The Report Committee for Himanshu Chauhan
certifies that this is the approved version of the following report:

Necessary and Sufficient Conditions on Partial Orders for Modeling Concurrent Computations

APPROVED BY

SUPERVISING COMMITTEE:

Vijay K. Garg, Supervisor

C. Greg Plaxton

Necessary and Sufficient Conditions on Partial Orders for Modeling Concurrent Computations

by

Himanshu Chauhan, B.S.

REPORT

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ENGINEERING

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2014

Necessary and Sufficient Conditions on Partial Orders for Modeling Concurrent Computations

Himanshu Chauhan, M.S.E.
The University of Texas at Austin, 2014

Supervisor: Vijay K. Garg

Concurrent computations have been modeled using partial orders in both event based and state based domains. We give necessary and sufficient conditions on partial orders for them to be valid state based or event based models of concurrent computations. In particular, we define notions of width-extensibility and interleaving-consistency of partial orders, and show that a partial order can be valid state based model of a concurrent computation *iff* it is width-extensible. Distributed computations that involve asynchronous message passing are a subset of concurrent computations. For asynchronous distributed computations, a partial order can be a valid state based model iff it is width-extensible and interleaving-consistent. We show a duality between the event based and state based models of concurrent computations, and give algorithms to convert partial orders from the event based domain to state based domain and vice-versa.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A distributed computation is usually modeled as a set of events ordered by the partial order relation called the happened-before [8] relation. For many applications such as global predicate detection [6] and checkpointing [10], a distributed computation has also been modeled as a partial order on the states of processes. Events and states are fundamentally different concepts. Events are instantaneous and states have duration. The state captures values of all the variables (including program counter) at a process, whereas the event captures the transition of the system from one state to the other. [1] Although, there are multiple reports which model computations as partially ordered sets (posets), there is no clear theory that brings out the distinction between posets for event based computations and state based computations. For example, consider the poset in Fig. 3.3 on page 12. Is it a valid event based (or state based) model for some computation? What is the class of posets that characterize event based and state based models? Any model of a concurrent computation must also define the notion of a consistent global state. Are the definitions different in state based and event based models?

[1]Alternatively, one may model states as instantaneous and events with duration. The point is that either the state or the event must be modeled with duration.

One of the main goals of this report is to answer these questions. We study the relationship between the event based models and state based models, characterize the exact class of posets that can be used to model computations in either framework and propose a duality between the two models that allows easy translation of algorithms from one model to the other.

We model a computation in the event domain as a labeled poset in which every event is labeled with the subset of (sequential) processes on which that event is shared. The labeling of events must satisfy the condition that all events that have a label for process P_i must be totally ordered. The notion of shared events is similar to models such as CSP [2] and CCS [9]. In these computations, two or more processes may execute a shared event resulting in state transition for them. For example, a distributed computation with blocking sends (in which the sender waits for the receiver to be ready to receive the message) can be modeled using shared events. We give an equivalent state based model and show that a state based poset is a valid model of a concurrent computation *iff* it is width-extensible (width-extensibility is defined in Chapter 3). The poset in Fig. 3.3 is not width-extensible and hence it is not a valid state based model.

The notion of consistent global states in the event based model corresponds to down-sets (or, order ideals) of the poset for that model. It is well known that the set of down-sets forms a distributive lattice [2]. Conversely,

[2]Concepts related to posets are defined in Chapter 2

Birkhoff has shown that every finite distributive lattice can be generated as the set of down-sets of a poset [1]. Thus, finite distributive lattices completely characterize the set of consistent global states in the event based model. The notion of consistent global states in the state based model is different and corresponds to width-antichains. Dilworth [5] proved that the set of all width-antichains also forms a distributive lattice and Koh [7] has shown that every finite distributive lattice can be generated as the set of width-antichains of a poset. The lattice of width-antichains is in general a sublattice of the lattice of down-sets. Thus, the notion of consistent global states is different in the event based models and the state based models, a distinction that has not been explored in distributed computing literature. Table 1.1 captures the relationship between the structure of consistent global states in the state model and the event model. Distributed computations in which communication between processes is achieved by asynchronous message passing are a special case of our analysis. Using the event set notion, we can say that an *asynchronous distributed computation* is one in which the event set of different processes are disjoint. Such a computation is usually modeled by the Lamport's happened-before poset of events. Even though, such computations are sometimes modeled using happened-before poset of states [6], the relationship between the poset for events and state is not clear in the literature. In this report, we define a class of posets called width-extensible and interleaving-consistent and show that a state based poset is a valid model of an asynchronous distributed computation *iff* it is width-extensible and interleaving-consistent.

The key contributions of this work are summarized below.

- we establish the correspondence between event based and state based models of concurrent computations.

- we provide complete characterization, by establishing necessary and sufficient conditions, of partial orders that can model concurrent computations under the state based model. We prove that the class of width-extensible posets forms a complete characterization of the state based models of concurrent computations, and the class of width-extensible and interleaving-consistent posets forms a complete characterization of the state based models of asynchronous distributed computations.

- we give algorithms to translate event based models to state based models and vice-versa. We establish the correspondence between the notions of the consistent global states in the event based and the state based models.

This report is organized as follows. Chapter 2 gives background definitions for posets. Chapter 3 gives the definition of concurrent computation in the event based and the state based model. Chapter 4 gives the definition of asynchronous distributed computation in the event based and the state based model. Appendix A gives proofs of some of the technical lemmas.

	Event Based Model	**State Based Model**
Poset	Any poset	Must be 'width-extensible'
Consistent Global State	Downset in poset	Width-Antichain in poset
Concurrent Computation	'Shared' events allowed	No 'shared' states
Asynchronous Distributed Computation	No 'shared' events	Must be 'interleaving-consistent' and 'width-extensible' poset

Table 1.1: Characterization of Posets and Concepts under the two models

Chapter 2

Background: Posets and Lattices

We assume that the reader is familiar with the basic concepts of posets and lattices [4]. A partially ordered set (or *poset*) is a pair $P = (X, \leq)$ where X is a set and $\leq$ is a reflexive, antisymmetric, and transitive binary relation on X. We write $x \leq y$ when $(x, y) \in P$. If either $x \leq y$ or $y \leq x$, we say that x and y are *comparable*; otherwise, we say x and y are *incomparable* or *concurrent*, and denote this relation by $x \mid\mid y$. A subset $Y \subseteq X$ is called an *antichain (chain)*, if every distinct pair of points from Y is incomparable (comparable) in P. The *width* (*height*) of a poset is defined to be the size of a largest antichain (chain) in the poset. All antichains of size equal to the width of the poset are called *width-antichains* in this report. Let $\mathcal{A}(P)$ denote the set of all width-antichains of P. Order $\leq$ is defined over $\mathcal{A}(P)$ as:
$A \leq B \;\; (A, B \in \mathcal{A}(P))$ iff $\forall a \in A, \exists b \in B : a \leq b$ in P.

