# Event Spaces and Their Linear Logic

Vaughan Pratt
Computer Sci. Dept., Stanford, CA 94305, USA
pratt@cs.stanford.edu

April 25, 1991

**Abstract**

Boolean logic treats disjunction and conjunction symmetrically and algebraically. The corresponding operations for computation are respectively nondeterminism (choice) and concurrency. Petri nets treat these symmetrically but not algebraically, while event structures treat them algebraically but not symmetrically.

Here we achieve both via the notion of an event space as a poset with all nonempty joins representing concurrence and a top representing the unreachable event. The symmetry is with the dual notion of state space, a poset with all nonempty meets representing choice and a bottom representing the start state. The algebra is that of a parallel programming language expanded to the language of full linear logic, Girard's axiomatization of which is satisfied by the event space interpretation of this language.

Event spaces resemble finite dimensional vector spaces in distinguishing tensor product from direct product and in being isomorphic to their double dual, but differ from them in distinguishing direct product from direct sum and tensor product from tensor sum. Event spaces also resemble complete semilattices, differing only in the substitution of top for bottom.

This work was supported by the National Science Foundation under grant number CCR-8814921.

## 1 Introduction

A basic problem in the theory of concurrent computation is to define and reconcile nondeterminism and concurrency, respectively the disjunction and conjunction of behavior. Now in classical logic disjunction and conjunction have an appealing algebraic definition: we take them to be respectively the join $x \vee y$ and meet $x \wedge y$ of a distributive lattice, a concept axiomatizable with finitely many equations such as $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$.

Join and meet are completely symmetric: the order dual of a distributive lattice, obtained by interpreting $\leq$ as $\geq$, is still a distributive lattice with join and meet interchanged. When the lattice has a complement operation, as in a Boolean algebra, this duality is explicitly expressed in the language via De Morgan's laws, $(x \vee y)' = x' \wedge y'$ and $(x \wedge y)' = x' \vee y'$.

To date there has been no model of concurrency that combines algebra and symmetry the way Boolean algebra does for logic. The places and transitions of Petri nets are symmetric and act respectively as disjunction and conjunction. However there is no algebra of Petri nets whose operations correspond to respectively nondeterministic choice and concurrence of nets while preserving

that symmetry. Conversely Winskel's event structures and prime algebraic domains [Win86] have an attractive algebraic theory but are not at all symmetric.

In this paper we give an algebraic model of computation that treats choice and concurrency symmetrically. It subsumes event structures with a minimum of machinery, while its algebra constitutes a parallel programming language whose syntax and axiomatics are most succinctly described as being those of full linear logic.

Our starting point is Winskel's notion of event structure, based on Birkhoff's duality of finite posets and distributive lattices [Bir33], extended to the infinite case by Stone [Sto36, Sto37]. This duality associates to each poset $S = (X, \leq)$ the distributive lattice $2^{S^{\mathrm{op}}}$ consisting of the order ideals of $S$, and to each distributive lattice $A = (X, \vee, 0, \wedge, 1)$ the poset $2^{A^{\mathrm{op}}}$ consisting of the lattice ideals of $A$, in such a way that $A$ is isomorphic to $2^{S^{\mathrm{op}}}$ if and only if $S$ is isomorphic to $2^{A^{\mathrm{op}}}$. We offer a brief tutorial on this duality at the end of this paper and a longer one elsewhere [Pra91].

In the context of computation this duality has a natural interpretation in terms of schedules and automata, whose elements denote respectively events and states. This interpretation was found by Nielsen, Plotkin and Winskel [NPW81], who then extended the duality to incorporate a symmetric irreflexive binary *conflict* relation $\#$, with $x\#y$ forbidding the occurrence of $x$ together with $y$. They required conflict to be *persistent*: if $x\#y$ and $y \leq z$ then $x\#z$. This extension, which they called an *event structure*, was subsequently developed much further by Winskel [Win86]. In the presence of conflict the automaton dual to $(X, \leq, \#)$ is the subposet of the distributive lattice dual to $(X, \leq)$ consisting of the conflict-free states, those not containing both $x$ and $y$ when $x\#y$. A more general notion of conflict removes the binary restriction on conflict, allowing a set of events to be in conflict even though no proper subsets of that set are in conflict. The dual automaton is called *coherent* just when conflict remains binary.

Now the automaton dual to an event structure may not be a distributive lattice, due to conflict having deleted the top and other nonempty joins. However all nonempty meets are retained. Figure 2 below gives examples of such automata.

With this in mind let us take the existence of nonempty meets and the empty join as *definitive* of the kind of automaton we want. Specifically we define a *state space* to be a nonempty poset with all nonempty meets (including infinite meets). We provide an explicit constant symbol $q_0$ for the universal meet or empty join or bottom. A *map* of state spaces is a function between state spaces preserving nonempty meets and $q_0$.

We then define the obvious dual notion to state space, which we shall call an *event space*. We define this to be a nonempty poset with all nonempty joins (including infinite joins), and having a constant symbol $\infty$ for the universal join or empty meet or top which will serve as the permanently deferred or impossible event. A *map* of event spaces is a function between event spaces preserving nonempty joins and $\infty$. (Ordinarily we would call $q_0$ $\perp$ and $\infty$ $\top$, but we have here reserved these symbols for respectively the two-event event space and the two-state state space.)

Section 2 treats events organized by a nonempty-join operation $\bigvee$ and a constant $\infty$ into an event space, an algebra of events. This is the "inside" view of event spaces, namely a theory whose individuals are events. In this view we see that event spaces subsume event structures, by permitting an arbitrary set of events to be in conflict. But whereas event structures schedule only atomic events, event spaces may schedule compound events.

Section 3 treats event spaces organized by several arithmetic operations into an "algebra" of event spaces. This is the "outside" view of event spaces, namely a theory whose individuals are event spaces.

The operations of this algebra are simultaneously those of a natural parallel programming language and of linear logic. Their event space interpretation turns out to satisfy Girard's axiomatization of linear logic perfectly.

The duality of event and state spaces is more than just a matter of interchanging meet and join in their respective definitions. There is a specific bijection between event spaces and state spaces[1] which has a strikingly simple description. The event space $A^\perp$ dual to a given state space $A$ is constructed from $A$ by removing $q_0$ from the bottom and reinstalling it as $\infty$ at the top. When $q_0$ is removed some meets disappear; it is a theorem that we shall prove later that when it is reinstalled at the top all nonempty joins absent from $A$ now appear. Performing the inverse operation on any event space, namely removing top and reinstalling it as bottom, similarly yields the dual state space, removing some joins and supplying all missing nonempty meets.

This leads to the even more striking conclusion that in this framework, with the exception of the initial state $q_0$ and the unattainable event $\infty$, *individual events and individual states are the same notion.* Every state can be viewed as an event and vice versa.

The difference emerges only when we consider events or states in combination with each other. States combine via meet while events combine via join. The meet of a set of states constitutes the initial state for that set, while the join of a set of events amounts to their union, collecting all the events of the set into one compound event.

Event spaces behave very much like vector spaces. In particular they admit linear transformations: the set $b \multimap a$ of linear transformations from event space $b$ to event space $a$ itself forms an event space. (We denote both event spaces and state spaces by lower case variables $a, b, c$ when treating their "external" algebra.) There is also a direct product $a \times b$, a tensor product $a \otimes b$, and a dual space $a^\perp = a \multimap \perp$ satisfying $a^{\perp\perp} = a$ analogous to the vector space $V^*$ dual to $V$.

However event spaces differ from vector spaces in three ways. First, $V^{**} = V$ holds only for finite dimensional vector spaces $V$, whereas $a^{\perp\perp} = a$ holds for all event space. Second, the default notion of distance between two vectors in a vector space is their difference, imposing an inflexible triangle *equality* and requiring the addition of a separately defined norm to relax this to a more flexible triangle inequality. In an event space however, taking distance to be the truth of $x \leq y$ yields an inherently flexible triangle inequality, namely reflexivity. (We develop this relationship between posets and metric spaces in concurrency modeling considerably further elsewhere [CCMP91].) Third, sum and product of vector spaces degenerate to the same operation, and similarly for tensor sum and tensor product, whereas for event spaces these are all distinct operations, respectively concurrence and choice, and cointeraction and interaction.

