A DiGraph Model for Extended Event Structures

Padmanabhan Krishnan
Department of Computer Science
Email:paddy@cosc.canterbury.ac.nz

David J. Bryant
Department of Mathematics
Email:djb@math.canterbury.ac.nz

University of Canterbury,Private Bag 4800
Christchurch, New Zealand

1 Introduction

A geometric model for concurrency has been described by Pratt in [3]. While the intuition is easily understood, the formal description using n-complexes requires some mathematical maturity. The purpose of this paper is two fold. The first is to construct a model directly based on 3^d (let there be d events and each event can be be in three states, viz., not started, running and terminated) which is described informally in [3]. This model is based on directed graphs and should be easier to understand than a model based on algebraic topology though we still incorporate the notions of tears, holes, persistent choice etc. The second is to study a few extensions. We make explicit the notion of transience, i.e., an event has start, transient and end phases. We also add a notion of resources. Before we develop our model, we present a quick overview of the issue of parallelism and choice in concurrency and its handling in a geometric framework.

Consider the process (a | b). In an interleaving model such as CCS [2], it would be identified with (a·b + b·a). In a geometric model the interleaving of a and b is represented by a square defined by the points (0,0), (1,0), (0,1) and (1,1) where the edges from (0,y) to (1,y) defines the transition a and the edges from (x,0) to (x,1) defines the transition b. In a true concurrency model a notion of a and b executing jointly is essential. This is captured in the geometric model by filling the hole present in the square. This idea can be generalised to higher dimensions giving rise to n-dimensional cubes.

Event structures [5] are models of concurrency with a notion of causality and consistency. A prime event structure is a triple (E, $<$, #) where E is a set of events, $<$ is a causal order and # a conflict relation. For two events e_1 and e_2, if $e_1 < e_2$, the event e_1 must occur before e_2 can occur while if $e_1 \# e_2$, then both e_1 and e_2 cannot occur. The effect of the causality and conflict relations on the geometric model is to remove certain faces. Pratt in [3] shows that the two models are duals, i.e., the event structure defines a schedule on the event occurrences while the geometry is an automaton and performs the computation. Executing an event

will require some resources. A calculus for real-time systems with resource requirements is described in [1]. In it they assume that each resource can execute at most a single event with multiple events being exhibited by synchronising multiple resources.

In this paper we consider a digraph model for concurrency which can handle explicit transience and a simple notion of resource constraints. We first describe a general digraph automaton and then show how certain types of event structures can be modelled using digraph automata.

2 DiGraph Automata

Towards defining the automata, we define a set {0,T,1} and the linear order $0 < T < 1$ on it. The value 0 indicates that a given action has not yet started, the value T indicates that the given is being executed while the value 1 indicates completion. This is identical to the intuition described in [3]. To simplify our exposition we say T is the successor of 0 and 1 is the successor of T.

Definition: 1 *Given a set of events E, a state is a function from E to {0,T,1}. We say a state g is a successor of a state f iff*

- *For every e in E, $g(e) \geq f(e)$ holds*
- *There is an e in E such that g(e) is the successor of f(e)*

A digraph automaton $\mathcal{G}$ is a directed graph where the nodes are elements are states such that

- *There is an initial state which maps every element of E to 0.*
- *If there is an edge from f to g, g is the successor of f.*
- *If there is a state f and events e_1 and e_2 such that $f(e_1) = f(e_2) = T$,*

  there is a state g reachable from f such that $g(e_1) = g(e_2) = 1$.

  there is a state h such that $h(e_1) > 0$ and $h(e_2) = 0$
- *For every e there is a state (say f) reachable from the initial state such that $f(e) > 0$.*

A node in the automaton indicates the state of the computation. An edge in the automaton indicates a move from one state to the next state. The conditions on the egdeg states that if two actions can be executed concurrently then in all branches where one of the events is started converge, i.e., concurrency differs from

