Meet- and join-closure of CTL operators

Sujatha Kashyap and Vijay K. Garg

Parallel and Distributed Systems Lab
ECE Department
University of Texas at Austin
Austin, TX 78712, USA

{kashyap, garg}@ece.utexas.edu

January 22, 2008

Abstract

Results from lattice theory have successfully been applied by several researchers, *e.g.* [1, 7, 9, 10], to reduce the complexity of verification of distributed computations. The set of all reachable states of a distributed computation forms a lattice under a certain partial order relation. A property is said to exhibit meet-closure if the set of reachable states satisfying it is closed under the lattice meet operation, and join-closure if the set is closed under the lattice join operation. Techniques that apply lattice-theoretic concepts to the verification of distributed computations have largely leveraged the principles of meet- and join-closure of the properties to be verified. It has been shown [7, 10] that exploiting such closure properties allows for the development of verification algorithms that have time and space complexity that is polynomial in the number of events in the computation. Since the number of reachable states in a computation is usually exponential in the number of events, these algorithms significantly alleviate the problem of state space explosion in formal verification.

The temporal logic CTL [2] is popularly used to specify the properties to be verified on a distributed computation. It is thus desirable to study whether various CTL operators preserve the properties of meet- and join-closure. Some CTL operators were previously studied in [10]. In this paper, we extend on the work in [10], and identify additional CTL operators that preserve these closure properties. We also revisit some of the operators which were shown *not* to preserve meet-closure in [10], and identify some conditions under which they *do* preserve meet-closure.

1 Introduction

An execution of a distributed program is called a (distributed) computation. In a seminal paper, Lamport [4] noted that, in the absence of a global clock, it is impossible to derive the exact sequential order in which events from different processes occur. He argued that the events that occur in a distributed computation are best viewed as a partial order, in which two events are related only if one of them causally precedes the other. For example, the send of a message causally precedes its receive. A reachable (global) state of the computation is any state that can be reached by executing events in a manner consistent with the partial order relation. A state is thus the culmination of the execution of some events. Consequently, we can view a state as a set of events.

In [5], it was shown that the set of reachable states of a computation forms a lattice. The application of lattice theoretic concepts to the verification of distributed computations was pioneered in [1], where the class of *meet-closed* predicates was introduced. It was shown that, under certain conditions, a polynomial time algorithm exists for solving reachability for meet-closed predicates. In [7], the class of *regular* predicates was introduced. A predicate is regular iff it is both meet- and join-closed. A polynomial-time algorithm was also given in [7] for extracting *all* the states of the computation that satisfy a given regular predicate.

Since meet-closed and regular predicates admit efficient verification algorithms, it is desirable to determine which commonly-verified properties are meet-closed or regular. The Computation Tree Logic, or CTL [2], is widely used to specify properties to be verified on computations. In [10], it was shown that several CTL operators, such as EF, EG

and AG, preserve both meet- and join-closure. Thus, these temporal operators could be used to construct properties that could then be efficiently verified using the algorithms developed in [1, 7, 9, 10].

In this paper, we consider the "until" and "release" operators from CTL, and show the circumstances under which these operators preserve meet- and join-closure. In [10], it was shown that the AF operator does not preserve meet-closure. In this paper, we show that AF preserves a stronger property, called *biregularity*.

The paper is organized as follows. In Section 2, we present some background material and introduce notations. Section 3 presents our results on the closure properties preserved by various CTL operators. Finally, some concluding remarks are presented in Section 4.

2 Background and Notation

A distributed computation can be viewed as a partially ordered set of events [4, 11, 6], where the partial order relation is an indication of causality. Lamport called this the "happened-before" relation [4], and denoted it by $\rightarrow$. For instance, if an event e denotes the sending of a message, and f the corresponding receive event, then $e \rightarrow f$.

