A Benchmark for Comparing Different Approaches for Specifying and Verifying Real-Time Systems

C. L. Heitmeyer and B. G. Labaw
Naval Research Laboratory

R. D. Jeffords
Locus, Inc.

1 Introduction

To be considered correct or useful, *real-time* systems must deliver results within specified time intervals, either without exception or with high probability. Recently, a large number of formal methods have been invented for specifying and verifying real-time systems. It has been suggested (see, e.g., [11]) that these formal methods need to be tested out on actual real-time systems. Such testing will allow the scalability of the methods to be assessed and will also uncover new problems requiring formal solution.

However, before these methods can be applied productively to industrial systems, greater understanding is needed of how they compare—e.g., what classes of problems they are designed to solve, the availability of mechanical support, etc. To provide insight into the utility of different methods for solving real-time problems, we have developed a generic version of a real-time railroad crossing system. Our plan is to use this example as a benchmark for comparing different formalisms. In this paper, we define the problem, describe three classes of formalisms that can be applied, and summarize efforts currently in progress to specify the system of interest and prove properties about its behavior.

2 Generic Railroad Crossing Problem

2.1 Background

The original example, developed by Leveson to illustrate her software safety techniques, involves a system operating a gate at a railroad crossing [9]. The system must satisfy a safety property. The purpose of the safety property is to ensure that the system cannot enter an unsafe state—i.e., a state in which a train is in the crossing but the crossing gate is not down. In 1988, Jahanian and Stuart published a real-time version of this problem to demonstrate their method for verifying safety properties about specifications in a graphical language called Modechart [8]. To make the problem somewhat more realistic, we have generalized it. While the Jahanian-Stuart version describes a system with a single track and at most two trains in the region of interest both traveling in the same direction, our version allows several tracks and an unspecified number of trains traveling in both directions. In addition to the safety property, our version includes a *utility property*. The purpose of the utility property is to avoid a degenerate solution, e.g., one that lowers the gate and keeps it lowered. Safety-critical systems must not only operate safely. To be useful, they must perform certain functions within specified time intervals. That is, they must exhibit bounded liveness.

2.2 Problem Statement

The system to be developed operates a gate at a railroad crossing. The railroad crossing I lies in a region of interest R, i.e., $I \subset R$. A set of trains travel through R on multiple tracks in both directions. A sensor system determines when each train enters and exits region R. To describe the system formally, we define a gate function $g(t) \in [0, 90]$, where $g(t) = 0$ means the gate is down and $g(t) = 90$ means the gate is up. We also define a set $\{\lambda_i\}$ of *occupancy intervals*, where each occupancy interval is a time interval during which one or more trains are in I. The ith occupancy interval is represented as $\lambda_i = [\tau_i, \nu_i]$, where τ_i is the time of the ith entry of a train into the crossing when no other train is in the crossing and ν_i is the first time since τ_i that no train is in the crossing (i.e., the train that entered at τ_i has exited as have any trains that entered the crossing after τ_i).

Given two constants ξ_1 and ξ_2, $\xi_1 > 0$, $\xi_2 > 0$, the problem is to develop a system to operate the crossing gate that satisfies the following two properties:

Safety Property	$t \in \cup_i \lambda_i \Rightarrow g(t) = 0$	(The gate is down during all occupancy intervals)
Utility Property	$t \notin \cup_i [\tau_i - \xi_1, \nu_i + \xi_2] \Rightarrow g(t) = 90$	(The gate is up as often as possible)

3 Formal Approaches

Several formalisms are available to specify the system described above and to reason about its properties. These formalisms fall into three classes:

- **General-Purpose Theorem Provers** (e.g., Boyer-Moore [1], EVES [10], EHDM [12], PVS [13], HOL [7]),
- **Model Checkers** (e.g., Clarke's CTL [2], the Modechart verifier [14]), and
- **Process Algebras** (e.g., CSR [6], Cleaveland's Concurrency Workbench [3], and CSP [5]).

We note that verification tools based on the two latter approaches, model checking and process algebras, are highly specialized and provides verification with little human intervention. In contrast, a proof generated with a general-purpose theorem prover usually requires considerable human guidance.

