Robust Monitoring for Medical Cyber-Physical Systems

Bernd Finkbeiner
CISPA Helmholtz Center for Information Security
Saarland Informatics Campus
Saarbrücken, Germany
finkbeiner@cispa.saarland

Andreas Keller
Center for Bioinformatics
Saarland Informatics Campus
Saarbrücken, Germany
andreas.keller@ccb.uni-saarland.de

Jessica Schmidt
Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
schmidt-jessica@stud.uni-saarland.de

Maximilian Schwenger
CISPA Helmholtz Center for Information Security
Saarland Informatics Campus
Saarbrücken, Germany
maximilian.schwenger@cispa.saarland

ABSTRACT

Some medical implants act autonomously: they assess the current health status of a patient and administer treatment when appropriate. An improper treatment, however, can cause serious harm. Here, the decision logic leading to the treatment relies on data obtained from sensors — an inherently imperfect medium. Coping with these inaccuracies requires the logic to be *robust* in the sense that slight perturbations in the measurements do not significantly alter the decision. Determining the extent to which an algorithm is robust automatically does not scale well for complex and opaque components. This is particularly problematic when machine learning is involved. Yet, the analysis is feasible for simpler safety-related components such as a runtime monitor, which observes the system and intervenes in a treatment when necessary. Its significantly lower complexity generally allows for providing static guarantees on the runtime behavior of the monitor. Complementing these guarantees with a robustness analysis constitutes a major step toward certifiable medical cyber-physical systems controlled by opaque, machine-learned components. Hence, this paper reports on ongoing research in the direction of a robustness analysis for the runtime monitoring framework RTLola.

CCS CONCEPTS

• **Computer systems organization** → **Reliability**; *Real-time system specification*; • **Hardware** → *Safety critical systems*;

KEYWORDS

Medical Cyber-Physical Systems, Runtime Verification, Robustness, Stream-based Monitoring

ACM Reference Format:
Bernd Finkbeiner, Andreas Keller, Jessica Schmidt, and Maximilian Schwenger. 2021. Robust Monitoring for Medical Cyber-Physical Systems. In *Proceedings of Medical CPS*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Cyber-physical systems (CPS) use sensors to perceive their surroundings. These are inherently imperfect both in terms of reliable timing and precision. Not only are single readings subject to noise, *outlier*-readings can fail to reflect reality entirely. This is especially relevant in a safety-critical context, such as for medical cyber-physical systems (MCPS). These systems need to be robust with respect to these inaccuracies such that measurement errors cannot be responsible for improper treatment. As an example, consider an artificial pancreas (AP) [6], which is equipped with a sensor measuring the blood glucose level. A single reading of $200\,\mathrm{dL\,mg^{-1}}$ pre- and succeeded by readings ranging from 70 to $80\,\mathrm{dL\,mg^{-1}}$ does not warrant intervention. The component's susceptibility to misbehave under such erroneous inputs is captured by a notion of *robustness* [18]. A high level of robustness indicates that subtle changes in the input do not significantly alter the output.

Measuring the robustness is a computationally challenging task. The complexity and opaqueness of controllers for CPS render this task infeasible in general, particularly when machine learning is involved. A substantially less complex, yet essential component is a runtime monitor. The monitor is a separate component of the MCPS which takes sensor measurements and control decisions to assess the current state of the system. If this information indicates that the system is close to an unsafe state, the monitor intervenes. Concretely, it can prohibit an AP from falsely administering a dose of insulin.

The monitor is a more light-weighted component compared to the controller itself. While the controller has to *find* an appropriate reaction to the situation at hand, the monitor merely *validates* the decision. This allows a framework like RTLola [10] to employ a transparent and mathematically rigorous approach to monitoring

This work was partially supported by the German Research Foundation (DFG) as part of the Collaborative Research Center Foundations of Perspicuous Software Systems (TRR 248, 389792660), by the European Research Council (ERC) Grant OSARES (No. 683300), and by the Aviation Research Programm LuFo of the German Federal Ministry for Economic Affairs and Energy as part of "Volocopter Sicherheits-Technologie zur robusten eVTOL Flugzustands-Absicherung durch formales Monitoring" (No. 20Q1963C)..