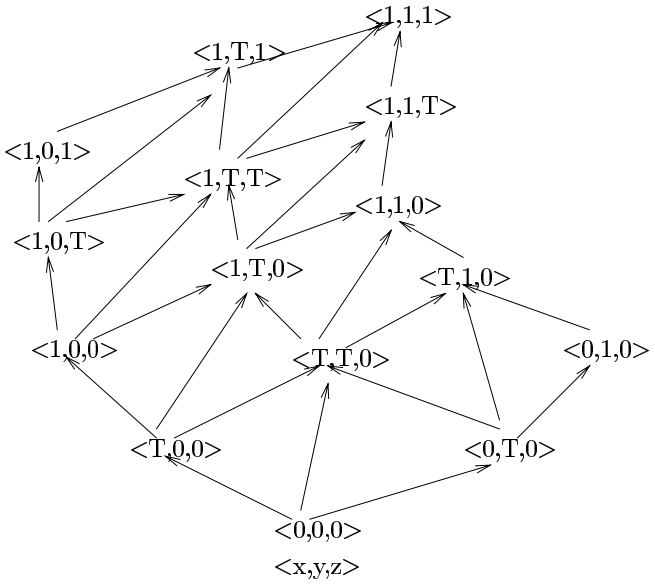

Figure 1: Example

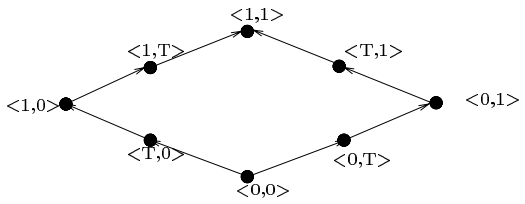

Figure 2: Automaton with a Hole

choice. Furthermore, true concurrency is not demanded. We also require that every event can be activated (not necessarily in every execution).

An example of a digraph automaton over three events (say x,y and z) is given in figure 1. It represents the process ((x·z) | y).

Definition: 2 *A digraph automaton $\mathcal{G}$ has a hole due to events events e_1 and e_2 iff*

- *There is no reachable state f such that $f(e_1 = f(e_2) = T$*

- *There is a reachable state g such that $g(e_1) = g(e_2) = 1$*

Holes are essentially created by permitting interleaving of two events (g and h in the definition) but not their true concurrency (f in the definition). The example in figure 1 does not have a hole. The automaton corresponding to the interleaved execution of the process (a | b), shown in figure 2, will have a hole. There is a function where both events are 1 but there is no function which has both active.

Definition: 3 *A digraph automaton $\mathcal{G}$ has a tear if there exist events e_1 and e_2 such that $\forall$ f, $f(e_1) > 0$ implies there is no path $f_0, f_1, \ldots f_n$ with $f_0 = f$ and $f_n(e_2) = T$.*

We say that the automaton has a tear due to e_1 and e_2.

Non-deterministic choice gives rise to tears. For example the automaton representing (a + b) will have a tear.

3 Extensions to Event Structures

Pratt in [3, 4] describes the duality between geometric automata and event structures or schedules. In this section we consider an extended form of schedules and show the duality between certain geometric automata and schedules. The first extension we consider is to decompose an event into a start, a transition and a finish component. While this has been used implicitly in [3], there is no equivalent use in the event structure model. We first describe the extension on an elementary event structure, i.e., one without conflict and then extend it to general event structures.

As we have considered the set {0,T,1}, we have implicitly assumed that events have a 'duration' i.e., can be in state T. Hence it is possible to interpret the precedence relation $<$ as precedence for starting the events, but once events have been started in the right order they can be executed concurrently. Resources will dictate the restriction on executing concurrently.

Definition: 4 *A resource constrained elementary event structure is (E, $<$, $\mathcal{C}$) where*

E is a set of events

$<$ is a precedence relation on E such that (E,$<$) is a irreflexive partial order.

