Intractability Results in Predicate Detection

Sujatha Kashyap
Dept. of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712
Email: kashyap@ece.utexas.edu

Vijay K. Garg
Dept. of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712
Email: garg@ece.utexas.edu

Abstract

It has been shown that global predicate detection in a distributed computation is an NP-complete problem in general. However, polynomial-time predicate detection algorithms exist for some classes of predicates, such as stable predicates, observer-independent predicates, conjunctions of local predicates etc. We show here that, given a class of predicates for which polynomial-time detection algorithms exist, it is in general NP-hard to determine whether a given boolean predicate is a member of that class.

We also explore the importance of the *efficient advancement property* for linear predicates. In particular, we show that there is no polynomial-time algorithm for the detection of linear predicates that do not satisfy the efficient advancement property, unless NP=RP.

Keywords: distributed systems, distributed debugging, predicate detection, NP-completeness.

I. Introduction

Global predicate detection is a fundamental problem in distributed computing, arising in contexts such as the design, testing and debugging of distributed programs. The absence of shared memory and a shared clock in a distributed system makes it difficult to observe global properties in such systems. Also, the set of events that make up a distributed computation can only be partially ordered [1], and a distributed computation can have a large number of distinct, yet valid, executions that satisfy this partial order. Examining all possible valid executions to determine whether a global property was satisfied results in a combinatorial explosion.

The problem of predicate detection has been shown to be NP-complete in general [2]. However, polynomial-time algorithms exist for detecting certain classes of predicates. Examples of such predicate classes are stable predicates [3], conjunctions of local predicates [4] and observer-independent predicates [5]. Polynomial-time algorithms also exist for detection of linear [2] and regular [6] predicates that satisfy the *efficient advancement property*.

Thus, we have a set of polynomial-time algorithms that can efficiently detect certain "tractable" classes of predicates in a distributed computation. It is thus natural to ask, given a predicate and a distributed computation, whether the predicate belongs to one of these classes. In this paper, we show that this problem is NP-hard in general.

We also explore the importance of the efficient advancement property for the detection of linear predicates and regular predicates. We show that, unless NP=RP, there is no efficient algorithm for detecting linear or regular predicates that do not satisfy the efficient advancement property.

II. Model and Background

We model a distributed program as a set of processes $\{P_1, ..., P_n\}$ with no shared memory and no global clock. Each process P_i executes sequentially, and is modeled as a sequence of (local) states s_{i1}, ..., s_{im}. Process P_i communicates with process P_j through asynchronous messages. Communication channels are not assumed to be FIFO. Each process P_i has a set of local variables X_i. The value of a local variable on a process changes only on transitions between states on that process. Thus, in any state s_{ij} on process P_i, the value of each variable $x \in X_i$ is well-defined.

A distributed computation is viewed as a set of states (partially) ordered by a *happened-before* relation that is analogous to Lamport's happened-before relation between events, $\rightarrow$ [1]. Formally, this happened-before relation states that, given two states s and t from the set of all states S, $s \rightarrow t$ iff s occurs before t on the same process, or s is the send of a message and t is the corresponding receive of the message, or $\exists u \in S : (s \rightarrow u) \wedge (u \rightarrow t)$. We define a distributed program as a set of states together with the happened-before relation, and denote it by $(S, \rightarrow)$.

Two states s and t are *concurrent* $(s||t)$ if $(s \not\rightarrow t) \wedge (t \not\rightarrow s)$. Given a set of processes $\{P_1, ..., P_n\}$ in a distributed computation, a *cut* G is a collection of local states $G \subseteq S$, such that G includes exactly one state from each process P_i, i.e., $|G| = n$, where n is the number of processes in the system. We denote the local state from P_i in G by $G[i]$. A cut is called a *consistent cut* or a *global state* iff $\forall i, j \in \{1..n\}, i \neq j : G[i]||G[j]$. Given two cuts G and H, we say that $G \leq H$ iff $\forall i \in \{1..n\} : (G[i] \rightarrow H[i]) \vee (G[i] = H[i])$.

We denote the set of all consistent cuts of a distributed computation $(S, \rightarrow)$ by $C(S)$. $C(S)$ forms a distributive lattice under the relation $\subseteq$ [7], [6]. An *execution* of a distributed computation $(S, \rightarrow)$ is a path through the lattice $C(S)$, starting from the global state which is the *inf* of $C(S)$, and ending at the global state which is the *sup* of $C(S)$.

