The Theory of Timed Automata
Rajeev Alur
AT&T Bell Laboratories
Murray Hill, NJ 07974, USA.

David Dill 1
Department of Computer Science
Stanford University, Stanford, CA 94305, USA.

Abstract. We propose timed automata to model the behavior of real-time systems over time. Our definition provides a simple, and yet powerful, way to annotate state-transition graphs with timing constraints using finitely many real-valued clocks. A timed automaton accepts timed words — strings in which a real-valued time of occurrence is associated with each symbol. We study timed automata from the perspective of formal language theory: we consider closure properties, decision problems, and subclasses. We discuss the application of this theory to automatic verification of real-time requirements of finite-state systems.

Keywords: Real-time systems, Automatic verification, Formal languages and Automata theory.

1 Introduction

Modal logics and ω-automata for qualitative temporal reasoning about concurrent systems have been studied in great detail (selected references: [Pnu77, MP81, EC82, Lam83, WVS83, Var87, Pnu86, CES86]) These formalisms abstract away from time, retaining only the sequencing of events. In the linear time model, it is assumed that an execution can be completely modeled as a linear sequence of states or system events, called an execution trace (or just trace). The behavior of the system is a set of execution sequences. Since a set of sequences is a formal language, this leads naturally to the use of automata for the specification and verification of systems. When the systems are finite-state, as many are, we can use finite automata, leading to effective constructions and decision procedures for automatically manipulating and analyzing system behavior. Even when automata are not used directly, they are never far away; for example, automata theory proves useful in developing the basic decision procedures for propositional linear temporal logic.

1Supported by the NSF under grant MIP-8858807.
Although the decision to abstract away from quantitative time has had many advantages, it is ultimately counterproductive when reasoning about systems that must interact with physical processes; the correct functioning of the control system of airplanes and toasters depends crucially upon real time considerations. We would like to be able to specify and verify models of real-time systems as easily as qualitative models. Our goal is to modify finite automata for this task.

For simplicity, we discuss models that consider executions to be infinite sequences of events, not states (the theory with state-based models differs only in details). Within this framework, it is possible to add timing to an execution trace by pairing it with a sequence of times, where the i'th element of the time sequence gives the time of occurrence of the i'th event. At this point, however, a fundamental question arises: what is the nature of time?

One alternative, which leads to the discrete-time model, requires the time sequence to be a monotonically increasing sequence of integers. This model is appropriate for certain kinds of synchronous digital circuits, where signal changes are considered to have changed exactly when a clock signal arrives. One of the advantages of this model is that it can be transformed easily into an ordinary formal language. Each timed trace can be expanded into a trace where the times increase by exactly one at each step, by inserting a special silent event as many times as necessary between events in the original trace. Once this transformation has been performed, the time of each event is the same as its position, so the time sequence can be discarded, leaving an ordinary string. Hence, discrete time behaviors can be manipulated using ordinary finite automata. Of course, in physical processes events do not always happen at integer-valued times. The discrete-time model requires that continuous time be approximated by choosing some fixed quantum a priori, which limits the accuracy with which physical systems can be modeled.

The fictitious-clock model is similar to the discrete time model, except that it only requires the sequence of integer times to be non-decreasing. The interpretation of a timed execution trace in this model is that events occur in the specified order at real-valued times, but only the (integer) readings of the actual times with respect to a digital clock are recorded in the trace. This model is also easily transformed into a conventional formal language. First, add to the set of events a new one, called tick. The untimed trace corresponding to a timed trace will include all of the events from the timed trace, in the same order, but with \( t_{i+1} - t_i \) number of ticks inserted between the i'th and the \((i + 1)\)th events (note that this number may be 0). Once again, it is conceptually simple to manipulate these behaviors using finite automata, but the compensating disadvantage is that it represents time only in an approximate sense.

We prefer a dense-time model, in which time is a dense set, because it is more realistic physically. In this model, the times of events are real numbers, which increase monotonically without bound. Dealing with dense time in a finite-automata framework is more difficult than the other two cases, because it is not obvious how to transform a set of dense-time traces into an ordinary formal language. Instead, we have developed a theory of timed formal languages and timed automata to support automated reasoning about such systems.