Concurrent Kripke Structures

Vineet Gupta*
Dept. of Computer Science
Stanford University, CA 94305
vgupta@cs.stanford.edu

January 9, 2005

Abstract

We consider a class of Kripke Structures in which the atomic propositions are events. This enables us to represent worlds as sets of events and the transition and satisfaction relations of Kripke structures as the subset and membership relations on sets. We use this class, called event Kripke structures, to model concurrency. The obvious semantics for these structures is a true concurrency semantics. We show how several aspects of concurrency can be easily defined, and in addition get distinctions between causality and enabling, and choice and nondeterminism. We define a duality for event Kripke structures, and show how this duality enables us to convert between imperative and declarative views of programs, by treating states and events on the same footing. We provide pictorial representations of both these views, each encoding all the information to convert to the other.

We define a process algebra of event Kripke structures, showing how to combine them in the usual ways—parallel composition, sequential composition, choice, interaction and iteration. Various properties of these connectives like associativity and distributivity are proved. We then show that Winskel's event structures can be embedded in the class of event Kripke structures, and define partial synchronous composition, the primary connective for event structures, for event Kripke structures, and show its equivalence to Winskel's definition.

Note: This work was done jointly with Vaughan Pratt, Stanford University.

*This work was supported by ONR under grant number N00014-92-J-1974, a gift from Mitsubishi and an Accel Partners Fellowship.

1 Event Kripke Structures

In the past decade, various desirable properties of models of concurrent processes have been discussed: branching time, true concurrency, action refinement, disjunctive enabling, conflict, interchangeable state and event based views, real time consistency etc. Various models, having many of these properties have been proposed: Petri nets [Pet62], synchronization trees [Mil80], Mazurkiewicz traces [Maz77], pomsets [Gra81, Pra82], event structures [NPW81, Win86], causal automata [Gun91, Gun92] etc.

A careful examination of these models reveals that most of them exhibit concurrent behavior by explicitly inserting concurrency into a basically sequential model, such as by having many tokens in a network or multiple strings being read concurrently. Thus they end up defining concurrency instead of modelling it, and one can never be sure whether the definition is good enough. For example, in event structures, conflict and enabling are primitives, but other primitives could be necessary to capture concurrency fully. In this paper we attempt to discover what true concurrency is from a model of concurrency, rather than defining it a priori, in the same vein as discovering logic as the theory of Boolean algebras rather than trying to characterize it with a long list of equations. We start from a well known model for temporal logic, a Kripke structure, and use it to derive our model.

A Kripke structure $(W, \models, \Pi_0, \rightarrow)$ consists of a binary transition relation $w \rightarrow w'$ between states $w, w' \in W$, and a binary satisfaction relation $w \models p$ between states in W and atomic propositions $p \in \Pi_0$, extended by induction on the height of formulas to compound propositions $p \wedge q, \neg p, \ldots \in \Pi$.

We study a special class of Kripke structures which looks more like event structures[NPW81]. The relevant property of an event here is that once it becomes true, it stays true. We restrict the elements of Π_0 to events, that is $\forall w, w' \in W, w \models p$ and $w \rightarrow w'$ implies $w' \models p$. Now the transition relation between states can be derived from the satisfaction relation by $w \rightarrow w'$ iff $\forall p, w \models p \Rightarrow w' \models p$. In fact we can treat each state as a set of the atomic propositions which are true in it (we identify states which have the same sets of propositions true in them, by extensionality). Thus we define an *event Kripke structure*, referred to as an eks, as a pair (E, Q), where E is a set of events and $Q \subseteq 2^E$ a set of states. Then $q \models e$ iff $e \in q$ and $q \rightarrow q'$ iff $q \subset q'$. An equivalent way of describing an eks is as a $|E| \times |Q|$ boolean matrix, with each column being the characteristic function of a state.

In addition to having all the desirable properties of models of concurrency mentioned above, eks's can distinguish between enabling an action and causing an action, a distinction which has not been made in any of the above models. In fact in some of the above models, causality has been deliberately erased by insisting on algebraic lattices [Dro89], presumably to match Petri nets and

domain theory. Our model shows that not only is this unnecessary, it actually simplifies mathematics to allow this distinction. One has only to compare the definition of an event domain in [Dro89] with the definition of an eks to be convinced of this.

In the next sections, we will study some properties of event Kripke strutures. We show how aspects of concurrent behavior are naturally present in an eks, and give a process algebra to combine eks's in various ways, showing how these operations correspond to the usual operations for concurrent processes. Finally we connect them with event structures, showing how any event structure can be represented as an eks.

These structures have previously been studied by Barr[Bar79], as instances of Chu's construction on sets, and by Lafont and Streicher[LS91] as games over 2. Brown and Gurr[BG90] have used similar structures to study Petri nets.

1.1 Theory of an eks

An eks (E, Q) is a set Q of subsets of 2^E. We can therefore write it out as a formula in Disjunctive Normal Form in propositional logic, with variables E and clauses Q, with each clause corresponding to a $q \in Q$ containing all the variables in E, negating all the ones that are not present in q. We call this formula Γ. For example, $\Gamma((\{a,b,c\},\{\phi,\{a\},\{ab\},\{abc\}\})) = \bar{a}\bar{b}\bar{c} \vee a\bar{b}\bar{c} \vee ab\bar{c} \vee abc$, writing $\bar{a}$ for $\neg a$ and abc for $a \wedge b \wedge c$. Since there is no bound on the cardinality of the sets E and Q, we will allow infinite conjunctions and disjunctions in our formulas. So an eks can be equivalently represented as an infinitary boolean propositional formula Γ. This is usually the most compact representation of an eks, though the matrix representation is computationally the most efficient.