An even closer cousin to the event space is the complete semilattice, a poset with all joins. The notion of event space is obtained from that of complete semilattice by replacing the empty join or bottom by the empty meet or top. As P. Johnstone has noted [Joh78], the category **CSLat** of complete semilattices is self-dual, $\mathbf{CSLat} \cong \mathbf{CSLat}^{\mathrm{op}}$.[2] Johnstone obtains this very neatly as a direct consequence of Freyd's adjoint functor theorem. The category **Ev** of event spaces is likewise self-dual (proof left to another paper). While we do not know how to make this self-duality a corollary of the adjoint functor theorem, one is led to the thought that there may be a variant of the adjoint

[1] We are here regarding event spaces and state spaces as being defined only up to isomorphism, which is to say that the bijection is really between isomorphism classes of event spaces and isomorphism classes of state spaces. In terms of categories we have only an equivalence of categories, not an isomorphism.

[2] Johnstone neglected to state a stronger property of this self-duality, that the isomorphism puts isomorphic objects in correspondence. Without this his corollary that $a + b = a \times b$ in **CSLat** is unsound, witness event spaces.

functor theorem in which adjoints preserve certain balanced mixtures of limits and colimits, of which $\mathbf{Ev} \cong \mathbf{Ev}^{\mathrm{op}}$ is a corollary. The main use of this self-duality in this paper is in showing that $\mathbf{Ev}$ is symmetric monoidal, $a \otimes b = b \otimes a$.

The above account obtains event spaces as dual to state spaces, which in turn are a natural abstract formulation of the dual of event structures. However event spaces were not actually found in this way. Instead they arose while contemplating the duality of bipointed sets and cubical sets (as top-and-bottom-less Boolean algebras), and the self-duality of complete semilattices (posets with all joins). The *modus operandi* of the former, in combination with the delicate balance of the latter, prompted us to explore the consequences of small perturbations of the latter. It is remarkable that so small a change as interchanging top and bottom in complete semilattices could have such a beneficial impact both inside (subsumption of event structures) and out (our calculus of event spaces).

## 2 Definition and Examples

An *event space* is a nonempty partially ordered set $S = (X, \leq)$ such that every nonempty subset $Y \subseteq X$ has a join or least upper bound $\bigvee Y$ in $X$. The join $\bigvee X$ of the whole set must be the top element, which we denote $\infty$.

The following Hasse diagrams[3] depict the nine event spaces having up to four points (events).

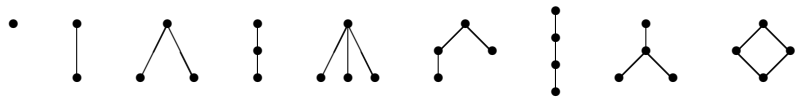


Figure 1. Event Spaces with at most 4 events.

A *state space* is the dual notion to event space, namely a nonempty partially ordered set $A = (X, \leq)$ such that every nonempty subset $Y \subseteq X$ has a meet or greatest lower bound $\bigwedge Y$ in $X$. The meet $\bigwedge X$ of the whole set must be the bottom element, which we denote $q_0$ (the traditional notation for start state) or $-\infty$.

The state space $S^{\perp}$ dual to a particular event space $S$ is obtained by deleting $\infty$ at the top and adjoining $q_0$ at the bottom. This operation is equivalent for Figures 1.1-1.7 to taking the order dual (turning the diagram upside down), and for 1.8 and 1.9 interchanging them as well as inverting them, as shown in Figure 2.

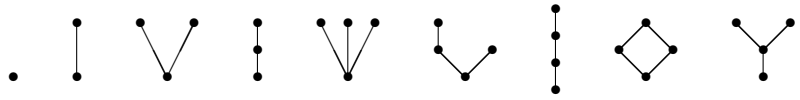


Figure 2. State Spaces Corresponding to Figure 1.

[3] A *Hasse diagram* depicts a partially ordered set as an undirected graph whose edges have an implicit upward orientation. Hence $x \leq y$ is represented by the existence of a path from point $x$ leading upwards to point $y$. Since posets are reflexive there is an implicit self-loop at every vertex, and since they are transitive there is an implicit edge from $x$ to $z$ for every upward path $x \leq y \leq z$. Deleting $y$ from the poset does not mean that $x \leq z$ no longer holds but rather that the implicit edge from $x$ to $z$ in the Hasse diagram now needs to be made explicit.

The number of event spaces (and hence state spaces) of each cardinality from 1 to 10 is 1, 1, 2, 5, 15, 53, 222, 1078, 5994, 37622. (It is straightforward to verify that this is also the number of lattices of each cardinality from 2 to 11: just add a bottom element to every event space, and delete it from every lattice.) The corresponding numbers for posets with 0 to 9 points are 1, 1, 2, 5, 16, 63, 318, 2045, 16999, 183231. I am grateful to Pat Lincoln, Vineet Gupta, and the problem solving class CS 204 for computing and verifying these figures.

Let us now illustrate the use of event spaces with some naive theories of the notion of family. Figure 3 depicts these theories while Figure 4 depicts the corresponding state spaces.

We begin with the free event space on a set $\{m, w, c\}$, figure 3(a). (We define "free" later.) Let us take 3(a) to denote the unconstrained evolution of a family consisting of a man, a woman, and a child. $\infty$ represents the event that will never happen, while the remaining events represent the entry of a nonempty subset of those three individuals into the family. For example $m \vee w$ is the event where both the man and the woman enter the family. Equivalently it may be viewed as the state in which the man and the woman are both in the family.

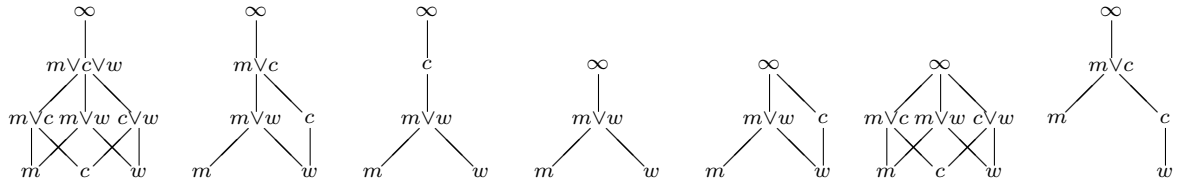


Figure 3. Event Space Representation of Some Theories of Families.

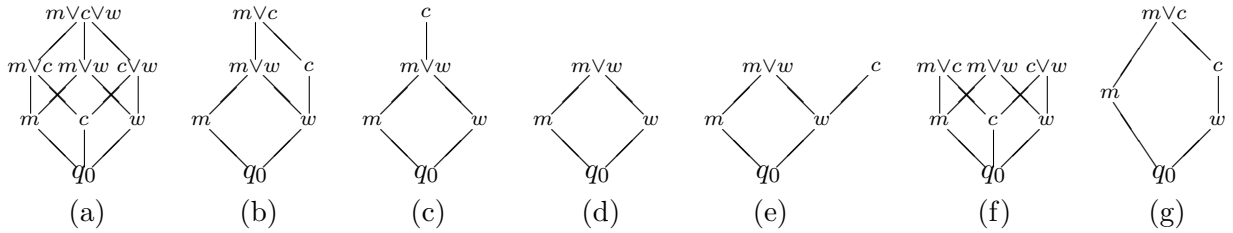


Figure 4. State Spaces Corresponding to Figure 3

We regard the man and woman as each having entered the family when each has made up their mind to do so. When both have made up their mind we have the *concurrent* event $m \vee w$, namely an event $x$ representable as the join of two incomparable events. Let us call a nonconcurrent event other than $\infty$ *atomic*. In Figure 1, $\infty$ is concurrent in 3, 5, 6, and 9 while the only non-$\infty$ concurrent event is the one in 8 just below $\infty$. In Figure 3 the concurrent events are those with $\vee$ in their label, along with $\infty$ in 3(e) and 3(f).

One naturally supposes that one of $m$ or $w$ must have made up their mind to join the family no earlier than the other. This would imply that $m \vee w$ would be equal to one of $m$ or $w$, or to both if they decided simultaneously. That the events $m$, $w$, and $m \vee w$ are all distinct shows that we have not made this supposition. In this naive account of the family we shall identify $m \vee w$ with marriage, which we could take to be a protocol for establishing common knowledge of the respective decisions of the man and the woman to so unite ("Do you take this...?"). As such it constitutes from at least a legal viewpoint the earliest moment at which the concurrency of $m$ and $w$ is observable. This is at once an event and a state.