Given a subset $Y \subseteq X$, the *meet* of Y, if it exists, is the greatest lower bound of Y and the *join* of Y is the least upper bound. A poset $P = (X, \leq)$ is a *lattice* if joins and meets exist for all finite subsets of X. Let P be a poset with a given chain partition of width w. In a distributed computation, P would be the set of events executed under the happened-before partial order. Each chain would correspond to a total order of events executed on a single process. In such a poset, every element e can be identified with a tuple (i, k) which represents the kth event in the ith process.

A subset Q is a downset of P if it satisfies the constraint that if f is in Q and e is less than or equal to f, then e is also in Q. In distributed computing, when a distributed computation is modeled as a poset of event, the downsets are called *consistent cuts*, or *consistent global states* [3]. The set of downsets is closed under both union and intersection and therefore forms a lattice under the set containment order.

Chapter 3

Model of Concurrent Computations

In this chapter we give event and state based models of concurrent computations. The definition of the event based model is quite standard; our contribution is in the definition of the state based model.

3.1 Event Based Model

A concurrent computation is usually modeled as a set of events E together with a partial order *happened-before* [8] denoted by $\rightarrow$. However, implicit in this model is the decomposition of these events into chains denoting the processes where these events are executed. We make this decomposition explicit in our model because the translation of the event based model into the state based model depends upon the decomposition.

Definition 1 (Event Based Concurrent Computation)**.** *A concurrent computation on n processes, $\hat{E}$, is a tuple $(E, \rightarrow, \pi)$ where E is the set of events, $\rightarrow$ is an irreflexive partial order relation on E, and π maps every event to a subset of processes from $\{1..n\}$ such that for all $i \in \{1..n\} : E_i = \{e \in E \mid i \in \pi(e)\}$ is totally ordered under $\rightarrow$.*

Thus, π maps events on a single process to a total order, such that

E_i is the totally ordered set of events executed on process P_i. Note that an event could be assigned to multiple processes. If an event $e \in E_i \cap E_j$, then e is a 'shared' event for process P_i and P_j.

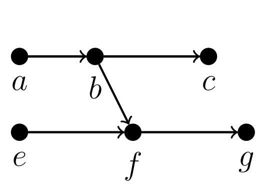
Fig. 3.1(a) shows a model of a concurrent computation on two processes in the event based model for message passing use case in which there is no shared memory between the processes and communication is only through messages. Fig. 3.2(a) shows a concurrent computation on two processes that synchronize using a barrier. Note that the model of Fig. 3.2(a) allows us to represent synchronous messages where the sender blocks for the receiver to be ready. Such synchronous messages are represented by a single event e such that $\pi(e)$ includes the sender as well as the receiver. The model also allows us to represent barriers which require multiple processes to wait until all the processes participating in the barrier execute it. It can also model behavior of finite communicating sequential processes [2].

Note: In all the figures throughout this report, events are depicted with dark filled circles, and states are depicted with empty circles.

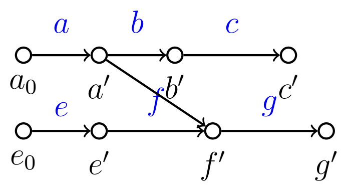
Definition 2 (Consistent Global State). *Under the event based model, for a computation* $(E, \rightarrow, \pi)$*,* $G \subseteq E$ *is a* consistent global state *of the computation if*

$$\forall e, f \in E : (f \in G) \wedge (e \rightarrow f) \Rightarrow (e \in G)$$

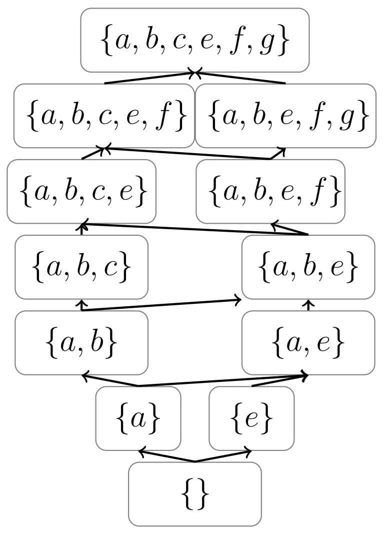
Note that this definition of is independent of π and coincides with the definition of a *down-set* of a poset [4]. Fig. 3.1(a) shows a computation under the event based model, and the corresponding consistent global states of the computation are shown (in a lattice format) in Fig. 3.1(c).



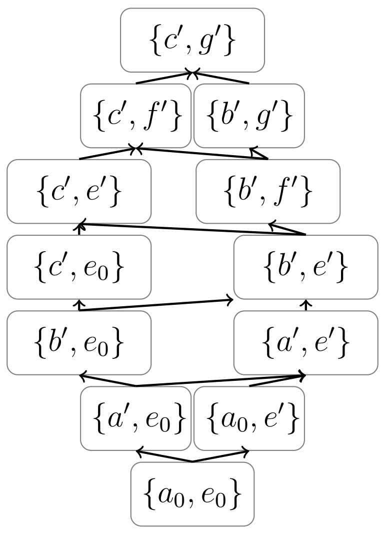
(a) Event Based Model



(b) State Based Model



(c) Consistent Global States in Event Model



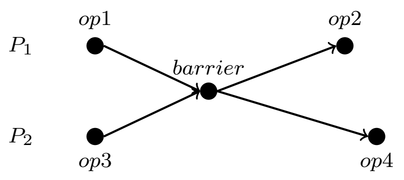
(d) Consistent Global States in State Model

Figure 3.1: Example: Consistent Global States of a Computation

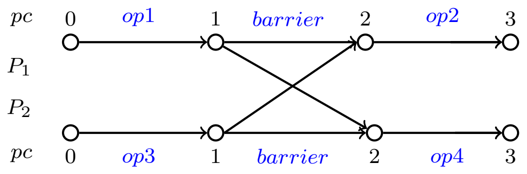
3.2 State Based Model

For many applications such as concurrent debugging, it is more natural to model a concurrent computation based on states rather than events. For example, we may be interested in the global state in which all processes have taken their local checkpoint. We first give an intuition for state based model of concurrent computations.

The happened-before diagram based on events in Fig. 3.1(a) is equivalent to the diagram in Fig. 3.1(b), which is based on states. An event is always executed on some state, and the state before the event's execution 'existed-before' the state resulting from the execution. In Fig. 3.1(a) event execution of a gets translated into an edge between two states, a_0 the state that existed before event a was executed, and state a' - the state post a's execution. Thus, we have $a_0 < a'$.



(a) Barrier in event based model



(b) Equivalent barrier in the state based model

Figure 3.2: Event and State based models for Barriers

From now on, we restrict ourselves to states generated during an execution of the program. Although some concepts carry over from events to states, there are some important differences. For example, any partially ordered set of events in which all events on a single process are totally ordered is a concurrent computation in the happened-before model. However, not every poset of states is a valid concurrent computation. Consider the state based computation shown as a partial order in the example of Figure 3.3. In this example, although the states form partial orders, the induced graph in the event based model has a cycle. The techniques involved for finding the induced graph and detecting the cycles are shown in the next chapter. Thus we can allow only those partial orders on states that do not induce cycle on the order on events.

