What are the fundamental software abstractions for designing reliable cyber-physical systems operating in uncertain environments?

David Broman[1,2] and Jim Woodcock[3]

[1]KTH Royal Institute of Technology, Stockholm, Sweden; [2]Stanford University, California, USA and [3]University of York, York, UK

Context

Cyber-physical systems (CPS) combine computations in embedded systems with physical dynamics, typically interconnected using networks. Predictability and reliability of combined software and hardware systems are crucial for CPS design (Lee, 2008), but all parts in a real CPS are not necessarily predictable. Specifically, operating in a physical environment introduces different aspects of uncertainty, including uncertainties in sensors, the physical dynamics of the system itself and interactions with other systems or humans. There is a large body of knowledge about reasoning with uncertainty in fields and topics such as statistics, automatic control and sensor fusion. However, how to program for uncertainty in complex cyber-physical systems needs to be better understood. What are the fundamental software abstractions? How do they compose and interact?

Time, timeliness and functional correctness are vital CPS and real-time systems concepts. Many formalisms, languages and tools exist for CPS (Carreira et al., 2020; Broman et al., 2012), including programming and modelling both the cyber part, as well as the physical part. However, there needs to be scientifically proven and practically evaluated approaches that combine software and platform aspects (execution time, scheduling and hardware properties) with methods for handling uncertainties in the physical environment. Probabilistic machine learning (Ghahramani, 2015) and probabilistic programming (Goodman, 2013) have emerged as promising directions in a Bayesian setting. However, these approaches have yet to be combined (in a principled way) with the hardware and software aspects of CPS. This research question concerns finding, formalising and evaluating such fundamental software abstractions, including but not limited to programming models, programming language abstractions, operating systems abstractions and modelling abstractions.

If you believe you can contribute to answering this Question with your research outputs find out how to submit in the Instructions for authors (https://www.cambridge.org/core/journals/research-directions-cyber-physical-systems/information/author-instructions/preparing-your-materials). This journal publishes Results, Analyses, Impact papers and additional content such as preprints and "grey literature." Questions will be closed when the editors agree that enough has been published to answer the Question so before submitting, check if this is still an active Question. If it is closed, another relevant Question may be currently open, so do review all the open Questions in your field. For any further queries check the information pages (https://www.cambridge.org/core/journals/research-directions-cyber-physical-systems/information/about-this-journal) or contact this email (cps@cambridge.org).

References

Broman D, Lee EA, Tripakis S and Törngren M (2012) Viewpoints, formalisms, languages, and tools for cyber-physical systems. In *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*, pp. 49–54.

Carreira P, Amaral V and Vangheluwe H (2020) Multi-paradigm modelling for cyber-physical systems: foundations. In *Foundations of Multi-Paradigm Modelling for Cyber-Physical Systems*. Cham: Springer, pp. 1–14.

Ghahramani Z (2015) Probabilistic machine learning and artificial intelligence. *Nature* **521**, 7553, 452–459. https://doi.org/10.1038/nature14541.

Goodman ND (2013) The principles and practice of probabilistic programming. *ACM SIGPLAN Notices* **48**, 1, 399–402. https://doi.org/10.1145/2480359.2429117.

Lee EA (2008) Cyber physical systems: design challenges. In *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*. IEEE. pp. 363–369.