The theory of an eks is now defined to be all infinitary propositional logic formulae ϕ such that $\Gamma \rightarrow \phi$ is a tautology. For example, the formula $\neg c \vee b$ is in the theory of the eks mentioned above. If a formula ϕ is in the theory, we write $\Gamma \models \phi$.

In later sections we shall show how the theory of an eks expresses all the interesting facts about the eks. In particular, we can determine whether an event precedes or follows another, whether two events are in conflict etc.

2 Duality

In the definition, we represented an event structure as a set of events, which were primitive, and a set of states formed from those events. However, there is nothing special about events—we could as well take the states as primitive, leading to a duality between states and events, which we discuss in this section.

Pictorial Representation of an EKS. An eks (E, Q) is any subset Q of 2^E. Thus we can draw it as an E dimensional cube, and then mark the elements of Q on the cube as dots. This is called the partial boolean algebra representation of an eks, as the set 2^E forms a boolean algebra. In the infinite case this is a complete atomic boolean algebra, a CABA[1]. For example, the eks mentioned above, $(\{a, b, c\}, \{\phi, \{a\}, \{ab\}, \{abc\}\})$ can be represented as follows

Partial Boolean Algebra Partial Distributive Lattice

Figure 1

¿From the above description of a partial boolean algebra, the dots may occupy only a few dimensions of the boolean algebra. Since the dots are all the useful information of the eks, there seems no point in carrying around the other dimensions, which we dump by insisting that all eks's we consider must have the T_0 property[2]—the formula $a \leftrightarrow b$ must not be present in the theory of the eks for any pair of events a, b, since $a \leftrightarrow b$ means these are equivalent events which can be identified. As a result of this identification, the boolean algebra cannot be very large, because it must be generated from the dots using the logical operations[3]. *We will assume the T_0 property for all eks's henceforth.*

The information given in the partial boolean algebra can be more succinctly and transparently given in the figure on the right. That figure was obtained by taking the dots and retaining only those holes which can be formed from the dots using unions and intersections but not complements (In this case no holes could be thus generated). The resulting structure is called a partial distributive lattice (pdlat), by analogy with partial boolean algebras. The lattice made up of the dots and holes is a profinite distributive lattice.[4] Note that we do not

[1] In the finite case, "complete atomic" is automatic and hence redundant.

[2] This terminology comes from topology, by considering E as the set of points and Q as the set of open sets. Eks's are a generalization of topological spaces, in that they do not impose any conditions on the open sets.

[3] If the eks is T_0, then for each event in some state, we can take the intersection of all the states containing this event and the complements of all the states not containing it to get a singleton set containing this event. There may be at most one event not in any state, and this is the complement of the union of all the states. Then the rest of the boolean algebra may be generated by the singletons. Conversely, if it is not T_0 as $a \leftrightarrow b$ is in the theory, the singleton set $\{a\}$ cannot be obtained by any logical operations.

[4] In the finite case this is just a distributive lattice. In the infinite case it is most easily understood as a lattice isomorphic to the lattice of order filters or upward-closed subsets of a partial order. This is an even stronger requirement than being a complete and completely distributive lattice, witness the unit interval [0,1] of reals standardly ordered, which meets that condition but is too connected to be profinite. Dedekind cuts as order filters in the unit interval

always achieve such succinctness, for example, when the partial boolean algebra is made up of dots only the pdlat is just the partial boolean algebra.

It is possible for us to use pdlats to represent eks's because of the following proposition :-

Proposition 1 *Partial Boolean algebras are in 1-1 correspondence with partial distributive lattices.*

Proof: The distributive lattice may be obtained from the CABA as the lattice of order filters of the set of dots as partially ordered by the CABA, which is embedded in the CABA; equivalently, the complete sublattice of the CABA generated by the set of dots. Conversely the CABA may be obtained from the profinite distributive lattice as the power set of its dimensions or primes, which embeds that lattice by embedding each element as the set of primes below it; equivalently, generate the Boolean complements of each dot and close up under meets and joins. ∎

Duality between States and Events. Thus far we have considered events as primary and states as sets of events. However we can alternatively consider states as atomic, and events as sets of states, with a state belonging to an event iff the event belongs to that state. If we are willing to be more adventurous with foundations of set theory, we may do both— the resulting circularity in the membership relation can be coped with by dropping the Foundation Axiom of ZF set theory and replacing it by e.g. Aczel's Anti-Foundation Axiom. This results in states and events being treated on the same footing, something which has not been done in previous models.

Schedules and Automata. The operational interpretation of an eks $A = (E, Q)$ is as an automaton, which does events in E and advances from one state in Q to another, according to the transition relation (which is just the subset relation, as stated above). We call the states in Q the *real* states of A and the holes in the underlying lattice the forbidden states, namely the states in which the automaton should never find itself. Thus the automaton can rest and make choices only in the real states, while it must rush through the forbidden states. We take the view the any choices made in the forbidden states are made by the environment. The dimensions of the lattice, defined as equivalence classes by Droste[Dro89], are the events of E.

We can dualise $A = (E, Q)$ by defining $A^{\perp} = (Q, E)$, where Q is the set of states, and E is the set of events each of which is taken to be a set of states. We give a few examples of this in figure 2, drawing the eks's as pdlats. Drawn as

of rationals however is by definition profinite, the difference being that the rationals appear twice. Eliminating the extra copies connects up the real interval and eliminates profiniteness, along with the means for recovering the rationals.

its dual[5], the eks is a schedule. The dots now represent the events which have to be performed, and the lattice structure represents the constraints on these events, some of which we will discuss in the next section.

