Theoretical Computer Science 643 (2016) 38–64

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

A temporal logic for micro- and macro-step-based real-time systems: Foundations and applications ☆

Matteo Rossi *, Dino Mandrioli, Angelo Morzenti, Luca Ferrucci

Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133, Milano, Italy

ARTICLE INFO

Article history:
Received 22 June 2015
Received in revised form 16 May 2016
Accepted 27 June 2016
Available online 7 July 2016
Communicated by P. Aziz Abdulla

Keywords:
Metric temporal logic
Formal and automatic verification
Micro- and macro-steps
Non-standard analysis
Petri nets
Stateflow/Simulink

ABSTRACT

Many systems include components interacting with each other that evolve at possibly very different speeds. To deal with this situation many formal models adopt the abstraction of "zero-time transitions", which do not consume time. These, however, have several drawbacks in terms of naturalness and logic consistency, as a system is modeled to be in different states at the same time. We propose a novel approach that exploits concepts from non-standard analysis and pairs them with the traditional "next" operator of temporal logic to introduce a notion of micro- and macro-steps; our approach is enacted in an extension of the TRIO metric temporal logic, called X-TRIO. We study the expressiveness and decidability properties of the new logic. Decidability is achieved through translation of a meaningful subset of X-TRIO into Linear Temporal Logic, a traditional way to support automated verification. We illustrate the usefulness and the generality of our approach by applying it to provide a formal semantics of timed Petri nets, which allows for their automated verification. We also give an overview of a formal semantics of Stateflow/Simulink diagrams, defined in terms of X-TRIO, which has been applied to the automated verification of a robotic cell.

1. Introduction

Modern complex time-critical systems often require sophisticated time modeling approaches to support specification and verification of their properties. Traditional approaches to modeling time-dependent system behavior are roughly categorized into continuous and discrete ones: in the former case both system state and the time variable range over a continuous domain—e.g., the reals—and system evolution is formalized by making the state a continuous function of the independent variable "time"; in the latter case both time and the system state range over a discrete domain—typically, the integers. Computing devices, which in most cases are synchronized by a clock, are traditionally formalized by such discrete models as automata of some type.

Such a traditional approach has proven inadequate for most modern applications which involve complex systems with components whose time behavior and modeling needs are quite heterogeneous: think, e.g., of a continuous industrial process monitored and controlled by computing devices. In such systems, some components evolve through discrete steps, but

☆ Work supported by the European Commission, Programme IDEAS-ERC, Project 227977-SMScom. This work contains material previously published in [1] and [2]. This work was done while Luca Ferrucci was at Politecnico di Milano.

* Corresponding author.
E-mail addresses: matteo.rossi@polimi.it (M. Rossi), dino.mandrioli@polimi.it (D. Mandrioli), angelo.morzenti@polimi.it (A. Morzenti), luca.ferrucci@polimi.it (L. Ferrucci).

http://dx.doi.org/10.1016/j.tcs.2016.06.042
0304-3975/

their pace is determined by the environment in which they are embedded, which produces asynchronous stimuli to be reacted on with severe time constraints. As a consequence, such discrete steps often exhibit durations that may differ even by orders of magnitude–contrast, e.g., the switching of a transistor to a fire alarm and to the reaction of the automatic fire sprinklers.

To deal with such sharply different types of timing features in system components, many models studied in the literature offer the use of micro- and macro-steps, of which, normally, only the latter ones "consume time": micro-step durations, being negligible w.r.t. macro ones, are roughly assimilated to zero-time. Notions of zero-time transitions appear very naturally when reasoning about computations of embedded systems (see, e.g., [3], Chapter 6), and more generally of cyber-physical systems: they are also a natural mathematical tool to formalize Zeno behaviors, i.e., an accumulation of an infinite sequence of events with no "sensible time advancement". For instance, in [4], while developing and analyzing a formal model of an aerospace satellite system, the authors repeatedly emphasize the need for "algorithmic detection of Zeno behavior". Using zero-time transitions simplifies models and their analysis, but it is not without drawbacks, because the system can be in different, maybe even infinitely many, states, at the same time, with an obvious risk of engendering contradictions. We refer the reader to [5] for a more complete view of the time modeling issue in the literature.