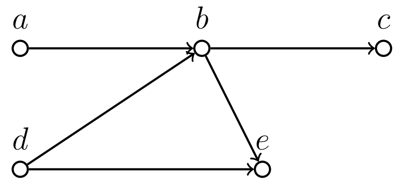


Figure 3.3: Poset on states with no valid event based poset

We claim that a poset can only be a valid state based model of a concurrent computation if it satisfies a notion called *width-extensibility*.

Definition 3 (Width-extensible Poset)**.** *A poset $(X, <)$ is width-extensible if and only if for every antichain $A \subseteq X$, there exists a width-antichain W containing A.*

Informally, when states of a concurrent computation are modeled as a poset, this property requires that for any set of incomparable local states there is a possible consistent global state that includes these local states. We will show later that in the state based model, the consistent global states correspond to width antichains (and not down-sets).

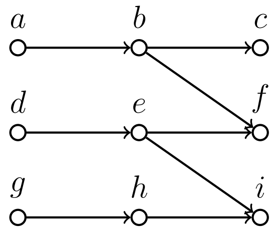


Figure 3.4: A poset that is not *width-extensible*

The poset in Fig. 3.3 is not width-extensible because there is no width-antichain that contains b. Whereas the posets in Fig. 3.1(b) and 3.2(b), both are width-extensible.

In the above definition we can not substitute "for all antichains" by "for all antichains of size 1". In the example of Fig. 3.4, there is a width-antichain for every individual element a to i. This can be easily verified as $\{a, d, g\}$, $\{b, e, h\}$, and $\{c, f, i\}$ are all width-antichains. However, there is no width-antichain that contains $\{b, i\}$.

However, we show that it is sufficient to restrict our attention to antichains of size two.

Theorem 1. *A poset $(X, <)$ is width extensible if and only if for every antichain A of size at most two, there exists a width-antichain W containing A.*

Proof. The necessity is obvious. We prove sufficiency. We want to prove that if every antichain of size at most two is contained in a width-antichain, then every antichain (of any size) is also contained in a width-antichain. Let w be the width of the poset $(X, <)$ and $\{C_1, C_2, ..., C_w\}$ be a chain decomposition of size w. Consider an antichain A of size $k, 3 \leq k \leq w$. If $w = k$, then A itself is a width-antichain, and we have the result. Suppose $w > k$, and A is not contained in any width-antichain. Hence, there is some chain C_i such that A does not have any elements from C_i. We know that for any pair of elements $a, b \in A$, with $a \neq b$, the antichain $\{a, b\}$ is width-extensible. Let $I_i(a, b)$ denote the maximal interval on C_i that contains all the elements that are incomparable to both a and b. As $\{a, b\}$ is width-extensible, we know that $I_i(a, b)$ is non-empty. Now consider $a, b, c \in A$, where all three are distinct. The width-extensibility of size two antichains guarantees that $I_i(a, b)$, $I_i(b, c)$, and $I_i(a, c)$ are all non-empty. Since every pair of these intervals have non-empty intersection, and all intervals are set of one or more consecutive states in C_i, we get that $I_i(a, b) \cap I_i(b, c) \cap I_i(a, c) \neq \phi$. This means that $\exists d \in C_i : (d||a) \wedge (d||b) \wedge (d||c)$, ie. d is concurrent to a, b, and c. Hence, d can be added to A. By repeating this argument for all chains that do not have any element in A, we can extend A to a width-antichain.

□

We can now define the state based model of a concurrent computation as following:

Definition 4 (State based Concurrent Computation). *A concurrent computation on n processes, $\hat{S}$, is a tuple $(S, <, \pi)$ where S is the set of local states, $(S, <)$ is a width-extensible poset, and π is a map from S to $\{1..n\}$ such that for all distinct states $s, t \in S$ for all $i \in \{1..n\}, S_i = \{s \in S | i \in \pi(s)\}$ is totally* *ordered under* $<$.

$$\pi(s) = \pi(t) \Rightarrow (s < t) \vee (t < s)$$

Thus, π partitions S such that every block of the partition S_i is totally ordered. The relation $<$ between states captures the 'existed-before' notion discussed earlier. Fig. 3.1(b) and 3.2(b), are state based models of event based computations shown in Fig. 3.1(a) and 3.2(a). Note that in these figures (of state based models), the events are shown as edge labels above the edges that capture $<$ (existed-before) relation on the states. We now show the difference in the definitions of consistent global states in the state based and event based model.

Definition 5 (Consistent Global State). *Under the state based model of a concurrent computation* $(S, <, \pi)$*, a subset* $T \subseteq S$ *of size equal to the width of poset* $(S, <)$ *is a consistent global state if* $\forall s, t \in T : s||t$.

It is clear that the consistent global states correspond to width-antichains. Fig. 3.1(d) shows all the width-antichains of the state based computation of Fig. 3.1(b).

The order "$<$" is defined over consistent global states using the "$\leq$" relation defined over width-antichains in Chapter 2. Under the state based model, for any two consistent global states A, B we have: $A < B$ iff $A \leq B \wedge A \neq B$. Hence, $A < B \Rightarrow \exists a \in A, \exists b \in B : a < b$ in $(S, <)$.

At this point we have two notions of a consistent global state of a concurrent computation: one in the event based model and the other in the state based model. What is the relationship between these two definitions? We show that there is 'one-to-one' correspondence between consistent global states in the event based and the state based models.

Lemma 1. *Let* $(E, \rightarrow, \pi)$ *and* $(S, <, \pi)$ *be event and state based models of a computation. There is* $1-1$ *correspondence between consistent global states of* $(E, \rightarrow, \pi)$ *and consistent global states of* $(S, <, \pi)$.

Proof. In Appendix. $\square$

3.3 Translation between State based and Event based models

Let $\hat{E} = (E, \rightarrow, \pi)$ be a computation in the event based model. Let π decompose E into $(E_i \mid i = 1, 2, \dots n)$. For each $i = 1, 2, \dots n$, let $|E_i| = n_i (\geq 1)$. Suppose the elements of E_i are named as follows: $E_i : (i, 1) \rightarrow (i, 2) \rightarrow \dots \rightarrow (i, n_i - 1) \rightarrow (i, n_i)$. Note that if an event is 'shared' between two processes i and j, then it will have two labels (i, x) and (j, y), with $1 \leq x \leq n_i$, and $1 \leq y \leq n_j$. By extension of this rule, an event that is 'shared' between k processes would have k labels. We translate an event based model into the state based model $\hat{S} = (S, <, \pi)$ using the steps of Algorithm 1. We call this algorithm the ES transform. A special case of this transform, on disjoint chain partitions in π, was used by Koh in [7] to prove properties of lattice of width-antichains.

