Science of Computer Programming 123 (2016) 2–41

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Runtime enforcement of regular timed properties by suppressing and delaying events

Yliès Falcone [a,*], Thierry Jéron [b], Hervé Marchand [b], Srinivas Pinisetty [c]

[a] Univ. Grenoble Alpes, INRIA, LIG, F-38000 Grenoble, France
[b] INRIA Rennes – Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes Cedex, France
[c] Aalto University, Finland

ARTICLE INFO

Article history:
Received 2 September 2014
Received in revised form 26 January 2016
Accepted 26 February 2016
Available online 4 March 2016

Keywords:
Verification
Monitoring
Runtime enforcement
Timed specifications

ABSTRACT

Runtime enforcement is a verification/validation technique aiming at correcting possibly incorrect executions of a system of interest. In this paper, we consider enforcement monitoring for systems where the physical time elapsing between actions matters. Executions are thus modelled as timed words (i.e., sequences of actions with dates). We consider runtime enforcement for timed specifications modelled as timed automata. Our enforcement mechanisms have the power of both delaying events to match timing constraints, and suppressing events when no delaying is appropriate, thus possibly allowing for longer executions. To ease their design and their correctness-proof, enforcement mechanisms are described at several levels: enforcement functions that specify the input–output behaviour in terms of transformations of timed words, constraints that should be satisfied by such functions, enforcement monitors that describe the operational behaviour of enforcement functions, and enforcement algorithms that describe the implementation of enforcement monitors. The feasibility of enforcement monitoring for timed properties is validated by prototyping the synthesis of enforcement monitors from timed automata.

1. Introduction

Runtime enforcement [1–5] is a verification and validation technique aiming at correcting possibly-incorrect executions of a system of interest. In traditional (untimed) approaches, the enforcement mechanism is a *monitor* modelled as a transducer that inputs, corrects, and outputs a sequence of events. How a monitor transforms the input sequence is done according to a specification of correct sequences, formalised as a property. Moreover, a monitor should satisfy some requirements: it should be *sound* in the sense that only (prefixes of) correct sequences are output; it should also be *transparent* meaning that the output sequence preserves some relation with the input sequence, depending on the authorised operations.

Runtime enforcement monitors can be used in various application domains. For instance, enforcement monitors can be used for the design of firewalls, to verify the control-flow integrity and memory access of low-level code [6], or implemented in security kernels or virtual machines to protect the access to sensitive system resources (e.g., [7]). In [8], we discuss some other uses of enforcement monitors such as resource allocation and the implementation of robust mail servers.

* Corresponding author.
E-mail addresses: ylies.falcone@imag.fr (Y. Falcone), thierry.jeron@inria.fr (T. Jéron), herve.marchand@inria.fr (H. Marchand), srinivas.pinisetty@aalto.fi (S. Pinisetty).

http://dx.doi.org/10.1016/j.scico.2016.02.008
0167-6423/

In this paper, we consider *runtime enforcement of timed properties*, initially introduced in [5,9]. In timed properties (over finite sequences), not only the order of events matters, but also their occurrence dates affect the satisfaction of the property. It turns out that considering time constraints when specifying the behaviour of systems brings some expressiveness that can be particularly useful in some application domains when, for instance, specifying the usage of resources. In Section 2, we present some running and motivating examples of timed specifications related to the access of resources by processes. We shall see that, in contrast to the untimed case, the amount of time an event is stored influences the satisfaction of properties.

In [5], we propose preliminary enforcement mechanisms restricted to safety and co-safety timed properties. Safety and co-safety properties allow to express that "something bad should never happen" and that "something good should happen within a finite amount of time", respectively. In [9], we generalise and extend the initial approach of [5] to the whole class of timed regular properties. Indeed, some regular properties may express interesting behaviours of systems belonging to a larger class that allows to specify some form of transactional behaviour. Regular properties are, in general, neither prefix nor extension closed, meaning that the evaluation of an input sequence w.r.t. the property also depends on its possible future continuations. For instance, an incorrect input sequence alone may not be correctable by an enforcement mechanism, but the reception of some events in the future may allow some correction. Hence, the difficulty that arises is that the enforcement mechanism should take conservative decisions and change its behaviour over time taking into account the evaluation (w.r.t. the property) of the current input sequence and its possible continuations. Roughly speaking, in [5,9], enforcement mechanisms receive sequences of events composed of actions and delays between them, and can only increase those delays to satisfy the desired timed property; while in this paper, we consider absolute dates and allow to reduce delays between events (as described in detail in the following paragraph).

Contributions In this paper, we extend [9] in several directions. The main extension consists in increasing the power of enforcement mechanisms by allowing them to suppress input events, when the monitor determines that it is not possible to correct the input sequence, whatever is its continuation. Consequently, enforcement mechanisms can continue operating, and outputting events, while in our previous approaches the output would have been blocked forever. This feature and other considerations also drove us to revisit and simplify the formalisation of enforcement mechanisms. We now consider events composed of actions and absolute dates, and enforcement mechanisms are *time retardant with suppression* in the following sense: monitors should keep the same order of the actions that are not suppressed, and are allowed to increase the absolute dates of actions in order to satisfy timing constraints. Note, this allows to decrease delays between actions, while it is not allowed in [5,9]. As in [5,9], we specify the mechanisms at several levels, but in a revised and simplified manner: the notion of enforcement function describes the behaviour of an enforcement mechanism at an abstract level as an input–output relation between timed words; requested properties of these functions are formalised as soundness, transparency, optimality, and additional physical constraints[1]; we design adequate enforcement functions and prove that they satisfy those properties; the operational behaviour of enforcement functions is described as enforcement monitors, and it is proved that those monitors correctly implement the enforcement functions; finally enforcement algorithms describe the implementation of enforcement monitors and serve to guide the concrete implementation of enforcement mechanisms. Interestingly, although all untimed regular properties over finite sequences can be enforced [10], some enforcement limitations arise for timed properties (over finite sequences). Indeed, we show that storing events in the timed setting influences the output of enforcement mechanisms. In particular, because of physical time, an enforcement mechanism might not be able to output certain correct input sequences. Finally, we propose an implementation of the enforcement mechanisms for all regular properties specified by one-clock timed automata (while [5,9] feature an implementation for safety and co-safety properties only).

Paper organisation The rest of this paper is organised as follows. In Section 2, we introduce some motivating and running examples for the enforcement monitoring of timed properties, and illustrate the behaviour of enforcement mechanisms and the enforceability issues that arise. Section 3 introduces some preliminaries and notations. Section 4 recalls timed automata. Section 5 introduces our enforcement monitoring framework and specifies the constraints that should be satisfied by enforcement mechanisms. Section 6 defines enforcement functions as functional descriptions of enforcement mechanisms. Section 7 defines enforcement monitors as operational description of enforcement mechanisms in the form of transition systems. Section 8 proposes algorithms that effectively implement enforcement monitors. In Section 9, we present an implementation of enforcement mechanism in Python and evaluate the performance of synthesised enforcement mechanisms. In Section 10, we discuss related work. In Section 11, we draw conclusions and open perspectives. Finally, to ease the reading of this article, some proofs are sketched and their complete versions can be found in Appendix A.

2. General principles and motivating examples

In this section, we describe the general principles of enforcement monitoring of timed properties, and illustrate the expected input/output behaviour of enforcement mechanisms on several examples.

[1] The two latter constraints are specific to runtime enforcement of timed properties.