BMC via on-the-fly determinization

Toni Jussila*, Keijo Heljanko**, Ilkka Niemelä***

Helsinki University of Technology, Laboratory for Theoretical Computer Science, P.O. Box 5400, FI-02015 TKK, Finland
e-mail: {Toni.Jussila,Keijo.Heljanko,Ilkka.Niemela}@tkk.fi

Published online: 25 January 2005 – © Springer-Verlag 2005

Abstract. This paper develops novel bounded model checking (BMC) techniques for asynchronous parallel systems. The aim is to increase the efficiency of BMC by exploiting the inherent concurrency in such systems. This added efficiency is gained by covering more reachable states within a given bound using two techniques. Firstly, a nonstandard execution mode, step executions, where multiple actions can take place simultaneously is applied. Secondly, the number executions the system can have is reduced by modeling the execution of the system components as if they were determinized. This determinization technique also enables the removal of the internal transitions of the components. Step executions can be further restricted to a subclass called process executions without losing any reachable states.

The paper presents a translation scheme for BMC of reachability properties. The translation is from an asynchronous system where the components are modeled as labeled transition systems (LTSs) to a propositional formula. The models of the formula correspond to the step executions of the original system where each component is replaced with its determinized counterpart. The formula for step executions can be easily extended in such a way that its models correspond to the process executions of the system. The translation scheme has been implemented and some experimental comparisons performed. The results show that the bound needed to detect a violation of a reachability property is, for step and process executions, in most cases lower than in interleaving executions and that the running time of the model checker using process executions is smaller than that using steps. Moreover, the performance compares favorably to a state-of-the-art interleaving BMC implementation in the NuSMV system.

Keywords: Verification – Bounded model checking – Labeled transition system

1 Introduction

The purpose of this paper is to develop efficient bounded model checking (BMC) techniques for asynchronous systems modeled as labeled transition systems (LTSs). BMC is a verification technique that considers only executions of bounded length of the chosen formalism [1]. The general model checking problem for properties specified in linear temporal logic (LTL) is known to be \textbf{PSPACE}-complete w.r.t. the system description given, for instance, as an LTS system studied in this paper. However, the bounded case is in \textbf{NP} (assuming the used bound is given in unary encoding). The very idea is to compile the system under verification, the property to be verified and a bound \( k \) on the length of the execution to a propositional formula having a model iff the system has an execution of length \( k \) that violates the property. The methodology has been successfully applied in industrial settings [2, 3].

The aim of this work is to increase the efficiency of BMC by exploiting the inherent concurrency in asynchronous systems. The standard approach to such systems is to use interleaving executions, where exactly one
action is occurring at a time. For example, consider the system in Fig. 1. It presents 2n LTSs following the standard notation of presenting the states as circles and the transitions from a state to another as arrows. The arrows are labeled with symbols like \( \alpha_1 \) or \( \tau \).

The 2n components form a system whose global states (denoted by \( s \)) are 2n-tuples of local states, one local state from each component. The initial global state is the tuple where each component is in its initial state (marked in the picture with a wedge and labeled \( s_i \) or \( s_i' \), where \( 1 \leq i \leq n \)).

The system can move from global state \( s \) to global state \( s' \) with label \( \alpha_i \) (denoted \( s \xrightarrow{\alpha_i} s' \)) iff every component having \( a_i \) among its labels can execute a transition labeled \( \alpha_i \). That means that before the transition each such component is in a local state having an outgoing transition labeled \( \alpha_i \). In the next global state, each of these components is in the target state of that transition while every other component remains in its original state. However, the label \( \tau \) is an exception to the rule. It denotes an internal transition, and thus a single component can execute its \( \tau \)-transitions in isolation. An interleaving execution of the example system reaching the global state \( \langle u_1, u_1', \ldots, u_n, u_n' \rangle \) from the initial state \( \langle s_1, s_1', \ldots, s_n, s_n' \rangle \) is as follows:

\[
\langle s_1, s_1', \ldots, s_n, s_n' \rangle \xrightarrow{\alpha_1} \langle t_1, t_1', \ldots, s_n, s_n' \rangle \xrightarrow{\alpha_2} \\
\cdots \xrightarrow{\alpha_n} \langle t_1, t_1', \ldots, t_n, t_n' \rangle \xrightarrow{\tau} \langle u_1, u_1', \ldots, t_n, t_n' \rangle \\
\cdots \xrightarrow{\tau} \langle u_1, u_1', \ldots, u_n, u_n' \rangle.
\]

Here the idea is to encode interleaving executions more compactly by allowing multiple occurrences of actions in different components of the system simultaneously. For the example in Fig. 1 with the interleaving model, the number of steps needed to reach every state of the system is 3n. If simultaneous executions of independent actions are allowed, only two steps are needed (independent of the value of \( n \)).

The approach of allowing independent actions to take place simultaneously is further combined with an on-the-fly determination construction where for each component a set of states in which that component can be is maintained. The situation is illustrated in Fig. 2. On the left-hand side is a nondeterministic LTS and on the right-hand side its determined counterpart. The purpose of this construction is to (i) reduce the number of executions of the system and (ii) shorten their length by removing internal transitions.

In this work, the concurrent execution of independent actions combined with on-the-fly determination of components is referred to as step executions. The idea is illustrated in the following example execution of the system from Fig. 1:

\[
\{\{s_1\}, \{s_1'\}, \ldots, \{s_n\}\} \xrightarrow{\\{a_1, \ldots, a_n\\}} \\
\{\{t_1, u_1\}, \{t_1', u_1'\}, \ldots, \{t_n, u_n\}, \{t_n', u_n'\}\}.
\]

The execution is different from the interleaving model in that the actions \( a_1, \ldots, a_n \) are executed simultaneously. Secondly, due to on-the-fly determination, component \( L_1 \), for instance, reaches the set of states \( \{t_1, u_1\} \).

Without compromising reachable states, step executions can be further restricted to process executions satisfying an extra condition on visible actions.

Based on these ideas, a bounded model checking (BMC) procedure of reachability properties of an LTS system is developed by devising a translation scheme from the LTSs to a propositional formula. The novelty of the translation is that the models of the formula are the step executions of the system, i.e., in the models (interpreted as executions):

- several independent actions can be executed simultaneously and
- in each execution state, each component can be in a set of its local states, i.e., the formula models the executions of the determined version of the component.

Yet the size of the formula remains linear w.r.t. both the bound and the system description. In addition, the translation can be done without explicitly constructing the determined versions of the components, i.e., handling determinization on the fly. A simple addition to the formula modeling step executions results in a formula modeling process executions.

The approach has been applied to a set of deadlock checking problems, and the data obtained justify the following points. Firstly, step and process executions need