Let E be the set of events that occur during an execution of a distributed program. The poset $(E, \rightarrow)$ is called a *trace*, and is denoted by σ. Any linearization of $(E, \rightarrow)$ is a valid sequence of execution for the events in E. Conversely, every valid execution sequence of these events gives us a linearization of the poset $E(, \rightarrow)$.

A subset $G \subseteq E$ of a poset $(E, \rightarrow)$ is called a *down-set* if, whenever $f \in G$, $e \in E$ and $e \rightarrow f$, we have $e \in G$. There is a 1-1 correspondence between the program states that can be encountered in any valid execution sequence of σ, and down-sets of $(E, \rightarrow)$. Executing the events in G in an order that is consistent with $\rightarrow$ leads to a valid state of σ. Conversely, every state in σ can be reached by executing the events in some down-set in accordance with the $\rightarrow$ relation. For simplicity of presentation, in this paper we overload the term "down-set" to mean both a subset of events, and a state of σ.

Progress in a computation is measured by the execution of additional events from the current state. For down-sets G and H of a trace $(E, \rightarrow)$, $G \subseteq H$ iff H is reachable from G in the full state space graph.

Let $\mathcal{D}$ be the set of all down-sets of a poset $(E, \rightarrow)$. From lattice theory [3], the poset $(\mathcal{D}, \subseteq)$ is a lattice[1], in which the meet and join operations are given by set intersection and union, respectively. That is, if G and H are down-sets of $(E, \rightarrow)$, so are $(G \cap H)$ and $(G \cup H)$. The lattice of down-sets of a trace σ will be denoted by $\mathcal{L}(\sigma)$. This view of the states of a trace as elements of a lattice was previously explored in [11, 5, 7], among others. Figure 1 illustrates the relationship between the poset representation of a trace and the down-set lattice.

3 Lattice properties of CTL formulae

Let σ be a trace, and p be a formula.

Definition 1. Meet-closed *[1]: p is meet-closed iff:*

$$\forall G, H \in \mathcal{L}(\sigma) : [(G \models p) \wedge (H \models p) \Rightarrow (G \cap H) \models p]$$

Definition 2. Join-closed*: p is join-closed iff:*

$$\forall G, H \in \mathcal{L}(\sigma) : [(G \models p) \wedge (H \models p) \Rightarrow (G \cup H) \models p]$$

Definition 3. Regular *[7]: p is regular iff it is meet- and join- closed.*

Definition 4. Biregular*: p is biregular iff both p and $\neg p$ are regular.*

There are ten CTL operators: EX, AX, EG, AG, EF, AF, EU, AU, ER, and AR. In this paper, we do not consider the next-time operators, EX and AX. The remaining operators can be expressed in terms of EU and EG. Denote the i^{th} state on a path π by π^i.

- $s \models EG(p)$ iff there exists a path π starting from s such that $\forall i \geq 0 : \pi^i \models p$.
- $s \models E[pUq]$ iff there exists a path π starting from s such that for some $j \geq 0 : \pi^j \models q$, and $\forall i < j : \pi^i \models p$.

[1]A lattice is a poset in which every finite subset of elements has a supremum (least upper bound) and infimum (greatest lower bound).