The process in 3(a) is very general, allowing for the possibility of an unrelated man, woman, and already born child conspiring to form a family of three, with the child being adopted. Events $m \vee c$ and $c \vee w$ denote the mutual agreement of those pairs, which along with the agreement $m \vee w$ we are referring to as marriage are all assumed to be prerequisites to the formation of the whole family, namely the event $m \vee c \vee w$.

Let us now rule out adoption and consider henceforth only children born into a family. Call this requirement *motherhood*, a consequence of which is $w \leq c$.

Application of motherhood to 3(a) yields 3(b), removing some concurrency but not all: we still have the concurrency between the marriage $m \vee w$ and the child $c$ inferrable from the parallelogram. We infer a potential race condition, as in the closing scene of *Irma La Douce* in which the marriage is pronounced only seconds before the child is born.

We may describe the passage from 3(a) to 3(b) as either the result of deleting $c$ and $m \vee c$ from (a) and renaming $c \vee w$ to $c$ and $m \vee c \vee w$ to $m \vee c$, or as the result of identifying $c \vee w$ with $c$ and $m \vee c \vee w$ with $m \vee c$ in (a). The former views (b) as a *subspace* of (a), the latter as a *quotient space*.

What makes the latter view particularly attractive is that it can be identified with an equation, namely $c \vee w = c$, synonymous with $w \leq c$. Let us fix this idea with some more examples.

The concurrency in 3(b) can be removed by specifying whether or not the child is born in wedlock, respectively 3(c) and 3(g). (Distinguish this requirement from that of the man being the biological father of the child, which is outside the scope of this example.) For now consider just 3(c). (This does describe Irma's marriage, but had the timing been sufficiently close as to raise doubts or disagreement, the observers might feel obliged to report having *observed* only 3(b) even while agreeing among themselves that one of 3(c) or 3(g) (discussed below) must surely have taken place in God's eyes.)

As with the relationship of 3(b) to 3(a) we can regard 3(c) as either a subspace or a quotient of 3(b). The latter view is captured by the equation $m \vee c = c$ or $m \leq c$, a child may only be born into a family already equipped with a man. In combination with motherhood this becomes $m \vee c \vee w = c$, or $m \vee w \leq c$.

Suppose now that the man requires that the marriage if any be childless; this gives rise to 3(d). While it is clear how 3(d) can be seen as a subspace of 3(c), it is not clear how it could be a quotient. This is where the inaccessibility of $\infty$ enters: we may regard 3(d) as the result of identifying $c$ and $\infty$ in 3(c), hence forbidding $c$ not by its absence but by its position at the end of time. This is not of itself a conflict in our sense, but it leads naturally into that notion.

*Conflict.* The event $\infty$ is the last event, which we view as never happening. When $\bigvee Y = \infty$ for any set $Y$ of events not containing $\infty$ we say that $Y$ is *in conflict*. The examples in Figure 1 that contain sets of events in conflict are 3, 5, 6, and 9, while those in Figure 3 are (e) and (f). The corresponding automata in Figures 2 and 4 have more than one maximal element (final state).

Note that it is possible for a large set of events to be in conflict without any subset being in conflict. An example of this appears as Figure 3(f), constructed from figure 3(a) by identifying $m \vee w \vee c$ and $\infty$. This identification puts the set $\{m, w, c\}$ in conflict without putting any of its subsets in conflict. Concretely this means that any two out of the three pairs $m \vee c$, $m \vee w$, and $c \vee w$ are permitted, but nothing larger. Thus this schedule amounts to a program making a three-way decision determining which of the three is eventually left out.

Conflict is also meaningful in continuous situations. Consider the point $x = 1 - 1/(1 + t)$ moving as a function of time $t$ ranging over the nonnegative reals. The point starts at 0 and moves towards 1

without ever reaching it. Its behavior is therefore modeled by the event space $[0,1]$ of reals with its standard order, with $\bigvee$ interpreted as sup and $\infty$ as 1. Any subset of $[0,1]$ with sup 1, e.g. $[0,1)$, is in conflict, that is, at no time can the set of points visited thus far have sup 1.

On the other hand the point $x = t$ where $t$ ranges over $[0,1]$ reaches $x = 1$ in unit time. A suitable event space describing this situation consists of the unit interval as before but with the integer 2 adjoined to take over the role of $\infty$ from 1. Now no subset of $[0,1]$ is in conflict.

In this continuous example conflict was used only to encode inaccessibility of a limit, not an actual choice. Conflict in a finite set of events on the other hand necessarily entails a choice. Consider imposing the childless-marriage requirement on Figure 3(b), in the absence of wedlock. This yields 3(e), which permits the woman to have a child out of wedlock. This presents her with a *conflict*: she can have either a child or a man in her family, but not both.

Event spaces do just as well as event structures in scheduling atomic events and specifying which sets are in conflict. However event spaces can also schedule concurrent events, which event structures cannot. For example we may write $c \leq m \vee w$ to indicate that the child is born out of wedlock, or more precisely that it enters the family before the marriage.

Applying this constraint to the free event space 3(b) yields the non-free space 3(g). Note that the corresponding state space 4(g) is a well-known nondistributive lattice. We will explain free spaces in the next section.

Example 3(g) expresses marriage to a woman with a child born out of wedlock and amounts to Figure 3(b) plus the condition $c \leq m \vee w$, yielding Figure 3(g). Here the man marries the unit consisting of the woman and her child, which we take to be linearly ordered (if discretely ordered we obtain another non-free example). The essential feature of 3(g) is that there is only one event following $m$, $c$, and $w$, namely $m \vee c \vee w$, or $m \vee c$ given that $w \leq c$ here. That is, we wish to dispense with the distinctions between the two-element subsets of $\{m, w, c\}$ making up 3(b) (or 3(a) in the absence of $w \leq c$), and simply say that when $m$ is marrying a woman with a child there is only one resulting concurrent event. For when we draw such distinctions as in 3(b) then we admit the possibility of $c$'s entering the family later and hence legitimately instead of as an instantaneous consequence of $m$'s marriage to $w$.

## 3 Event Maps and Free Event Spaces

*Event maps.* Given event spaces $S = (X_S, \bigvee_S, \infty_S)$ and $T = (X_T, \bigvee_T, \infty_T)$, an *event map* $f : S \to T$ is a homomorphism of event spaces. That is, it is a function $f : X_S \to X_T$ satisfying $f(\bigvee_S Y) = \bigvee_T f(Y)$ for all nonempty $Y \subseteq X_S$, and $f(\infty_S) = \infty_T$.

This is in contrast to a monotone function or poset map, which is a function $f : (X_S, \leq_S) \to (X_T, \leq_T)$ such that if $x \leq_S y$ then $f(x) \leq_T f(y)$. Now if $x \leq_S y$ then $x \vee_S y = y$ so $f(x) \leq_T f(x) \vee_T f(y) = f(x \vee_S y) = f(y)$. Hence every event map is a poset map.

We do not however have the converse. In Figure 1, the map from example 1 to example 2 that takes the single element of 1 to the lower element of 2 gives an example of a poset map that is not an event map because it fails $f(\infty_1) = \infty_2$. Another such example is the map from 3 to 2 that takes the two lower elements of 3, call them $x$ and $y$, to the lower element of 2, for then we have $f(\infty_3) = f(x \vee_3 y) = f(x) \vee_2 f(y) \neq \infty_2$.

Implicit in our definition of event map as a homomorphism of event spaces is the *signature* of an

event space, that is, the "official" operations and constants. The signature consists of the operation $\bigvee$ and the constant $\infty$. An event map is a homomorphism of event spaces by virtue of preserving the operations and constants of the signature.

Note the more substantive role $\infty$ now plays. Previously $\infty$ was merely a synonym for $\bigvee X$. However the most we can infer from $f(\bigvee_S Y) = \bigvee_T f(Y)$ is that $f(\bigvee_S X_S) = \bigvee_T f(X_S)$. But $\bigvee_T f(X_S)$ need not be $\bigvee_T X_T$, witness the map $f$ from example 1 to example 2 in Figure 1 taking the one element of 1 to the lower element of 2.

One use of event maps is to describe the process of adding events. An *injective* event map $f : S \to T$, one for which $f(x) = f(y)$ implies $x = y$, makes $T$ the result, up to isomorphism, of adding events to $S$. The requirement $f(\infty_S) = \infty_T$ then means that every added event must precede $\infty$. This confers on $\infty$ the status of an absolutely final event, one that no event can ever follow.

When $f(x) = \infty$ this means only that $f$ has pushed $x$ sufficiently far into the future as to be unable to distinguish it from $\infty$. It does not lessen the absoluteness of $\infty$ itself, which can never be mapped to anything earlier than $\infty$.