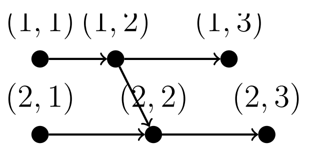
The ES transform creates $\hat{S}$ - a state based model - from an event

Algorithm 1: ES Transform

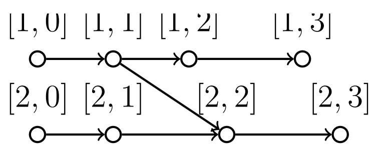
```
Input: Event Based Model $\hat{E} = (E, \rightarrow, \pi)$
Output: State Based Model $\hat{S} = (S, <, \pi)$

1  $S_i = \{\}$
2  for $i = 1 \dots n$ do
3      for $k = 0 \dots n_i$ do
4          Add $[i, k]$ to $S_i$
5      for $k = 0 \dots n_i - 1$ do
6          Define $[i, k] < [i, k+1]$ in $S_i$
   /* $S_i$ is now $(|E_i| + 1)$-element chain */
7  $\hat{S} = \cup(S_i \mid i = 1, 2 \dots n)$
8  for $i = 1 \dots n$ do
9      for $j = 1 \dots n$ && $j \neq i$ do
10         if $(i, r+1) \rightarrow (j, s)$ in $E$ then
11             Define $[i, r] < [j, s]$ in $\hat{S}$
```

based model $\hat{E}$. Hence every chain S_i contains the states of process i, such that event (i, k), $1 \leq k \leq n_i$, causes a transition from state $[i, k-1]$ to $[i, k]$ on the state chain S_i (lines $1 - 7$). On chain S_i, the state $[i, 0]$ represents the initial state of the process i, and $[i, n_i]$ being the last element, represents the final state of the process i. Lines $8 - 11$ ensure that the causal dependency induced by happened-before relation between events of different processes is translated to existed-before relation between corresponding states. An example of the transformation is shown in Figs. 3.5 and 3.6.

We show that $\hat{S}$ generated by applying the ES transform on $\hat{E}$ is a valid state based model, i.e., it is a width-extensible poset. We first show that it is a poset.

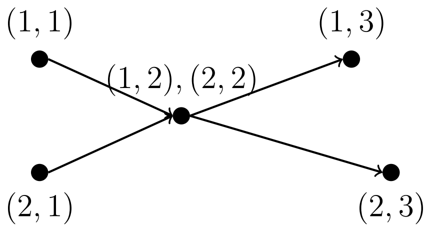


(a) $(E, \to, \pi)$

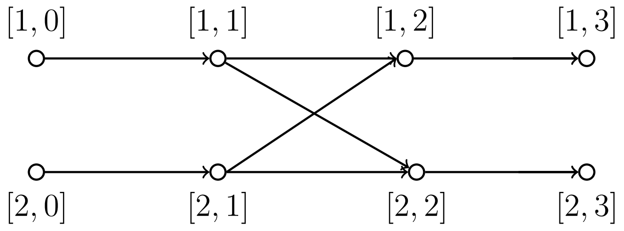


(b) $(S, <, \pi)$

Figure 3.5: Event to State transform for message passing computation of Figure 3.1a



(a) Barrier in event based model



(b) Equivalent barrier in the state based model

Figure 3.6: Event to State transform for barrier shown in Fig. 3.2

Lemma 2. *$\hat{S}$ is a poset under the "$<$" relation.*

Proof. Extension of proofs in Koh's paper [7]. In the Appendix. $\square$

We next show that $(S, <, \pi)$ constructed from any $(E, \to, \pi)$ by applying the ES transform is width-extensible.

To that end, we first observe properties $(\alpha_1), (\alpha_2)$, and (α_3) of $\hat{S}$.

Lemma 3. *If $\hat{S}$ is constructed from a event based concurrent computation using ES transformation, then:*
For $1 \leq i, j, k \leq n$

(α_1) $\forall i, j$: $[i, 0] \mid\mid [j, 0]$. *All initial states are concurrent.*

(α_2) $\forall i, j$: $[i, n_i] \mid\mid [j, n_j]$. *All final states are concurrent.*

(α_3) $\forall i, j, k$: $[i, s] < [j, t] \wedge [j, t-1] < [k, u]$, *for* $i \neq j \wedge j \neq k$, $\Rightarrow [i, s] < [k, u]$.

Proof. Extension of proofs in Koh's paper [7]. In the Appendix. □

We now describe the underlying intuition behind these conditions. The first condition (α_1) ensures that all n initial states are pairwise concurrent. This is a valid requirement as all the processes would start in some default (individual) state, and at the start of the computation these states would not have any dependency amongst them.
The second condition, given by (α_2), ensures that all n final states are pairwise concurrent. This is also a valid requirement because irrespective of the events/commands executed all the n processes end up in some individual final state at the end of the computation. Hence, when the computation is finished all the final states would not have any dependency amongst them, and thus be concurrent to each other.
The third condition (α_3) guarantees that causal dependency between events under the state based model translates to causal dependency between corresponding states under the state based model. Note that the labels of states

in the dependency relation are different from those of events. Suppose that for two events e and f, we have $e \rightarrow f$ under the event based model, $\hat{E}$. Then (α_3) translates that dependency from $\hat{E}$ to $\hat{S}$ such that the state preceding the execution of e is guaranteed to have 'existed' <u>before</u> the state that is generated <u>after</u> the execution of f.

We now show the important result that any state based model that is generated by applying the ES transform on an event based model is a valid state based model. To be a valid state based model, it is sufficient that the generated poset be width-extensible.

Theorem 2. *Let* $(S, <, \pi)$ *be any state based model of concurrent computation that satisfies* (α_1)*,* (α_2) *and* (α_3)*. Then,* $(S, <)$ *is width-extensible.*

Proof. In Appendix. □

We have now established that every poset that provides the three conditions $(\alpha_1), (\alpha_2)$, and (α_3) is width-extensible. We now prove the converse — every width-extensible poset guarantees the three conditions α_1, α_2, and α_3. The goal of proving this is to establish that α_1, α_2, and α_3 are necessary and sufficient conditions for width-extensibility.

Theorem 3. *Let* $(S, <)$ *be an width-extensible poset. Consider any chain partition* π *of* $(S, <)$*. Then,* $\hat{S} = (S, <, \pi)$ *satisfies* (α_1)*,*(α_2) *and* (α_3)*.*

Proof. We show the contrapositive. If (α_1) is violated, then there exists an initial state t such that there exists a state s different from t which is less than t. Then, s is less than all states in the process containing t. Therefore, the antichain $\{t\}$ cannot be extended to a width-antichain. The proof for (α_2) is dual. If (α_3) is violated, then there exists $[i,s]$, $[j,t]$ and $[k,u]$, where $i \neq j \wedge j \neq k$, such that $[i,s] < [j,t]$ and $[j,t-1] < [k,u]$ but $[i,s] \not< [k,u]$. We now do a case analysis on the relationship between $[i,s]$ and $[k,u]$.

