Runtime Analysis with R2U2: A Tool Exhibition Report

Johann Schumann[1], Patrick Moosbrugger[2], and Kristin Y. Rozier[3]

[1] SGT, Inc., NASA Ames, Moffett Field, CA, USA, Johann.M.Schumann@nasa.gov
[2] Vienna University of Technology, Vienna, Austria, moosbrugger@cps.tuwien.ac.at
[3] Iowa State University, Ames, IA, USA, kyrozier@iastate.edu

Abstract. We present R2U2 (Realizable, Responsive, Unobtrusive Unit), a hardware-supported tool and framework for the continuous monitoring of safety-critical and embedded cyber-physical systems. With the widespread advent of autonomous systems such as Unmanned Aerial Systems (UAS), satellites, rovers, and cars, real-time, on-board decision making requires unobtrusive monitoring of properties for safety, performance, security, and system health. R2U2 models combine past-time and future-time Metric Temporal Logic, "mission time" Linear Temporal Logic, probabilistic reasoning with Bayesian Networks, and model-based prognostics.

The R2U2 monitoring engine can be instantiated as a hardware solution, running on an FPGA, or as a software component. The FPGA realization enables R2U2 to monitor complex cyber-physical systems without any overhead or instrumentation of the flight software. In this tool exhibition report, we present R2U2 and demonstrate applications on system runtime monitoring, diagnostics, software health management, and security monitoring for a UAS. Our tool demonstration uses a hardware-based processor-in-the-loop "iron-bird" configuration.

1 Introduction and Tool Overview

The Realizable, Responsive, Unobtrusive Unit (R2U2) is a framework for runtime System Health Management (SHM) of cyber-physical systems. R2U2 is unique in that it combines several different runtime reasoning "building blocks" to provide a more effective runtime analysis than can be accomplished via any one of them alone; [10] and [11] give an overview of the building block architecture and provide ideas and examples for tool configurations. Building blocks include temporal logic runtime observers, Bayes Net (BN) decision-makers, and sensor filters; the framework is extensible in that it is easy to connect the inputs and outputs of different types of reasoning blocks. Other notable advantages of R2U2 are its zero-overhead hardware implementation, dual-encodings of temporal logic observers to include both time- and event-triggered results, implementations of future-time and past-time observers, and efficient use of Bayesian reasoning over observer outputs to provide temporal diagnostics.

R2U2 reasons efficiently about temporal behaviors using temporal logic runtime observers. These observers encode Metric Temporal Logic (MTL) [5] and Mission-Time Linear Temporal Logic (LTL) [6] formulas. MTL adds discrete time bounds to the temporal operators of LTL formulas; for R2U2 we bound operators in units of ticks of the system clock, so a singular bound of `[100]` designates the operator holds for the next 100 clock ticks and a paired bound of `[5,20]` designates that the operator holds from

5 to 20 clock ticks from now. We defined Mission-Time LTL [6] in recognition that many requirements for missions of air- and spacecraft, for example, are most naturally written in LTL but there is an (often unspecified) assumption that the eventualities guaranteed by strong operators ($\Diamond$ and $\mathcal{U}$) are fulfilled during the mission. Therefore, we consider such formulas to be in Mission-Time LTL, where we automatically fill in MTL-like time bounds on eventualities to give an appropriate finite-trace semantics that guarantees satisfaction during the current mission, or mode of flight. Uniquely, R2U2 encodes every future-time temporal logic specification twice: once as an asynchronous observer and once as a synchronous observer. Asynchronous, or event-triggered, observers return a verdict ($true$ or $false$) in the first clock-tick that the formula can be evaluated. Their output is a tuple including the clock-tick(s) they have a verdict for and that verdict, where the clock-tick(s) may be in the past in the case of future-time formulas for which there was not previously sufficient information to evaluate fully. Asynchronous observers resemble traditional runtime monitors with one important difference: they always report both success *and* failure of the formula (rather than just reporting failures) as both evaluations provide valuable information to influence the probabilistic evaluations of the BNs. Synchronous, or time-triggered, observers return a three-valued verdict ($true$, $false$, or $maybe$) at every tick of the system clock. This is useful to provide intermediate information for probabilistic BN reasoning as well as a “liveness” check that the monitoring framework is responsive. We defined and proved correct FPGA-based implementations of asynchronous and synchronous runtime observers [6].