Medical CPS, 18. May 2021, Virtual

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

even in the complex setting of CPS. It is based on the RTLola specification language with formal semantics. The framework compiles a specification into either a hardware description language [2] with traceability annotations or a high-level programming language with annotations enabling an automatic verification [11]. This lays a foundation for comprehensible and thus certifiable monitoring. The language also enables an analysis that provides several guarantees on the runtime behavior of the monitor, such as an upper bound on the memory consumption. As a result, an RTLola monitor is able to complement an opaque controller with mathematically sound static guarantees.

This work-in-progress paper reports on an extension of the suite of static analyses for RTLola specifications. For this, it first presents a formal definition of ε-δ-Robustness for stream-based languages such as RTLola. Intuitively, it requires that an ε perturbation distributed over the set of sequences of input values alters the output only by at most δ. The input perturbation follows a malicious *resource distribution model*, i.e., arbitrarily many data points may be changed provided the difference with respect to either the L_1 or L_∞ metric remains below ε. The second part of the work concerned with devising an SMT encoding of the robustness property for an RTLola specification. This allows an SMT solver such as z3 to automatically determine the robustness of the specification. A preliminary empirical evaluation reveals that the running time of the solver scales sufficiently well.

While the resource distribution model is particularly suited for dealing with random noise, it is not the only sensible one. A continuation of this work also considers the *outlier model*, which allows for a certain number of outliers, independent of their absolute deviation from reality. A third model is the *natural model*. Here, changes are not spurious and abrupt as are noise and outliers, but gradual. This models a temporary but natural deviation from the norm, such as the time leading up to a re-calibration of a sensor. Figure 1 illustrates all three models. Note, that the choice depends on the specifications: a natural increase might indicate a critical situation and hence warrant an immediate response from the system. Thus, robustness concerning natural perturbations is undesired.

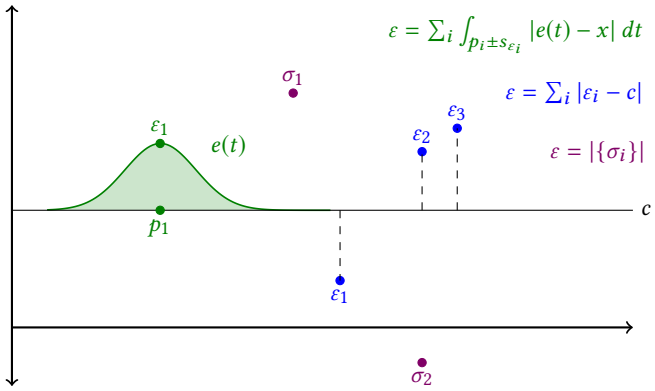

Figure 1: Illustration of three input perturbation models: resource allocation model in blue, outlier model in purple, and natural model in green. This paper presents the resource distribution model.

1.1 Related Work

For safety-critical systems, trustworthiness plays an increasingly important role. This includes traceability [1], certifiability [3], verification [15], and robustness. The former two are particularly hard when neural networks are involved [5, 25], which happens increasingly more often [16, 26]. Verification thereof is still in its early stages with successes in autonomous driving [19], computer vision [20], and chaotic, continuous systems [23] and aircraft traffic control [12]. Robustness is a complementary approach. It was investigated for hybrid systems [22], i.e., systems with a mix of discrete and continuous components, and internet-of-things devices [14].

Analyzing the robustness of a runtime monitor scales significantly better than analyzing the system itself. These monitors usually operate with respect to a formal specification language such as a temporal logic. Dokhanchi et al. [9] monitor the robustness of the past fragment of metric temporal logic [13]. In contrast to that, Mascle et al. [17] monitor a variant of linear temporal logic [21] which incorporates robustness in its language [28]. Their notion of robustness, however, diverges from the one presented in this paper. In their work a system is robust if a mild violation of the assumptions on the input only translate to mild violations of the guarantees provided by the system.

Other noteworthy robustness-related analyses are fuzzing [4] and reliability metrics. The former executes the system at hand with different inputs, subjecting them to slight systematic changes between executions and compares their outputs. While this scales well, it requires the system as a whole to be repeatedly executable under controlled conditions and does not provide formal guarantees. Reliability metrics such as *mean time between failure* and *mean time to failure* [27] provide insight into the expected failure rates of a system. This analysis is on a more concrete level than the one presented in this paper as it is concerned with factor like hardware decay.

2 ROBUST RTLOLA

RTLola is a stream-based runtime monitoring framework revolving around a formal specification language of the same name. It analyzes such a specification and generates an executable monitor plus static guarantees on its runtime behavior. The RTLola framework is a suitable target for a robustness analysis since it is specifically designed for safety-critical CPS focusing on comprehensiveness and certifiability.