$\mathcal{C}$ is a collection of subsets of E indicating true concurrency where $\mathcal{C}$ satisfies

$\emptyset \in \mathcal{C}$

$\forall\ e \in E, \{e\} \in \mathcal{C}$

$X \in \mathcal{C}$ *and* $Y \subseteq X$, *then* $Y \in \mathcal{C}$

$x,y \in X$, $X \in \mathcal{C}$, $x<z<y$ *implies* $z \in X$.

While the structure of $\mathcal{C}$ is similar to the structure of the enabling relation in event structures, the interpretation is different. An intuitive semantics of $\mathcal{C}$ is as follows. There are two main cases.

The first is if there is a set X $\in \mathcal{C}$ such that x and $y \in$ X. There are two sub cases to consider.

If $x < y$, it is required that y starts only after x starts and finishes only after x finishes.

If x is not related to y then x and y can be executed in a truly concurrent fashion as the resources permit concurrent execution.

The second main case is if there is no set X $\in \mathcal{C}$ such that x and $y \in$ X.

If $x < y$ then x has to finish before y can start.

If x and y are unrelated, x and y can start in any arbitrary order. However, only one of the events can be active at any given time. This yields the interleaving model with atomic actions.

In general, the effect of $\mathcal{C}$ is to remove certain nodes and edges from a digraph automaton. If $\mathcal{C}$ consists of only the singleton sets the structure is identical to the poset schedule in [3, 4].

The last property satisfied by elements of $\mathcal{C}$ deserves some comment. It states that if two events x and y can be executed concurrently and it is required that there is an event z which has to start before y can start which itself requires x to start, then it is possible to execute x,y and z concurrently. In our current model we cannot specify that z must terminate before y and only x and y can be executed concurrently.

The sets in $\mathcal{C}$ model resource constraints and indicate the set of events which do not compete for a common resource. Singleton events are always in the set as we assume that resources are available to generate all events. If this is not the case we can always consider the relevant subset of E.

We now consider the digraph dual of a given extended event structure. Given an extended event structure (E, $<$, $\mathcal{C}$), define the set $\mathcal{G}$ of functions from E to {0,T,1} such that

- (x $<$ y) implies for all f $\in \mathcal{G}$, f(x) $<$ f(y)

- $\forall$ S $\subseteq$ E, $\forall$ s_1,s_2 $\in$ S, f(s_1) = f(s_2) = T implies S $\in \mathcal{C}$.

The edges in $\mathcal{G}$ are as before. Note that this digraph is *not* a Hasse diagram of the poset induced by{0,T,1} on $\mathcal{G}$. If events e_1 and e_2 can be executed in parallel, i.e., they are not related by $<$ and there is a set in $\mathcal{C}$ which contains both of them, it is possible to move directly from the start state to a state in which both the events are active. This state is the join of the states in which the events are started in a sequence. But as the join state can be reached in 'one transition' the digraph contains an explicit edge. A pictorial representation is given in figure 3.

Proposition 1 *Given a path P in $\mathcal{G}$. Let S be set of completed events in P, i.e., all e such that there is a f in P and f(e) = 1. S is a valid configuration of the underlying event structure.*

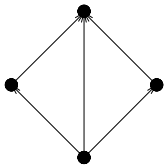

Figure 3: Digraph which is not a Hasse Diagram

In summary we represent each 'cell' in an n-complex as defined in [3] by a function from the set of events to {0,T,1}. The dimension of the cell can be derived from its function representation by counting the events mapped by f to T.

This establishes the compilation of an event structure into a geometric automaton. The process of decompilation is simple. The poset induced by the digraph on the set of functions without a T in their range can be used to generate the set of events and the precedence relation. To construct the two element sets of $\mathcal{C}$, consider exactly 2 events such that is a function which maps only the two elements to T. In general, to construct the n-element sets of $\mathcal{C}$, consider exactly n events such that there is a function which maps all and only those to T. This together with compilation is a duality in that compilation composed with decompilation is the identity.