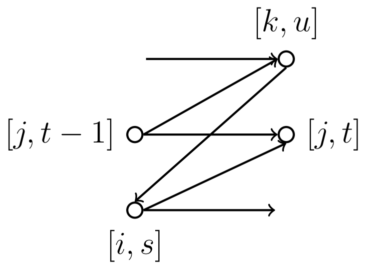


Figure 3.7: Case 1 when α_3 is violated

Case 1: $[k,u] < [i,s]$. Depicted in Fig. 3.7. In this case we claim that there is no width-antichain that contains $[i,s]$. Since $[i,s] < [j,t]$, for any state w on process P_j that is concurrent with $[i,s]$, we get $w \leq [j,t-1]$. Since $[j,t-1] < [k,u]$ none of the states on process P_k greater than $[k,u]$ are eligible to be in the width-antichain with w. Furthermore, all states less than or equal to $[k,u]$ are not eligible because $[k,u] < [i,s]$.

Case 2: $[k,u]$ is incomparable with $[i,s]$. In this case we claim that there is no width-antichain that includes both $[k,u]$ and $[i,s]$. No state greater than or equal to $[j,t]$ can be included from P_j because $[i,s] < [j,t]$. No state less than or equal to $[j,t-1]$ can be included from P_j because $[j,t-1] < [k,u]$. $\square$

Note that α_3 only requires $i \neq j \wedge j \neq k$. It is possible that $i = k$. The proof also holds for the case when $i = k$.

With these theorems, we have shown that the conditions α_1, α_2 and α_3 are necessary and sufficient for a poset to be width-extensible. We now show another important result. The following theorem establishes that width-extensibility is a sufficient condition for modeling a concurrent computation under the state based model. First, we outline how to generate an event based model of a concurrent computation from $(S, <)$. Let π be any chain partition of $(S, <)$. We construct an event based model $(E', \rightarrow, \pi')$ of a concurrent computation by applying the SE transform (a reverse transform to ES) whose steps are shown in Algorithm 2. Lines $1 - 11$ perform a reversal of steps of ES transform. Lines $13 - 18$ try to collapse events that are 'shared' between processes by performing a strongly connected component (SCC) decomposition, and using the SCCs for identifying shared events. If an SCC has events from the same process, then that results in a same process cycle - an invalid event based computation. Fig. 3.8 shows the E'_{temp} (and not the final E') generated during the execution when SE transform is applied to $(S, <, \pi)$ given by Fig. 3.6 (b). After the SCC decomposition based 'collapsing' on this E'_{temp}, the generated E' is same as Fig. 3.6 (a) Recall that we claimed invalidity of a state based model poset shown in Fig. 3.3 with the reason that such a state model would cause cycles when converted to an event based model. Let us assign state labels to the states shown in that figure:

$a = [1, 0], b = [1, 1], c = [1, 2], d = [2, 0], e = [2, 1]$.

Algorithm 2: SE Transform

```
Input: State Based Model $\hat{S} = (S, <, \pi)$
Output: Event Based Model $\hat{E} = (E, \rightarrow, \pi')$ OR Report $\hat{S}$ invalid
for $i = 1 \ldots n$ do
    $E'_i = \{\}$
    for $k = 1 \ldots n_i$ do
        Add $(i, k)$ to $E'_i$
    for $k = 1 \ldots n_i - 1$ do
        Define $(i, k) \rightarrow (i, k+1)$ in $E'_i$
; /* $E'_i$ is now $(|S_i| - 1)$-element chain */
$E'_{temp} = \cup(E'_i \mid i = 1, 2 \ldots n)$
for $i = 1 \ldots n$ do
    for $j = 1 \ldots n$ && $j \neq i$ do
        if $[i, r-1] < [j, s]$ in $S$ then
            Define $(i, r) \rightarrow (j, s)$ in $E'_{temp}$
$E' = E'_{temp}$
foreach SCC $C_s$ in SCC Decomposition of $E'_{temp}$ do
    if AllOnDifferentChain $(C_s)$ then
        Replace $C_s$ with one element $e$ in $E'$
        Assign all labels of nodes in $C_s$ to $e$ in $E'$
    else
        Report $S$ as not width-extensible
routine AllOnDifferentChain $(SCC\ \ C_s)\{$
  if each node in $C_s$ is on a different chain
    return true
  else
    return false
$\}$
```

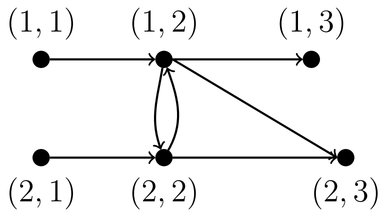


Figure 3.8: E'_{temp} generated by SE transform on $\hat{S}$ of Figure 3.2b

Now apply the SE transform of Alg. 2 to this state poset. The resulting $(E, \to)$ would be the following:

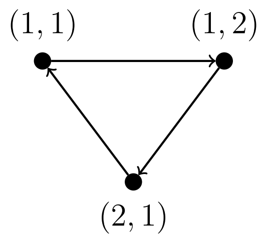


Figure 3.9: Event model generated from states of Fig. 3.3

And we know from the properties of happened-before relation that such a cycle can not exist in a valid event based model.

The next theorem shows that width-extensibility is sufficient for modeling concurrent computations under the state based model.

Theorem 4. *Let $(S, <)$ be any width-extensible poset. Then, there exists a concurrent computation for which it is the state based model.*

Proof. We show a concurrent computation in the event based model such that when we convert that event based computation to state based model, we get the poset $(S,<)$. We first create a width chain partition π of $(S,<)$ to get $(S,<,\pi)$.

We then generate an event based model $\hat{E'} = (E', \rightarrow, \pi')$ from $(S,<,\pi)$ using SE transformation. It can be easily verified that applying the ES transformation of Alg. 1 to $(E', \rightarrow, \pi')$ leads to $(S,<,\pi)$. It suffices to show that $(E', \rightarrow)$ is a partial order.

Irreflexivity: Assume, $(i,r) \rightarrow (i,r)$ in $E'(\pi')$. This would require $r<r$ in $\hat{S}$ - a contradiction.

Transitivity: Consider $(i,r) \rightarrow (j,s) \wedge (j,s) \rightarrow (k,t)$ where $i \neq j \wedge j \neq k$. The first relation in the event based model is possible only if $[i,r-1] < [j,s]$ in $\hat{S}$. Similarly, we also get $[j,s-1] < [k,t]$. Hence:

$$[i,r-1] < [j,s] \wedge [j,s-1] < [k,t]$$

By using (α_3) on $\hat{S}$ we have

$$\Rightarrow [i,r-1] < [k,t]$$

$$\equiv (i,r) \rightarrow (k,t)$$

in E'.

When $i=j=k$, the transitivity of states on the one chain can be shown trivially. Now let us consider the case when $i=j \wedge j \neq k$. Then, $(i,r) \rightarrow (j,s) \wedge (j,s) \rightarrow (k,t)$ in E' requires $r<s$, as $i=j$, and $[j,s-1] < [k,t]$ in $\hat{S}$.

