., S_k) = S_1 \cup ... \cup S_k$ is later than $S_1, ...S_k$ (and accordingly for inf). This means that a set of "points" in time (which is the temporal interpretation of consistent cuts) does always have a common future and a common past.

3 Logical Time

Our goal consist in defining a notion of time for distributed systems which, on the one hand, is realizable within such systems (and should in therefore not make use of global clocks) but, on the other hand, has some useful properties which justifies the name "time". For example, we would like to be able to assign time values to events such that it is possible to infer potential causality between events or to exclude it in the sense that a "later" event cannot influence an "earlier" event. Furthermore, it would be nice if a global snapshot were consistent if all local snapshot events are "simultaneous" with respect to the notion of time. We will see that so-called vector time, which will be defined in the following section, does indeed have this property.

The most obvious models for *real time* are the rational and the real numbers together with their linear order. These sets are dense; for any two numbers there is another number that lies between them. This is different for typical *logical* models of time for distributed systems. In a distributed computation as we defined it, nothing happens between two successive events. Hence time needs only be advanced with the occurrence of an event and is therefore discrete. This is nothing uncommon in computer science; an event-driven simulation system is a typical application where this is also the case. Such a model of time requires that each event of the distributed computation gets a unique instant in time. Formally, this means that a function $C: E \longrightarrow T$ must be found which assigns a *timestamp* $C(e)$ of a suitable set T to each event $e \in E$. The comparison of timestamps of different events should allow to draw certain conclusions about the relation of the events. It seems to be plausible that at least the notions "earlier" or "later" should exist within the notion of time. Hence logical time should be a partial order $(T, <)$. From an abstract point of view, the function C can be



(1,1,0,0)
(2,1,0,0)
(0,1,0,0)
(0,2,0,0)
(2,3,3,1)
(0,0,1,1)
(2,1,2,1)
(2,1,3,1)
(0,0,0,1)
(0,0,0,2)
0 1 0 0
0 2 0 0
1 2 0 0
1 2 0 1
1 2 1 1
2 2 1 1
2 2 2 1
2 2 3 1
2 2 3 2
2 3 3 2
vector time of the ideal observer

Figure 3: Propagation of time knowledge.

called a *logical clock*. A reasonable requirement on C is that it conforms to the causality relation:

$$\forall e, e' \in E: \; e \prec e' \implies C(e) < C(e').$$

This condition is often called the *clock condition*. Stated verbally, it reads "an event e should get a smaller timestamp than an event e' if e can causally affect e'".

In his seminal article "Time, Clocks, and the Orderings of Events in a Distributed System" [9], Lamport presented in 1978 a scheme based on an integer domain T for the values $C(e)$ of the clock function C. His logical clock is realized by a system of counters (one for each process) and a simple message handling protocol. The scheme lacks a desirable property, however, because by mapping the events onto linearly ordered integers some structure is lost. Events which are causally independent get assigned timestamps as if they happen in a certain order. Hence, by checking the timestamps of events, it is usually not possible to assert that some event could *not* affect some other event. For that purpose, the time domain T must represent the event structure E in an isomorphic way. This means that the converse implication of the clock condition should also hold. This is in fact possible with "vector time" as will be shown in the following section.

4 Vector Time

To motivate the concept of vector time discussed below, assume that each process P_i has a simple logical clock implemented by a counter which is incremented by 1 each time an event happens. An idealized external observer who has immediate access to all local clocks knows at any moment the local times of all processes. An appropriate structure to store this global time knowledge is a vector with one component for each process. The example depicted in Figure 3 illustrates the idea.

Because of message propagation delays in distributed systems, the instantaneous knowledge of the idealized observer cannot be realized. Our aim, however, is to construct a mechanism by which each process gets – without extra messages – an *optimal approximation* of this notion of global time. For this purpose each process should be informed at the earliest possible moment about all events which did already happen. For that, we equip each process P_i with a clock C_i which consists of a *vector* of length n, where n is the total number of processes. A clock C_i is initialized with the null vector; it "ticks" immediately before the execution of an event by incrementing the value of its own component:

$$C_i[i] \ := \ C_i[i] + 1.$$

Each message contains a vector timestamp t, where t is the vector time of the local vector clock of the sender when the message is sent. By receiving a timestamped message, a process learns about global time approximation of the other processes. The receiver combines its own time knowledge C_i with the approximation t it receives with the message by

$$C_i \ := \ \sup(C_i, t),$$

where sup denotes the componentwise maximum operation. The timestamp $C(e)$ of an event e occurring at process P_i is the value of clock C_i at the moment of the execution of e. (For receive events this is the value after updating the clock.) Figure 3 illustrates the propagation of time knowledge and the updating of the vector clocks.