An RTLola specification consists of input and output streams, as well as trigger declarations. An input stream represents a typed data source of the monitor, such as a sensor. Output streams declare how to filter and process input data to assess the state of the system accurately. Lastly, triggers are boolean conditions upon which satisfaction the monitor raises an alarm. This alarm can lead to the initiation of emergency landings in aircraft or the intervention regarding a spurious treatment in MCPS.

Consider the following excerpt of a specification for an AP.

```
input glucose: UInt64
output clean_glucose := if glucose < 300
    then glucose else clean_glucose.last(dft: 90)
trigger clean_glucose > 120 ∨ clean_glucose < 60
```

The specification declares an input stream containing the current blood glucose level. The output stream `clean_glucose` filters suspected measurement-errors out of the `glucose` input. For this, it uses a conditional statement producing either the currently measured level or the last clean value. If such a last value does not exist, because the monitor was just started, then it uses the default value instead. The trigger issues an alarm if the glucose level indicates hyper- or hypoglycemia.

When disregarding the trigger, the specification is intuitively robust, as illustrated in Figure 2.[1]

2.1 Robustness

The definition of robustness for an RTLola specification is based on its models. Note that in terms of models, triggers constitute boolean outputs, allowing us to refrain from treating them differently.

Definition 2.1 (Models and Validity). A *model* $M = (\mathbb{I}, \mathbb{O})$ of an RTLola specification Φ is a set of finite sequences of values. Each sequence $V^* \in \mathbb{I} \cup \mathbb{O}$ represents either an input or an output stream where V is the domain of RTLola values. As a short-hand notation, let $in(M) = \mathbb{I}$, $out(M) = \mathbb{O}$. Furthermore, let $in(\Phi)$ and $out(\Phi)$ provide the set of input and output streams of Φ, respectively. Finally, $\Phi(\mathbb{I}) = \mathbb{O}$ denotes that a monitor for the specification Φ generates $\mathbb{O}$ for the input sequences $\mathbb{I}$. A model M is *valid* for Φ, i.e., $M \models \Phi$ iff $\Phi(in(M)) = out(M)$.

Robustness states that mild variations in $\mathbb{I}$ only slightly alter $\mathbb{O}$. This requires a notion of distance for finite sequences. Hence, let $d\colon V^* \times V^* \to \mathbb{R}^+$ be a metric on vectors such as the L_1 or L_∞ metrics. The multi-vector extension thereof is $d^*\colon (V^*)^n \times (V^*)^n \to \mathbb{R}^+$ for some $n \in \mathbb{N}$ with $d^*(\tau, \tau') = \sum_{i=1}^{n} d(\tau_i, \tau_i')$.

Definition 2.2 (ε-δ-Robustness). A specification Φ is ε-δ-robust iff for any two models M and M' with $M \models \Phi$ and $M' \models \Phi$:

$$d^*(in(M), in(M)) \leq \varepsilon \implies d^*(out(M), out(M')) \leq \delta$$

Recall the example specification. When determining its robustness, it is evident that a change in the input stream will either be reflected identically in the output stream or be ignored entirely. The latter case results in an ε-0-robustness, i.e., the monitor is fully robust, whereas the former leads to an ε-ε-robustness. Hence, the overall robustness disregarding the trigger is an ε-ε-robustness.

Remark. *Note that boolean input and outputs — in particular triggers — require a substantially different treatment than numeric streams. The remainder of the paper only considers numeric values until Section 5 outlines how to lift this restriction.*

[1] Note that the sample trace illustrates the robustness, not necessarily a healthy glucose-history.

3 SMT ENCODING

This section devices an SMT encoding for the robustness analysis of an RTLola specification. The encoding is parameterized by the bound ε and computes two models M and $\overline{M}$. The length of each stream is bounded by another input parameter n. The encoding ensures that the input distance $d^*(in(M), in(\overline{M}))$ remains below ε while the output distance $d^*(out(M), out(\overline{M}))$ is maximized. To this end, it generates a fresh variable for each stream and position within the stream. Suppose $\mathbb{I}$ and $\mathbb{O}$ contain an id for each stream. Then, for any stream s, the variable v_η^s represents its η-th value with $0 < \eta \leq n$.

