Theoretical Computer Science 515 (2014) 96–122

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Robust synthesis for real-time systems ☆

Kim G. Larsen [a], Axel Legay [b], Louis-Marie Traonouez [b,*], Andrzej Wąsowski [c]

[a] Aalborg University, Science Selma Lagerlöfs Vej 300, 9220 Aalborg East, Denmark
[b] IRISA/INRIA Rennes, 263 Avenue du Général Leclerc, 35042 Rennes Cedex, France
[c] IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark

ARTICLE INFO

Article history:
Received 21 June 2012
Received in revised form 19 July 2013
Accepted 22 August 2013
Communicated by P. Aziz Abdulla

Keywords:
Stepwise refinement
Timed I/O automata
Timed games
Specification theory
Robustness

ABSTRACT

Specification theories for real-time systems allow reasoning about interfaces and their implementation models, using a set of operators that includes satisfaction, refinement, logical and parallel composition. To make such theories applicable throughout the entire design process from an abstract specification to an implementation, we need to reason about the possibility to effectively implement the theoretical specifications on physical systems, despite their limited precision. In the literature, this implementation problem has been linked to the robustness problem that analyzes the consequences of introducing small perturbations into formal models.

We address this problem of robust implementations in timed specification theories. We first consider a fixed perturbation and study the robustness of timed specifications with respect to the operators of the theory. To this end we synthesize robust strategies in timed games. Finally, we consider the parametric robustness problem and propose a counter-example refinement heuristic for computing safe perturbation values.

1. Introduction

Component-based design is a software development paradigm well established in the software engineering industry. In component-based design, larger systems are built from smaller modules that depend on each other in well delimited ways described by interfaces. The use of explicit interfaces encourages creation of robust and reusable components.

Practice of component-based design is supported by a range of standardized middleware platforms such as CORBA [1], OSGi [2] or WSDL [3]. These technologies are usually not expressive enough to handle intricate correctness properties of safety-critical concurrent real-time software—a domain where component-based design would be particularly instrumental to address the strict legal requirements for certification [4]. To aid these needs, researchers work on developing trustworthy rigorous methods for component-oriented design. In the field of concurrency verification this includes compositional design (specification theories, stepwise-refinement) and compositional model checking. Akin to algebraic specifications, *specification theories* provide a language for specifying component interfaces together with operators for combining them, such as parallel (structural) composition or conjunction (logical composition), along with algorithms for verification based on refinement checking.

For real-time systems, timed automata [5] are the classical specification language. Designs specified as timed automata are traditionally validated using model checking against correctness properties expressed in a suitable timed temporal logic

☆ The research presented in this paper has been supported by MT-LAB, a VKR Centre of Excellence for the Modeling of Information Technology.
* Corresponding author. Tel.: +33 299847456; fax: +33 299847171.
E-mail addresses: kgl@cs.aau.dk (K.G. Larsen), axel.legay@inria.fr (A. Legay), louis-marie.traonouez@inria.fr (L.-M. Traonouez), wasowski@itu.dk (A. Wąsowski).

0304-3975/$ – see front matter
http://dx.doi.org/10.1016/j.tcs.2013.08.015

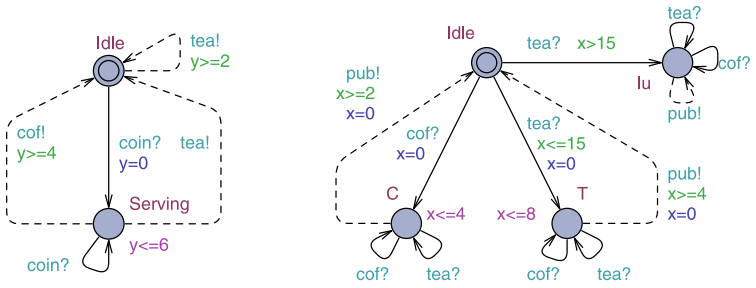

Fig. 1. Timed specifications with timed I/O automata.

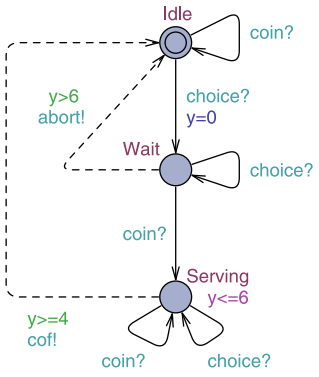

Fig. 2. Non-robust specification of a coffee machine.

[6]. Mature modeling and model-checking tools exist, such as Uppaal [7], that implement this technique and have been applied to numerous industrial applications [8–12].

In [13], we have proposed a specification theory for real-time systems based on timed automata. A specification theory uses refinement checking instead of model checking to support compositionality of designs and proofs from ground up. We build on an input/output extension of timed automata model to specify both models *and properties*. The set of state transitions of the timed systems is partitioned between inputs, representing actions of the environment, and outputs that represent the behavior of the component. The theory is equipped with a game-based semantic. The two players, Input and Output, compete in order to achieve a winning objective—for instance safety or reachability. These semantics are used to define the operations of the theory, including satisfaction (can a specification be implemented), refinement (how two specifications compare), logical composition (superposition of two specifications), structural composition (combining smaller components into larger ones), and quotient (synthesizing a component in a large design).

Let us illustrate the main concepts with an example. Fig. 1(a) displays a specification of a coffee machine that receives an input coin? and outputs either coffee (cof!) or tea (tea!). It can be composed with the specification of a researcher in Fig. 1(b) by synchronizing input with output labeled with the same channel name (cof and tea). In Fig. 1(b) the researcher specifies that if tea arrives after 15 time units, she enters into an error state lu. We can say that if there exists an environment for these two specifications that avoids reaching this error state then the two specifications are compatible [14]. In this particular example, such an environment simply needs to provide coin? sufficiently often. In general deciding existence of safe environments is reduced to establishing whether there exists a winning strategy in the underlying timed safety game.

Besides compatibility checking, the theory of [13] is equipped with a consistency check to decide whether a specification can indeed be implemented. Unfortunately, this check does not take limitations and imprecision of the physical world into account. This is best explained with an example. Consider the refined specification of the coffee machine in Fig. 2. This machine first proposes to make a choice of a drink, then awaits a coin, and, after receiving the payment, delivers the coffee. If the payment does not arrive within six time units, the machine aborts the drink selection and returns to the initial state, awaiting a new choice of a beverage. Already in this simple example it is quite hard to see that implementing a component satisfying this specification is not possible due to a subtle mistake. Suppose that the environment makes the choice? action in the Idle location, waits six time units, and then provides the coin? action. In such execution the system reaches the location Serving with clock y at value 6. The invariant $y \leqslant 6$ in the Serving location requires now that any valid implementation must deliver the coffee (cof!) immediately, in zero time units. No physical system would permit this, so we say that this state is not robustly consistent.