Safety, Liveness and Fairness in Temporal Logic

A. Prasad Sistla
Department of Electrical Engineering and Computer Science, University of Illinois at Chicago, Chicago, Illinois, USA

Keywords: Concurrent programs; Verification; Temporal logic; Safety properties; Liveness properties; Fairness properties; Algorithms

Abstract. In this paper we present syntactic characterization of temporal formulas that express various properties of interest in the verification of concurrent programs. Such a characterization helps us in choosing the right techniques for proving correctness with respect to these properties. The properties that we consider include safety properties, liveness properties and fairness properties. We also present algorithms for checking if a given temporal formula expresses any of these properties.

1. Introduction

In the verification of concurrent programs two kinds of properties are of primary importance and have been extensively investigated ([Lam77]): safety properties and liveness properties. Safety properties assert that something bad never happens, while liveness properties assert that something good will eventually happen. The classification of properties into safety properties and liveness properties allows us to choose the most appropriate proof method for proving correctness with respect to these properties. For example, methods based on global invariants have been extensively used for safety properties, while methods based on proof lattices or well-founded induction have been employed for liveness properties (see [OwL82]).
Temporal Logic has been proposed in [Pnu77] (also see [MaP92]) as an appropriate formalism in the specification and verification of concurrent programs. Since then, many different versions of temporal logics have been used in the verification of concurrent programs [MaP89, LPZ85, SiC85, CES86]. Due to this wide interest, it becomes important to present a syntactic classification of formulas in temporal logic that specify safety and liveness properties.

Knowing whether a formula specifies a safety or a liveness property, helps us in choosing the right proof method for verifying that a given concurrent program satisfies the property specified by the formula. Additionally, it is well known that all fair executions 2 of a program satisfy a safety property iff all executions (fair or unfair) satisfy the property. This means that we need not be concerned about fairness for proving safety properties. On the other hand, fairness is often essential in proving general liveness properties. Thirdly, proving that a program satisfies a property may not always be feasible due to many reasons. In such cases, monitoring the executions of a program for violations of the property may be an alternative. It is possible, in principle, to monitor an execution of a concurrent program for violations of safety properties. In contrast, this is not possible for liveness properties. Thus, knowing if a formula expresses a safety property helps us determine whether we can monitor the execution of a program for violations of the property. Motivated by such concerns, in this paper, we investigate the possibility of syntactically characterizing safety, liveness and fairness properties in Propositional Linear Temporal Logic (PTL).

Formally, a property is simply a set of infinite sequences of states. A definition of safety properties was first given by Lamport [Lam85] (we call these as L-safety properties), and a more general definition is given in [AlS85]. We focus on the later definition. A safety property is the same as a limit closure property defined in [Eme83]. In [ADS86], it has been shown that the L-safety properties are the same as the class of safety properties that are closed under stuttering. Intuitively, closure under stuttering requires that the property be insensitive to successive repetition of any state of a sequence.

In this paper, we introduce a new class of properties called strong safety properties. A property \( C \) is a strong safety property if it is a safety property that is closed under stuttering, and is insensitive to deletion of states, i.e. from any sequence in \( C \) if we delete an arbitrary number of states, then the resulting sequence is also in \( C \). The class of strong safety properties is a strict subset of the class of safety properties that are closed under stuttering. Intuitively, closure under stuttering requires that the property be insensitive to successive repetition of any state of a sequence.

We give a syntactic characterization of PTL formulas that express safety properties, safety properties closed under stuttering and strong safety properties. Specifically, we show that all positive formulas formed using only \( \forall \) (unless) and \( \bigcirc \) (nexttime) express safety properties, and all positive formulas formed using only \( \forall \) express safety properties that are also closed under stuttering. We go on to show that all positive formulas formed using only \( \Box \) (always) express strong safety properties. In this case, we show the completeness result as well, i.e. every strong safety property expressible in PTL can be expressed using positive formulas that only use the \( \Box \) modal operator. In fact, we prove a stronger result that shows that any strong safety property expressed by a finite state automaton can be expressed by a positive formula that only uses the \( \Box \) temporal operator.

---

2 We give a formal definition of fairness properties, which may be viewed as a special class of liveness properties.