The class of event spaces and their associated event maps form a category we shall call **Ev**. We have avoided categorical language for the sake of reaching a larger audience. If you are fluent in category theory you should have little difficulty translating the concepts of this paper into your terms, though you might understandably be impatient with the length this has added to the paper.

*Free Event Spaces.* We have seen that event spaces are everything that posets are and more. The event spaces that are not more are called the *free* event spaces. Those that are more are not free by virtue of being constrained by equations, either $\bigvee Y = \infty$ expressing conflict or $\bigvee Y = \bigvee Z$ scheduling compound events, e.g. $c \le m \vee w$ in 3(g). Such constraints are not representable by order alone, in contrast say to $m \vee w \le c$ in the free space 3(c) which is mere poset information, namely $m \le c$ and $w \le c$.

Associated with each event space $S = (X, \bigvee, \infty)$ is its *underlying* poset $U(S) = (X, \le)$ where $x \le y$ just when $x \vee y = y$. Another associated poset is its *compact* subposet, consisting of the *compact* elements of $S$, namely those elements $x \ne \infty$ such that for any nonempty $Y \subseteq X$, $x = \bigvee Y$ implies $x \in Y$. In the finite case the compact elements of $S$ are just those other than $\infty$ with at most one edge leading up to it in the Hasse diagram of $S$. In the seven examples of Figure 3 these are in every case the elements labeled with just a letter, one of $m$, $c$, or $w$. Exercise: identify the 19 compact elements in Figure 1.

A *free* event space $F(P)$ *on* a poset $P$ is an event space that behaves just like $P$ with respect to labeling. More formally, for every event space $S$ (serving as a source of labels) the set of event maps $f : F(P) \to S$ are in bijective correspondence with the set of monotone functions or poset maps $f' : P \to U(S)$, where $f'$ is just the restriction of $f$ to $P$. We may "find" $P$ inside $F(P)$ as just its compact elements.

In Figure 1, event spaces 3, 5, 6, and 9 are not free because in each case $\infty$ is the join of the set of compact elements. Hence if we take $S = 2$, the 2-element event space with elements $0 \le \infty$, and label the compact elements 0, then any extension of this labeling to an event map must map $\infty$ to both 0 and $\infty$; hence this poset map from $P$ to $U(S)$ has no corresponding event space map from $F(P)$ to $S$, showing that the event space in question is not free. The remaining spaces of Figure 1 are free.

In Figure 3 the same argument shows that spaces (e) and (f) are not free. In space (g), still taking $S = 2$, label $m, w, c$ respectively $0, 0, \infty$. Now $m \vee c$ is the join of both $\{m, w\}$ and $\{m, c\}$ and hence

must be labeled both 0 and $\infty$. The remaining spaces of Figure 3 are free.

This characterization of $F(P)$ as the event space equivalent to its compact subposet is not very constructive. An explicit construction of $F(P)$ given only $P$ is as the set of nonempty order ideals[4] of $P$, with $\bigvee$ interpreted as union, together with a separate top element $\infty$.

The correspondence between the maps from $F(P)$ to $S$ and those from $P$ to $U(S)$ yields two interesting maps. Taking $S = F(P)$, the identity event map from $F(P)$ to itself corresponds to a poset map called $\eta_P : P \to UF(P)$, the *embedding* of $P$ in $F(P)$. Taking $P = U(S)$, the identity poset map from $U(S)$ to itself corresponds to an event map called $\varepsilon_S : FU(S) \to S$. If we think of the points of $FU(S)$ as terms whose variables are the points of $S$ then $\varepsilon_S$ is the *evaluation* map giving the value of each term.[5]

If $\varepsilon_S$ is evaluation then its restriction to $S$, the variables of $FU(S)$, must be just the identity on $S$. This is expressed formally by the requirement that the composition of $\varepsilon_S$, more precisely of $U(\varepsilon_S) : UFU(S) \to U(S)$, with $\eta_{U(S)} : U(S) \to UFU(S)$ be the identity on $U(S)$. Less intuitive but equally clear formally, we require that the composition of $\varepsilon_{F(P)} : FUF(P) \to F(P)$ with $F(\eta_P) : F(P) \to FUF(P)$ (the image of $\eta_p$ under $F$) be the identity on $F(P)$.

If $P$ was obtained as the compact subposet of $F(P)$ then $\eta_P$ is merely the corresponding inclusion. If in addition $P$ happened to have all nonempty joins this would make it the underlying poset $U(S)$ of some event space $S$, in which case $\varepsilon_S : FU(S) \to S$ would be the identity on $P$ and take $\infty$ to itself; the remaining elements of $F(P)$ are of the form $x = \bigvee_{F(P)} Y$ for $Y \subseteq P$ (exercise) and so are mapped by $\varepsilon_S$ to $\bigvee_S Y$, within $S$.

If on the other hand we obtained $F(P)$ by explicit construction from $P$ via order ideals then $\eta_P$ must be given as part of the construction: it is the function taking each $x \in P$ to the principal order ideal generated by $x$. Either way $\eta_P$ is injective, that is, it embeds $P$ in $UF(P)$, allowing us to think of $P$ as a subposet of $UF(P)$ whether or not it actually is.

*Set-Free Event Spaces.* Substituting "set" for "poset" everywhere in the definition of free event space yields the notion of free event space on a set instead of on a poset. We distinguish this notion of free via the term set-free. Of the five free event spaces in Figure 1, only 1, 2, and 8 are set-free, while in Figure 3 (b) and (c) are the two free event spaces that are not set-free. If instead of sets we had substituted event structures for posets we would arrive at the notion of free event space on an event structure. Figures 3(f) and 3(g) are the only counterexamples we have seen to free event spaces on an event structure, and 3(f) is only a counterexample if we restrict to coherent event structures, those with only a binary conflict relation #.

*Conflict-Free Event Spaces.* We shall call those event spaces such that $\bigvee Y = \infty$ implies $\infty \in Y$ *conflict-free*. The corresponding notion of underlying object $U(a)$ is just that of forgetting the constant $\infty$. The matching $F(a)$ adjoins $\infty$ at the top of $a$. Note that in the absence of $\infty$ it becomes possible to have an empty event space, the free event space on which is then the one-point space.

Conflict-free event spaces are as the name suggests free of conflicts.

[4] An *order ideal* of a poset $(X, \leq)$ is a subset $Y \subseteq X$ such that if $x \leq y$ then $y \in Y$ implies $x \in Y$. The union of any set of nonempty order ideals is clearly a nonempty order ideal. An order ideal is *principal* when it has a top, equivalently when it is the set of elements less or equal to a single element, said to *generate* that ideal.

[5] Categorically speaking these are of course the components of an adjunction $F \dashv U$ with unit $\eta$ and counit $\varepsilon$. However it is not necessary to know any category theory to understand these components and how they fit together.

# 4 A Calculus of Event Spaces

We turn now from the logic of events in a single event space to a calculus of event spaces. This calculus will resemble the algebra of natural numbers under addition $x+y$, multiplication $xy$, exponentiation $x^y$, and "negation" $0^x$, and similarly that of propositions under disjunction $p \vee q$, conjunction $p \wedge q$, implication $q \to p$, and negation $\neg p = p \to 0$.

The corresponding operations for event spaces will be tensor sum $a \oplus b$, tensor product $a \otimes b$, implication $b \multimap a$, and negation $a^\perp$. However we will also be interested in the underlying poset $U(a)$ of each event space $a$. We therefore expand this algebra with a parallel list of poset operations, namely direct sum $a + b$, direct product $a \times b$, and implication $b \Rightarrow a$. We use the same negation operation $a^\perp$ for both event spaces and posets. The zeroary forms of $\oplus$, $\otimes$, $+$, and $\times$ are the respective constants $\perp$, $\top$, 0, and 1.

This appears to call for two sorts of terms in this calculus, one for event spaces and one for posets. However we can take advantage of the fact that $F(P)$ "behaves like" $P$ and work with $FU(a)$ rather than $U(a)$, so that all objects are officially event spaces even when trying to behave like posets. We write $FU(a)$ as $!a$. Since all other monotone (i.e. covariant) operations have duals, ! may as well too, so we abbreviate $(!(a^\perp))^\perp$ to $?a$.

