Necessary and Sufficient Conditions on Partial Orders for Modeling Concurrent Computations

Himanshu Chauhan and Vijay Garg
Parallel & Distributed Systems Lab, ECE, UT Austin
Outline

1. Introduction
   - Partial orders
   - Event based models

2. Event-State Duality
   - Conversion between models
   - Width-extensibility
   - Interleaving-consistency

3. Applications & Conclusion
Modeling Computations

- Study/identification of key characteristics

- Verification
  - Predictive Analysis
Run program and observe trace
Predictive Analysis

1. Run program and observe trace
2. Model trace as partially ordered set
Predictive Analysis

1. Run program and observe trace
2. Model trace as partially ordered set
3. Explore all consistent cuts
Predictive Analysis

1. Run program and observe trace
2. Model trace as partially ordered set
3. Explore all consistent cuts
4. Check predicate in each consistent cut
Predictive Analysis

1. Run program and observe trace
2. Model trace as partially ordered set
3. Explore all consistent cuts
4. Check predicate in each consistent cut
$P = (X, \prec)$ is a partially ordered set (poset)

- $X$: a set of elements
- $\prec$: irreflexive, anti-symmetric, and transitive binary relation
Event Based Model

- ‘Happened-before’ (→) relation [Lamport 78]

<table>
<thead>
<tr>
<th>Proc. 1</th>
<th>Proc. 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: local event</td>
<td>e: local event</td>
</tr>
<tr>
<td>b: mutex.lock</td>
<td>f: mutex.lock</td>
</tr>
<tr>
<td>c: mutex.unlock</td>
<td>g: mutex.unlock</td>
</tr>
<tr>
<td>d: local event</td>
<td>h: local event</td>
</tr>
</tbody>
</table>

![Diagram showing the 'Happened-before' relation between events on two processes, P1 and P2.](image)
Event Based Model

Distributed System

<table>
<thead>
<tr>
<th>Proc. 1</th>
<th>Proc. 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: local event</td>
<td>e: local event</td>
</tr>
<tr>
<td>b: send msg</td>
<td>f: receive msg</td>
</tr>
<tr>
<td>c: local event</td>
<td>g: local event</td>
</tr>
</tbody>
</table>

![Diagram](image)
Incomparable Elements

- \( P = (X, \prec) \) is a poset
- \( x, y \in X \) are *incomparable* in \( P \) if \( x \not\prec y \) and \( x \not\succ y \).
  - Not transitive

\[
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
  g & h & i \\
\end{array}
\]

\[a \parallel e \checkmark \quad b \parallel f \checkmark \quad b \parallel i \ X.\]
**Antichain of Poset** $P = (X, \prec)$

- **Antichain**: $Y \subseteq X$ such that every pair of elements in $Y$ is concurrent.

Every element by itself is an antichain (of size 1).
**Width Antichain**: Antichain of maximum size.

\{b, e\} is an antichain, but not a width antichain.

\{b, e, h\} is a width antichain.
Let $E$ be the set of events of a computation, and $P = (E, \rightarrow)$ be its partial order model. Then $G \subseteq E$ is a consistent cut (of the computation) if $\forall e, f \in E : (f \in G) \land (e \rightarrow f) \Rightarrow e \in G$.
Notion of States

- States: capture values of local variables
  - have duration
  - events are instantaneous

Many predicates are easier to detect in states

- mutex violation: \( P_1.\text{inCS} = 1 \land P_2.\text{inCS} = 1 \)

- deadlock: \( P_1.\text{waitingFor} = P_2 \land P_1.\text{waitingFor} = P_1 \)
Notion of States

- States: capture values of local variables
  - have duration
  - events are instantaneous

- Events cause transition between states

\[ \text{Pre}(a) \quad a \quad \text{Post}(a) \]
Notion of States

- States: capture values of local variables
  - have duration
  - events are instantaneous

- Events cause transition between states

- Many predicates are easier to detect in states
  - mutex violation ($P_1.\text{inCS} = 1 \land P_2.\text{inCS} = 1$)
  - deadlock ($P_1.\text{waitingFor} = P_2 \land P_1.\text{waitingFor} = P_1$)
Event Model Properties

- Extensively studied/used
  - Any poset can be a model for some computation.
Event Model Properties

- Extensively studied/used
  - Any poset can be a model for some computation.
  - Set of all consistent cuts is a distributive lattice
Extensively studied/used