Obviously, the events of process P_i are sequentially numbered by the i-th component of clock C_i. Or, to put it differently, before event e, $|C(e)[i]| - 1$ other events did already happen on the same process. In fact, the vector timestamp $C(e)$ of an event e contains in a compact way the complete knowledge about all those events from which e is (potentially) causally dependent. For example, $C(e)[k] = j$ signifies that event e depends on the first, the second,... the j-th event of process P_k; but that it is not dependent of any later event of process P_k. As an alternative to the operational characterization of the vector timestamp $C(e)$ of an event e, $C(e)$ can hence be defined formally as follows:

Definition 4.1 (vector timestamp of an event).
The vector timestamp $C(e)$ of an event e is a vector of $\mathbf{N}^n$ such that for its i-th component

$$C(e)[i] \ = \ |\{e' |\, e' \textit{ is an event of process } \ P_i \ \wedge \ e' \preceq e\}|.$$

One may easily check that this definition is realized by the above mentioned rules for the handling of vector clocks and timestamps. In order to be able to compare time vectors, we define the following relations:

Definition 4.2 (vector time order).
For two time vectors u, v we define

$u \leq v \;:\Leftrightarrow \forall i: \; u[i] \leq v[i]$,
$u < v \;:\Leftrightarrow \; u \leq v \;\wedge\; u \neq v$,
$u||v \;:\Leftrightarrow \; \neg(u < v) \;\wedge\; \neg(v < u)$.

One should observe that '$\leq$' (and hence also '$<$') is a *partial* order. '$||$', which is a reflexive and symmetric (but non-transitive!) relation, can be viewed as a generalization of simultaneity of real time. However, whereas in real time the “now” is merely a durationless point between past and future, simultaneity in vector time has a larger extension.

By Definition 4.1 and by the above mentioned rules it is possible to assign a time vector to each event. In a canonical way it is also possible, however, to assign a time vector $\tau(S)$ to a *cut* S:

Definition 4.3 (time vector of a cut).
Let S be a cut. The vector defined by

$$\tau(S)[i] \;=\; |\{e \in S| \; e \textit{ is an event of process } P_i\}|$$

is called the time vector of cut S.

From Definition 4.3 and Definition 2.5 we get directly:

Corollary 4.4 (later time vector of cut).
S_1 is later than S_2 iff $\tau(S_2) \leq \tau(S_1)$.

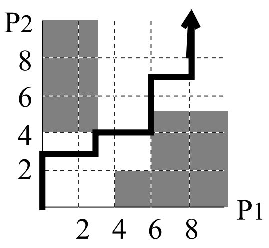
By assigning time vectors to cuts, the lattice structure of cuts or consistent cuts is isomorphically mapped to the corresponding vector sets:

Theorem 4.5 (time lattice).
The time vectors of cuts form a lattice with respect to the partial order '$\leq$'; the time vectors of consistent cuts form a sublattice of the lattice.

Proof. For any two time vectors $\tau(S_1), \tau(S_2)$ there exists the infimum $\tau(S_1 \cap S_2)$ and the supremum $\tau(S_1 \cup S_2)$. $\square$

Besides their order-theoretic interpretation, lattices do also have an algebraic interpretation. If one defines two commutative and associative operations *inf* and *sup* by $\inf(x, y) = u \Leftrightarrow u[i] = \min(x[i], y[i])$, and $\sup(x, y) = v \Leftrightarrow v[i] = \max(x[i], y[i])$ for all components i, one can easily show that $x \leq y \Leftrightarrow x = \inf(x, y)$. The algebraic interpretation means that it is possible to *compute* (i.e., to execute algebraic operations) with time vectors and then use the interpretation of the lattice as a partial order to *compare* time vectors.

