Research Directions: Cyber-Physical Systems

www.cambridge.org/cbp

Time-Sensitive Software

Edward A. Lee[1] and Jim Woodcock[2]

[1]UC Berkeley, Berkeley, USA and [2]University of York, York, UK

Question

Cite this article: Lee EA and Woodcock J (2023). Time-Sensitive Software. *Research Directions: Cyber-Physical Systems*. **1**, e1, 1–3. https://doi.org/10.1017/cbp.2023.1

Received: 28 January 2023
Accepted: 1 February 2023

Author for correspondence: Edward A. Lee, Email: eal@berkeley.edu

Context

Timing of software execution is usually considered a *performance* property rather than a *correctness* property. But in software for cyber-physical systems, timing is often a critical feature of the execution of the software. Today, no widely used programming language specifies timing. Instead, timing is an emergent consequence of a particular implementation and is sensitive to every detail of the hardware on which the software runs and to what other software may be sharing the same hardware. Even a small change in the hardware or software context can lead to drastically different timing behaviour, making maintenance and upgrades difficult.

This question seeks contributions that turn this picture around. Just as a programmer delegates to the compiler and the microprocessor corrects execution of the program logic, we seek ways to similarly delegate the delivery of timing requirements. The contributions can include programming language enhancements, compilation techniques, innovative computer architectures, modelling methods and formal analysis techniques.

Some problems to consider include:

- How to model time (e.g., discrete, dense, super-dense; totally or partially ordered; linear vs. branching time; logical vs. physical time; hard vs. soft real time).
- How to analyse temporal properties (e.g., temporal logics, explicit-time logics, fundamental limits; temporal calculi).
- Programming language constructs (e.g., mechanisms for concurrency and composition; synchronous vs. asynchronous; comparisons of timing constructs in legacy programming languages; priorities vs. explicit timing requirements).
- Static analysis techniques (e.g., execution time analysis; feasibility analysis).
- Operating-system capabilities (e.g., real-time operating systems; clock synchronisation; distributed systems coordination; scheduling).
- Measurement and evaluation (e.g., testing frameworks; benchmarking; repeatability).

How to contribute to this Question

If you believe you can contribute to answering this Question with your research outputs, find out how to submit them 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).

Competing interests. The authors declare none.

© The Author(s), 2023. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

References

On models of time:

1. **Lee I and Davidson S** (1987) Adding time to synchronous process communications. *IEEE Transactions on Computers*.
2. **Stankovic JA** (1988) Misconceptions about real-time computing: a serious problem for next-generation systems. *Computer*, **21**, 10–19.
3. **Alur R and Henzinger T (1991)** Logics and models of real time: A survey. In REX Workshop, Mook, The Netherlands, J. W. De Bakker, C. Huizing, W. P. De Roever and G. Rozenberg (Eds.), June 3–7 , vol. LNCS 600, Berlin/Heidelberg: Springer, pp. 74–106.
4. **Abadi M and Lamport L** (1994) An old-fashioned recipe for real time. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, **16**, 1543–1571.
5. **Alur R and Dill DL** (1994) A theory of timed automata. *Theoretical Computer Science*, **126**, 183–235.
6. **Thiele L, Chakraborty S and Naedele N** (2000) Real-time calculus for scheduling hard real-time systems. In International Symposium on Circuits and Systems (ISCAS), vol. 4, pp. 101–104.

