Deciding Weak Monitorability for Runtime Verification

Zhe Chen, Yifan Wu, Ou Wei, and Bin Sheng
College of Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Nanjing, Jiangsu, China
zhechen@nuaa.edu.cn

ABSTRACT

An important problem in runtime verification is monitorability. If a property is not monitorable, then it is meaningless to check it at runtime, as no satisfaction or violation will be reported in finite steps. In this paper, we revisit the classic definition of monitorability, and show that it is too restrictive for practical runtime verification. We propose a weaker but more practical definition of monitorability, say *weak monitorability*, and show how to decide weak monitorability for runtime verification.

CCS CONCEPTS

• **Security and privacy** → *Logic and verification*; • **Theory of computation** → *Formalisms*; *Modal and temporal logics*; • **Software and its engineering** → *Software reliability*; *Software testing and debugging*;

KEYWORDS

runtime verification, dynamic analysis, weak monitorability, decision procedure

ACM Reference Format:
Zhe Chen, Yifan Wu, Ou Wei, and Bin Sheng. 2018. Deciding Weak Monitorability for Runtime Verification. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195077

1 INTRODUCTION

Runtime verification [5] deals with those verification techniques that construct monitors to observe an execution trace of a system, and check whether the trace satisfies or violates the given correctness properties, where a property specifies a set of admissible traces of the system [4]. Formally, a property is usually given as an LTL (Linear Temporal Logic) formula, an NBA (Nondeterministic Büchi Automaton) or an ω-regular expression, which actually represents an ω-regular language. Runtime verification checks whether a trace (i.e., a word) is a member of the language.

An important problem in runtime verification is *monitorability* [1, 3, 6]. Note that runtime verification checks whether a property is satisfied or violated after observing a *finite* trace. We say a property is *monitorable* if the satisfaction or violation of the property will probably be reported after observing a finite trace. However, note that a property usually specifies the admissible *infinite* traces of the system under consideration. This means, the property may never be satisfied or violated by a finite trace, unless all its extensions satisfy or violate the property. Thus, before running a system for runtime verification, we should first ensure the monitorability of the property. If a property is not monitorable, then it is meaningless to check it at runtime, as no satisfaction or violation will be reported in finite steps.

Pnueli and Zaks [6] were the first to formalize a notion of monitorability. Bauer et al. [1, 3] formalized a more complete notion of monitorability (Section 2). However, we will show that their definition is too restrictive for practical runtime verification (Section 3). Thus, we propose a weaker but more practical definition of monitorability, say *weak monitorability*, in Section 4, and show how to decide weak monitorability in Section 5.

2 MONITORABILITY

Definition 2.1 (Monitorability [1, 3]). Let Σ be an alphabet of symbols, $L \subseteq \Sigma^\omega$ be an ω-language. L is called

- *positively determined* by $u \in \Sigma^*$, if $u\Sigma^\omega \subseteq L$.
- *negatively determined* by $u \in \Sigma^*$, if $u\Sigma^\omega \cap L = \emptyset$.
- *u-monitorable* for $u \in \Sigma^*$, if $\exists v \in \Sigma^*$, s.t. L is positively or negatively determined by uv.
- *monitorable* if it is u-monitorable for every $u \in \Sigma^*$. □

In other words, L is *monitorable* if every finite word can be extended to a finite word that witnesses the satisfaction or violation of L. In fact, good prefixes can positively determine L, while bad prefixes can negatively determine L [2]. Thus, L is *monitorable* if every finite word can be extended to a good or bad prefix for L.

3 MONITORABILITY IS TOO RESTRICTIVE

The major objective of deciding the monitorability of an ω-language is to check whether the corresponding property is suitable for runtime verification. If an ω-language is non-monitorable, then the corresponding monitor will possibly never report satisfaction or violation of the property after finite steps. If an ω-language is monitorable, then the monitor will probably report satisfaction or violation after finite steps. However, this is also not guaranteed. That is, the monitor will also possibly never report satisfaction or violation after finite steps, because real executions may never generate good or bad prefixes. Thus, the only reason why we should use only monitorable properties is that their satisfaction or violation will "possibly" be reported after finite steps.

We believe that the definition of monitorability is too restrictive for practical runtime verification, because the set of monitorable ω-languages is so small that it excludes many useful properties whose

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05…$15.00
https://doi.org/10.1145/3183440.3195077

monitors will possibly report satisfaction or violation after finite steps in real executions. For example, the LTL formula $\boldsymbol{X}p \wedge \boldsymbol{GF}p$ is non-monitorable, as any finite word whose second symbol is p can never be extended to a good or bad prefix, due to the infinite continuations $\emptyset\emptyset\cdots$ and $pp\cdots$ respectively. This means, the monitor will never report satisfaction or violation after finite steps for these finite words. However, if the second symbol is not p in real executions, then the monitor will report violation after two steps. Thus, this property should be better recognized as monitorable, as it can also provide useful information for verification.