A *subcut* of a distributed computation is a subset $G \subseteq S$ containing at most one state from each process, so that $|G| \leq n$, where n is the total number of processes in the distributed system under consideration. A consistent subcut is a subcut in which all states are pairwise concurrent. It has been shown [2] that any consistent subcut can be extended to a consistent cut.

A predicate B is *possibly* true in a computation (denoted by $possibly : B$) [8] iff it holds true at one or more global states in $C(S)$. A predicate B is *invariant* (denoted by $invariant : B$) iff the predicate is true at every global state in $C(S)$.

The modalities *possibly* and *invariant* are duals of each other, *i.e.*, $possibly : B = \neg invariant : \neg B$. Chase and Garg [2] showed that detecting $possibly : B$ for a general boolean predicate B is an NP-complete problem. Thus, detecting $invariant : B$ is co-NP-complete for a general boolean predicate B.

For the discussions in this paper, we assume that a predicate can be evaluated in polynomial time on a global state, *i.e.*, it can be determined to be true or false at that global state in polynomial time. Under this assumption, efficient algorithms have been proposed for detecting various classes of predicates, *i.e.*, predicates that exhibit a certain structure or satisfy certain properties. We focus here on classes of predicates that satisfy certain properties with respect to the lattice of all consistent global states, $C(S)$. Linear, post-linear [2], and regular [6] predicates are examples of such classes of predicates.

Definition 1: Let $C(S)$ be the set of consistent cuts of a computation $(S, \rightarrow)$. A predicate B is *meet-closed* with respect to the computation $(S, \rightarrow)$ iff

$$\forall G, H \in C(S) : B(G) \wedge B(H) \Rightarrow B(G \cap H)$$

Given a distributed computation $(S, \rightarrow)$, a predicate B, and a cut $G \subset S$, a state $G[i]$ is called *forbidden* if its inclusion in any cut H, where $G \leq H$, implies that B is false in H.

Definition 2: Given a boolean expression B,

$$forbidden(G, i) \stackrel{\text{def}}{=} \forall H : G \leq H : (G[i] \neq H[i]) \vee \neg B(H)$$

Definition 3: A boolean predicate B is *linear* with respect to the computation $(S, \rightarrow)$ iff

$$\forall G \in C(S) : \neg B(G) \Rightarrow \exists i : forbidden(G, i)$$

A predicate B is linear iff it is meet-closed [2]. Local predicates, conjunctions of local predicates and most channel predicates are linear. A *post-linear* predicate is the dual of a linear predicate.

Definition 4: Let $C(S)$ be the set of consistent cuts of a computation $(S, \rightarrow)$. A predicate B is *post-linear* iff

$$\forall G, H \in C(S) : B(G) \wedge B(H) \Rightarrow B(G \cup H)$$

Or, equivalently,

$$\forall G \in C(S) : \neg B(G) \Rightarrow \exists i : \forall H \leq G : (G[i] \neq H[i]) \vee \neg B(H)$$

A predicate that is both linear and post-linear is called a *regular* predicate.

Definition 5: Let $C(S)$ be the set of consistent cuts of a computation $(S, \rightarrow)$. A predicate B is regular with respect to $(S, \rightarrow)$ iff

$$\forall G, H \in C(S) : B(G) \wedge B(H) \Rightarrow B(G \cap H) \wedge B(G \cup H)$$

The set of all global states satisfying a regular predicate forms a sublattice of $C(S)$. Local predicates, conjunctions of local predicates, and many channel predicates are regular.

Chase and Garg [2] proposed an efficient algorithm for detecting $possibly : B$ for a linear predicate B, under the assumption that the forbidden state can be determined in polynomial time. This assumption is called the *efficient advancement property*.

III. Recognizing Linear and Regular Predicates

As discussed earlier, efficient detection algorithms exist for various classes of predicates. Thus, given a boolean expression B, one would like to determine if it belongs to a "tractable" predicate class, in which case detection of the predicate may be performed efficiently. We first consider the classes of linear and regular predicates. In this section, we show that determining whether a given boolean expression is linear with respect to a given distributed computation is a co-NP-complete problem. We also show that this problem is co-NP-complete for regular predicates, as well as post-linear predicates.