The next step is to encode the output stream expressions. For this, let enc_η be a function that generates an SMT formula for the η-th evaluation of an RTLola output stream expression. Here, arithmetical and logical expressions, which include conditionals, are straight-forward to encode. Stream accesses are replaced by the respective variable representing the stream at a certain point in time. As an example, suppose the expression of stream s requires access to the last value of stream s'. Then, $enc_\eta(s)$ replaces the stream access by $v_{\eta-1}^{s'}$. Note that in case $\eta - 1 = 0$, the encoding uses the default value declared in the expression instead.

The last component of the encoding incorporates the distance function. For this, it introduces a fresh variable δ. The translation of the actual metric into SMT is straight-forward.

The final encoding requires two models, M and $\overline{M}$. Let variables and the expression encoding for $\overline{M}$ carry the same decoration.

$$\max \delta \; s.t.$$

$$\underbrace{\delta = \sum_{i \in out(\Phi)} d(v^{s_i}, \overline{v}^{s_i})}_{(1)} \wedge \underbrace{\sum_{i \in in(\Phi)} d(v^{s_i}, \overline{v}^{s_i}) \leq \varepsilon}_{(2)} \wedge$$

$$\underbrace{\left(\bigwedge_{i \in out(\Phi)} \bigwedge_{1 \leq \eta \leq n} v_\eta^{s_i} = enc_\eta(s_i) \right)}_{(3)} \wedge \underbrace{\left(\bigwedge_{i \in out(\Phi)} \bigwedge_{1 \leq \eta \leq n} \overline{v}_\eta^{s_i} = \overline{enc}_\eta(s_i) \right)}_{(4)}$$

Encoding the running example for $in(\Phi) = \{c\}$, $out(\Phi) = \{g\}$, $n = 2$, and some ε results in the following:

$$\begin{aligned}
(1) &= d(v^c, \overline{v}^c) \\
(2) &= d(v^g, \overline{v}^g) \leq \varepsilon \\
(3) &= \left(v_1^c = enc_1(c)\right) \wedge \left(v_2^c = enc_2(c)\right) \\
&= \left(v_1^c = \mathit{if}\ v_1^g < 300\ \mathit{then}\ v_1^g\ \mathit{else}\ 90\right) \\
&\quad \wedge \left(v_2^c = \mathit{if}\ v_2^g < 300\ \mathit{then}\ v_2^g\ \mathit{else}\ v_g^2\right) \\
(4) &= \left(\overline{v}_1^c = \overline{enc}_1(c)\right) \wedge \left(\overline{v}_2^c = \overline{enc}_2(c)\right) \\
&= \left(\overline{v}_1^c = \mathit{if}\ \overline{v}_1^g < 300\ \mathit{then}\ \overline{v}_1^g\ \mathit{else}\ 90\right) \\
&\quad \wedge \left(\overline{v}_2^c = \mathit{if}\ \overline{v}_2^g < 300\ \mathit{then}\ \overline{v}_2^g\ \mathit{else}\ \overline{v}_2^g\right)
\end{aligned}$$

Remark. *Note that the encoding is parameterized in both ε and n. The latter is necessary because RTLola specifications can be cyclic as a stream can access its own past values. In this case, well-definedness*

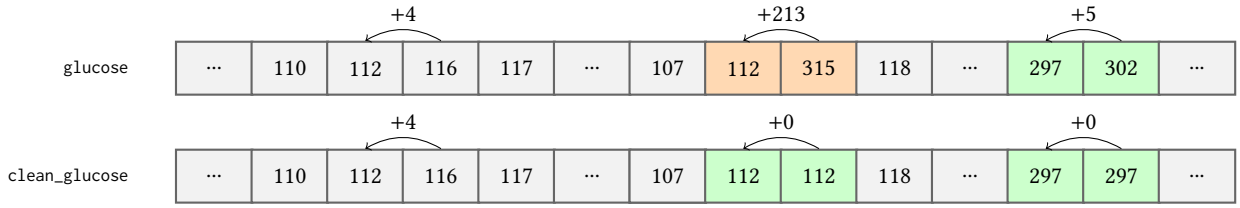

Figure 2: Sample input and output trace for the running example specification. When the glucose readings are in a "sane" range, the output stream mirrors the input. Values over the threshold of 300 are classified as outliers and thus ignored. Hence, perturbations in the input result in lower or equal alterations in the output.

