Modelisation of Timed Automata in Coq

Christine Paulin-Mohring

Université Paris-Sud, Laboratoire de Recherche en Informatique,
UMR 8623 CNRS, Bâtiment 490, F-91405 Orsay Cedex, France,
paulin@lri.fr, http://www.lri.fr/~paulin

Abstract. This paper presents the modelisation of a special class of
timed automata, named p-automata in the proof assistant Coq. This
work was performed in the framework of the CALIFE project[1] which
aims to build a general platform for specification, validation and test of
critical algorithms involved in telecommunications. This paper does not
contain new theoretical results but explains how to combine and adapt
known techniques in order to build an environment dedicated to a class
of problems. It emphasizes the specific features of Coq which have been
used, in particular dependent types and tactics based on computational
reflection.

1 Introduction

An important research area is the design of methods and tools for simplifying
the design of zero-default programs. Formal methods are mainly used in the
context of specific algorithms involved in critical systems. The behavior of the
program and its expected specification are represented in a mathematical model
and then a proof of correctness has to be produced. This task represents often a
lot of work but it cannot be avoided in order to gain a high-level of confidence
in the appropriate behavior of the program. Formal models are indeed required
for getting the highest level of certification for systems manipulating critical
informations like smart-cards. The generalization of formal methods requires to
provide appropriate computer tools and specialized libraries for simplifying the
task of development of new algorithms.

1.1 Theorem Proving vs Model-Checking

Theorem proving and model-checking are two different formal methods for the
validation of critical systems. Model-checking tries to automatically prove prop-
erties of programs represented as labelled transition systems. A specification on
transition systems is often expressed as a temporal logic formula that is checked
using exploration techniques on the graph representing the transition system.
There are well-known problems when using model-checking: the model of the
program has to be simplified in order to be treated automatically, then it be-
comes more difficult or impossible to insure that what is proven corresponds

1 http://www.loria.fr/Calife
to the real world. Because they are complex programs, model-checkers are not bug-free. Model-checkers are appropriate tools for the automatic detection of errors. On the other hand, if the program correctness property is expressed as a mathematical statement, it is possible to handle it using general tools for theorem proving. The interest of this method is to be able to represent complicated systems involving non-finite or non-specified data or continuous time. However, for such theories, the complete automation of proofs is hopeless and then the work has to performed by hand.

During the last years, there has been several attempts to combine these two methods. One possibility is to use a theorem prover to reduce an abstract problem into one that can be handle by a model-checker used as an external tool. It is possible to trust the model-checker like in the system PVS [19] or to be more suspicious and keep a trace of theorems which depend on properties justified by a non-certified procedure like in HOL [13]. Another attempt is to integrate model-checkers to theorem provers in a certified way [24,26,22] but it raises the problem of efficiency. Theorem proving can also be used in order to prove that a property is an invariant of the labelled transition system which models the system [12]. Interactive theorem prover based on higher-order logic are also appropriate tools for mechanizing meta-reasoning on the sophisticated formalisms involved in the modelisation of concurrent systems like the π-calculus [15,16].

1.2 Studying a Conformance Protocol

In this paper, we are considering the interaction between theorem proving and automatic proof-procedures in the context of the study of a conformance protocol (named ABR) normalized by France Telecom R&D to be used in ATM networks. This example has been widely studied since it was introduced as an industrial challenge for formal methods. Proofs of this protocol have been both done using theorem provers and automatic tools. A recent synthetic presentation of these experiments can be found in [4].

The purpose of the conformance algorithm is to control the bit rate available to the users in an ATM network. The system receives information from the network on the maximum rate available at time $t$, it also receives the data from the user and checks that the actually used rate does not exceed the available rate. But there are delays in the transmission, such that the information sent by the network at time $t$ will only be known to the user after time $t + \tau_3$ and no later than $t + \tau_2$. The general principle is that the user is always allowed to choose the best rate available to him or her. Assume a maximum rate $R_1$ arrives at time $t_1$, the user should wait until $t_1 + \tau_3$ to take this rate into account. Assume $R_1$ is less than the rate available before arrival of $R_1$, the user may wait until $t_1 + \tau_2$ to conform to the $R_1$ rate. If a new better rate $R_2 > R_1$ arrive at time $t_2$ such that $t_2 + \tau_3 < t_1 + \tau_2$, the user will be allowed to ignore $R_1$ and adopt the rate $R_2$ already at time $t_2 + \tau_3$. The algorithm has to program a schedule of maximum available rates. But because it has only a small amount of memory, the algorithm only computes an approximation of the schedule: it stores the currently available rate and at most two other rates with the corresponding time. The verification