Observe that $i = j$ and $r < s$ means that $r-1$, $s-1$, s form a totally ordered set, such that $r-1 \leq s-1$. Hence, we get $[i, r-1] \leq [j, s-1] \wedge [j, s-1] < [k, t]$. By transitivity of $<$ in $\hat{S}$, this leads to $[i, r-1] < [k, t]$ which is the desired condition for $(i, r) \rightarrow (k, t)$ in E'. The proof for the case of $i \neq j, j = k$ is similar.

$\square$

Thus, $(\alpha_1) - (\alpha_3)$ provide a complete characterization of a state based model for a concurrent computation. In the next chapter, we discuss asynchronous distributed computations, and show that their state based models are a special case of models of concurrent computations formalized in this chapter.

Chapter 4

Modeling Asynchronous Distributed Computations

Asynchronous distributed computations, such as message-passing, can be considered a special case of our model in which there are no 'shared' events. Thus, the event based model is defined based on a chain decomposition π in which all chains are disjoint. We use a short form notation ADC for 'asynchronous distributed computation' from here on. The event based model of ADCs is given by the following definition:

Definition 6 (Event based model of ADC). *An asynchronous distributed computation on n processes, $\hat{E}$, is a tuple $(E, \rightarrow, \pi)$ where E is the set of events, $\rightarrow$ is the happened-before relation on E, and π is a map from E to $\{1..n\}$ such that for all distinct events $e, f \in E$*

$$\pi(e) = \pi(f) \Rightarrow (e \rightarrow f) \vee (f \rightarrow e)$$

Thus, π partitions E such that every block of the partition is totally ordered under $\rightarrow$.

Such an event based model - with no 'shared' events - leads to a state based model that satisfies some stronger properties than those satisfied by the state based model of the previous section. Given that the communication between processes is asynchronous – no two processes can make a 'jump' together from their individual states to next states as if there was a 'shared' execution. Hence, the poset $(S, <)$ exhibits a property that we call 'interleaving-consistency'.

Definition 7 (Interleaving-consistent Poset). *A poset $(X, <)$ is interleaving-consistent if for every width-antichain W that is not equal to the biggest width-antichain, there exists a width-antichain $W' > W$ such that $|W \cap W'| = |W| - 1$.*

Let $\mathcal{A}(X)$ be the set of all width-antichains of a poset $(X, <)$. The biggest width-antichain of $(X, <)$ is the width-antichain $A \in \mathcal{A}(X)$ such that $\nexists A' \in \mathcal{A}(X) : A < A'$. Informally, interleaving-consistency requires that any possible global state (modeled as a width-antichain) can be advanced on some process to reach another possible global state. Fig. 3.5(a) shows an ADC under the event based model, and the corresponding poset of the state based model in Fig. 3.5(b) is interleaving-consistent. However, the event based computation in Fig. 3.6(a) is not an ADC, and thus the resulting state based model's poset in Fig. 3.6(b) is not interleaving-consistent - the processes make a 'jump' together from states $[1, 1], [2, 1]$ to $[1, 2], [2, 2]$. Interleaving-consistency in a poset is captured by the following rule:
For $1 \leq i, j \leq n$, if $i \neq j$, then $[i, s-1] < [j, t] \Rightarrow \neg([j, t-1] < [i, s])$.

Thus, for an ADC, a poset $(S, <)$ that models its states is now characterized by:
For $1 \leq i, j, k \leq n$

(α_1) $\forall i, j$: $[i, 0] \mid\mid [j, 0]$.

(α_2) $\forall i, j$: $[i, n_i] \mid\mid [j, n_j]$.

(α_3) $\forall i, j, k$: $[i, s] < [j, t] \wedge [j, t-1] < [k, u]$, for $i \neq j \wedge j \neq k$, $\Rightarrow [i, s] < [k, u]$.

(α_4) if $i \neq j$, then $[i, s-1] < [j, t] \Rightarrow \neg([j, t-1] < [i, s])$.

We say that ADCs are a special case of a concurrent computations as they form a subset of concurrent computations completely characterized by $(\alpha_1) - (\alpha_3)$ that has an additional characteristic of (α_4).

The state based model for ADCs is formally defined as:

Definition 8 (State-Based Model of ADC). *An asynchronous distributed computation on n processes, $\hat{S}$ is a tuple $(S, <, \pi)$ where S is the set of states, $<$ is an irreflexive partial order relation on S such that $(S, <)$ is a width-extensible and interleaving-consistent poset, and π maps every state to a process from $\{1..n\}$ such that for all $i \in \{1 \ldots n\}, S_i = \{s \in S | i \in \pi(s)\}$ is totally ordered under $<$.*

Lemma 4. *Suppose $\hat{S} = (S, <, \pi)$ is obtained by applying ES transform on a valid event based model $\hat{E} = (E, \rightarrow, \pi)$. Then $\hat{S}$ satisfies (α_1) - (α_4).*

Proof. Proofs for (α_1) - (α_3) were shown in previous section. Suppose $(S, <)$ doesn't satisfy (α_4) and thus we have $[i, s-1] < [j, t] \Rightarrow ([j, t-1] < [i, s])$ in $(S, <)$. But this would require $(i, s) \rightarrow (j, t) \wedge (j, t) \rightarrow (i, s)$ in E - a contradiction. $\square$

Lemma 5. *Let $\hat{S} = (S, <, \pi)$ be as defined in Lemma 4. Then, $(S, <)$ is interleaving-consistent.*

Proof. Suppose $(S, <)$ satisfies (α_4), but is not interleaving-consistent. Hence, there is some antichain A of $(S, <)$ that is not the biggest, and still can not be extended along just one process to form another antichain A'. Let $[i, a_i]$ denote be the element from chain i that belongs to A. Our assumption means that $\nexists i : A - \{[i, a_i]\} + \{[i, a_i + 1]\}$ is a width-antichain. Hence $\forall i, \exists j \neq i : [i, a_i] < [j, a_j + 1]$. Given that S is finite (we can not keep on finding a 'new' j for every 'new' i we consider), we know that to satisfy this requirement there must exist $k, k \neq j \wedge k \neq i$ such that $[j, a_j] < [i, a_i + 1] \wedge [k, a_k] < [j, a_j + 1] \wedge [i, a_i] < [k, a_i + 1]$. See Fig. 4.1 for an illustration. From the previous Lemma, we know that $(S, <)$ satisfies

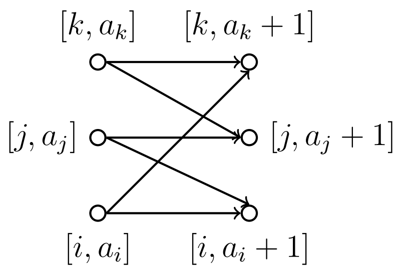


Figure 4.1: Illustration for proof of Lemma 5

$(\alpha_1) - (\alpha_3)$. Applying (α_3) to this we get $[k, a_k] < [i, a_i + 1]$. But this leads to $[i, a_i] < [k, a_i + 1] \wedge [k, a_k] < [i, a_i + 1]$ - a contradiction with α_4. ☐

Theorem 5. *Let $(S, <)$ be any width-extensible, and interleaving-consistent poset. Consider any chain partition π of $(S, <)$. Then, $\hat{S} = (S, <, \pi)$ satisfies*

$(\alpha_1) - (\alpha_4)$.

