A Specification-Oriented Semantics for Asynchronous Real-Time Programming

Alvaro E. Arenas

Universidad Autónoma de Bucaramanga, Laboratorio de Computo Especializado
Calle 48 No 39 - 234, Bucaramanga, Colombia
aearenas@bumanga.unab.edu.co

Abstract

The construction of computational models and the derivation of their algebraic properties have proven to be a great aid in software development and in the design and implementation of general-purpose programming languages. This paper presents a model for a real-time language, in which processes interact asynchronously via communication queues and derives its main algebraic laws. The paper applies Hoares's Unifying Theories of Programming [6] to study concepts such as real-time and asynchronous communication.

Keywords: Formal semantics, real-time systems, asynchronous communication.

Resumen

La construcción de modelos computacionales y la derivación de sus propiedades algebraicas ha sido de gran ayuda en el desarrollo de software y en el diseño e implantación de lenguajes de programación. Este artículo presenta un modelo para lenguajes de tiempo real, cuyos procesos se comunican asincrónicamente vía colas de comunicación, y deriva las leyes algebraicas del lenguaje. El trabajo aplica la Unificación de Teorías de Programación de Hoare [6] en el estudio de sistemas de tiempo real y comunicación asincrónica.

Palabras claves: Semántica formal, sistemas de tiempo real, comunicación asincrónica.

1 Introduction

This paper presents a computational model and algebraic laws for a real-time language, in which processes communicate asynchronously via time-stamped data areas called shunts. A shunt can be seen as a directed channel with the capability of buffering messages. When the sender transmits a message, it is time-stamped by the run-time environment and deposited into the corresponding shunt; the sender then proceeds with its execution. When the receiver reads a shunt, it takes the oldest message deposited in it. However, if the shunt is empty, the receiver is blocked until a message arrives. The main advantage of this asynchronous mechanism is the loose coupling it provides between system parts: a sender is never blocked because a receiver is not ready to communicate. This communication scheme is adopted by several asynchronous models such as versions of CSP [8], SDL [11], or the ObjectTime language ROOM [14] among others.

We follow the approach suggested by Hoare in [6] for the design of programming languages. The approach starts by identifying the observations that can be made of a process. Then, a particular process is modelled as the subset of observations to which it can give rise, which it is usually represented by the strongest predicate describing those observations. As a consequence, algebraic laws of the program operators can be derived using the predicate calculus. We have derived laws that coincides with classical laws for concurrent and real-time languages and new laws that characterise the behaviour of crucial operators such as time-out.

The following section introduces the observations of real-time processes. Section