In fact the schedule and automaton are different ways of looking at the same eks. We can also write the theory of an eks in terms of the state variables, and the two theories have exactly the same information because they are derived from the same boolean formula written in terms of different variables. However, in a schedule, the direction of time is downwards, as indicated by the little arrow on the side, whereas in an automaton, time flows upwards. This is the only way to determine whether an unlabeled pdlat is a schedule or an automaton: any pdlat can be interpreted in either way.

We can label the events of an eks by a labelling function $\lambda : E \to Act$ going from events to actions. This allows us to consider the case when multiple instances of the same action occur. We will not go into this any further except to remark that we can have complementary action labels and silent actions, allowing synchronization, hiding and relabelling as in CCS[Mil90].

Time and Information Duality. The duality between states and events enables us to see a similar duality between time and information. Events are regarded as instantaneous, occurring at a particular time, and they add to the total accumulated information. Dually, no information is accumulated in a state, which however adds to the total amount of time that has passed since the beginning of the process. Thus we can plot the progress of a behavior on an graph with time on the x-axis and information on the y-axis. The states are then horizontal line segments, and the events are vertical. The graph is monotonically increasing, and we shall have more to say about this in the conclusion. The graph is also piecewise horizontal and vertical, but as the number of events grows to infinity it starts looking like a monotonically increasing curve.

3 Modelling Concurrent Behavior

We now show how various concurrent phenomena can be represented in an eks. The different forms of formulae in the theory of the eks A represent different forms of constraints on the events. We can also use the theory of the dual eks to represent constraints on states, and some constraints are easier to comprehend this way. We use $\Gamma^\perp$ for the formula of the dual eks $A^\perp$.

Temporal precedence. For any two events a, b, if $\Gamma \models b \to a$, then no state contains b and not a. This means that b can be executed only after a has been executed. This means that a precedes b in time, and the collection of all such

[5]In terms of the matrix representation, forming the dual corresponds to transposing the matrix. This is true because in taking the dual we interchange the states with events, and take the converse of the membership relation.

constraints defines a temporal order on the events of G. We write $a \leq b$ for any such pair of events. Note that the temporal order is the converse of the logical order, and this is the reason for the downwards flow of time for the schedule. (We always have the logical order going up.)

Conflict. If $\Gamma \models \neg(a \wedge b)$, then no state contains both a and b. This means that we can never execute both a and b, which is interpreted as a *conflict* between a and b. This can be generalised to conflict between arbitrary numbers of events. While this notion has been explicitly introduced in event structures [NPW81, Win86], it follows quite naturally from our definition. It is illustrated in Figure 2(c-d), where b and c are in conflict.

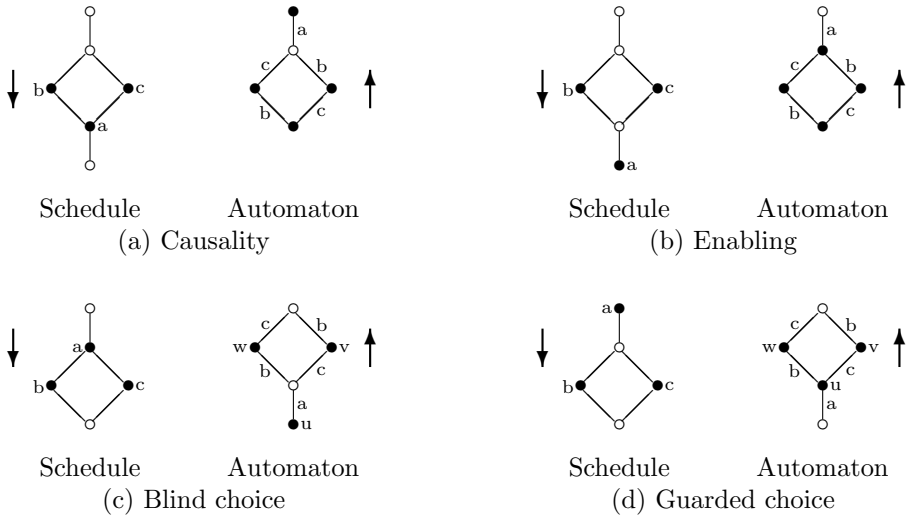

Figure 2

Causality and enabling. If $\Gamma \models a \leftrightarrow b \wedge c$ then any state containing b and c must also have a. We interpret this as b and c together *cause* a. This can be generalized to an arbitrary formula on the right side of the implication. Note that the T_0 condition does not allow an event to be caused by a single event, as the two events are identified.

The above eks is distinct from the eks in which $\Gamma \models a \rightarrow b \wedge c$. Then b and c just *enable* a, which means that while it is necessary for them to be done to do a, it is not sufficient, as a need not be done immediately. This can also be generalized to an arbitrary condition, including the singleton event, when it becomes temporal precedence. The difference between these is like the difference between the necessary and sufficient conditions for a theorem, while the enabling formula represents the necessary conditions, the causing formula is both necessary and sufficient.

Nondeterminism. Dual to the above notion is the notion of guarded vs. blind choice. If $\Gamma \models a \leftrightarrow b \vee c$, then as soon as a is done, one of b or c must be done immediately. This means that any information obtained by doing a could not be used in selecting between b and c, making this a *blind* choice. This is more clearly visible from the automaton side in Figure 2(c). The choice between states v and w has to be made at the forbidden state above u. We interpret this as being made by the environment, making this a nondeterministic choice for the automaton.