We have described $a^\perp$ as a state space when $a$ is an event space, again seeming to call for two sorts. For the time being we shall take $a^\perp$ to be the order dual of what we previously took it to be so that $a^\perp$ will remain an event space. In a later section we shall introduce the notion of sign as a homogeneous way of bringing in state spaces, analogous to the usual treatment of positive and negative integers as one sort rather than two.

These nine operations and four constants are exactly those of Girard's linear logic [Gir87], a connection we will return to in a later section. The operations $a + b$, $a \times b$, and $a \otimes b$ arise naturally as operations of a parallel programming language, also treated in its own section.

We now interpret these operations and constants for event spaces. We define the negation $a^\perp$ of $a$ as the order dual of all of $a$ except $\infty$, which remains at the top; we prove below that this yields an event space. (Applying this form of negation to Figure 1 yields not Figure 2 but rather its order dual; all that changes in Figure 1 is then that 8 and 9 are interchanged.) We have already seen the definitions of $!a$ and $?a$. And we define the constants 0 and 1 as both being the one-point event space, Figure 1.1, and $\perp$ and $\top$ as the two-point event space, Figure 1.2. (We distinguish 0 from 1 and $\perp$ from $\top$ later using signed event spaces.)

We now define the three arithmetic operations for each of event spaces and posets, which we may tabulate thus.

| | Implication via Maps | Product via Curry | Sum via De Morgan |
|---|---|---|---|
| Event Space Arithmetic : | $b \multimap a$ | $a \otimes b$ | $a \oplus b$ |
| Poset Arithmetic : | $b \Rightarrow a$ | $a \times b$ | $a + b$ |

First we define column 1, the implications. We take $b \multimap a$ to be the poset of all event maps from $b$ to $a$, ordered *pointwise*, that is $f \leq g$ just when $\forall x[f(x) \leq g(x)]$; we show shortly that this poset is in fact an event space. We take $b \Rightarrow a$ to be the poset of all poset maps from $U(b)$ to $U(a)$, which as we have seen corresponds to the poset of all event maps from $FU(b)$ to $a$, i.e. $b \Rightarrow a \;=\; !b \multimap a$.

Next we define column 2, the products. For event spaces the Curry principle or *s-m-n* theorem or

residuation for event spaces is $(b \otimes c) \multimap a = c \multimap (b \multimap a)$, uniquely (up to isomorphism) defining the *tensor product* $a \otimes b$. For posets this principle becomes $(b \times c) \Rightarrow a = c \Rightarrow (b \Rightarrow a)$, uniquely (up to isomorphism) defining the *direct product* $a \times b$.

Lastly we define column 3, the sums. Define *tensor sum* $a \oplus b$ to be $(b^\perp \otimes a^\perp)^\perp$, the De Morgan dual of tensor product. Similarly define *direct sum* to be the De Morgan dual of direct product, $a + b = (a^\perp \times b^\perp)^\perp$.

Tensor product $a \otimes b$ may be defined alternatively as $(b \multimap a^\perp)^\perp$, via the reasoning $a \otimes b = ((a \otimes b) \multimap \perp)^\perp = (b \multimap (a \multimap \perp))^\perp = (b \multimap a^\perp)^\perp$. Direct product $a \times b$ may be defined alternatively as having an underlying set consisting of the Cartesian product of the underlying sets of $a$ and $b$. Its top is $(\infty_a, \infty_b)$ while join is computed pointwise: the join of $Y$ is $(\bigvee_a Y_a, \bigvee_b Y_b)$ where $Y_a = \{x \mid \exists y[(x, y) \in Y\}$ and similarly for $Y_b$. It is readily seen that $a \times b$ so defined is an event space. While we have no better elementary definition of $a + b$ than as the De Morgan dual of $a \times b$, categorically speaking $a + b$ and $a \times b$ are naturally described as respectively coproduct and product.

An important observation is $b \multimap a = a^\perp \multimap b^\perp$, whose routine but tedious elementary proof we omit; we hope to find a more elegant proof along the lines of P. Johnstone's proof of **CSLat** $\cong$ **CSLat**$^{\mathrm{op}}$ as mentioned in the introduction. Formulating this as $b \multimap a^\perp = a \multimap b^\perp$ immediately yields $a \otimes b = b \otimes a$.

This phenomenon of having two parallel lists of arithmetical operations is not unusual, occurring for example with relation algebras [JT48], vector spaces, and relevant logic [Dun86]. Concentrating on both rows of column 2, products, we find respectively a structured and unstructured product as follows for each of these examples. For relations we have composition $R; S$ and intersection $R \cap S$. For vector spaces we have tensor product $U \otimes V$ and direct product $U \times V$. For relevant logic we have cotenability $a \circ b$ and conjunction $a \wedge b$.

In each case we sometimes want to do arithmetic with the structure present and sometimes without it. The latter arithmetic is performed in effect on the set of elements of the structure, i.e. its underlying set, which is how we tend to work with event spaces from a set theory perspective rather than category theoretically. (Thus this is how to do set theory within category theory.) In our calculus we have chosen the underlying poset instead, although the axioms of the calculus would look about the same had we chosen the underlying set. In either case the underlying objects form a cartesian closed category (tensor product is direct product), in contrast to event spaces for which tensor product as interaction is distinct from direct product as choice.

**Theorem 1** *The poset $a^\perp$ constructed from $a$ as above by inverting (taking the order dual of) all of $a$ but $\infty_a$ is an event space.*

*Proof*: The construction automatically equips $a^\perp$ with a top, namely $\infty$. It remains to show that the join of any nonempty subset $Y$ of $a^\perp$ exists in $a^\perp$. The following closely parallels the argument that a complete semilattice is a complete lattice.

If $\infty$ is in $Y$ then $\bigvee Y = \infty$. Otherwise consider the set $\mathcal{L}Y$ of lower bounds on $Y$ in $a$. If $\mathcal{L}Y$ is empty then the only upper bound on $Y$ in $a^\perp$ will be $\infty$, which is then the least upper bound on $Y$, the desired join $\bigvee Y$. If $\mathcal{L}Y$ is nonempty then it has a join $j$ in $a$. Now every element of $Y$ is an upper bound on $\mathcal{L}Y$, whence $j$ as the least upper bound must be a lower bound on $Y$. Hence it is the greatest lower bound on $Y$. But then it is the least upper bound on $Y$ in $a^\perp$. ∎

Although we defined negation explicitly in elementary terms, there is an equivalent algebraic defini-

tion of $a^\perp$.

**Theorem 2** $a^\perp = a \multimap \perp$.

*Proof*: We identify each event map $f : a \multimap \perp$ with its kernel $f^{-1}(0)$ (0 being the lower element of $\perp$), which we call an *event space ideal*. The empty set is an event space ideal, and corresponds to the top map, which is therefore $\infty$ in $a \multimap \perp$. Since event maps preserve joins, every nonempty event space ideal must contain its join. But this makes it a principal order ideal, of which there is exactly one per element of $a$. Conversely every principal order ideal determines a map of $a \multimap \perp$ except the one generated by $\infty_a$, which corresponds to a map taking $\infty$ to 0, which is not an event map. We may therefore put the non-$\infty$ elements of $a$ in one-one correspondence with the non-top maps of $a \multimap \perp$; let $f_x$ denote the map corresponding to $x$. It is clear that $x \leq y$ if and only if $f_y \leq f_x$. Hence the poset of nontop maps of $a \multimap \perp$ is the order dual of the poset of non-$\infty$ elements of $a$, while the top map stays at the top. ∎

**Theorem 3** *The poset $b \multimap a$ consisting of all event maps from $b$ to $a$, ordered pointwise, is an event space.*

*Proof*: The top event map is of course just the constantly top map, satisfying $f(x) = \infty_a$ for all events $x \in b$.

Given any nonempty set $F$ of event maps $f : b \to a$, take its join $\bigvee F$ to be the function $g : b \to a$ defined as $g(x) = \bigvee_{f \in F} f(x)$ for each event $x$ in $b$. It remains to show that $g$ is an event map.

Now $g(\infty_b) = \bigvee_{f \in F} f(\infty_b) = \bigvee \infty_a = \infty_a$, whence $g$ preserves $\infty$. That $g$ preserves nonempty joins follows thus.

$$\begin{aligned} g(\bigvee_{y \in Y} y) &= \bigvee_{f \in F} f(\bigvee_{y \in Y} y) \\ &= \bigvee_{f \in F} \bigvee_{y \in Y} f(y) \\ &= \bigvee_{y \in Y} \bigvee_{f \in F} f(y) \\ &= \bigvee_{y \in Y} g(y). \end{aligned}$$

∎

## 5 Signed Event Spaces

As things stand, if $a$ is an event space so is $a^\perp$. However it is very natural to view $a^\perp$ as the state space dual to $a$, not by inverting the non-$\infty$ portion of $a$ but instead by simply moving $\infty$ from top to bottom and renaming it $q_0$, as we saw in Figures 2 and 4.