Not only formal models must adequately represent system behavior, but they must also support mechanisms to effectively analyze—in a rigorous and possibly automated fashion—their properties. In fact, the fairly recent success of model-checking-based techniques has spurred a great interest in "push-button" analysis tools, which, by definition, are based on decidable formalisms. A flurry of (temporal) logic-based models has been developed in the last decades, looking for the best trade-off between expressiveness and naturalness on one hand, and decidability and complexity of the analysis on the other [6,7].

In this paper we propose a novel approach to **deal in a joint way with the three issues** mentioned above: **modeling systems that evolve stepwise** without a synchronizing clock; allowing for **steps whose durations may differ by orders of magnitude**, yet avoiding the logical trap and counterintuitive semantics of zero-time approximation; **supporting automatic analysis**, which we achieve by identifying a suitable decidable fragment of a novel logic formalism that is in general undecidable. We will anchor our approach to our own temporal logic language TRIO [8], but we emphasize that it can be applied to other temporal logics such as, for example, MTL [9].

In a nutshell, we first extend the original TRIO language in two major ways to make it suitable to model systems with the above features. On the one hand, we introduce a "next-step" operator imported from classic temporal logic; since we use the traditional textual symbol X for this operator, we name our augmented language X-TRIO. On the other hand, we borrow from Nonstandard Analysis (NSA) [10] the concept of infinitesimal number to formalize the non-null but negligible duration of micro steps, as opposed to that of macro steps which is represented by standard numbers.

In its full generality TRIO includes full arithmetic and is, therefore, undecidable; thus, we look for suitable restrictions that make it decidable, but still general enough to formalize and analyze the main properties of various systems of industrial relevance. Among the many possible ones, the approach used in this paper is based on a syntax inspired by decidable versions of Metric Temporal Logic (MTL) [9] and a "fine tuning" of the temporal domain over which X-TRIO formulae are interpreted. The decidability of the chosen X-TRIO subset is then demonstrated through translation into Linear Temporal Logic, which enables the use of any LTL-based satisfiability solver such as, for example, $\mathbb{Z}$ot [11], to analyze its formulae. Even if we devote a necessary technical effort to devise a decidable subset of X-TRIO suitable for a typical "push-button" verification, we also emphasize the applicability of X-TRIO to the modeling and—possibly semiautomatic—analysis of complex systems that require more expressive languages. A few examples in Sections 3 and 6 will be devoted to clarify the use of the various versions of X-TRIO.

This paper is structured as follows. Section 2 puts our work in the context of the existing literature, with particular attention to the formalisms based on timed words and 0-time transitions. Section 3 presents the general syntax and semantics of the X-TRIO language, with some examples of use, and Section 4 analyzes some properties of a propositional version thereof, which is shown to be undecidable. Section 5 introduces a further restricted fragment of X-TRIO to achieve decidability with a reasonable complexity. Section 6 presents two case studies of application of X-TRIO to formalize the semantics of two classical operational formalisms, namely timed Petri nets and Simulink/Stateflow diagrams, and to prove their properties, whether in fully automatic way or by means of deductive techniques based on a suitable axiomatization. Section 7 concludes and hints at future developments.

This work includes and extends material previously published in [1] and [2]. Precisely, [1] introduced a first propositional version of X-TRIO which is extended here to a more general version, and whose syntax, semantics and decidability issues have been investigated in more detail in this paper; [2] presents the complete case study on the formalization of Simulink/Stateflow diagrams which is summarized in Section 6.2.

2. Related work

Notions of zero-time transitions, micro- and macro-steps appear very naturally when reasoning about computations of embedded systems, thus it is not surprising to find such concepts in the literature on real-time temporal logics. Since the very early developments in this field, approaches were introduced that admit zero-time transitions at the price of associating multiple states to single time instants [12]. Our approach is akin to that of [9], which introduces a general framework accommodating suitable time structures supporting the notion of micro- and macro-steps, focusing on naturalness and read-