Algorithmic Verification of Linear Temporal Logic Specifications *

Yonit Kesten**, Amir Pnueli***, and Li-on Raviv

Abstract. In this methodological paper we present a coherent framework for symbolic model checking verification of linear-time temporal logic (LTL) properties of reactive systems, taking full fairness into consideration. We use the computational model of a fair Kripke structure (FKS) which takes into account both justice (weak fairness) and compassion (strong fairness). The approach presented here reduces the model checking problem into the question of whether a given FKS is feasible (i.e. has at least one computation).

The contribution of the paper is twofold: On the methodological level, it presents a direct self-contained exposition of full LTL symbolic model checking without resorting to reductions to either CTL or automata. On the technical level, it extends previous methods by dealing with compassion at the algorithmic level instead of adding it to the specification, and providing the first symbolic method for checking feasibility of FKS's (equivalently, symbolically checking for the emptiness of Streett automata).

The presented algorithms can also be used (with minor modifications) for symbolic model-checking of CTL formulas over fair Kripke structures with compassion requirements.

1 Introduction

Two brands of temporal logics have been proposed over the years for specifying the properties of reactive systems: the linear time brand LTL [GPSS80] and the branching time variant CTL [CE81]. Also two methods for the formal verification of the temporal properties of reactive systems have been developed: the deductive approach based on interactive theorem proving, and the fully automatic algorithmic approach, widely known as model checking. Tracing the evolution of these ideas, we find that the deductive approach adopted LTL as its main vehicle for specification, while the model-checking approach used CTL as the specification language [CE81], [QS82].

This is more than a historical coincidence or a matter of personal preference. The main advantage of CTL for model checking is that it is state-based and, therefore, the process of verification can be performed by straightforward labeling

---

* This research was supported in part by an infra-structure grant from the Israeli Ministry of Science and Art and a gift from Intel.
** Dept. of Com. Sys. Eng., Ben Gurion University, ykesten@bgu.ac.il
*** Weizmann Institute of Science, amir@wisdom.weizmann.ac.il
of the existing states in the Kripke structure, leading to no further expansion or unwinding of the structure. In contrast, LTL is path-based and, since many paths can pass through a single state, labeling a structure by the LTL sub-formulas it satisfies necessarily requires splitting the state into several copies. This is the reason why the development of model-checking algorithms for LTL always lagged several years behind their first introduction for the CTL logic.

The first model-checking algorithms were based on the enumerative approach, constructing an explicit representation of all reachable states of the considered system [CE81], and were developed for the branching-time temporal logic CTL. The LTL version of these algorithms was developed in [LP85] for the future fragment of propositional LTL (PTL), and extended in [LPZ85] to the full PTL. The basic fixed-point computation algorithm for the identification of fair computations presented in [LP85], was developed independently in [EL85] for FCTL (fair CTL). Observing that upgrading from justice to full fairness (i.e., adding compassion) is reflected in the automata view of verification as an upgrade from a Buchi to a Street automaton, we can view the algorithms presented in [EL85] and [LP85] as algorithms for checking the emptiness of Street automata [VW86]. An improved algorithm solving the related problem of emptiness of Street automata, was later presented in [HT96]. The development of the impressively efficient symbolic verification methods and their application to CTL [BCM+92] raised the question whether a similar approach can be applied to PTL. The first satisfactory answer to this question was given in [CGH94], which showed how to reduce model checking of a future PTL formula into CTL model checking. The advantages of this approach is that, following a preliminary transformation of the PTL formula and the given system, the algorithm proceeds by using available and efficient CTL model checkers such as SMV.

A certain weakness of all the available symbolic model checkers is that, in their representation of fairness, they only consider the concept of justice (weak fairness). As suggested by many researchers, another important fairness requirement is that of compassion (strong fairness) (e.g., [GPSS80], [LPS81], [Fra86]). This type of fairness is particularly useful in the analysis of systems that use semaphores, synchronous communication, and other special coordination primitives. A partial answer to this criticism is that, since compassion can be expressed in LTL (but not in CTL), once we developed a model-checking method for LTL, we can always add the compassion requirements as an antecedent to the property we wish to verify. A similar answer is standardly given for symbolic model checkers that use the $\mu$-calculus as their specification language, because compassion can also be expressed as a $\mu$-calculus formula [SdRG89]. The only question remaining is how practical this is.

In this methodological paper (summarizing an invited talk), we present an approach to the symbolic model checking of LTL formulas, which takes into account full fairness, including both justice and compassion. The presentation of the approach is self-contained and does not depend on a reduction to either CTL model checking (as in [CGH94]) or to automata. The treatment of the LTL component is essentially that of a symbolic construction of a tableau by assigning