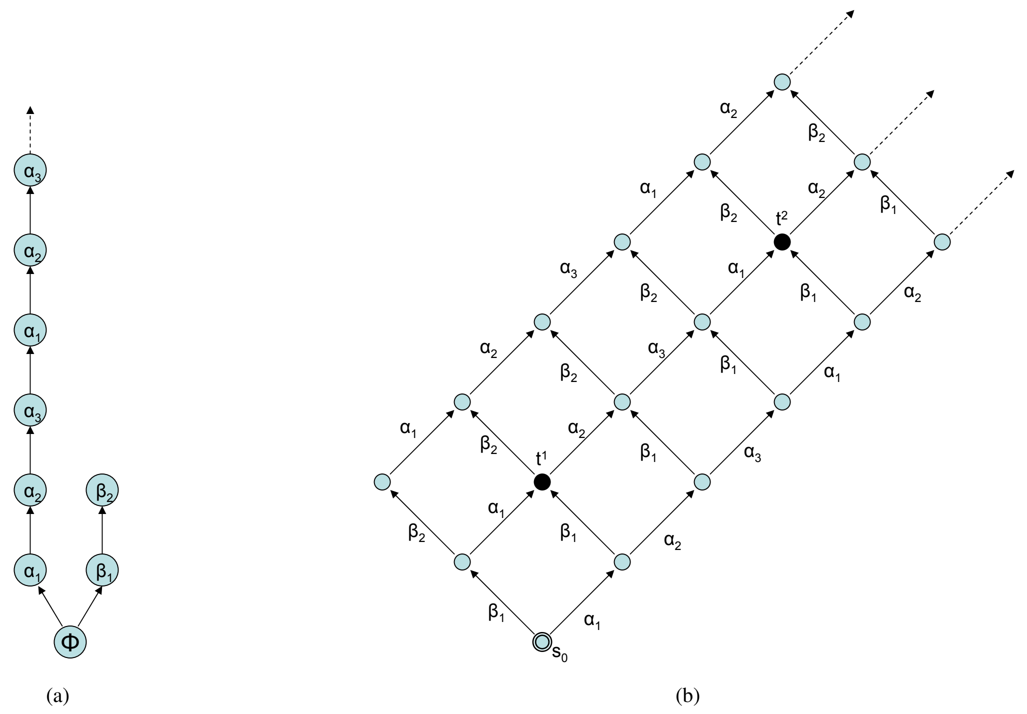

Figure 1: (a) The poset representation of a trace σ. (e) The lattice of down-sets $\mathcal{L}(\sigma)$.

- $EF(p) = E[true\, U\, p]$
- $E[pRq] = E[qU(p \wedge q)] \vee EG(q)$
- $AG(p) = \neg EF(\neg p)$
- $AF(p) = \neg EG(\neg p)$
- $A[pUq] = \neg E[\neg pR\neg q]$
- $A[pRq] = \neg E[\neg pU\neg q]$

In the rest of this paper, we present the closure properties exhibited by various CTL operators. We first explore the strongest closure property, biregularity. We consider concurrent systems, where the system is modeled as a set of processes. Each process P_i has a set of transitions T_i, and a set of *local* variables V_i that can only be changed by transitions in T_i. All the transitions in T_i are pairwise dependent, that is, if $\alpha, \beta \in T_i$, then $(\alpha, \beta) \in D$. A transition in T_i can also change the values of shared (global) variables. A formula ϕ is called a *process-local state formula* iff its truth value is purely determined by the current values of the variables V_i of some process P_i.

Theorem 1. *Process-local state formulae are biregular.*

Proof. Let σ be a trace, and p a process-local state formula defined on the local variables of process P_j. Since no two transitions from P_j are independent, no two transitions from P_j can commute with each other. So, the events from P_j must occur in the same sequence in every path of σ.

Let s be the starting state of σ, and v be a maximal path of $\mathcal{L}(\sigma)$. Let v_j be the restriction of v to events from P_j, *i.e.*, v_j is obtained from v by deleting all events from processes other than P_j. Let G and H be any two down-sets of σ such that $G \models p$ and $H \models p$. Let u and w be any two paths in $\mathcal{L}(\sigma)$, leading, respectively, from s to G and s to H. Then, both u_j and w_j (derived in a similar fashion as v_j from v) are prefixes of v_j. Thus, either u_j is a prefix of w_j, or w_j is a prefix of u_j. WLOG, say u_j is a prefix of w_j.

