An Extended Duration Calculus for Hybrid Real-Time Systems

Zhou Chaochen\textsuperscript{1} Anders P. Ravn\textsuperscript{2}, and Michael R. Hansen\textsuperscript{3}

\textsuperscript{1} UNU/IIST
Apartado 3058, Macau
E-mail: zcc@iist.unu.edu and zcc@unuiist.attmail.com
\textsuperscript{2} Department of Computer Science
Technical University of Denmark
E-mail: apr@id.dth.dk
\textsuperscript{3} Fachbereich Informatik
Universität Oldenburg, Germany
E-mail: michael.hansen@informatik.uni-oldenburg.de.
On leave from: Department of Computer Science
Technical University of Denmark

Abstract. Duration Calculus is a real-time interval logic which can be used to specify and reason about timing and logical constraints on discrete states in a dynamic system. It has been used to specify and verify designs for a number of real-time systems. This paper extends the Duration Calculus with notations to capture properties of piecewise continuous states. This is useful for reasoning about hybrid systems with a mixture of continuous and discrete states. The proof theory of Duration Calculus is extended such that results proven using mathematical analysis can be used freely in the logic. This provides a flexible interface to conventional control theory.

Keywords: requirements, software design, hybrid systems, real-time systems, control theory, interval logic, durations.

1 Introduction

A central step in development of software for real-time, embedded systems is to capture the requirements for a program from the wider requirements for the system of which the program is a component. The Duration Calculus \cite{16} introduces a notation for that task. It has already been used to specify requirements and designs for a number of real-time systems, including an on-off gas-burner \cite{1, 13}, a railway crossing \cite{14}, and in an early paper \cite{12} a simple auto pilot.

Duration Calculus is more than a specification notation, it is a calculus that allows the designer of a system to verify a design with respect to requirements through a deduction. Duration Calculus is fundamentally an interval logic for reasoning about timing and logical properties of systems modelled by discrete

\* Partially funded by \textbf{ProCoS} ESPRIT BRA 7071, by the Danish Technical Research Council \textbf{RAPID}, and by the Danish Natural Science Research Council
(Boolean) states. This is not sufficient when dealing with hybrid systems, where states can be both discrete and continuous. Already the auto pilot example, mentioned above, required certain extensions to give an interface to differential equations. The purpose of this paper is to develop a fuller theory of this kind with a simple interface to mathematical analysis. Such an interface permits development of a hybrid system using both control theory and Duration Calculus within a consistent formal framework. Mathematical analysis or its specialisation: control theory, is used where it is most suitable, e.g. to analyse continuous parts of the systems, while the Duration Calculus is used to analyse critical durations and discrete progress properties of the system.

The following section gives a short introduction to the Duration Calculus (DC) by means of a simple example and motivates the need for an interface to control theory. Section 3 defines the syntax and semantics of the Extended Duration Calculus (EDC), while section 4 develops the proof theory. Section 5 illustrates the use of the calculus on larger examples. Finally in section 6 there is a summary and a comparison to other work.

2 Controlling a valve

This section introduces the concepts of embedded real-time systems and illustrates the use of DC to specify and reason about properties of such systems. The description is based on a very simple example of a valve, which controls the gas supply to a burner. We shall first take a typical programmer's view of the valve as a Boolean variable which can be in only one of two states. Later on we shall take the more detailed view of a control engineer and see it as having a state varying continuously between 0 (off) and 1 (on). The challenge is now to ensure that the two views are consistent, and to be able to deduce properties of the abstract representation from properties of the concrete representation.

2.1 Embedded real-time computing systems

An embedded computing system contains a program placed in one or more digital computers, which are connected through channels to sensors and actuators that measure or change quantities in a piece of equipment. The equipment is a physical system evolving over time and used in a certain environment for a definite purpose (Figure 1).

When it is required that the computer reacts within specific time bounds, the system is called a (hard) real-time system. The real-time constraints usually reflect that the physical process state defined by the equipment should not deviate too much from an internal state of a control program run by the computer.

The concept of an embedded, real-time system is very broad, and instead of attempting a general definition we illustrate it by the simple example of a valve, which is an actuator. It is used in a gas burner (a piece of equipment), and can influence the environment by leaking gas.

A valve can conveniently be modelled as a function of time. Since we are essentially dealing with physical processes, the time domain, Time, is taken to