Theoretical Computer Science 594 (2015) 120–142

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Next-preserving branching bisimulation

Nisansala Yatapanage [a,*,1], Kirsten Winter [b]

[a] School of Computing Science, Newcastle University, Newcastle upon Tyne, NE1 7RU, UK
[b] School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, QLD 4072, Australia

ARTICLE INFO

Article history:
Received 3 April 2013
Received in revised form 7 May 2015
Accepted 11 May 2015
Available online 15 May 2015
Communicated by R. van Glabbeek

Keywords:
Bisimulation
Transition system
Temporal logic
CTL*
Model checking
Slicing
Behavior Trees

ABSTRACT

Bisimulations are equivalence relations between transition systems which assure that certain aspects of the behaviour of the systems are the same in a related pair. For many applications it is not possible to maintain such an equivalence unless non-observable (stuttering) behaviour is ignored. However, existing bisimulation relations which permit the removal of non-observable behaviour are unable to preserve temporal logic formulas referring to the next step operator. In this paper we propose a *family of next-preserving branching bisimulations* to overcome this limitation.

Next-preserving branching bisimulations are parameterised with a natural number, indicating the nesting depth of the **X** operators that the bisimulation preserves, while still allowing non-observable behaviour to be reduced. Based on van Glabbeek and Weijland's notion of branching bisimulation with explicit divergence, we define the novel parameterised relation for which we prove the preservation of CTL* formulas with an **X** operator-nesting depth that is not greater than the specified parameter. It can be shown that the family of next-preserving bisimulations constitutes a hierarchy that fills the gap between branching bisimulation and strong bisimulation.

As an example for its application we show how this definition gives rise to an advanced slicing procedure that creates a *formula-specific* slice, which constitutes a reduced model of the system that can be used as a substitute when verifying this formula. The result is a novel procedure for generating slices that are next-preserving branching bisimilar to the original model for any formula. We can assure that each slice preserves the formula it corresponds to, which renders the overall verification process sound.

1. Introduction

A bisimulation relation defines an equivalence between two transition systems. Two models are bisimilar if their observable behaviours are indistinguishable and thus satisfy the same properties. A variety of bisimulation relations have been defined in the literature and they vary in what is considered observable and what is not. The meaning of "observable" is relative to the type of bisimulation relation. For example, strong bisimulation [1,2] considers all steps to be observable whereas weak forms of bisimulation relations regard silent or stuttering steps (i.e., steps that do not change the state of the model) as non-observable.

* Corresponding author.
E-mail addresses: yatapanage@acm.org (N. Yatapanage), kirsten@itee.uq.edu.au (K. Winter).

[1] The concepts of this paper were developed while the first author was a PhD student at the Institute for Integrated and Intelligent Systems, Griffith University, Australia.

http://dx.doi.org/10.1016/j.tcs.2015.05.013
0304-3975/

Strong bisimulation is known to preserve all temporal properties specified in the logic CTL*, the super-set of the linear and branching time temporal logics, LTL and CTL (see [3]). However, the properties that are preserved through weak forms of bisimulation are only those temporal properties which do not contain the temporal next-step operator **X** (we refer to these as CTL^*_{-X} formulas). This is due to the fact that stuttering steps can affect the validity of formulas that contain the next-step operator. Hence, it seems that the preservation of full CTL* requires strong bisimulation. On the other hand, using strong bisimulation is not always suitable, since some applications rely on the fact that stuttering steps can be neglected. An example of this is *slicing* [4], a technique in which a program or model is reduced by eliminating parts which are irrelevant according to a given criterion. The sliced model and the original are related only through weak forms of bisimulation, since the irrelevant (or stuttering) steps do not exist in the slice. Our aim is to find a compromise with a new bisimulation that is strong enough to preserve all CTL* formulas but at the same time weak enough to allow for some reduction, namely of those stuttering steps that have no effect (on the validity of a given formula). This judgement is obviously measured with respect to a particular formula and, as a consequence, leads to a new *parameterised* bisimulation relation.