Now, let u' be any path from s to $(G \cap H)$ in $\mathcal{L}(\sigma)$. Then, $u'_j = u_j$. Since the truth value of p is determined purely by events from process P_j, and $G \models p$, we have $(G \cap H) \models p$. Similarly, let w' be some path from s to $(G \cup H)$ in $\mathcal{L}(\sigma)$. Then, $w'_j = w_j$, hence $(G \cup H) \models p$.

Finally, the negation of a process-local state formula is also a process-local state formula. Thus, $\neg p$ is also regular, which implies that p is biregular. $\square$

In [10], it was shown that if p is regular, then so are $EF(p)$ and $AG(p)$. This leads to the following theorem:

Theorem 2. *[10] If p is biregular, then $EF(p)$ and $AG(p)$ are biregular.*

Proof. Since p is (bi)regular, from [10], we know that $EF(p)$ and $AG(p)$ are regular. Now, we need to show that $\neg EF(p)$ and $\neg AG(p)$ are regular.

$$\begin{array}{rrcrr} & G \models \neg EF(p) & \text{and} & H \models \neg EF(p) & \\ \Rightarrow & G \models AG(\neg p) & \text{and} & H \models AG(\neg p) & \{\neg EF(p) = AG(\neg p)\} \\ \Rightarrow & (G \cap H) \models AG(\neg p) & \text{and} & (G \cup H) \models AG(\neg p) & \{\neg p \text{ is regular, so } AG(\neg p) \text{ is regular}\} \\ \Rightarrow & (G \cap H) \models \neg EF(p) & \text{and} & (G \cup H) \models \neg EF(p) & \end{array}$$

Similarly:

$$\begin{array}{rrcrr} & G \models \neg AG(p) & \text{and} & H \models \neg AG(p) & \\ \Rightarrow & G \models EF(\neg p) & \text{and} & H \models EF(\neg p) & \{\neg AG(p) = EF(\neg p)\} \\ \Rightarrow & (G \cap H) \models EF(\neg p) & \text{and} & (G \cup H) \models EF(\neg p) & \{\neg p \text{ is regular, so } EF(\neg p) \text{ is regular}\} \\ \Rightarrow & (G \cap H) \models \neg AG(p) & \text{and} & (G \cup H) \models \neg AG(p) & \end{array}$$

$\square$

In [10], it was also shown that $EG(p)$ is regular when p is regular. In [8], it was shown that $AF(p)$ is join-closed for regular p, but was not meet-closed for regular p. Here, we show that $AF(p)$ is meet-closed when p is biregular.

Lemma 3. *$AF(p)$ is meet-closed for biregular p.*

Proof. Assume, for contradiction, that $G \models AF(p)$ and $H \models AF(p)$, but $(G \cap H) \models \neg AF(p)$.

$$\begin{array}{rrr} & (G \cap H) \models \neg AF(p) & \\ \Rightarrow & (G \cap H) \models EG(\neg p) & \\ \Rightarrow & (G \cap H) \models \neg p & \\ \Rightarrow & (G \models \neg p) \vee (H \models \neg p) & \{\text{Since } \neg p \text{ is biregular}\} \end{array}$$

WLOG, let $G \models \neg p$. Since $(G \cap H) \models EG(\neg p)$, there exists a path π starting from $(G \cap H)$ such that $\forall i : \pi^i \models \neg p$. Then, we can construct the following path ρ, starting from G, as follows:

$$\rho = G \cup \pi^0, G \cup \pi^1, G \cup \pi^2, ...$$

Since $G \models \neg p$, and $\forall i : \pi^i \models \neg p$, by the join-closedness of $\neg p$, every state of ρ satisfies $\neg p$. Thus, ρ is a witness path for $G \models EG(\neg p)$, which implies $G \models \neg AF(p)$, which contradicts our initial assumption that $G \models AF(p)$. $\square$