7. **Broy M**, Refinement of time. *Theoretical Computer Science*, **253**, 3–26.
8. **Kaynar DK, Lynch N, Segala R and Vaandrager F** (2006) The Theory of Timed I/O Automata (*Synthesis Lectures on Computer Science*). Morgan Claypool Publishers.
9. **Lee EA (2006)** Concurrent semantics without the notions of state or state transitions. In International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS), Paris, France, E. Asarin and P. Bouyer (Eds.), September 25–27, LNCS, vol. 4202: Springer-Verlag. doi: 10.1007/11867340_2.
10. **Liu X, Matsikoudis E and Lee EA** (2006) Modeling timed concurrent systems. In *CONCUR 2006 - Concurrency Theory*, Bonn, Germany, August 27–30, LNCS, vol. **4137**. Springer, pp. 1–15. doi: 10.1007/11817949_1.
11. **Nain S and Vardi MY** (2007) Branching vs. linear time: Semantical perspective. In *ATVA*.
12. **André C, Mallet F and de Simone R** (2007) Modeling Time(s). In *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS/UML)*, Nashville, TN, United States, pp. 559–573. doi: 10.1007/978-3-540-75209-7_38.
13. **Liu X and Lee EA** (2008). CPO semantics of timed interactive actor networks. *Theoretical Computer Science*, **409**, 110–125. doi: 10.1016/j.tcs.2008.08.044.
14. **Lee EA** (2009) Computing needs time. *Communications of the ACM*, **52**, 70–79. doi: 10.1145/1506409.1506426.
15. **Furia CA, Mandrioli D, Morzenti A and Rossi M** (2010) Modeling time in computing: A taxonomy and a comparative survey. *Computing Surveys*, **42**, 6:1–6:59.
16. **Benveniste A, Caillaud B and Pouzet M** (2010) The fundamentals of hybrid systems modelers. In IEEE Conference on Decision and Control (CDC), Atlanta, GA, USA, Dec. 15–17. doi: 10.1109/CDC.2010.5717614.
17. **Matsikoudis E and Lee EA** (2013) On fixed points of strictly causal functions. In International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS), Buenos Aires, Argentina, LNCS, vol. 8053. Springer-Verlag, pp. 183–197. doi: 10.1007/978-3-642-40229-6_13.
18. **Boulanger F, Jacquet C, Hardebolle C and Prodan I** (2014) TESL: A language for reconciling heterogeneous execution traces. In ACM/IEEE Conference on Formal Methods and Models for Codesign (MEMOCODE), Lausanne, Switzerland. doi: 10.1109/MEMCOD.2014.6961849.
19. **Cremona F, Lohstroh M, Broman D, Lee EA, Masin M and Tripakis S** (2017) Hybrid co-simulation: It's about time. *Software and Systems Modeling*, **18**, 1655–1679. doi: 10.1007/s10270-017-0633-6.
20. **Lee EA** (2018) Models of timed systems. In *FORMATS*, Beijing, China, September 4-6, **LNCS**, vol. **11022**. Springer, pp. 17–33.
21. **Ernst R, Kuntz S, Quinton S and Simons M** (2018) The logical execution time paradigm: New perspectives for multicore systems (Dagstuhl Seminar 18092). *Dagstuhl Reports*, **8**, 122–149. doi: 10.4230/DagRep.8.2.122.
22. **Gemlau K, Köhler L, Ernst R and Quinton S** (2021) System-level Logical execution time: augmenting the logical execution time paradigm for distributed real-time automotive software. *ACM Transactions on Cyber-Physical Systems*, **5**, 1–27. doi: 10.1145/3381847.
23. **Lee EA** (2021) Determinism. *ACM Transactions on Embedded Computing Systems (TECS)*, **20**, 1–34. doi: 10.1145/3453652.
24. **Lee EA, Bateni S, Lin S, Lohstroh M and Menard C** (2021) Quantifying and Generalizing the CAP Theorem, arXiv:2109.07771 [cs.DC], September 16. [Online]. Available: https://arxiv.org/abs/2109.07771.

On time in concurrent and distributed systems

25. **Lamport L** (1978) Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, **21**, 558–565. doi: 10.1145/359545.359563.
26. **Liskov B** (1993). Practical uses of synchronized clocks in distributed systems. *Distributed Computing*, **6**, 211–219. doi: 10.1007/BF02242709.
27. **Jantsch A** (2003) *Modeling Embedded Systems and SoCs - Concurrency and Time in Models of Computation*. Morgan Kaufmann.
28. **Zhao Y, Lee EA and Liu J** (2007) A programming model for time-synchronized distributed real-time systems. In Real-Time and Embedded Technology and Applications Symposium (RTAS), Bellevue, WA, USA, April 3–6. IEEE, pp. 259–268. doi: 10.1109/RTAS.2007.5
29. **Meseguer J and Ölveczky PC** (2010) Formalization and correctness of the PALS architectural pattern for distributed real-time systems. In *Forrmal Methods and Software Engineering*, LNCS, vol. 6447. Springer, pp. 303–320.
30. **Corbett JC *et al.*** (2012) Spanner: Google's globally-distributed database. In *OSDI*. doi: 10.1145/2491245.
31. **Brewer E** (2017) Spanner, TrueTime & the CAP Theorem, Google, February 14. [Online]. Available: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45855.pdf
32. **Lee EA and Lohstroh M** (2021) Time for all programs, not just real-time programs. In International Symposium on Leveraging Applications of Formal Methods (ISoLA). doi: 10.1007/978-3-030-89159-6_15.