R2U2 expands upon the failure reporting of traditional runtime monitors to provide advanced diagnostics by combining the temporal logic observers with light-weight Bayesian Networks (BNs) that reason over the observer outputs and (possibly filtered) sensors signals. Our R2U2 model can have modular, usually rather small Bayesian networks for groups of highly-related faults that might occur for one hard- or software component. We designed and experimentally evaluated efficient FPGA-based encodings of our BNs in [4], demonstrating their ability to perform efficient diagnostics for safety and performance requirements. Recognizing that violations of security properties that occur through tampering with sensor inputs may also have unique temporal patterns, we expanded on this work with a series of case studies for UAS in [8]. A possibly innocuous off-nominal reading or event, followed by a specific temporally-displaced pattern of behavior is often indicative of a hard-to-diagnose security threat, such as dangerous MAV (Micro Air Vehicle) commands, ground station denial-of-service attempts, or GPS spoofing; [8] defines and demonstrates R2U2 configurations that efficiently diagnose these during runtime.

2 Tool Architecture

In its usual configuration, R2U2 obtains data from sensors, actuators, and the flight software using a read-only (serial) interface (Fig. 1A). This enables R2U2 to continuously monitor multiple signals during runtime with minimal instrumentation of the flight software. Altering safety-critical software or hardware components can cause difficulties maintaining flight certification. R2U2 itself is implemented in VHDL that is compiled into an FPGA configuration. For our experiments, we use an Adapteva Parallella board [1] that provides a suitable FPGA and runs a Linux system for data logging and development. Software-only versions of R2U2 are available and can be executed on any Linux-based system, preferably on a separate hardware unit to avoid interaction with the flight software and hardware.

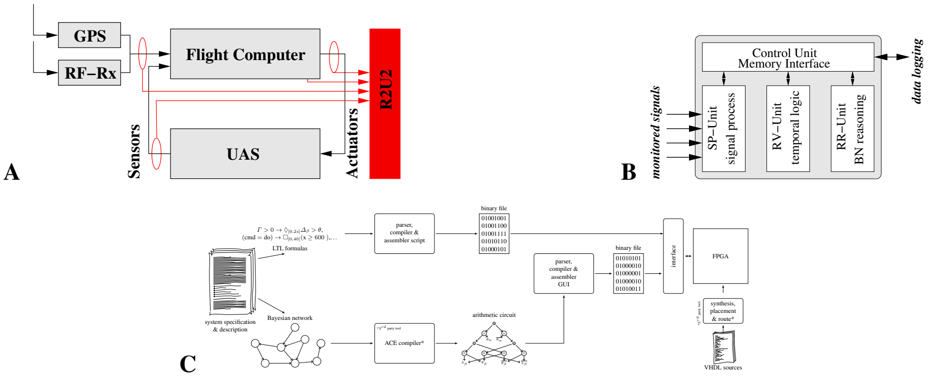

Fig. 1. A: Schematics of R2U2 for a small UAS. **B**: R2U2 architecture **C**: R2U2 tool chain

R2U2 models consist of temporal logic formulas, Bayesian networks, and specifications of signal-preprocessing and filtering. These models can be designed in a modular and hierarchical manner to enable the designer to easily express properties containing temporal, model-based, and probabilistic aspects. For graphical modeling of the Bayesian networks, we use the freely available tool SamIam [2]. With the other parts of the model in textual format, our tool-chain (Fig. 1C) compiles temporal formulas and Bayesian network reasoners into a compact and efficient binary format. The compiled model then can be directly downloaded onto the R2U2 execution engine without having regenerate code or configuration, which could take considerable time for an FPGA.

MTL and LTL formulas are compiled into code for a special purpose processor that is instantiated on the FPGA or emulated in software. Efficient and correct algorithms for the temporal operators [6] avoid the construction of potentially large finite state machines. The Bayesian network is compiled into an arithmetic circuit [3], which can be efficiently evaluated in bounded time using a special purpose processor on the FPGA. Filtering and thresholding of the (floating-point) input signals is done by the SP-Unit. Fig. 1B shows the high-level architecture of the R2U2 engine. All algorithms of R2U2 are fully static, do not require any dynamic structures or memory allocation, and have known and bounded runtime behavior, making the tool suitable for execution on embedded architectures.

3 Examples and Applications

R2U2 has been used for UAS to continuously monitor numerous properties and perform root cause analysis [4]. These properties typically address safety ("Is the airspeed always higher than the stall-speed?"), performance ("Have we reached our desired waypoint within 10s of ETA?"), or security ("Has our GPS system be spoofed?").