We define the decision problems of predicate recognition for linear and regular predicates as follows.

LINEARITY: Given a boolean expression b and a distributed computation $(S, \rightarrow)$, is b linear with respect to $(S, \rightarrow)$?

REGULARITY: Given a boolean expression b and a distributed computation $(S, \rightarrow)$, is b regular with respect to $(S, \rightarrow)$?

Theorem 1: LINEARITY is co-NP-complete.

Proof: LINEARITY is in co-NP: If the given predicate is not linear, then there exist global states G and H in which the predicate is true, such that the predicate is false in the global state $G \cap H$. The states G and H form the certificate for LINEARITY to be in co-NP, since it can be verified in polynomial-time that the predicate holds true in G and H, but is false in $G \cap H$. Thus, LINEARITY is in co-NP.

LINEARITY is co-NP-hard: We transform an arbitrary instance of the well-known co-NP-complete problem of determining whether a given boolean expression is a tautology, to an instance of LINEARITY.

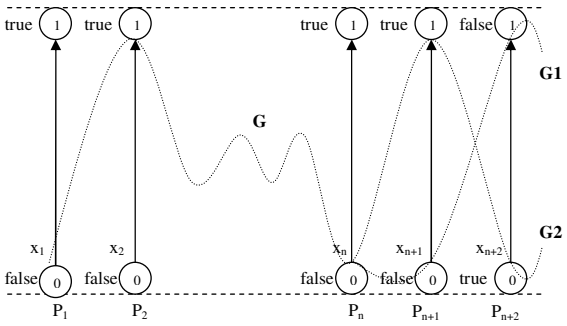

Fig. 1. Transformation for Theorems 1 and 2.

Let *b* be a boolean expression involving variables $x_1, x_2, ..., x_n$. $\forall i \in \{1..n\}$, we place each x_i on a separate process, P_i. Each of these n processes has two local states, a *true* state and a *false* state, which corresponds to the value taken by the variable x_i in that local state. We also define two new variables, x_{n+1} and x_{n+2}, and place them on processes P_{n+1} and P_{n+2} respectively. Process P_{n+1} has two local states: an initial *false* state and a final *true* state, and process P_{n+2} has two local states: an initial *true* state and a final *false* state. Figure 1 shows this transformation. It is evident that this transformation can be achieved in polynomial time.

We define

$$B = b \vee x_{n+1}x_{n+2} \vee \overline{x}_{n+1}\overline{x}_{n+2}$$

We claim that B is linear iff b is a tautology. If b is a tautology, then B is trivially linear, because B will be true for all global states. Conversely, if b is not a tautology, then there exists a subcut involving processes P_1, ..., P_n in which b evaluates to false. Let us call this subcut G. We can now extend the subcut G to form two cuts, $G1$ and $G2$, in which the predicate B is true, as shown in Figure 1.

$$G1 = (G, 0, 1)$$

and

$$G2 = (G, 1, 0)$$

However,

$$G1 \cap G2 = (G, 0, 0)$$

in which the predicate B is false. Thus, B is not linear. ■

Theorem 2: REGULARITY is co-NP-complete.

Proof: REGULARITY is in co-NP: If the given predicate is not regular, then there exist global states G and H in which the predicate is true, such that the predicate is false either in $G \cap H$ or in $G \cup H$. The global states G and H thus form the certificate for REGULARITY to be in co-NP.

REGULARITY is co-NP-hard: The transformation in Theorem 1 holds for REGULARITY as well. That is, b is a tautology iff $B = b \vee x_{n+1}x_{n+2} \vee \overline{x}_{n+1}\overline{x}_{n+2}$ is regular. If b is a tautology, then B is trivially regular, because B is true for all global states. Conversely, if b is not a tautology, then B is not regular because both the intersection and union of $G1$ and $G2$ result in a global state in which B is false. Thus, REGULARITY is co-NP-hard. ■

Note that the above transformation can also be used to show that the problem of deciding whether a given boolean predicate is post-linear is co-NP-complete.