Theorem 4. *If p is biregular, then $AF(p)$ and $EG(p)$ are biregular.*

Proof. Given p is biregular. Then, from [10], $EG(p)$ is regular. Also, from [8], $AF(p)$ is join-closed. From Lemma 3, $AF(p)$ is meet-closed. Hence, $AF(p)$ is regular. Now, we need to show that $\neg AF(p)$ and $\neg EG(p)$ are regular.

$$\begin{array}{rrcrr} & G \models \neg AF(p) & \text{and} & H \models \neg AF(p) & \\ \Rightarrow & G \models EG(\neg p) & \text{and} & H \models EG(\neg p) & \{\neg AF(p) = EG(\neg p)\} \\ \Rightarrow & (G \cap H) \models EG(\neg p) & \text{and} & (G \cup H) \models EG(\neg p) & \{\neg p \text{ is regular, so } EG(\neg p) \text{ is regular}\} \\ \Rightarrow & (G \cap H) \models \neg AF(p) & \text{and} & (G \cup H) \models \neg AF(p) & \end{array}$$

Similarly:

$$\begin{array}{rrcrr} & G \models \neg EG(p) & \text{and} & H \models \neg EG(p) & \\ \Rightarrow & G \models AF(\neg p) & \text{and} & H \models AF(\neg p) & \{\neg EG(p) = AF(\neg p)\} \\ \Rightarrow & (G \cap H) \models AF(\neg p) & \text{and} & (G \cup H) \models AF(\neg p) & \{\neg p \text{ is regular, so } AF(\neg p) \text{ is regular}\} \\ \Rightarrow & (G \cap H) \models \neg EG(p) & \text{and} & (G \cup H) \models \neg EG(p) & \end{array}$$

□

We now explore which temporal operators preserve the weaker property of regularity.

Theorem 5. *If p and q are regular, then $(p \wedge q)$ is regular.*

Proof.

$$\begin{array}{rrcrr} & G \models (p \wedge q) & \text{and} & H \models (p \wedge q) & \\ \Rightarrow & (G \models p) \wedge (G \models q) & \text{and} & (H \models p) \wedge (H \models q) & \\ \Rightarrow & [(G \cap H) \models p] \wedge [(G \cap H) \models q] & \text{and} & [(G \cup H) \models p] \wedge [(G \cup H) \models q] & \{p \text{ and } q \text{ are regular}\} \\ \Rightarrow & (G \cap H) \models (p \wedge q) & \text{and} & (G \cup H) \models (p \wedge q) & \end{array}$$

□

Theorem 6. *The following temporal formulae are regular, if p and q are regular:*

- $E[qRp]$
- $E[pU(p \wedge q)]$

Proof. We show that $E[qRp]$ is regular, for regular p and q. Let $G, H \in \mathcal{L}(\sigma)$ such that $G \models p$ and $H \models p$. We have:

$$\begin{array}{rrr} & (G \models E[qRp]) \wedge (H \models E[qRp]) & \\ \Rightarrow & (G \models p) \wedge (H \models p) & \{\text{definition of } E[qRp]\} \\ \Rightarrow & (G \cap H) \models p & \{\text{meet-closure of } p\} \\ \text{and} & (G \cup H) \models p & \{\text{join-closure of } p\} \end{array}$$

Recall that $E[qRp] = E[pU(p \wedge q)] \vee EG(p)$.

- **Case 1:** Both G and H satisfy $E[pU(p \wedge q)]$.

  In the lattice $\mathcal{L}(\sigma)$, there exist finite paths π and ρ, starting from G and H respectively, such that $\pi^{end} \models q$ and $\rho^{end} \models q$, where π^{end} and ρ^{end} are the final states on π and ρ, respectively. We can construct a path λ starting from $(G \cap H)$ as follows:

  $$\lambda = G \cap H, G \cap \rho^1, G \cap \rho^2, ..., G \cap \rho^{end}, \\ \pi^1 \cap \rho^{end}, \pi^2 \cap \rho^{end}, ..., \pi^{end} \cap \rho^{end}$$

  From the meet-closure of p and q, it follows that λ is a witness for $E[pU(p \wedge q)]$. Similarly, we can construct ν starting from $(G \cup H)$:

  $$\nu = G \cup H, G \cup \rho^1, G \cup \rho^2, ..., G \cup \rho^{end}, \\ \pi^1 \cup \rho^{end}, \pi^2 \cup \rho^{end}, ..., \pi^{end} \cup \rho^{end}$$

  From the join-closure of p and q, it follows that ν is a witness for $E[pU(p \wedge q)]$.

