A Model for Collaborative Runtime Verification (Extended Abstract)

Bas Testerink
Utrecht University
Utrecht, The Netherlands
B.J.G.Testerink@uu.nl

Nils Bulling
Delft University of Technology
Delft, The Netherlands
N.Bulling@tudelft.nl

Mehdi Dastani
Utrecht University
Utrecht, The Netherlands
M.M.Dastani@uu.nl

ABSTRACT

Runtime verification concerns checking whether a system execution satisfies a given property. In this paper we propose a model for collaborative runtime verification where a network of local monitors collaborates in order to verify properties of the system. A local monitor has only a local view on the execution of the system; thus, it can verify a specific system property with respect to its local view. However, the local monitor can also receive inputs from other local monitors about the evaluation of the properties that they are trying to verify. This information can be combined to allow the verification of more complex properties. A network built from such collaborating local monitors is called a *collaborative monitor*.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Formal Methods

General Terms

Runtime Verification, Security

Keywords

Collaborative Monitoring, Runtime Verification

1. INTRODUCTION

Verifying a given property of a system means checking whether the system's behavior satisfies that property. The verification of multi-agent systems such as traffic and economic markets is challenging and it is not always possible to use offline verification techniques due to their complexity or, in case of model checking techniques, the lack of a sufficient model of the system. In some cases we can verify systems at runtime by verifying properties of a running system by observing the execution trace. However, runtime verification also has limitations as noted and analyzed in [2]. E.g. the evaluation of a property is always based on a finite execution trace. A decision must be taken without knowing the future evolution of the system. Moreover, monitoring a running system is often a decentralized process. There is not always a central point for data collection which further complicates the detection of erroneous behavior. A decentralized framework for runtime monitoring systems has been proposed in [1].

In this paper we propose a model for collaborative runtime verification where a network of local monitors verifies properties of a system in a collaborative way. We call such a network a *collaborative monitor*. For the structure of collaborative monitors we draw inspiration from decentralized monitoring techniques such as wireless sensor networks (WSNs)[3]. Like sensor nodes in WSNs, local monitors in our model have a local view. This view allows a monitor to evaluate a local property. Additionally, local monitors share the evaluations of properties with each other. Based on aggregation methods known from WSNs, a local monitor can then combine the evaluation of its local property with the ones received from its peers. This allows local monitors to verify more complex properties without revealing too much information as only the evaluations of the properties (true, false, unknown) but not all the local information of local monitors, e.g. the property themselves, are communicated.

Collaborative runtime monitors can be applied to various scenarios such as the surveillance of traffic and computer networks. However, in such environments adversaries may try to attack the monitor, e.g. by eavesdropping the communication. WSNs are well analyzed in terms of robustness, security risks and appropriate countermeasures (cf. [4]). We aim at developing safe and robust collaborative runtime verification systems by drawing inspiration from the safety and robustness analyses of WSNs.

2. COLLABORATIVE VERIFICATION

We use linear-time temporal logic (LTL) [5] with the three-valued Boolean finite LTL semantics of [2] to specify and verify system properties. LTL is well-known for being suitable for the specification of properties in the context of (runtime) verification [6]. For illustration, consider the formula $\varphi = G\psi$ which specifies the property: ψ is always true. The decentralized runtime monitoring framework of [1] is constructed from local monitors which can only observe the truth value of a predefined subset of propositional variables. Clearly, this restricts the set of LTL properties which can be verified by local monitors. To some extent this limitation is overcome by allowing local monitors to communicate their observations in the form of a (rewritten) LTL formula towards their peers. A disadvantage of this approach is that the rewritten formulae carry structured information which can be exploited by attacks on the network. The decen-

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

tralized LTL monitoring framework of [7] is similar to the framework above, but truth values of propositional variables rather than rewritten formulae are shared. The key difference to [1] is that the local monitors have a predefined, in general not fully connected, communication topology. As a consequence, a local monitor can in general not know the truth of all propositional variables.

In this abstract, we present a model that is related to [7] and [1]. The observation capabilities of a local monitor are captured by an LTL formula, instead of a set of observable propositional variables. The intuition is that the monitor can observe the truth of the formula for each finite execution trace of the system. Local monitors also receive the truth values of LTL formulae from neighboring local monitors. An important difference to the frameworks of [7] and [1] is the kind of information that is communicated. In our model communicated information has less structure. A local monitor combines the received inputs, using an aggregation function, with the evaluation of the property it tries to verify. As a consequence, it is able to verify a more complex property by only receiving structureless inputs, without knowing how these inputs were computed. Then, the evaluation of the complex property of a local monitor is shared with other local monitors. We define the aggregation function as a Boolean function. Hence, the aggregated evaluation shared by a local monitor corresponds to a (fixed) Boolean combination of LTL formulae as is detailed below.

Collaborative monitors. More formally, we define a *local monitor* by $m = (f, \varphi)$, where f is a Boolean function over k Boolean variables ($k \geq 1$) called *aggregation function*, and φ is an LTL formula, called *observation formula*. A *collaborative monitor* is specified by $C = (M, \mathsf{com})$, where M is a non-empty set of local monitors and $\mathsf{com} : M \to 2^M$ is a function that returns, given a local monitor m, the local monitors from which m receives evaluations of properties.