Just having $\Gamma \models b \vee c \rightarrow a$ would make it a *guarded* choice, whence any information obtained from a can be used to make a deliberate choice. On the automaton side, the choice between v and w is made in the state u, as $\Gamma^{\perp} \models u \leftrightarrow v \wedge w$, and this state contains the information gathered by doing a to decide which way to go. This is like a conditional branch in a program, where the event a checks some condition, and then either one of two alternatives is chosen.

If b and c were not in conflict here, then this would not be a real choice. The hole at the top of the automaton would be a permitted state, and then b and c could be executed concurrently, showing that this model has true concurrency. In fact interleaving semantics would not work here at all, as is clear from Figure 2(a), when a has to be done concurrently with either b or c.

Remark. The reader may have noticed that when a meet is present in the schedule, a corresponding join is absent in the automaton, and vice versa, and similarly for joins. For example, a is the join of b and c in Figure 2(c), and the meet of v and w is not present (it is a hole). This is a general principle, so the dual of a set (which has no structure) is a highly structured object, a CABA, and the dual of a meet semilattice is another meet semilattice, as it can have no joins.

Disjunctive Enabling. If $\Gamma \models c \rightarrow a \vee b$, then in order for c to happen, at least one of a or b must have happened. For example, in a candy machine, inserting either a dollar bill or 4 quarters will enable the machine to supply a candy bar. This is called disjunctive enabling [Win86, Gun91, Gun92]. The dual behavior is *postponed concurrency*, given by $\Gamma \models a \wedge b \rightarrow c$, whereby a and b can be done concurrently only after c is completed.

If we draw an automaton for disjunctive enabling with three events, i.e. one which has $\Gamma \models c \rightarrow a \vee b$, then it looks like a 3-dimensional cube, with all vetices marked by dots except the vertex c, which is the only vertex which does not satisfy the formula $c \rightarrow a \vee b$. This illustrates a general method for programming an automaton on n events from a logical specification—from the n-cube, mark with dots only those vertices which satisfy all the equations. The resulting structure gives a partial boolean algebra for the specification, and this can be converted into a pdlat or eks.

Action Refinement. Given an eks (E, Q), and another eks (E_1, Q_1) where

$\phi \in Q_1$ and $E_1 \in Q_1$, (i.e. it is conflict free), we can refine any event e in (E, Q) by (E_1, Q_1) to get a new eks (E', Q') as follows: $E' = E - \{e\} \cup E_1$, and the set Q' is obtained from Q. If $q \in Q$, and $e \notin q$, then $q \in Q'$. If $e \in q$, then $q - \{e\} \cup E_1 \in Q'$. Also, for these q, if $\forall a \in q, a \neq e \Rightarrow e \not\leq a$, then for each $q' \in Q_1$, $q - \{e\} \cup q' \in Q'$.

What this means is that instead of doing e, we do the eks (E_1, Q_1). The third clause assures that the intermediate states of (E_1, Q_1) are added iff no event in the state is preceded by e, since otherwise the event e must have been completed in some previous state. Thus if we looked upon events as intervals, a dot means that the event before it and the event after it are disjoint in time, whereas a hole means that they overlap.

4 Process Algebra

We now define a process algebra for combining various eks's. The operations considered are parallel composition $A|B$, sequential composition $A;B$, choice or summation $A \sqcup B$, iteration A^* and interaction $A \otimes B$. In order to define sequential composition and iteration, we add the concept of final states to the definition of an eks. An eks now is (E, Q, F), where $F \subseteq Q$. If $A = (E, Q, F)$ is an eks, we define functions $E(A) = E, Q(A) = Q$ and $F(A) = F$, giving respectively the events, states and final states of A. Note: in the following expressions, the set union operator has the lowest precedence.

Parallel composition $A|B$ of A and B is the independent execution of A and B. We assume $E(A) \cap E(B) = \phi$. Define $E(A|B) = E(A) \cup E(B)$, $Q(A|B) = \{q \mid q = q_1 \cup q_2, q_1 \in Q(A), q_2 \in Q(B)\}$ and $F(A|B) = \{q \mid q = q_1 \cup q_2, q_1 \in F(A), q_2 \in F(B)\}$. This is an eks as $F(A|B) \subseteq Q(A|B) \subseteq 2^{E(A|B)}$. Note that any events which can be executed from a state in A can be executed from any corresponding state in $A|B$, and similarly for B, showing that this represents independent execution of A and B. A state in the composition is final iff both A and B are in their final states.

The symbol for parallel composition used here is to conform to the usage in the process algebra community. However our preferred symbol for it is $+$ [GP93], since this is a coproduct in an appropriate category. Consequently we prefer $\mathbf{0}$ for the eks 1 defined below, since it is an initial object in that category.

Sequential composition $A;B$ represents the execution of B after completing the execution of A, i.e. after A has reached a final state. Once B starts executing, A may not execute any further. Define $E(A;B) = E(A) \cup E(B) \times F(A)$, i.e. have one copy of B for each final state of A. Then $Q(A;B) = Q(A) \cup \{q \mid q = q_1 \cup q_2 \times \{q_1\}, q_1 \in F(A), q_2 \in Q(B)\}$, saying that the automaton can be in a state of A, or having reached a final state of A, execute events in the copy of B associated with that final state of A. $F(A;B) = \{q \mid q = q_1 \cup q_2 \times \{q_1\}, q_1 \in$

$F(A), q_2 \in F(B)\}$, saying that a final state is one in which A in a final state and B has also reached a final state afterwards.