Time vectors can be represented in an n-dimensional space yielding an interesting pictorial description of the lattice. If one does only consider time vectors associated to consistent cuts of a given computation (as will become clear further down, only such time vectors can appear as timestamps of events and messages),


P2
8
6
4
2
P1
2
4
6
8

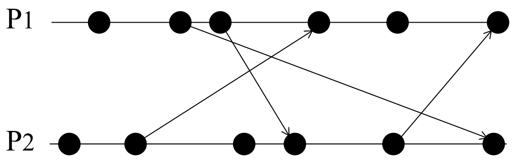

P1
P2

Figure 4: n-dimensional lattice and corresponding time diagram.

one gets esthetically interesting structures. For $n = 3$ these structures have the shape of eroded cubes; Figure 4 shows an example for $n = 2$. In such a structure, each path originating at the null vector and moving in the direction of the diagonal determines a sequence of events which is a linearization of the causality relation '$\prec$'. Therefore, each such path corresponds to a *consistent observation* of the distributed computation. In general there are many different consistent observations of a given distributed computation. Since the intersection of all linearizations of a partial order yields the partial order itself, the facts which are common to all observations is precisely the underlying causality relation. The eroded shape of the n-dimensional cube is due to the fact that time cannot always spread freely in all direction; some dimensions might be closed temporarily because a send event must always be observed before its corresponding receive event.

Definition 4.1 assigns timestamps to events, whereas Definition 4.3 assigns timestamps to cuts. The link between timestamps of events and timestamps of cuts is established by the fact that the "causal past" of an event forms a consistent cut:

Definition 4.6 (causal past $\downarrow e$).
The causal past of an event e is defined by $\downarrow e = \{e' | e' \preceq e\}$.

Theorem 4.7 ($\downarrow e$ is consistent).
For an event $e \in E$, $\downarrow e$ is a consistent cut.

The proof follows immediately from Definition 2.3 and Definition 4.6 and from the transitivity of '$\preceq$'. Now it becomes obvious that the timestamp of an event is nothing but the time vector of its causal past:

Theorem 4.8 (time is causal past).

$$C(e) = \tau(\downarrow e).$$

Proof. $\tau(\downarrow e)[i] \overset{def}{=} |\{e' \in \downarrow e \mid e' \text{ is an event of process } P_i\}| = |\{e' \preceq e \mid e' \text{ is an event of process } P_i\}| \overset{def}{=} C(e)[i]$. □

With the help of time vectors Theorem 4.8 "identifies" an event with its causal past. Seen in that way, vector time appears to be something quite natural – time *is* the set of past events! Time vectors just allow a compact implementation and an algebraic representation – and hence a computationally tractable realization – of set theoretic operations on the causal past of events. The notion of time itself is defined by the causal past of the events.

In order to prove our main Theorem 4.11 below, we need a result that characterizes consistent cuts as closed sets with respect to causal past:

Lemma 4.9 *Let S be a consistent cut. Then $e \in S \iff \downarrow e \subseteq S$.*

Proof.
(1) "$\Rightarrow$": For $e \in S$ and $x \preceq e$ we have $x \in S$ for any x because S is consistent. In particular we have $x \in \downarrow e \Rightarrow x \preceq e \Rightarrow x \in S$, hence $\downarrow e \subseteq S$.
(2) "$\Leftarrow$": From $\downarrow e \subseteq S$ we have $x \in \downarrow e \Rightarrow x \in S$. Because $e \in \downarrow e$, we have $e \in S$.
□

Because the isomorphism of the two lattice structures entails $\tau(S_1 \cup S_2) = sup(\tau(S_1), \tau(S_2))$, it is possible to generalize Theorem 4.8 to arbitrary consistent cuts S:

Corollary 4.10 *Let $S = \{e_1, ..., e_k\}$ be a consistent cut.
Then $\tau(S) = sup(C(e_1), ..., C(e_k))$.*

Proof. Because of Lemma 4.9, $S = \downarrow e_1 \cup, ..., \cup \downarrow e_k$. Hence, $\tau(S) = \tau(\downarrow e_1 \cup, ..., \cup \downarrow e_k) = sup(\tau(\downarrow e_1), ..., \tau(\downarrow e_k))$. According to Theorem 4.8 this equals $sup(C(e_1), ..., C(e_k))$.
□

