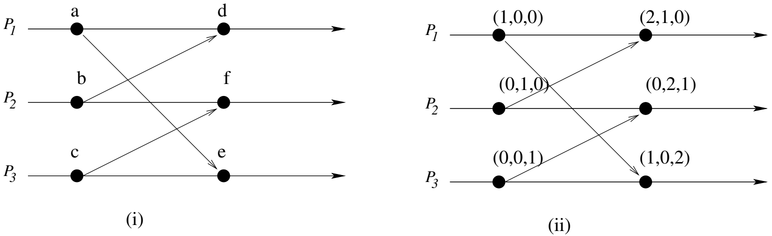
Maximal Antichain Lattice Algorithms for Distributed Computations

Vijay K. Garg

Parallel and Distributed Systems Lab,
Department of Electrical and Computer Engineering,
The University of Texas at Austin,
Austin, TX 78712
http://www.ece.utexas.edu/˜garg

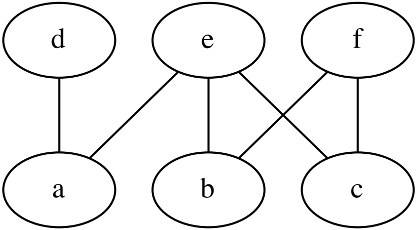
Model of a Distributed Computation: Poset

distributed computation = poset (partially ordered set)
$(E, \rightarrow)$ where
$E =$ is the set of events, and
$\rightarrow$ is (happened-before) relation.



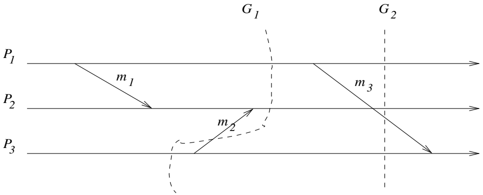
Events can be timestamped in an online fashion using Vector Clocks such that $e \rightarrow f \equiv V(e) < V(f)$.

Computing Meet and Join



- Meet of a subset of events
  meet of {d, e}
  meet of {a,b}
  meet of {e,f}
- Join of a subset of events
- Lattice: a poset in which all finite subsets have meets and joins.

Consistent Cuts of a Distributed System



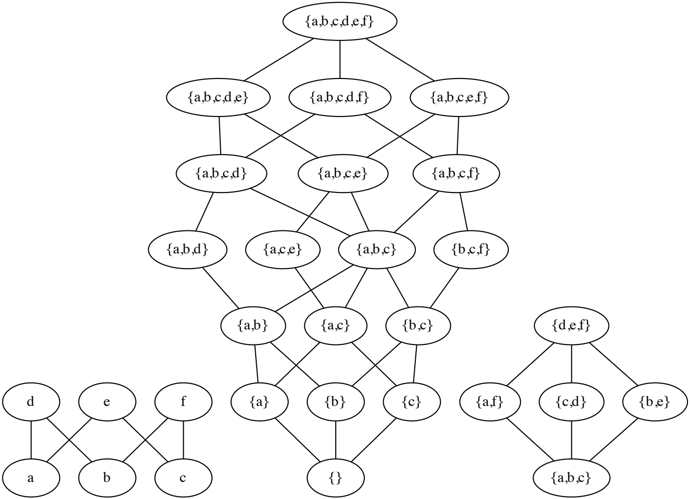
Consistent cut = set of events executed so far

A subset G of E is a consistent cut (consistent global state) if

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

Same as the *order* ideal of the partial order $(E, \to)$.

Motivation: Detecting Global Conditions in Distributed Systems

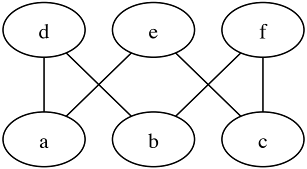


- Traversing a significantly smaller lattice of maximal antichains rather than consistent cuts for certain predicates

Outline of the Talk

- Motivation
- Incremental Lattice Algorithms
- Lattice Enumeration Algorithms
- Applications to Global Predicate Detection
- Conclusions and Future Work

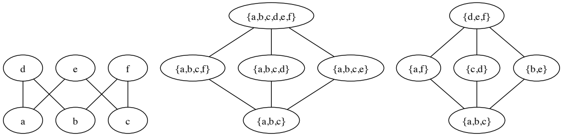
Ideals and Antichains



Poset $P = (X, \leq)$

- Ideal: $Q \subseteq X$ is an ideal $\equiv$ if f is in Q and $e \leq f$, then e is also in Q.
- Antichain: $Y \subseteq X$ is called an *antichain*, if every distinct pair of elements from Y are incomparable.