Choice or summation $A \sqcup B$ is the execution of A or B but not both. We define $E(A \sqcup B) = E(A) \cup E(B), Q(A \sqcup B) = Q(A) \cup Q(B), F(A \sqcup B) = F(A) \cup F(B)$. If $E(A)$ and $E(B)$ are disjoint, this executes either one of A or B but not both, and once having started A, no event in B can be executed, since there is no state containing both events of A and events of B. However we do not insist on the disjointness, since we wish to include the possibility of choosing between two copies of A etc.

Define the null eks Φ to be (ϕ, ϕ, ϕ), the eks which does nothing. Also define the eks 1 to be $(\phi, \{\phi\}, \{\phi\})$, which will be the identity for $A; B$ and $A|B$.

Proposition 2 *The following properties hold*

(1)	$A \sqcup A$	$=$	A
(2)	$A \sqcup B$	$=$	$B \sqcup A$
(3)	$A \sqcup (B \sqcup C)$	$=$	$(A \sqcup B) \sqcup C$
(4)	$A \sqcup \Phi$	$=$	A
(5)	$A \mid 1$	$\cong$	A
(6)	$A \mid B$	$\cong$	$B \mid A$
(7)	$A \mid (B \mid C)$	$\cong$	$(A \mid B) \mid C$
(8)	$A; 1$	$\cong$	$1; A \cong A$
(9)	$\Phi; A$	$=$	Φ
(10)	$A; (B; C)$	$\cong$	$(A; B); C$
(11)	$(A \sqcup B) \mid C$	$\cong$	$A \mid C \sqcup B \mid C$
(12)	$(A \sqcup B); C$	$\cong$	$A; C \sqcup B; C$
(13)	$C; (A \sqcup B)$	$\cong$	$C; A \sqcup C; B$

Proof: The first seven properties follow from the properties of set union. The eighth follows as the cartesian product of any set with the singleton is isomorphic to itself. The ninth holds as the product of an empty set with anything is empty. In the next 4 properties, the arguments for the equality or isomorphism of F are similar to the respective arguments for Q, and so are omitted.

(10) $E(A; (B; C)) = E(A) \cup E(B; C) \times F(A) = E(A) \cup E(B) \times F(A) \cup E(C) \times F(B) \times F(A)$. But $F(B) \times F(A) \cong F(A; B)$, so we get $E(A; (B; C)) \cong E((A; B); C)$.

$$\begin{aligned} Q(A; (B; C)) &= Q(A) \cup \{q \mid q = q_1 \cup q_2 \times \{q_1\}, q_1 \in F(A), q_2 \in Q(B; C)\} \\ &= Q(A) \cup \{q \mid q = q_1 \cup q_2 \times \{q_1\}, q_1 \in F(A), q_2 \in Q(B)\} \\ &\quad \cup \{q \mid q = q_1 \cup q_2 \times \{q_1\} \cup q_3 \times \{q_1\} \times \{q_2\}, q_1 \in F(A), q_2 \in F(B), q_3 \in Q(C)\} \\ &\cong Q(A; B) \cup \{q \mid q = q_1 \cup q_2 \times \{q_1\}, q_1 \in F(A; B), q_2 \in Q(C)\} \\ &\cong Q((A; B); C) \end{aligned}$$

(11) $E((A \sqcup B) \mid C) = E(A) \cup E(B) \cup E(C) = E((A \mid C) \sqcup (B \mid C))$.

$$\begin{aligned} Q((A \sqcup B)|C) &= \{q \mid q = q_1 \cup q_2, q_1 \in Q(A \sqcup B), q_2 \in Q(C)\} \\ &= \{q \mid q = q_1 \cup q_2, q_1 \in Q(A), q_2 \in Q(C)\} \cup \\ &\quad \{q \mid q = q_1' \cup q_2, q_1' \in Q(A \sqcup B), q_2 \in Q(C)\} \\ &= Q((A|C) \sqcup (B|C)) \end{aligned}$$

(12) $E((A \sqcup B); C) = E(A) \cup E(B) \cup E(C) \times F(A \sqcup B) = E((A;C) \sqcup (B;C))$, as $F(A \sqcup B) = F(A) \cup F(B)$.

$$\begin{aligned} Q((A \sqcup B); C) &= Q(A \sqcup B) \cup \{q \mid q = q_1 \cup q_2 \times \{q_1\}, q_1 \in F(A \sqcup B), q_2 \in Q(C)\} \\ &= Q(A) \cup \{q \mid q = q_1 \cup q_2 \times \{q_1\}, q_1 \in F(A), q_2 \in Q(C)\} \cup \\ &\quad Q(B) \cup \{q \mid q = q_1 \cup q_2 \times \{q_1\}, q_1 \in F(B), q_2 \in Q(C)\} \\ &= Q(A;C) \cup Q(A;C) \\ &= Q((A;C) \sqcup (B;C)) \end{aligned}$$

(13) Similar to (12). These distributive laws can be extended to an infinite summation also. ∎

The iteration of A, denoted A^* represents the repeated execution of A finitely many times, as proved below. Define $E(A^*) = \{(a,i) \mid a \in E(A), i \in \mathbf{N}\}$, where $\mathbf{N}$ is the set of natural numbers. Then $Q(A^*) = \{\phi\} \cup \{q \mid q = \bigcup_{0<i<n}(q_i, i) \cup (q_n, n), q_i \in F(A)$ for $i < n, q_n \in Q(A), n \in \mathbf{N}\}$, where $(q,i) = \{(a,i) \mid a \in q\}$. The set of final states is defined similarly, $F(A^*) = \{\phi\} \cup \{q \mid q = \bigcup_{0<i\leq n}(q_i, i), q_i \in F(A)$ for $i \leq n, n \in \mathbf{N}\}$.