On programming languages with time

33. **Martin T**, Real-Time Programing Language PEARL - Concept and Characteristics. In Computer Software and Applications Conference (COMPSAC), Chicago, 1978, pp. 301–306.
34. **MokAK** (1987) Annotating ADA for real-time program synthesis. In IEEE *Conference on Computer Assurance (COMPASS)*, IEEE.
35. **Henzinger TA, Horowitz B and Kirsch CM** (2003) Giotto: A time-triggered language for embedded programming. Proceedings of IEEE, 91, 84–99. doi: 10.1109/JPROC.2002.805825.
36. **Andalam S, Roop PS and Girault A** (2010) Predictable multithreading of embedded applications using PRET-C. In *Formal Methods and Models for Codesign (MEMOCODE)*, Grenoble, France, June 4. IEEE/ACM, pp. 159–168. doi: 10.1109/MEMCOD.2010.5558636.
37. **Elmqvist H, Otter M and Mattsson SE** (2012) Fundamentals of Synchronous Control in Modelica. In *International Modelica Conference*, Munich Germany, September 3–5. doi: 10.3384/ecp1207615.
38. **Natarajan S and Broman D** (2018) Timed C: An extension to the C Programming language for real-time systems. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Porto, Portugal, April 11–13, pp. 227–239. doi: 10.1109/RTAS.2018.00031.
39. **Lohstroh M, Menard C, Bateni S and Lee EA** (2021) Toward a Lingua Franca for deterministic concurrent systems. *ACM Transactions on Embedded Computing Systems (TECS)*, **20**, Article 36. doi: 10.1145/3448128.

On formal analysis of timed systems

40. **Manna Z and Pnueli A** (1993) Verifying hybrid systems. *Hybrid Systems*, LNCS, vol. **736**, pp. 4–35.
41. **Clarke EM and Emerson EA** (1981). Design and Synthesis of Synchronization Skeletons using Branching-Time Temporal Logic. In *Workshop on Logic of Programs*, LNCS, vol. **131**. Springer-Verlag.
42. **Matsikoudis E, Stergiou C and Lee EA** (2013) On the schedulability of real-time discrete-event systems. In International Conference on Embedded Software (EMSOFT), Montreal, Canada, September 29–October 4. ACM. doi: 10.1109/EMSOFT.2013.6658590.
43. **Sirjani M, Lee EA and Khamespanah E** (2020) Verification of cyberphysical systems. *Mathematics*, **8**. doi: 10.3390/math8071068.

On the physics of time

44. **Galison P** (2003) *Einstein's Clocks, Poincaré's Maps*. New York: W. W. Norton & Company.
45. **Muller RA** (2016) Now – The Physics of Time. W. W. Norton and Company.
46. **Rovelli C** (2018) *The Order of Time*. New York: Riverhead Books.

https://doi.org/10.1017/cbp.2023.1 Published online by Cambridge University Press

On microarchitectures and time

47. **Kopetz H and Bauer G** (2003) The time-triggered architecture. Proceedings of the IEEE, 91, 112–126.
48. **Wilhelm R** (2003) Run-time guarantees for real-time systems. In *Formal Modeling and Analysis of Timed Systems (FORMATS)*, LNCS, vol. **2791**. Springer, pp. 166–167.
49. **Thiele L and Wilhelm R** (2004) Design for timing predictability. *Real-Time Systems*, **28**, 157–177. doi: 10.1023/B:TIME.0000045316.66276.6e.
50. **Edwards SA and Lee EA** (2007) The Case for the Precision Timed (PRET) Machine. In *Design Automation Conference (DAC)*, San Diego, CA, June 4–8.
51. **Wilhelm R *et al.*** (2008) The worst-case execution-time problem - overview of methods and survey of tools, *ACM Transactions on Embedded Computing Systems (TECS)*, **7**, 1–53.
52. **Wilhelm R, Grund D, Reineke J, Schlickling M, Pister M and Ferdinand C** (2009) Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems. *IEEE Transactions on Computer Aided Design*, **28**, 966–978. doi: 10.1109/TCAD.2009.2013287.
53. **Schoeberl M** (2009) Time-predictable computer architecture. *EURASIP Journal on Embedded Systems*, Article ID **758480**, 17 pages. doi: 10.1155/2009/758480.
54. **Lee EA, Reineke J and Zimmer M** (2017) Abstract PRET machines. In *IEEE Real-Time Systems Symposium (RTSS)*, Paris, France.

https://doi.org/10.1017/cbp.2023.1 Published online by Cambridge University Press