Maximal Antichains



- Maximal Antichain: An antichain A is maximal if every element not in A is comparable to some element in A.
  $\{d, e\}$ is an antichain but not a maximal antichain
  $\{d, e, f\}$ is a maximal antichain
- Maximal Ideal: An ideal Q is maximal antichain ideal if its maximal elements forms a maximal antichain.
  $\{a, b, c, d\}$ and $\{a, b, c, e\}$ are maximal ideals
  $\{a, b, c, d, e\}$ is not a maximal ideal

Important Lattices Associated with a Poset

Lattice of	Interpretation in DC	References
Ideals	consistent global states	[Mattern88, CM91,..]
Normal Cuts	Smallest lattice containing P	[Garg OPODIS12]
Maximal Ideals	State for maximal antichain	[JRJ94, this paper]

Table: Summary of Lattices for Distributed Computations modeled as a poset P

Related Work: Incremental Algorithms

Elements of the computation arrive in a sorted order

Input: poset P, its lattice of maximal antichains L, element x

Output: $L' :=$ lattice of maximal antichains of $P \cup \{x\}$

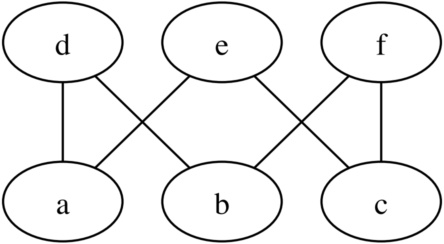
Incremental Algorithms	Time Complexity	Space Complexity
[Jourdan, Rampon, Jard 94]	$O(w^3 m)$	$O(mn \log n)$
[Nourine and Raynaud 99, 02]	$O(mn)$	$O(mn \log n)$
Algorithm ILMA [this paper]	$O(wm \log m)$	$O(mw \log n)$
Algorithm OLMA [this paper]	$O(m_x w^2 \log w_L))$	$O(w_L w \log n)$

Symbol	Definition	Symbol	Definition
n	size of the poset P	m	size of the lattice L
w	width of the poset P	m_x	# (strict ideals $\geq D(x)$)
w_L	width of the lattice L		

Strict Ideals

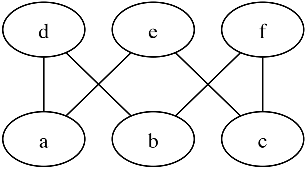
$D(A)$ = the set of elements strictly smaller than A

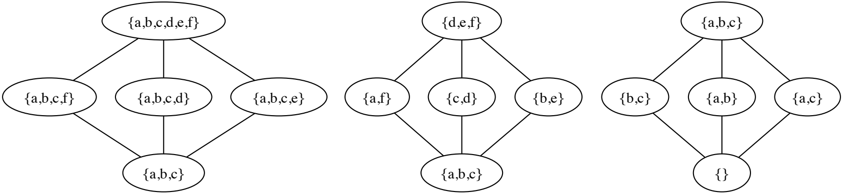
Strict Ideal: A set Y is a strict ideal if there exists an antichain A such that $D(A) = Y$.



Example: $\{a, b\}$ is a strict ideal. $\{a, b, c, d\}$ is not.

Equivalence of maximal ideals and strict ideals





Lattices of maximal ideals, maximal antichains, and strict ideals are isomorphic.

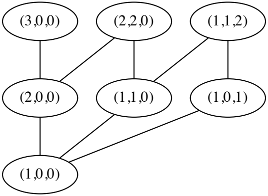
Mapping from strict ideals to maximal antichains: S is mapped to the minimal elements of the complement of S.