Proposition 3 $A^* \cong 1 \sqcup A \sqcup A;A \sqcup A;A;A \sqcup \ldots$

Proof: Since the event sets $E(A)$ and $E(B)$ must be disjoint for $A;B$, we index the events of $A;A;A\ldots$ with the number of the occurrence of A. Then the event sets on both sides of the equation are identical.

Consider the states in $A^n = A;A;\ldots;A$, i.e. A repeated n times. For $n = 1$ these are just $Q(A) = \{q \mid q = \bigcup_{0<i<n}(q_i, i) \cup (q_n, n), q_i \in F(A)$ for $i < n, q_n \in Q(A), n = 1\}$ and $F(A) = \{q \mid q = \bigcup_{0<i\leq n}(q_i, i), q_i \in F(A)$ for $i \leq n, q_n \in Q(A), n = 1\}$. Assume this is true for $n = m$. Then $Q(A^{m+1}) = Q(A^m) \cup \{q \mid q = q_1 \cup (q_{m+1}, m+1), q_1 \in F(A^m), q_{m+1} \in A\}$ by definition. But $F(A^m) = \{q \mid q = \bigcup_{i\leq m}(q_i, i), q_i \in F(A)\}$. So we get the required form for $Q(A^{m+1})$, i.e. $\{q \mid q = \bigcup_{0<i<n}(q_i, i) \cup (q_n, n), q_i \in F(A)$ for $i < n, q_n \in Q(A), n \leq m\}$, and similarly for $F(A^{m+1})$. Since $Q(A^*)$ is just the union of all these state sets, and $F(A^*)$ is the union of all the final state sets, we get an isomorphism between the state sets and the final state sets on both sides. ∎

This gives us all the necessary properties of A^*. In particular, we can prove that $(A^*)^* \cong A^*, A^*;A^* \cong A^*$ and $(A \sqcup B)^* \cong (A^*;B^*)^*$. A^* is the least fixed point of the function $f(A) = 1 \sqcup A;f(A)$. We can similarly define $A^\dagger$ as the fixed point of $f(A) = 1 \sqcup A|f(A)$, mentioned earlier in [Pra86].

Interaction, $A \otimes B$, earlier known as orthocurrence or flow[Pra86, CCMP91], represents one process flowing through another. For example, if process A is

three trains running sequentially, and process B represents four stations on the track, then there are $3 \times 4 = 12$ events, corresponding to each train arriving at each station. For each train, the stations must arrive in the same order, and for each station the trains must arrive in the same order too. This leads us to define $E(A \otimes B) = E(A) \times E(B)$, and $Q(A \otimes B) = \{q \subseteq E(A \otimes B) \mid \forall b \in E(B),$ if $q_b = \{a \mid (a,b) \in q\}$, then $q_b \in Q(A)$, and similarly for events in $E(A)\}$. $F(A \otimes B)$ is defined similarly, by replacing Q by F in this definition.

Proposition 4 *The following properties hold*

$$\begin{array}{lrcl} (1) & A \otimes B & \cong & B \otimes A \\ (2) & A \otimes (B \otimes C) & \cong & (A \otimes B) \otimes C \\ (3) & (A|B) \otimes C & = & (A \otimes C)|(B \otimes C) \end{array}$$

Proof:

We omit the proofs of the first two identities, as they follow from the definition.

(3) $E((A|B) \otimes C) = E((A \otimes C)|(B \otimes C))$, as $\times$ distributes over set union.

$$\begin{array}{rcl} Q((A|B) \otimes C) & = & \{q \mid \forall c \in E(C), q_c \in Q(A|B), \forall d \in E(A|B), q_d \in Q(C)\} \\ & = & \{q \mid \forall c \in E(C)[q_c = (q_c)_1 \cup (q_c)_2, (q_c)_1 \in Q(A), (q_c)_2 \in Q(B)], \\ & & \quad \forall a \in E(A)[q_a \in Q(C)], \forall b \in E(B)[q_b \in Q(C)]\} \\ & = & \{q \mid q = q_1 \cup q_2, \forall c \in E(C)[(q_1)_c \in Q(A), (q_2)_c \in Q(B)], \\ & & \quad \forall a \in E(A)[(q_1)_a \in Q(C)], \forall b \in E(B)[(q_2)_b \in Q(C)]\} \\ & = & Q((A \otimes C)|(B \otimes C)) \end{array}$$

In the third step, we separated out the q into two states, one whose first components came from A, and the other, whose first components came from B. Then we used the fact that q_a is the same as $(q_1)_a$, and same for q_b. The proof for the final states is similar. Interaction does not distribute over sequential composition or choice. ∎

5 Relationship with Event Structures

We show that eks's subsume the most general case of event structures as defined by Winskel [Win88b]. As we have shown above, conflict and enabling of events can be expressed in an eks. Since these are the only two concepts required in the definition of an event structure, we can show that event structures can be modeled by eks's, for what amount to trivial reasons. However event structures cannot distinguish causality and enabling, showing that eks's are strictly more general than event structures.