Efforts are currently in progress to apply one or more examples of each class to the railroad crossing problem. ORA Canada has developed a solution and completed proofs of both the safety and utility properties using their general-purpose theorem prover, EVES. SRI has developed a proof of the safety property and is working on a proof of the utility property using their newly developed theorem prover, PVS. Solutions using CSP and an approach based on Modechart specifications are in progress at NRL. Bill Roscoe of Oxford developed the original CSP specification of the problem and is part of a team that built a tool named FDR (Failure Divergence Refinement) [5] to automatically check that CSP specifications satisfy certain properties. To develop insight into the styles of specification and verification that are most natural for a given formalism, the preceding efforts are, to the extent feasible, proceeding independently.

4 Summary

Once the specifications and proofs are complete, several criteria can be applied to compare the formalisms [4]. Among these criteria are

- How easy is it to reason about time using the formalism? How understandable are specifications and proofs in the formalism?
- For what classes of timing properties is the formalism suitable?
- What is the quality of the mechanical tools available to support the formalism?
- Is the formal method better suited to some application domains than to others?
- How general is the formalism? Is it designed to specify and verify only timing properties? What other properties can be specified and verified using the formalism?
- Is the formalism more suited to a particular phase of software development than to others, e.g., requirements rather than detailed design?
- Does the formalism handle continuous as well as discrete time?

Addressing these issues should help determine what each formalism's strengths are and how the formalism can be used productively to develop industrial-strength real-time systems.

References

[1] R. S. Boyer and J. S. Moore, *A Computational Logic Handbook,* Boston, MA, 1988.

[2] E. M. Clarke, E. Emerson, A. Sistla, "Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications," *ACM Trans. on Prog. Lang. and Systems,* Vol. 8, No. 2, April, 1986.

[3] R. Cleaveland, "The Concurrency Workbench: A Semantics-Based Tool for the Verification of Concurrent Systems," Tech. Rpt. TR-91-26, North Carolina State Univ., Raleigh, NC, 1991.

[4] P. C. Clements, C. E. Gasarch, R. D. Jeffords, "Evaluation Criteria for Real-Time Specification Languages," Rpt. 6935, Naval Research Lab., Wash, DC, 1992.

[5] "Failure Divergence Refinement, User Manual and Tutorial," Version 1.2, Formal Systems (Europe) Ltd., Oxford, UK, 3 December 1992.

[6] R. Gerber and I. Lee, "Communicating Shared Resources," A Model for Distributed Real-Time Systems," *Proc., Real-Time Systems Symposium,* Santa Monica, CA, Dec. 1989, 68-78.

[7] M. Gordon, "Mechanizing Programming Logics in Higher Order Logic," Tech. Rpt. 145, University of Cambridge, Sept. 1988.

[8] F. Jahanian and D. A. Stuart, "A Method For Verifying Properties of Modechart Specifications," *Proc., Real-Time Systems Symposium,* Huntsville, AL, 6-8 Dec. 1988.

[9] N. G. Leveson and J. L. Stolzy, "Analyzing Safety and Fault Tolerance Using Time Petri Nets, " *TAPSOFT: Joint Conf. on Theory and Practice of Software Development,* Springer-Verlag, March 1985.

[10] Odyssey Research Associates, "Introduction to EVES: Exercises and Notes," Ottawa, CAN, January, 1992.

[11] J. S. Ostroff, "Survey of Formal Methods for the Specification and Design of Real-Time Systems," *Tutorial on Specification of Time*, 1991 (to appear).

[12] J. Rushby, F. von Henke, and S. Owre, "An Introduction to Formal Specification and Verification Using EHDM," Tech. report SRI-CSL-91-2, SRI International, Menlo Park, CA, 1991.

[13] N. Shankar, "Verification of Real-Time Systems Using PVS," SRI International, Menlo Park, CA, 1993 (to appear).

[14] D. A. Stuart, "Implementing a Verifier for Real-Time Systems," *Proc., Real-Time Systems Symposium,* Orlando, FL, Dec. 1990, 62-71.