We can now show that a stronger form of the clock condition holds for vector time:

Theorem 4.11 (isomorphism of causal structure and temporal structure).

$$\forall e, e' \in E : e \prec e' \iff C(e) < C(e').$$

Proof. $e \preceq e' \overset{4.6}{\iff} e \in \downarrow e' \overset{4.7,\ 4.9}{\iff} \downarrow e \subseteq \downarrow e' \overset{2.5,\ 4.4}{\iff} \tau(\downarrow e) \leq \tau(\downarrow e') \overset{4.8}{\iff} C(e) \leq C(e')$. (Also note that different events are assigned different timestamps and that each event has a unique timestamp.) □

Theorem 4.11 has an easy interpretation on time diagrams. An event e' has a larger timestamp than event e if and only if there is a chain in the form of a causal path from e to e'. Obviously, the value of a vector component can only increase along such a path. If, conversely, an event e' has a larger timestamp than another event e, then there must exist a path from e to e' along which the "time knowledge" of $C(e)$ is propagated. Theorem 4.11 does also indicate how to test two events for causal independency:

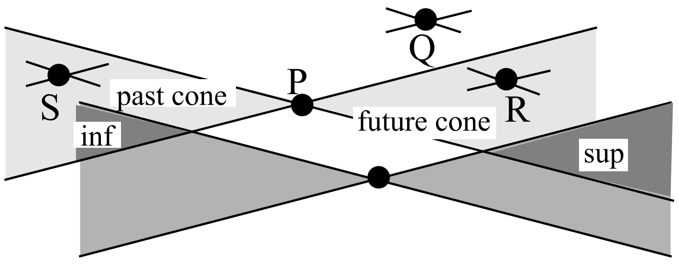

Q
P
S
R
past cone
future cone
inf
sup

Figure 5: Light cones in Minkowski's space-time.

Corollary 4.12 $\forall e, e' \in E: \; e||e' \iff C(e) \,||\, C(e')$.

Put somewhat sloppy, the corollary asserts that exactly those events are mutually independent which happen simultaneously.

5 Minkowski's Relativistic Space-Time

In this section we discuss the analogy between Minkowski's space-time model on the one hand and our time diagrams and partially ordered events on the other hand. The analogy results from the relativistic effect which occurs whenever the propagation delays of signals cannot be neglected. In fact, because of the limits imposed by the speed of light[4], real time is not linearly ordered but merely a partial order, just like vector time. In Minkowski's model $n-1$-dimensional space and one-dimensional time are combined together to give an n-dimensional picture of the world. Because of the bounded propagation delays of signals in this model, an event a that happens at a certain space-time point can only affect another event b if b lies in the *light cone* of a.

Figure 5 depicts the situation for $n = 2$. Events P and Q are causally independent, whereas event S may affect event P (because S lies in the *past light cone* of P) and P may affect R (because R lies in the *future light cone* of P). The "potentially affects" relation is transitive (if X is in the past light cone of Y and Y is in the past light cone of Z then X is in the past light cone of Z), but independency is not transitive: P and Q are mutually independent, as well as Q and R. However, P and R are not causally independent.

For $n = 2$ the light cones form a lattice. Figure 5 depicts the construction. The future light cone of the supremum is formed by the intersection of the two future light cones, and the intersection of the two past light cones forms the past light cone of the infimum. By identifying a cone with its originating point, the lattice structure of the cones is mapped onto the space-time points. (This corresponds to our identification of cuts and cut lines.)

[4]Interestingly, the French physicist Paul Langevin calls the speed of light the "speed limit of causality" in the English translation of his works.

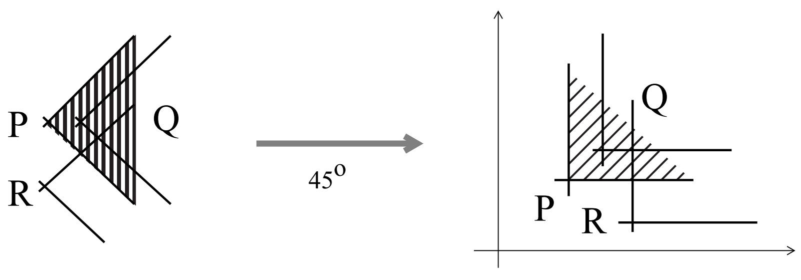

P
Q
R
45°
Q
P
R

Figure 6: Isomorphism of vector time order and light cone order.