4 Non-Persistent Choice

In this section we consider the effect of the conflict. As $(e_i \# e_j)$ means only one of the two can occur, modelling persistent choice is trivial. Conflict between events is called persistent if $(e_i \# e_j)$ and $(e_j < e_k)$ then $(e_i \# e_k)$. Adding persistence to prime event structures can result in incoherencies. For example, if $e_i < e_k$ and $e_j < e_k$ then $e_k \# e_k$. This occurs because in a prime event structure, (e # e′ < e″) implies e # e″. In other words, in a prime event structure the choice is persistent while the causal ordering indicated that either of e_i or e_j can cause e_k. Hence making the choice disappears when exhibiting event e_k. This means that a structure of the form $(e_i + e_j)$; e_k is disallowed in a prime event structure. One restricts the type of elements in $\mathcal{G}$, to those that satisfy $\forall$ f $\in \mathcal{G}$, (0 < f(e_i) implies f(e_j) = 0) and (0 < f(e_j) implies f(e_i) = 0). In other words, if an event is activated, the other event shall always be dormant.

To model non-persistent choice, [5] defines an enabling relation on a set of events. The only restriction on the relation # is that it be irreflexive and symmetric. That is, an event structure is a triple (E, #, $\vdash$) where E is a set of events, # a binary irreflexive relation (conflict) and $\vdash \subseteq$ (Con $\times$ E) an enabling relation where Con is set of finite conflict free subsets of E. Thus the event structure $\begin{array}{ccc} e_i & \# & e_j \\ \searrow & & \swarrow \\ & e_k & \end{array}$ can be considered to be joining

(by $e_i \# e_j$) of the two structures $e_i \rightarrow e_k$ and $e_j \rightarrow e_k$. The difference between joining and not joining can be explained intuitively as in the first process, there is only one structure attached to (e_i # e_j) while the second consists of two distinct structures. Pratt in [3] indicates that the first structure has a hole (where e_i and e_j cannot be active) while the second structure has a tear. A way to convert a tear into a hole is to get hold of the ends of the tear and stitch them together.

Definition: 5 *Given an automaton $\mathcal{G}$, if*

- *there exists an edge from f to g such that f(e_i) = T and g(e_i) = 1 and f(e_j) = g(e_j) = 0 and*
- *there exists an edge from h to k such that h(e_j) = T and k(e_j) = 1 and h(e_i) = k(e_i) = 0*

then JOIN($\mathcal{G}$, e_i,e_j) is an automaton such that for every g′ with an edge from k to g′ there exists an edge from g to g′ and for every k′ with an edge from g to k′ there exists an edge from k to k′ Furthermore all edges in $\mathcal{G}$ are preserved.

The JOIN operation adds edges from one branch to another thus simulating the stitching.

It is easy to see where the stitches occur. If there is an *edge* from a function f where f(e) = 0 to another function g where g(e) = 1, one can conclude that it is a stitch. Otherwise, all successors to a function changed a 0 to a T.

Proposition 2 *Given a automaton $\mathcal{G}$ with a tear due to e_1 and e_2, then JOIN($\mathcal{G}$, e_1, e_2) has a hole due to e_1 and e_2.*

The dual of a general $\mathcal{G}$, is an extended event structure and it can be generated as follows.

Assert $e_i \# e_j$ iff for all paths $f_0, f_1, f_2 \ldots$ where f_0(e) = 0 for all e $f_k(e_i) = 0$, $f_l(e_i)$ = T and $f_m(e_i) = 1$ (with $k < l < m$) implies that for all n $f_n(e_j) \neq$ T.

The following conditions can be used to generate the valid configurations S $\in$ Con. All single event sets are in Con. S $\in$ Con iff $e_1, e_2 \in$ S if there exists a path $f_0, f_1, f_2 \ldots$ such that $f_i(e_1)$ = T , $f_j(e_2)$ = T and $f_k(e_i)$ = $f_k(e_j) = 1$ with i and j less than k.

