Formula-Dependent Abstraction for CTL Model Checking

Junyan Qian, Lingzhong Zhao, Guoyong Cai, and Tianlong Gu
Department of Computer Science and Technology,
Guilin University of Electronic Technology, Guilin 541004, China
qjy2000@gmail.com

Abstract. We present a state abstraction that is defined with respect to a given CTL formula. Since it does not attempt to preserve all ACTL formula, like simulation does, we can expect to compute coarser abstraction. Specifically, the abstraction is used to reduce the size of each Kripke structure, so that their product will be smaller. When the abstraction is too coarse, we show how refinement can be applied to produce a more precise abstract model. We also extend the notion of formula-dependent abstraction to Kripke structure with fairness, and define the coarsest abstraction that preserves the given CTL formula interpreted with respect to the fair paths. The method is exact and fully automatic, and handles full CTL.

1 Introduction

Model checking [1] is a fully automatic technique of verifying finite-state reactive-system for the correctness of design specifications such as hardware design and protocol verification. Branching time propositional temporal logic has been proven very useful in the automatic verification of concurrent finite-state systems [2]. The systems are modeled using labeled state transition structures called Kripke or temporal structures [3]. The properties that one wishes to verify can be expressed in terms of a branching time temporal logic such as CTL [8]. An efficient search procedure, which was originally developed in 1981 by Clarke and Allen Emerson [4], is used to determine automatically whether the model satisfies the specifications [3].

Oftentimes, the straightforward approach to model checking concurrent systems is to construct the product of the component structure to yield a single structure, and then proceed to model check the single structure. However, the size of the product structure can be exponential in the number of component structures. This is known as the state explosion problem. In order to ameliorating the state explosion problem, Abstraction, where one build up a smaller abstract model in a manner which ensures that the specification holds for the original model if it holds for the abstract model, is a main method for reducing the state space of the checked system by hiding some of the system details that might be irrelevant for the checked property.

Our goal is to develop an algorithm that alleviates the explosion problem by formula-dependent abstraction in each component structure. The abstraction is used to simplify the component before taking their product, thus leading to a smaller product
structure. It is well known that simulation preorder relation is a coarse abstraction that preserves the truth of all ACTL* formulas. However, in general we are interested in model checking a system with respect to just a few formulas, and hence preserving all ACTL formulas is stronger than needed. Thus, we investigate a formula-dependent abstraction that preserves the truth of a particular formula of interest, but possibly not of other formulas. This leads to a coarser abstraction, and thus to a greater opportunity for simplification.

In many cases, we are only interested in the correctness along fair computation paths. We also extend the notion of formula-dependent abstraction to Kripke structures with fairness. Fairness constraints allow us to reason about only those infinite paths in the Kripke structure with satisfy some fairness specification, which is evaluated over the infinite path. Often, it is more natural to think of the fairness constraints as part of the system specification, instead of part of the property being verified. We will refer to Kripke structures with fairness constraints as fair Kripke structures, and the problem of checking a CTL formula on a fair Kripke structure as the FairCTL model checking.

The remainder of this paper is organized as follows: Section 2 discusses related work, and Section 3 presents some preliminaries. In Section 4 we develop our formula-dependent abstraction in Kripke structure, and in Section 5 we describe parallel composition. Section 6 model checking the abstract model, and Section 7 extends formula-dependent abstraction to fair Kripke structure. Finally, Section 8 gives conclusions.

2 Related Work

As key issue in model checking, Abstraction [16] is a method for reducing the state space of the checked system. The reduction is achieved by hiding (abstracting away) some of the system details that might be irrelevant for the checked property. When model checking using abstraction, the main concern is that the abstractions must be property-preserving. There are two forms of property preservation: Weak Preservation and Strong Preservation. An abstraction is a weak property preserving if a set of properties true in the abstract system has corresponding properties in the concrete system that are also true, while an abstraction is a strong preserving abstraction if a set of properties with truth values either true or false in the abstract system has corresponding properties in the concrete system with the same truth values.

It is usually difficult and expensive to compute a precise abstraction directly. In order to reduce the complexity, approximation is often used. There are two main forms of approximation abstraction: over-approximation and under-approximation. In over-approximation, more behaviors are added in the abstract system than are present in the concrete system. This approach provides a very popular class of weakly preserving abstractions for universally quantified path properties. However, over-approximation often only works well for safety (or invariant) properties. In under-approximation, the behaviors are removed when going from the concrete to the abstract system. Under-approximation is also often found in the construction of an environment for a system to be checked.