From this perspective the non-$\infty$ events turn into states *without changing the partial order*. What does change are the joins and top, which are replaced by meets and a bottom. This is easily seen when it is noticed that this new notion of $a^\perp$ is no more than the order dual of the old notion of $a^\perp$ as an event space. Since the latter had the nonempty joins and top of an event space, the former as its order dual must have all nonempty meets and a bottom.

We define a *signed* event space to be a pair $(a, s)$ where $a$ is an event space and $s \in \{0, 1\}$ is a "sign bit" giving the direction of time. When the sign bit is zero the associated event space $a$ is a schedule, its points are events, and $x \leq y$ means that $x$ occurs before $y$. When the sign bit is one, $a$ becomes a state space or automaton whose points are now states, and the temporal relationship that we write as $x \leq y$ in the event space interpretation of $a$ is now to be viewed as the transition $y \geq x$ from $y$ to $x$.

We extend the calculus of event spaces to a calculus of signed event spaces via the usual Boolean calculus of zero and one. Sum, product, and exponential, whether direct (poset) or tensor (event space), are respectively disjunction, conjunction, and implication. Negation exchanges 0 and 1. The comonad $!a$ has nothing to forget and so acts as the identity. We call the resulting category **2**.

The category **Sev** of signed event spaces is then the product $\mathbf{Ev} \times \mathbf{2}$.

In **Sev**, negation turns event spaces into state spaces and vice versa. The maps from an event space $s$ to a state space $a$ must be the states of the state space $s \multimap a$, since $0 \multimap 1 = 1$. This makes excellent sense: the state of an event space as a whole is given by the states of each of its events. When $a = \top$, the automaton whose two states can be viewed as *not-done* and *done*, $s \multimap \top$ is the automaton whose states indicate which *atomic* states of $s$ are done. But since $s$ also includes its compound events the states of $s \multimap \top$ are actually just the events of $s$.

Now in **Ev** we have $0 = 1$ and $\perp = \top$. In **2** however we have $0 = \perp$ while $1 = \top$. Hence in $\mathbf{Ev} \times \mathbf{2}$ all four constants are distinct. This removes the one objectionable degeneracy we are aware of in **Ev**.

The theory of signed event spaces can be seen to be the intersection of the theory of event spaces with the theory of 0 and 1. Since the only significant isomorphisms in the difference that we are aware of are $0 = 1$ and $\perp = \top$, this intersection then accomplishes for us only that much.

Had we performed this intersection for the category **CSLat** of complete semilattices to yield the category of signed complete semilattices, or posets with all joins, we would have removed in addition the degeneracy of sum and product in **CSLat**. I do not know whether $\mathbf{CSLat} \times \mathbf{2}$ contains any isomorphisms representable with such formulas that are not also present in $\mathbf{Ev} \times \mathbf{2}$. However if $!a$ were (the event space mimicking) the underlying set instead of the underlying poset of $a$ we would have a distinction, since then $!a$ and $?a$ would be isomorphic complete atomic Boolean algebras, the set of atoms of $!a$ and the set of coatoms of $?a$ both being the underlying set of $a$. That degeneracy is not removed by the sign trick since it is present in **2** where both ! and ? are the identity.

Since the only difference between a schedule and its dual automaton is the location of the one bound, respectively at the end or the beginning, it is natural to ask, why not just put in both bounds and be done with the distinction altogether between schedules and automata?

One problem this unification creates is that $b \multimap a$, which is at the heart of our calculus, gains new maps since there is now an additional element in $a$ to send elements of $b$ to. This in turn throws off the nice balance of the calculus: $b^\perp$ need have neither bound.

Indeed this is the principle behind the duality of bipointed sets, sets with two distinguished elements, and Boolean algebras without top or bottom. Contemplation of this duality, which Bill Lawvere

suggested to me in a phone conversation as a simple construction of the theory of cubical sets, led me to the idea of evening things up by moving one or the other of the bounds over to the other side of the duality, and doing the same with one of the nonempty meets or the nonempty joins. When all the joins including bottom as the empty join are on the same side this yields complete semilattices, for which sum and product are the same operation up to isomorphism, and there is no mechanism for representing conflict. The other way round yields event spaces, in which sum denotes concurrence while product denotes choice (whence they had better not be isomorphic), and conflict is now representable as the equality of top with a join.

Another change with such a unification is that the product or sum of a schedule with an automaton now has a quite different meaning. In **Sev** such a product orients the schedule opposite to the automaton so as to put their bounds at the same end before performing the addition or multiplication. In the unified view product is computed with the bounds at opposite ends. Whether this different meaning is better or worse is a very interesting question that clearly depends on why one cares about the product of an automaton with a schedule in the first place. If all we want is an internally consistent calculus, signed event spaces achieve this much. If for whatever reason we want temporal coherence in mixed products, some other approach is called for.

*Partial Distributive Lattices.* We have been investigating the possibility of a unification of schedules and automata into a single much larger category that includes as well most of the other basic structures encountered in the context of the Birkhoff-Stone duality, including sets, posets, distributive lattices, topological spaces, Stone spaces, frames, and locales. This is the category **PDL** of *partial distributive lattices*, complete distributive lattices whose meet and join operations are only partial but satisfy the distributive lattice axioms where defined. A representative example of a PDL is a subset of a power set, with meet and join defined as in the power set just when they remain in the subset.

The power set on $n$ elements has 1, 2, 4, 16, 256, ...subsets starting at $n = -1$ (taking $2^{-1} = 0$); these are grouped into respectively 1, 2, 4, 12, 70, ... PDL isomorphism classes. Once a PDL appears for a given $n$ it remains present at all larger $n$, with a single exception: the "inconsistent" PDL occurs only at $n = 1$. This is the only PDL satisfying $0 = 1$, i.e. bottom coincides with top; it is dual to the empty PDL. The 70 classes at $n = 3$ partition as 1+3+5+11+20+16+9+4+1 by size from 0 (the empty subset) to 8 (the whole 3-cube) elements. The internal hom is obvious on reflection. Duality, defined by taking the complete two-element lattice as the dualizer in the usual way, is involutary for some but not all PDL's, e.g. the unit interval of reals with all meets and joins, and the 3-cube less one atom and its complementary coatom (found by V. Gupta). We hope to report further on properties and applications of this fascinating and potentially very useful category in a future paper.

*Final states.* We define a *final state* of an automaton to be the inf of a maximal chain of the automaton, necessarily a member of that chain. Maximal chains without such infs represent nonterminating computations. This uses the poset structure to code with an inf what Hoare has represented with $\surd$, and the absence of which we represented as the limbo state $\Lambda$ in an early process logic paper [Pra79], coded here with an infinite chain lacking an inf.

# 6 Programming with Arithmetic

We give here the sense in which sum is concurrence, product is choice, and tensor product is interaction or orthocurrence [Pra86, CCMP91].

For schedules defined as posets, concurrence $P \| Q$ can be defined simply as juxtaposition or poset coproduct $P + Q$, a point of view advocated by Grabowski [Gra81] and the author [Pra85, Pra86]. For event spaces that "mimic" a poset $P$, namely the free event space $F(P)$, coproduct remains the appropriate definition because $F$ is a left adjoint and hence preserves (distributes over) coproducts. In particular the event space $F(P)+F(Q)$ mimicking $P+Q$ is $F(P+Q)$, which by this distributivity is $F(P) + F(Q)$, the coproduct of the spaces mimicking $P$ and $Q$ respectively.

Figure 3(b) for example is the concurrence of event $m$ with the sequence $w \leq c$. As a poset this concurrence has just those three events, but as an event space it has besides $\infty$ the additional events $m \vee w$ and $m \vee c = m \vee c \vee w$. Figure 3(d) is the coproduct of the two two-element spaces $\{m, \infty\}$ and $\{w, \infty\}$, while 3(a) is the coproduct of 3(d) with $\{c, \infty\}$.

