Proving Liveness by Backwards Reachability

Parosh Aziz Abdulla, Bengt Jonsson, Ahmed Rezine, and Mayank Saksena

Dept. of Information Technology, P.O. Box 337, S-751 05 Uppsala, Sweden
{parosh,bengt,rahmed,mayanks}@it.uu.se

Abstract. We present a new method for proving liveness and termination properties for fair concurrent programs, which does not rely on finding a ranking function or on computing the transitive closure of the transition relation. The set of states from which termination or some liveness property is guaranteed is computed by a backwards reachability analysis. The role of ranking functions is replaced by a check for a certain commutativity property. The method is not complete. However, it can be seen as a complement to other methods for proving termination, in that it transforms a termination problem into a simpler one with a larger set of terminated states. We show the usefulness of our method by applying it to existing programs from the literature. We have also implemented it in the framework of Regular Model Checking, and used it to automatically verify non-starvation for parameterized algorithms.

1 Introduction

The last decade has witnessed impressive progress in the ability of tools to verify properties of hardware and software systems (e.g., [8, 14, 22]). The success has to a large extent concerned safety properties, e.g., absence of run-time errors, deadlocks, race conditions, etc. Progress in verification of liveness properties has been less prominent. Safety properties can be checked by computing (an approximation of) the set of reachable states. In contrast, checking liveness is more difficult, and multiple fairness requirements can make the situation even more complicated.

For finite-state systems and some parameterized systems, the automata-theoretic approach [33] can be used to reduce the verification of liveness properties to checking repeated reachability, i.e., the absence of fair non-productive loops. In enumerative model checkers [22] this requires a repeated search through the state space; in symbolic model checkers, a natural but more expensive technique is to compute the transitive closure of the transition relation.

For general infinite-state systems, which is often what must be considered for software model checking, the difference between safety and liveness properties is even larger. For some classes of systems, such as lossy channel systems, checking safety properties is decidable [5], whereas checking liveness properties is undecidable [4]. The main approach for proving fair termination involves finding auxiliary assertions associated with well-founded ranking functions and helpful directions (e.g., [24]). Finding such ranking functions is not easy, and automation requires techniques adapted to specific data domains. Techniques have been developed for programs with integers or reals [10–12, 16, 17], functional programs, [23], and parameterized systems [20, 21].