On the Similarities of Aircraft and Humans: Monitoring CPS with StreamLAB

Jan Baumeister*, Bernd Finkbeiner*, Maximilian Schwenger*, Hazem Torfah*
*Reactive Systems Group
Saarland University
Saarbrücken, Germany
{*jbaumeister, finkbeiner, schwenger, torfah*}@react.uni-saarland.de

***Abstract*—Modern medicine increasingly relies on medical cyber-physical systems for diagnosis and treatment of patients. Yet, misbehavior can have dire consequences raising the need for formal guarantees on the runtime behavior. Static verification would provide such guarantees, but is infeasible due to the complexity of the systems and the human physiology. In these cases, more light-weight verification techniques such as runtime monitoring are still applicable. The monitor analyzes a single execution of the system and raises an alarm as soon as it detects unexpected behavior. The shape of unexpected behavior is described in a formal specification language such as the one provided by the StreamLab framework, which can express complex real-time constraints. Yet the language is sufficiently restrictive to allow for static analyses determining an upper bound on the memory consumption. As a result, the underlying monitoring framework StreamLab can synthesize an embedded runtime monitor on an FPGA. The practicality of this approach is validated by current case studies on avionics. Moreover, the low memory and energy consumption indicate that a deployment on implantable devices is possible. In this abstract we thus showcase the practicality of StreamLab in the medical domain and suggest it as a suitable candidate for runtime monitoring on medical devices.**

Medical cyber-physical systems (MCPS) are ubiquitous, ranging from large magnetic resonance imaging scanners to implantable artificial cardiac pacemakers. While they constitute an indisputable advancement in diagnosis and treatment, even minor mistakes can lead to unforeseen ramifications. This can be seen in the Therac-25[1] incident. A quick handling of the machine by experienced technicians triggered a data race that lead to severe radiation poisoning in patients, causing multiple cases of debilitation and three deaths. Formal verification of the system's control software could have prevented the incident. However, the complexity of such systems coupled with incomplete knowledge about the patient's physiology and the exact workings of the human body render a static verification infeasible. Moreover, the control logic is often based on empirical data. As an example, consider the detection and subsequent treatment of seizures based on brain waves collected in an electroencephalogram (EEG). Precise identification and description of seizure patterns can be a hard task even for specialists who rely on their long experience. Yet, treatment of epilepsy requires manual tweaking of parameters until they fit the patient. The resulting controllers perform reasonably well, but are based on the doctor's experience and thus hard to explain and reason about. Similarly, recent successes in whole-brain seizure detection with a trained classifier [1] indicate that machine learning can help in seizure detected. However, the resulting controller is similarly hard to explain, as per usual for machine learned components. Static verification requires insight into the decision logic, so such incomprehensible controllers exacerbate the verification process further. While control logic based on empirical data — either designed by an experienced doctor or machine learned — performs well in practice, it is neigh impossible to verify it. However, controllers that are simple enough to verify perform poorly in practice.

The avionics industry faces a similar problem. Software based on machine learning performs extraordinarily well but cannot be verified statically. This hinders the certification of the aircraft and thus prevents it from getting a permission to fly. As a remedy, dynamic verification techniques such as runtime monitoring are commonly used. Here, a specification written in a language with formal semantics describes the desired behavior of a system on an abstract level. A monitor then assesses the system's state based on sensor values. As soon as the specification is violated, an alarm is raised and the system can initiate mitigation measures such as switching to a less effective, verified controller. The monitor treats the control logic as a blackbox: it sees *which* decision

This work was partially supported by the European Research Council (ERC) Grant OSARES (No. 683300) and by the DFG as part of TRR 248 (No. 389792660)

[1]https://www.bugsnag.com/blog/bug-day-race-condition-therac-25

```
input  CSL: Float
input  rec, stim: Bool
output twich := abs(derive(3,CLS))
output avg_long @100mHz := twich.aggregate(over:2000s, using:avg)
output avg_short @1kHz := twich.aggregate(over:2ms, using:avg)
output spike := avg_short - avg_long.hold() > ε
trigger @1kHZ spike ∧ ¬rec.aggregate(over:2ms, using:any) "seizure not recognized"
trigger @1kHZ rec.aggregate(over:5ms, using:any) ∧ ¬stim.aggregate(over:3ms, using:any)
                                                    "stimulation not triggered"
```

Figure 1: A simplified RTLOLA specification to monitor an implantable responsive neurostimulator.

has been made but does not require information on *how* it was made. There are already efforts in transferring runtime verification techniques to the medical domain: Chen et al. [2] monitored a glucose control unit to regulate the insulin infusion and Abbas et al. [3] monitored EKGs to detect cardiac arrhythmias.

