May 1988

Modeling Reliable Distributed Real-Time Programs

Victor Wolfe
University of Pennsylvania

Susan B. Davidson
University of Pennsylvania, susan@cis.upenn.edu

Insup Lee
University of Pennsylvania, lee@cis.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-81.

Modeling Reliable Distributed Real-Time Programs

Abstract

A model for distributed hard real-time programs should incorporate real-time characteristics and be capable of analyzing time-related reliability issues. We introduce a model called the Real-Tie Selection/ Resolution (RT-SIR) Model with these capabilities and demonstrate it by example.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-88-81.

MODELING RELIABLE DISTRIBUTED REAL-TIME PROGRAMS

Victor Wolfe, Susan B. Davidson
and Insup Lee

MS-CIS-88-81
GRASP LAB 157

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

October 1988

Acknowledgements: This research was supported in part by NSF grants IRI86-10617, DCR 8501482, DMC 8512838, MCS 8219196-CER, U.S. Army grants DAA29-84-K-0061, DAA29-84-9-0027 and a grant from AT&T's Telecommunications Program at the University of Pennsylvania.

Modeling Reliable Distributed Real-time Programs

Victor Wolfe, Susan Davidson, Insup Lee
Department of Computer and Information Science
The University Of Pennsylvania
Philadelphia, PA 19104

May 1988

A model for distributed hard real-time programs should incorporate real-time characteristics and be capable of analyzing time-related reliability issues. We introduce a model called the Real-Time Selection/Resolution (RT-S/R) Model with these capabilities and demonstrate it by example.

Requirements For A Model. Hard real-time programs should express the following characteristics:

- Timing constraints including a start time interval, a deadline interval, an expected execution time, and a period.
- Timing constraints on a series of actions.
- Timing constraints on a group of parallel actions.
- Nested timing constraints.
- *Timed atomic actions* which are atomic with respect to timing constraints.
- Action to be taken upon the violation of a timing constraint.
- The dependency of the execution of one process on the timing of a concurrent process.

We wish to model a language construct called a *timed action* which provides for these characteristics in real-time programs. Timed Actions are scheduling blocks (processes) that incorporate computation constraints (timing and integrity) on a block of commands that logically performs a single function in a distributed real-time system. A timed action consists of a header, which lists constraints and scheduling information; a body, which may have nested serial or parallel timed actions; and a handler which handles violations of the constraints listed in the header. The actions may be specified as being atomic with respect to any of the constraints listed in the header. By allowing nested timed actions within the body of a timed action, nested constraints are expressed.

The S/R Model. The S/R model, a finite state machine approach developed by Aggarwal, Kurshan and others [AKS83,ABM86], is a clean, powerful, formally founded tool capable of modeling the concurrent and nested execution of timed actions. It uses labeled graphs to represent the processes of a concurrent system (see Figure 1). Nodes represent states of the computation that the process performs. There is a set of *selections* (italicized in brackets next to the nodes in Figure 1) associated with each node that gives information about the intention of the process to the system. Edges are labeled with expressions of a boolean algebra containing the process selections. Each process non-deterministicly chooses a selection from the set associated with its current node. An edge leading from the current node that results in TRUE when its label is evaluated using the chosen selection is "followed" to determine the new current node. By allowing transitions to depend on selections of other processes, the S/R model reflects a concurrent process's dependency on other processes. The model is formally defined as a calculus allowing the

combination of nested parallel processes to form higher level process [AKS83]. However, the S/R model contains no provisions for modeling timing constraints and analyzing reliable time-related behavior of programs.

Adding Timing Constraints to the S/R Model. The RT-S/R model is an extension of the S/R model with timers associated with each node that keep the relative time elapsed since entering the node. The timers are used in predicates in the labels of edges. A timer is set initially to be zero and is reset and started each time its associated node is visited. After visiting a node, the node's timer is available for use as a condition in transition labels both in the local process and external processes.

A deadline on execution is modeled as a predicate on a transition from the node which tests if the node's timer exceeds the desired deadline. Expected execution time, although not the same as a deadline, is modeled similarly as a predicate on a transition out of a node testing whether the node's timer exceeds the expected execution time constraint. A deadline on the execution of the entire process (or any chain of states) is represented by a transition out of every node in the chain with a predicate testing if the initial node's timer exceeds the deadline. Start time constraints can be modeled by adding an idle state node prior to the initial node of the process. An edge is placed from the idle node to the initial node with a label containing a predicate testing if the start time constraint is satisfied. Timed sporadic activity, where a sporadic process can have a time interval between invocations, is modeled similarly by keeping it in an idle state until it can be invoked again.

To illustrate these real-time concepts we use the example given by Aggarwal in [ABM86] of a quiz situation. In this system there are two processes, a student and a teacher. The teacher gives a problem to the student, the student thinks about the problem and answers. The teacher then marks the student right or wrong and gives another problem.

