Theoretical Computer Science ••• (••••) •••–•••

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Efficient abstraction algorithms for predicate detection ☆

Aravind Natarajan [a], Himanshu Chauhan [b,*], Neeraj Mittal [a], Vijay K. Garg [b]

[a] School of Engineering and Computer Science, The University of Texas at Dallas, United States
[b] Department of Electrical and Computer Engineering, The University of Texas at Austin, United States

ARTICLE INFO

Article history:
Received 9 April 2014
Received in revised form 17 November 2015
Accepted 24 December 2015
Available online xxxx

Keywords:
Predicate detection
Temporal logic
Online algorithm
Distributed algorithm

ABSTRACT

Analyzing a distributed computation is a hard problem in general due to the combinatorial explosion in the size of the state-space with the number of processes in the system. By abstracting the computation, unnecessary state explorations can be avoided. Computation slicing is an approach for abstracting distributed computations with respect to a given predicate. To detect a predicate in a timely manner during run-time verification, it is important to be able to compute the slice in an *incremental* manner as and when a new event is generated in the system. In this work, we present several online abstraction algorithms for computing the slice of a distributed computation with respect to regular predicates and its temporal extensions. The family of regular predicates is fairly rich and covers many commonly used predicates for run-time verification. We first present several online algorithms for computing the slice with respect to certain temporal logical regular predicates all of which are centralized. We then present a distributed algorithm for computing the slice with respect to a regular state based predicate; it distributes the work and storage requirements across the system, thus reducing the space and computation complexity per process. We also extend the slicing technique to non-regular predicates by presenting algorithms for computing the slice for two temporal logic operators when the predicate used in the operator is not regular.

1. Introduction

Verifying system states and global properties is a critical aspect of maintaining consistency and detecting constraint violations for reliable distributed system. Global predicate detection [1] for run-time verification is an important technique for detecting violations of invariants for debugging and fault-tolerance in distributed systems. The problem involves analyzing a single run, often called a *trace*, of a computation to construct all possible states of the system that could possibly satisfy a given predicate that encodes some constraint violation. It is a challenging task on a large system with a large number of processes due to the combinatorial explosion of the state space. In this paper, we focus on predicate detection in asynchronous distributed systems. The problem, however, is not only applicable to conventional distributed systems, but also to multicore computing. With growing popularity of large number of CPU-cores on processing chips, some manufacturers [2] are exploring the distributed computing model on chip with no shared memory between the cores. On such machines, information

☆ Supported by NSF CNS-1346245, NSF CNS-1115808, and the Cullen Trust for Higher Education Endowed Professorship. Parts of this work appeared in the Proceedings of the 32nd IEEE Symposium on Reliable Distributed Systems (SRDS), 2013 and the Proceedings of the 15th International Conference on Distributed Computing and Networking (ICDCN), 2014.

* Corresponding author.
E-mail address: himanshu@utexas.edu (H. Chauhan).

http://dx.doi.org/10.1016/j.tcs.2015.12.037
0304-3975/

exchange between the cores is performed by only using message passing [3]. Recent research efforts [4] have shown that with sufficiently fast on-chip networking support message passing based architecture on multicore chips can be significantly fast in performance for some specific computational tasks. Hence, techniques of predicate detection for distributed systems can also prove beneficial for monitoring computations on such systems.

Multiple algorithms have been proposed in literature for detection of global predicates in both offline and online manner (*e.g.* [1,5,6]). Online predicate detection is important for many system models: continuous services (such as web-servers), collection of continuous observations (such as sensor-networks), and parallel search operations on large clusters. However, performing predicate detection in a manner that is oblivious to the structure of the predicate can lead to large run-time, and high memory overhead. The approach of using mathematical abstractions for designing and analyzing computational tasks has proved to be significantly advantageous in modern computing. In the context of predicate detection, and run-time verification, the problem of abstraction can be viewed as the problem of taking a distributed computation as input and outputting a smaller distributed computation that abstracts out parts that are not relevant to the predicate under consideration. The abstract computation may be exponentially smaller than the original computation resulting in significant savings in predicate detection time.