Proof. Width-extensibility guarantees $(\alpha_1) - (\alpha_3)$. Suppose (α_4) is not satisfied, i.e., there exist $[i,s], [j,t]$, for $i \neq j$, such that $[i,s-1] < [j,t] \wedge ([j,t-1] < [i,s])$. Let $[j,r]$ be the largest state on S_j that is incomparable with $[i,s-1]$. Note that $r \leq t-1$. It is clear that $[j,r] < [j,t]$ because $[i,s-1] < [j,t]$. Since $([j,t-1] < [i,s])$, we also get that $[j,r] < [i,s]$.

Let $\mathcal{W}$ be the set of all width-antichains that include both $[i,s-1]$ and $[j,r]$. Let A be the biggest antichain in $\mathcal{W}$. We claim that there does not exists any width-antichain $A' \geq A$ such that $|A' - A| = 1$, and thus not satisfying (α_4), contradicts with interleaving-consistency. If A' differs from A on a chain different from i and j, then it violates that A is the biggest antichain that contains $[i,s-1]$ and $[j,r]$. Hence, to satisfy interleaving-consistency, A' must differ from A on either i, or j. Suppose $A' - A = [i,s]$ then because A' is a width-antichain – we get that $[j,r]$ is incomparable with $[i,s]$, a contradiction. If $A' - A = [j,r+1]$, then we get that $[i,s-1]$ is incomparable with $[j,r+1]$, which contradicts the definition of $[j,r]$.

□

Theorem 6. *Let $(S,<)$ be any poset that is width extensible and interleaving-consistent. Then, there exists an ADC for which it is the state-based model.*

Proof. Let π be any chain partition of $(S,<)$. Apply SE transform of Alg. 2 to generate an event based model $(E', \rightarrow)$ from $(S,<,\pi)$. It is trivial to verify that applying ES transform of Alg. 1 to $(E', \rightarrow)$ leads to $(S, <)$. It suffices to show that $(E', \rightarrow)$ is a partial order.

Irreflexivity: Assume, $(i, r) \rightarrow (i, r)$ in $E'(\pi)$. This would require $r < r$ in $\hat{S}$ - a contradiction.

Transitivity: Consider $(i, r) \rightarrow (j, s) \wedge (j, s) \rightarrow (k, t)$ where $i \neq j \wedge j \neq k$. The first relation in the event based model is possible only if $[i, r-1] < [j, s]$ in $\hat{S}$. Similarly, we also get $[j, s-1] < [k, t]$. Hence:

$$[i, r-1] < [j, s] \wedge [j, s-1] < [k, t]$$

By using (α_3) on $\hat{S}$ we have

$\Rightarrow [i, r-1] < [k, t] \equiv (i, r) \rightarrow (k, t)$ in E'.

When $i = j = k$, the transitivity of states on the one chain can be shown trivially. Now let us consider the case when $i = j \wedge j \neq k$. Then, $(i, r) \rightarrow (j, s) \wedge (j, s) \rightarrow (k, t)$ in E' requires $r < s$, as $i = j$, and $[j, s-1] < [k, t]$ in $\hat{S}$. Observe that $i = j$ and $r < s$ means that $r-1$, $s-1$, s form a totally ordered set, such that $r-1 \leq s-1$. Hence, we get $[i, r-1] \leq [j, s-1] \wedge [j, s-1] < [k, t]$. By transitivity of $<$ in $\hat{S}$, this leads to $[i, r-1] < [k, t]$ which is the desired condition for $(i, r) \rightarrow (k, t)$ in E'. The proof for the case of $i \neq j, j = k$ is similar.

Finally, consider the case when $i = k, i \neq j \wedge r = t$. In this case, the left hand side of $(i, r) \rightarrow (j, s) \wedge (j, s) \rightarrow (k, t)$ is equivalent to $[i, r-1] < [j, s] \wedge [j, s-1] < [i, r]$ as $i = k, r = t$. However note that (α_4) prohibits this case. Hence, the left hand side is false; and thus the condition is trivially true.

$\square$

Chapter 5

Conclusion

In this work, we developed a theory that establishes a one to one correspondence between the event based and state based models of concurrent computations. We laid down the necessary and sufficient conditions on partial orders that can be ised to model the two categories, namely synchronous and asynchronous, of concurrent computations. Our theory can be used to directly convert any algorithm for analysis of event based models to state based models.

Appendix

Lemma 1: Let $(E, \rightarrow, \pi)$ and $(S, <, \pi)$ be event and state based models of a computation. There is $1-1$ correspondence between consistent global states of $(E, \rightarrow, \pi)$ and consistent global states of $(S, <, \pi)$.

Proof. Let G be any consistent global state of $(E, \rightarrow, \pi)$. We will show how to construct the corresponding consistent global state T of $(S, <, \pi)$. Suppose that G contains at least one event from P_i. Then, let (i, k) be the largest event from process P_i. In this case, we add $[i, k]$ to T. If G does not contain any event from P_i, then we add $[i, 0]$ to T. Clearly, T has exactly n states, one from each process. We show that the global state T is also consistent. If not, suppose $[i, s]$ and $[j, t]$ be two states in T such that $[i, s] < [j, t]$. This implies that $(i, s+1) \rightarrow (j, t)$, under the event based model, contradicting that G is consistent because G contains (j, t) but does not contain $(i, s+1)$. It is also easy to verify that the mapping from the set of consistent global states is $1-1$.

Conversely, given a consistent global state T in the state based model, we construct a consistent global state in event based model in $1-1$ manner as follows. For all states $[i, k] \in T$ we include all events (i, k') such that $k' \leq k$. Note that when k equals 0, no events from P_i are included. It can again be easily verified that whenever T is a consistent global state, G is a consistent prefix. $\square$

Lemma 2: $\hat{S}$ is a poset under the "$<$" relation.

Proof. We show that the relation "$<$" on $\hat{S}$ is transitive and asymmetric, and thus irreflexive.

Claim (i) The relation "$<$" is asymmetric.
Proof: Let $[i, r], [j, s] \in \hat{S}$ such that $[i, r] < [j, s]$. Clearly, $[j, s] \not< [i, r]$ if $i = j$; otherwise we would get $s \to r$ in E. Assume $i \neq j$ and $[j, s] < [i, r]$. Then by definition, we have $(i, r + 1) \to (j, s) \to (j, s + 1) \to (i, r)$ in E, which is impossible as it violates the asymmetry of $\to$ in E.

Claim (ii) The relation "$<$" is transitive.
Proof: Let $[i, r], [j, s], [k, t] \in \hat{S}$ such that $[i, r] < [j, s]$ and $[j, s] < [k, t]$. Assume $i \neq j$ and $k \neq j$. Then we have $(i, r + 1) \to (j, s) \to (j, s + 1) \to (k, t)$ and hence $(i, r + 1) \to (k, t)$ in E, which implies that $[i, r] < [k, t]$ whether $i = k$ or $i \neq k$. The cases for $i = j$ or $j = k$ can be proved similarly.