stems from the default values provided to `last` *[7, 24]. Hence, the encoding unrolls the cyclic computation up to n. For some specifications, the unrolling can find a fixed point, so increasing n cannot reveal a greater level of robustness.*

3.1 Interpretation of the SMT Output

If the SMT solver terminates, it will always yield a positive result since an RTLola specification describes a computable, total function. A result of $\delta = 0$ proves that the output of the monitor is independent of the input. Note that a stream in such a specification can still have different values in different positions; it can, for example, count the number of times it was evaluated. Other than that, the output is $\delta = \alpha\varepsilon$ where α is a polynomial of degree $\leq n$ where each coefficient is a linear combination of scalars occurring in the specification. If the degree is less than n, further increasing n cannot lead to greater values of δ. If the degree is exactly n, then $\lim_{n\to\infty} \delta$ can be ∞ or be bound by a polynomial of a greater degree than n.

4 EVALUATION

Even though the work is not fully mature, an evaluation allows for estimating the feasibility for varying values of n and different kinds of specifications. The prototype implementation is written in Rust and uses the open-source frontend of the RTLola framework[2]. It uses the z3[3] [8] SMT solver as a backend.

All experiments were conducted on an Intel® Core™ i7-9750H CPU running at 2.60 GHz. The resource consumption when computing the encoding of specifications was entirely negligible. Figure 3 reports the running times plotted against n for two different specifications. Each dot is the average over 100 repetitions, superimposed by a Lowess trend line. The underlying specification for the plot on the left-hand side is robust and monitors an artificial pancreas similar albeit more complex than the running example. The running time never exceeded 40 s for the L_1 metric and 100 s for the L_∞ metric. When compared to the plot on the right-hand side, the impact of n was relatively low. The reason behind this is that its underlying specification is not robust, i.e., δ is unbounded. Lastly, note that the choice of ε has no measurable impact on the running time.

The bottom line of the evaluation is that the running time significantly worsens for non-robust specifications. Hence, determining whether a specification is robust for *some* δ before-hand can improve the applicability of the approach. However, even without this information, the running time is sufficiently small to encourage a continuation of the work.

[2]Project: rtlola.org, Frontend: https://crates.io/crates/rtlola-frontend
[3]Frontend: https://crates.io/crates/z3

5 OUTLOOK

The algorithm outlined in this paper allows for analyzing the robustness of specifications composed of a subset of RTLola. There are two major limitations, which will be addressed in future work: asynchrony and proper treatment of non-numeric values.

5.1 Asynchrony

One major advantage of RTLola over its predecessor Lola [7] is asynchrony. In Lola, every input stream receives a new value simultaneously, prompting the monitor to re-compute every output stream. In RTLola, however, the monitor can receive partial inputs, i.e., only *some* inputs receive a new value. As a result, the monitor only evaluates the relevant subset of outputs. Moreover, some outputs are computed *periodically*, independent of the arrival of input values.

The incorporation of periodic streams into the SMT encoding is straight-forward. Asynchronous arrivals, however, severely increase the degree of freedom for the SMT solver: each new value is time-stamped, and between two values of a stream, there may be an arbitrary amount of new values on another stream. Without some limitations on the temporal behavior of streams, the SMT solver might be unable to terminate within a reasonable time.

5.2 Robustness of Boolean Values

The computation of the robustness of the running example disregarded the trigger due to its boolean nature. Nevertheless, the specification contains three threshold checks, two in the trigger and one in the conditional expression. The former are fragile by nature: a glucose reading of 60 dL mg^{-1} is considered perfectly fine, whereas for any $\varepsilon \in \mathbb{R}^+$, $60 - \varepsilon$ raises the alarm. This is intended as the threshold precisely determines when a value is supposed to be classified as problematic. Hence, the fragility of the range check should not negatively influence the robustness. However, the other check influences the numerically-valued output stream, rendering it relevant for the robustness of the specification. Identifying which boolean checks need to be ignored and which do not is subject of the ongoing work.