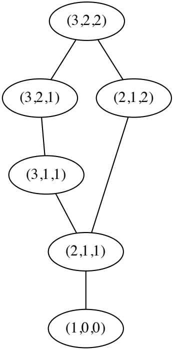
Input: P: a finite poset as a list of vector clocks
L: lattice of maximal antichains of vector clocks
x: new element
Output: $L' :=$ Lattice of maximal antichains of $P \cup \{x\}$ initially L

// Step 1: Compute the set $D(x)$
Let V be the vector clock for x on process P_i;
$S := V$; $S[i] := S[i] - 1$;
// Step 2:
if $S \notin L$ then
    $L' := L' \cup \{S\}$;
    forall vectors $W \in L$:
        if $max(W, S) \notin L$ then $L' := L' \cup max(W, S)$;

Example

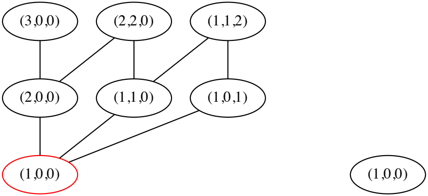
Poset and its lattice of maximal antichains





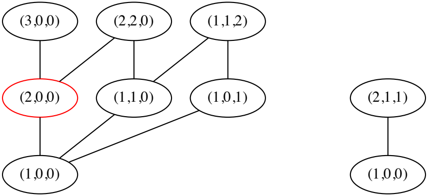
Example

Step 1: $D[x] = (1, 0, 0)$
$D(x) = (0, 0, 0)$, strict ideals added: $(0, 0, 0)$
Set of Maximal Antichains $= \{(1, 0, 0)\}$



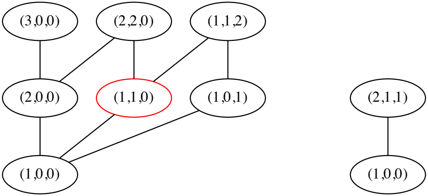
Example

Step 2: $D[x] = (2, 0, 0)$
$D(x) = (1, 0, 0)$, strict ideals added: $(1, 0, 0)$
Set of Maximal Antichains $= \{(1, 0, 0), (2, 1, 1)\}$



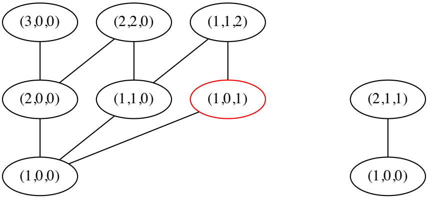
Example

Step 3: $D[x] = (1, 1, 0)$
$D(x) = (1, 0, 0)$, strict ideals added: ϕ
Set of Maximal Antichains $= \{(1, 0, 0), (2, 1, 1)\}$



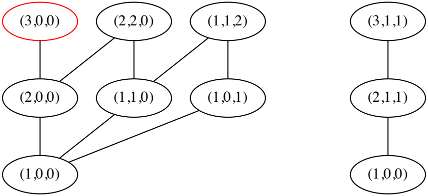
Example

Step 4: $D[x] = (1, 0, 1)$
$D(x) = (1, 0, 0)$, strict ideals added: ϕ



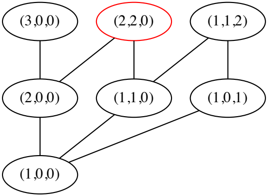
Example

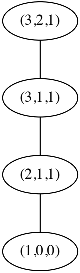
Step 5: $D[x] = (3, 0, 0)$
$D(x) = (2, 0, 0)$, strict ideals added: $(2, 0, 0)$
Maximal antichain added: $(3, 1, 1)$



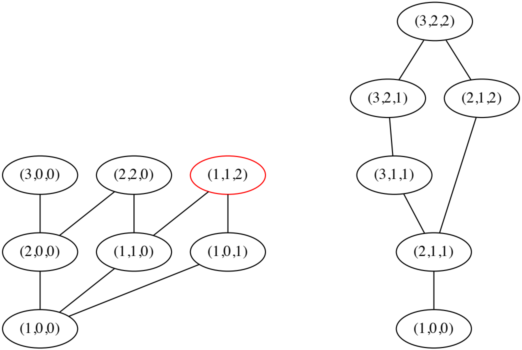
Example

Step 6: $D[x] = (2, 2, 0)$
$D(x) = (2, 1, 0)$, strict ideals added: $(2, 1, 0)$
Maximal antichain added: $(3, 2, 1)$





Step 6: $D[x] = (1, 1, 2)$
$D(x) = (1, 1, 1)$, strict ideals added: $(1, 1, 1)$, $(2, 1, 1)$
Maximal antichains added: $\{(2, 1, 2), (3, 2, 2)\}$



