Theoretical Computer Science 594 (2015) 201–222

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Synthesis for continuous time

Tim French, John McCabe-Dansted, Mark Reynolds [*]

The University of Western Australia, Australia

A R T I C L E   I N F O

Article history:
Received 19 October 2012
Received in revised form 30 May 2015
Accepted 15 June 2015
Available online 19 June 2015
Communicated by R. van Glabbeek

Keywords:
Temporal logic
Real-number time
Synthesis

A B S T R A C T

When considering applications of reasoning in formal temporal logics, and particularly applications that need reasoning about distributed concurrent systems, multi-agent systems, or understanding natural language, it is convenient to use a logic based on a real-number flow of time rather than the usual discrete flow.
Such a logic may be employed for several distinct reasoning tasks and one that has potential to be one of the most useful is building a model from a specification, a simple form of synthesis.
In this paper, we consider the task of model-building for temporal logic specifications over the real-number linear flow of time. We present a new notation for giving a detailed description of the compositional construction of such a model and an efficient procedure for finding such a description from the temporal specification.

1. Introduction

Temporal logics support formal reasoning about processes over time and thus support investigation into the correctness of complex systems. For many applications there are good reasons, e.g., interleaving of operations of separate components, continuous input signals, and varying the granularity of abstraction, to use a logic based on some sort of dense model of time rather than the traditional discrete model based on natural numbers and its standard temporal logic [20]. Examples include multi-agent systems, natural language understanding, concurrency, and refinement. However, while techniques for rigorous or automated reasoning with discrete time temporal logics have had a long history of sustained development, work with dense and more general linear flows of time has been slow and intermittent.

Very general Until and Since connectives, based closely on natural language counterparts, were introduced in [15] for interpretation over various linear flows. They can be used in a variety of different, but related, logics by taking this temporal language and restricting the semantics to operate only on certain classes of linear flows.

Of the temporal logics for dense time, perhaps the most natural and well-established is RTL, the propositional temporal logic over real-number time using Kamp's Until and Since language. In a famous "expressive completeness" result [15] showed that RTL is as expressive as the monadic first-order logic of the real numbers, and so is at least as expressive as any other standard temporal logic which could be defined over real-number time.

Reasoning in RTL is fairly well understood: complete Hilbert-style axiom systems for RTL are given in [8] and [24]. Satisfiability and validity in RTL is decidable [3]. However, the two different decision procedures in [3] are both very complex and not suitable for any kind of practical implementation. One uses Rabin's non-elementarily complex decision procedure for the second-order monadic logic of two successors. Furthermore, deciding validity in the equally expressive first-order

[*] Corresponding author.
E-mail address: mark.reynolds@uwa.edu.au (M. Reynolds).

http://dx.doi.org/10.1016/j.tcs.2015.06.032
0304-3975/© 2015 Elsevier B.V. All rights reserved.

monadic logic of the real numbers is a non-elementary problem [33]. More recently, there has been some more positive news as [28] showed that deciding validity or satisfiability in RTL is PSPACE-complete.

However, satisfiability checking is not the only practically important reasoning task and not the only reasoning task that could be automated. In this paper, we will look instead at the *synthesis*, or automatic implementation task. Across temporal logics there are a variety of similar problems that have been called synthesis or *uniformisation* and we give a brief account in Section 8. The general idea is that a specification for the behaviour of a system is the input, and the output is a specific system design or implementation which is guaranteed to meet the specification. This task is most challenging in the context of an open or reactive system but here, in the context of a closed system, with no propositions controlled by an external environment, we just have a model building exercise. We seek an algorithm which can output a complete description of a specific model of the input formula, whenever the input formula is satisfiable. In case of an unsatisfiable input formula, the algorithm should just report the unsatisfiability. Thus, this problem is at least as hard as unsatisfiability checking. Applications include "automated programming", or automated construction of a system that is guaranteed to meet its specification, as well as automated construction of counter-examples during debugging.

To the best of our knowledge, we make the first attempt at investigating synthesis for RTL in this paper. The idea here is to provide an account of what is possible in this direction in a fairly general way. It is expected that further work can make improvements when the sorts of models are limited, for example to assume *finite variability* [12], or if there are extra features in the specification language such as metric constraints [1].

A hurdle in the way of developing tools for synthesis for infinite temporal models is that we have to devise a way of being able to output a necessarily finite "complete" description of an infinite temporal structure. Towards this end, we first present a new suitable notation for describing models in a concrete way. The compositional approach presented here was hinted at in [25], and traces back to pioneering work in [18] and [3]. It uses a small number of distinct operations for putting together a larger model from one or more smaller ones, or copies thereof. For example, the *shuffle* construct makes a new linear structure from a dense mixture of copies of a finite number of simpler ones. A good overview of the mathematics of linear orders can be found in [32].

We introduce a formal model expression language for defining a model via these inductive operations. In fact, we first give a language for making general linear structures in this way and then define a restricted sub-language (the real model expression language) capable of specifying structures which have the real-number flow of time. Having a formal model building language opens up the possibility for workable definitions of such tasks as synthesis and model checking for real-flowed structures. Questions of expressibility can be formalised and the computational complexity of these reasoning tasks can be assessed.

One of our results here, echoing the earlier work of [18,3], and others, is that the real model expression language is able to describe some real-flowed model of every satisfiable RTL formula.

The major advance of this paper on earlier ones is that we also present an FPSPACE procedure for finding the real model expression of a model from any given satisfiable RTL formula. We also show the problem is FPSPACE-complete and put an exponential bound on the length of expressions. The real model expression tells us exactly (up to isomorphism) how to make a specific real-flowed model of the formula. This is our first synthesis result.

Some of the proofs here use the mosaic techniques for temporal logic developed in [28]. These mosaics are small pieces of a real-flowed structure. In that paper, we tried to find a finite set of small pieces which is sufficient to build a real-number model of a given formula. We showed that if a formula was satisfiable, then we could find a sufficient set of mosaics. In this paper, we go further and show how to build a compositional model, i.e., one corresponding to an expression in our model language, when there is such a set of mosaics.

The extension here is built on a series of lemmas mirroring some of the earlier results, but keeping a much closer track on the relationship between mosaics and some compositionally built structures which witness them. See Section 8 for some more details on what is novel and what is re-used.

In Section 8, we also describe a possible alternative route to synthesis results over the reals, and over general linear time, not using mosaics, but instead using some of the recent work by Rabinovich [23] as well as a new type of automata introduced by Cristau in [5].

An advantage of the mosaic-based synthesis approach that we take is that there already exist fairly intuitive mosaic-based tableau reasoning techniques for temporal logics over classes of linear flows of time [27,30]. These already give us saturated sets of mosaics and decomposition trees from which (in future work) we should be able extract model descriptions for satisfiable formulas without too much trouble. It is hoped that the work in this paper will lead to the development of a tableau-style reasoner for the case of the reals. A satisfiable formula will have a tableau from which a real model expression can be read off.

Because of Kamp's expressive completeness theorem [15], and subsequent algorithmic accounts of it [9], we also obtain, as a corollary to the RTL synthesis result, a synthesis procedure for the monadic second-order logic of the reals.

In Section 2 we present RTL. In Section 3 we introduce the compositional approach to building linear models. In Section 4 we remind ourselves of useful properties of mosaics from [28]. In Section 5 we use mosaics to give a proof of the expressiveness result for RTL: satisfiability implies satisfiability in a compositional model. In Section 6 we present the new synthesis result for RTL. In Section 7 we build on the RTL results to provide the synthesis result for monadic first-order logic over the reals. Related work is summarised in Section 8 and we conclude in Section 9.