- Any poset can be a model for some computation.
- Set of all consistent cuts is a distributive lattice

![Poset model](image)

![All consistent cuts of model](image)

**Distributive Lattice**
Main idea: Define key properties of the class of posets that model states of concurrent computations.
1. Introduction
   - Partial orders
   - Event based models

2. Event-State Duality
   - Conversion between models
   - Width-extensibility
   - Interleaving-consistency

3. Applications & Conclusion
Events and States

- Events are instantaneous
- States have duration
- Events cause transition between states

\[ \text{Pre}(a) \quad a \quad \text{Post}(a) \]
State based Models

For each event $e_i \in P_i$, $\exists$ local states $Pre(e_i)$ and $Post(e_i)$:

$Pre(e_i) \prec$ (existed before) $Post(e_i)$. 

Event-State Duality
For each event $e_i \in P_i$, $\exists$ local states $Pre(e_i)$ and $Post(e_i)$:
$Pre(e_i) \prec$ (existed before) $Post(e_i)$.

- Process $P_i$ with $n_i$ events will have $n_i + 1$ local states.
Generating State Models

- From Event based Model

\[ \text{Pre}(a) \quad a \quad \text{Post}(a) \]

- If \( a \rightarrow b \) then \( \text{Pre}(a) \prec \text{Post}(b) \)

\[
\begin{array}{ccc}
(1, 1) & (1, 2) & (1, 3) \\
\downarrow & \downarrow & \downarrow \\
(2, 1) & (2, 2) & (2, 3) \\
\end{array}
\quad
\begin{array}{cccc}
[1, 0] & [1, 1] & [1, 2] & [1, 3] \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\end{array}
\]
Transitivity

- $a \rightarrow b \land b \rightarrow c \Rightarrow a \rightarrow c$

- $\text{Pre}(a) \prec \text{Post}(b) \land \text{Pre}(b) \prec \text{Post}(c) \Rightarrow \text{Pre}(a) \prec \text{Post}(c)$
Lemma 1: The mapping between consistent cuts of event based model and state based model of a computation is one-to-one.

Analysis/detection algorithms under one modeling scheme can be used for other, and vice-versa.
Mapping Consistent Cuts

Main Results
Properties of State Models

Is any poset a valid state model?
For a poset in state based model:

- All initial local states of processes are concurrent.
- All final local states of processes are concurrent.
- Inter process state-transitivity of ‘existed-before’ is satisfied.
Definition: A poset \((X, \preceq)\) is width-extensible if and only if for every antichain \(A \subseteq X\), there exists a width-antichain \(W\) containing \(A\).

[Diagram showing a width-extensible and a non-width-extensible poset]
Width-Extensible Poset

Main Results
Width-Extensible Poset

Main Results
Definition: A poset \((X, \prec)\) is width-extensible if and only if for every antichain \(A \subseteq X\), there exists a width-antichain \(W\) containing \(A\).
Theorem 1: A poset can model states of a concurrent computation if and only if it is width-extensible.
Every antichain must be contained in a width-antichain.

Check for all antichains

Expensive!
Theorem 2: A poset is width-extensible if and only if for every antichain $A$ of size at most two, there exists a width-antichain $W$ containing $A$.

Suffices to verify (poset) for antichains of size up to two.
Asynchronous Computations

Main Results

Not a valid asynchronous model
Asynchronous Computations

Main Results

Not a valid asynchronous model
Interleaving-Consistent Poset: A poset \((X, \preceq)\) 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 \(|W \cap W'| = |W| - 1\).

Such structures can not occur.
Main Results

- **Theorem 3**: A poset can model states of an asynchronous concurrent computation if and only if it is width-extensible as well as interleaving-consistent.
1. Introduction
   - Partial orders
   - Event based models

2. Event-State Duality
   - Conversion between models
   - Width-extensibility
   - Interleaving-consistency

3. Applications & Conclusion
   - Checkpointing theory
   - Predicate detection
Checkpointing

- Zig-zag paths and cycles [Netzer and Zu’95]
- R-graph [Wang’97]
- Both techniques effectively check for state models being correct (width-extensible).
Predicate Detection

- **Width-predicate**: predicate that requires states from every involved process.

- **Efficient detection algorithms** using the bijection of state and event models.
Conclusion

- Study models on processes’ *states*.

- Algorithms for generating state models from event models, and vice-versa.

- Characteristics of posets that model states.