Given a collaborative monitor $C = (M, \mathsf{com})$ and a local monitor $m = (f, \varphi) \in M$, the observation formula φ is evaluated by m at runtime on the finite execution trace of a system. All monitors from $\mathsf{com}(m)$ send their aggregated evaluations to m. Then, m aggregates these inputs obtained from its peers in $\mathsf{com}(m)$ together with the current evaluation of its observation formula φ by means of the aggregation function f. Thus, the output of f corresponds to the evaluation of a complex LTL formula of which the specific structure is not known to m. We refer to this complex, implicitly given formula by Q_m. The evaluation of Q_m proceeds recursively. We require that com does not contain cycles between local monitors to ensure that the computation stops.

Given this restriction, the evaluation is computed as follows. All local monitors $m \in M$ in a collaborative monitor $C = (M, \mathsf{com})$ evaluate their complex property Q_m every time the system makes a transition. Firstly, the local monitors m without neighbors, i.e. those with $\mathsf{com}(m) = \emptyset$, compute Q_m and communicate the evaluation to their neighbors. Next, each monitor $m' \in M$ that has received evaluations from all monitors in $\mathsf{com}(m')$ uses its aggregation function to compute an aggregated evaluation which is, in turn, shared with all monitors $m'' \in M$ with $m' \in \mathsf{com}(m'')$ that are expecting an input from m'. This procedure is repeated for all local monitors.

As an example, consider a collaborative monitor $C = (\{m_1, m_2\}, \mathsf{com})$ at a parking lot where $m_1 = (f_1, \varphi_1)$, $m_2 = (f_2, \varphi_2)$, $\mathsf{com}(m_1) = \{m_2\}$ and $\mathsf{com}(m_2) = \emptyset$. C's purpose is to verify whether an agent a does not need to pay (because a has a disabled placard or has a parking permit), or pays before leaving. m_1 is located at the parking lot's exit, its observation formula is $\varphi_1 = (\neg leave_a)\mathcal{U}pay_a$ (reading: agent a does not leave until it has payed). m_2 is located at the parking spaces and can scan front windows. Its observation formula is $\varphi_2 = placard_a \vee permit_a$ (reading: agent a has a disabled placard, or a parking permit). f_2 returns given the evaluation of φ_2 the same evaluation, hence $\mathsf{Q}_{m_2} = \varphi_2$. f_1 returns given the evaluation of φ_1 and Q_{m_2} the evaluation of their disjunction, hence $\mathsf{Q}_{m_1} = \mathsf{Q}_{m_2} \vee \varphi_1$. Thus, given a finite trace σ, Q_{m_1} is true if at the initial state of σ agent a has a disabled placard, or a parking permit, or does not leave before paying. Note that m_1 can verify Q_{m_1} without being aware, e.g., whether agent a is disable or not. Hence if communication between m_1 and m_2 is intercepted, then this information cannot be obtained—privacy of a is ensured.

3. FUTURE WORK AND CONCLUSIONS

Arguably, formal analyses of collaborative monitors and aspects related to robustness and security is important in order to design many real world multi-agent systems. We aim to investigate these issues by taking inspiration from a related field on decentralized monitoring: wireless sensor networks. As a first step, we proposed in this abstract a collaborative runtime verification model using LTL as the specification language. A local monitor verifies properties by observing the system and by communicating with its peers. A collaborative monitor consists of such collaborating local monitors and allows to verify more complex properties. At the same time it takes into account security issues by aggregating information, which hides structured data from possible attackers.

REFERENCES

[1] A. Bauer and Y. Falcone. Decentralised LTL monitoring. In D. Giannakopoulou and D. Méry, editors, *Formal Methods 2012*, volume 7436 of *LNCS*, pages 85–100. Springer Berlin Heidelberg, 2012.

[2] A. Bauer, M. Leucker, and C. Schallhart. Runtime verification for LTL and TLTL. *ACM Trans. Softw. Eng. Methodol.*, 20(4):14:1–14:64, Sept. 2011.

[3] B. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proc. of the 22nd Int.Conf. on Distr. Comp. Systems Workshops*, pages 575–578, 2002.

[4] A. Pathan, H.-W. Lee, and C. S. Hong. Security in wireless sensor networks: issues and challenges. In *The 8th Int. Conf. of Advanced Communication Technology*, volume 2, pages 1043–1048, Feb 2006.

[5] A. Pnueli. The temporal logic of programs. In *Proc. of the 18th Annual Symposium on Foundations of Comp. Sci.*, pages 46–57, Oct 1977.

[6] K. Y. Rozier. Survey: Linear temporal logic symbolic model checking. *Comput. Sci. Rev.*, 5(2):163–203, May 2011.

[7] B. Testerink, M. Dastani, and J.-J. Meyer. Norm monitoring through observation sharing. In A. Herzig and E. Lorini, editors, *Proceedings of the European Conference on Social Intelligence*, pages 291–304, 2014.