Automaton based Model Checking Using Multiway Decision Graphs

Fang Wang, Sofiène Tahar, and Otmane Ait Mohamed
Concordia University
Address: Electrical & Computer Engineering Department, 1455 Maisonneuve, Montreal, Quebec. H3H 1M8.
Tel: (514) 848-2424 ext. 3124 Email: f_wang@ece.concordia.ca

Abstract. *In this paper we present a formal hardware verification tool implementing an automata based model checking algorithm using Multiway Decision Graphs. The tool first transforms the property formula into a Generalized Buchi Automaton. It then uses a newly developed algorithm to check the language emptiness on the product machine composed from the system under verification and the constructed property automaton. We have implemented the proposed tool and applied it to a number of benchmark designs as well as a larger case study of an Asynchronous Transfer Mode switch fabric.*

1. Introduction

To overcome the state space explosion problem of the existing formal verification tools, Multiway Decision Graphs (MDGs) [1] have been introduced into this area. Rather than based on Boolean logic, MDGs describe the design model with an Abstract State Machine (ASM) which accepts abstract variables and uninterpreted function symbols. Several formal verification applications have been developed in the MDG tool package, including sequential equivalence checking [1], invariant checking [1], and regular model checking (MC) [5]. In this paper, a new automata based model checking approach is developed and integrated into the MDG package. Automata-based model checking is one of the most successful methods of formal verification, which is based on the idea of checking the language emptiness of the product of the system under verification and the automaton accepting all the models violating the property to be verified [4].

2. MDG LEC Tool

The structure of our MDG based Language Emptiness Checking (LEC) tool is shown in Figure 1. It verifies an ASM with respect to a first-order logic formula L_{MDG^*}, which is transformed into a Propositional Linear-Time Temporal Logic (PLTL) formula argument with small ASM models. Using an external tool Wring [3], we generate from the PLTL formula, a Generalized Buchi Automaton (GBA) which is then composed with the model ASM to build a product automaton. Finally, we check the language emptiness using a new checking algorithm based on MDG operators.

An L_{MDG^*} formula is defined by Next_let_formulas [5] and the linear temporal operators G (always), F (eventually), U (until), and R (release) in the regular form [3] [4]. The Next_let_formulas are defined with atomic formulas, Boolean connectives, the temporal operator X (next-time) and the LET *(v = t)* IN *p* operator, where *v* is an ordinary variable [1], *t* is an ASM-variable [1], and *p* is a Next_let_formula. The atomic formulas are: True, False, and *t1 = t2*, where *t1* is an ASM-variable, and *t2* is an ASM-variable, or a constant, or an ordinary variable or a function of ordinary variables.

The transformation of an L_{MDG^*} formula into a PLTL is obtained by generating circuit descriptions for Next_let_ formulas in the L_{MDG^*} formula. The circuit description is supported by MDG-HDL [1], which is a register-transfer level hardware description language accepting abstract variables and function symbols. The constructed GBA from the PLTL formula is also represented by a circuit description. The product automaton can be automatically generated using available operators in the MDG package.

The new language emptiness algorithm is adapted from the forward SCC-hull algorithm proposed in [2]. As shown in Figure 2, the algorithm, *LangEmptyCheck*, takes as its arguments an ASM G = (X, Y, Z, FI, FT, FO) and a set of Buchi fairness conditions CL = (C1,...,Cm), where X, Y and Z are sets of input, state and output variables, FI is the set of initial states, FT is the transition relation and FO is the output relation, respectively. *LangEmptyCheck* first computes the set of reachable states, and then removes the states that cannot be reached by fair cycles until reaching a fixpoint. If the fixpoint is empty, it returns 'verified'; otherwise, it returns 'failed'. The operators used are: Conj (conjunction of two sets with different abstract variables), New_state (direct successors of a set of states), Pbys (difference of two sets) [1]. ReAn is a procedure to compute all the states satisfying a given condition [1].

