Liveness in Timed and Untimed Systems *

Rainer Gawlick1 Roberto Segala1 Jørgen Søgaard-Andersen1,2
Nancy Lynch1

1 Laboratory for Computer Science, MIT
2 Department of Computer Science, Technical University of Denmark

Abstract. We present a coordinated pair of general labeled transition system models for describing timed and untimed concurrent systems. Both of the models incorporate liveness properties as well as safety properties. The models are related via an embedding of the untimed model into the timed model, which preserves all the interesting attributes of the untimed model. Both models include notions of environment-freedom, which express the idea that the liveness properties can be guaranteed by the system, independently of the behavior of the environment in which it operates. These environment-freedom conditions are used to prove compositionality results for both models. This pair of models, which generalize several existing models, is intended to comprise a general formalism for the verification of timed and untimed concurrent systems.

1 Introduction

The increasing need for reliable software has led the scientific community to develop many formalisms for verification. Particularly important are formalisms that can model distributed and concurrent systems and those that can model real time systems, i.e., systems that rely on time constraints in order to guarantee correct behavior. Formalisms should be able to support the verification of both safety and liveness properties [3]. Roughly speaking, a liveness property specifies that certain desirable events will eventually occur, while a safety property specifies that certain undesirable events will never occur.

In this paper, we present a coordinated formalism that permits modeling and verification of safety and liveness properties for both timed and untimed systems. The formalism consists of two models, one timed and one untimed, with an embedding of the untimed model into the timed model. Both models come equipped with notions of external behavior and of implementation, which are based simply on traces. The formalism is intended to support a variety of verification techniques, including simulation methods, compositional reasoning, algebraic methods, and temporal logic methods.

The Input/Output (I/O) automaton model of Lynch and Tuttle [10] and it timed extension by Merritt, Modugno and Tuttle [13], have been used successfully in the past as a formalism for verification. I/O automata are state machines with a labeled transition relation where the labels, also called actions, model communication. A key feature of I/O automata is the explicit distinction between input and output actions, which characterize the events under the control of the environment and those under the control

of the automaton, respectively. I/O automata include a special type of liveness property called \textit{fairness}, also known as \textit{weak fairness}. An I/O automaton behaves fairly if it gives infinitely many turns to each of its subcomponents; a fair trace is a sequence of actions that occur during a fair execution. The distinction between input and output is used to justify the use of the simple notion of \textit{fair trace inclusion} as a notion of implementation, which in turn is important to the simulation based proof methods \cite{10-12}. The generalization to the timed case adds upper and lower time bounds for some of the subcomponents and augments the traces to include time information.

Unfortunately, I/O automata do not quite meet our needs. The problem is that there are some liveness properties that cannot be expressed naturally using just the simple notion of I/O automaton fairness; see \cite{15} for an example. This motivates the attempt to generalize the I/O automaton model to handle more general liveness properties, while retaining an implementation notion based on some form of trace inclusion.

A simple and natural generalization is suggested by the work of Abadi and Lamport \cite{2}, which models a machine as a pair \((A, L)\) consisting of an automaton \(A\) and a subset \(L\) of its executions satisfying the desired liveness property. The implementation notion can then be expressed by \textit{live trace inclusion} just as fair trace inclusion expresses implementation for I/O automata. Unfortunately, if \(L\) is not restricted, simple examples show that live trace inclusion is not compositional (c.f. Example 3).

In this paper, we identify the appropriate restrictions on \(L\), in the untimed and the timed model, so that live trace inclusion is compositional for the pair \((A, L)\). A pair \((A, L)\) satisfying these restrictions on \(L\) is called a \textit{live I/O automaton} in the untimed model and a \textit{live timed I/O automaton} in the timed model. The restrictions on \(L\) are given by a property called \textit{environment-freedom}, which captures the intuitive idea that a live (timed) I/O automaton may not constrain its environment. The environment-freedom property is defined, using ideas from Dill \cite{7}, by means of a two-person game between a live (timed) I/O automaton and its environment. Specifically, the environment provides any input, while the system tries to react so that it behaves according to its liveness property \(L\). A live (timed) I/O automaton has a \textit{winning strategy} against its environment if it has a way to behave according to \(L\) independently of its environment. If a live (timed) I/O automaton has a winning strategy it is said to be environment-free.

The environment-freedom property for the timed model is a natural extension of the one for the untimed model up to some technical details involving the so-called \textit{Zeno executions}. The close relation between the two definitions allows the timed and untimed models to be tied together, thus (for example) permitting the verification for timed implementations of untimed specifications. Specifically, we define an embedding, similar to the \textit{patient transducer} of \cite{17}, that converts live I/O automata into live timed I/O automata without timing constraints. The embedding, which is omitted from this abstract (see \cite{8}), preserves the environment-freedom property and the trace preorder relations of the untimed model. Furthermore, it commutes with the parallel composition operator.

Our model is closely related to several others in the literature. It captures the I/O automata of \cite{19}, the failure free complete trace structures of \cite{7}, and the timed I/O automata of \cite{13}. It generalizes the notion of \textit{strong I/O feasibility} introduced in \cite{17}. The untimed model is similar to the model of \cite{2}. However, the generalization of \cite{2} to the timed case \cite{1} is very complex, possibly because of the absence of a clear role for \textit{time} in the interaction between the automaton and its environment. In contrast, our generalization to the timed case is simple, and follows naturally from the untimed case.

It is already clear that our formalism supports a wide range of proof methods, including \textit{simulation methods} as described in \cite{11,12} (and extended to handle liveness in \cite{8,16}), \textit{compositional reasoning} as justified by the theorems of this paper, and \textit{tempo-