The rubber band transformations mentioned in Section 2, which change the metrical but not the topological structure of time diagrams, and hence leave the causality relation invariant, correspond to the so-called Lorentz transformations of space-time. These are causality preserving transformations of coordinates which leave the structure of light cones invariant.

With vector timestamps it is possible to check for two given events whether they are causally related or not. The cone structure of space-time yields a similar criterion. If c denotes the maximum attainable speed, then two events e_1 and e_2 with coordinates (x_1, t_1) and (x_2, t_2) are causally related if one of them lies within the cone of the other one, i.e., if $c^2\,(t_2 - t_1)^2 \; - \; (x_2 - x_1)^2 \; \geq \; 0$. This can be generalized to n-dimensional spaces ($n \geq 2$) by defining the relation

$$U \leq V \quad :\Longleftrightarrow \quad \sum_{i=1}^{n-1} (u_i - v_i)^2 \;\leq\; c^2 (u_n - v_n)^2 \;\;\wedge\;\; u_n \leq v_n$$

for two arbitrary space-time points $U = (u_1, ..., u_n)$, $V = (v_1, ..., v_n)$. Here, the n-th dimension takes the role of time. Usually, c is normalized to 1, which means that the spatial units are "light years" if time is measured in "years". One can show that this light cone relation is a partial order and that for any two points there exists a point which is greater than both of them.

Interestingly, for $n = 2$ vector time order and light cone order are basically identical. By considering normalized past cones with $c = 1$ and turning the whole system counterclock-wise by 45° (see Figure 6), one sees that the "future" of a point P consists of all those points which lie above and to the right of P. Hence, for two points $P = (p_1, p_2)$, $Q = (q_1, q_2)$ one has $P \leq Q$ if and only if $p_1 \leq q_1$ and $p_2 \leq q_2$. According to Definition 4.2 this can be written in a "vectorial way" as $(p_1, p_2) \leq (q_1, q_2)$. This means that for $n = 2$ vector time and space-time have basically an identical structure!

It is also possible to find a suitable interpretation of light cones in vector time. The past cone $\widehat{P}$ of a space-time point P consists of all those points Q which can affect P, i.e., $\widehat{P} = \{Q \,|\, Q \leq P\}$ where '$\leq$' is the light cone order defined above. Formally, this corresponds to our definition $\downarrow\! e \;=\; \{e' \,|\, e' \preceq e\}$ of the causal past of an event (cf. Definition 4.6). In a similar way it would be possible to define the "causal future" $\uparrow\! e \;=\; \{e' \,|\, e \preceq e'\}$ of an event which corresponds to the future cones. Because in a time diagram no event of $\downarrow e$ can be to the right of e, the "past cone" $\downarrow e$ of an event is indeed completely located to the left of the event. Usually, however, such an abstract cone does not have the perfect shape of a geometrical cone. But since the abstract cones $\downarrow e$ or $\uparrow e$ represent consistent cuts (cf. Theorem 4.7), it is possible to draw an abstract cone in a cone-like way by using a similar construction as sketched in the proof of Theorem 2.4.

For our real world relativistic space-time yields an image of reality which is more accurate than linear "standard time". Its close analogy to vector time is not accidental – both models assume that cause and effect are not atomic and that hence some other (causally independent) events may happen in between. The analogy indicates that vector time might be the "correct" model of time for distributed systems. In fact, vector time is a useful concept in the theory of distributed computations [16]. It has been used in distributed debugging systems [7, 5], for measuring the degree of concurrency of distributed computations [3, 15], to implement consistent views in distributed databases [18, 8, 21, 6, 10], to compute globally consistent snapshots [11], and to implement so-called causal broadcasts [2]. Several optimizations to reduce the amount of information of the general scheme [17, 13] or to generalize the principle have been proposed [1]; as Charron-Bost showed in [4], however, there is in the general case no more compact representation of the causality structure than vector time.