3. Application

We applied our MDG LEC tool to a number of benchmark designs as well as an Asynchronous Transfer Mode (ATM) switch fabric. For the latter, we specified a set of properties in L_{MDG^*}, and then described the ATM design as ASMs. Table 1 shows the experimental results of the model checking of five sample properties using our MDG LEC and the existing MDG MC tools, including CPU time, memory usage and number of MDG nodes generated. An'*' in the table indicates that a property cannot be verified.

4. Conclusions

In this paper, we presented a new tool supporting automata based model checking using MDGs. The tool, MEG LEC, can accept more general property templates than the existing regular MDG model checking tool. MDG LEC is also superior to other existing automata based model checking tools, such as FormalCheck, since it accepts abstract variables and uninterpreted function symbols.

5. References

[1] F. Corella, Z. Zhou, X. Song, M. Langevin and E. Cerny. Multiway Decision Graphs for Automated Hardware Verification. Formal Methods in System Design, Vol. 10, February 1997, pp. 7-46.

[2] K. Ravi, R. Bloem and F. Somenzi. A Comparative Study of Symbolic Algorithms for the Computation of Fair Cycles. In Formal Methods in Computer-Aided Design, LNCS 1954, Springer Verlag, 2000, pp 143-160.

[3] F.Somenzi and R. Bloem. Efficient Buchi automata from LTL formulae. In Computer Aided Verification, LNCS 1855, Springer Verlag, 2000, pp 247-263.

[4] M. Y. Vardi. An Automata-theoretic Approach to Linear Temporal Logic. In Logics for Concurrency: Structure versus Automata, LNCS 1043, Springer-Verlag, pp. 238-266.

[5] Y. Xu, E. Cerny, X. Song, F. Corella and O. Ait Mohamed. Model checking for a first-order logic using multiway decision graphs. In Computer Aided Verification, LNCS 1427, Springer Verlag, 1998, pp 219-231.

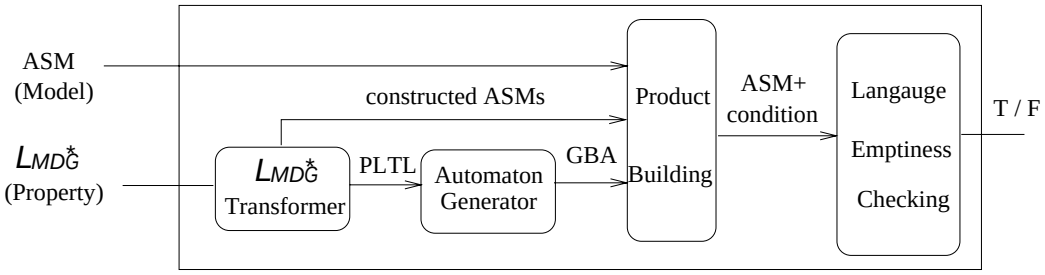

Figure 1. Structure of the MDG LEC Tool

```
  LangEmptyCheck (G, CL)    /* G = (X, Y, ∅, F_I, F_T, ∅), CL = (C_1, … , C_m) */
1 BEGIN
2       Old := ∅;
3       New := ReAn (G, True);
4       Do {
5       Old : = New;
6               For  C ∈ CL  {
7                       F_I := Conj (New, C);    /* Updating the initial states of G */
8                       New:= ReAn (G, True);
9                               }
10              New := New_state (New);
11              If (New = ∅) then return 'verified';
12       } until (PbyS (Old,New) = F)
13      If (PbyS (Old,New) = F) then return 'failed';
14 END
```

Figure 2. MDG LEC Algorithm

Table 1. Experimental Results with MDG LEC and MDG MC

Prop-erties	**MDG LEC**			**MDG MC**		
	CPU Time (Sec)	Memory (MB)	MDG Nodes	CPU Time (Sec)	Memory (MB)	MDG Nodes
P1	413	40	127518	492	42	143563
P2	399	39	124423	463	41	139970
P3	472	59	176500	742	60	189574
P4	476	60	178284	*	*	*
P5	455	50	153339	498	44	150377