In this paper we define a novel form of bisimulation, referred to as the *family of next-preserving branching bisimulations*, which is based on van Glabbeek and Weijland's branching bisimulation [5]. The relations contained in this family are parameterised with a natural number which indicates the number of stuttering steps that are preserved at critical points in the system. This number indicates which formulas are preserved by the bisimulation. It can be shown that any formula with a nesting depth of **X** operators smaller than or equal to the specified parameter is preserved in the parameterised next-preserving bisimilar system: we give a proof that next-preserving branching bisimulation *of depth xd* preserves any CTL* formula with an **X**-nesting depth of *xd* or less. The next-preserving bisimulations are stronger than weak forms of bisimulation and weaker than strong bisimulation. In fact, they can be ordered into a hierarchy of bisimulations enclosed by branching bisimulation (at the weak end of the spectrum) and strong bisimulation (at the strong end). These results are useful in any application that aims for a reduction of the model, such as slicing (e.g., [4,6]) or partial order reduction [7].

In our context, the driving force behind the definition of the next-preserving branching bisimulation was the intention of improving the slicing algorithm used for temporal logic verification, namely model checking [8,9]. The aim of slicing is to reduce (prior to model checking) a large model to a bisimilar slice which can be model checked instead of the original model, to combat the state explosion problem inherent to model checking. Based on our past experience in the domain of safety analysis (e.g., [10–12]), we found that there are many cases in which the temporal logic formula of interest includes the next-step operator, and hence these formulas must be preserved by the slice. In the second part of this paper we introduce the resulting new slicing approach for the *Behavior Tree* modelling notation, which is a graphical formal notation aiming to support the user in capturing informal requirements [13,14]. Once a model has been derived from the requirements, it can be analysed using a model checker. To guarantee soundness of the approach we need to ensure that the same temporal logic formulas are satisfied in the slice as in the original model. As a vehicle to prove the soundness of our slicing approach, the notion of next-preserving branching bisimulation with respect to a particular formula is used, which guarantees the preservation of this formula (even if it contains the next-step operator). Our approach for deriving a next-preserving slicing algorithm for Behavior Trees, we believe, can also be used to derive similar slicing algorithms for other notations.

The remainder of this paper is organised as follows. Section 2 provides the necessary background and motivation. Firstly, preliminary concepts used throughout the paper are introduced, such as doubly labelled transition systems (Section 2.1), CTL* (Section 2.2) and branching bisimulation (Section 2.3). This is followed by some preliminary discussion on the preservation of the next operator, including motivating examples (Section 2.4). Section 3 constitutes the core of our work, where we define our notion of a *family of next-preserving branching bisimulations*. The definition is followed by some properties of this family in Section 3.1. Utilising these properties, Section 3.2 delivers a proof of the preservation of CTL*. Section 3.3 proposes a specialisation for next-preserving bisimulations where parameterisation is extended so that a next-preserving bisimulation becomes specific for a particular formula. Section 3.4 presents the family of next-preserving bisimulations as a hierarchy that fills the gap between strong bisimulation and branching bisimulation.

In the second half of the paper, these results are then applied in the context of slicing for Behavior Trees. In Section 4 we provide the definitions on which basic slicing of Behavior Trees is built. This summarises the results from [15] and gives the background. In Section 5 we demonstrate how the definition of the family of next-preserving branching bisimulations gives rise to a new sophisticated slicing procedure which generates a next-preserving bisimilar slice for a specific formula. Finally, Section 6 relates our work to other results in the literature and we conclude in Section 7.

2. Bisimulation and the next operator

This section gives a brief introduction to doubly-labelled transition systems, the temporal logic CTL* and existing bisimulation relations, followed by a discussion on the next operator, providing the motivation for defining next-preserving branching bisimulations.

2.1. Doubly-labelled transition systems

Bisimulations are equivalence relations, normally defined either between labelled transition systems, which have labels on transitions, or Kripke structures, which have labels on states. For our purposes, we have chosen to use *doubly-labelled*