Deciding Global Partial-Order Properties

Rajeev Alur* Ken McMillan** Doron Peled***

Abstract. Model checking of asynchronous systems is traditionally based on the interleaving model, where an execution is modeled by a total order between events. Recently, the use of partial order semantics that allows independent events of concurrent processes to be unordered is becoming popular. Temporal logics that are interpreted over partial orders allow specifications relating global snapshots, and permit reduction algorithms to generate only one representative linearization of every possible partial-order execution during state-space search. This paper considers the satisfiability and the model checking problems for temporal logics interpreted over partially ordered sets of global configurations. For such logics, only undecidability results have been proved previously. In this paper, we present an EXPSPACE decision procedure for a fragment that contains an eventuality operator and its dual. We also sharpen previous undecidability results, which used global predicates over configurations. We show that although our logic allows only local propositions (over events), it becomes undecidable when adding some natural until operator.

1 Introduction

The model checking problem is to decide whether a finite-state description of a reactive system satisfies a temporal-logic specification. The solutions to this problem have been implemented, and the resulting tools are increasingly being used as debugging aids for industrial designs. All of these solutions employ the so-called interleaving semantics in which a single execution of the system is considered to be a totally-ordered sequence of events. The (linear) semantics of the system is, then, a set of total-order executions that the system can possibly generate. The specifications are written in the linear temporal logic LTL. The model checker checks if every execution of the system satisfies the LTL-specification.

In contrast to the interleaving semantics, the partial order semantics considers a single execution as a partially ordered set of events. The partial order semantics does not distinguish among total-order executions that are equivalent up to reordering of independent events, thereby, resulting in a more abstract and faithful representation of concurrency, and has attracted researchers in concurrency theory for many years [8, 12].

* University of Pennsylvania and Bell Laboratories, Email: alur@cis.upenn.edu
** Cadence Berkeley Labs, Email: mcmillan@cadence.com
*** Bell Laboratories and CMU, Email: doron@research.bell-labs.com

4 In the alternative branching-time paradigm, the semantics of the system is a labeled state-transition graph, and the specification is given as a formula of the Computation tree logic (CTL) [2]. Branching-time versions of partial-order semantics are possible (see, for instance, [4]), but are not studied in this paper.
Partial-order semantics, unlike the interleaving semantics, distinguishes between non-determinism and concurrency. Consequently, specification languages over partial orders can permit a direct representation of properties involving causality and concurrency, for example, serializability.

Partial order specifications are also interesting due to their compatibility with the so-called partial order reductions. The partial-order equivalence among sequences can be exploited to reduce the state-space explosion problem: the cost of generating at least one representative per equivalence class is typically significantly less than the cost of generating all interleavings [5, 9, 10, 15]. If the specification could distinguish between two sequences of the same equivalence class, as is the case with \(\text{LTL}\), the above equivalence cannot be used: the same equivalence class may contain both a sequence that satisfies the specification and a sequence that does not. It is possible to refine the equivalence relation, providing more representatives at the expense of using a bigger state space [10]. The alternative solution is to use a specification logic that is directly interpreted over partial orders. This latter approach demands study of decision problems for partial order logics.

How does one define a temporal logic over partial orders? Two approaches have been proposed to write temporal requirements over a model consisting of a set of partially ordered events, or local states of processes. In local partial order logics, the truth of a formula is evaluated at a local state, and the temporal modalities relate causal precedences among local states. Examples of such logics include \(\text{TRPTL}\) [13] and TLC [1]. In global partial order logics, the truth of a formula is evaluated in a global state, also called a configuration or a slice, which consists of a consistent set of local states. The temporal modalities of a global logic, such as \(\text{ISTL}\) [6], relate causal precedences among configurations. Global partial order logics are strictly more general than the local ones. In a partial order, unlike in a total order, there are many ways to proceed from one state to the next, and consequently, the syntax of partial order logics, uses path quantifiers as in \(\text{CTL}\). For example, if \(p\) is a global state predicate asserting that states of all processes are consistent with one another, then the ISTL-formula \(\exists p\) asserts that \(p\) is a possible global snapshot. A system satisfies \(\exists p\) if every partial-order execution has some linearization containing a \(p\)-state. It should be noted that this property cannot be specified in \(\text{LTL}\) or \(\text{CTL}\) or any local partial-order logic.

Before we consider the model checking for partial order logics, let us briefly review the solution to the model checking problem for \(\text{LTL}\). A system \(M\) is viewed as an \(\omega\)-automaton \(A_M\) that generates all possible total-order executions of \(M\). To check whether the system satisfies an \(\text{LTL}\)-formula \(\varphi\), the model-checking algorithm first constructs an \(\omega\)-automaton \(A_{\neg \varphi}\) that accepts all the satisfying models of \(\neg \varphi\), and tests emptiness of the intersection of the languages of the two automata \(A_M\) and \(A_{\neg \varphi}\) [16]. For algorithmic verification of partial order logics, the solution is to construct, given a partial order specification \(\varphi\), an \(\omega\)-automaton \(A_{\neg \varphi}\) that accepts the linearizations of the partial order models of \(\neg \varphi\). To check whether the system \(M\) satisfies a partial order formula \(\varphi\), we need to test emptiness of the intersection of \(A_{\neg \varphi}\) and the automaton \(A_M\). Since we know that the automaton \(A_{\neg \varphi}\) does not distinguish among the linearizations of the same partial order, the above approach yields a correct result even if \(A_M\) generates only one linearization of each partial order execution of \(M\).