An event structure is a set of events E, with a conflict relation $\#$ and an enabling relation $\vdash$, defined in [Win88a]. A configuration is any subset of events

which could have occurred in the event structure, so it must be conflict free, and every event must be caused by some previous events in the set according to the enabling relation. Given an event structure $E = (E, \#, \vdash)$, let $\mathcal{F}(E)$ be its set of configurations. We form the eks $G = (E, \mathcal{F}(E))$, which is clearly equivalent to the event structure, and this leads us to the following theorem.

Proposition 5 *For every event structure that is formed from its family of configurations, there is an eks which has the same properties:*

1. *If $a\#b$ in the event structure, then $\Gamma(G) \models \neg(a \wedge b)$.*
2. *If $X_1 \vdash e, \ldots, X_n \vdash e$ in the event structure, then $\Gamma(G) \models e \rightarrow \bigvee X_i$, where X_i is taken to be the conjunction of the events in X_i.*

The labelling of eks's enables this embedding to respect the labelling function of labelled event structures too. The definition of sum in Winskel corresponds to the definition of (disjoint) sum above:

Proposition 6 *Let E_0 and E_1 be two event structures, and G_0 and G_1 be their corresponding eks's. Then $G = G_0 \sqcup G_1$ is the eks corresponding to $E = E_0 + E_1$ as defined by Winskel.*

Proof: The sum of E_0 and E_1 is the disjoint union of their events along with their individual conflict and enabling relations, such that every event of E_0 is in conflict with E_1. Then $E(G)$ corresponds to the events of E. Also, any configuration of E is a configuration of E_0 or of E_1, as no mixed configuration is possible. This is exactly the state set of G. ∎

There is no universally accepted definition of sequential composition for event structures. Baeten and Vaandrager[BV92] define sequential composition, and our definition does agree with theirs operationally. They define a special event, a $\surd$ which is interpreted as the last event of the process executing the event structure. Sequential composition $E_0; E_1$ is then defined by refining each $\surd$ of E_0 with E_1. We can mimic this by letting each configuration in which a $\surd$ can be performed next as a final state of G_0, the eks corresponding to E_0, and omitting $\surd$ from the set of events. The correspondence between the states of $G_0; G_1$ and the configurations of $E_0; E_1$ is then clear.

In [Win88b], Winskel defined the partial synchronous composition of two event structures. It is possible to define this for eks's also, in a way that respects the above embedding.

We let $A||B$ stand for the partial synchronous product of A and B. Then $E(A||B) = E(A) \cup E(B) \cup E(A) \times E(B)$. $Q(A||B)$ is a set of subsets of $E(A||B)$, each element q of which satisfies the following properties:

1. The sets $q \cap E(A)$, and for each $b \in E(B)\{a \mid (a,b) \in q\}$ are pairwise disjoint. Similarly, the sets $q \cap E(B)$, and for each $a \in E(A)\{b \mid (a,b) \in q\}$ are pairwise disjoint.

2. Define $q_A = (q \cap E(A)) \cup Pr_1{}^6(q \cap E(A) \times E(B))$ and $q_B = (q \cap E(B)) \cup Pr_2(q \cap E(A) \times E(B))$. Then $q_A \in Q(A)$ and $q_B \in Q(B)$.

3. If $q' \subseteq q_A$ and $q' \in Q(A)$ then $\exists q'' \in Q(A||B)[q'' \subseteq q$ and $q' = q''_A]$ and the same for B.

The first property says that an event of A may either occur alone or may synchronize with exactly one event of B. The second condition says that if we extract all the events of A which have happened in a state q, then these must form a state of A, called q_A. The third condition says that if can reach the state q_A in A by doing some events, then we should be able to do the same events (some synchronized with events in B) to reach the state q. This prevents some undesirable states, eg the state $\{(a,d),(b,c)\}$ is not allowed when forming $(a;b)||(c;d)$. It is eliminated as there is no way to reach it by first doing a and then b, as that would lead to doing d before c. The only states allowed in the above example are[7] $\{\phi, a, a+b, c, c+d, a+c, a+c+d, a+c+b, a+b+c+d, ac, ac+b, ac+d, ac+b+d, ad+c, ad+c+b, cb+a, cb+a+d, a+c+bd, ac+bd\}$. All the other states are ruled out as they do not satisfy one or more of the conditions.

This definition agrees with that of Winskel[Win88b].

Proposition 7 *Let E_1 and E_2 be two event structures, and A_1 and A_2 the eks's corresponding to them. Then the eks $A_1||A_2$ corresponds to the event structure E formed by the partial synchronous composition of E_1 and E_2.*

Proof: Outline: The events of E match the events of $A_1||A_2$, by definition. Each state in $A_1||A_2$ is conflict free, by the first condition. Also, if any two events were in conflict in E_1, then they would not occur in any state of A_1, so they would remain in conflict in states of $A_1||A_2$, by condition 2. Finally we can use the third condition to prove by induction that all states in $A_1||A_2$ are secured. Conversely, since the only states that were dropped were the ones that did not satisfy these conditions, the remaining states are exactly the configurations of E. ∎

Another model which is superficially similar to eks's is Gunawardena's causal automata[Gun91, Gun92]. However the semantics for causal automata are quite different from that of eks's, in that if $a \to b$ is true in a causal automaton, then b must have occurred before a, whereas in eks's, this condition means only that a cannot occur before b. As a result, causal automata can express deadlock, which we have not been able to express using eks's. The logical constraints that

[6] $Pr_1(X)$ is the set of first components of elements of X.
[7] We write $ac + b$ for $\{(a,c),b\}$.

causal automata can express on events are a strict subset of the constraints expressible in eks's, for example causal automata cannot distinguish between choice and non-determinism, or between causality and enabling.

