A Formal Framework for ASTRAL Inter-level Proof Obligations

Alberto Coen-Porisini  
Dipartimento di Elettronica e Informazione  
Politecnico di Milano, Milano 20133, Italia  

Richard A. Kemmerer  
Reliable Software Group - Department of Computer Science  
University of California, Santa Barbara, CA 93106  

Dino Mandrioli  
Dipartimento di Elettronica e Informazione  
Politecnico di Milano, Milano 20133, Italia

Abstract. ASTRAL is a formal specification language for realtime systems. It is intended to support formal software development, and therefore has been formally defined. ASTRAL is provided with structuring mechanisms that allow one to build modularized specifications of complex systems with layering. A realtime system is modeled by a collection of process specifications and a single global specification. Each process specification consists of a sequence of levels; each level is an abstract data type view of the process being specified. In this paper further details of the ASTRAL refinement process, which were not fully developed in previous papers, are presented. ASTRAL also supports formal proofs of specification correctness. Formal proofs in ASTRAL can be divided into two categories: inter-level proofs and intra-level proofs. The former deal with proving that the specification of level i+1 is consistent with the specification of level i, while the latter deal with proving that the specification of level i is consistent and satisfies the stated critical requirements. This paper concentrates on inter-level proofs. The necessary proof obligations to assure that a refinement is a correct implementation are presented. The approach is illustrated through a communication example.

1. Introduction

ASTRAL is a formal specification language for realtime systems. It is intended to support formal software development, and therefore has been formally defined. [GK 91a] discusses the rationale of ASTRAL's design and demonstrates how the language builds on previous language experiments, and [CKM 94] gives further details of the ASTRAL environment and critical requirement components. [GK 91b] discusses how ASTRAL specifications can be formally analyzed by translating them into TRIO and then using the TRIO validation theory.

§ This research was partially supported by the National Science Foundation under grant CCR-9204249
Recently, a number of approaches have been proposed to build formal proofs for realtime systems [ACD 90, FMM 94, GF 91, HL 94, JM 94, LRD 95, Ost 89, Suz 90]. In most cases, they are based on low level formalisms, i.e., abstract machines and/or assertion languages that are not provided with modularization and abstraction mechanisms. As a consequence, the proofs lack structure, which makes them unsuitable for dealing with complex real-life systems.

On the contrary, ASTRAL is provided with structuring mechanisms that allow one to build modularized specifications of complex systems with layering [GK 91a, GK 91b]. In this paper further details of the ASTRAL refinement process, which were not fully developed in previous papers, are presented.

A distinguishing feature of ASTRAL is that formal proofs can be divided into two categories: inter-level proofs and intra-level proofs. The former deal with proving that the specification of level i+1 is consistent with the specification of level i, while the latter deal with proving that the specification of level i is consistent and satisfies the stated critical requirements. The intra-level proofs are discussed in detail in [CKM 94]. This paper concentrates on inter-level proofs.

In the next section a brief overview of ASTRAL is presented. Section 3 discusses layering and the implementation statement. Section 4 presents the proof obligations that must be carried out to prove that a lower level implementation is a correct refinement of the upper level; that is, such proofs guarantee that if the critical requirements are satisfied in the upper level specification, then they will be satisfied by the lower level specification. Section 5 presents an example of a two level ASTRAL specification along with a proof of the inter-level proof obligations for the specification. Finally, in section 6 some conclusions from this research are presented and possible future directions are proposed.

2. Overview of ASTRAL

ASTRAL uses a state machine process model and has types, variables, constants, transitions, and invariants. A realtime system is modeled by a collection of state machine specifications and a single global specification. Each state machine specification represents a process type of which there may be multiple instances in the realtime system. State variables and transitions may be explicitly exported by a process; this makes the variable values readable by other processes and the transitions executable by the external environment (exported transitions cannot be executed by another process). Interprocess communication is via the exported variables, and is accomplished by referencing the value of an exported variable for a particular instance of the process. A process can reference the value of any exported variable of a process type or the start or end time of an exported transition. Start(Op, t) is a specification predicate that is true if and only if transition Op starts at time t and there is no other time after t and before the current time when Op starts (i.e., t is the time of the last occurrence of Op). For simplicity, the functional notation Start(Op) is adopted as a shorthand for "time t such that Start(Op, t)" whenever the quantification of the variable t (whether existential or universal) is clear from the context. Start-k(Op) is used to give the start time of the kth previous occurrence of Op. The end time of a transition Op may be specified similarly using End(Op) and End-k(Op).

The ASTRAL computation model views the values of all variables being modified by a transition as being changed by the transition in a single atomic action that occurs