To make this system real-time, as most quiz situations are, we add the following time constraints:

- The student must give an answer within 5 minutes of receiving a problem and takes at least 1 minute to complete it. The action to be taken upon violation of the five minute constraint is to generate a partial answer.
- Both processes must complete within 10 minutes of starting.
- The teacher has an expected execution constraint of between one and two minutes for giving a problem and takes at least one minute to score it.
- Similarly the student has an expected execution time of between one and two minutes to generate a partial answer.

Figure 1 shows a RT-S/R model of the timed quiz. To model the 5 minute working constraint on the student we use the timer associated with the **work** state to express the timing constraint as a label for a transition, S:Twork > 5min, out of the work state to the **partial** state. The semantics of modeling call for reseting the **work** state timer upon entering the **work** state and remaining in the **work** state until the student answers or the timer runs out. Other timing constraints are similarly modeled.

Constraints on a group of parallel actions can be expressed by nesting them as RT-S/R processes under a higher-level process representing a surrounding timed action. The constraints are placed on the higher-level process and semantics of the RT-S/R model make the nested actions inherit the constraints of the higher-level action. To model periods we also nest the periodic actions in a higher level action which executes the activity. The actions that are to be periodic are modeled as non-periodic processes and nested within a periodic surrounding process that causes their periodic execution. For example, if we wish to make the quiz periodic with a question given every 10 minutes, we can nest the teacher and student processes within a surrounding periodic "quiz" process that provides a period frame for them every 10 minutes.

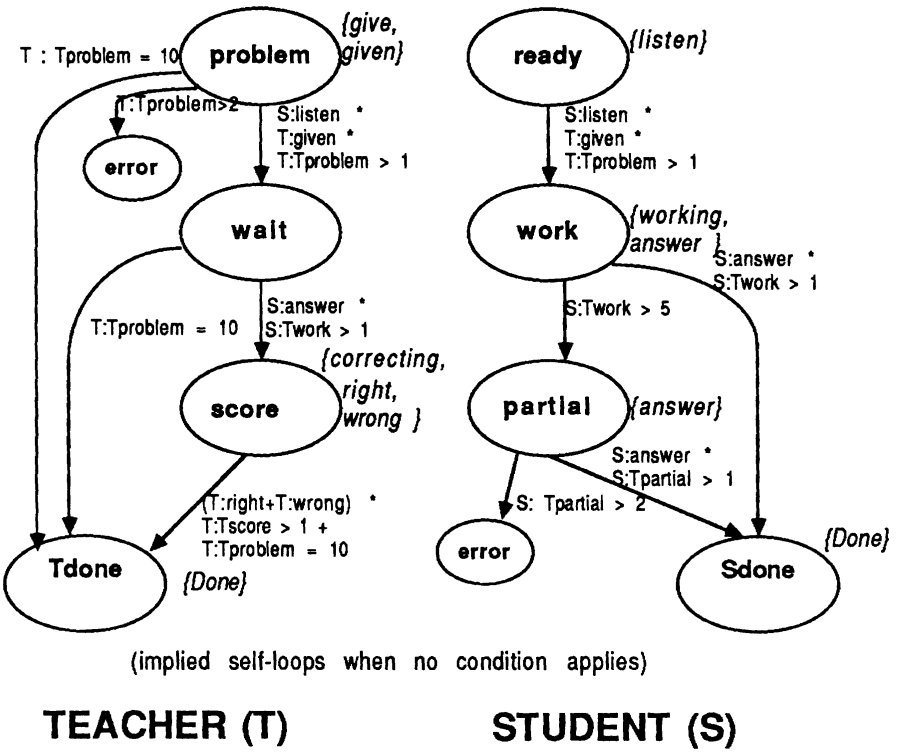

Figure 1: RT-S/R Model of Timed Quiz Problem

Analysis of Time-Related Reliability Properties. A real-time model should have analysis tools that analyze the following time-related reliability properties:

- What states are reachable under the timing constraints?
- Can deadlock occur? In addition to typical deadlock analysis it is desirable to be able to analyze *time deadlock*. Deadlock occurs when the reachability graph has terminal states that are not desired final states. Because timing constraints may eliminate branches of the reachability graph, terminal deadlock states caused by timing may be introduced and should be detected.
- Will desirable results eventually happen (liveness properties), and at what time?
- Are safety properties, including those involving timing constraints, ever violated?

Aggarwal et al. provide for analysis of the S/R model through the SPANNER [ABM86] system. It performs reachability analysis on a global reachability graph formed by combining graphs of individual processes. The reachability graph can be examined for system information such as deadlock, livelock and liveness properties. SPANNER's reachability analysis does not support analysis of timing properties since it is has no notion of timing.

To analyze time-related reliability properties, we augment the SPANNER reachability analysis by determining the interval of possible values for timers in each reachable RT-S/R model state. With these time intervals, time-related questions can be answered by looking for reachable states which have timers within a specified time range. In addition to determining timer intervals, the timed reachability analysis can also detect time deadlock conditions.