The author would like to thank Bernadette Charron-Bost and Reinhard Schwarz for helpful remarks on an earlier version of the paper and for fruitful discussions on the general concepts of causality and vector time.

References

[1] M. AHUJA, T. CARLSON, GAHLOT A. Passive-space and Time View of Computing. Technical report, Computer and Information Science Research Center, Ohio State University, Columbus, 1991.

[2] K. BIRMAN, A. SCHIPER, P. STEPHENSON. Lightweight Causal and Atomic Group Multicast. Technical Report TR 91-1192, Computer Science Department, Cornell University, 1991.

[3] B. CHARRON-BOST. Combinatorics and Geometry of Consistent Cuts: Application to Concurrency Theory. *In: J-C Bermond, M. Raynal (eds) Proc. of the 3nd International Workshop on Distributed Algorithms, Springer-Verlag LNCS, pp. 45-56*, 1989.

[4] B. CHARRON-BOST. Concerning the Size of Logical Clocks in Distributed Systems. *Information Processing Letters*, 39:11–16, 1991.

[5] J. FIDGE. *Dynamic Analysis of Event Orderings in Message-Passing Systems.* PhD thesis, Department of Computer Science, The Australian National University, 1989.

[6] M. FISCHER, A. MICHAEL. Sacrificing Serializability to Attain High Availability of Data in an Unreliable Network. In *ACM SIGACT-SIGOPS Symp. on Principles of Database Systems*, pages 70–75, 1982.

[7] D. HABAN, W. WEIGEL. Global Events and Global Breakpoints in Distributed Systems. *Proc. 21st Hawaii International Conference on System Sciences, Vol. II, pp. 166-175*, 1988.

[8] D.B. JOHNSON, W. ZWAENEPOEL. Recovery in Distributed Systems Using Optimistic Message Logging and Checkpointing. *Journal of Algorithms*, 11(3):462–491, 1990.

[9] L. LAMPORT. Time, Clocks, and the Orderings of Events in a Distributed System. *Comm. of the ACM 21:7, pp. 558-565*, 1978.

[10] B. LISKOV, R. LADIN. Highly-Available Distributed Services and Fault-Tolerant Distri buted Garbage Collection. *Proc. of the 5th ACM Symposium on Principles of Distributed C omputing*, pages 29–39, 1986.

[11] F. MATTERN. Verteilte Basisalgorithmen. *Springer-Verlag, Informatik-Fachberichte Bd. 226*, 1989.

[12] F. MATTERN. Distributed Control Algorithms (Selected Topics). *F. Ozguner (Ed.) Parallel Computing on Distributed Memory Multiprocessors, Springer-Verlag*, 1992.

[13] S. MELDAL, S. SANKAR, J. VERA. Exploting Locality in Maintaining Potential Causality. Technical Report CSL-TR-91-466, Computer Systems Laboratory, Stanford University, 1991.

[14] M. RAYNAL. *Synchronisation et Etat Global dans les Systèmes Répartis.* Eyrolles, 1992.

[15] M. RAYNAL, M. MIZUNO, M. NEILSEN. Synchronization and Concurrency Measures for Distributed Computations. Technical Report 610, IRISA, University of Rennes, France, 1991.

[16] R. SCHWARZ, F. MATTERN. Detecting Causal Relationships in Distributed Computations: In Search of the Holy Grail. Technical Report 215/91, Department of Computer Science, University of Kaiserslautern, Germany, 1991.

[17] M. SINGHAL, A. KSHEMKALYANI. An Efficient Implementation of Vector Clocks. Technical report, Dept. of Computer and Information Science, The Ohio State University, 1991.

[18] R. STROM, S. YEMINI. Optimistic Recovery in Distributed Systems. *ACM Transactions on Computer Systems*, 3(3):204–226, 1985.

[19] G. TEL. *Topics in Distributed Algorithms*, volume 1 of *Cambridge International Series on Parallel Computing.* Cambridge University Press, Cambridge, U.K., 1991.

[20] M. ČAPEK. Time–Space Rather than Space–Time. *Diogenes*, (123):30–49, 1983.

[21] K. VENKATESH, T. RADHAKRISHNAN, H.F. LI. Optimal Checkpointing and Local Recording for Domino-Free Rollback Recovery. *Information Processing Letters 25, pp. 295-303*, 1987.