We propose using the STREAMLAB [4] framework on MCPS, which has been applied to unmanned aerial vehicles by the German Aerospace Center (DLR). STREAMLAB revolves around the stream-based specification language RTLOLA [5] in which samples of sensors are modeled as input streams and transformed into output streams containing statistical information. These are used to identify undesired behavior. The language design focusses on providing simple primitives for complex computations, such as a sliding real-time window over an input stream computing the discrete integral. RTLOLA has a formal semantics allowing for formal guarantees on the runtime behavior as well as static analyses determining the space and time requirements to monitor the specification. These analyses enable a compilation of a specification into a VHDL monitor that is synthesizable on a field-programmable gate array (FPGA) [6]. As a result, the monitor can be deployed on embedded devices such as drones or medical implants.

We showcase the expressiveness of the language by presenting a significantly simplified specification that attempts to monitor an implantable responsive neurostimulator for epilepsy treatment. The RTLOLA specification in Figure 1 monitors whether a seizure was correctly recognized based on the EEG of a single cortical channel and whether a stimulation was triggered subsequently. The first line declares the readings of the cortical strip lead as input stream as 64-bit wide floating point number. The second line declares boolean streams indicating the recognition of a seizure and an ongoing stimulation, respectively. Line 3 computes the absolute twitch, i.e., the third derivation of the amplitude measured over the CSL. The next two lines compute the running long-term and short-term average twitch over 2000s and 2ms, respectively. Both streams are annotated with a computation frequency: `avg_long` is computed every 10 s and `avg_short` every 1 ms. A spike is then defined as a signification difference between the long- and the short-term average twitch, indicating the onset of a seizure. Lastly, triggers declare desirable properties; violations are reported to the controller. The first trigger checks whether a spike led to a recognition signal, the second trigger asserts that a recognition leads to a stimulation.

The specification showcases the simplicity of RTLOLA specifications; even complex computations can be expressed in a concise way. Moreover, the interpretation of the specification requires only 3 µs per event on general-purpose hardware. Coupled with the recently presented compilation to an FPGA, we want to initiate a discussion regarding the applicability of STREAMLAB on medical cyber-physical systems.

REFERENCES

[1] P. Fergus, A. Hussain, D. Hignett, D. Al-Jumeily, K. Abdel-Aziz, and H. Hamdan, "A machine learning system for automated whole-brain seizure detection," *Applied Computing and Informatics*, vol. 12, no. 1, pp. 70 – 89, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210832715000022

[2] S. Chen, O. Sokolsky, J. Weimer, and I. Lee, "Data-driven adaptive safety monitoring using virtual subjects in medical cyber-physical systems: A glucose control case study," *JCSE*, vol. 10, no. 3, 2016. [Online]. Available: https://doi.org/10.5626/JCSE.2016.10.3.75

[3] H. Abbas, R. Alur, K. Mamouras, R. Mangharam, and A. Rodionova, "Quantitative regular expressions for monitoring cardiac arrhythmias," in *MT@CPSWeek 2018*, 2018, pp. 1–2. [Online]. Available: https://doi.org/10.1109/MT-CPS.2018.00007

[4] P. Faymonville, B. Finkbeiner, M. Schledjewski, M. Schwenger, M. Stenger, L. Tentrup, and H. Torfah, "Streamlab: Stream-based monitoring of cyber-physical systems," in *CAV 2019*, 2019, pp. 421–431. [Online]. Available: https://doi.org/10.1007/978-3-030-25540-4_24

[5] P. Faymonville, B. Finkbeiner, M. Schwenger, and H. Torfah, "Real-time stream-based monitoring," *CoRR*, vol. abs/1711.03829, 2017. [Online]. Available: http://arxiv.org/abs/1711.03829

[6] J. Baumeister, B. Finkbeiner, M. Schwenger, and H. Torfah, "FPGA stream-monitoring of real-time properties," in *EMSOFT 2019*, 2019.