IV. Predicate Recognition

In the previous section, we showed that the problems of recognizing linear, regular and post-linear predicates are co-NP-complete. We now turn to the question: what other classes of predicates cannot be recognized in polynomial-time?

Let us denote the set of all non-satisfiable boolean expressions by C_{false}. Observe that any predicate $B_{false} \in C_{false}$ is stable, observer-independent, linear, post-linear and regular with respect to *any* distributed computation, because it would evaluate to $false$ in every global state.

Theorem 3: Given a predicate class $\mathcal{C}$ and a distributed computation $(S, \rightarrow)$, such that:

- $C_{false} \subseteq \mathcal{C}$, and
- $\forall B' \in \mathcal{C} : possibly : B'$ can be detected efficiently for the computation $(S, \rightarrow)$.

It is NP-hard to determine whether a given boolean predicate B is a member of the class $\mathcal{C}$, *i.e.*, $B \in \mathcal{C}$, with respect to the computation $(S, \rightarrow)$.

Proof: We show that if a polynomial-time algorithm exists for determining membership in a predicate class $\mathcal{C}$ as defined above, then the satisfiability problem (SAT) can also be solved in polynomial time. Let B be a boolean expression involving variables $x_1, x_2, ..., x_n$, for which we wish to solve SAT. We create a distributed computation as follows: $\forall i \in \{1..n\}$, we place each x_i on a separate process, P_i. Each of these n processes has two local states, a *false* state and a *true* state, which corresponds to the value taken by the variable x_i in that local state, as shown in Figure 2.

Assume that there exists a polynomial-time algorithm $member(\mathcal{C}, B, S)$, which can determine whether $B \in \mathcal{C}$, with respect to the computation $(S, \rightarrow)$. If $member(\mathcal{C}, B, S)$ returns $false$, then B is satisfiable, since $C_{false} \subseteq \mathcal{C}$. If it returns $true$, then since $possibly : B$ can be detected efficiently for all $B \in \mathcal{C}$, we can determine satisfiability for B (B is satisfiable iff $possibly : B$ is true).

■

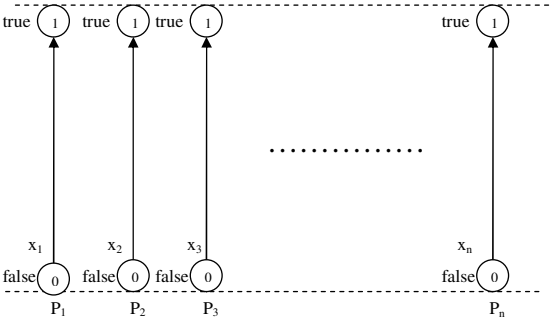

Fig. 2. Distributed computation used in the proofs of Theorems 3 and 4.

Now, let us denote the set of all tautological boolean expressions by C_{true}. Note that any predicate $B_{true} \in C_{true}$ is stable, observer-independent, linear, post-linear and regular with respect to any distributed computation, because it would evaluate to $true$ in every global state.

Theorem 4: Given a predicate class $\mathcal{C}$ and a distributed computation $(S, \rightarrow)$, such that:

- $C_{true} \subseteq \mathcal{C}$, and
- $\forall B' \in \mathcal{C} : invariant : B'$ can be detected efficiently for the computation $(S, \rightarrow)$.

It is co-NP-hard to determine whether a given boolean predicate B is a member of the class $\mathcal{C}$, *i.e.*, $B \in \mathcal{C}$, with respect to the computation $(S, \rightarrow)$.

Proof: We show that if a polynomial-time algorithm exists for determining membership in a predicate class $\mathcal{C}$ as defined above, then the co-NP-complete problem of determining whether a given boolean expression is a tautology (TAUTOLOGY) can also be solved in polynomial time. Let the boolean expression B, involving variables $x_1, x_2, ..., x_n$, be an arbitrary instance of TAUTOLOGY. We create a distributed computation identical to that used in the proof of Theorem 3.