Choice as product $a \times b$ is a more delicate notion. If we split up $a$ as $\alpha + \{\infty\}$ and $b$ as $\beta + \{\infty\}$ then

$$a \times b = \alpha \times \beta + \alpha \times \{\infty\} + \{\infty\} \times \beta + (\infty, \infty)$$

**Theorem 4** *For conflict-free event spaces $a, b$, any two elements of $a \times b$ in conflict must be one from each of $\alpha \times \{\infty\}$ and $\{\infty\} \times \beta$.*

*Proof*: Consider $(x, y)$ and $(x', y')$. Neither can be top, $(\infty, \infty)$. Their join is $(x \vee x', y \vee y')$. Since $a$ and $b$ are conflict-free we must have either $x$ and $y'$ being $\infty$, or $x'$ and $y$, as claimed. ∎

If we are given $a \times b$ unlabelled, with $a$ and $b$ conflict-free, even without the help of labels it is possible to identify two maximally conflict-free sets such that each element of one is in conflict with each element of the other. We may then infer that these must be $\alpha \times \infty$ and $\infty \times \beta$, which we may identify respectively with components $a$ and $b$ of the choice. We may think of the unconflicted portion $\alpha \times \beta$ as the portion of $a \times b$ responsible for making the choice of $a$ or $b$.

The simplest example of such a choice is Figure 1.9, which is the product of two copies of Figure 1.2. The choice of which events to perform is between the two lower left events or the two lower right events, as the dual Figure 2.9 makes explicit. Since bottom is common to these we regard it as part of the decision making process. The two remaining events on the left and right, which together are in conflict, are then what we will have chosen to do. We were able to draw this distinction between decision making and resulting decision solely on the basis of the abstract structure of the event space, without reference to labels on events.

Interaction or othocurrence $a \otimes b$ is the notion of interacting particle systems such as colliding galaxies and a sequence of trains passing through a sequence of stations. Interaction is defined, and example applications given, in [Pra86] (where it is notated $a \times b$ because the coincidence $a \times b = a \otimes b$ in **Pos** led us to believe interaction was ordinary product) and in [CCMP91] by which time we had understood the distinction between direct and tensor product. A characteristic of interaction is that the interaction of two linear or one-dimensional posets is a rectangular or two-dimensional poset.

What we shall verify here is that $\otimes$ is the proper operation for the interaction of those event spaces mimicking posets, namely free event spaces. Given two such event spaces $F(P)$ and $F(Q)$, their the event space mimicking the interaction $P \times Q$ of posets $P$ and $Q$ is $F(P \times Q)$. But this is equal to $F(P) \otimes F(Q)$, showing that at least in the case of free event spaces interaction is $\otimes$.

There are two programming connectives that do not fit into this system of arithmetic in an obvious way, namely demonic choice and concatenation. Demonic choice of $a$ and $b$ removes $\infty_a$ and $\infty_b$, takes the disjoint union of the result (as posets), then puts one copy of $\infty$ back. This description

makes it clear that demonic choice is self-dual: $(a \sqcup b)^\perp = a^\perp \sqcup b^\perp$. In Figure 1, $3 = 2 \sqcup 2$, $5 = 3 \sqcup 2$, and $6 = 2 \sqcup 4$. Choice as product is angelic in that the common part of the choice represents decision making; in demonic choice there is no provision for decision making, one takes pot luck. This version of demonic choice, which may not agree with everyone's understanding of the notion (I would appreciate advice on this), is equivalent to the choice having been made "in the beginning," i.e. in the start state, a computational form of original sin.

Concatenation $a; b$ is not self-dual. Thus far we have given all definitions in terms of event spaces. Concatenation is more intuitively understood in terms of state spaces. The concatenation $a; b$ of state spaces $a$ and $b$ joins a separate copy of $b$ onto each final (maximal) state of $a$, by identifying the start state of $b$ with that final state. In Figure 2, $4 = 2; 2$, $7 = 2; 4$, and $9 = 2; 3$. If $a$ is the unit real interval $[0, 1)$ open at 1 it has no final state, i.e. does not terminate, whence $a; b = a$.

To define $a; b$ for event spaces, first define dual concatenation $a \fatsemi b$ of event spaces as the order dual of state space concatenation. Thus in Figure 1, $4 = 2 \fatsemi 2$, $7 = 2 \fatsemi 4$, and $8 = 2 \fatsemi 3$. Then take $a; b = (a^\perp \fatsemi b^\perp)^\perp$ (where $a^\perp$ is event space dual, invert all but $\infty$). Thus in Figure 1, $4 = 2; 2$, $7 = 2; 4$, and $9 = 2; 3$.

We have not investigated the nature of these operations for nonfree event spaces, other than to note that they are well-defined for them. However the nonfree examples we have considered suggest that the interpretation of $a + b$ as concurrence, $a \times b$ as choice, and $a \otimes b$ as interaction continue to make good sense.

# 7 Event Spaces as a Model of Linear Logic

Linear logic was introduced by Girard in 1987 [Gir87, Gir89]. Its language corresponds exactly to the arithmetic portion of our calculus and thus excludes demonic choice and concatenation. (Though we have been faithful to the basic language of linear logic our choice of notation for its operations and constants is closer to that of Seely [See89] and Barr [Bar91a, Bar91b] than of Girard.)

In assessing the significance of linear logic it is tempting to focus on the closed but not cartesian closed aspect. But that aspect is already present in relation algebras and relevant logic. What is novel and beautiful about linear logic is that, starting from a closed category, in our case event spaces, it adds two useful operations. First an abstraction operator $!a$ which forgets enough operations to turn the closed structure into a cartesian closed structure better suited to elementary ("element-oriented") logic as opposed to categorical logic, in our case posets though we could just as well have chosen sets, and which permits expression of such relationships as $!(a \times b) = !a \otimes !b$ and $b \Rightarrow a = !b \multimap a$. Second an involutary duality $a^\perp$ that gives every monotone operation its De Morgan dual ($\otimes$, $\times$, and $!$ are dual to $\oplus$, $+$, and $?$ respectively) and also makes other useful connections such as $b \multimap a = b^\perp \oplus a$.

We dispense here with the traditional Gentzen sequent formulation of linear logic in favor of a more direct categorical description; consult Seely [See89] for details of the connection between the two.

A model of linear logic is a category $C$ with two closed monoidal structures having $b \multimap a$ and $b \Rightarrow a$ as the respective "internal hom(functor)s" or implications. The left adjoint to $b \multimap -$ is $b \otimes -$, a symmetric operation ($a \otimes b = b \otimes a$), while the left adjoint to $b \Rightarrow -$ is $b \times -$. Furthermore $\times$ is ordinary product in $C$, which is to say that $\Rightarrow$ confers a cartesian closed structure on $C$ (whence $\times$ is also symmetric). There exists a dualizing object $\perp$, with the dual $a^\perp$ defined as $a \multimap \perp$ and satisfying $a^{\perp\perp} = a$. The products $\otimes$ and $\times$ have respective De Morgan duals $a \oplus b = (b^\perp \otimes a^\perp)^\perp$ and $a + b = (b^\perp \times a^\perp)^\perp$, the latter being coproduct. The units of $\otimes$, $\times$, $\oplus$, and $+$ are respectively

$\top = \bot^\bot$, a final object 1, $\bot$ (already mentioned), and an initial object 0.

As Seely points out [See89], the two structures are related by a *comonad* or cotriple $(!, \varepsilon, \delta)$ consisting of an endofunctor $! : C \to C$, and for each object $a \in ob(C)$ a counit $\varepsilon_a : !a \to a$ and a comultiplication $\delta_a : !a \to !!a$. The comonad satisfies certain conditions that achieve the effect of an adjunction between $C$ and a category of underlying objects of objects of $C$ [BW85, p.95].

From this we may derive many other properties, for example:

$$\begin{aligned}
!a \otimes !b &= ((!a \otimes !b) \multimap \bot)^\bot \\
&= (!b \multimap (!a \multimap \bot))^\bot \\
&= (b \Rightarrow (a \Rightarrow \bot))^\bot \\
&= ((a \times b) \Rightarrow \bot)^\bot \\
&= (!(a \times b) \multimap \bot)^\bot \\
&= !(a \times b) \\
a \otimes b &= ((a \otimes b) \multimap \bot)^\bot \\
&= (b \multimap (a \multimap \bot))^\bot \\
&= (b \multimap a^\bot)^\bot
\end{aligned}$$

In the case of event spaces, $C = \mathbf{Ev}$, we have already described the interpretations of all the operations, and it should be clear that the above conditions on a model of linear logic have all been met. It remains only to complete the specification of the comonad, namely $(FU, \varepsilon, F\eta U)$ where $FU = \mathbf{Ev} \xrightarrow{U} \mathbf{Pos} \xrightarrow{F} \mathbf{Ev}$, $\varepsilon$ (the family $\langle \varepsilon_a \rangle$ of evaluation maps) is the counit of the adjunction $F \dashv U$, and $\eta$ (the family $\langle \eta_a \rangle$ of embeddings of generators) is the unit of the same adjunction.

