Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 280 (2011) 23–33

www.elsevier.com/locate/entcs

Event B Development of a Synchronous AADL Scheduler

Jean-Paul Bodeveix and Mamoun Filali

IRIT - Universit de Toulouse, France

Abstract

This paper presents the modelisation of the semantics of a subset of the architecture description language AADL using Event-B. Elements of the semantics of the considered subset are gradually introduced in order to make possible the traceability of the formal text against the informal specification. Starting from a very general computational model, we incrementally add elements of AADL by constraining or instantiating it and finally introduce a family of schedulers. The Rodin platform is used to prove the correctness of this development.

Keywords: refinement, scheduling, architecture description languages

1 Introduction

The Architecture Analysis and Description Language (AADL) [5] is the result of a long experience in the design of modeling languages targeting the development of safety critical real-time systems. Compared to most modeling languages, AADL has a rather precise semantics and has been designed to allow the early analysis of real-time systems. In order to show the correctness of analysis or code generation tools, it is necessary to define a formal semantics of AADL. In this paper, we presents the modelisation of the semantics of a subset of AADL using Event-B [2]. Similar works have already been done, each time using translational semantics targeting a specific formal language (Fiacre [3], BIP (Behavior, Interaction, Priority)[7] [4], Signal/Polychrony[6]), . . .). Here, we propose a different approach which mainly aims at explaining the formal model by using a refinement-based development. Elements of the semantics of the considered subset are gradually introduced in order to make possible the traceability of the formal text against the informal specification. Then, an abstract scheduler is introduced as a refinement and proved to be correct.

The paper is organized as follows. Section 2 introduces the AADL language and defines the subset we are interested in. Section 3 presents the Event-B development of the semantics of this subset. Section 4 draws some conclusions.

doi:10.1016/j.entcs.2011.11.015

2 AADL

AADL (Architecture and Analysis Description Language) [5] is a language standardized by the SAE for modeling real-time critical embedded systems. It allows the early analysis of real time systems. An AADL model defines the dynamic architecture of the considered system as a set of communicating threads. Threads have real-time properties (dispatch protocol, period, relative deadline, WCET[1], etc.). They communicate through ports or shared data using several communication protocols. The main goal of AADL is to allow to check that the hardware architecture which is made of processors, memories, buses and devices is well suited to the software architecture. Among the verified considered properties, we can cite schedulability (tasks complete before their deadline), response time, bus load, etc. In order to make such analysis possible, the execution model of AADL is precisely defined. Here, we consider a small *synchronous* subset which consists of threads periodically dispatched, each having its own period. They only communicate through data ports. The AADL execution model defines the semantics of this subset. It can be summarized by the following points:

- At dispatch, a thread reads its input ports
- Dispatched thread execute (i.e. access to the processor) during at most their WCET, until completion, under the control of a scheduler.
- Completion must occur before deadline for the system to be schedulable.
- At completion, a thread writes its results computed from its input ports to its output ports. In the following, we suppose each thread has only one output port.
- If two threads linked by immediate connections are dispatched simultaneously, the sender's completion occurs before the receiver's start of execution, immediate links transfer data at that time and the corresponding input ports are read again by the thread, thus refreshing the value obtained at dispatch.
- The graph of nodes connected by immediate links should be acyclic.
- Delayed and unsynchronized immediate connectors read output ports at the sender deadline and update input ports at receiver dispatch. The update causally precedes the simultaneous reading of input ports.

These rules ensure deterministic executions whatever is the scheduling policy, as soon as deadlines are not missed. Models, which can scheduled, behave as if execution and communication where instantaneous. In such a way, this subset of AADL can be seen as a verifiable implementation of a synchronous model. In the following, we show how to derive by successive refinements an implementation satisfying the previous requirements. Refinement steps introduce elements of the specification that are preserved during the development.

[1] Worst case execution time