Hence $\hat{S}$ forms a poset under the "$<$" relation. □

Lemma 3: If $\hat{S}$ is constructed from a event based asynchronous distributed computation using ES transformation, then:

(α_1) $\forall i, j$, where $1 \leq i, j \leq n$: $[i, 0] \mid\mid [j, 0]$. $[i, 0]$ represents the initial state of the chain S_i in state based model.

(α_2) $\forall i, j$, where $1 \leq i, j \leq n$: $[i, n_i] \mid\mid [j, n_j]$. $[i, n_i]$ represents the last element of S_i.

(α_3) $\forall i, j, k$, where $1 \leq i, j, k \leq n$: $[i, s] < [j, t] \wedge [j, t-1] < [k, u]$, for $i \neq j \wedge j \neq k$, $\Rightarrow [i, s] < [k, u]$.

Proof. (α_1) follows immediately from the construction of $\hat{S}$ because there is no state $[i, s]$ such that $[i, s] < [i, 0]$ for any i. That is, on any state chain S_i there does not exist a state that is a precursor to the initial state of S_i. Hence, all the initial states must be concurrent.

Similarly (α_2) follows when applied to the last states of S_i in a similar manner - as there is no state on any state chain S_i that is a successor of the final state of S_i.

(α_3): $[i, s] < [j, t] \wedge [j, t-1] < [k, u]$. Using the construction rules, we can infer that $(i, s+1) \rightarrow (j, t) \wedge (j, t) \rightarrow (k, u)$ in E. Which by transitivity means $(i, s+1) \rightarrow (k, u)$. Hence, $[i, s] < [k, u]$ in S.

$\square$

Theorem 3: Let $(S, <, \pi)$ be any state based model of concurrent computation that satisfies (α_1), (α_2) and (α_3). Then, $(S, <)$ is width-extensible.

Proof. We show that any antichain $A \subset S$ can be extended to a width antichain. It is sufficient to show that when $|A| < n$, there exists an antichain $A \subset B$ such that $|B| = |A| + 1$. Consider any process P_i that does not contribute a state to A. We will show that there exists a state in S_i that is concurrent with all states in A. Let s and s' be two distinct states in A.

We first claim that for any state s and any process P_i, there exists a nonempty sequence of consecutive states called the " *interval* concurrent to s on P_i" and denoted by $I_i(s)$ such that:

1. $I_i(s) \subseteq S_i$ — i.e., the interval consists of only states from process P_i, and
2. $\forall t \in I_i(s) : t || s$ — i.e., all states in the interval are concurrent with s.

For a state $v \in S_i$, let $index(v)$ denote the index of state v on S_i. Thus $0 \leq index(v) \leq n_i$. Define $I_i(s).lo = \min\{v \mid v \in S_i \ \wedge \ v \not< s\}$. This is well-defined since $[i, n_i] \not< s$ due to (α_2). Similarly, on account of (α_1), we can define $I_i(s).hi = \max\{v \mid v \in S_i \ \wedge \ s \not< v\}$.

We show that $I_i(s).lo \leq I_i(s).hi$ by the following case analysis.

Case 1: There exists $v : I_i(s).hi < v < I_i(s).lo$.

Since $v < I_i(s).lo$ implies $v < s$ and $I_i(s).hi < v$ implies $s < v$, we get a contradiction ($v < s < v$).

Case 2: $index(I_i(s).hi) + 1 = index(I_i(s).lo$.

Let $I_i(s).lo$ be r^{th} state on S_i, i.e., $I_i(s).lo = [i, r]$. Then, $I_i(s).hi = [i, r-1]$. Let s correspond to state $[j, t]$. From the definition of $I_i(s).lo$, $[i, r-1] < [j, t]$. From the definition of $I_i(s).hi$, $[j, t] < [i, r]$. We now have, $[j, t] < [i, r]$ and $[i, r-1] < [j, t]$. From (α_3), we get $[j, t] < [j, t]$ which contradicts irreflexivity of $<$.

From the above discussion it follows that $I_i(s).lo \leq I_i(s).hi$. Furthermore, for any state t such that $I_i(s).lo \leq t \leq I_i(s).hi$, $t \not< s$ and $s \not< t$ holds.

From the above claim, we know that $I_i(s)$ and $I_i(s')$ are both non-empty. We show that $I_i(s) \cap I_i(s') \neq \emptyset$. If not, without loss of generality assume that $I_i(s).hi < I_i(s').lo$.

Case 1: $index(I_i(s).hi) + 1 = index(I_i(s').lo)$.

Let $I_i(s).hi$ be r^{th} state on S_i, i.e., $I_i(s).hi = [i, r]$. Then, $I_i(s').lo = [i, r + 1]$. Suppose that $s = [j, u]$ and $s' = [k, v]$. From the definition of $I_i(s).hi$ we get that $[j, u] < [i, r + 1]$. From the definition of $I_i(s').lo$ we get that $[i, r] < [k, v]$. Hence, from (α_3), we get that $[j, u] < [k, v]$ contradicting that s and s' are concurrent.

Case 2: There exists $v : I_i(s).hi < v < I_i(s').lo$.

This implies that $s < v$ (because $I_i(s).hi$ precedes v) and $v < s'$ (because v precedes $I_i(s').lo$). Thus $s < s'$, a contradiction with (A) being an antichain. Therefore, $I_i(s) \cap I_i(s') \neq \emptyset$.

Because any interval $I_i(s)$ is a total order, it follows that:

$$\bigcap_{s \in A} I_i(s) \neq \emptyset$$

We now choose any state in $\bigcap_{s \in A} I_i(s)$ to extend A. □

Bibliography

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

[2] Stephen D Brookes, Charles AR Hoare, and Andrew W Roscoe. A theory of communicating sequential processes. *Journal of the ACM (JACM)*, 31(3):560–599, 1984.

[3] K. M. Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Trans. Comput. Syst.*, 3(1):63–75, February 1985.

[4] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, UK, 1990.

[5] R. P. Dilworth. A decomposition theorem for partially ordered sets. *Ann. Math. 51*, pages 161–166, 1950.

[6] V. K. Garg and B. Waldecker. Detection of weak unstable predicates in distributed programs. *IEEE Trans. on Parallel and Distributed Systems*, 5(3):299–307, March 1994.

[7] KM Koh. On the lattice of maximum-sized antichains of a finite poset. *Algebra Universalis*, 17(1):73–86, 1983.

[8] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. of the ACM*, 21(7):558–565, July 1978.

[9] Robin Milner. *A calculus of communicating systems*. Springer-Verlag New York, Inc., 1982.

[10] R. H. B. Netzer and J. Xu. Necessary and sufficent conditions for consistent global snapshots. *IEEE Trans. on Parallel and Distributed Systems*, 6(2):165–169, February 1995.