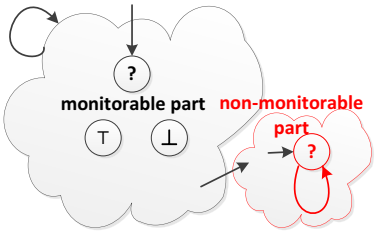

Figure 1: The monitor of a non-monitorable ω-language

Besides the above example, the definition of monitorability actually rules out a large set of useful properties. If an ω-language is non-monitorable, then the corresponding monitor [3] must be in the form shown in Fig. 1. The state space of the monitor must consist of two parts: the monitorable part containing the states from which $\top$ or $\bot$ states (denoting satisfaction and violation, resp.) are reachable, and the non-monitorable part containing the states from which $\top$ and $\bot$ states are not reachable. The definition of monitorability actually rules out all monitors that have a non-monitorable part. However, note that the state space of a monitor could be very large, but the non-monitorable part may be placed in a corner that could possibly never be reached in real executions, i.e., real executions always stay in the monitorable part. In this case, the monitor will possibly report satisfaction or violation after finite steps. Thus, it should be better recognized as monitorable, as it can provide useful information for verification.

Formally, the definition of monitorability rules out all ω-languages that are u-monitorable only for some (but not all) finite words $u \in \Sigma^*$. However, the finite words that are not u-monitorable could possibly never be observed in real executions, i.e., real executions always generate u-monitorable finite words. In this case, the monitor will possibly report satisfaction or violation after finite steps. Thus, these ω-languages should be better recognized as monitorable. In essence, we should relax the "universal quantifier" restriction to obtain a weaker but more practical definition of monitorability.

4 WEAK MONITORABILITY

We propose weak monitorability as follows.

Definition 4.1 (Weak monitorability). L is called *weakly monitorable* if it is u-monitorable for some $u \in \Sigma^*$. □

In other words, L is *weakly monitorable* if some finite word can be extended to a finite word that witnesses the satisfaction or violation of L. In terms of good and bad prefixes, L is *weakly monitorable* if some finite word can be extended to a good or bad prefix for L. As a corollary, we can show that L is *weakly monitorable* if there exists a good or bad prefix for L.

It is easy to show that the set of weakly monitorable ω-languages strictly includes the set of monitorable ω-languages, as any monitorable ω-language is also weakly monitorable. But the converse does not hold. For example, the LTL formula $\boldsymbol{X}p \wedge \boldsymbol{GF}p$ is non-monitorable, as the finite word pp can never be extended to a good or bad prefix, due to the infinite continuations $\emptyset\emptyset\cdots$ and $pp\cdots$ respectively. But it is weakly monitorable, as the finite word ϵ or p can be extended to a bad prefix $p\emptyset$ but never a good prefix. This means, the monitor will probably report the violation of the property after finite steps for these finite prefixes, but never report the satisfaction. The LTL formula $\boldsymbol{GF}p$ is non-monitorable, and is also weakly non-monitorable, as any finite word can never be extended to a good or bad prefix.

5 DECISION PROCEDURE

Using the monitor construction procedure developed by Bauer et al. [3], the weak monitorability of an LTL formula ϕ can be determined by checking whether a $\top$ or $\bot$ state is reachable from the initial state of the resulting monitor. Because such a state is reachable iff there exist good or bad prefixes, thus ϕ is weakly monitorable. This checking procedure can be done in linear time. However, note that this checking procedure is based on the monitor construction procedure that requires 2ExpSpace. Thus, when an ω-regular language L is given in terms of an LTL formula, the weak monitorability problem of L can be decided in 2ExpSpace.

Let us consider other representations of ω-regular languages. If L is given as an NBA, we first have to explicitly complement the NBA, and the rest of the procedure stays the same. However, the complement operation also involves an exponential blowup. If L is given as an ω-regular expression, we first have to build an NBA for the expression, which can be done in polynomial time, and the rest of the procedure stays the same as the case of NBA. Hence, independent of the concrete representation, the weak monitorability problem of an ω-regular language can be decided in 2ExpSpace.

ACKNOWLEDGMENTS

This work is supported by the Joint Research Funds of National Natural Science Foundation of China and Civil Aviation Administration of China (No. U1533130).

REFERENCES

[1] Andreas Bauer. 2010. Monitorability of *omega*-regular languages. *CoRR* abs/1006.3638 (2010).
[2] Andreas Bauer, Martin Leucker, and Christian Schallhart. 2007. The Good, the Bad, and the Ugly, But How Ugly Is Ugly?. In *Proceedings of the 7th International Workshop on Runtime Verification, RV 2007 (Lecture Notes in Computer Science)*, Vol. 4839. Springer, 126–138.
[3] Andreas Bauer, Martin Leucker, and Christian Schallhart. 2011. Runtime Verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 20, 4 (2011), 14.
[4] Zhe Chen, Zhemin Wang, Yunlong Zhu, Hongwei Xi, and Zhibin Yang. 2016. Parametric Runtime Verification of C Programs. In *Proceedings of the 22nd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2016) (Lecture Notes in Computer Science)*, Vol. 9636. Springer, 299–315.
[5] Martin Leucker and Christian Schallhart. 2009. A brief account of runtime verification. *Journal of Logic and Algebraic Programming* 78, 5 (2009), 293–303.
[6] Amir Pnueli and Aleksandr Zaks. 2006. PSL Model Checking and Run-Time Verification Via Testers. In *Proceedings of the 14th International Symposium on Formal Methods, FM 2006 (Lecture Notes in Computer Science)*, Vol. 4085. Springer, 573–586.