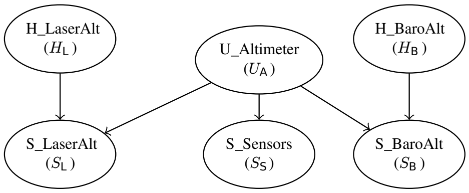

Fig. 2. Sensor failure detection BN from [6]

For example, the relationship property "A pitch-up should cause the UAS to climb within 5s" can be expressed by the following MTL formula: $\Box(pitch^{up} \rightarrow \Diamond_{[0,5]}(\Box_{[2]}(v_z^b > 20\,\text{ft/min})))$, where v_z^b is the vertical speed measured by the baro-altimeter. Here, we have refined the requirement that within the last 5 seconds, we have to encounter at least a 2second stretch of uninterrupted climbing in order to filter out short-term effects like turbulence.

Checking the consistency of several sensors can be an important help to figure out if a sensor is broken, and if so, which one. In our example (see [6]), the UAS is equipped with a barometric altimeter, a laser altimeter, and an inertial measurement unit (IMU) for navigation. Because of sensor noise, it would be hard to directly compare the values. We rather abstract the readings from each sensor into "climbing" and "descending". We feed these data to the sensor nodes of our the Bayesian network model (Fig. 2, bottom row). Given this information, R2U2 can calculate, in real-time, the posteriors of the health nodes (H_LaserAlt and H_BaroAlt) indicating their most likely health status. This Bayesian network allows us to incorporate domain knowledge (e.g., the laser altimeter is more likely to fail than the barometric altimeter) and complex interrelationships between components. For details of this example see [6,7].

The tool demonstration website [7] contains a number of relevant examples illustrating the monitoring of safety and performance properties, monitoring a UAS for possible cyber-attacks [8], and incorporating battery prognostics [9]. We will demonstrate multiple examples with R2U2 on our "iron-bird," which contains the Arduino flight hardware including sensors and servos, and the Parallella board with R2U2 running on FPGA or in software.

4 Summary

R2U2 is designed for continuous runtime analysis of safety-critical and embedded cyber-physical systems, for example, UAS. The modeling framework uses a synergistic combination of past- and future-time MTL, mission-time LTL, Bayesian Networks, and prognostics models. The R2U2 framework and tool is demonstrated on our UAS iron-bird, a processor-in-the-loop setup for a small UAS. R2U2 can be instantiated on an FPGA or as a software application and can be used for monitoring safety, security, and performance properties, as well as performing diagnostics for wide ranges of software and cyber-physical systems.

Detailed information about R2U2, documentation, examples, and demo scripts can be found at [7]; we are in the application process for a NASA Open Source License.

Acknowledgments The development of R2U2 was in part supported by NASA ARMD grant NNX14AN61A, ARMD 2014 I3AMT Seedling Phase I NNX12AK33A, and NRA NNX08AY50A.

References

1. Adapteva: The Parallella System. `http://adapteva.com` (2016)
2. Automated Reasoning Group, UCLA: SamIam Sensitivity Analysis, Modeling, Inference and More (SamIam). `http://reasoning.cs.ucla.edu/samiam/` (2016)
3. Darwiche, A.: A differential approach to inference in Bayesian networks. Journal of the ACM 50(3), pp. 280–305 (2003)
4. Geist, J., Rozier, K.Y., Schumann, J.: Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems. In: Proc. RV15. LNCS, vol. 8734, pp. 215–230. Springer (2014)
5. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-time systems 2(4), pp. 255–299 (1990)
6. Reinbacher, T., Rozier, K.Y., Schumann, J.: Temporal-logic based runtime observer pairs for system health management of real-time systems. In: Proc. 20th TACAS. LNCS, vol. 8413, pp. 357–372. Springer (2014)
7. Schumann, J., Moosbrugger, P., Rozier, K.Y.: Runtime Analysis with R2U2: A Tool Exhibition Report (Tool Demonstration Website) `http://temporallogic.org/research/RV16/` (2016)
8. Schumann, J., Moosbrugger, P., Rozier, K.Y.: R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems. In: Proc. RV15. LNCS, vol. 9333, pp. 233–249. Springer (2015)
9. Schumann, J., Roychoudhury, I., Kulkarni, C.: Diagnostic Reasoning using Prognostic Information for Unmanned Aerial Systems In: Proc. PHM2015. (2015)
10. Schumann, J., Rozier, K.Y., Reinbacher, T., Mengshoel, O.J., Mbaya, T., Ippolito, C.: Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems. In: Proc. PHM2013. pp. 381–401 (2013)
11. Schumann, J., Rozier, K.Y., Reinbacher, T., Mengshoel, O.J., Mbaya, T., Ippolito, C.: Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems. Int. Journal of Prognostics and Health Management (IJPHM) 6(1), pp. 1–27 (2015)