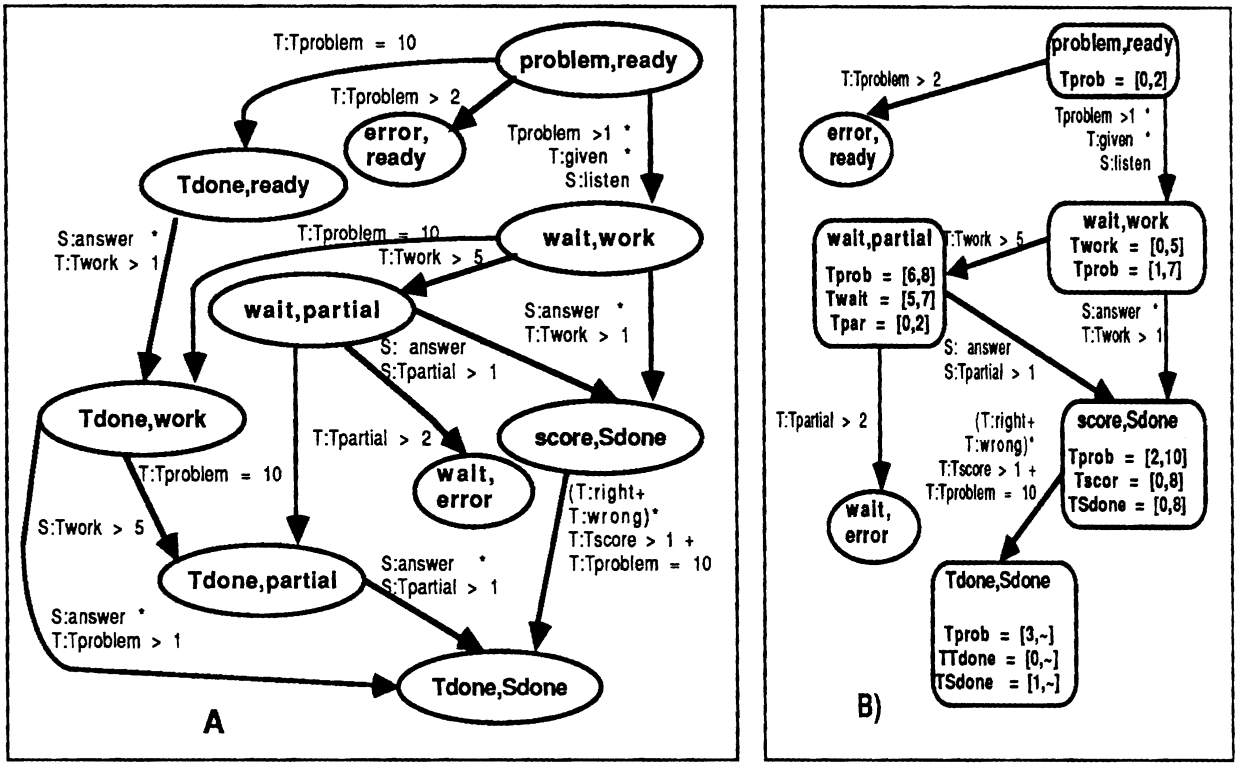

Figure 2: Reachability Graph of Quiz

Figure 2 shows the reachability graphs for the timed quiz without timing incorporated (A); and with timing information (B).[1] Notice that in B) states such as (**Tdone,work**) were eliminated because they were not reachable due to timing. The time intervals provided allow answering time-related reachability questions such as what is the maximum amount of time the teacher can wait? Answer – find all reachable states where teacher is in **wait** and examine the interval for the timer associated with **wait** (7 minutes). Or, what is the minimum amount of time the teacher process takes to execute the entire process? Answer – the minimum of all lower bounds of the teacher's **problem** timer in a reachable state where the teacher is in state **Tdone** (3 minutes).

The main drawback of reachability analysis techniques is that the state explosion for all but simple models can be unmanageable. We are investigating a incremental approach to generating the reachability graph where timing constraints and the use of time intervals reduces the number of reachable states (as it did from part A to part B of Figure 2). It is also possible to take advantage of the hierarchical nature of programs developed with timed actions to reduce complexity of the analysis; levels of the program can analyzed separately and recombined. Other techniques of reducing the reachable state-space, such as probabilistic reduction, are also being investigated.

Current Status. We are currently formalizing the RT-S/R model and designing the augmentation to the SPANNER analysis tool. We are using the RT-S/R model to complete development of the timed action language construct and considering implementation issues.

[1]For simplicity Figure B) only shows some timing intervals. The symbol represents an infinite upper bound. In practice, these processes would be nested within an action whose deadline would determine the value for .

References

[ABM86] Sudhir Aggarwal, D. Barbara, and K.Z. Meth. Spanner - a tool for the specification, analysis, and evaluation of protocols. *IEEE Transactions on Software Engineering*, September 1986.

[AKS83] Sudhir Aggarwal, R.P Kurshan, and K. Sabnani. A calculus for protocol specification and validation. In *IFIP Third International Conference on Protocol Specification, Testing, and Verification*, pages 19–34, May 1983.

[Das85] B. Dasarthy. Timing constraints of real-time systems: constructs for expressing them, methods of validating them. *IEEE Transactions on Software Engineering*, SE-11(1):80–85, January 1985.