We have argued in outline that the full category **Ev** of event spaces, a naturally arising class of structures, and their homomorphisms (in the standard sense) is a model of full linear logic. Moreover product and sum are distinct. We would appreciate hearing about any other equally simple model of linear logic that meets all these conditions.

# 8 The Birkhoff-Stone Duality of Schedules and Automata

For the benefit of strangers to the Birkhoff-Stone duality of schedules and automata we briefly review its essential features in this section. More detailed tutorials may be found elsewhere including [Pra91].

The simplest case of the duality takes a schedule to be a set $X$ of events, and its dual automaton to be its power set $2^X$, forming a Boolean algebra under the operations of union, intersection, and complement relative to $X$. We take the empty subset of $X$ to be the start state of the automaton and $X$ itself to be the final state. To view $2^X$ as an automaton in the usual sense, associate to each pair $Y \neq Z$ of states satisfying $Y \subset Z \subseteq X$ a transition corresponding to (i.e. labeled with) the nonempty set $Z - Y$ of events. A path from the initial to the final state via such transitions determines a partition of $X$ into a finite sequence of nonempty blocks, each block representing the parallel execution of the events in that block, with the path as a whole then describing an execution of all the events of $X$ in some order. An alternative view of $2^X$ as an automaton imposes the additional restriction that transitions be labeled only with singletons, corresponding to a strictly sequential execution of the events of $X$.

We may usefully enhance such a schedule by equipping it with a partial order $\leq$. The effect of this

enhancement on the dual automaton $2^X$ is to eliminate certain states, leaving only those states $Y$ that are order ideals of $X$, i.e. $x \leq y$ and $y \in Y$ implies $x \in Y$.

An order ideal may be viewed equivalently as a monotone function from $X^{\mathrm{op}}$, the order dual of $X$, to **2**, the linearly ordered two-element poset. These functions are the characteristic functions of the corresponding order ideals. Thus we may take the automaton to be the set $2^{X^{\mathrm{op}}}$ of all such functions from $X^{\mathrm{op}}$ to **2**. In the special case when $X$ is discretely ordered ($x \leq y$ implies $x = y$, i.e. $X$ is just a set) this automaton is a Boolean algebra, but in general it is a distributive lattice, lacking only the complement operation characteristic of a Boolean algebra.

When $X$ is infinite more can be said. For $X$ discrete, $2^{X^{\mathrm{op}}}$ is a complete atomic Boolean algebra. Complete means that every set $\mathcal{Y}$ of subsets of $X$ has a join or sup, namely its union $\bigcup \mathcal{Y}$, and a meet or inf, namely its intersection $\bigcap \mathcal{Y}$. In particular the empty set of subsets has join $\emptyset$ (bottom) and meet $X$ (top), while the set $2^{X^{\mathrm{op}}}$ of all subsets of $X$ has join $X$ and meet $\emptyset$. An atom is an element $x$ such that $x \wedge y$ is either $x$ or bottom. Atomic means that every element except bottom is above some atom.

In general $2^{X^{\mathrm{op}}}$ is a *profinite* distributive lattice [Joh82, p.250], one characterization of which is a distributive lattice bearing the Stone topology. This means that it is complete in the same sense as for Boolean algebras, but the notion of atomic is generalized to the requirement that every element be the join of a set of compact elements. An element $x$ is *compact* when it is the join only of sets containing $x$; equivalently, when the join of the set of elements strictly less than $x$ is itself strictly less than $x$.

Stone's original duality put the Stone topology on the set [Sto36] or poset [Sto37] side. In the latter case this view of a poset topologized with a Stone topology was not apparent from Stone's paper and was found much later by Priestley [Pri70].

In the case of the distributive lattice $2^{X^{\mathrm{op}}}$ the compact elements are exactly those subsets $Y$ of $X$ for which there exists an element $y \in Y$ such that no proper subset of $Y$ containing $y$ belongs to $2^{X^{\mathrm{op}}}$. In terms of states, these are the states containing an event absent from all earlier states, i.e. the earliest opportunity for that event to occur.

# 9 Acknowledgments

Rob van Glabbeek turned my attention from the overly abstract $n$-dimensional complexes of [Pra91] to more concrete and plausible cubical sets, and a phone conversation with Bill Lawvere then further turned it from cubical sets to their dual objects bipointed sets. The connection between that duality and Stone duality then dawned on me, and the additional intuition conveyed by the self-duality of **CSLat** [Lat76, Joh78] led quickly to **Ev**, though the concrete interpretation of the phenomena in **Ev** in terms of unreachable events and initial states took time. That $!a$ even made sense in **Ev** was completely unobvious to me until I had digested the relevant parts of the papers of Barr [Bar91a] and Seely [See89], after which I wondered why I did not see it right away. Thanks also to Michael Barr for much helpful email correspondence concerning monads and comonads. Guo Qiang Zhang, Stefano Kasangian, and Jeremy Gunawardena provided valuable feedback on the paper.

## References


[Bar91a] M. Barr. $*$-Autonomous categories and linear logic. *Math Structures in Comp. Sci.*, 1(2), 1991.

[Bar91b] M. Barr. Accessible categories and models of linear logic. *J. Pure and Applied Algebra*, 1991. To appear.

[Bir33] G. Birkhoff. On the combination of subalgebras. *Proc. Cambridge Phil. Soc*, 29:441–464, 1933.

[BW85] M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer-Verlag, 1985.

[CCMP91] R.T Casley, R.F. Crew, J. Meseguer, and V.R. Pratt. Temporal structures. *Math. Structures in Comp. Sci.*, 1(2):179–213, July 1991.

[Dun86] J.M. Dunn. Relevant logic and entailment. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume III, pages 117–224. Reidel, Dordrecht, 1986.

[Gir87] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir89] J.-Y. Girard. Towards a geometry of interaction. In *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics*, pages 69–108, held June 1987, Boulder, Colorado, 1989.

[Gra81] J. Grabowski. On partial languages. *Fundamenta Informaticae*, IV.2:427–498, 1981.

[Joh78] P.T. Johnstone. A note on complete semilattices. *Algebra Universalis*, 8:260–261, 1978.

[Joh82] P.T. Johnstone. *Stone Spaces*. Cambridge University Press, 1982.

[JT48] B. Jónsson and A. Tarski. Representation problems for relation algebras. *Bull. Amer. Math. Soc.*, 54:80,1192, 1948.

[Lat76] D. Latch. Arbitrary products are coproducts in complete ($\vee$-)semilattices. *Algebra Universalis*, 6:97–98, 1976.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri nets, event structures, and domains, part I. *Theoretical Computer Science*, 13, 1981.

[Pra79] V.R. Pratt. Process logic. In *Proc. 6th Ann. ACM Symposium on Principles of Programming Languages*, pages 93–100, San Antonio, January 1979.

[Pra85] V.R. Pratt. Some constructions for order-theoretic models of concurrency. In *Proc. Conf. on Logics of Programs, LNCS 193*, pages 269–283, Brooklyn, 1985. Springer-Verlag.

[Pra86] V.R. Pratt. Modeling concurrency with partial orders. *Int. J. of Parallel Programming*, 15(1):33–71, February 1986.

[Pra91] V.R. Pratt. Modeling concurrency with geometry. In *Proc. 18th Ann. ACM Symposium on Principles of Programming Languages*, pages 311–322, January 1991.

[Pri70] H.A. Priestley. Representation of distributive lattices. *Bull. London Math. Soc.*, 2:186–190, 1970.

[See89] R.A.G Seely. Linear logic, $*$-autonomous categories and cofree algebras. In *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics*, pages 371–382, held June 1987, Boulder, Colorado, 1989.

[Sto36] M. Stone. The theory of representations for Boolean algebras. *Trans. Amer. Math. Soc.*, 40:37–111, 1936.

[Sto37] M. Stone. Topological representations of distributive lattices and brouwerian logics. *Časopis Pěst. Math.*, 67:1–25, 1937.

[Win86] G. Winskel. Event structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, LNCS 255*, Bad-Honnef, September 1986. Springer-Verlag.