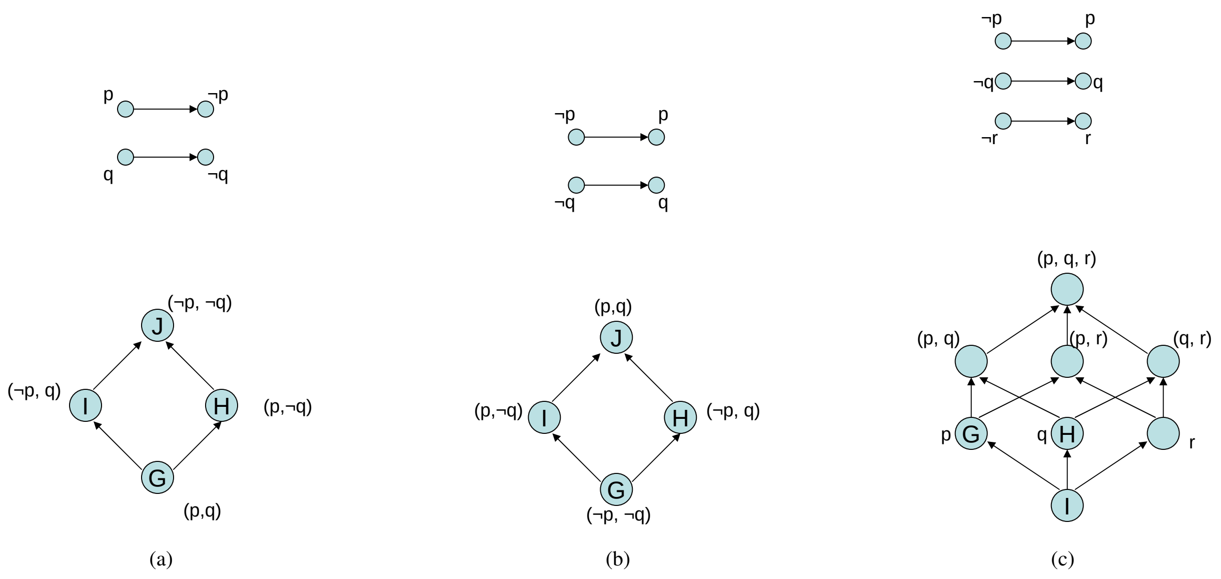

Figure 2: (a) $H \models \neg E[pU(p \wedge q]$, and $I \models \neg E[pU(p \wedge q]$, but $G = (H \cap I) \models E[pU(p \wedge q]$. Also, $H \models \neg E[pRq]$, and $I \models \neg E[pRq]$, but $G = (H \cap I) \models E[pRq]$. (b) H and I satisfy $E[pUq]$ and $A[pUq]$. $G = (H \cap I)$ satisfies neither. (c) G and H satisfy $A[pUq]$, while $I = (G \cap H)$ does not.

- **Case 2:** Either G or H satisfies $EG(p)$.

  WLOG, let $G \models EG(p)$. Let π be a witness path starting from G. We first show that there exists a $k \geq 0$ such that $H \subseteq \pi^k$.

  The witness path π is maximal, *i.e.*, it will eventually contain all the events in E. Since $H \subseteq E$, there exists some state on π that contains all the events in H. Thus, there exists a $k \geq 0$ such that $H \subseteq \pi^k$.