Computation slicing is an abstraction technique for efficiently finding all global states, of a distributed computation, that satisfy a given global predicate, without explicitly enumerating all such global states [5]. The *slice* of a computation with respect to a predicate is a sub-computation that satisfies the following properties: (a) it contains all global states of the computation for which the predicate evaluates to true, and (b) of all the sub-computations that satisfy condition (a), it has the least number of global states. The slice has much fewer global states than the computation itself—exponentially smaller in many cases—resulting in substantial savings.

In this work, we focus on abstracting distributed computations with respect to *regular* predicates and its temporal extensions (defined in Section 3). The family of *regular* predicates contains many useful predicates that are often used for run-time verification in distributed systems.

1.1. Challenges and contributions

Computing the slice of a computation with respect to a general predicate is an NP-Hard problem in general [7]. Many classes of predicates have been identified for which the slice can be computed efficiently in polynomial-time (*e.g.*, regular predicates, co-regular predicates, linear predicates, relational predicates, stable predicates) [7,5,8]. Polynomial-time slicing algorithms have also developed for temporal logic predicates obtained by applying temporal operators *AG*, *EG* and *EF* to regular predicates [9–11]. For convenience, we refer to non-temporal logic predicates as *state-based predicates* (evaluated on a single global state) and temporal logical predicates as *path-based predicates* (evaluated on sequences of global states).

Most of the existing algorithms for slicing a distributed computation are *offline* in nature; they assume that all events are known *a priori*. Further, all of the existing algorithms for slicing a distributed computation are *centralized* in nature; the slice is computed by a single *slicer* process.

Online slicing algorithms To detect a predicate in a timely manner during run-time verification, it is desirable to be able to compute the slice in an incremental manner that minimizes the amount of work that needs to be done when a new event is generated in the system and may cause some property of interest to become true. Online slicing algorithms have been developed for state-based predicates [5]. However, to our knowledge, no online slicing algorithms currently exist for path-based predicates. The slice for a state-based predicate can only grow on arrival of a new event. But, the slice for a path-based predicate may grow or shrink on arrival of a new event. In this work, we present centralized online algorithms, Pseudo-Codes 1–5 in § 5, for computing the slice of a distributed computation with respect to temporal logic predicates $AG(B)$, $EG(B)$ and $EF(B)$, where B is a state-based regular predicate. Our algorithms have amortized time-complexity of $O(n^2)$, where n denotes the number of processes in the system.

Distributed slicing algorithm For systems with large number of processes, centralized algorithms require a single process to perform high number of computations, and to store very large data. In comparison, a distributed online algorithm significantly reduces the per process costs for both computation and storage. Additionally, for predicate detection, the centralized online algorithm requires at least one message to the slicer process for every relevant event in the computation, resulting in a bottleneck at the slicer process. A method of devising a distributed algorithm from a centralized algorithm is to decompose the centralized execution steps into multiple steps to be executed by each process independently. However, for performing online abstraction using computation slicing, such an incremental modification is inefficient as direct decomposition of the steps of the centralized online algorithm requires that each process sends its local state information to all the other processes whenever the local state (or state interval) is updated. In addition, a simple decomposition leads to a distributed algorithm that wastes significant computational time as multiple processes may end up visiting (and enumerating) the same global state. Thus, the task of devising an efficient distributed algorithm for *slicing* is non-trivial. In this paper, we propose a distributed online algorithm, Pseudo-Code 6 in § 6, to detect state based regular predicates. This algorithm exploits not only the nature of the predicates, but also the collective knowledge across processes. The optimized version of our algorithm reduces the required storage per *slicing* process, and computational workload per *slicing* process by $O(n)$.

Please cite this article in press as: A. Natarajan et al., Efficient abstraction algorithms for predicate detection, Theoret. Comput. Sci. (2016), http://dx.doi.org/10.1016/j.tcs.2015.12.037