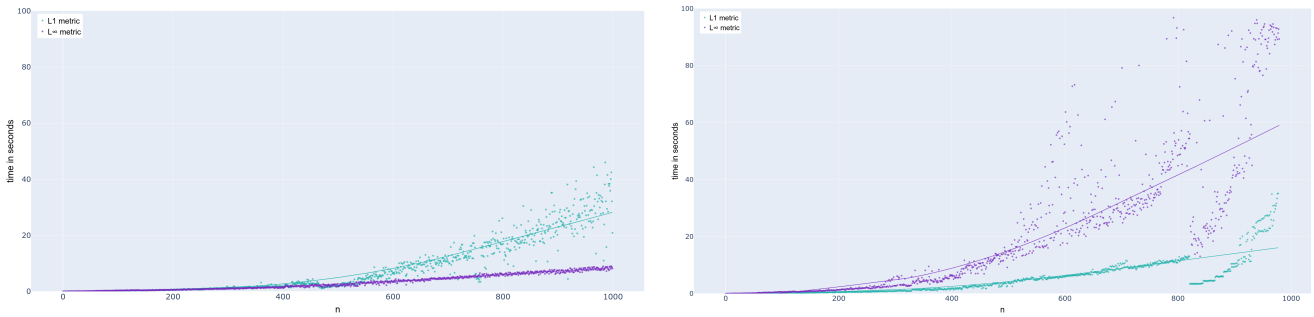

Figure 3: Plots reporting on the running time as a function on n. The underlying specification for the plot on the left-hand side is robust whereas the one for the right-hand side plot is not. Each dot is the average running time over 100 repetitions. Superimposed: Lowess trend line.

5.3 Conclusion

Robustness is an important metric for safety-critical systems such as MCPS. Determining the robustness of an RTLola specification is a worthwhile goal since the framework focuses on improving the certifiability of CPS particularly in presence of machine-learned components. Past successes regarding integrating RTLola monitors into low-resource environments such as autonomous drones make it an intriguing candidate when designing MCPS. The promising results of the preliminary experimental evaluation suggest that extensions to the encoding will still scale sufficiently well.

REFERENCES

[1] Vincent Aravantinos. Traceability of deep neural networks. *CoRR*, abs/1812.06744, 2018.

[2] Jan Baumeister, Bernd Finkbeiner, Maximilian Schwenger, and Hazem Torfah. Fpga stream-monitoring of real-time properties. *ACM Transactions on Embedded Computing Systems (TECS)*, 18(5s):1–24, 2019.

[3] DF Bedford, G Morgan, and J Austin. Requirements for a standard certifying the use of artificial neural networks in safety critical applications. In *Proceedings of the international conference on artificial neural networks*. Citeseer, 1996.

[4] Sofia Bekrar, Chaouki Bekrar, Roland Groz, and Laurent Mounier. Finding software vulnerabilities by smart fuzzing. In *Fourth IEEE International Conference on Software Testing, Verification and Validation, ICST 2011, Berlin, Germany, March 21-25, 2011*, pages 427–430. IEEE Computer Society, 2011.

[5] Chih-Hong Cheng, Frederik Diehl, Yassine Hamza, Gereon Hinz, Georg Nührenberg, Markus Rickert, Harald Ruess, and Michael Truong-Le. Neural networks for safety-critical applications - challenges, experiments and perspectives. *CoRR*, abs/1709.00911, 2017.

[6] Claudio Cobelli, Eric Renard, and Boris Kovatchev. Artificial pancreas: Past, present, future. *Diabetes*, 60:2672–82, 11 2011.

[7] Ben D'Angelo, Sriram Sankaranarayanan, César Sánchez, Will Robinson, Bernd Finkbeiner, Henny B. Sipma, Sandeep Mehrotra, and Zohar Manna. LOLA: runtime monitoring of synchronous systems. In *12th International Symposium on Temporal Representation and Reasoning (TIME 2005), 23-25 June 2005, Burlington, Vermont, USA*, pages 166–174. IEEE Computer Society, 2005.

[8] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[9] Adel Dokhanchi, Bardh Hoxha, and Georgios E. Fainekos. On-line monitoring for temporal logic robustness. In Borzoo Bonakdarpour and Scott A. Smolka, editors, *Runtime Verification - 5th International Conference, RV 2014, Toronto, ON, Canada, September 22-25, 2014. Proceedings*, volume 8734 of *Lecture Notes in Computer Science*, pages 231–246. Springer, 2014.

[10] Peter Faymonville, Bernd Finkbeiner, Malte Schledjewski, Maximilian Schwenger, Marvin Stenger, Leander Tentrup, and Hazem Torfah. Streamlab: Stream-based monitoring of cyber-physical systems. volume 11561 of *Lecture Notes in Computer Science*, pages 421–431. Springer, 2019.