  We use the above property to construct a path λ starting from $(G \cup H)$:

  $$\lambda = G \cap H, \pi^1 \cap H, \pi^2 \cap H, ..., (\pi^k \cap H = H)$$

  Let ρ be the witness path for $E[qRp]$ starting from H. Then, the required witness path for $E[qRp]$ from $(G \cap H)$ is given by $\lambda.\rho$.

  To demonstrate join-closure, we construct the following path ν starting from $(G \cup H)$:

  $$\nu = (G \cup H), (G \cup H) \cup \pi^1, (G \cup H) \cup \pi^2, ...$$

  From the join-closure of p, it follows that ν is a witness path for $(G \cup H) \models EG(p)$.

The proof that $E[pU(p \wedge q)]$ is regular is the same as Case 1 above. □

As $EF(p) = E[true\, U(true \wedge p)]$, and $EG(p) = E[false\, R\, p]$, we have[2]:

Corollary 7. *If p is a regular formula, so are $EF(p)$ and $EG(p)$.*

The operators $E[qRp]$ and $E[pU(p \wedge q)]$ are, however, not biregular. Figure 2(a) depicts provides a counterexample. We now explore operators that are *not* regular.

Theorem 8. *The following temporal operators are* **not** *meet-closed, for regular (even biregular) p and q:*

- $E[pUq]$
- $A[pUq]$

[2] $true$ and $false$ are trivially meet- and join-closed.

- $A[pRq]$

Proof.

- $E[pUq]$, $A[pUq]$: Counter-example in Figure 2(b).
- $A[pRq]$: Counter-example in Figure 2(c).

□

4 Conclusion and Future Work

We have examined CTL operators in addition to those studied in [10]. We showed that the operator ER and a variation of the operator EU preserve regularity. We also showed that the AF operator preserves biregularity. The results here extend the family of formulae that are known to preserve meet- and join-closure.

In future work, we intend to extend our family of verification algorithms ([1, 7, 9, 10]) to include algorithms for detecting formulae constructed using the EU, ER, and AF operators.

References

[1] C. M. Chase and V. K. Garg. Efficient detection of restricted classes of global predicates. In *WDAG '95: Proceedings of the 9th International Workshop on Distributed Algorithms*, pages 303–317, London, UK, 1995. Springer-Verlag.

[2] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs, Workshop*, pages 52–71, London, UK, 1982. Springer-Verlag.

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

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

[5] F. Mattern. Virtual time and global states of distributed systems. In *Proc. of the International Workshop on Distributed Algorithms*, pages 215–226, 1989.

[6] A. W. Mazurkiewicz. Basic notions of trace theory. In *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, School/Workshop*, pages 285–363, London, UK, 1989. Springer-Verlag.

[7] N. Mittal and V. K. Garg. Computation slicing: Techniques and theory. In *DISC '01: Proceedings of the 15th International Conference on Distributed Computing*, pages 78–92, London, UK, 2001. Springer-Verlag.

[8] A. Sen. *Techniques for Formal Verification of Concurrent and Distributed Program Traces*. PhD thesis, University of Texas at Austin, May 2004.

[9] A. Sen and V. K. Garg. Detecting temporal logic predicates on the happened-before model. In *IPDPS '02: Proceedings of the 16th International Symposium on Parallel and Distributed Processing*, pages 76–83, Washington, DC, USA, Apr. 2002. IEEE Computer Society.

[10] A. Sen and V. K. Garg. Detecting temporal logic predicates in distributed programs using computation slicing. In *OPODIS '03: Proceedings of the 7th International Conference on Principles of Distributed Systems*, pages 171–183, La Martinique, France, Dec. 2003.

[11] G. Winskel. Event structures. In *Advances in Petri nets 1986, part II on Petri nets: applications and relationships to other models of concurrency*, pages 325–392, New York, NY, USA, 1987. Springer-Verlag New York, Inc.