Analysis of the Incremental Algorithm 1

1. Simple Algorithm
2. To check if $max(S, W) \in L$, maintain L as a binary search tree
3. Requires storage of the the entire lattice (exponential in size of the poset in the worst case)

Space Efficient Incremental Algorithm

Input: a finite poset P, x maximal element in $P' = P \cup \{x\}$

Output: enumerate M such that $L_{MA}(P') = L_{MA}(P) \cup M$

$S :=$ the vector clock for x on process P_i;
$S[i] := S[i] - 1$;
if S is not a strict ideal of P then
// $BFS(S)$: Do Breadth-First-Search traversal of M
    $\mathcal{T}$:= set of vectors initially $\{S\}$;
    while $\mathcal{T}$ is nonempty do
        $H :=$ delete the smallest vector from $\mathcal{T}$;
        enumerate H;
        foreach process k with next event e do
            explore H using e;
        endfor;
    endwhile;
endif;

Outline of the Talk

- Motivation
- Incremental Lattice Algorithms
- Lattice Enumeration Algorithms
- Applications to Global Predicate Detection
- Conclusions and Future Work

- Global predicate detection requires enumeration not construction of the lattice
- Lattice of maximal antichains may be exponential in the number of processes

Related Work: Enumeration Algorithms

Input: a nonempty finite poset P
Output: enumerate all elements of $L :=$ DM-completion of P

Algorithm	Time	Space
[Jourdan, Rampon, Jard 94]	$O((n+w^2)wm)$	$O(mn \log n)$
[Nourine and Raynaud 99, 02]	$O(mn^2)$	$O(mn \log n)$
Algorithm ILMA [this paper]	$O(nwm \log m)$	$O(mw \log n)$
BFS-MA [this paper]	$O(mw^2 \log m)$	$O(w_L w \log n)$
DFS-MA [this paper]	$O(mw^4)$	$O(nw \log n)$
Lexical by [Ganter84]	$O(mn^3)$	$O(n \log n)$

The parameters are:
n: size of the poset P,
m: size of the lattice L of normal cuts of P,
w: width of the poset P,
w_L: width of the lattice L.

- *closure*(Y) = smallest maximal antichain ideal that contains Y. The operator *closure*(Y) is monotone, extensive and idempotent.
- Idea: View the lattice of maximal antichains as a directed graph and enumerate the nodes of the graph using the closure operator.
- Difficulty: Usual DFS on graph cannot be employed as the graph cannot be stored. Cannot mark which nodes have been visited before.

Depth First Search Enumeration of Maximal Antichain Ideals

Input: a finite poset P, starting state G
Output: DFS Enumeration of Maximal Antichain Ideals of P
(1) output(G);
(2) **foreach** event e enabled in G do
(3) $K := closure(G \cup \{e\})$;
(4) if (K covers G) and (not visited before) then
(5) DFS(K);

How to avoid revisiting cuts?

Visit a state only from the maximum predecessor.

(4) if K does not cover G then go to the next event;
(5) $M :=$ get-Max-predecessor(K) ;
(6) if $M = G$ then
(7) DFS-MaximalIdeals(K);

- To check whether K covers G: use the efficient characterization provided by [Reuter 91].
- To choose the maximum predecessor Expand the nodes of the dual poset and choose the biggest predecessor

Outline of the Talk

- Motivation
- Incremental Lattice Algorithms
- Lattice Enumeration Algorithms
- Applications to Global Predicate Detection
- Conclusions and Future Work

Application to Global Predicate Detection

Definition (Antichain-Consistent Predicate)

A global predicate is an *antichain-consistent predicate* if

1. its evaluation depends only on maximal events of a consistent global state and
2. if it is true on a subset of processes, then presence of additional processes does not falsify the predicate.

Examples of antichain-consistent predicates

- Violation of mutual exclusion: "there is more than one process in the critical section."
- Violation of resource usage: "there are more than k concurrent activation of certain service,"
- Global Control Point: The predicate, B, "Process P_1 is at line 35 and P_2 is at line 23 concurrently,"

Theorem

There exists a consistent global state that satisfies an antichain-consistent predicate B iff there exists a maximal ideal that satisfies B.

Conclusions and Future Work

- An Incremental Algorithm to generate the lattice of maximal antichains
- BFS and DFS enumeration of maximal antichains
- Applications to global predicate detection

Question: Is there a space-efficient algorithm with time complexity $O(mw \log n)$?