6 Summary

We have defined a process algebra of eks's, and given various properties for the connectives. We have also shown a duality for eks's which enables us to look at each eks as a declarative program, a schedule or as an imperative program, an automaton. This would enable a programmer to choose the point of view in which it is easier to write any program, and also allows for an easy conversion between the two views. This duality is a form of Stone duality[Joh82], and we will discuss this connection elsewhere.

The natural semantics for eks's is a true concurrency semantics, as in event structures and Petri nets. However eks's also enable us to specify nondeterminism and choice, and causality, which the other two models do not.

The major limitation of our approach is that choosing conflicting alternatives ensures that we are in different branches of the automaton, and can never come together again. This seems to be a feature of all existing schedule automaton dualizations[NPW81, Win86, Pra92]. It makes it necessary to unfold any loops in an automaton, and duplicating the actions, as in our definition of A^*. This may have undesirable consequences for algorithms using eks's, and complicate decision problems. It also does not take into account the fact that some conflicts, like in mutual exclusion, are temporary. This permanence of conflict is the reason why the information-time graphs for eks's are monotonically increasing. In other words, eks's never forget any information about any choices made in the past, even if some of this information is useless.

We hope to overcome this problem by extending schedule automaton duality to structures in which conflict is not so permanent. One approach for doing this seems to be higher dimensional automata[Shi85, Pra91, GJ92], which distinguish a **mutex** b from $ab + ba$ by representing the former as a square with no interior, and the latter as in eks's with labelling. Eks's cannot make this distinction now, but higher dimensional automata do not yet have the schedule automata duality, and we hope to reconcile the two.

We have not shown completeness of the properties of the eks connectives, and plan to address this in the future. The set of connectives can be embellished with the other operators from linear logic, and eks's then form a model for it, but this model is also not complete. However it does give us a logic to reason with eks's, and is useful in constructing a verification language for concurrency. This connection is explored in[Pra93].

References

[Bar79] M. Barr. *$*$-Autonomous categories, LNM 752.* Springer-Verlag, 1979.

[BG90] C. Brown and D. Gurr. A categorical linear framework for Petri nets. In J. Mitchell, editor, *Logic in Computer Science*, pages 208–218. IEEE Computer Society, June 1990.

[BV92] J.C.M. Baeten and F.W. Vaandrager. An algebra for process creation. *Acta Informatica*, 29(4):303–334, 1992.

[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.

[Dro89] M. Droste. Event structures and domains. *Theoretical Computer Science*, 68:37–47, 1989.

[GJ92] E. Goubault and T.P. Jensen. Homology of higher dimensional automata. In *Proc. of CONCUR'92, LNCS 630*, pages 254–268, Stonybrook, New York, August 1992. Springer-Verlag.

[GP93] V. Gupta and V.R. Pratt. Gates accept concurrent behavior. In *Proc. 34th Ann. IEEE Symp. on Foundations of Comp. Sci.*, November 1993.

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

[Gun91] J. Gunawardena. Geometric logic, causality and event structures. In J. C. M. Baeten and J. F. Groote, editors, *CONCUR'91 - 2nd International Conference on Concurrency Theory*, pages 266–280. Springer LNCS 527, 1991.

[Gun92] J. Gunawardena. Causal automata. *Theoretical Computer Science*, 101:265–288, 1992.

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

[LS91] Y. Lafont and T. Streicher. Games semantics for linear logic. In *Proc. 6th Annual IEEE Symp. on Logic in Computer Science*, pages 43–49, Amsterdam, July 1991.

[Maz77] A. Mazurkiewicz. Concurrent program schemas and their interpretation. In *Proc. Aarhus Workshop on Verification of Parallel Programs*, 1977.

[Mil80] R. Milner. *A Calculus of Communicating Systems, LNCS 92.* Springer-Verlag, 1980.

[Mil90] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 19, pages 1201–1242. Elsevier Science Publishers B.V. (North-Holland), 1990.

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

[Pet62] C.A. Petri. Fundamentals of a theory of asynchronous information flow. In *Proc. IFIP Congress 62*, pages 386–390, Munich, 1962. North-Holland, Amsterdam.

[Pra82] V.R. Pratt. On the composition of processes. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Programming Languages*, January 1982.

[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.

[Pra92] V.R. Pratt. Arithmetic + logic + geometry = concurrency. In *Proc. First Latin American Symposium on Theoretical Informatics, LNCS 583*, pages 430–447, São Paulo, Brazil, April 1992. Springer-Verlag.

[Pra93] V.R. Pratt. The second calculus of binary relations. In *Proceedings of MFCS'93*, Gdańsk, Poland, 1993. Springer-Verlag.

[Shi85] M. Shields. Deterministic asynchronous automata. In E.J. Neuhold and G. Chroust, editors, *Formal Models in Programming*. Elsevier Science Publishers, B.V. (North Holland), 1985.

[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.

[Win88a] G. Winskel. A category of labelled Petri nets and compositional proof system. In *Proc. 3rd Annual Symposium on Logic in Computer Science*, Edinburgh, 1988. Computer Society Press.

[Win88b] G. Winskel. An introduction to event structures. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, REX'88, LNCS 354*, Noordwijkerhout, June 1988. Springer-Verlag.