Assume that there exists a polynomial-time algorithm $member(\mathcal{C}, B, S)$, which can determine whether $B \in \mathcal{C}$ with respect to the computation $(S, \rightarrow)$. If $member(\mathcal{C}, B, S)$ returns $false$, then B is not a tautology, since $C_{true} \subseteq \mathcal{C}$. If it returns $true$, then since $invariant : B$ can be detected efficiently for all $B \in \mathcal{C}$, we can determine if B is a tautology (B is a tautology iff $invariant : B$ is true). ■

V. Efficient Advancement Property

We stated earlier that efficient detection of linear predicates relies on the assumption that the given predicate satisfies the efficient advancement property, that is, the forbidden state can be identified in polynomial time. The question that arises is, do all linear predicates satisfy the efficient advancement property? If not, then is it possible to efficiently detect linear predicates that do not satisfy this property? We show here that, unless NP=RP, there exist linear predicates that cannot be detected in polynomial-time.

We use a result by Valiant and Vazirani [9], which states that satisfiability is NP-hard under randomized reductions even for instances that have at most one satisfying assignment (USAT). Valiant and Vazirani's proof uses a randomized polynomial-time algorithm that reduces a given instance of SAT to an instance of USAT.

Theorem 5: **(Valiant-Vazirani)** If there exists a randomized polynomial-time algorithm for solving instances of SAT having at most one satisfying assignment, then NP=RP.

We know that a predicate having at most one satisfying assignment in a given distributed computation is linear with respect to that computation. Given any instance B of USAT, involving variables $x_1, x_2, ..., x_n$, one can create a distributed computation as depicted in Figure 2, where B would be a linear predicate. Detecting $possibly : B$ for the computation in Figure 2 is thus equivalent to solving USAT for B. Since we know that linear predicates that satisfy efficient advancement can be detected in polynomial-time, this indicates that linear predicates that do not exhibit efficient advancement may not be detectable in polynomial-time, even by a randomized algorithm. Furthermore, since every instance of USAT is also a regular predicate for the computation in Figure 2, detection of regular predicates is also NP-hard under randomized reductions.

VI. Conclusion

In this paper, we have shown several intractability results in the area of predicate detection in distributed computations. We have shown the co-NP-hardness of deciding whether a given boolean expression is linear, post-linear or regular with respect to a given distributed computation. Further, we showed the NP-hardness of determining membership in classes of predicates that have efficient algorithms for $possibly$ detection, and the co-NP-hardness of determining membership in classes on which $invariant$ can be detected efficiently. We also showed that polynomial-time detection may not be possible for linear and regular predicates, unless the efficient advancement property is satisfied.

References

[1] L. Lamport, "Time, clock and the ordering of events in a distributed system," *Communications of the ACM (CACM)*, vol. 21, no. 7, pp. 558–565, July 1978.

[2] C. Chase and V. K. Garg, "On techniques and their limitations for the global predicate detection problem," in *Proceedings of the Workshop on Distributed Algorithms*, Le Mont-Saint-Michel, France, Sept. 1995, pp. 303–307.

[3] K. M. Chandy and L. Lamport, "Distributed snapshots: Determining global states of distributed systems," *ACM Transactions on Computer Systems*, vol. 3, no. 1, pp. 63–75, Feb. 1985.

[4] V. K. Garg and B. Waldecker, "Detection of weak unstable predicates in distributed programs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 3, pp. 299–307, Mar. 1994.

[5] B. Charron-Bost *et al.*, "Local and temporal predicates in distributed systems," *ACM Transactions on Programming Languages and Systems*, vol. 17, no. 1, pp. 157–179, Jan. 1995.

[6] V. K. Garg and N. Mittal, "On slicing a distributed computation," in *21st International Conference on Distributed Computing Systems (ICDCS 01)*, Washington-Brussels-Tokyo, Apr. 2001, pp. 322–329.

[7] F. Mattern, "Virtual time and global states of distributed systems," in *Parallel and Distributed Algorithms: proceedings of the International Workshop on Parallel & Distributed Algorithms*, M. Cosnard *et al.*, Eds. Elsevier Science Publishers B. V., 1989, pp. 215–226.

[8] R. Cooper and K. Marzullo, "Consistent detection of global predicates," in *Proceedings of the ACM/ONR Workshop on Parallel and Distributed Debugging*, Santa Cruz, California, 1991, pp. 163–173.

[9] L. G. Valiant and V. V. Vazirani, "NP is as easy as detecting unique solutions," *Theoretical Computer Science*, vol. 47, no. 1, pp. 85–93, 1986.