First-Order LTL Model Checking Using MDGs

Fang Wang, Sofiène Tahar, and Otmane Ait Mohamed

Department of Electrical and Computer Eng
Concordia University, Montreal, Quebec, Canada
{f_wang, tahar, ait}@ece.concordia.ca

Abstract. In this paper, we describe a first-order linear time temporal logic (LTL) model checker based on multiway decision graphs (MDG). We developed a first-order temporal language, $\mathcal{L}_{MDG}^*$, which expresses a subset of many-sorted first-order LTL and extends an earlier language, $\mathcal{L}_{MDG}$, defined for an MDG based abstract CTL model checking. We derived a set of rules, enabling the transformation of $\mathcal{L}_{MDG}^*$ formulas into generalized Büchi automata (GBA). The product of this GBA and the abstract state machine (ASM) model is checked for language emptiness. We have lifted two instances of the generalized Strongly Connected Component (SCC)-hull (GSH) checking algorithm [17] to support abstract data and uninterpreted functions based on operators available in the MDG package. Experimental results have shown the superiority of our tool compared to the same instances of GSH implemented with BDDs in VIS.

1 Introduction

Formal verification has received considerable attention from the electrical engineering, computer science and the industry communities, where many BDD based formal verification tools being developed over the years. These, however, suffer from the well-known state space explosion problem. Multiway Decision Graphs (MDGs) [5] have been introduced as one way to reduce this problem. MDGs are based on a many-sorted first-order logic with a distinction between concrete and abstract sorts. Abstract variables are used to represent data signals, while uninterpreted function symbols are used to represent data operations, providing a more compact description of circuits with complex data path. Many MDG based verification applications have been developed during the last decade, including invariant checking, sequential equivalence checking, and abstract CTL model checking [21] of abstract state machines (ASM) [5]. The MDG tools are available at [22].

In this paper we introduce a new MDG verification application by implementing automata based model checking of a subset of first-order linear time temporal logic (LTL). Generally, LTL model checking verifies a Kripke structure with respect to a propositional linear time temporal logic (PLTL) formula. A PLTL formula $\phi$ is valid if it is satisfied by all paths of the Kripke structure $M$. The validation of $\phi$ can be done by converting its negation into a Generalized Büchi Automaton (GBA) [19] $B_{\neg\phi}$, composing the automaton with the
model $M$, and checking its language emptiness [19]. The main idea of the work we describe in this paper is to lift classical LTL model checking procedures to the language emptiness checking (LEC) of a GBA encoded with MDGs. To this end, we define an extended temporal logic, called $\mathcal{L}_{MDG}^*$, for which we have developed a set of derivation rules that transform $\mathcal{L}_{MDG}^*$ properties into PLTL formulas augmented with a transformation circuit, which will be composed with the system model (ASM) under verification. We use an automata generator to get a GBA for the negation of this PLTL formula. Language emptiness checking based on two instances of the GSH algorithm [17] is finally performed on the product of this latter and the composed ASM described earlier. We call this new MDG verification application MDG LEC.

The rest of the paper is organized as follows: Section 2 describes related work. Section 3 overviews the notion of multiway decision graphs. Section 4 defines the first order linear temporal logic $\mathcal{L}_{MDG}^*$ and related transformation rules. Section 5 describes the language emptiness checking algorithms. Section 6 provides a case study applying the developed MDG LEC tool on an ATM (Asynchronous Transfer Mode) switch fabric. Finally, Section 7 concludes the paper.

2 Related Work

The idea of first-order temporal logic model checking is not new. For instance, Bohn et. al [3] presented an algorithm for checking a first-order CTL specification on first-order Kripke structure, an extension of “ordinary” Kripke structures by transitions with conditional assignments. The algorithm separates the control and data parts of the design and generates the first-order verification conditions on data. The control part can be verified with Boolean model checking, while the data part of the design has to be verified using interactive theorem proving. Compared to this work, our logic is less expressive since $\mathcal{L}_{MDG}^*$ cannot accept existential quantification. However, in our approach the property is checked on the whole model automatically, while in [3] a theorem prover is needed to validate the first-order verification conditions. Besides, our method can be applied on any finite state models, while their application is limited to designs that can be separated into data and control parts.

Hojati et.al [13] proposed an integer combinational/sequential (ICS) concurrency model to describe hardware systems with datapath abstraction. They used symbols such as finite relations, interpreted and uninterpreted integer functions and predicates, and proceeded the verification of ICS models using language containment. For a subclass of “control-intensive” ICS models, integer variables in the model can be replaced by enumerated variables, hence enabling a verification at the Boolean level without sacrificing accuracy. Compared to ICS, our ASM models are more general in the sense that the abstract sort variables in our system can be assigned any value in their domain, instead of a particular constant or function of constants as in ICS models. For the class of ICS mod-