DejaVu: A Monitoring Tool for First-Order Temporal Logic[*]

Klaus Havelund[1], Doron Peled[2], and Dogan Ulus[3]

[1]Jet Propulsion Laboratory, California Institute of Technology, USA
[2]Department of Computer Science, Bar Ilan University, Israel
[3]Verimag, Université Grenoble-Alpes, France

1 Introduction

Runtime Verification (RV) is aimed at analyzing individual execution traces and temporal behaviors observed from running programs and systems. Its traditional purpose is in detecting the lack of conformance with respect to a formal specification. While very early RV systems were based on specifications given in some form of propositional temporal logic, recent efforts have focused on monitoring so-called *parametric* specifications over events that carry data. Since a monitor for such specifications has to store observed data, the challenge is to have an efficient representation and manipulation of data. The fundamental problem is that the actual values of the data are not necessarily bounded or provided in advance.

In this paper, we describe our monitoring tool, DejaVu, which implements our algorithm [HPU17] for monitoring first-order past linear-time temporal logic over a sequence of events that carry data. We propose the use of Binary Decision Diagrams (BDDs) [Bry86] for representing and manipulating sets of observed data since (1) BDDs provide highly compact representations, (2) operations over BDDs, in particular complementation, are very efficient, and (3) the monitor construction for the propositional case shown in [HR02] naturally extends to BDDs. Our experiments show a substantial improvement in performance compared to a related tool.

2 Syntax and Semantics

First-Order Past Linear-time Temporal Logic (FO-PLTL) includes Boolean connectives, past temporal modalities PREVIOUSLY (●) and SINCE ($\mathcal{S}$), and the existential quantifier over data variables. The formulas of FO-PLTL are defined by the following grammar:

$$\varphi \;=\; \top \mid x = \lambda \mid \neg\,\varphi \mid \varphi_1 \vee \varphi_2 \mid \bullet\,\varphi \mid \varphi_1 \;\mathcal{S}\; \varphi_2 \mid \exists x.\;\varphi$$

where x is a variable over a data domain X and $x = \lambda$ is an atomic formula that denotes a dynamic test between a value assigned to x and the result of a function λ over current events. Other modalities ONCE (⧫) and HISTORICALLY (■) are derived as usual. Then, the satisfaction relation $(s, i, \gamma) \vDash \varphi$ indicates that the sequence s of events satisfies φ at the position i relative to an assignment γ over all free variables in φ. We inductively define the relation $\vDash$ as follows:

$$\begin{array}{lll}
(s,i,\gamma) \vDash x = \lambda & \leftrightarrow & (x \mapsto \lambda s_i) \in \gamma \\
(s,i,\gamma) \vDash \neg\,\varphi & \leftrightarrow & (s,i,\gamma) \nvDash \varphi \\
(s,i,\gamma) \vDash \varphi_1 \vee \varphi_2 & \leftrightarrow & (s,i,\gamma) \vDash \varphi_1 \text{ or } (s,i,\gamma) \vDash \varphi_2 \\
(s,i,\gamma) \vDash \bullet\,\varphi & \leftrightarrow & (s,i-1,\gamma) \vDash \varphi \\
(s,i,\gamma) \vDash \varphi_1 \;\mathcal{S}\; \varphi_2 & \leftrightarrow & \exists j \in [0,i].\ (s,j,\gamma) \vDash \varphi_2 \text{ and} \\
 & & \quad \forall k \in (j,i].\ (s,k,\gamma) \vDash \varphi_1 \\
(s,i,\gamma) \vDash \exists x.\ \varphi & \leftrightarrow & \exists a \in \mathit{dom}(x).\ (s,i,\gamma \cup \{(x \mapsto a)\}) \vDash \varphi
\end{array}$$

[*]The research performed by the first author was carried out at Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. The research performed by the second author was partially funded by Israeli Science Foundation grant 2239/15: "Runtime Measuring and Checking of Cyber Physical Systems". © 2018. All rights reserved.

3 DejaVu and Evaluation

The DejaVu system consists of a program *translate* : *Spec* $\to (\mathbb{E}^+ \to \mathbb{B}^+)$, which generates a *monitor* program when provided an FO-PLTL specification. The generated monitor takes as input a sequence of events (a trace), and returns a Boolean value for each position in the trace. The tool is implemented in SCALA using the standard approach where a parser parses the specification and produces an abstract syntax tree, which is then traversed and translated into the monitor program.

The performance of DejaVu is evaluated by comparing against MonPoly [BKMZ15] using three properties. The ACCESS property states that if a file is accessed by a user, then the user should have logged in and not yet logged out, and the file should have been opened and not yet closed. The FILE property states that if a file is closed, then it must have been opened (and not yet closed) with some mode (e.g. read or write). Finally, the FIFO property is a conjunction of four subformulas about data entering and exiting a queue.

Table 1: Performance comparison of DejaVu and MonPoly

Property	Trace Length	MonPoly (sec)	DejaVu (sec)
			bits per var.: 20 (40, 60)
ACCESS	11,006	**1.9**	**3.1** (3.3, 3.2)
	110,006	**241.9**	**6.1** (9.1, 10.9)
	1,100,006	**58455.8**	**36.8** (61.9, 88.8)
FILE	11,004	**61.1**	**2.8** (2.8, 3.0)
	110,004	**7348.7**	**6.3** (6.5, 8.6)
	1,100,004	**DNF**	**30.3** (43.9, 59.5)
FIFO	5,051	**158.3**	**195.4** (OOM, -)
	10,101	**1140.0**	**ERR** (-, -)

Table 1 shows the result of the evaluation. The properties were evaluated with each tool on traces of sizes up to 1 million events. Traces have the general form that initially numerous opening events (login, open, enter) occur, in order to accumulate a large amount of data stored in the monitor, after which a smaller number of corresponding closing events (logout, close, exit) occur. In addition, for each trace, we experimented with three different bit-vector sizes —20, 40 and 60 bits— for data encodings in BDDs, which corresponds to the ability to store approximately (and unnecessarily for these traces) a million, a trillion, and a quintillion different values for each variable, respectively. In the table, the following abbreviations are used: DNF = Did Not Finish (during 16 hours), OOM = Out of Memory, and ERR = an array index out of bound error in the BDD package.

We can see from Table 1 that with respect to the first two properties, DejaVu outperforms MonPoly by a factor up to 3000. Those are substantial differences, and demonstrates that BDDs may be an interesting way to refer to stored data. However, for the last FIFO property, the two systems are somewhat comparable, although MonPoly seems to do better on this particular property. Increasing the number of bits allocated per variable, from 20 to 40 and 60, does not seem to have a substantial impact on the performance, except for the FIFO property, where it causes an OOM result for 40 bits.

References

[BKMZ15] David Basin, Felix Klaedtke, Samuel Müller, and Eugen Zălinescu. Monitoring metric first-order temporal properties. *Journal of the ACM (JACM)*, 62(2):15, 2015.

[Bry86] Randal E Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on computers*, 100(8):677–691, 1986.

[HPU17] Klaus Havelund, Doron Peled, and Dogan Ulus. First-order temporal logic monitoring with BDDs. In *Formal Methods in Computer-Aided Design (FMCAD)*, 2017.

[HR02] Klaus Havelund and Grigore Rosu. Synthesizing monitors for safety properties. In *TACAS*, volume 2, pages 342–356. Springer, 2002.