[11] Bernd Finkbeiner, Stefan Oswald, Noemi E. Passing, and Maximilian Schwenger. Verified rust monitors for lola specifications. In Jyotirmoy Deshmukh and Dejan Nickovic, editors, *Runtime Verification - 20th International Conference, RV 2020, Los Angeles, CA, USA, October 6-9, 2020, Proceedings*, volume 12399 of *Lecture Notes in Computer Science*, pages 431–450. Springer, 2020.

[12] Jean-Baptiste Jeannin, Khalil Ghorbal, Yanni Kouskoulas, Ryan Gardner, Aurora Schmidt, Erik Zawadzki, and André Platzer. Formal verification of acas x, an industrial airborne collision avoidance system. In *2015 International Conference on Embedded Software (EMSOFT)*, pages 127–136. IEEE, 2015.

[13] Ron Koymans. Specifying real-time properties with metric temporal logic. *Real Time Syst.*, 2(4):255–299, 1990.

[14] Florian Kriebel, Semeen Rehman, Muhammad Abdullah Hanif, Faiq Khalid, and Muhammad Shafique. Robustness for smart cyber physical systems and internet-of-things: From adaptive robustness methods to reliability and security for machine learning. In *2018 IEEE Computer Society Annual Symposium on VLSI, ISVLSI 2018, Hong Kong, China, July 8-11, 2018*, pages 581–586. IEEE Computer Society, 2018.

[15] Francesco Leofante, Nina Narodytska, Luca Pulina, and Armando Tacchella. Automated verification of neural networks: Advances, challenges and perspectives. *CoRR*, abs/1805.09938, 2018.

[16] Alexander Selvikvåg Lundervold and Arvid Lundervold. An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127, 2019.

[17] Corto Mascle, Daniel Neider, Maximilian Schwenger, Paulo Tabuada, Alexander Weinert, and Martin Zimmermann. From LTL to rltl monitoring: improved monitorability through robust semantics. In Aaron D. Ames, Sanjit A. Seshia, and Jyotirmoy Deshmukh, editors, *HSCC '20: 23rd ACM International Conference on Hybrid Systems: Computation and Control, Sydney, New South Wales, Australia, April 21-24, 2020*, pages 7:1–7:12. ACM, 2020.

[18] C McPhail, HR Maier, JH Kwakkel, M Giuliani, A Castelletti, and S Westra. Robustness metrics: How are they calculated, when should they be used and why do they give different results? *Earth's Future*, 6(2):169–191, 2018.

[19] Sina Mohseni, Mandar Pitale, Vasu Singh, and Zhangyang Wang. Practical solutions for machine learning safety in autonomous vehicles. *arXiv preprint arXiv:1912.09630*, 2019.

[20] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Towards practical verification of machine learning: The case of computer vision systems. *CoRR*, abs/1712.01785, 2017.

[21] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 46–57. IEEE Computer Society, 1977.

[22] Matthias Rungger and Paulo Tabuada. A notion of robustness for cyber-physical systems. *IEEE Trans. Autom. Control.*, 61(8):2108–2123, 2016.

[23] Karsten Scheibler, Leonore Winterer, Ralf Wimmer, and Bernd Becker. Towards verification of artificial neural networks. In Ulrich Heinkel, Daniel Kriesten, and Marko Rößler, editors, *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, MBMV 2015, Chemnitz, Germany, March 3-4, 2015*, pages 30–40. Sächsische Landesbibliothek, 2015.

[24] Maximilian Schwenger. Let's not Trust Experience Blindly: Formal Monitoring of Humans and other CPS. Master thesis, Saarland University, 2019.

[25] Mark Sendak, Michael Gao, Marshall Nichols, Anthony Lin, and Suresh Balu. Machine learning in health care: A critical appraisal of challenges and opportunities. *eGEMs*, 7(1), 2019.

[26] Jose Roberto Ayala Solares, Francesca Elisa Diletta Raimondi, Yajie Zhu, Fatemeh Rahimian, Dexter Canoy, Jenny Tran, Ana Catarina Pinho Gomes, Amir H Payberah, Mariagrazia Zottoli, Milad Nazarzadeh, et al. Deep learning for electronic health records: A comparative review of multiple deep neural architectures. *Journal of Biomedical Informatics*, 101:103337, 2020.

[27] Scott Speaks. Reliability and mtbf overview. *Vicor reliability engineering*, 2010.

[28] Paulo Tabuada and Daniel Neider. Robust linear temporal logic. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 10:1–10:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.