To generate the enabling relation $S \vdash e_i$ iff there is a path $f_0, f_1, f_2 \ldots$ such that for all $e \in S$ ($f_j(e)$ = T or $f_j(e) = 1$) and $f_j(e_i) = 0$ and there is an edge from f_j to f_k such that $f_k(e_i)$ = T.

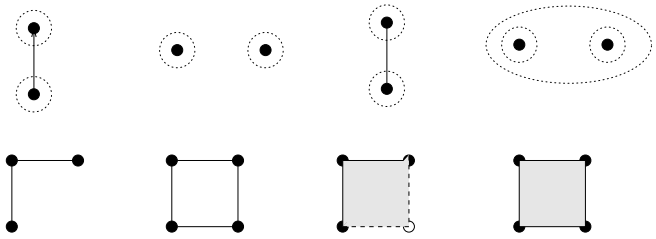

Figure 4: Examples With Resource Constraints

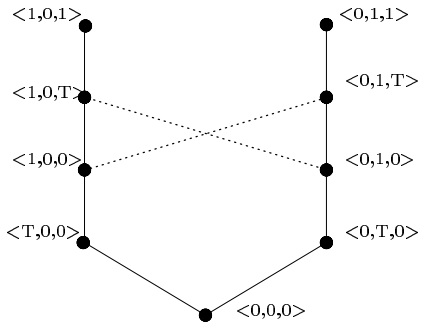

Figure 5: Example with Stitches

5 Examples and Conclusions

The first set of examples is presented in figure 4. The sequence of figures at the top presents the event structure pictorially while the sequence of figures at the bottom represents the corresponding geometric automaton. We have drawn the geometric automata using the Euclidean representation. The digraph is easily generated.

The leftmost figure indicates that events e_1 and e_2 are ordered, furthermore, only one can be active at any given time. This is the usual sequencing. The next figure represents the usual interleaving, i.e., the events can be started in any order, but only one can be active at any given time. The third sub-figure requires e_1 to start before e_2 and e_1 to finish before e_2 but they can be active together. The constraint removes the edge where e_2 starts before e_1 and the edge where e_2 finishes before e_1. This structure cannot be presented using the standard n-complex idea. If we restrict ourselves, as Pratt suggests [3], to homotopy classes, we lose the ordering requirements on starting and finishing.

The rightmost figure represents true concurrency.

Our next example illustrates an automaton with stitches. Consider an extended event structure where E is {a,b,c}, (a # b) Con = {(a,c),(b,c)} with {a} $\vdash$ c and {b} $\vdash$ c. Only the empty set and single element sets are members of $\mathcal{C}$. The geometric automaton is represented in figure 5. In it the dotted lines represent the stitches. The automaton with the stitches represents the process $(a+b);c$. To obtain a geometric automaton which represents the process $(a;c+b;c)$, the stitches from the above automaton are removed.

In conclusion, a digraph automaton contains the information relevant to causality, concurrence and choice that a geometric automaton or a higher dimensional automaton contains. The digraph automaton has a simpler presentation. We have shown how an explicit transition state can be used to characterise resource constraints. We have also indicated the dual between certain digraph automata and extended event structures.

References

[1] R. Gerber and I. Lee. Specification and Analysis of Resource-Bound Real-Time Systems. In J. deBakker, editor, *Proceedings of the REX Workshop on Real-Time: Theory in Practice: LNCS 600*, pages 371–396. Springer Verlag, 1991.

[2] R. Milner. *Communication and Concurrency.* Prentice Hall International, 1989.

[3] V. Pratt. Modelling Concurrency with Geometry. In *Conference Record of the ACM Symposium on Principles of Programming Languages*, pages 311–322, 1991.

[4] V. Pratt. Arithmetic + Logic + Geometry = Concurrency. In *LATIN'92 LNCS 583*, pages 430–447, Sao Paulo, Brazil, April 1992. Springer-Verlag.

[5] G. Winskel. An Introduction to Event Structures. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354*. Springer Verlag, 1989.